FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Detekční Pravidla pro Detekci Ransomware ve Formátech YARA a Sigma
Student:	Bc. Stanislav Lepič
Vedoucí:	Ing. Simona Fornůsek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem této práce je analyzovat vzorky ransomware přes různé jeho rodiny a identifikovat a popsat typické znaky jeho chování vhodné pro tvorbu detekčních pravidel. Poté navrhneme automatická detekční pravidla ve formátech YARA a Sigma. Dále testujeme účinnost a efektivitu vytvořených pravidel a diskutujeme případné problémy a jejich možná řešení.

V rámci práce:

1. Nastudujte problematiku ransomware v kybernetické bezpečnosti, prostudujte vzorky ransomware napříč rodinami a identifikujte typické znaky chování ransomware.

2. Navrhněte vhodná detekční pravidla pro detekci ransomware, pro implementaci ve formátech YARA a Sigma a následně je implementujte.

3. Experimentálně otestujte detekci ransomware s použitím vytvořených pravidel a ověřte účinnost a efektivitu navržených pravidel.

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 15. února 2024 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Detekční Pravidla pro Detekci Ransomware ve Formátech YARA a Sigma

Bc. Stanislav Lepič

Katedra informační bezpečnosti
Vedoucí práce: Ing. Simona Fornůsek, Ph.D.

9. května 2024

Poděkování

Děkuji své vedoucí Ing. Simoně Fornůsek, Ph.D. za všechny cenné rady a vedení při vytváření této práce. Děkuji všem svým blízkým, kteří mě ve studiu podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Lepič, Stanislav. *Detekční Pravidla pro Detekci Ransomware ve Formátech YARA a Sigma*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Abstrakt

Tato práce se zaměřuje na analýzu a obranu proti ransomware pomocí detekčních pravidel. Poskytuje přehled o různých typech ransomware a zkoumá jejich chování od infikování systému až po vydírání oběti. Dále se zabývá metodami statické i dynamické analýzy škodlivého softwaru. Kromě toho zkoumá také techniky, které jsou využívány k obraně proti analýze. Následně popisuje práci s pravidly ve formátech YARA a Sigma. V části návrhu jsou v těchto formátech implementována pravidla mířící na obecnou detekci vzorků ransomware.

Klíčová slova ransomware, detekce, yara, sigma, pravidla, malware vzorky, analýza malware, šifrování

Abstract

This thesis focuses on analysis and defense against ransomware using detection rules. It provides an overview of the different types of ransomware and explores their lifecycle from infecting the system to extorting the victim. It also deals with methods of static and dynamic analysis of malicious software. In addition, it also examines the techniques that are used to defend against analysis. Subsequently, work with rules in YARA and Sigma formats is described. In the design part, rules are implemented in these formats aimed at general detection of ransomware samples.

Keywords ransomware, detection, yara, sigma, rules, malware samples, malware analysis, encryption

Obsah

Seznam obrázků

Úvod

Jednou z nejzávažnějších kybernetických hrozeb posledních let je ransomware. Vyděračský software, který šifruje data a vydírá svou oběť, aby zaplatila výkupné za jejich znovuzpřístupnění. Jeho nástup ještě umocnilo rozšíření kryptoměn, jejichž užívání je spojeno s takřka úplnou anonymitou. Ačkoliv se útočníci zaměřují převážně na větší společnosti, kde je pravděpodobnější poškození důležitých dat a tedy zaplacení výkupného, není výjimkou, že obětmi se stávají i běžní uživatelé. Není také velkým překvapením, že cílová platforma prakticky každého ransomware je operační systém Windows. Propracovanější rodiny ransomware však využívají i verze pro Linux a MacOS.

Tato práce se zabývá analýzou a obranou proti ransomware. Výsledek je určen každému, kdo má zájem dozvědět se více o aktuálních hrozbách v kyberprostoru, speciálně pak těm, kdo se zajímají o detekci nebo analýzu malware obecně.

