VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

METODY ANALÝZY A DETEKCE RANSOMWARU

METHODS OF RANSOMWARE ANALYSIS AND DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **SAMUEL VOJTÁŠ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. LUKÁŠ ZOBAL**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Vojtáš Samuel**
Program: Informační technologie
Název: **Metody analýzy a detekce ransomwaru**
Methods of Ransomware Analysis and Detection
Kategorie: Bezpečnost

Zadání:

1. Studujte problematiku škodlivého kódu (malware) a jeho jednotlivých typů. Speciálně se zaměřte na typ ransomware. Dále se seznamte s metodami statické a dynamické analýzy binárního spustitelného kódu jako jsou reverzní inženýrství, sandboxing či dekompilace.
2. Po domluvě s konzultantem ze společnosti Avast detailně analyzujte vybrané vzorky ransomware kmenů pomocí zmíněných metod. Detailně zdokumentujte tento fenomén posledních let z pohledu jeho vývoje v čase, používaných technik, technologií a současných trendů. Pomocí technického vybavení společnosti Avast se rovněž zaměřte na vyhledávání kmenů nových.
3. Díky získaným poznatkům navrhněte metody pro efektivní obranu proti této hrozbě. Může jít o detekční pravidla v některém ze standardních jazyků (YARA, Suricata atd.), kryptoanalytické nástroje atd.
4. Po konzultaci s vedoucím realizujte navržené metody z bodu 3 v praxi a otestujte je na sadě alespoň tisíce prevalentních vzorků moderního ransomwaru. Klaďte důraz na využití výstupů práce v praxi.
5. Svoji práci zhodnoťte a uveďte výhled na další možný vývoj.

Literatura:

- SIKORSKI, Michael a Andrew HONIG. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press, 2012. ISBN 978-1593272906.
- SIDOR, Samuel. Analýza a detekce malwaru typu RAT. Brno, 2019, 47 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.
- SZOR, Peter. The Art of Computer Virus Research and Defense. Addison-Wesley Professional, 2005. ISBN 978-0321304544.
- EILAM, Eldad. Reversing: Secrets of Reverse Engineering. Wiley, 2005. ISBN 978-0764574818.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů a rozpracování bodu čtvrtého.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zobal Lukáš, Ing.**
Konzultant: Zezula Ladislav, Mgr., Avast
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 13. října 2021

Abstrakt

Cieľom tejto práce je poukázať na hrozbu škodlivého kódu a popísať jeho jednotlivé typy. Špeciálne sa zameriava na ransomvér – jeho historický vývoj, spôsob analýzy, detekciu a zotavenie z neho. Predstavené sú aj rôzne techniky reverzného inžinierstva a pojmy s ním súvisiace ako statická a dynamická analýza alebo sandboxing. Taktiež sa pojednáva tvorba detekčných mechanizmov a klasifikácia škodlivého kódu. Firma Avast poskytla vzorky niekoľkých rodín moderného ransomvéru pre analýzu s cieľom vytvorenia detekčných YARA pravidiel a popísania chovania vzoriek. Text ukazuje proces vývoja detekčných mechanizmov na hrozbu ransomvéru a spôsob dešifrovania súborov pri rodinách ransomvéru, ktoré obsahovali chyby v kryptografii. Na konci práce sú zhrnuté výsledné dáta hovoriace o efektivite implementovaných obranných mechanizmov.

Abstract

The purpose of this thesis is to demonstrate the threat of malware and to describe its forms. Special focus is put on ransomware – its historical evolution, method of analysis, detection, and recovery from it. Various techniques of reverse engineering are also introduced alongside concepts related to it, such as static and dynamic analysis or sandboxing. Paper centers around creating detection mechanisms and malware classification. Company Avast provided samples of several ransomware families for the analysis to create detection YARA rules and to describe samples' behavior. The process of development of detection mechanisms for ransomware threats is shown alongside the method to decrypt files encrypted by various ransomware families that contained cryptography errors. The end of the thesis sums up the resulting data regarding the efficiency of defense mechanisms.

Kľúčové slová

Malvér, Reverzné inžinierstvo, Ransomvér, Kryptografia, YARA

Keywords

Malware, Reverse engineering, Ransomware, Cryptography, YARA

Citácia

VOJTÁŠ, Samuel. *Metody analýzy a detekce ransomwaru.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Lukáš Zobal

Metody analýzy a detekce ransomwaru

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Lukáša Zobala. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje z ktorých som čerpal.

.....
Samuel Vojtáš
9. mája 2022

Poďakovanie

Ďakujem svojemu vedúcemu bakalárskej práce Ing. Lukášovi Zobalovi za pripomienky a ochotu pomáhať pri písaní práce. Ďalej by som chcel poďakovať aj Ing. Jakubovi Křoustkovi, Ph.D. a Mgr. Ladislavovi Zezulovi za odbornú pomoc, konzultácie a podporu.

Obsah

Kapitola 1

Úvod

V súčasnosti sa informačné technológie nachádzajú všade okolo nás. Používajú ich ľudia rôzneho veku, z rôznych oblastí a s rôznymi znalosťami o nich. Aj keď tieto technológie môžu pôsobiť bezpečne, fakt, že im ľudstvo poskytuje takmer plnú dôveru a využíva ich v najrôznejších odvetviach ich robí ľahko zneužiteľnými. Človeku v modernej dobe slúži počítač alebo mobil na veľké množstvo každodenných aktivít bez toho, aby si uvedomoval, že sa môže stať obeťou hackerských útokov, škodlivého kódu alebo podvodu práve cez tieto zariadenia.

Programy s cieľom páchať škodlivú aktivitu vznikali už v počiatkoch informačnej éry a odvtedy sa s každoročným pokrokom v oblasti informatiky tento problém stáva stále viac a viac komplikovanejším. Aké konkrétne hrozby môžu postihnúť človeka pri práci s mobilom alebo počítačom? Čo má robiť užívateľ a čo majú robiť väčšie spoločnosti, aby sa minimalizovalo riziko spojené s informačnými technológiami?

Táto bakalárska práca sa zaoberá hrozbami škodlivého kódu, ako sa šíri a ako sa pred ním chrániť. Špeciálne zameranie je na ransomvér – čo to vlastne je a ako sa historicky formoval. Za posledné roky bol problém ransomvéru spozorovaný nie len na počítačoch, ale aj v doprave, zdravotníctve, bankovníctve a množstve iných odborov – hlavne kvôli prítomnosti veľkého objemu dát. Hrozba tohto typu programu v dôsledku informatizácie rôznych odvetví priemyslu stále stúpa.

Cieľ práce sa bude sústrediť na detekciu ransomvéru a návrh kryptoanalytických nástrojov, ktoré môžu zvrátiť jeho nežiadúce účinky. Detekcia bude implementovaná pomocou pravidiel nástroja YARA dovoľujúce detekovať a klasifikovať program podľa rôznych statických a behaviorálnych znakov.

V kapitole 2 bude predstavené reverzné inžinierstvo a spôsob ako zistiť chovanie programu z binárneho súboru. Rozoberá hlavné prístupy analýzy škodlivého kódu a možnosť ich aplikácie.

Kapitola 3 hovorí o typoch škodlivého kódu s detailnejším pohľadom na ransomvér. Obsahuje popisy chovania jednotlivých druhov, ich delenie a spôsob ich detekcie pomocou YARA pravidiel.

Kapitola 4 pojednáva o samotnej analýze jednotlivých vzoriek moderného ransomvéru poskytnutých firmou Avast a návrhom ich detekčných mechanizmov alebo kryptoanalytických nástrojov. Opisuje proces analýzy malvéru a prípady použitia nástrojov pre jej zjednodušenie. Návrh kryptoanalytických nástrojov v podobe dekryptorov závisí od konkrétnej vzorky a jej spôsobu šifrovania súborov. Podľa toho bude rozhodnuté, či dekryptor je možné vytvoriť alebo nie.

V kapitole 5 sú popísané experimentálne výsledky – počet prichádzajúcich vzoriek do firmy Avast a počet vzoriek detekovaných implementovanými YARA pravidlami spolu s popisom vytvorených dekryptorov.

Kapitola 6 obsahuje zhodnotenie práce a výhľad do budúcnosti pre boj s ransomvérom.

Kapitola 2

Reverzné inžinierstvo

Reverzné inžinierstvo je proces získavania vedomostí z čohokoľvek, čo bolo vytvorené človekom. V minulosti išlo hlavne o hmotné produkty, ktoré boli postupne rozobraté, aby sa zistilo ako fungujú vo svojej podstate. Tieto informácie boli potom väčšinou použité na zlepšenie ďalších výrobkov. Tradične išlo o domáce spotrebiče ako televízor, alebo rádio. Avšak s rýchlym vývojom informačných technológií sa reverzní inžnieri rýchlo zamerali na modernejšie technológie ako napr. softvér alebo hardvér. Táto práca sa sústreduje na reverzné inžinierstvo softvéru. Softvérové reverzné inžinierstvo spočíva v analýze programu a preskúmaní, čo daný softvér robí. Táto disciplína vyžaduje poznatky rôznych odborov informatiky, no hlavnou prerekvizitou stále ostáva zvedavosť [7].

S vývojom počítačov sa začal objavovať aj druh softvéru, ktorý je vytvorený so zlým úmyslom vykonávať aktivitu, ktorú si užívateľ nevyžiadal. Takýto druh programov sa jedným slovom nazýva malvér (od slova *malicious software* – škodlivý softvér). Môže sa vyskytovať v mnoho rôznych podobách, napr. spustiteľný súbor alebo skript. Nie vždy je možné na prvý pohľad odhaliť úmysel malvéru, a preto v tomto prípade prichádza reverzné inžinierstvo ako nástroj na zistenie chovania škodlivého kódu [9].

Reverzné inžinierstvo malvéru rozdeľujeme na dve základné kategórie, a to statická a dynamická analýza. Statická analýza program skúma z rôznych pohľadov bez jeho spustenia. Zatiaľ čo, dynamická analýza skúma program za behu s cieľom zistiť, čo vykonávaný program robí [12].

2.1 Statická analýza

Statická analýza malvéru sa snaží využiť všetky možné prostriedky, ktoré zo škodlivého kódu môžu byť zistené bez jeho spustenia. Ide o vyhľadávanie informácií o malvéri v antivírusových databázach – aj neskúsený užívateľ môže za pomoci antivírusových programov väčšinou zistiť, či daný program je alebo nie je škodlivý – alebo analýza samotného škodlivého kódu. Ak je malvér vo forme binárneho súboru, už len jeho hlavička dokáže malvérovému analytikovi dodať veľké množstvo informácií. Pokročilá statická analýza však už zahŕňa komplikovanejšie techniky – hlavné úsilie sa sústredí na porozumenie inštrukcií programu pomocou techník ako disassembly alebo dekompilácia. Poznatky potrebné pre tento druh analýzy zahŕňajú veľmi dobrú znalosť assembly inštrukcií, rôznych konštruktov kódu typických pre malvér a konceptov v operačných systémoch, na ktorý je malvér navrhnutý [12].

Identifikácia vzorky pomocou jej hešu

Identifikácia malvéru podľa jeho mena je neefektívna, pretože sa môže stať, že rovnakým vzorkám sú priradené rozdielne mená. Preto sa pre vyhľadávanie vzoriek vo vírových databázach používa hash (ten je pre danú vzorku vždy rovnaký). Takisto ak by došlo k zmene malvéru, hash slúži ako zaistenie integrity nad danou vzorkou – hash modifikovanej vzorky by nebol rovnaký s originálnym hashom. Medzi štandardne používané hashovacie algoritmy patrí MD5[1], SHA1[2] alebo SHA256[3] [9].

Určenie typu súboru vzorky

Jedným z prvých krokov statickej analýzy je obvykle identifikácia typu formátu súboru. Vzorky malvéru, často dostupné vo forme binárneho súboru, prichádzajú v rôznych formátoch. Pre operačný systém Linux sa jedná hlavne o formát typu *Executable and Linkable Format* (ELF) a pre Microsoft Windows hlavne o formát *Portable Executable (PE)*. Môže ísť taktiež o skript alebo iný druh softvéru. Súborový typ sa dá zistiť pomocou príkazu `file` na operačnom systéme Linux. Veľká obľúbenosť operačného systému Windows zapríčinila, že väčšina vzoriek malvéru sú navrhnuté priamo naň. Preto pri analýze zohráva veľkú rolu znalosť formátu PE. Takéto súbory sa dajú spoznať podľa špecifickej hlavičky a vo väčšine prípadov majú prípony napr. `.exe` alebo `.dll` [7].

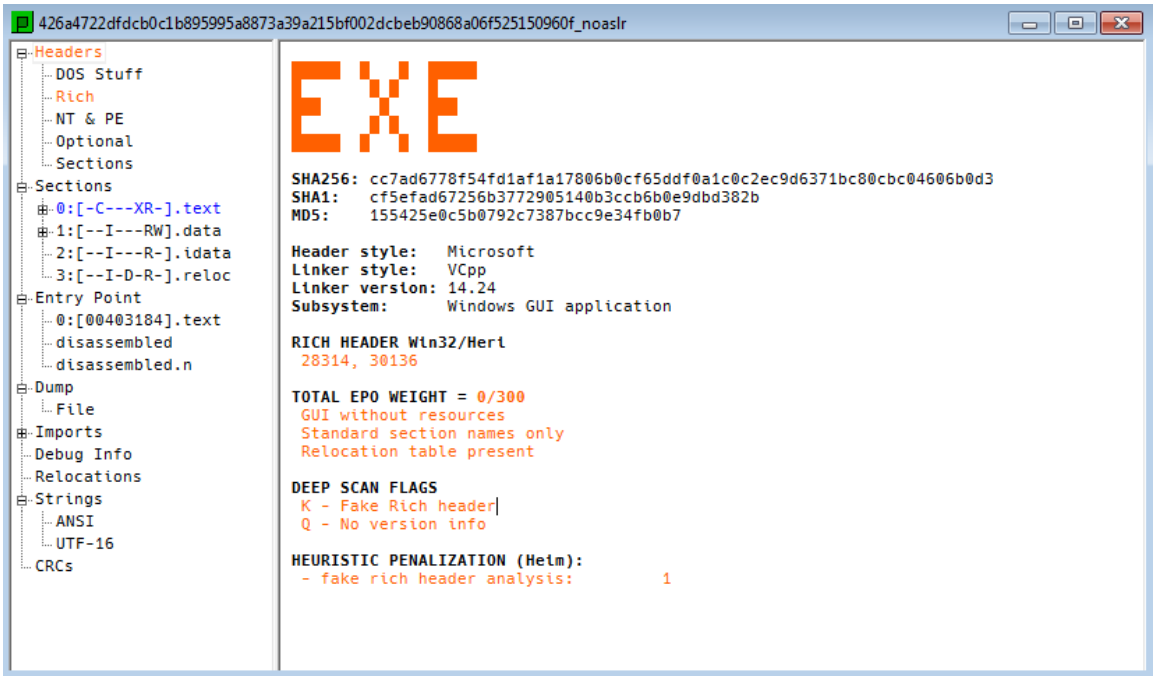
Hlavička PE súboru obsahuje informácie o tom ako sú dáta členené do sekcií. S pomocou programov ako PEView[4], RetDec[5] alebo PEscope[6] sa na základe informácií z hlavičky PE súboru dajú zistiť mená sekcií a ich umiestnenie v rámci programu [12]. Je možné sa aj do sekcií pozrieť a zistiť, čo sa v nich nachádza [7].

Extrakcia informácií z binárneho súboru

Ak už je známy formát súboru a metadáta o ňom, jedným z ďalších krokov býva extrakcia reťazcových literálov z binárneho súboru. Program príkazového riadku `strings` dokáže hľadať sekvencie alfanumerických znakov v rámci súboru. V niektorých častiach malvéru býva nutné využiť reťazce, a kvôli tomu sa príkazu `strings` často darí extrahovať použité IP adresy, doménové mená, command-line príkazy, ktoré majú byť spustené alebo modifikované registre [9].

Binárny súbor môže tiež obsahovať ďalšie informácie ukryté v rámci svojich sekcií a neskôr pri behu programu ich extrahovať a použiť. Malvérový analytik ich dokáže v rámci statickej analýzy získať vďaka nástrojom ako Resource Hacker[7]. Ten dovoluje zdroj z binárneho súboru vybrať a uložiť do samostatného súboru – môže ísť zase napríklad o reťazce znakov, iný binárny súbor alebo šifrovací kľúč. Malvér obvykle pracuje s funkciami z rôznych knižníc. Nástroj pre Windows – Dependency Walker[8] – je schopný vytvoriť výpis všetkých importovaných (funkcie, ktoré chce program používať) a exportovaných (funkcie, ktoré program umožňuje používať ostatným programom) funkcií. Importy a exporty dokážu o softvéri povedať veľa, obzvlášť ak sa jedná o malvér, ktorý zväčša používa funkcie

[1] https://www.ietf.org/rfc/rfc1321.txt
[2] https://tools.ietf.org/html/rfc3174
[3] https://datatracker.ietf.org/doc/html/rfc4634
[4] PEView dostupný z https://www.aldeid.com/wiki/PEView
[5] RetDec dostupný z https://retdec.com/
[6] PEscope dostupný z https://github.com/T1m3M/PEscope
[7] Resource Hacker dostupný z http://www.angusj.com/resourcehacker/
[8] Dependency Walker dostupný z https://www.dependencywalker.com/



Obr. 2.1: Náhľad na štruktúru súboru formátu PE pomocou nástroja PEscope

operačného systému (napr. funkcie pre vytvorenie, prepísanie a premenovanie súboru môžu naznačovať, že ide o ransomvér – ransomvér bude popísaný v ďalších sekciach) [12].

Deobfuskácia a unpacking

Malvér často prichádza v obfuskovanej alebo v packovanej verzii. Autori škodlivého kódu používajú na sťaženie analýzy svojich programov obfuskačné mechanizmy, packery alebo cryptory. Packery sú programy, ktoré na vstupe majú pôvodný malvér a na výstupe produkujú pomocou kompresie nový program, na ktorý je ťažšie použiť metódy reverzného inžinierstva. Cryptory fungujú podobným spôsobom, ale namiesto kompresie používajú šifrovanie. Pre automatické unpackovanie je potrebné zistiť packer, ktorý bol na vzorku použitý. V tomto pomáhajú nástroje ako PEiD[9] alebo DIE (Detect It Easy)[10] [9].

Medzi najpoužívanejšie packery patrí napr. UPX[11], PECompact[12] alebo ASPack[13] [12]. Na frekventovane používané packery boli vytvorené automatické unpackery, ktoré dokážu vzorku malvéru dostať do pôvodnej podoby. V iných prípadoch musí malvérový analytik proces packovania zvrátiť manuálne sám.

Disassembly a dekompilácia

Pri pokročilej statickej analýze sa dostáva na radu disassembler. Je to nevyhnutný nástroj pre každého malvérového analytika, pretože poskytuje vyššiu abstrakciu nad inštrukciami binárneho súboru. Disassembler transformuje byty strojového kódu na jazyk symbolických

[9] PEiD dostupný z https://www.aldeid.com/wiki/PEiD
[10] DIE dostupný z https://github.com/horsicq/Detect-It-Easy
[11] UPX dostupný z https://upx.github.io/
[12] PECompact dostupný z https://bitsum.com/portfolio/pecompact/
[13] ASPack dostupný z http://www.aspack.com/

inštrukcií (assembly language). Tie sú potom vhodne vizualizované tak, aby sa dali ľahšie pochopiť. Napríklad inštrukcie, ktoré spolu súvisia sú spájané do blokov alebo skoky, ktoré sa môžu udiať v rámci prevádzania programu sú znázornené ako šípky medzi jednotlivými blokmi. Výsledný jazyk symbolických inštrukcií sa skladá, na rozdiel od surových bytov, zo zapamätateľných názvov operačných kódov. Mená operandov inštrukcií sa dajú meniť, aby sa zjednodušilo pochopenie programu. Niektoré časti kódu nemusí ani disassembler správne vyhodnotiť, no analytik si jednoducho môže dáta prispôsobiť podľa seba, pomenovať funkcie programu, ktoré zanalyzoval alebo definovať štruktúry, ktoré sa v programe používajú [5].