V analytické části jsou nejprve popsány druhy ransomware a jeho typické chování. Dále pak metody analýzy standardně využívané v oblasti malware, včetně obranných mechanizmů, které útočníci používají, aby zabránili detekci a zkoumání svých škodlivých programů. Následně jsou popisovány formáty, ve kterých jsou detekční pravidla vytvářena pro použití v různorodých bezpečnostních nástrojích. Důraz je po celou dobu kladen na představení volně dostupných nástrojů a služeb, poskytujících užitečné prostředky v boji proti škodlivému software.

Praktická část se věnuje analýze samotných vzorků ransomware, které jsou vybrány na základě jejich aktuálního výskytu v posledním roce. Výsledkem jsou sestavená pravidla ve formátech YARA a Sigma, cílená na obecnou detekci ransomware.

Kapitola 1

Ransomware

Ransomware (z anglického ransom „výkupné“ a software) je typ škodlivého softwaru, který útočí na počítačový systém tím, že buď blokuje přístup k systému nebo šifruje data v něm uložená. Poté požaduje od oběti výkupné za obnovení přístupu či dešifrování dat. Existují různé varianty ransomware - některé šifrují soubory na disku, zatímco jiné pouze blokují systém a prostřednictvím výhrůžných zpráv se snaží donutit uživatele k zaplacení výkupného.

Ačkoliv první známý ransomware AIDS je datován do roku 1989, největšího rozmachu se této hrozbě dostalo až s příchodem kryptoměn, které zločincům poskytují ideální anonymní způsob výběru výkupného.

Obrázek 1.1: Vývoj ransomware [1]

1.1 Druhy ransomware

Ransomware bývá tradičně rozdělován do dvou základních kategorií a to nešifrovací a šifrovací.

1.1.1 Nešifrovací ransomware

Do této kategorie patří ransomware, jehož úspěšnost je přímo úměrná se schopností přesvědčit oběť o tom, že není jiná možnost než zaplatit výkupné. Jedná se např. o tzv. *lockery*, které počítač uzamknou a pod různými smyšlenými příběhy nutí k zaplacení peněz. Vzhledem k tomu, že data většinou nejsou nijak dotčena, je možné je ze zařízení získat zpět. Speciální variantou uzamykacího ransomware je MBR (Master boot record) locker, který přepsáním MBR zabrání zavedení operačního systému a namísto toho načte rozhraní požadující zaplacení výkupného.

Na principu sociálního inženýrství funguje i tzv. *scareware*, který míří čistě na strach a nezkušenost obětí. Může se projevovat jako vyskakující okna imitující antivirový software, sdělující, že počítač byl napaden a je nutné v co nejkratší době podniknout určité kroky k nápravě. Často se také zmiňuje ztráta nebo zveřejnění osobních dat oběti, která je těmito metodami tlačena např. k nákupu falešného produktu nebo volání na zpoplatněné linky.

Stále populárnějším přístupem je data před zašifrováním odcizit a uživatele tlačit k zaplacení výkupného pod pohrůžkou jejich zveřejnění (tzv. *Double Extortion*). To bývá velmi efektivní u společností, kterým tím hrozí únik osobních údajů zákazníků nebo zveřejnění firemních tajemství. Ke zveřejnění dat používají ransomware skupiny své vlastní webové stránky (tzv. *Leak Sites*) nebo účty na sociálních sítích. Před zveřejněním dat mohou útočníci veřejně publikovat, že proběhlo jejich odcizení a zároveň spustit odpočet, do kterého je nutné zaplatit výkupné. Tím je na společnost vyvíjen tlak a zároveň je jí znemožněno únik dat utajit před zákazníky nebo státními orgány. V případě cenných informací je útočníci mohou také nabídnout k prodeji nebo dražbě, což probíhá typicky na internetových fórech sítě Tor.

Speciálním případem ransomware jsou pak tzv. *wipery*, které mají za cíl destrukci cílových dat a systémů. Příkladem může být ransomware *NotPetya*, který v roce 2017 útočil převážně na území Ukrajiny, jehož analýzou bylo zjištěno, že neumožňuje data žádným způsobem obnovit, i kdyby bylo výkupné zaplaceno. Výzkumníci se tedy domnívají, že žádost o výkupné zde byla spíš vedlejší činnost a hlavním cílem byla pouhá destrukce. [2]

1.1.2 Šifrovací ransomware

Rodiny ransomware způsobující největší škody jsou prakticky vždy šifrovací varianty mířící přímo na uživatelova data, která zašifrují a následně vyžadují výkupné za dešifrovací klíč. Typicky míří na složky, kde se dá očekávat výskyt

osobních souborů jako obrázky nebo dokumenty. Cílem není znepřístupnit systém uživateli, právě naopak. Útočníci potřebují, aby uživatel zjistil, že jeho data jsou nepřístupná a aby skrz instrukce škodlivého programu zaplatil výkupné. Pokud je ransomware navržen bez chyb, je vzhledem k silným šifrovacím algortimům prakticky nemožné data dešifrovat zpět. I přesto existuje projekt No More Ransom[a], který nabízí identifikaci a dekryptory pro stovky různých druhů ransomwarů. Ty často vznikají díky rozkrytí konkrétní zločinecké skupiny ze strany policie a zabavení infrastruktury, na které byl ransomware provozován.

1.2 Chování ransomware

Chování ransomware se neustále zdokonaluje a přizpůsobuje aktuální době. Zatímco dříve se jednalo o často jednoduchý způsob zamknutí počítače nebo pouhé vyděšení oběti s časovým nátlakem, aktuální generace jsou často vyspělé a komplexní programy používající kvalitní šifrování, exfiltraci dat, nebo komunikaci s útočníkem. Jsou také schopné bránit se vlastní analýze pomocí obfuskací a nebo blokování spuštění ve virtuálních prostředí standardně používaných pro analytickou činnost.

1.2.1 Infikování systému

Jedním z nejběžnějších způsobů šíření ransomware je prostřednictvím sociálního inženýrství. Útočníci mohou posílat tzv. *phishingové emaily*, které vypadají jako legitimní zprávy od známých společností, institucí nebo dokonce od blízkých osob. Tyto emaily obsahují škodlivé přílohy nebo odkazy, které jakmile jsou otevřeny nebo rozkliknuty, stahují samotný ransomware nebo tzv. *dropper*, který se vydává za validní software a následně vypustí do cílového zařízení samotný ransomware. Následně se snaží přesvědčit uživatele ke spuštění programu s vyššími oprávněními, které ransomware dají volné pole působnosti.

Dalším způsobem je *exploitace zranitelností*. Ransomware může být šířen prostřednictvím zneužití zranitelností v softwaru, operačním systému nebo síťové infrastruktuře. Útočníci hledají a využívají díry v zabezpečení, které mohou umožnit vzdálený přístup k zařízení a instalaci ransomware aniž by uživatel cokoliv zpozoroval. Velmi populární protokolem je v tomto ohledu *RDP (Remote Desktop Protocol)*. Ten bývá často nezabezpečen kvůli špatné konfiguraci a také jsou u něj pravidelně objevovány bezpečností zranitelnosti, které útočníci zneužívají. [1] Velmi běžnou funkcí ransomware je také zneužití zranitelností vedoucích k elevaci oprávnění, díky kterým není třeba uživatele žádat o jejich přidělení a riskovat tak případné prozrazení.

[a] `https://www.nomoreransom.org`

1.2.2 Perzistence

K zajištění *perzistence* na zařízení, tedy schopnosti přečkat restart systému nebo změnu přihlašovacích údajů, využívají rodiny ransomware několik základních technik. Jendou z nich je vložení aplikace do registrů definujících aplikace spouštěné při startu systému nebo registrů specifikující knihovny načítané za běhu do systémových knihoven. [3] Další možností je vytvoření události v plánovači úloh *SCHTASKS.EXE*, která se provede při každém startu systému. Stejně tak je možné vytvořit služby systému Windows v nástroji *SC.EXE*, které jsou při startu systému spouštěny na pozadí. V neposlední řadě je ve Windows možné vložit skript do startovní složky *C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup*, který se postará o spuštění samotného škodlivého programu. [4]