Aj keď je disassembler skvelý nástroj pre pochopenie kódu z binárneho súboru, pri väčších aplikáciách sa analýza komplikuje. Na prevod inštrukcií binárneho súboru do ešte abstraktnejšej verzie sa používa dekompilátor, ktorý na výstupe ukazuje vyšší programovací jazyk. Typ vyššieho programovacieho jazyka závisí od dekompilátora a môže to byť napr. C, Java, Python alebo iný. Užívateľ v dekompilátore môže nastavovať dátové typy premenným a meniť signatúry funkcií tak, ako v iných programovacích jazykoch vyššej úrovne [12].

Hoci sa môže zdať, že dekompilátor je ideálny na pochopenie malvéru, no proces prevodu strojového kódu na vysoko-úrovňový jazyk je zložitý a často nepresný. Malvérový analytik sa preto často musí pri nesprávnej dekompilácií pozrieť do disassembleru pri komplikovaných častiach programu [5]. Rozdiel medzi disassemblerom a dekompilátorom sa dá vidieť na obrázkoch 2.2 a 2.3.

Existuje mnoho rôznych disassemblerov a dekompilátorov, no tieto dve varianty sú najfrekventovanejšie.

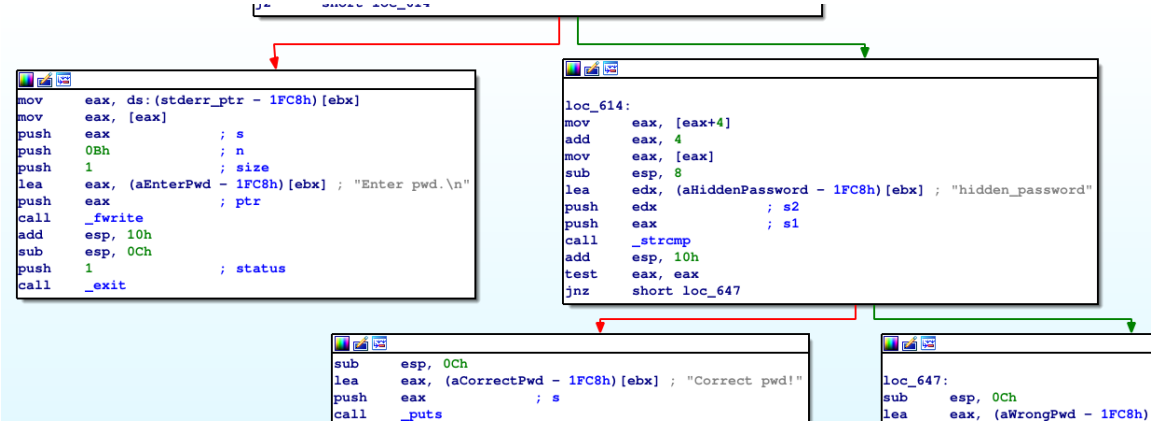
- IDA Pro & Hex-Rays Decompiler[14] – Komerčný disassembler a dekompilátor pre množstvo formátov spustiteľných súborov a operačných systémov od rovnakej firmy Hex Rays. IDA Pro umožňuje interaktívne meniť disassemblovaný výstup v prípade, že pri spracovaní súboru nevyhodnotí niektoré byty správne. Pre disassembler IDA Pro existuje aj bezplatná verzia, no nemá všetky funkcie. Firma Hex Rays si za roky vybudovala okolo svojich produktov veľkú komunitu, čo dokazuje aj množstvo pluginov, ktoré sa do IDA Pro dajú pridať [5].

- Ghidra[15] – Open-source nástroj, ktorý taktiež zahŕňa disassembler aj dekompilátor. Vyšiel ako projekt NSA (National Security Agency) v roku 2019, a preto nestihol získať toľko užívateľov. Ghidra má oproti IDA Pro veľkú výhodu, a to, že ide o open-source licenciu, teda program môžu užívatelia získať bez ďalších poplatkov [6].

2.2 Dynamická analýza

Dynamická analýza skúma malvér jeho spustením a pozorovaním, čo malvér robí so systémom, na ktorom beží. Takýto spustený škodlivý kód generuje stopy na operačnom systéme, podľa ktorých sa dá zistiť jeho správanie, napríklad vytváranie súborov, generovanie sieťovej komunikácie alebo zapisovanie do registrov MS Windows. Pri pokročilej dynamickej analýze sa často pristupuje k riadenému vykonávaniu vzorky malvéru, kedy analytik sleduje takmer každú vykonávanú inštrukciu. Informácie o vykonávaní malvéru nie je možné získať pomocou statickej analýzy [12].

[14] IDA Pro dostupná z https://hex-rays.com/ida-pro/
[15] Ghidra dostupná z https://ghidra-sre.org/



Obr. 2.2: Výstup disassembleru

```
void __cdecl main(int argc, const char **argv, const char **envp)
{
  if ( argc != 2 )
  {
    fwrite("Enter pwd.\n", 1u, 0xBu, stderr);
    exit(1);
  }
  if ( !strcmp(argv[1], "hidden_password") )
    puts("Correct pwd!");
  else
    puts("Wrong pwd!");
}
```

Obr. 2.3: Výstup dekompilátoru

Spustenie malvéru v bezpečnom prostredí

Neriadené prenechanie kontroly malvéru nad počítačom môže viesť k aktivácií škodlivého kódu a napadnutie tohto systému, prípadne celej siete. Preto je nevyhnutné si vytvoriť bezpečné prostredie, ktoré malvéru zabráni v aktivácii na pracovnej stanici. Na tento účel sa používajú virtuálne stroje. Ak škodlivý kód zničí systém, virtuálny stroj sa dá jednoducho nainštalovať odznova, alebo obnoviť zo snapshotu (uložený predchádzajúci stav celého operačného systému). Virtuálne stroje uľahčujú celý priebeh analýzy, no je treba dávať pozor aj na fakt, že oddelenie od hostiteľského operačného systému nemusí byť úplné, teda virtuálny stroj a hostiteľský systém môžu zdieľať určité prostriedky, napríklad sieť alebo súbory. Pri nesprávnom nastavení môže malvér poškodiť zdieľané súbory alebo generovať sieťovú prevádzku, ktorá môže poškodiť danú sieť. Niektoré vzorky malvéru overujú prítomnosť testovacieho prostredia a môžu sa chovať inak. Programovo sa s pomocou niektorých nastavení v systéme dá zistiť, že sa jedná o virtuálny stroj. Ak malvér zistí, že je spúšťaný na virtualizovanom operačnom systéme, môže byť naprogramovaný tak, aby skončil predčasne čo znemožní jeho dynamickú analýzu [9].

Protože väčšina malvéru je vytvorená pre operačný systém Microsoft Windows, vzorky prichádzajú vo forme PE súboru (viď statická analýza malvéru) ako priamo spustiteľný súbor (väčšinou s príponou `.exe`) alebo dynamická knižnica (väčšinou s príponou `.dll`). V prípade priamo spustiteľného súboru je spustenie jednoduché, keďže je to primárny účel tohto typu súboru, no dynamické knižnice nie sú samostatne schopné spustenia. Dynamické knižnice sú len prilinkované do iných programov, avšak väčšina debuggerov je aj

napriek tomu schopná ich spustiť pomocou jednoduchého súboru, ktorý si knižnicu prilinkuje. Týmto spôsobom je teda možný aj debugging DLL [12].

Sandboxing

Sandbox je virtualizované izolované prostredie používané v kyberbezpečnosti na zisťovanie či program je škodlivý alebo nie. Sandbox dozerá nad aktivitou, ktorú vzorka softvéru vykonáva, či už ide o sieťovú komunikáciu, modifikáciu súborového systému alebo zápis do registrov MS Windows. Kedže pri sandboxingu sa využíva izolované prostredie – ktoré okrem toho ešte aj zvykne bežať na vzdialenom serveri – riziko, že malvér poškodí počítač analytika prudko klesá [12].

Monitoring systémových a sieťových zdrojov

Pri dynamickej analýze môže malvér vykonávať veľké množstvo zmien v operačnom systéme, ktoré analytik ľahko prehliadne. Pre zistenie chovania malvéru pri dynamickej analýze je potrebné monitorovať aktivitu operačného systému a dostať všetky záznamy z monitoringu na jedno miesto. To umožňuje program Process Monitor[16], ktorý zbiera takéto informácie a dovoľuje ich filtrovať podľa viacerých kritérií ako napríklad meno procesu, typ aktivity alebo PID. Nástroj Process Explorer[17] poskytuje informácie o spustených procesoch na systéme užívateľa, ktoré môžu byť použité v rámci filtrov v Process Monitore. Na rozdiel od vstavaného programu pre Windows – Task Manager – dokáže detailnejšie sledovať procesy, vlákna a k nim aj otvorené súbory alebo nalinkované dynamické knižnice. Odchyt sieťovej prevádzky sa dá zabezpečiť klasickými nástrojmi pre monitoring siete – Wireshark[18] alebo tcpdump[19]. Tie poskytujú prehľad sieťovej prevádzky cez veľké množstvo podporovaných protokolov a malvérový analytik s nimi ľahko dokáže zistiť, čo sa na sieti deje. Na čítanie a zapisovanie do otvorených sieťových spojení slúži nástroj Netcat[20]. Napriek jeho jednoduchej funkcionalite sa tento nástroj dá použiť v množstve rôznych situácií pri monitorovaní alebo simulovaní sieťovej prevádzky. Ako bolo vysvetlené v predchádzajúcej sekcii, zdieľanie siete na virtuálnom stroji s hostiteľským systémom môže viesť k bezpečnostným rizikám. Uplatňuje sa preto radšej prístup, kedy server služby, na ktorý sa má malvér pripojiť je simulovaný vo vnútornej sieti, ktorú používa len virtuálny stroj (alebo virtuálne stroje). Pre simuláciu serverov rôznych služieb je možné využiť nástroj INetSim[21] podporujúci služby ako napríklad HTTP/HTTPS, FTP alebo SMTP. Ak je potrebné simulovať DNS server, môže byť zvolený aj ApateDNS[22] – dokáže mimikovať DNS server bežiaci na lokálnom počítači [12].

Debugging

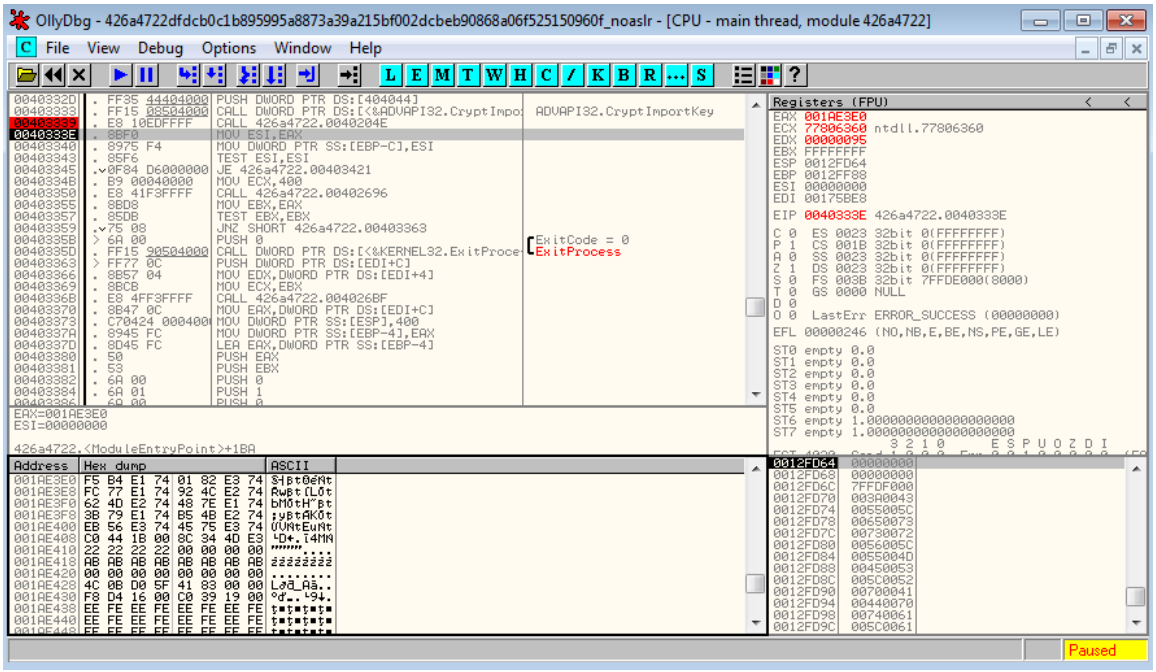
V pokročilej analýze sa často uplatňuje technika debuggingu. Používa sa ňu špeciálny program debugger, ktorého primárny účel je hľadanie chýb programu (z anglického *bug*). V rámci dynamickej analýzy sa dá použiť na veľmi podrobnú analýzu malvéru. Pri debuggovaní sa

[16] Process Monitor dostupný z https://docs.microsoft.com/en-us/sysinternals/downloads/procmon
[17] Process Explorer dostupný z https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer
[18] Wireshark dostupný z https://www.wireshark.org/
[19] tcpdump dostupný z https://www.tcpdump.org/
[20] Netcat dostupný z https://nmap.org/ncat/
[21] INetSim dostupný z https://www.inetsim.org/
[22] ApateDNS dostupný z https://www.aldeid.com/wiki/Mandiant-ApateDNS

dá program prechádzať inštrukcia za inštrukciou (tento proces sa nazýva krokovanie programu). Ak je však potrebné preskočiť časť programu, je možné nastaviť tzv. breakpoint – keď debugger narazí na breakpoint, prevádzanie programu je pozastavené pre jeho analýzu [9]. Spôsob nastavenia breakpointu môže byť rôzny, ako príklad sa používajú:

- Softvérové breakpointy – Na zvolenú adresu je vložené špeciálne prerušenie, podľa ktorého debugger vie, že má nastať zastavenie prevádzania programu.
- Hardvérové breakpointy – Používajú špeciálne registry, čo zapríčiňuje ich vyššiu rýchlosť, no limituje ich maximálny počet (kedže je obmedzený aj počet týchto špeciálnych registrov). Debugger zastaví prevádzanie programu pri prístupe do pamäti na zvolenej adrese.
- Podmienené breakpointy – Špeciálny typ softvérového breakpointu, kedy sa program zastaví ak je splnená nejaká podmienka. Tieto breakpointy sú najpomalšie, pretože pri každom prevádzaní inštrukcie sa musí podmienka kontrolovať.

Na obrázku 2.4 sa dá vidieť príklad debuggera (inštrukcie programu vľavo hore, stav registrov vpravo hore, pamäťový priestor procesu vľavo dole, stav zásobníku procesu vpravo dole).



Obr. 2.4: Ukážka debuggera OllyDbg

Debugger môže fungovať na dvoch úrovniach – na úrovni zdrojového kódu, kedy sa krokuje po riadkoch, alebo na úrovni assembly inštrukcií, kedy sa krokuje po jednotlivých inštrukciach. Debugging na úrovni zdrojového kódu sa používa pri vývoji aplikácií a iných systémov, pre analýzu malvéru sa používa debugging na úrovni assembly inštrukcií. Táto metóda sa volí kvôli nedostupnosti zdrojového kódu a neprítomnosti ladiacich informácií [12]. Je dôležité aj zvoliť debugger na základe módu, v ktorom malvér beží. Jedná sa o kernel-mode alebo user-mode. Nie každý debugger podporuje obidva módy [5].

- User-mode debugger – Ak malvér beží v user-mode – v ktorom bežia skoro všetky obyčajné aplikácie, ktoré si užívateľ stiahne do počítača – nie je potrebný špeciálny debugger. Väčšina je stavaná na user-mode programy. Debugger sa napojí na zvolený proces a malvérový analytik môže začať nastavovať breakpointy a analyzovať program. Nevýhodou však je, že user-mode debugger sa môže napojiť len na jeden proces súčasne [5]. Populárnymi user-mode debuggerami pri analýze malvéru sú OllyDbg[23] (pre 32-bitové programy) alebo x64dbg[24] (pre 64-bitové programy).

- Kernel-mode debugger – Slúži na získanie uceleného pohľadu o operačnom systéme. Tento typ debuggeru debugguje celý operačný systém, ale taktiež sa v niektorých prípadoch dá využiť na debugging jedného procesu ako user-mode debugger. Jadro OS a ovládače hardvérových zariadení bežia v kernel-mode, a preto jediná cesta na ich analýzu je práve kernel-mode debugging. Kernel-mode je vyhovujúci pre malvér typu rootkit, ktorý sa chce na systéme schovať a využíva na to práve kernel-mode. Medzi najčastejšie kernel-mode debuggere patrí WinDbg[25] priamo od firmy Microsoft, pretože kernel-mode debugging je úzko spojený s operačným systémom [12].

[23] OllyDbg dostupný z https://www.ollydbg.de/

[24] x64dbg dostupný z https://x64dbg.com/

[25] WinDbg dostupný z https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools

Kapitola 3

Problematika škodlivého kódu

Malvér je softvér, prevádzajúci nevyžiadané, nechcené, alebo škodlivé operácie. Môže existovať v rôznych formách – spustiteľný súbor, skript, zhluk inštrukcii uložený v jednom reťazci (shellcode) alebo v ľubovoľnom inom softvéri. Autori malvéru ho vytvárajú s cieľom znemožnenia prevádzania operácií na počítači, ukradnutia informácií od obete, neautorizovaný prístup k systému, posielanie spamu alebo z iného dôvodu. Ich dôvody pramenia častokrát z vidiny finančného zisku alebo potreby zviditeľniť sa. Reverzné inžinierstvo, ktorému bola venovaná predchádzajúca kapitola slúži ako hlavný nástroj pre návrh možných opatrení proti malvéru a jeho šíreniu [9].

3.1 Klasifikácia malvéru

Množstvo nových vzoriek malvéru pribúda každým dňom a autori škodlivého kódu ho používajú na najrôznejšie účely. Niekedy chcú infikovať len jeden systém na získanie dôverných informácií, no inokedy sa snažia o znefunkčnenie celej siete. Keďže účel malvéru nie je len jeden, postupom času sa vyformovalo niekoľko kategórií, ktorá každá vykonáva škodlivý kód iným spôsobom. Malvér nemusí striktne dodržiavať len jednu kategóriu a môže patriť do viacerých zároveň. Zoznam typov malvéru, ktoré budú uvedené nie je konečný, patrí sem mnoho ďalších typov, ktoré pre túto prácu nie je nutné uvádzať a taktiež malvér je tak dynamická téma, že stále pribúdajú nové kategórie [9, 13].

- Vírus – Program, ktorý je schopný svojej replikácie pripojením vlastných kópií do už existujúcich neškodlivých programov. Množenie vírusov napadá ďalšie a ďalšie programy. Vyžadujú interakciu užívateľa na spustenie.
- Červ – Ide o programy využívajúce počítačovú sieť, ich replikácia prebieha z jedného užívateľa siete na druhého bez jeho interakcie. Snažia sa zneužiť konkrétnu zraniteľnosť systému a teda nemusia napadnúť celú sieť, ale len systémy, ktoré túto zraniteľnosť majú.
- Trójsky kôň – Malvér, ktorý sa tvári ako bezpečný program, ale taktiež obsahuje aj zamaskovaný škodlivý kód. Na prvý pohľad je obtiažne spoznať, že ide o malvér, pretože jeho podstatnú časť tvorí reálny program, ktorý užívateľ naozaj chcel, no len zlomok z neho vykonáva nežiadúce činnosti.
- Backdoor ('Zadné dvierka') – Škodlivý program, ktorý na počítači obeti vytvára server, na ktorý sa útočník ako klient pripája. Backdoor sa používa na ovládanie systému,

na ktorý bol získaný prístup, napríklad pomocou exploitov. Útočník cez backdoor dokáže ovládať pomocou príkazov počítač obete.