1.2.3 Šíření

Po úspěšném infikování systému bývá první krok ransomware deaktivovat ochranné prvky zařízení a začít se šířit do dalších zařízení na lokální síti. Z infikovaného zařízení začne např. pomocí nástrojů *Mimikatz* nebo *CredentialsFileView* získávat přihlašovací údaje, které je možné využít k elevaci oprávnění a připojení na další zařízení v síti. [5] Použito může být připojení skrz protokoly jako *RDP, SMB, SSH, VNC* nebo *FTP*. V případě nenalezení vhodných přihlašovacích údajů může ransomware využít útok hrubou silou. Velmi populární mezi rodinami ransomware je také zneužití konkrétních zranitelností ve výše zmíněných protokolech. Například nechvalně proslulé rodiny ransomware WannaCry a NotPetya využívají zranitelnost *EternalBlue* v protokolu SMB, která unikla americké *NSA*. [6] Ransomware ZCryptor zase zvyšuje své šance na šíření do dalších počítačů infikováním všech vyměnitelné disků kopií sebe sama ještě před zašifrováním uživatelský dat. [7]

1.2.4 Smazání záloh

Windows nabízí obnovu systému pomocí služby *Volume Shadow Service*. Pomocí té je možné vytvářet a spravovat snímky operačního systému, kdy snímkem je myšlen stav systému v určitém čase. Tyto snímky jsou nazývány *stínové kopie (shadow copies)* a s jejich pomocí je možné systém obnovit do předchozího stavu před událostmi narušení. Aby oběti neměly možnost obnovit systém do stavu před útokem, většina rodin ransomware tyto kopie ničí a deaktivují jejich vytváření. Toho dosahují pomocí nástroje systému Windows nazvaného *VSSADMIN.EXE*, pro jehož použití je vyžadováno zvýšené správcovské oprávnění. Alternativně mohou útočníci smazat stínové kopie pomocí *Windows Management Instrumentation (WMI)*, toho je možné docílit nástrojem příkazové řádky jménem *WMIC.EXE*. [8]

1.2.5 Exfiltrace a zašifrování dat

K exfiltraci dat oběti je stále častěji využíváno legitimních nástrojů určených pro správu nebo vzdálené ovládání počítače. To nejenže usnadňuje práci útočníků s vývojem vlastních nástrojů, ale zároveň snižuje schopnosti detekčních nástrojů. Tyto programy mohou být navíc již instalovány na cílovém zařízení, což opět zjednodušuje celý útok. Mezi často zneužívané nástroje patří např. Rclone, AnyDesk, WinSCP nebo ScreenConnect. Stejně tak mohou být zneužity vestavěné nástroje systému Windows jako RDP nebo PowerShell. Jejich chování je možné detekovat pomocí změn v registrech nebo spuštěním specifického příkazu, který je zločinci často využíván. [9]

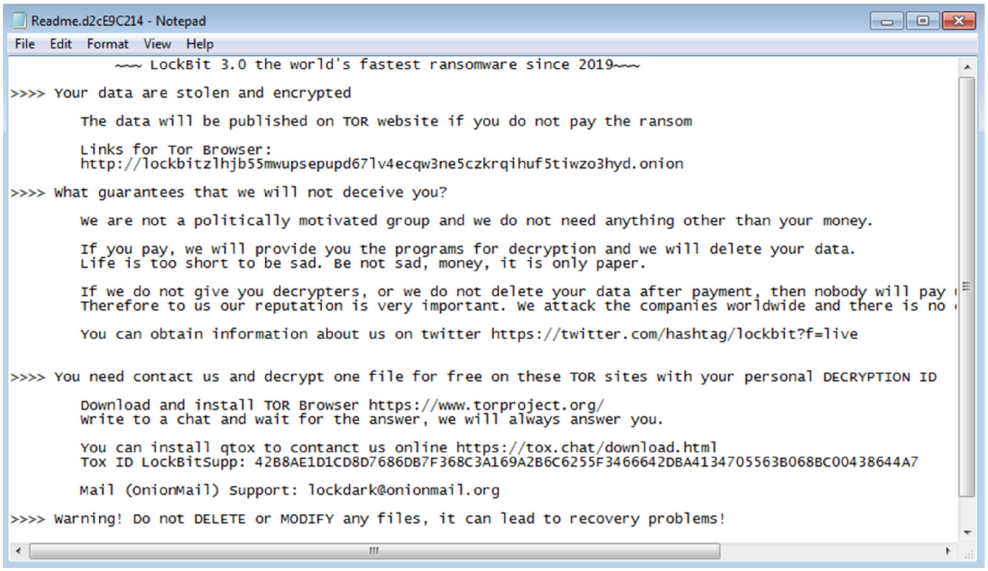
Komunikace z infikovaného zařízení probíhá přes tzv. *Command and control* (C&C) servery. Na ty jsou typicky šifrovaně zasílána data o oběti a použité šifrovací nebo dešifrovací klíče. Adresy těchto C&C serverů bývají nejčastěji napevno vložené přímo v ransomware, méně často jsou dynamicky generovány pomocí tzv. *Domain generation algorithms*, čímž stěžují detekci známých adres. Využívány bývají nejčastěji standardní šifrovací algoritmy jako AES pro symetrické šifrování a RSA pro asymterické. I zde existují výjimky rodin ransomware, které nevyužívají žádné C&C servery pro externí komunikaci. [10]

Aby nebyla ohrožena funkčnost systému, ransomware typicky využívá předdefinovaný seznam přípon souborů, u kterých je nejpravděpodobnější, že budou obsahovat osobní data uživatele. Následně prochází složky jako např. *Dokumenty*, *Plocha* nebo sdílené síťové složky a šifruje všechna nalezená data splňující parametry. Po jejich zašifrování většinou přidá vlastní koncovku souboru, čímž, mimo jiné, předejde opakovanému zašifrování stejného souboru. Většina rodin ransomware používá stále stejnou koncovku, dle které ji lze snadno identifikovat, existují ale i výjimky, přiřazující náhodnou koncovku. V menším procentu případů je změněn i název souboru na náhodný řetězec, čímž je opět komplikováno případné obnovení dat. Některé rodiny ransomware volí postupné šifrování dat, které se neprojeví výrazným zatížením procesoru, jiné míří čistě na rychlost, bez ohledu na zátěž.

Pro zabránění opakovaného infikování jednoho zařízení ransomware kontroluje skrz Windows API funkce jako *CreateMutex* nebo *OpenMutex*, jestli již neexistuje mutex se specifickým jménem, který byl při prvním běhu ransomware vytvořen. Vícenásobné spuštění by mohlo vést k nefunkčnosti ransomware, opakovanému zašifrování dat, zašifrování různými klíči a nebo snazší detekci např. antivirovým programem. [11]

1.2.6 Vydírání

Posledním krokem je zobrazení výzvy tzv. *ransom note* uživateli, ve které mu je sděleno, že jeho data byla zašifrována a za jejich zpřístupnění je nutné zaplatit. Zpráva může být zobrazena jako samostatné okno programu, pozadí pracovní plochy a nebo jako soubor, např. ve formátech .txt nebo .htm vložené



Obrázek 1.2: Lockbit 3.0 ransom note [13]

na pracovní plochu nebo do adresářů se zašifrovanými soubory.

Ransom note může specifikovat časové okno, ve kterém je nutné výkupné zaplatit a po jehož uplynutí bude požadovaná částka navýšena nebo budou data rovnou zničena, případně zveřejněna. Dále se zde mohou nacházet údaje potřebné k zaplacení výkupného jako unikátní identifikátor oběti, identifikátor kryptoměnové peněženky, na které má být výkupné zaplaceno nebo *.onion* webová adresa využívaná anonymní sítí *Tor*, kde jsou oběti sděleny další pokyny. Výzva k instalaci prohlížeče *Tor*[b] bývá jedním z nejčastějších kroků obsažených v ransom note. Alternativně je po oběti požadováno, aby útočníky kontaktovala na přiložené emailové adrese. Ty bývají často hostovány na službách, díky kterým není možné je vystopovat. Útočníci navíc mohou emailové adresy velmi snadno měnit nebo dynamicky generovat.