- Remote Access Trojan (RAT) – Typ backdooru, ktorý dáva útočníkovi administratívnu kontrolu nad nadpadnutým počítačom.
- Keylogger – Keylogger nepozorovane beží na systéme a zaznamenáva informácie o stisknutých klávesiach. Snaží sa ukradnúť senzitívne dáta, ako užívateľské mená, heslá alebo PINy. Tieto informácie môžu byť potom využité pre krádež identity. Odhalenie tohto útoku býva veľmi ťažké a môže trvať niekoľko dní aj dlhšie, kým obeť zistí, že bola napadnutá.
- Exploit – Technika, ktorá využíva objavené zraniteľnosti v iných programoch. Zraniteľnosť v softvéri poskytuje spôsob, akým útočník môže infikovať systém obete.
- Ransomvér – Malvér, ktorý znemožní obeti používať časť svojho systém a jeho prostriedky, napríklad zašifrovaním osobných dát (fotky, práca, ...). Útočník potom za odblokovanie vyžaduje od obeti výkupné (ransom). Tento druh malvéru bude detailnejšie popísaný v sekcii 3.3.
- Rootkit – Malvér, ktorý umožňuje útočníkovi privilegovaný prístup k infikovanému systému a snaží sa čo najviac skryť svoju prítomnosť pred iným softvérom. Väčšinou útočníci vniknú do systému pomocou rôznych exploitov a získajú privilegovaný prístup. Na zaistenie perzistencie môžu inštalovať rootkit do bežne používaných nástrojov – v takomto prípade hovoríme o user-mode rootkite – alebo pri sofistikovanejších prístupoch do komponentov samotného jadra – nazývaný kernel-mode rootkit.
- Adware – Malvér, ktorý násilne ukazuje reklamy na infikovanom systéme a môže aj bez vedomia užívateľa inštalovať nechcený softvér. Adware sa väčšinou infiltruje do počítača pri sťahovaní bezplatných nástrojov.
- Flooder – Používaný pre útok na sieť počítačov. Flooder vytvára záťaž na zneprístupnenie služieb na sieti (Denial of Service alebo DoS). Ak je zneprístupnenie služieb uskutočnené simultánne z viacerých počítačov, hovoríme o distribuovanom zneprístupnení služieb (DDoS alebo Distributed Denial of Service) a systémy, ktoré útok vykonávajú sa nazývajú zombies.
- Botnet – Sieť počítačov infikovaná rovnakým malvérom. Útočník ovláda každý jeden uzol siete (nazývaný bot) a dokáže im hromadne posielať príkazy cez command and control server (kontrolný server útočníka). Pomocou botnetov môžu tiež vzniknúť DDoS útoky, kedy botnet posiela obrovskú záťaž na napadnutý server vo forme sieťovej prevádzky.
- Downloader – Program používaný v prípadoch, kedy je obtiažne doručiť celý malvér na počítač obete. Tieto programy sú veľkosťou menšie, čo má za príčinu ľahkú dopravu na systém, ktorý má byť infikovaný. Downloader sám o sebe nekoná žiadnu škodlivú funkciu – naozajstný malvér stiahne zo siete a ten potom spustí.

3.2 Detekcia malvéru pomocou YARA pravidiel

Pravidlá pre popis malvéru patria k jednému z možných spôsobov na detekciu malvéru. Na ich vytvorenie existujú rôzne technológie, do ktorých patrí YARA[1], Snort[2], Surricata[3], Sigma[4] a rôzne iné. V tejto práci budú na detekciu malvéru používané YARA pravidlá – tie dovoľujú malvér odhaliť podľa statických sekvencií alebo behaviorálnych prvkov a klasifikovať ho do rodiny, ku ktorej patrí [12].

YARA je nástroj, pomocou ktorého sa dajú tvoriť statické (vyhľadávanie sekvencií) a dynamické (sledovanie správania malvéru) pravidlá pre klasifikáciu programov. Malvéroví analytici ju dokážu použiť na klasifikáciu vzoriek malvéru do jednotlivých typov a rodín a jeho následnú detekciu.

Detekovaný malvér je potrebné priebežne kontrolovať, z dôvodu možnej detekcie falošne pozitívnych vzoriek. Tvorba pravidiel by mala nastať po samotnej analýze škodlivého programu. Vtedy má malvérový analytik predstavu o tom, čo malvér robí a dokáže ho popísať.

Každé YARA pravidlo začína kľúčovým slovom `rule`, po ktorom prichádza jeho meno (v tomto prípade `static_example_rule`) a blokom oddeleným zloženými zátvorkami, v ktorom sú popísané jednotlivé sekcie.

- `meta` – voliteľná sekcia, obsahuje informácie o pravidle,
- `strings` – voliteľná sekcia, uvádza sa v prípade hľadania statických sekvencií,
- `condition` – obsahuje podmienku, ktorá musí byť splnená, aby bola vzorka týmto pravidlom detekovaná [4].

V ukážke 3.1 je možné vidieť štruktúru jednoduchého statického YARA pravidla.

```
rule static_example_rule
{
  meta:
    author = "Samuel Vojtas"
    description = "Ukazkove pravidlo"

  strings:
    $s01 = "ASCII retazec znakov" ascii
    $s02 = { 00 01 02 03 }
    $s03 = "WIDE retazec znakov" wide
    $s04 = { 12 34 56 }

  condition:
    ($s01 or $s02) and ($s03 or $s04)
}
```

Výpis 3.1: Jednoduché statické pravidlo

V tomto pravidle sa v sekcii `strings` nachádzajú štyri reťazce. Reťazce znakov (`$s01` a `$s03`) môžu byť uvedené s ich typom – `ascii`, `wide` alebo iné – ale aj bez neho. Ich typ špecifikuje v akej podobe sa vo vzorke nachádzajú. Hexadecimálne reťazce (`$s02` a `$s04`) sú

[1] Dokumentácia nástroju YARA dostupná na https://yara.readthedocs.io/
[2] Snort dostupný z https://www.snort.org/
[3] Surricata dostupná z https://suricata.io/
[4] Sigma dostupná z https://github.com/SigmaHQ/sigma

v zložených zátvorkach a popisujú byty obsiahnuté vo vzorke zapísané v hexadecimálnom tvare. Sekcia `condition` hovorí, že pravidlo je spustené keď skúmaná vzorka obsahuje aspoň jeden z reťazcov `$s01` a `$s02` a zároveň aspoň jeden z reťazcov `$s03` a `$s04` [4]. Sekcia `meta` bude kvôli úspornosti miesta v tejto práci v ďalších pravidlách vynechaná.

Toto pravidlo využíva len statické sekvencie. Informácie o chovaní sa dajú získať v spojení so sandboxovými technológiami. Tie analyzujú priebeh spustenia vzorky a generujú tzv. report. Tento report je potom v nástroji YARA použitý na detekciu pomocou behaviorálnych prvkov. V tejto práci je na takýto účel používaný sandbox Cuckoo[5], pre ktorý YARA obsahuje samostatný modul `cuckoo` používaný v pravidlách pre opis chovania malvéru [8].

Malvér môže svoje chovanie naprieč spusteniami meniť (napr. kvôli počítaču, času spustenia a pod.) a vytvorené súbory alebo URL, na ktoré sa snaží pripojiť nemusia byť vždy tie isté. Preto modul `cuckoo` vo svojich funkciách používa regulárne výrazy (oddelené lomítkom `/.../` a popísané v uvedenej dokumentácii k nástroju YARA), aby analytik škodlivého kódu mohol špecifikovať väčší rozsah možných artefaktov, ktoré malvér vykoná [4].

```
import "cuckoo"

rule behavioral_example_rule
{
  condition:
    2 of [
      cuckoo.filesystem.write(/\\Desktop\\readme\.txt/),
      cuckoo.network.http_request(/google\.com$/),
      cuckoo.process.executed_command(/vssadmin delete shadows \/all \/quiet/i)
    ]
}
```

Výpis 3.2: Jednoduché behaviorálne pravidlo

Modul `cuckoo` obsahuje rôzne spôsoby popisu na chovanie programov. V pravidle `behavioral_example_rule` (ukážka 3.2) sa dajú vidieť niektoré z nich. Konkrétne toto pravidlo obsahuje konštrukciu `m of n`, ktorá hovorí, že pravidlo je aktivované ak vzorka vykoná aspoň `m` (v tomto prípade 2) z `n` (v tomto prípade 3) uvedených udalostí. V tomto prípade:

- zapíše do súboru `readme.txt` v adresári `Desktop`,
- pošle HTTP požiadavku na URL končiace sa `google.com` a
- spustí príkaz `vssadmin delete shadows /all /quiet` (`/i` na konci regulárneho výrazu hovorí, že ide o case-insensitive príkaz, teda nezáleží na veľkosti písmen).

Behaviorálne prvky sa dajú kombinovať rôznymi spôsobmi, aby popísali čo najdetailnejšie chovanie vzorky malvéru, ktorá má byť klasifikovaná[6].

3.3 Ransomvér

Ransomvér je jeden z druhov malvéru, ktorý určite patrí k tým najdeštruktívnejším. Do tejto kategórie spadá každý softvér, ktorého účelom je znemožniť užívateľom prácu s ich

[5] Cuckoo dostupný z https://cuckoosandbox.org/

[6] Modul Cuckoo pre YARA pravidlá dostupný z https://yara.readthedocs.io/en/stable/modules/cuckoo.html

počítačom s cieľom vybrať výkupné, aby počítač a všetko na ňom bolo ako v stave pred útokom. Môžeme sem zaradiť dva hlavné typy ransomvéru podľa spôsobu vydierania. Tie ktoré zašifrujú a znemožnia prístup k súborom systému, a tie, ktoré užívateľovi znemožnia prístup k samotnému systému ako takému. Hlavný zdroj tejto sekcie je [1].

Zjednodušený postup útoku malvéru typu ransomvér

Útok ransomvéru má zaužívaný postup, ktorý vo väčšine prípadov dodržiava. Jedná sa o niekoľko postupných krokov ako sa dostane od útočníka až po obeť s odopretým prístupom ku svojmu počítaču alebo súborom. Vykonávané kroky sú znázornené na obr. 3.1.

Obr. 3.1: Zjednodušený postup útoku malvéru typu ransomvér

1. Nasadenie – Spôsob prepravy malvéru z počítača útočníka na počítač obete.
2. Inštalácia – Malvér sa zväčša zväčša snaží o dosiahnutie perzistencie, ransomvér nie je výnimkou. V prípade potreby komunikácie s autorom musí byť neustále spustený.
3. Očakávanie príkazov – Niektoré varianty ransomvéru začnú so svojou činnosťou hneď po inštalácii, zatiaľ čo ostatné môžu čakať na ďalšie príkazy z kontrolného serveru.
4. Deštrukcia (Šifrovanie) – Samotné odoprenie systému užívateľovi. Po ukončení tejto fázy užívateľ nemá prístup k niektorým zdrojom, ktoré boli vybrané ransomvérom.
5. Vydieranie – Požadovanie výkupného od obete, ktoré väčšinou prebieha pomocou anonymných internetových transakcií, napr. Bitcoin[7].

Po vytvorení malvéru ho útočník potrebuje dostať na systém obete. Počítač, ktorý má byť infikovaný v sebe môže obsahovať zraniteľnosť. Tá umožní útočníkovi dostať sa do vnútra systému a ransomvér stiahnuť a aj spustiť. V iných prípadoch musí byť obeť oklamaná tak, aby ransomvér spustila bez vedomia toho, že ide o škodlivý kód. V takejto situácii sa používajú techniky sociálneho inžinierstva – obeti môže byť napr. poslaný spam, ktorý v prílohe obsahuje malvér.

Spam, ako jeden z frekventovaných šíriteľov ransomvéru, núti poskytovateľov elektronickej pošty dávať pozor a filtrovať prichádzajúcu poštu tak, aby minimalizovali riziko. Medzi inými, ide aj o sledovanie typu súboru, ktorý je zaslaný. Útočníci sa snažia teda svoj kód skryť pred týmito filtrami a neposielajú celý ransomvér, len kus programu, ktorý sa ľahšie infiltruje k obeti a daný ransomvér stiahne – môžu mať rôzne podoby, od makra v dokumentoch MS Office až po archívy, ktoré obsahujú škodlivý JavaScript.

Pri inštalácii sa ransomvér snaží zistiť na akom stroji operuje, resp. o akú obeť sa jedná. V niektorých prípadoch môže so svojím vykonávaním prestať a nepokračovať ďalej – mnoho ransomvérov sa rozhoduje o ďalšom vykonávaní podľa jazyka, ktorý obeť používa na svojom

[7]Decentralizovaná digitálna mena, viac na https://bitcoin.org/en/

systéme alebo sa ukončí pri zistení, že beží vo virtuálnom stroji (obeť môže byť malvérový analytik a týmto ransomvér znepríjemní jeho prácu). Ak je vyžadovaná perzistencia na danom systém, vo väčšine prípadov sa používa zápis do registrov MS Windows, aby bol škodlivý kód spustený spolu so štartom počítača.

Niekedy ransomvér potrebuje kontaktovať svojho autora. Môže vyžadovať ďalšie príkazy, aby vedel ako pokračovať alebo pošle dáta zo systému obete, ktoré sú potom použité v procese vydierania.

Po tomto kroku sa zvyčajne prechádza k deštrukcii. Používanie vybranej časti systému je užívateľovi odopreté. Môže ísť o zašifrovanie súborov alebo znemožnenie ovládania niektorých častí systému (napr. prekrytie celej obrazovky správou od útočníka o zaplatení výkupného, ktorá sa nedá vypnúť – tento typ ransomvéru sa používal hlavne v minulosti a dnes už sa nepoužíva). V súčasnosti je rozšírenejšia varianta so zašifrovanými súbormi, a preto sa ďalej v tejto práci bude ako ransomvér označovať práve tento druh.

Až keď sú všetky vybraté súbory zašifrované, na počítači obete sa objaví tzv. správa o výkupnom (z angl. ransom note) podobná ako na obr. 3.2. Hovorí o tom, ako má obeť v prípade záujmu o dešifrovanie súborov útočníka kontaktovať, sumu a spôsob zaplatenia výkupného. V súčasnosti sa využíva aj tzv. dvojité vydieranie (double extortion), kedy ransomvér môže naviac užívateľovi hroziť aj zverejneným privátnych dát, v prípade, kedy nezaplatí.

Obr. 3.2: Správa o výkupnom z ransomvéru WannaCry

Proces šifrovania súborov

Prvým krokom pri šifrovaní súborov je samotný výber toho, čo bude zašifrované. Ransomvér cieli na súbory, ktoré sú pre obeť cenné. Tie sa nachádzajú napríklad v domovských adresároch užívateľa alebo môže ísť o zálohy systému, aby nebolo možné z nich obnoviť systém z predchádzajúceho stavu.

Pri výbere spôsobu samotného šifrovania prichádzajú do úvahy dve varianty – symetrické a asymetrické šifrovanie. Každé z nich má svoje výhody a nevýhody.

Symetrické šifrovanie využíva len jeden kľúč – ten šifruje aj dešifruje. Tento proces je rýchly a nedochádza k veľkému preťaženiu procesoru pri použití na veľké množstvo súborov. Väčšina programov používajúca symetrické šifrovanie využíva algoritmu AES (Advanced Encryption Standard)[8], Salsa20[9] alebo RC4 (Rivest Cipher 4)[10].

Asymetrické šifrovanie používa dva typy kľúčov, verejný, ktorým zašifruje dáta a súkromný, ktorým sú dáta dešifrované. Týmto spôsobom ransomvér dokáže zašifrovať súbor tak, že jediný útočník ho vie priviesť do pôvodného stavu. Verejný kľúč sa môže už priamo nachádzať v malvéri alebo je stiahnutý cez spojenie s útočníkom. Narozdiel od symetrického šifrovania je tento proces pomalší a používa kľúče väčšej dĺžky. Medzi najznámejšie algoritmy využívajúce asymetrické šifrovanie patrí RSA[11] a ECDH (Elliptic-curve Diffie–Hellman)[12].

Autori ransomvéru sa snažia využiť výhody symetrického a asymetrického šifrovania a minimalizovať ich nevýhody. Preto najčastejšie dochádza ku kombinácii oboch prístupov. Najfrekventovanejšia technika v dnešných ransomvéroch je použitie symetrického šifrovania na jednotlivé súbory s generovaním unikátneho kľúča pre každý súbor. Každý kľúč je potom asymetricky šifrovaný verejným kľúčom a vložený do samotného symetricky zašifrovaného súboru. Ak autor ransomvéru využije tento spôsob správne, neexistuje spôsob ako ich dešifrovať bez informácií od útočníka.

Tvorca ransomvéru môže pri jeho vytváraní urobiť chybu v kryptografii. Niektoré praktiky, ktoré autori ransomvéru používajú umožňujú zistiť jednotlivé kľúče v procese šifrovania. Kľúč môže byť nedopatrením uvedený v ransomvéri alebo sa dá množinu jeho možných hodnôt výrazne obmedziť – napríklad ak ransomvér používa algoritmy na vytvorenie kľúču, ktoré produkujú hodnoty s príliš krátkou dĺžkou. V tomto prípade sa dajú kľúče zistiť pomocou tzv. brute-force útokov, kedy sa jednoducho skúšajú súbory dešifrovať pomocou všetkých možných hodnôt z intervalu, ktorý sa stanoví. Programy nazývané dekryptory využívajú takýchto chýb a umožňujú sprístupniť zašifrované súbory bez potreby platenia výkupného.

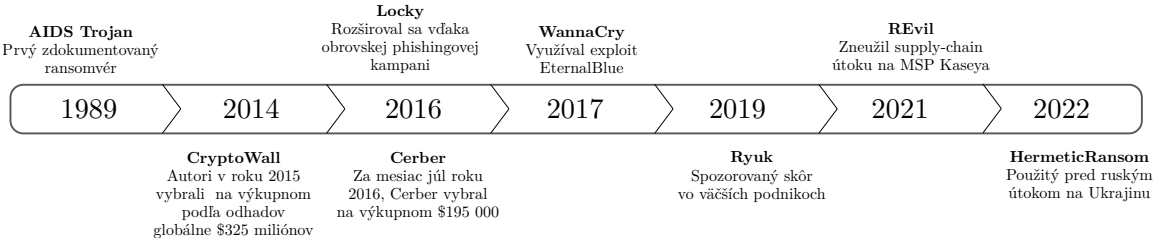
História a rozšírené rodiny ransomvéru

História ransomvéru sa dátuje späť do roku 1989 – za prvý zdokumentovaný ransomvér sa považuje AIDS Trojan [11]. Najprv sa šifrovacie ransomvéry veľmi neujali, až niekoľko rokov po tomto incidente sa téma ransomvéru a výberu výkupného za dešifrovanie súborov dostala do väčšieho povedomia. V tejto časti budú popísané rodiny, ktoré sa v rámci histórie stali najviac známymi. Na obr. 3.3 sa dajú vidieť spolu s časom začiatku ich šírenia. Okrem nich existuje veľké množstvo ďalších rodín a v súčasnosti prichádzajú stále nové a nové.