V minulosti běžně využívané metody výběru výkupného jako prémiové SMS nebo předplacené karty (např. paysafecard[c]) prakticky úplně nahradily kryptoměny nabízející úplnou anonymitu. Mezi kryptoměnami pak velmi výrazně převažuje *Bitcoin*. [12]

Některé komplexnější rodiny ransomware umožňují obětem i komunikaci s pachatelem nebo ukázkové dešifrování souboru, které má podpořit důvěryhodnost útočníků.

[b] `https://www.torproject.org/`
[c] `https://www.paysafecard.com`

1.3 Ransomware jako služba

Nejvýznamnějším trendem poslední let je tzv. Ransomware as a Service (RaaS). Cílem skupin provozující tyto služby je zjednodušit ransomware útoky pro zločince, kteří nemají technické dovednosti na vytvoření vlastního ransomware. Ti buď za podíl ze získaného výkupného nebo prostým nákupem dostávají přístup k samotnému ransomware. Ten může mít různé formáty, například zdrojový kód, který si kupující sám zkompiluje, předkompilované binární soubory nebo rozhraní, do kterého kupující zadá informace o obětech. [14] Takto škodlivý program zločinci sami vloží do zařízení nebo infrastruktury, ke které získali přístup. Nákup této služby může zahrnovat i další podpůrné prostředky jako tipy a návody pro nejsnazší infikování zařízení a následné šíření, infrastrukturu pro komunikaci s obětí, online podporu atd. RaaS obvykle nabízí sofistikovaný software s online ovládacím panelem, který poskytuje přehled o aktuálních útocích a platbách. [15]

Jeden z prvních případů RaaS skupiny je datován do roku 2012 k ransomware Reveton. Ten je také často přezdíván jako FBI virus, neboť po zamknutí počítače se vydával za tuto vládní organizaci a obviňoval jeho majitele z nelegální činnosti, za jejíž provozování musí zaplatit pokutu. [16]

Mezi nejznámější skupiny, provozujícím RaaS můžeme zařadit např. Hive, REvil, Dharma, LockBit, ALPHV a další. Vzhledem ke škodám dosahujícím stovek milionů dolarů a napadení mnoha cílů kritické infrastruktury států jsou tyto skupiny v neustálém hledáčku policejních orgánů. [17] To už vedlo k pozastavení nebo rozbití několika těchto organizací. Naposledy se jednalo o skupinu LockBit, která byla po mezinárodní operaci *Cronos* významně ochromena. Její infrastruktura a značné množství financí bylo zabaveno a po technické analýze byl vydán program umožňující dešifrování poslední verze tohoto ransomware. [18]

Existuje mnoho projektů sledujících aktuální oběti ransomware a mapujících jednotlivé zločinecké skupiny. Zmínit můžeme např. ransomware.live[d], ransomwatch[e] nebo twitterový účet @RansomwareNews[f].

[d] `https://www.ransomware.live`
[e] `https://ransomwatch.telemetry.ltd`
[f] `https://twitter.com/RansomwareNews`

Kapitola 2

Analýza ransomware

2.1 Statická analýza

Statická analýza je metoda analýzy programu, která se provádí bez jeho skutečného spuštění či vykonání. Tato analýza se zaměřuje na zkoumání struktury, obsahu a vlastností daného objektu na základě statických informací, jako je zdrojový kód, strojový kód, obsah souboru nebo metadata.

2.1.1 PE formát

PE formát (z angl. Portable Executable) je formát souborů pro spustitelné soubory (EXE), dynamické knihovny (DLL), objektové soubory (OBJ) a další, používaný v operačním systému Windows. PE formát krom samotného kódu programu obsahuje i další metadata včetně importovaných a exportovaných funkcích, času vytvoření souboru, cílové architektury a informací o jejich rozložení v paměti atd. Obsahuje všechny informace potřebné k tomu, aby mohl zavaděč operačního systému načíst daný spustitelný soubor do paměti a spustit jej.

DOS hlavička

Začátek PE souboru obsahuje 64bajtovou DOS hlavičku, která slouží k zajištění zpětné kompatibility s historickým operačním systémem MS-DOS. Tato hlavička obsahuje v prvních 2 bajtech tzv. magickou hodnotu „MZ“ (0x5A4D), identifikující spustitelný. Tato hodnota označovaná jako *e_magic* je při tvorbě detekčních pravidel hojně využívána. Poslední 4 bajty DOS hlavičky slouží k uložení offsetu (odsazení), na kterém se nachází *PE signatura*. Tato hodnota je označována jako *e_lfanew*.

Po DOS hlavičce následuje sekce nazývána DOS Stub, která slouží k zachování zpětné kompatibility s operačním systémem MS-DOS. Ta obsahuje

krátký program, který při načtení v MS-DOS vypíše chybovou hlášku *This program cannot be run in DOS mode.* [19]

NT záhlaví

První část NT Headers je samotná *PE signatura* obsahující hodnotu „PE\0\0“ (0x50450000). Na tu navazuje *Souborové záhlaví* (File Header) obsahující informace jako typ architektury procesoru, na kterém je program spustitelný, časové razítko vytvoření souboru, velikost Volitelného záhlaví a další charakteristiky. *Volitelné záhlaví* (Optional Header) má proměnnou velikost a dle druhu souboru v něm nemusí některé části vůbec existovat. Zajímavá data pro analýzu zde lze získat např. z prvních 2 bajtů označených jako *magic*, které ukazují, zda je soubor určen pro 32bitovou nebo 64bitovou architekturu. Hodnota *subsystem* definuje, který Windows subsystém je vyžadován ke spuštění aplikace. Typicky jde o Windows GUI (graphical user interface) nebo CUI (character user interface) subsystém. Poslední sekcí jsou Datové adresáře (Data Directories), které uchovávají adresy na důležité struktury v souboru, jako jsou tabulky importovaných a exportovaných funkcí, tabulka zdrojů nebo tabulka výjimek. [19]

Sekce

Po *Volitelném záhlaví* a před samotnými *Sekcemi* se nachází *Záhlaví sekcí* (Section Headers), která obsahují informace o sekcích PE souboru.

Sekce jsou kontejnery dat spustitelného souboru, zabírající zbytek PE souboru za *Záhlaví sekcí*. Některé sekce mají speciální názvy, které označují jejich účel a definují např., zda je možné kód v sekci upravovat nebo spouštět. Mezi několik základních sekcí patří:

.text obsahuje spustitelný kód programu, který se skládá z instrukcí pro procesor.

.data obsahuje inicializovaná data, sem patří globální a statické proměnné, které mají přiřazenou explicitní hodnotu v kódu, nebo během běhu programu.

.bss obsahuje neinicializovaná data. Tato sekce je rezervována pro globální proměnné a statické proměnné, které nejsou explicitně inicializovány v kódu.

.rdata obsahuje konstantní data, která jsou určena pouze pro čtení (read-only). Sem patří například textové řetězce, konstanty a další data, na která se odkazuje z kódu programu.

.edata obsahuje informace o exportovaných funkcích a datových položkách z daného modulu (DLL).

.idata obsahuje informace o importovaných funkcích a knihovnách (DLL), které jsou použity v programu.

.reloc obsahuje informace o relokacích pro dynamické přesouvání kódu a dat v paměti.

.rsrc obsahuje zdroje používané programem, mezi něž patří obrázky, ikony nebo vložené binární soubory.

.tls (thread-local storage) obsahuje data, která jsou specifická pro každé vlákno (thread) v programu.

2.1.2 Hashe souborů

Kryptografické hashovací funkce vytvářejí alfanumerický řetězec, jenž je pro konkrétní soubor nebo vstupní data jedinečným a neměnným identifikátorem. Na rozdíl od šifrování je hashování jednosměrná funkce navržená tak, aby bylo prakticky nemožné vypočítat původní vstup pouze na základě hodnoty hash.

Detekce malware pomocí hash hodnoty je nejjednodušší a nejrychlejší způsob, který ovšem při velmi drobné změně vstupních dat zásadně změní výsledný hash. Tento nedostatek se snaží řešit tzv. *fuzzy hashe*, kde podobná vstupní data mají i velmi podobný hash.