AIDS Trojan

Ako už bolo uvedené, AIDS Trojan sa považuje za jeden z prvých ransomvérov. Vytvoril ho harvardský biológ Joseph L. Popp a zaslal 20 tisíc infikovaných disket návštevníkom AIDS konferencie, ktorú poriadalo World Health Organization. Po 90 reštartovaniach infikovaného

[8] https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf
[9] https://cr.yp.to/snuffle/spec.pdf
[10] https://www.geeksforgeeks.org/what-is-rc4-encryption/
[11] https://datatracker.ietf.org/doc/html/rfc3447
[12] http://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Haakegaard+Lang.pdf



Obr. 3.3: Časová os najvýznamnejších rodín ransomvéru

systému, ransomvér schoval vybrané adresáre a zašifroval mená súborov. Výška výkupného dosahovala $189 [11].

CryptoWall

CryptoWall[13] si v čase svojej pôsobnosti od konca roku 2013 po marec 2016 prešiel šiestimi veľkými aktualizáciami. FBI odhadlo, že za rok 2015 boli tvorcovia tohto ransomvéru schopní na výkupnom vybrať až $325 miliónov po celom svete. Šíril sa cez phishingové kampane hlavne v emailoch s prílohou prípony `.scr` alebo za pomoci rôznych druhov exploitov (najviac Angler exploit kit[14]).

TeslaCrypt

TeslaCrypt[15] bol prvýkrát spozorovaný v roku 2015. Šírenie prebiehalo cez spam alebo rôzne zraniteľnosti v systémoch. Popri šifrovaní dôležitých dokumentoch sa zameriaval aj na šifrovanie herných súborov. Skupina zodpovedná za tento ransomvér sa rozhodla v roku 2016 ukončiť svoje pôsobenie a zverejnila súkromný kľúč, s ktorým sa dajú dešifrovať všetky súbory zašifrované ransomvérom TeslaCrypt. Niektoré antivírusové spoločnosti vydali dekryptor, ktorý následky ransomvéru TeslaCrypt dokáže zvrátiť.

Cerber

Ransomvér Cerber[16] – ako ukazuje niekoľko indikátorov – bol pravdepodobne vytvorený skupinov hackerov z Ruska, ktorá bola dobre financovaná. Nasvedčuje tomu pravidelné vydávanie nových verzií pre odstraňovanie zraniteľností a fakt, že v priebehu procesu šifrovania súborov sa pri obetiach zisťovalo, v ktorom štáte sa nachádzajú alebo aký jazyk používajú na klávesnici. Ak je tento jazyk ruský, ransomvér skončil so svojím prevádzaním.

Cerber ransomvér sa používal ako RaaS (Ransomware-as-a-Service), teda jeho autori poskytujú ich klientom tento program s tým, že ak dôjde k prevzatiu výkupného, zisk si tvorcovia Cerberu a ich klienti rozdeľujú. Suma peňazí, ktorá sa na výkupnom vybrala sa vyšplhala až do výšky $195 tisíc za mesiac júl v roku 2016.

Cerber sa šíril hlavne cez spam a rôzne druhy exploitov. Najlepším spôsobom ako sa pred ním chrániť je edukácia ohľadom spamu a otvárania príloh z nevyžiadanej pošty, aktualizácie softvéru, tvorenie záloh systému a vypnutie makier v dokumentoch Microsoft Office.

[13] https://malpedia.caad.fkie.fraunhofer.de/details/win.cryptowall

[14] https://unit42.paloaltonetworks.com/unit42-understanding-angler-exploit-kit-part-1-exploit-kit-fundamentals/

[15] https://malpedia.caad.fkie.fraunhofer.de/details/win.teslacrypt

[16] https://malpedia.caad.fkie.fraunhofer.de/details/win.cerber

Locky

Vo februári 2016 sa prvýkrát objavil Locky ransomvér[17], ktorý v počte infikovaných užívateľov dosahoval obrovské čísla. Locky ransomvér sa šíril hlavne cez spam a to vo forme skomprimovaných archívov, ktoré obsahujú JavaScriptový súbor alebo Microsoft Office dokumentov s makrami. Taktiež existujú podozrenia, že za týmto ransomvérom je skupina ruských hackerov, pretože ako aj pri Cerber ransomvéri, aj Locky neútočí na užívateľov, ktorí sa nachádzajú v Rusku.

Táto skupina bola pravdepodobne vysoko financovaná, pretože predpokladaný počet nevyžiadaných emailových správ, ktoré dokázali poslať na vrchole svojej činnosti dosahoval až 4 milióny týždenne a taktiež vydávali pravidelné aktualizácie tohto ransomvéru.

WannaCry

WannaCry[18] využíval exploit známy pod menom EternalBlue, ktorý zneužíval chybu v Sambe (implementácia sieťového protokolu SMB)[19]. Neskoré zareagovanie firiem a odkladanie aktualizácii na opravu tejto chyby spôsobilo celosvetový nárast útokov tohto ransomvéru. Odhadovaný počet systémov, ktoré boli napadnuté je až 230 tisíc. Obetiam, ktoré zaplatili za výkupné neboli dáta poskytnuté späť [10].

Ryuk

Ransomvér Ryuk[20] cieli na organizácie a väčšie firmy. Prebral časť kódu zo staršieho ransomvéru Hermes, ktorý bol využívaný skupinou hackerov zo Severnej Kórei. Nebolo ale aj tak doposiaľ zistené, či Ryuk naozaj vznikol v tejto krajine a vedie sa spor či pochádza z Ruska alebo z už spomínanej Severnej Kórei. Keďže sa útočníci zameriavajú na organizácie vlastniace dôležité dáta, vyžaduje od nich väčšie výkupné ako je zvykom – od pätnásť až do päťdesiat bitcoinov. Ryuk sa skoro výhradne šíri v podobe trójskeho koňa a obsahuje množstvo techník na sťaženie forenznej analýzy alebo pokročilé perzistenčné mechanizmy. Jeho šifrovacia schéma pozostáva zo symetrického šifrovania, pričom kľúč je ďalej asymetricky zašifrovaný a vložený do samotného súboru. Ryuk niekedy vyberá na zašifrovanie aj súbory, ktoré sú zodpovedné za správny chod systému. Z tohto dôvodu môže vzniknúť situácia, kedy obeť nemá vôbec prístup k svojmu počítaču, pretože ten sa kvôli nefunkčnosti niektorých súborov nedokáže ani zapnúť [3].

REvil

REvil[21], tiež známy pod menom Sodinokibi, je jedna z moderných rodín ransomvéru packovaná vlastným packerom. Svoje pôsobenie začala v 2019, no vrchol dosiahla až v roku 2021 na Deň nezávislosti v USA. Vtedy využila supply-chain útoku[22] na MSP[23] Kaseya VSA (služba pre vzdialené monitorovanie podnikov), čo umožňilo infiltráciu do približne

[17] https://malpedia.caad.fkie.fraunhofer.de/details/win.locky
[18] https://malpedia.caad.fkie.fraunhofer.de/details/win.wannacryptor
[19] Samba dostupná z https://www.samba.org/
[20] https://malpedia.caad.fkie.fraunhofer.de/details/win.ryuk
[21] https://malpedia.caad.fkie.fraunhofer.de/details/win.revil
[22] Kybernetický útok, ktorý sa zameriava na menej zabezpečené organizácie v rámci dodávateľského reťazca (supply chain – sieť medzi organizáciou a jej dodávateľmi na vytvorenie produktu)
[23] Managed Service Provider – organizácia spravujúca informačné technológie iného zákazníka

40 organizácií. Takisto ako Cerber sa šíri v podobe RaaS, teda autori poskytujú ransomvér klientom, s ktorými si delia zisk [14].

HermeticRansom

HermeticRansom[24] je ransomvér napísaný v programovacom jazyku Go. Jeho šírenie v roku 2022 sa odohrávalo najmä pred útokom Ruska na Ukrajinu. Ide o slabo navrhnutý ransomvér, ktorého kryptografické schéma bolo možné prelomiť – v dôsledku firma Avast na tento ransomvér ako jedna z prvých vydala dekryptor [2].

Ochrana proti ransomvéru

Rôzne druhy ransomvéru používajú rôzne techniky šírenia, často dochádza k infekcii cez spam alebo cez zraniteľnosti v systémoch. Našťastie, oba problémy dokáže užívateľ aspoň čiastočne riešiť. Pravidelnými aktualizáciami systému je možné výrazne znížiť problém ransomvéru šíriaceho sa cez zraniteľnosti a exploity.

Čo sa týka nevyžiadanej pošty, je dôležité, aby užívatelia boli oboznámení s hrozbou spamu a ako s ním zaobchádzať. Takisto je dôležité dbať aj na bezpečnosť v rámci siete a mať správne nakonfigurovaný firewall tak, aby nedochádzalo k nežiaducim sieťovým spojeniam.

Súbory, ktoré sú pre užívateľa nejakým spôsobom dôležité, si môže zálohovať na bezpečné miesto (pri útoku ransomvérom sú súbory obnovené zo záloh) alebo použiť tzv. ransomvér štít[25], ktorý dovoľuje určiť adresár s takýmito súbormi a chrániť ich pred modifikáciou, zmazaním alebo zašifrovaním podozrivými aplikáciami.

Ak aj napriek prevencii užívateľ podľahne útoku ransomvéru, mnoho antivírusových firiem ponúka dekryptory na rôzne druhy rodín ransomvéru. Dekryptor pre konkrétny ransomvér, ktorý užívateľ potrebuje nemusí existovať, a preto sa táto možnosť berie ako posledná.

[24] https://malpedia.caad.fkie.fraunhofer.de/details/win.partyticket
[25] https://support.avast.com/en-us/article/Antivirus-Ransomware-Shield-FAQ/

Kapitola 4

Analýza a návrh obrany proti ransomvéru

Pre návrh vhodnej obrany proti ransomvéru je dôležité najprv konkrétny ransomvér analyzovať a do istej úrovni pochopiť ako funguje. Analýza bola vykonaná pomocou techník reverzného inžinierstvo popísaných v kapitole 2 cez statickú a dynamickú analýzu. V rámci statickej analýzy sa používali nástroje ako PEiD, DIE, De4dot[1] na unpackovanie a deobfuskovanie vzoriek, PEView a RetDec na zistenie formátu súboru a IDA Pro ako disassembler a dekompilátor (alebo dekompilátor z programu dnSpy[2] ak išlo o vzorky .NET frameworku). Dynamická analýza využila nástroje Process Monitor, Process Explorer, NetCat a Wireshark na monitorovanie systémových a sieťových udalostí a OllyDbg alebo dnSpy (ak išlo o vzorky .NET) ako debugger. Cuckoo sandbox poskytoval informácie použité počas dynamickej analýzy, ktoré zjednodušili tvorbu detekčných mechanizmov pre dané vzorky. V rámci boja proti ransomvéru autor prispel v dvoch kategóriach – vytvoril YARA pravidlá pre klasifikáciu a detekciu ransomvéru, a dekryptory pre rodiny s kryptografickými chybami, ktoré našiel počas svojej analýzy.

4.1 Detekcia pomocou YARA pravidiel

Vytvorené YARA pravidlá majú za úlohu správnu klasifikáciu a následnú detekciu malvéru – pre jej zlepšenie je vhodné pravidlá rozdelovať podľa rôznych typov (každý typ klasifikuje podľa iného javu). V tejto práci boli použité:

- Behaviorálne pravidlá – Dynamické pravidlá, ktoré popisujú správanie malvéru, teda zápis do registrov, vytváranie a vymazávanie súborov, pripojenie k určitej IP adrese a podobne.
- Pravidlá so statickými sekvenciami – Statické pravidlá výhodné pre nezapackované vzorky, ktoré obsahujú sekvencie bytov alebo znakov v pôvodnej podobe. Môžu sem patriť napríklad reťazce, ktoré sa vypisujú pri žiadaní ransomvéru o výkupné alebo sekvencie bytov, ktoré predstavujú kľúče pre šifrovanie.
- Pravidlá detekujúce známe objekty – Dynamické pravidlá detekujúce pomenované objekty operačného systému (objekty, ktoré vytvára operačný systém napr. na zaistenie synchronizácie paralelných procesov) vytvorené daným malvérom. Mená takýchto

[1] De4dot dostupný z https://github.com/de4dot/de4dot
[2] dnSpy dostupný z https://github.com/dnSpy/dnSpy

objektov sú zvyčajne uvedené priamo v kóde, alebo sa na základe deterministických algoritmov dajú zistiť, pomocou čoho môže byť malvér detekovaný.

Na nasledujúcich stranách budú popísané charakteristické črty jednotlivých rodín ransomvéru a ako ich využiť na jeho detekciu a následnú klasifikáciu. Rodiny vystavené analýze boli vybraté na základe ich súčasného šírenia alebo vďaka poznaniu, že varianty z týchto rodín sa často dajú dešifrovať.

K rodinám budú uvedené pravidlá vytvorené počas analýzy spolu s popisom chovania ransomvéru odkazujúceho sa na riadky kódu pravidla pomocou skratky *r.*, napríklad: Ransomvér vytvára súbor na pracovnej ploche (r. 13).

TimeTime

Ako jeden z prvých opísaných rodín ransomvéru je TimeTime. Ten sa ako prvýkrát objavil v decembri 2021. Ide o jednoduchý neobfuskovaný typ ransomvéru napísaný v jazyku C#. Pri svojom vykonávaní vymaže systémové zálohy (r. 23) a potom začne šifrovať vybrané súbory, ktorým pridáva príponu `.timetime` (r. 20). Taktiež v priebehu svojho prevádzania zapisuje do registra `HKEY_CURRENT_USER\TimeTime` (r. 22).

Do rôznych adresárov umiestňuje súbor `@__RECOVER_YOUR_FILES__@.txt` (r. 21), vyžadujúci od obeti výkupné vo výške $100. Podľa inštrukcií uvedených v správe o výkupnom má obeť zaplatiť danú sumu útočníkovi, ktorý im naspäť poskytne dešifrovací kľúč. Tento kód musí byť zadaný do programu `@_DECRYPTOR_@.exe` (niektoré varianty TimeTime ransomvéru ho samé obsahujú v svojom kóde a počas prevádzania ho vytvoria) a ten dané súbory dešifruje.

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule timetime_known_sequences
5  {
6    strings:
7      // utrzky spravy o vykupnom
8      $s01 = "---------------Time Time Ransomware---------------"
9      $s02 = "Please, find @_DECRYPTOR_@.exe on your desktop to pay the ransom. If
           you don't find it"
10     $s03 = "You got epicly pwned."
11     // prikaz na vymazanie systemovych zaloh
12     $s04 = "vssadmin delete shadows /all /quiet & wmic shadowcopy delete" wide
13   condition:
14     EXE and all of them
15 }
16
17 rule timetime_known_behavior_high
18 {
19   condition:
20     cuckoo.filesystem.file_write(/\.timetime/) and
21     cuckoo.filesystem.file_write(/@__RECOVER_YOUR_FILES__@\.txt/) and
22     cuckoo.registry.key_write(/^HKEY_CURRENT_USER\\TimeTime/) and
23     cuckoo.process.executed_command(/VSSADMIN DELETE SHADOWS \/ALL \/QUIET/i)
24 }
```

Výpis 4.1: Statické a behaviorálne pravidlo pre rasnomvér TimeTime

Statické pravidlo využíva znakových reťazcov vo vnútri binárneho súboru ransomvéru popísaných priamo v kóde – ide o časti správy o výkupnom alebo príkaz na vymazanie systémových záloh. Na uistenie toho, že sa jedná o spustiteľný súbor pod operačným systémom Microsoft je použité pravidlo `EXE` zo súboru `includes/all.yar` (interný súbor privátnych pravidiel pre určenie súborového formátu typu EXE alebo ELF), ktoré je na riadku 1 importované.

Pravidlá v ukážke 4.1 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 2. marca 2022.

Redeemer

Redeemer je ransomvér, ktorý po napadnutí začne šifrovať súbory obete a na rozlíšenie od ostatných súborov im pridáva príponu `.redeem` (r. 38). Na konci prevádzania vytvorí správu o výkupnom (`Read Me.TXT`, r. 37), kde podrobne popisuje ako má obeť kúpiť kryptomenu Monero[3] a zaplatiť ňou potrebnú sumu na dešifrovanie súborov. Útočník žiada výkupné vo výške 5 Monero, čo je v prepočte v roku 2022 približne 1000 EUR alebo 25 000 CZK.

Ku koncu prevádzania ransomvér vytvára pomenovaný systémový objekt (`RedeemerMutex`, r. 7), ktorý slúži na zistenie, či už ransomvér na danom systéme beží. Ak tento objekt existuje, iná inštancia ransomvéru už bola spustená. Takisto sa Redeemer zapíše aj do registrov MS Windows tak, aby bol spustený vždy pri štarte systému (r. 40). Teda ak sa jeho prvému behu nejakým spôsobom nepodarilo zašifrovať súbory, ransomvér bude spustený opäť po reštarte. Na konci prevádzania ransomvéru je ukázaná užívateľovi správa pomocou príkazu `msg`, ktorá ho naviguje k správe o výkupnom (r. 41).

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule redeemer_known_named_objects
5  {
6    condition:
7      cuckoo.sync.mutex(/^RedeemerMutex$/)
8  }
9
10 rule redeemer_known_sequences
11 {
12   strings:
13     // pripona zasifrovanych suborov
14     $s01 = ".redeem" wide
15     // meno suboru spravy o vykupnom
16     $s02 = "Read Me.TXT" wide
17     // cast identifikacie obeti
18     $s03 = "-----BEGIN REDEEMER PUBLIC KEY-----"
19     $s04 = "-----END REDEEMER PUBLIC KEY-----"
20     // registere MS Windows, do ktorych sa zapisuje
21     $s05 = "SOFTWARE\\Redeemer"
22     $s06 = "KeyHash"
23     $s07 = "LegalNoticeText"
24     $s08 = "LegalNoticeCaption"
25     // nazov skriptu ulozenom v ransomveri
26     $s09 = "rem.bat" wide
```

[3] Decentralizovaná digitálna mena, viac na https://www.getmonero.org/

```
27      // abeceda pre deobfuskaciu retazcov (pouziva Base64)
28      $s10 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
29    condition:
30      EXE and 7 of them
31  }
32
33  rule redeemer_known_behavior_high
34  {
35    condition:
36      3 of [
37        cuckoo.filesystem.file_write(/\\Read Me\.TXT$/),
38        cuckoo.filesystem.file_write(/\.redeem$/),
39        cuckoo.registry.key_write(/^HKEY_LOCAL_MACHINE\\SOFTWARE\\Redeemer\\KeyHash$
              /),
40        cuckoo.registry.key_write(/^HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows
               NT\\CurrentVersion\\Winlogon\\(LegalNoticeCaption|LegalNoticeText)$/),
41        cuckoo.process.executed_command(/MSG \* REDEEMER RANSOWMARE - THIS FILE
              CANNOT BE OPENED UNTIL DECRYPTED\. CHECK README\.TXT FOR MORE DETAILS HOW
               TO DECRYPT YOUR FILE\.$/i)
42      ]
43  }
```

Výpis 4.2: Všetky druhy pravidiel pre pre ransomvér Redeemer

Statické pravidlo opäť využíva napr. prípony zašifrovaných súborov, správu o výkupnom alebo použité registre. Medzi unikátne sekvencie znakov patrí štruktúra kryptografického kľúča začínajúca sa sekvenciou `$s03` a končiaca sa s `$s04` – podobá sa na štruktúru RSA kľúču. Redeemer na obfuskáciu reťazcov používal base64 kódovanie a teda potreboval abecedu z `$s10`, aby ich mohol deobfuskovať. Vďaka pravidlu `EXE` môže byť statické pravidlo detekované len pri spustiteľných súboroch pod operačným systémom Windows.

Pravidlá v ukážke 4.2 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 17. februára 2022.

Khonsari

Khonsari[4] je ransomvér, ktorý sa začal šíriť cez zraniteľnosť log4shell[5] krátko po jej odhalení v decembri 2021. Je napísaný v jazyku C#, a preto bol na jeho analýzu použitý disassembler a debugger dnSpy.

Reťazcové literály vo vnútri ransomvéru sa nedajú zistiť len pomocou statickej analýzy, pretože Khonsari ich obsahuje v obfuskovanej podobe. Tie sú až za behu programu deobfuskované pomocou jednoduchej XOR šifry. Takéto reťazce nie je výhodné použiť pri tvorení statického detekčného pravidla, pretože autor ransomvéru môže hocikedy zmeniť spôsob obfuskácie, čo by malo za príčinu znefunkčnenie pravidla.

Khonsari po spustení kontaktuje kontrolný server (r. 17) a potom sa ako ostatné ransomvéry pokúsi o šifrovanie obsahu vybraných adresárov. Zašifrovaným súborom pridáva príponu `.khonsari` (r. 18) a pred ukončením svojej činnosti na pracovnej ploche vytvorí súbor `HOW TO GET YOUR FILES BACK.TXT` (r. 19), informujúci obeť o telefónnom čísle a e-maili, ktoré má kontaktovať ak chce vrátiť súbory do pôvodného stavu.

```
1  include "includes/all.yar"
```

[4] https://malpedia.caad.fkie.fraunhofer.de/details/win.khonsari
[5] https://logging.apache.org/log4j/2.x/security.html

```
2  import "cuckoo"
3
4  rule khonsari_known_sequences
5  {
6    strings:
7      // .NET GUID analyzovanej varianty
8      $s01 = "$277e5e6a-4da6-4138-97fa-3fecbdad0176" ascii
9    condition:
10     EXE and any of them
11 }
12
13 rule khonsari_known_behavior_high
14 {
15   condition:
16     2 of [
17       cuckoo.network.host(/^3\.145\.115\.94$/),
18       cuckoo.filesystem.file_write(/\.khonsari$/),
19       cuckoo.filesystem.file_write(/\\HOW TO GET YOUR FILES BACK\.TXT$/)
20     ]
21 }
```

Výpis 4.3: Statické a behaviorálne pravidlo pre ransomvér Khonsari

Keďže Khonsari obsahoval obfuskované reťazce, do statického pravidla bolo vhodné pridať niečo, čo sa nemení pri zmene obfuskačných mechanizmov. V .NET aplikáciach sa vyskytuje `GUID`, ktorý jednoznačne identifikuje daný program. Statické pravidlo bude spustené ak sa jedná o spustiteľný súbor a zároveň obsahuje jedinečný `GUID` (`$s01`).

Pravidlá v ukážke 4.3 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 23. januára 2022.

HiddenTear

Rodina ransomvéru HiddenTear[6] začala už v roku 2015 ako open-source projekt zverejnený na GitHube. V tom čase išlo len o potvrdenie konceptu programátora a tento kód mal mať skôr edukačnú hodnotu, no časom vzniklo množstvo variant, ktoré čerpali práve z tejto verzie rasnomvéru a ich cieľ mal už škodlivý charakter.

Keďže HiddenTear vznikol pomerne dávno, firma Avast už mala pravidlo pre túto rodinu vytvorené[7]. Nové varianty tohto ransomvéru však stále vznikajú, a preto autor práce v rámci analýzy identifikoval aj takéto vzorky, aby s ich pomocou mohol vylepšiť už existujúce pravidlo. Časti nenapísané autorom sú vyznačené tromi bodkami. Z dôvodu, že pravidlo bolo príliš dlhé, je dostupné v sekcii s prílohami v prílohe A.

V rámci analýzy boli skúmané štyri varianty rodiny HiddenTear. Varianty sa v niektorých veciach líšili, no hlavné črty ostali nezmenené. HiddenTear je napísaný v jazyku C#. Po spustení sa snaží vytvoriť svoje kópie pod iným menom (napr. `surprise.exe`, `discord.exe` alebo `local.exe`, r. 44, 50) a následne spustí algoritmus pre získanie šifrovacieho hesla. Niektoré varianty obsahujú konštantné heslo priamo uložené v spustiteľnom súbore, iné vo väčšine používajú triedu `Random` jazyka C# na generovanie pseudo-náhodných čísel, ktoré použijú ako indexy do znakového reťazca. Vygenerovaním určitého počtu indexov ransomvér získava heslo, ktoré používa na symetrické šifrovanie súborov pomocou AES.

[6] https://malpedia.caad.fkie.fraunhofer.de/details/win.hiddentear

[7] Pravidlo vytvoril malvérový analytik firmy Avast.

Varianty pridávali zašifrovaným súborom prípony ako `.NYANCAT`, `.PPLIT`, `.BADFILE` alebo reťazec určitej dĺžky, pozostávajúci z náhodných znakov (r. 45, 53, 57).

Varianty sú väčšinou od rôznych tvorcov a teda správy o výkupnom sú rôzne alebo úplne chýbajú. V niektorých prípadoch HiddenTear dokonca mení pozadie pracovnej plochy, do ktorej správu o výkupnom zahrnie.

HiddenTear prichádza v súboroch s menšou dĺžkou, preto je v pravidlách uvedené, že majú byť aktivované len pri určitej veľkosti spustiteľného súboru (kľúčové slovo `filesize`, r. 32, 38). Statické pravidlo využíva časti zo správy o výkupnom, unikátny GUID alebo absolútnu cestu k debuggovacím symbolom, ktorá je do binárneho súboru vložená pri kompilácii (r. 16, 24).

Pravidlá v ukážke A.1 vytvorené na základe predchádzajúceho popisu ransomvéru boli prvýkrát aktualizované 2. februára 2022.

Nokoyawa

Nokoyawa[8] ransomvér bol prvýkrát zachytený v marci 2022 a svojím chovaním vzbudzuje dojem, že sa jedná o škodlivý kód podobný ransomvéru Hive[9], ktorý sa šíril ešte v roku 2021. Nokoyawa generuje náhodný buffer bytov pomocou kryptografickej funkcie `BCrypt-GenRandom` (r. 30) a ten používa ako kľúč k symetrickému šifrovaniu súborov, ktorým je pridaná prípona `.NOKOYAWA` (r. 28). Informácie o zaplatení výkupného sú uložené v súbore `NOKOYAWA_readme.txt` v adresároch, kde sa nachádzajú zašifrované súbory (r. 29).

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule nokoyawa_known_sequences
5  {
6    strings:
7      // DLL pouzite na kryptograficke funkcie
8      $s01 = "bcrypt.dll" wide
9      // cast nazvu spravy o vykupnom
10     $s02 = "_readme.txt" wide
11     // pripona zasifrovanym suborom
12     $s03 = "NOKOYAWA"
13     // kryptograficka funkcia pouzita na vygenerovanie nahodneho kluca
14     $s04 = "BCryptGenRandom"
15
16     // prepinace pre rozne moznosti spustenia
17     $t01 = "-network" wide
18     $t02 = "-help" wide
19     $t03 = "-file" wide
20     $t04 = "-dir" wide
21   condition:
22     EXE and 3 of ($s*) and 3 of ($t*)
23 }
24
25 rule nokoyawa_known_behavior_high
26 {
27   condition:
28     cuckoo.filesystem.file_write(/\.NOKOYAWA$/) and
```

[8] https://malpedia.caad.fkie.fraunhofer.de/details/win.nokoyawa
[9] https://malpedia.caad.fkie.fraunhofer.de/details/win.hive

```
29      cuckoo.filesystem.file_write(/\\NOKOYAWA_readme\.txt$/) and
30      cuckoo.process.resolved_api(/^bcrypt\.dll!BCryptGenRandom$/)
31  }
```

Výpis 4.4: Statické a behaviorálne pravidlo pre ransomvér Nokoyawa

Statické pravidlo obsahuje informácie o súboroch, ktoré budú vytvorené počas prevádzania, meno DLL a meno kryptografickej funkcie na generovanie náhodných bytov a prepínače ransomvéru, ktorými sa dá nastaviť ako bude Nokoyawa postupovať (reťazce označené písmenom `$t`). Ak je vzorka spustiteľný súbor a obsahuje aspoň 3 prepínače a 3 sekvencie začínajúce na `$s`, pravidlo je spustené.

Pravidlá v ukážke 4.4 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 14. marca 2022.

LokiLocker

LokiLocker[10] je ransomvér napísaný v jazyku C#, ktorý sa začal šíriť v auguste 2021. Analýza tohto ransomvéru je vďaka jeho obfuskačným metódam veľmi náročná aj napriek tomu, že bol použitý deobfuskátor de4dot. LokiLocker sa snaží pripojiť na svoj server a stiahnuť z neho dodatočné dáta (r. 29), následne začína s tradičným postupom ransomvéru popísaným v kapitole o ransomvéri. Zašifrovaným súborom pridáva príponu `.Loki` alebo `.Rainman` (r. 26) a informácie pre zaplatenie výkupného ukladá do súboru `Restore-My-Files.txt` (r. 27). Pre šifrovanie je vygenerovaný pár kľúčov unikátny pre každú obeť, ktorý je zašifrovaný a uložený do súboru `Cpriv.loki` (r. 28).

Útočník dáva obeti ultimátum, do kedy je potrebné splatiť výkupné. Ak sa tak nestane, ransomvér vymaže zašifrované súbory, aby sa v budúcnosti nedali dať do pôvodnej podoby.

```
1   include "includes/all.yar"
2   import "cuckoo"
3
4   rule lokilocker_known_sequences
5   {
6     strings:
7       // cast zo spravy o vykupnom
8       $s01 = "All your files have been encrypted by Loki locker!"
9       // URL pre stiahnutie dalsieho skodliveho obsahu
10      $s02 = "loki-locker.one" wide
11      // pouzivane emaily utocnikov
12      $s03 = "Decryptfiles@goat.si" wide
13      $s04 = "Decoder@firemail.cc" wide
14      $s05 = "Unlockpls.dr01@protonmail.com" wide
15      $s06 = "Unlockpls.dr01@yahoo.com" wide
16      // cast verejneho master kluca
17      $s07 = "zQ6+tkZblOQL5sLM4U24fa+vLl3znNlLehJJZRhHyo1SngiKAWpPI6U0Az+" wide
18    condition:
19      EXE and 3 of them
20  }
21
22  rule lokilocker_known_behavior_high
23  {
24    condition:
```

[10] https://malpedia.caad.fkie.fraunhofer.de/details/win.lokilocker

```
25      2 of [
26        cuckoo.filesystem.file_write(/\.(Loki|Rainman)$/),
27        cuckoo.filesystem.file_write(/\\Restore-My-Files\.txt$/),
28        cuckoo.filesystem.file_write(/\\Cpriv\.Loki$/),
29        cuckoo.network.http_request(/^http:\/\/loki-locker\.one\/index\.php$/)
30      ]
31  }
32
33  rule lokilocker_known_named_objects
34  {
35    condition:
36      cuckoo.sync.mutex(/LokiLocker/)
37  }
```

Výpis 4.5: Všetky druhy pravidiel pre ransomvér LokiLocker

Počas svojho vykonávania ransomvér taktiež vytvára pomenované systémové objekty s menom `LokiLocker` (r. 36), čo je dostatočne unikátne, aby bolo na tento objekt vytvorené pravidlo s pomenovanými objektami. Statické pravidlo obsahuje sekvencie zo správy o výkupnom alebo maily, ktoré má obeť kontaktovať. Nachádza sa tu aj časť verejného kľúča útočníka, ktorá sa používa na šifrovanie kryptografického kľúča generovaného pre každú obeť.

Pravidlá v ukážke 4.5 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 11. marca 2022.

Adhubllka

Adhubllka[11] je ransomvér šíriaci sa v mnohých variantách od júla 2020. Útočníci ponúkajú dešifrovanie jedného súboru zadarmo, aby dokázali, že súbory môžu byť privedené do originálnej podoby. Od obetí vyžadujú výkupné v podobe Bitcoinu cez anonymizovanú sieť Tor. Adhubllka je častokrát zachytená v rámci iného .NET programu, ktorý pri spustení tento ransomvér *odbalí* a spustí na počítači obete. Zašifrovaným súborom rôzne varianty pridávajú rôzne prípony, napríklad `.ADHUBLLKA`, `.MME`, `.DED` alebo `MRV` (r. 35, 38, 42) a každá varianta vytvára svoj špecifický súbor so správou o výkupnom: `Read_Me.txt`, `read_me.txt` alebo `ReadMe.txt` (r. 36, 39, 43).

```
1   include "includes/all.yar"
2   import "cuckoo"
3
4   rule adhubllka_known_sequences
5   {
6     strings:
7       // cast spravy o vykupnom
8       $s01 = "The only method of recovering files is to purchase an unique decryptor
            . Only we can give you this decryptor and only we can recover your files."
9       // komunikacny kanal pre komunikaciu s utocnikom
10      $s02 = "https://yip.su/2QstD5"
11      // konstanta pre sifrovanie pomocou Salsa20
12      $s03 = "expand 32-byte k"
13      $s04 = "W3CRYPTO LOCKER"
14      // spravy o vykupnom
```

[11] https://malpedia.caad.fkie.fraunhofer.de/details/win.adhubllka

```
    $s05 = "Read_Me.txt" wide
    $s06 = "ReadMe.txt" wide
    // pripony zasifrovanych suborov
    $s07 = ".MME" wide
    $s08 = ".MRV" wide
    $s09 = ".DED" wide
    // prikazy
    $s10 = "%ls\\%ls" wide
    $s11 = "Win32_ShadowCopy.ID='%s'" wide
    $s12 = "select * from Win32_ShadowCopy" wide
  condition:
    EXE and 7 of them
}

rule adhubllka_known_behavior_high
{
  condition:
    (
      // DED: 3cd2089cebca0e60e102171d986f897c55ff74d27b3c77fdc602d15c8d966e82
      // MME: ecd35d406bb556684bfb4cbe22b4cbfa047233e932376beffad063f8037a5124
      cuckoo.filesystem.file_write(/\.(MME|DED)$/) and
      cuckoo.filesystem.file_write(/\\Read_Me\.txt$/)
    ) or (
      cuckoo.filesystem.file_write(/\.ADHUBLLKA$/) and
      cuckoo.filesystem.file_write(/\\read_me\.txt$/)
    ) or (
      // MRV: 92138259847fff60c9e7091e535ea996eceaa6d0b2aace9a57370c1f1b27b4be
      cuckoo.filesystem.file_write(/\.MRV$/) and
      cuckoo.filesystem.file_write(/\\ReadMe\.txt$/)
    )
}
```

Výpis 4.6: Statické a behaviorálne pravidlo pre ransomvér Adhubllka

Statické pravidlo je aktivované ak sa jedná o spustiteľný súbor a obsahuje aspoň 7 z uvedených sekvencií, ako napr. útržok správy o výkupnom, odkaz na stránku komunikácie s útočníkmi, mená súborov správ o výkupnom, prípony zašifrovaných súborov alebo príkazy na vymazanie systémových záloh.

Pravidlá v ukážke 4.6 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 14. februára 2022.

Babuk

Ransomvér Babuk[12] sa šíri už od začiatku roku 2021. Ide o multiplatformový ransomvér na Windows aj Linux (uvedená analýza sa sústredí na variantu pre Windows) vytvorený skupinov hackerov z Ruska, ktorý sa po krátkej dobe zapísal medzi väčších hráčov. Zdrojový kód tohto ransomvéru bol zverejnený jedným z členov skupiny zodpovednou za Babuk[13].

Babuk používa na generovanie kryptografických kľúčov algoritmus eliptických kriviek a na šifrovanie súborov symetrický algoritmus ChaCha. Analýze ransomvéru Babuk boli podrobené tri varianty pridávajúce zašifrovaným súborom prípony `.babyk`, `.blaze` alebo

[12] https://malpedia.caad.fkie.fraunhofer.de/details/win.babuk

[13] https://twitter.com/vxunderground/status/1433758742244478982?s=20&t=FJXQvVtgsdK4TRpeMvYFuw

`.__NIST_K571__` (r. 45, 46, 47). Tieto varianty pri spúšťaní tvorili pomenované systémové objekty uvedené na riadkoch 8 a 9 a vymazávali systémové zálohy (r. 52). Na konci vytvorili súbor so správou o výkupnom (r. 49, 50). V niektorých prípadoch, kedy obeť nezaplatí výkupné, sú jej súbory exfiltrované (útočníci zverejnia dáta obete) na stránky dostupné cez anonymizované siete.

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule babuk_known_named_objects
5  {
6    condition:
7      ...
8      cuckoo.sync.mutex(/(\\|^)DoYouWantToHaveSexWithCuongDong$/) or
9      cuckoo.sync.mutex(/(\\|^)BlazeOne$/)
10 }
11
12 rule babuk_known_sequences
13 {
14   strings:
15     ...
16     // nazov systemoveho objektu
17     $s01 = "DoYouWantToHaveSexWithCuongDong"
18     // programy, ktore Babuk vypina pred sifrovanim
19     $s02 = "zhudongfangyu"
20     $s03 = "sophos"
21     $s04 = "BackupExecVSSProvider"
22     $s05 = "Tor Browser" wide
23     $s06 = "Windows.old" wide
24     // cast spravy o vykupnom
25     $s07 = "How To Restore Your Files.txt" wide
26     $s08 = "****BY BABUK LOCKER****"
27     $s09 = "http://babukq4e2p4wu4iq.onion"
28     // nastavenie
29     $s10 = "-lanfirst"
30     // prikaz na vymazanie systemovych zaloh
31     $s11 = "vssadmin.exe delete shadows /all /quiet" wide
32
33     // pripony zasifrovanych suborov
34     $f01 = ".babyk" wide
35     $f02 = ".blaze" wide
36     $f03 = ".__NIST_K571__" wide
37   condition:
38     EXE and any of ($f*) and 7 of ($s*)
39 }
40
41 rule babuk_known_behavior_high
42 {
43   condition:
44     (
45       cuckoo.filesystem.file_write(/\.babyk$/) or
46       cuckoo.filesystem.file_write(/\.__NIST_K571__$/) or
47       cuckoo.filesystem.file_write(/\.blaze$/)
48     ) and (
```

```
49        cuckoo.filesystem.file_write(/\\How To Restore Your Files\.txt$/) or
50        cuckoo.filesystem.file_write(/\\How To Decrypt\.txt/)
51      ) and
52      cuckoo.process.executed_command(/VSSADMIN\.EXE DELETE SHADOWS \/ALL \/QUIET$/i
          )
53  }
```

Výpis 4.7: Všetky druhy pravidiel pre ransomvér Babuk

Pravidlo pre pomenované objekty a statické pravidlo už boli napísané a autor práce ich len na základe nových variánt aktualizoval[14]. Pre časť pravidla, ktorú autor práce nenapísal je použitý znak troch bodiek. Statické pravidlo využíva v sekvenciach začínajúcich na `$s` mená programov, ktoré sa snaží ransomvér vypnúť, správu o výkupnom, použitý príkaz na vymazanie záloh systému a iné reťazce obsiahnuté v rámci kódu tohto ransomvéru. Sekvencie začínajúce na `$f` obsahujú mená prípon, ktoré rôzne varianty používajú. Pravidlo je aktivované ak je vzorka spustiteľný súbor, obsahuje aspoň jednu z daných prípon a aspoň 7 sekvencií z analyzovaných vzoriek.

Pravidlá v ukážke 4.7 vytvorené na základe predchádzajúceho popisu ransomvéru boli prvýkrát aktualizované 15. februára 2022.

BlackCat

BlackCat[15] bol prvýkrát spozorovaný koncom roku 2021. Jedná sa o jeden z novších ransomvérov – ktorého šírenie výrazne stúpa – napísaný v programovacom jazyku Rust, ktorý prichádza s rôznymi možnosťami spustenia pomocou jednotlivých prepínačov. Prepínače mu umožňujú napr. zapisovať logy do súboru (pomocou `-log-file <LOG_FILE>`), voľbu aktualizácie pozadia pracovnej plochy po ukončení šifrovania (pomocou `-no-wall`), ukázať užívateľské rozhranie (pomocou `-ui`) a ďalšie iné. BlackCat sa šíri v podobe RaaS (Ransomware-as-a-Service, autori ransomvéru predávajú BlackCat ďalej svojim klientom, ktorí ho využívajú) a zameriava sa skôr na väčšie firmy. BlackCat v sebe ukrýva konfiguráciu ransomvéru v tvare JSON, kde sa nachádzajú rôzne užitočné informácie o danej vzorke, ako napríklad aj algoritmus na šifrovanie súborov (AES/Chacha), prípona zašifrovaných súborov, názov súboru so správou o výkupnom a podobne.

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule blackcat_known_sequences
5  {
6    strings:
7      // pripony zasifrovanych suborov pri znamych variantach
8      $s00 = "7954i9r" ascii
9      $s01 = "dkrpx75" ascii
10     $s02 = "3gtksio" ascii
11     $s03 = "wpzlbji" ascii
12     $s04 = "o3bifnw" ascii
13     $s05 = "mfqssdj" ascii
14     $s06 = "gsl0fnn" ascii
15     $s07 = "kh1ftzx" ascii
```

[14] Pravidlá `babuk_known_sequences` a `babuk_known_named_objects` vytvorili malvéroví analytici firmy Avast

[15] https://malpedia.caad.fkie.fraunhofer.de/details/win.blackcat

```
16      $s08 = "sykffle" ascii
17      $s09 = "tx1mdf6" ascii
18      $s010 = "nnvjxgy" ascii
19      $s011 = "dkrpx75" ascii
20      // meno suboru so spravou o vykupnom
21      $s012 = "RECOVER-${EXTENSION}-FILES.txt" ascii
22      // cast spravy o vykupnom
23      $s013 = "Important files on your system was ENCRYPTED and now they have have
            \\\"${EXTENSION}\\\" extension." ascii
24
25      $s100 = "locker::core::" ascii
26      // kluce z JSON konfiguracie
27      $s110 = "\"config_id\"" ascii
28      $s111 = "\"note_file_name\"" ascii
29      $s112 = "\"note_full_text\"" ascii
30      $s113 = "\"enable_esxi_vm_snapshot_kill\"" ascii
31    condition:
32      EXE and(
33        2 of ($s0*) or
34        (#s100 > 10 and all of ($s11*))
35      )
36  }
37
38  rule blackcat_known_behavior_high
39  {
40    condition:
41      (
42        // pridavanie pripon zasifrovanym suborom
43        cuckoo.filesystem.file_write(/\.7954i9r$/) or
44        cuckoo.filesystem.file_write(/\.dkrpx75$/) or
45        cuckoo.filesystem.file_write(/\.3gtksio$/) or
46        cuckoo.filesystem.file_write(/\.wpzlbji$/) or
47        cuckoo.filesystem.file_write(/\.o3bifnw$/) or
48        cuckoo.filesystem.file_write(/\.xxxxxxx$/) or
49        cuckoo.filesystem.file_write(/\.mfqssdj$/) or
50        cuckoo.filesystem.file_write(/\.gsl0fnn$/) or
51        cuckoo.filesystem.file_write(/\.kh1ftzx$/) or
52        cuckoo.filesystem.file_write(/\.sykffle$/) or
53        cuckoo.filesystem.file_write(/\.tx1mdf6$/) or
54        cuckoo.filesystem.file_write(/\.nnvjxgy$/) or
55        cuckoo.filesystem.file_write(/\.dkrpx75$/)
56      ) and (
57        // subor so spravou o vykupnom
58        cuckoo.filesystem.file_write(/\\RECOVER-[a-zA-Z0-9]{1,8}-FILES\.txt$/) or
59        cuckoo.filesystem.file_write(/\\recover-[a-zA-Z0-9]{1,8}-files\.txt\.png$/)
60      )
61  }
```

Výpis 4.8: Statické a behaviorálne pravidlo pre ransomvér BlackCat

Statické pravidlo, ktoré autor práce vytvoril využíva prípon zašifrovaných súborov pri známych variantách, vstavanej JSON konfigurácie a reťazca `$s100`, ktorý sa v nájdených variantách BlackCat ransomvéru vyskytoval aspoň desaťkrát, a preto bol pridaný na spresnenie klasifikácie. Toto pravidlo je aktivované pre spustiteľné súbory ak v sebe obsahujú

aspoň dve sekvencie týkajúce sa pridaných prípon a vytvorených súborov alebo všetky sekvencie z konfigurácie a aspoň desať inštancií sekvencie `$s100`.

Vytvorené behaviorálne pravidlo popisuje všetky známe prípony využívané BlackCat ransomvérom a takisto regulárny výraz pre všetky mená správ o výkupnom spolu s obrázkami, ktoré sú nastavené na pracovnej ploche. Regulárny výraz popisuje súbory:

- začínajúce sa na `RECOVER-` nasledované jedným až ôsmimi alfanumerickými znakmi, ktoré končia s `-FILES.txt` (jedná sa o správy o výkupnom, r. 58) a súbory
- začínajúce sa na `recover-` nasledované jedným až ôsmimi alfanumerickými znakmi, ktoré končia na `-files.txt.png` (jedná sa o obrázky nastavené ako pozadie pracovnej plochy, r. 59).

Ak program pri svojom spustení pridá jednu z týchto prípon nejakému súboru a zároveň vytvorí správu o výkupnom alebo obrázok s vyššie uvedeným menom, tak sa pravidlo aktivuje a program je pomocou neho klasifikovaný ako BlackCat ransomvér.

Pravidlá v ukážke 4.8 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 2. februára 2022.

Prometheus

Vo februári 2021 bol prvýkrát spozorovaný Prometheus[16] ransomvér, ktorý značne čerpá z kódu už pred tým známeho ransomvéru Hakbit[17]. Prometheus sa šíri v zapackovanej verzii a obsahuje rôzne techniky na sťaženie jeho debuggovania. Na generovanie kryptografického kľúča je použitý generátor pseudo-náhodných čísel a šifrovanie súborov prebieha potom s pomocou symetrického algoritmu Salsa20. Spôsob šifrovania súborov sa podarilo prelomiť a viac o postupe ako Prometheus využíva kryptografiu bude napísané ďalej. Pravidlá pre túto rodinu sú príliš dlhé, a preto budú uložené v sekcii s prílohami v prílohe B.

Obr. 4.1: Správa o výkupnom rodiny Prometheus

[16] https://malpedia.caad.fkie.fraunhofer.de/details/win.prometheus
[17] https://malpedia.caad.fkie.fraunhofer.de/details/win.hakbit

Analýza tohto ransomvéru prebiehala na variante, ktorá pridáva zašifrovaným súborom príponu `.unlock` (r. 104, neskôr boli pridané aj ďalšie varianty). Tá obsahovala sekvencie pre vypisovanie debuggovacích správ, príponu a meno súboru správy o výkupnom. Keďže je ransomvér napísaný v .NET frameworku, statické pravidlo v sebe zahŕňa aj identifikátor GUID jedinečný pre túto variantu. Unpackovaná verzia obsahovala sekvencie zo statického pravidla, no Prometheus sa šíri výhradne zapackovaný a tým pádom naň vo väčšine prípadov statické pravidlo nebude stačiť a treba aj iné druhy.

Behaviorálne pravidlo v sebe obsahuje prípony zašifrovaných súborov a mená správ o výkupnom rôznych variant. Spustenie pravidla nastáva ak vzorka zapíše do takýchto súborov: ako do súboru so známou príponou Prometheus ransomvéra, tak aj do súboru so správou o výkupnom.

Rôzne varianty generujú systémové objekty s rôznymi menami. Pravidlo `prometheus_known_named_objects` obsahuje mená týchto systémových objektov a je aktivované pri tvorbe ľubovoľného z nich.

Kvôli prelomeniu kryptografie tejto rodiny autor práce hľadal čo najviac jej vzoriek, aby som dokázal zistiť, či všetky používajú kryptografiu rovnako. Šifrovanie bolo až na pár výnimiek rovnaké, no vzorky sa líšili hlavne v tom, či generovali jeden kľúč pre všetky súbory, alebo pre každý súbor nový kľúč (informácia o spôsobe šifrovania bola uložená v konfigurácií vzorky v binárnom súbore ransomvéru). Tie vzorky, ktoré generovali len jeden kľúč boli považované za dekryptovateľné. Viac v sekcii 4.2.

Pravidlá v ukážke B.1 vytvorené na základe predchádzajúceho popisu ransomvéru boli nasadené do prevádzky od 22. februára 2022.

4.2 Spôsoby šifrovania analyzovaných rodín

V kapitole 3.3 boli popísané dva najčastejšie druhy šifrovania, ktoré ransomvér používa – symetrické a asymetrické. Rôzne rodiny sa však líšili napríklad v spôsobe ich používania, výberom algoritmov pre ich uskutočnenie alebo aj stanovením častí súborov, ktoré budú zašifrované (celý súbor, jeho časť, hlavička, atď) v rámci optimalizácie.

Počas analýzy vybraných rodín ransomvérov bolo objavených niekoľko spôsobov ako útočník môže efektívne odoprieť obeti prístup k súborom, no niektoré postupy dovolovali aj vytvorenie programu, ktorý by schému kryptografie zlomil a zašifrované súbory vrátil do pôvodnej podoby bez vyplatenia výkupného (takýto program sa nazýva dekryptor). Medzi najčastejšie prípady, kedy bolo možné vytvoriť dekryptor patria tie, kedy ransomvér používa konštantný kľúč uvedený v jeho binárnom súbore alebo generuje slabý kryptografický kľúč (napr. generovanie 32-bitového kľúča pri používaní 128-bitového algoritmu AES). Slabý kľúč sa potom v niektorých prípadoch pomocou brute-force útoku, teda generovania všetkých možných hodnôt kľúča, dal zistiť a použiť na dešifrovanie súborov. Pri použití silného kryptografického kľúča brute-force nebol možný, pretože generovanie všetkých možných hodnôt kľúča by trvalo veľmi dlhú dobu.

V tejto kapitole budú zhrnuté tri najčastejšie spôsoby šifrovania súborov ransomvérom, ktoré autor práce počas analýzy identifikoval – ich silné a slabé stránky a možnosť ich zlomenia.

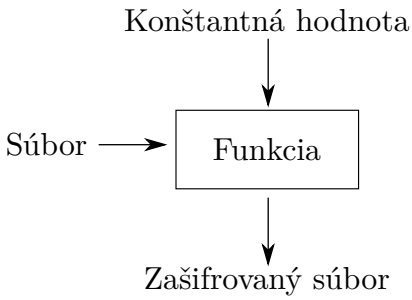
Šifrovanie súboru pomocou funkcie a konštantnej hodnoty

Najjednoduchší z analyzovaných rodín ransomvéru – TimeTime – obsahoval jeden z veľmi primitívnych spôsobov ako zašifrovať súbor. Išlo o použitie matematickej funkcie (v tomto

prípade sčítanie, ale iné rodiny ransomvéru môžu používať aj iné funkcie, napr. XOR) na každý byte súboru v kombinácií s konštantnou hodnotou (uvedené na obr. 4.2). Hodnoty použité vo funkcii sa ľahko pomocou dekompilácie dali zistiť a súbor prinavrátiť do pôvodného stavu.

Použité funkcie môžu byť reverzibilné, teda šifrovanie aj dešifrovanie prebieha tým istým spôsobom (napr. XOR), alebo nereverzibilné, kedy šifrovanie prebieha jednou funkciou a dešifrovanie funkciou k nej opačnou (sčítanie a odčítanie).

Vzorky ransomvéru TimeTime pridávali ku každému bytu súboru hodnotu 1. Takéto súbory sú dešifrovateľné – potrebné je hodnotu 1 naspäť odčítať. Autor bol tým pádom schopný vytvoriť dekryptor, ktorý na vstupe berie zašifrovaný súbor a na výstupe ho bez ďalšej znalosti dešifruje do originálnej podoby.



Obr. 4.2: Primitívne šifrovanie

Generovanie náhodného kľúča a symetrické šifrovanie súborov

Jeden z obvykle používaných spôsobov ransomvérov ako šifrovať súbory je vytvorenie kľúča a následné použitie symetrického šifrovania na jednotlivé súbory (symetrické šifrovanie je rýchlejšie ako asymetrické) ako je uvedené na obr. 4.3. Ako príklad analyzovaných rodín, ktoré toto schéma používali bol HiddenTear a Prometheus.

Prometheus aj HiddenTear obsahovali generátor pseudo-náhodných čísel jazyka C# z triedy `Random`. Takýto generátor používa seed (číslo použité na inicializáciu generátora) v rozsahu 32-bitového celého čísla (tento generátor náhodných čísel pre seed používa hodnotu milisekúnd od štartu počítača). Ransomvér inicializuje generátor pomocou seedu a nechá si vytvoriť kľúč potrebnej dĺžky (napr. 256 bitov alebo 32 bytov) pomocou svojho algoritmu (väčšinou išlo o generovanie náhodných bytov). Kľúč je potom použitý pri symetrickom šifrovaní súboru. Útočník síce vytvára kľúč s požadovanou dĺžkou, no jeho hodnota závisí na 32-bitovom čísle, ktoré pomocou brute-force útoku môže byť nájdené v rozumnom čase. Takéto varianty ransomvéru (v tomto prípade varianta `.NYANCAT` rodiny HiddenTear a veľká časť variánt rodiny Prometheus) sú tým pádom dekryptovateľné.

Pre varianty rodiny HiddenTear bolo možné v dôsledku ich analýzy zdokonaliť dekryptor firmy Avast. Ten na vstupe očakáva adresár so zašifrovanými súbormi a vzorový súbor v originálnej a v zašifrovanej forme.

Varianta rodiny HiddenTear pridávajúca príponu `.NYANCAT` používa slabý generátor náhodných čísel, ktorý je závislý na 32-bitovej hodnote. Inicializuje generátor a nechá si vytvoriť 32 indexov do reťazca znakov, z čoho vznikne heslo. Na toto heslo je tiež aplikované hashovanie vo forme SHA256 algoritmu. Súbory sú vytvoreným hashom zašifrované cez symetrické šifrovanie AES. Dekryptor využíva brute-force útok – poskytuje postupne všetky 32-bitové hodnoty a s nimi inicializuje generátor náhodných čísel, vytvorí kľúč, zašifruje

vzorový originálny súbor a porovná ho s vzorovým zašifrovaným súborom. Ak dôjde k zhode, kľúč bol nájdený a dekryptor s ním dešifruje aj ostatné súbory. Proces hľadania kľúču môže kľúč nájsť ihneď (ak prvý vytvorený kľúč je rovnaký ako ten použitý) alebo až do niekoľkých hodín.

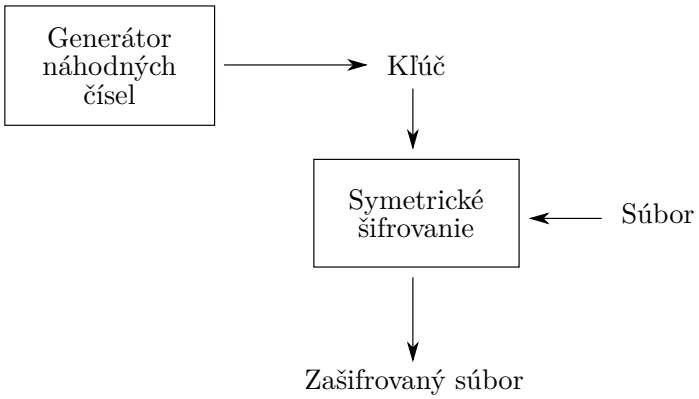
Iné varianta rodiny HiddenTear, ktorá pridáva súborom príponu `.PPLIT` obsahuje konštantné heslo `1524312qWPo`, z ktorého vytvorí SHA256 hash a použije ho ako kľúč v rámci symetrického šifrovania pomocou AES algoritmu. Originálny vzorový súbor je týmto spôsobom zašifrovaný a porovnaný s vzorovým zašifrovaným súborom. Ak sa oba zhodujú, znamená to, že súbor bol naozaj zašifrovaný daným ransomvérom a prichádza k dešifrovaniu ostatných súborov uvedených na vstupe. Pri tejto variante nie je nutné používať brute-force, pretože heslo je dopredu známe.

Keďže rodina ransomvéru Prometheus vznikla relatívne nedávno, Avast v čase písania tejto práce ešte nedisponoval dekryptorom na túto rodinu, a preto ho autor práce mohol vytvoriť od začiatku. Prometheus používa podobné schéma kryptografie ako HiddenTear – inicializuje generátor pseudo-náhodných čísel v triede `Random` jazyka C# pomocou 32-bitového seedu a nechá vygenerovať 32 ASCII znakov. Toto heslo je použité na symetrické šifrovanie pomocou algoritmu Salsa20. Líši sa však vo výbere časti súboru, ktoré budú dešifrované. Prometheus stanoví veľkosť súboru T (vo väčšine prípadov ide o 10 MiB), súbory menšie ako tento prah sú šifrované ako celok, súbory väčšie alebo rovné sú šifrované iným spôsobom. Naprieč variantami ide o:

- šifrovanie prvých T MiB,
- rozdelenie súboru na tretiny a šifrovanie len časti z každej tretiny alebo
- rozdelenie súboru na polovice a šifrovanie len časti z každej polovice.

Na koniec zašifrovaných súborov, ktorých veľkosť je väčšia alebo rovná ako T MiB je pridaný reťazec znakov `Blocks-T-` alebo `Thanos-T-`.

Dekryptor potom spúšťa proces hľadania kľuču pomocou brute-force, ktorý vytvára všetky možné kľúče použité na šifrovanie. Tieto kľúče sú aplikované na vzorový origínalny súbor a ak sa zhoduje s vzorovým zašifrovaným súborom, kľúč je nájdený a dekryptor pokračuje v dešifrovaní ostatných súborov, ktoré majú príponu danej varianty ransomvéru.



Obr. 4.3: Generovanie náhodného kľúča a symetrické šifrovanie súborov

Asymetrické šifrovanie náhodného kľúča a symetrické šifrovanie súborov

Jeden zo zaužívaných a overených postupov, ktoré používali rodiny ransomvéru ako Babuk, LokiLocker, Khonsari a niektoré varianty rodiny Prometheus, bolo použitie kombinácie symetrického a asymetrického šifrovania, ako je ukázané na obr. 4.4. Súbory boli hlavne kvôli rýchlosti šifrované symetricky, zatiaľ čo kľúče asymetricky.

Schéma pozostáva z troch typov kľúčov, ktoré budú v tejto práci nazývané:

- master kľúč
- session kľúč
- file kľúč

Autori týchto ransomvérov ako prvé vygenerovali asymetricky pár master kľúčov (súkromný a verejný master kľúč). Súkromný master kľúč bol uchovaný na mieste, kam sa dostane len útočník (nie je uložený na počítači obete) a verejný master kľúč bol pre obeť voľne dostupný – napríklad stiahnuteľný z internetu alebo uložený priamo v binárnom súbore ransomvéru.

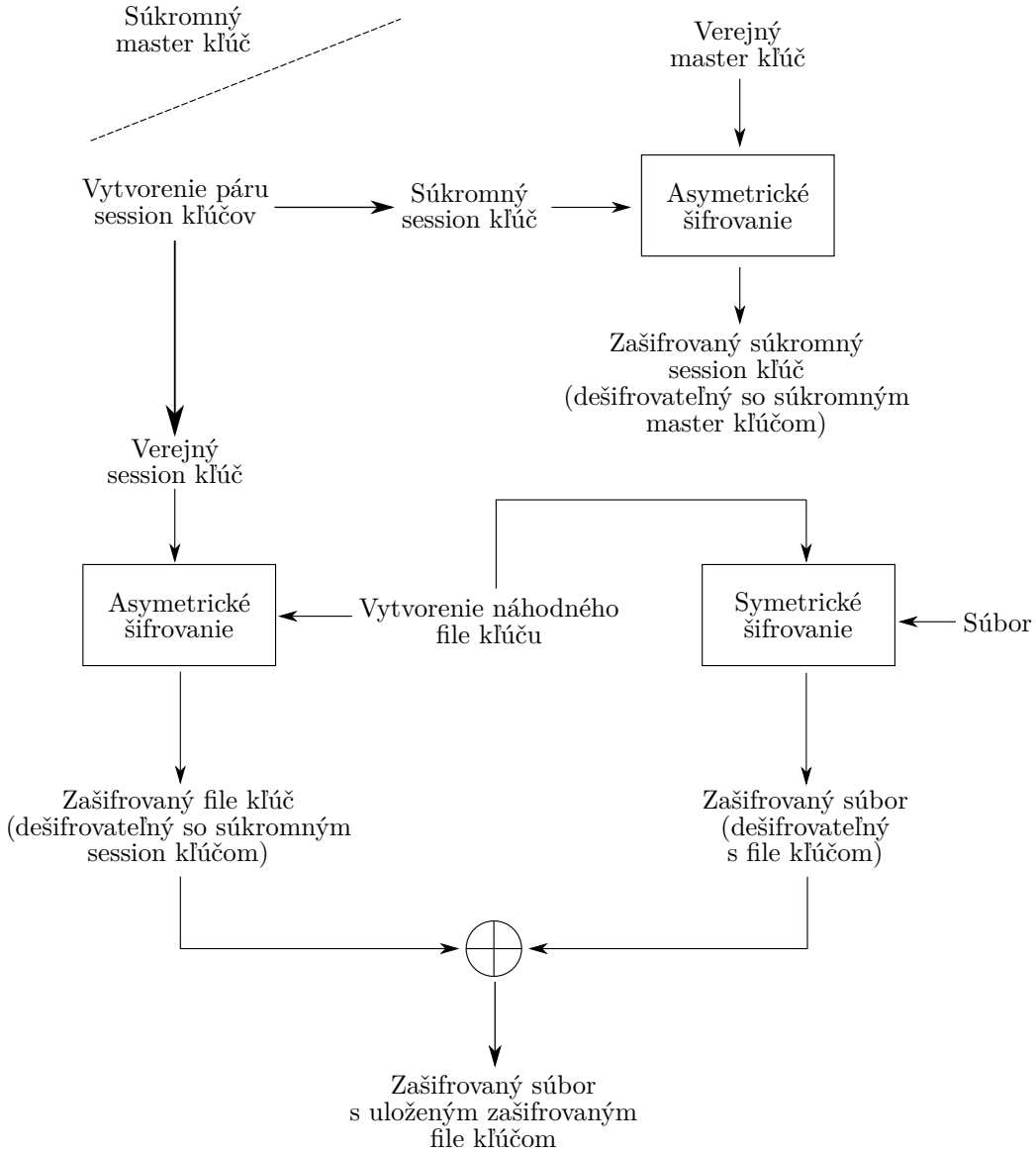
Rodiny, ktoré využívali toto schéma kryptografie pri napadnutí vytvorili nový pár asymetrických session kľúčov (takisto súkromný a verejný) unikátny pre každú obeť a pre každý súbor symetrický file kľúč (ten môže byť vytvorený napríklad generátorom náhodných čísel).

Každý súbor bol symetricky šifrovaný náhodným file kľúčom, zatiaľ čo file kľúč sa ďalej asymetricky šifroval pomocou verejného session kľúča. Jediný spôsob ako dešifrovať súbor je pomocou file kľúča a ten sa dá dešifrovať len pomocou súkromného session kľúča. Zašifrovaný file kľúč bol uložený v rámci zašifrovaného súboru na jeho začiatku alebo konci (záviselo od danej rodiny a varianty).

Tento postup prebiehal pre každý súbor zvlášť. Teda znalosť súkromného session kľúča je nevyhnutná na dešifrovanie každého súboru. Rodiny, ktoré toto schéma využívali však tento súkromný session kľúč zašifrovali verejným master kľúčom a uložili na počítači obete, čo vo výsledku robí útočníka (alebo toho, kto vlastní súkromný master kľúč) jediným, kto môže všetky súbory dešifrovať.

V niektorých prípadoch sa stáva, že skupiny zodpovedné za ransomvér na konci svojho prevádzania vydajú súkromné master kľúče, ktorými sú všetky obete schopné prinavrátiť súbory do pôvodného stavu. Táto situácia však nenastáva vždy a závisí len na skupine, či je ochotná kľúče poskytnúť.

Ak by sa aj pri tejto metóde použil algoritmus, ktorý vytvára slabé file kľúče (dali by sa pomocou procesu brute-force zistiť v rozumný čas), brute-force pre všetky súbory by s veľkou pravdepodobnosťou zabral veľmi dlhú dobu. A práve táto skutočnosť robí z uvedenej schémy kryptografie jednu z najefektívnejších metód pre autorov ransomvéru a súbory sa pri analyzovaných rodinách nedali dešifrovať. Jediné čo ostáva obetiam je čakanie, či útočníci zverejnia súkromné master kľúče. V prípadoch analyzovaných ransomvérov v čase písania tejto práce neboli vydané súkromné master kľúče.



Obr. 4.4: Asymetrické šifrovanie kľúča a symetrické šifrovanie súborov

Kapitola 5

Experimentálne výsledky

Po prevedení analýzy na 10 rôznych rodinách ransomvéru autor práce vytvoril dokopy 24 detekčných YARA pravidiel, dodal podporu pre dekryptor od firmy Avast na rodiny HiddenTear pre ďalšie dve nové varianty a od začiatku vytvoril dekryptor pre rodinu Prometheus. V tejto kapitole bude ukázané akú efektivitu mali detekčné pravidlá a ako prebiehalo testovanie dekryptorov.

5.1 Detekcie malvéru pomocou YARA pravidiel

YARA pravidlá boli umiestnené do systémov firmy Avast tak, aby novo-prichádzajúce vzorky mohli byť na základe týchto pravidiel klasifikované. Tie prechádzajú cez dva druhy systémov. Najprv sú podrobené statickým pravidlám, potom behaviorálnym. V dôsledku správnej klasifikácie vzorky sa zlepšuje úroveň detekcií malvéru u klientov. Teda jedna dobrá klasifikácia môže ovplyvniť detekcie aj na niekoľkonásobne väčšom počte endpointov.

Počet prichádzajúcich vzoriek ransomvéru v období od 23.1.2022 (nasadenie prvého pravidla) do 29.4.2022 bol približne 870 tisíc. Teda pravidlá napísané autorom museli byť aplikované aspoň nad týmto počtom vzoriek (boli aplikované aj nad inými typmi malvéru).

Analýza rodín mala za cieľ nájsť charakteristické vlastnosti vzoriek tak, aby detekčné pravidlo dokázalo zachytiť rovnaké vzorky bez tzv. false-positives (vzorky, ktoré nie sú malvér a nie sú danej rodiny) a v najlepšom prípade, aby zachytávalo aj nové varianty – čo nie vždy je možné, pretože nemusí byť k dispozícií dostatočné množstvo informácií, známych variant alebo autori ransomvéru natoľko menia svoj kód, že nové varianty sa doposiaľ napísaným pravidlám po čase začnú vyhýbať.

V tabuľke 5.1 môžeme vidieť jednotlivé počty *hitov* (za hit sa považuje jedna klasifikácia, teda vzorka prišla do jedného zo systémov, kde bola klasifikovaná – nehovorí o detekciach u klientov) daných pravidiel spolu s dátumom, od kedy pravidlá boli v prevádzke (ak išlo o vylepšenie pravidla, dátum hovorí o dni pridania vylepšenia). Každá rodina môže mať až tri druhy pravidiel umiestnené v jednotlivých stĺpcoch. Ak niektorý typ pravidla nebol pre rodinu vytvorený, je v tabuľke vyznačený pomlčkou. Pravidlá uvedené v zátvorke neboli písané od začiatku.

Ako prvé sa dá z tabuľky vidieť, že najväčší úspech mali pravidlá pre rodiny BlackCat, Babuk a HiddenTear. BlackCat v každej vzorke obsahovala konfiguráciu vo formáte JSON, čo predstavovalo spôsob ako ju ľahko detekovať. Zo zachytených vzoriek sa potom dali zistiť prípony, ktoré pridávajú zašifrovaným súborom, čo zase zlepšilo behaviorálne pravidlo.

Rodina	Dátum nasadenia	Statické	Behaviorálne	Známe objekty	FP [%]
Khonsari	23.1.2022	2	0	–	0
BlackCat	2.2.2022	104	21	–	0
HiddenTear	2.2.2022	(666)	(238)	–	0
Adhubllka	14.2.2022	65	23	–	0
Babuk	15.2.2022	(139)	85	(135)	0
Redeemer	17.2.2022	32	4	14	0
Prometheus	22.2.2022	0	51	65	0
TimeTime	2.3.2022	4	4	–	0
LokiLocker	11.3.2022	92	17	25	0
Nokoyawa	14.3.2022	8	4	–	0

Tabuľka 5.1: Tabuľka klasifikácií (aktualizovaná dňa 29. apríla 2022)

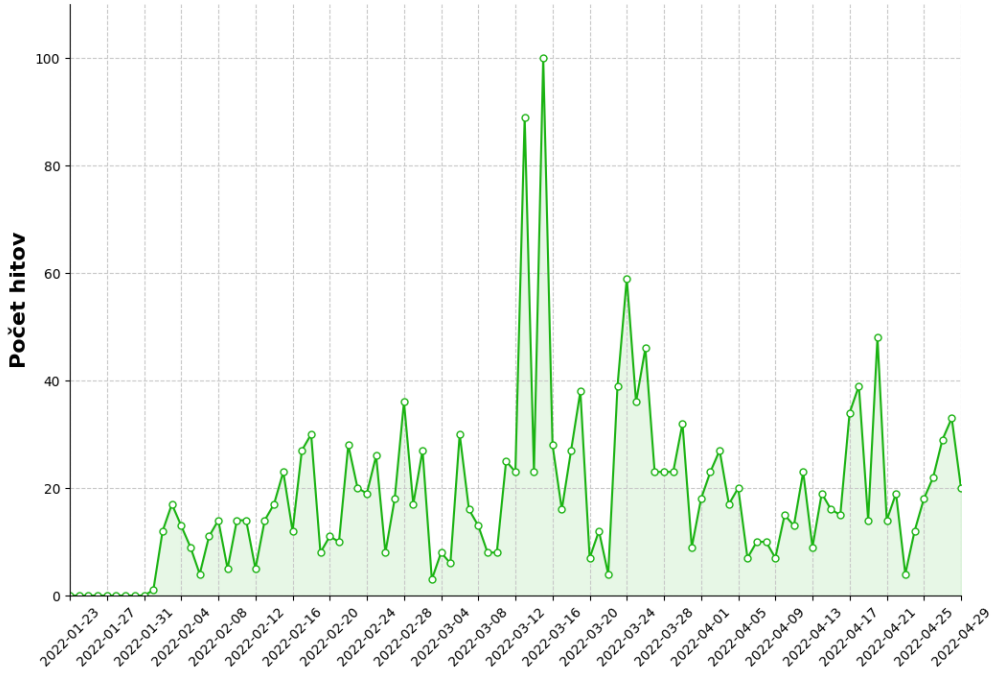
Pravidlá pre HiddenTear dosiahli obzvlášť veľkého počtu hitov. Je to hlavne z toho dôvodu, že sa jedná o rodinu, ktorá už pôsobí dlhší čas a existujúce pravidlá zachytávali veľké počty vzoriek. Počas analýzy však autor mohol toto pravidlo vylepšiť a k popísaným vzorkám pridať ďalšie štyri. Po vylepšení dokážu pravidlá správne klasifikovať nové varianty tejto rodiny vrátane tých, o ktorých autor zistil, že sú dešifrovateľné (varianty NYANCAT a PPLIT). Ak obeť pozná rodinu a ide o dešifrovateľnú variantu ransomvéru, môže využiť autorom vytvorený dekryptor a zašifrované súbory získať späť.

Medzi relatívne nové rodiny patrí Nokoyawa. Počet spozorovaných variánt tejto rodiny je zatiaľ celkom malý a tým pádom je obtiažne vytvoriť pravidlá, ktoré by zachytávali nové, ešte nevidené vzorky. Ak sa časom nájdu nové varianty, efektivita týchto pravidiel môže byť významne zvýšená.

Pravidlá s malým počtom hitov môžu tiež zodpovedať aj za jednorázové projekty, kedy ransomvér vznikne pri nejakej príležitosti a ďalej nie je vyvíjaný. Ide napríklad o Khonsari, ktorý sa prvýkrát objavil pri odhalení zraniteľnosti v dôležitom softvéri, no postupom času neboli spozorované žiadne ďalšie výskyty tohto ransomvéru. Nulový počet hitov pri behaviorálnom pravidle je v dôsledku toho, že kontrolný server, na ktorý sa Khonsari po štarte pripája, bol krátko po detekovaní tohto ransomvéru vypnutý. Ransomvér sa naň nedokáže pripojiť a svoje prevádzanie následne ukončí. Tým pádom na počítači neostávajú žiadne artefakty, ktoré by behaviorálne dokázali Khonsari detekovať. Do tejto kategórie môžeme zaradiť aj TimeTime ransomvér, ktorý sa počas analýzy javil ako 'amatérsky', z čoho sa dá usúdiť, že išlo len o nejaký pokus autora napísať funkčný ransomvér a zaslať ho len malému počtu obetí bez pridania zložitých obfuskačných mechanizmov.

Pravidlá na detekciu rodín ako Prometheus, Adhubllka, Redeemer alebo Lokilocker uspeli ako v detekcii už známych vzoriek, tak aj pri hľadaní nových. Rodina Prometheus prichádzala na systém obete vo výhradne silno obfuskovanej podobe, a preto statické pravidlo (smerujúce na neobfuskované varianty) nenašlo žiadne vzorky. Adhubllka naopak v sebe zahŕňala reťazce znakov, ktoré dopomohli jej efektívnej detekcii. Vyšší počet hitov všeobecne pri statických pravidlách oproti behaviorálnym môže nastať napríklad kvôli lepšiemu statickému pravidlu, útočníci môžu meniť prípony pridávané zašifrovaným súborom čo nemusí ovplyvniť statické pravidlá alebo vzorky pri dynamickej analýze v sandboxe môžu z nejakého dôvodu prerušiť prevádzanie a predčasne skončiť.

V grafe 5.1 sú znázornené počty hitov rozdelené podľa dátumu. Postupom času v dôsledku vytvárania nových YARA pravidiel sa počty klasifikácií zvyšovali. Ide o klasifikácie zo všetkých systémov firmy Avast, teda statické aj behaviorálne hity.



Obr. 5.1: Časový diagram klasifikácií implementovaných YARA pravidiel

Pravidlám sa pri ich vytvorení špecifikuje nízka spoľahlivosť. Vzorky nimi odchytené musia byť po určitom čase skontrolované, či nezachytávajú false-positives (stĺpec *Pomer FP*) a následne môže byť táto spoľahlivosť pravidlu navýšená (Avast ponúka systémy na automatizáciu tohto procesu). Prípady `khonsari_known_behavior_high` a `prometheus_known_sequences` síce nezachytávali žiadne vzorky, no v budúcnosti môžu odhaliť rodiny, ktoré sa opäť aktivovali alebo ich deriváty. Napísané pravidlá s navýšenou spoľahlivosťou môžu lepšie pomáhať s klasifikáciou ransomvéru. Spoľahlivosť bola navýšená 24 YARA pravidlám pre 10 rodín ransomvéru, ktoré dokopy odchytili od 23. januára do 29. apríla 1799 vzoriek ransomvéru.

5.2 Dešifrovanie súborov s dekryptormi

Zraniteľnosť ransomvérov podrobených analýze, ktorých kryptografia sa dala zlomiť spočívala v nespoľahlivom generátore náhodných čísel. Tento generátor závisel na už spomínanom 32-bitovom seede, ktorý predstavoval čas uplynutých milisekúnd od štartu počítača.

Pre správne overenie fungovania dekryptorov bol umiestnený na virtuálny stroj adresár so súbormi, ktoré slúžili ako *návnada*. Ich originálna podoba bola vopred známa a ransomvér ich počas svojho prevádzania zašifroval. Neskôr mohli byť zašifrované súbory dekryptorom dešifrované a porovnané s originálmi.

Obr. 5.2: Ukážka dekryptoru pre rodinu Prometheus

Samotný framework dekryptoru už bol vo firme Avast vytvorený a doň bolo treba zakomponovať modul pre Prometheus a vylepšiť modul pre HiddenTear. Po spustení dekryptoru je užívateľ vyzvaný na poskytnutie vzorového originálneho súboru spolu s týmto istým súborom ale v zašifrovanej podobe a taktiež adresáru (alebo rovno celého disku), ktorý obsahuje súbory na dešifrovanie. Po absolvovaní ďalších krokov sa dekryptor pustí do procesu hľadania kľúča. Dešifrované varianty boli porovnané s originálnymi súbormi pomocou programu `diff` na dokázanie ich zhody.

Dekryptor pre analyzované varianty rodiny HiddenTear a Prometheus vracal 100% uvedených súborov do podoby identickej s tou orignálnou. Dá sa povedať, že dekryptor bol schopný vrátiť ľubovoľný súbor do originálnej podoby s jedinou podmienkou – poskytnutie dvojice súborov originálny súbor a ten istý, ale zašifrovaný súbor.

Vytvorený dekryptor pre rodinu Prometheus je zverejnený na stránkach firmy Avast[1] a taktiež Europolu[2].

[1] Blogpost o dekryptore https://decoded.avast.io/threatresearch/decrypted-prometheus-ransomware/

[2] https://www.nomoreransom.org/en/decryption-tools.html

Kapitola 6

Záver

V práci bolo zhrnuté čo to malvér je a aké techniky používa. Boli vyčlenené aj hlavné typy škodlivého kódu. Pomocou reverzného inžinierstva bolo ukázané, ako prebieha analýza malvéru cez jednotlivé nástroje, ktoré analytici škodlivého kódu používajú. Práca sa najviac sústredila na ransomvér – od jeho počiatkov až po moderné trendy. Boli ukázané techniky ransomvéru a jeho najčastejšie zdroje nákazy spolu s možnou obranou – napr. pravidelné aktualizovanie softvéru alebo overovanie pravosti prichádzajúcich emailov. Taktiež práca informovala o použití kryptografie a výber medzi symetrickým a asymetrickým šifrovaním pri prevádzaní ransomvéru.

Samotná analýza prebiehala na poskytnutých vzorkách z firmy Avast a týkala sa desiatich rodín ransomvéru: Khonsari, BlackCat, HiddenTear, Adhubllka, Babuk, Redeemer, Prometheus, TimeTime, LokiLocker a Nokoyawa. Na každú rodinu bolo implementované aspoň jedno pravidlo, ktoré pomáha na klasifikáciu a následnú detekciu u užívateľov na základe statických sekvencií, behaviorálnych prvkov alebo pomenovaných systémových objektov v danej vzorke. Dokopy sa jednalo o 24 YARA pravidiel nasadených v čase od 23. januára 2022. Za dobu do 29. apríla 2022 pravidlá dokázali medzi novo-prichádzajúcimi vzorkami v Avaste spomedzi približne 870 tisíc vzoriek ransomvéru klasifikovať 1799 vzoriek do správnych rodín. Každá správne klasifikovaná vzorka môže však zasiahnuť veľké množstvo užívateľov – klasifikácie pomáhajú užívateľom Avastu z celého sveta na zlepšenie detekcie a ochrany proti ransomvéru.

Z pohľadu klasifikácie falošne-pozitívnych vzoriek bolo nutné klasifikované vzorky prezrieť a zistiť či sa naozaj jedná o daný malvér. Všetky implementované pravidlá – kontrolované aspoň mesiac po ich nasadení – vykazovali 0% falošne-pozitívnych vzoriek. Niektoré pravidlá síce nezachytávali žiadne nové vzorky, no v budúcnosti sa môže stať, že ransomvér, ktorý popisujú bude znovu aktivovaný.

Poslednou šancou obeti útoku ransomvérom môže byť dekryptor, ak ide o ransomvér, dešifrovateľný bez znalosti kľúča. Kvôli tomuto dôvodu bola počas analýzy taktiež skúmaná schéma kryptografie, ktorú jednotlivé rodiny ransomvéru používajú – spôsob použitia symetrického a asymetrického šifrovania, počet šifrovacích kľúčov a výber algoritmov. Cieľom bolo hľadanie chýb umožňujúce vytvárať dekryptor, ktorý by zašifrované súbory dokázal dostať do pôvodnej podoby. Medzi rodiny, ktoré takéto chyby obsahovali patrili TimeTime, HiddenTear a Prometheus.

Nie veľmi rozšírená rodina TimeTime v súlade so svojou jednoduchosťou obsahovala aj primitívne šifrovacie schéma, na ktoré bolo možné vytvoriť jednoduchý dekryptor. Rodiny HiddenTear a Prometheus patria medzi rozšírenejšie, a preto bolo vhodné dekryptor imple-

mentovať do systémov firmy Avast, aby mohli slúžiť širšej verejnosti – dekryptor pre rodinu Prometheus bol zverejnený ako na stránkach Avastu, tak aj Europolu.

V súčasnosti vznikajú nové druhy ransomvéru, na ktoré je potrebné vytvárať detekčné pravidlá s cieľom zníženia tejto hrozby. Je to boj medzi autormi a analytikmi malvéru, ktorý každým dňom pokračuje. Detekcia nemusí byť v každom prípade stopercentná – môže sa napr. jednať o vzorku, ktorá ešte nebola klasifikovaná. Niektorí užívatelia sú aj tak ransomvérom infikovaní, a preto je treba v budúcnosti ďalej skúmať ransomvér do hĺbky a pochopiť spôsob, akým jednotlivé druhy šifrujú súbory. V najlepšom prípade analytici nájdu chybu vedúcu k tvorbe dekryptoru, inak aspoň bližšie pochopia schému kryptografie, ktorá môže byť využitá pri vydaní šifrovacích kľúčov od autorov ransomvéru.

Literatúra

[1] Allan Liska, T. G. *Ransomware: Defending Against Digital Extortion*. Sebastopol, CA: O'Reilly Media, Inc., 2017. ISBN 978-1-491-96788-1.

[2] Avast. *Help for Ukraine: Free decryptor for HermeticRansom ransomware* [online]. [cit. 2022-04-29]. Dostupné z: https://decoded.avast.io/threatresearch/help-for-ukraine-free-decryptor-for-hermeticransom-ransomware/.

[3] Constantin, L. *Ryuk ransomware explained: A targeted, devastatingly effective attack* [online], 19. marca 2021 [cit. 2022-01-21]. Dostupné z: https://www.csoonline.com/article/3541810/ryuk-ransomware-explained-a-targeted-devastatingly-effective-attack.html.

[4] Culling, C. S. Which YARA Rules Rule: Basic or Advanced? [online]. Júl 2018, [cit. 2022-01-21]. Dostupné z: https://vt-gtm-wp-media.storage.googleapis.com/2.0-Which-YARA-Rules-Rule-Basic-or-Advanced-1.pdf.

[5] Eagle, C. *The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler*. San Francisco, CA: William Pollock, 2011. ISBN 1-59327-289-8.

[6] Eagle, C. a Nance, K. *The Ghidra Book: The Definitive Guide*. No Starch Press, 2020. ISBN 9781718501027.

[7] Eilam, E. *Reversing: Secrets of Reverse Engineering*. Indianapolis, Ind.: Wiley Publishing, Inc., 2005. ISBN 0-7645-7481-7.

[8] Jamalpur, S., Navya, Y. S., Raja, P., Tagore, G. a Rao, G. R. K. Dynamic Malware Analysis Using Cuckoo Sandbox. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* [online]. 2018, s. 1056–1060 [cit. 2022-01-21]. ISBN 978-1-5386-1974-2. Dostupné z: https://ieeexplore.ieee.org/document/8473346.

[9] K. A, Monnappa. *Learning Malware Analysis*. Birmingham, UK: Packt Publishing Ltd., 2018. ISBN 978-1-78839-250-1.

[10] Kaspersky. *What is WannaCry ransomware?* [online]. [cit. 2022-01-21]. Dostupné z: https://www.kaspersky.com/resource-center/threats/ransomware-wannacry.

[11] Laffan, K. *A Brief History of Ransomware* [online], 10. novembra 2015 [cit. 2022-01-21]. Dostupné z: https://www.varonis.com/blog/a-brief-history-of-ransomware/.

[12] Sikorski, M. a Honig, A. *Practical Malware Analysis*. San Francisco, CA: William Pollock, 2012. ISBN 1-59327-290-1.

[13] Szor, P. *The Art of Computer Virus Research and Defense*. Boston, MA: Addison Wesley Professional, 2005. ISBN 0-321-30454-3.

[14] Unit 42. *Threat Brief: Kaseya VSA Ransomware Attack* [online]. [cit. 2022-04-29]. Dostupné z: https://unit42.paloaltonetworks.com/threat-brief-kaseya-vsa-ransomware-attacks/.

Príloha A

Detekčné YARA pravidlá pre rodinu ransomvéru HiddenTear

```
1  include "includes/all.yar"
2  import "cuckoo"
3
4  rule hiddentear_known_sequences
5  {
6    strings:
7      ...
8      // Dexy's Hub variant: 16810702
           afb210bc9a485bc488d9a0d76713ece1f465e53839814ac2a04efeb4
9      // sprava o vykupnom
10     $s01 = "Hello!!! I just realised that you are trying to cheat on a pixel game!
            you are probably an asshole to cheat on a pixel game. get you pc fucked"
           wide
11
12     // NYANCAT variant: 2
           e0e1e007199b4069b774108e6eeb16e0775426d817a5af045e81e8db27cd4c1
13     // sprava o vykupnom
14     $s02 = "YOU HAVE BEEN HIT BY NYANCAT RANSOMWARE" ascii
15     // debuggovacie symboly
16     $s03 = "C:\\Users\\01t0rn\\Desktop\\Malware\\ransomware\\ransomware\\obj\\
           Debug\\ransomware.pdb" ascii
17     // .NET GUID
18     $s04 = "$54f7c396-02dd-4959-9b71-63d21987c171"
19
20     // VSOP variant: 41
           c416c85d5539456f118f6113acf828a1452d5c1e54d27d0b6957affdf41bbf
21     // sprava o vykupnom
22     $s05 = "All of your files are currently encrypted by VSOP strain." wide
23     // debuggovacie symboly
24     $s06 = "c:\\slam_ransomware_builder\\ConsoleApp2\\ConsoleApp2\\obj\\Debug\\
           ConsoleApp2.pdb"
25     // .NET GUID
26     $s07 = "$1e68dd04-75c2-4579-8a98-c294fc8746e1"
27
28     // BADFILE variant: 98
           d69ac63ce5b1869e66bfb8e461fd8aa30bc102d4b05cd2edea0cd65c553d23
```

```
29      // .NET GUID
30      $s08 = "$63cfa442-8f13-4da5-8692-d808c1609a59"
31    condition:
32      EXE and filesize < 10MB and any of ($s*)
33  }
34
35  rule hiddentear_known_behavior_high
36  {
37    condition:
38      filesize < 20MB and
39      (
40        // predchadzajuce varianty
41        ... or
42        (
43          // NYANCAT
44          cuckoo.filesystem.file_write(/\\Rand123\\local\.exe$/i) and
45          cuckoo.filesystem.file_write(/\.NYANCAT$/)
46
47        ) or (
48          // Dexy's Hub
49          cuckoo.filesystem.file_write(/\\Dexy\'s Hub\.exe$/) and
50          cuckoo.filesystem.file_write(/^C:\\surprise\.exe$/)
51        ) or (
52          // VSOP
53          cuckoo.filesystem.file_write(/\.PPLIT$/) and
54          cuckoo.filesystem.file_write(/\\README\.txt$/)
55        ) or (
56          // BADFILE
57          cuckoo.filesystem.file_write(/\.BADFILE$/) and
58          cuckoo.filesystem.file_write(/\\readme-bf\.txt/)
59        )
60      )
61  }
```

Výpis A.1: Statické a behaviorálne pravidlo pre ransomvér HiddenTear

Príloha B

Detekčné YARA pravidlá pre rodinu ransomvéru Prometheus

```
include "includes/all.yar"
import "cuckoo"

rule prometheus_known_sequences
{
  strings:
    // subor so spravou o vykupnom
    $s01 = "\\UNLOCK_FILES_INFO.txt" wide
    // pripona zasifrovanym suborom
    $s02 = ".unlock" wide
    // sekvencie pre debugging
    $s03 = "Files securing is about to start..." wide
    $s04 = "This console window will close by itself. DON'T CLOSE IT MANUALLY OR
        THE WHOLE PROCESS WILL TERMINATE!" wide
    // .NET GUID
    $s05 = "$F935DC23-1CF0-11D0-ADB9-00C04FD58A0B"
  condition:
    EXE and 3 of them
}

rule prometheus_known_named_objects
{
  condition:
    cuckoo.sync.mutex(/(^|\\)5e6d4793-ace7-48e2-8f7b-b2fc43e5c5bc$/) or
    cuckoo.sync.mutex(/(^|\\)2196a2e0-5486-4b86-9526-f8d629ad1953$/) or
    cuckoo.sync.mutex(/(^|\\)c1e863e6-d5c4-485a-a624-1818bcbf8923$/) or
    cuckoo.sync.mutex(/(^|\\)c79c440b-8191-464b-ab83-3c6f57d3b8f3$/) or
    cuckoo.sync.mutex(/(^|\\)809a6d7d-b5a7-44c9-ab70-205162c8a731$/) or
    cuckoo.sync.mutex(/(^|\\)3b257655-f219-48a9-a4db-c57417cd780b$/) or
    cuckoo.sync.mutex(/(^|\\)9772ba33-5371-4801-a917-761467f8f4aa$/) or
    cuckoo.sync.mutex(/(^|\\)13fa0b6a-1f1d-4fed-a3e6-48c6cce29bbe$/) or
    cuckoo.sync.mutex(/(^|\\)f4a58c72-3ca2-462a-addf-a468cc2a2031$/) or
    cuckoo.sync.mutex(/(^|\\)2a818f15-7b11-4616-bce2-207d2a920935$/) or
    cuckoo.sync.mutex(/(^|\\)f4a58c72-3ca2-462a-addf-a468cc2a2031$/) or
    cuckoo.sync.mutex(/(^|\\)acc4c94c-c19e-4833-bc3c-d2c82de6dfeb$/) or
    cuckoo.sync.mutex(/(^|\\)43834d3d-4885-45ca-b902-9b251d0e25f7$/) or
```

```
    cuckoo.sync.mutex(/(^|\\)81f3e39c-aebd-4d50-a6ad-30094060fec1$/) or
    cuckoo.sync.mutex(/(^|\\)d00b88f0-6e4f-4730-bb70-d8f3e4bb4ac5$/) or
    cuckoo.sync.mutex(/(^|\\)935228ff-647f-4340-b467-9b928b2a9603$/) or
    cuckoo.sync.mutex(/(^|\\)faa9303b-f3d1-4eb6-b4f4-e1ff5a3807ca$/) or
    cuckoo.sync.mutex(/(^|\\)173cd33a-5700-4da3-990c-fd6419cbcce9$/) or
    cuckoo.sync.mutex(/(^|\\)ab623ae1-d843-4ced-a69f-61bfce8bb7e5$/) or
    cuckoo.sync.mutex(/(^|\\)79f42b3e-31e2-40c6-83f2-6a58e47fc089$/) or
    cuckoo.sync.mutex(/(^|\\)f10069bc-dd06-4d49-af46-32594e037c23$/) or
    cuckoo.sync.mutex(/(^|\\)d24f34e1-97af-488a-b72a-e8208e1867e3$/) or
    cuckoo.sync.mutex(/(^|\\)a32a32d2-ad4a-4f2c-bcea-f45151a0f99e$/) or
    cuckoo.sync.mutex(/(^|\\)e8d0308a-a648-11eb-949e-d2d71f9c0aba$/) or
    cuckoo.sync.mutex(/(^|\\)7a82d557-c2e7-4b9c-842d-4a3703d2132f$/) or
    cuckoo.sync.mutex(/(^|\\)0a696ac0-cce0-4b00-993b-03bd8bddf465$/) or
    cuckoo.sync.mutex(/(^|\\)35edc205-72f1-4603-bd58-5c53cf3644a3$/) or
    cuckoo.sync.mutex(/(^|\\)c257c135-a501-4fe9-8035-d167b41b50c0$/) or
    cuckoo.sync.mutex(/(^|\\)955c3d8e-5ccb-4dd4-998b-6a806b4c4063$/) or
    cuckoo.sync.mutex(/(^|\\)6c33ef4e-5899-4c3d-a8ee-4ae1a7337b64$/) or
    cuckoo.sync.mutex(/(^|\\)bd7be270-0855-4919-8991-aacc5c40d1ce$/) or
    cuckoo.sync.mutex(/(^|\\)1c39b943-05c3-48a1-aa56-cee65c485d2c$/) or
    cuckoo.sync.mutex(/(^|\\)6e2cb3ff-0146-4be3-80f2-03dd2f78b55e$/) or
    cuckoo.sync.mutex(/(^|\\)04a02176-5f34-46cc-9136-cc5f8be7fd52$/) or
    cuckoo.sync.mutex(/(^|\\)53befca4-8615-4a1d-ac33-5cc764ea35c1$/) or
    cuckoo.sync.mutex(/(^|\\)871a5812-bd23-40f2-a580-a115eb421c04$/) or
    cuckoo.sync.mutex(/(^|\\)e501a520-5a92-4cf2-92f0-2bfde5c9a708$/) or
    cuckoo.sync.mutex(/(^|\\)8e5a4e96-740a-4fcd-a3c1-5def686a71e3$/) or
    cuckoo.sync.mutex(/(^|\\)871a5812-bd23-40f2-a580-a115eb421c04$/) or
    cuckoo.sync.mutex(/(^|\\)cfe806eb-bb58-40d9-8bc0-34b4cd09ff6a$/)
}

rule prometheus_known_behavior_high
{
  condition:
    (
      cuckoo.filesystem.file_write(/\.KPSTARGARD\[prometheusdec@yahoo\.com\]$/) or
      cuckoo.filesystem.file_write(/\.\[141-5D9-Y454\]$/) or
      cuckoo.filesystem.file_write(/\.\[54Z-YAD-AWLD\]$/) or
      cuckoo.filesystem.file_write(/\.AZCUEPUMPS\[prometheushelp@mail\.ch\]$/) or
      cuckoo.filesystem.file_write(/\.PROM\[prometheushelp@mail\.ch\]$/) or
      cuckoo.filesystem.file_write(/\.\[LZG-ZNM-YDNM\]$/) or
      cuckoo.filesystem.file_write(/\.\[4B3-977-SYMH\]$/) or
      cuckoo.filesystem.file_write(/\.alumni$/) or
      cuckoo.filesystem.file_write(/\.qxic$/) or
      cuckoo.filesystem.file_write(/\.reofgv$/) or
      cuckoo.filesystem.file_write(/\.t6sqgg$/) or
      cuckoo.filesystem.file_write(/\.crypt$/) or
      cuckoo.filesystem.file_write(/\.secure$/) or
      cuckoo.filesystem.file_write(/\.61gutq$/) or
      cuckoo.filesystem.file_write(/\.iml$/) or
      cuckoo.filesystem.file_write(/\.secure\[milleni5000@qq\.com\]$/) or
      cuckoo.filesystem.file_write(/\.tmedgroup$/) or
      cuckoo.filesystem.file_write(/\.uo8bpy$/) or
      cuckoo.filesystem.file_write(/\.9ten0p$/) or
      cuckoo.filesystem.file_write(/\.ejqvfp$/) or
      cuckoo.filesystem.file_write(/\.CRYSTAL$/) or
```

```
90        cuckoo.filesystem.file_write(/\.BRINKS_PWNED$/) or
91        cuckoo.filesystem.file_write(/\.kingdee$/) or
92        cuckoo.filesystem.file_write(/\.locked$/) or
93        cuckoo.filesystem.file_write(/\.61gutq$/) or
94        cuckoo.filesystem.file_write(/\.GHANAGAS\[prometheusdec@yahoo\.com\]$/) or
95        cuckoo.filesystem.file_write(/\.secure\[milleni5000@qq\.com\]$/) or
96        cuckoo.filesystem.file_write(/\.y9sx7x$/) or
97        cuckoo.filesystem.file_write(/\.secure\[milleni5000@qq\.com\]$/) or
98        cuckoo.filesystem.file_write(/\.skyland$/) or
99        cuckoo.filesystem.file_write(/\.VIPxxx$/) or
100       cuckoo.filesystem.file_write(/\.crypt$/) or
101       cuckoo.filesystem.file_write(/\.skyland$/) or
102       cuckoo.filesystem.file_write(/\.hard$/) or
103       cuckoo.filesystem.file_write(/\.secure$/) or
104       cuckoo.filesystem.file_write(/\.unlock$/)
105     ) and (
106       cuckoo.filesystem.file_write(/\\UNLOCK_FILES_INFO\.txt$/) or
107       cuckoo.filesystem.file_write(/\\RESTORE_FILES_INFO\.txt$/) or
108       cuckoo.filesystem.file_write(/\\RESTORE_FILES_INFO\.hta$/) or
109       cuckoo.filesystem.file_write(/\\RESTORE_FILES_INFO\.txt$/) or
110       cuckoo.filesystem.file_write(/\\HOW_TO_RECOVER_YOUR_FILES.txt$/) or
111       cuckoo.filesystem.file_write(/\\README_BRINKS\.txt\.txt$/) or
112       cuckoo.filesystem.file_write(/\\RESTORE_FILES_INFO\.txt$/) or
113       cuckoo.filesystem.file_write(/\\HOW_TO_DECYPHER_FILES\.txt$/) or
114       cuckoo.filesystem.file_write(/\\Instruction\.txt$/)
115     )
116 }
```

Výpis B.1: Všetky druhy pravidiel pre ransomvér Prometheus

Príloha C

Obsah priloženého DVD

- `rules` – adresár s detekčnými YARA pravidlami
- `reports` – adresár s behaviorálnymi reportami daných vzoriek z Cuckoo sandboxu
- `malware.zip` – archív so vzorkami malvéru (chránený heslom `infected`)
- `yara.exe` – YARA scanner od firmy Avast na zisťovanie účinnosti YARA pravidiel
- `test-rules.py` – testovací skript pre overenie funkčnosti YARA pravidiel
- `test-decryptors.py` – testovací skript na overenie funkčnosti vytvorených dekryptorov
- `decryptors/avast_decryptor_hiddentear.exe` – dekryptor pre rodinu HiddenTear
- `decryptors/Cryptor_HiddenTear.cpp` – zdrojový súbor dekryptoru pre rodinu HiddenTear
- `decryptors/avast_decryptor_prometheus.exe` – dekryptor pre rodinu Prometheus
- `decryptors/Cryptor_Prometheus.cpp` – zdrojový súbor dekryptoru pre rodinu Prometheus
- `decryptors/decryptor_timetime.py` – zdrojový súbor dekryptoru TimeTime
- `clean-files` – čisté súbory, ktoré budú porovnávané s dešifrovaným súbormi dekryptoru
- `encrypted-files` – zašifrované súbory jednotlivými rodinami ransomvérov
- `dokumentacia.pdf` – dokumentácia k YARA pravidlám a dekryptorom