Následuje popis hashů používaných službou VirusTotal[g], která je dnes standardem pro online analýzu souborů.

MD5 / SHA1 / SHA256 Standardní hashovací funkce používané k identifikování vzorků. MD5 a SHA-1 by se již neměly používat, protože jsou považovány za prolomené. Např. u MD5 je možné vytvářet kolize hashů způsobem, který umožňuje kontrolu nad obsahem. Oba však stále používají některé detekční technologie, protože jejich výpočet je rychlý a hodnoty nepotřebují mnoho úložného prostoru.

Authentihash je SHA-256 hash, který se vypočítává z podepsaných PE souborů a je důležitou součástí formátu digitálního podpisu Authenticode společnosti Microsoft. Jeho účelem je ověřit, že se souborem nebylo po jeho podepsání vydavatelem softwaru manipulováno.

Imphash Import Hash se používá pro PE soubory. Je vypočítán z obsahu tabulky importů. Spojí názvy importovaných funkcí a modulů do jednoho řetězce, převede jej na malá písmena a poté vytvoří jeho MD5 hash. Imphash je užitečný při hledání spojitostí mezi různými variantami a rodinami škodlivého kódu.

[g] `https://www.virustotal.com/`

SSDEEP je považován za standard ve světě fuzzy hashů. Používá rolling hash algoritmus k rozdělení souborů na fragmenty (chunky) a následně vytváří hashovací řetězec, který reprezentuje celkovou strukturu souboru. SSDEEP je navržen pro efektivní detekci podobných nebo změněných verzí souborů, původně byl vyvinut pro detekci spamu.

TLSH Trend-Micro Locality Sensitive Hash používá techniku zvanou Local Sensitive Hashing (LSH), která extrahuje rysy souborů a na jejich základě vytváří hash. TLSH je méně citlivý na změny ve srovnání s SSDEEP. [20]

2.1.3 Řetězce

Extrakce textových řetězců z binárních souborů je jedna z nejpřímočařejších metod, která funguje na principu identifikace čitelných textových sekvencí. I přesto, že tato technika působí jednoduše, je potřeba mít na paměti hned několik komplikací. Jedním z hlavních problémů je rozdílnost v kódování. Například kódování UTF-16 používá dvojice bajtů pro každý znak, což je odlišné od standardního ASCII kódování, které používá jediný bajt na znak. To může mít za následek nesprávné interpretace textových dat, pokud není použit správný algoritmus pro dané kódování.

Dalším problémem je skutečnost, že binární soubory mohou obsahovat náhodná uskupení tisknutelných znaků, což může vést k velkému množství nalezených řetězců, z nichž jen malá část je skutečně užitečná pro detekci. Je proto klíčové používat filtrační techniky, které pomáhají identifikovat a vybrat relevantní řetězce. To může zahrnovat stanovení minimální délky řetězce, vyhodnocování entropie a struktury textu nebo analýzu obsahu (například počet alfanumerických znaků nebo speciálních symbolů). [21]

Obecně jsou pro tvorbu detekčních pravidel vhodné co nejdelší a nejunikátnější řetězce, které mohou obsahovat např. názvy souborů nebo funkcí, URL adresy, IP adresy nebo nejrůznější identifikátory.

2.1.4 Disassembling a dekompilace

Disassembling je technika v oblasti reverzního inženýrství, která převádí strojový kód programu na jazyk strojových instrukcí – assembler, který je pro analytika mnohem jednodušeji čitelnější než samotný strojový kód. Disassemblovaný kód následně může sloužit ke statické analýze, ale je možné ho pomocí debuggeru využít i k analýze dynamické.

Dekompilace je proces, který se snaží ze strojového kódu zrekonstruovat původní zdrojový kód. Úspěšnost tohoto procesu je závislá na programovacím jazyku, ve kterém byl program vytvořen. Ve většině případů však není možné získat původní zdrojový kód, neboť při kompilace ze zdrojového kódu na strojový kód je ztraceno značné množství informací.

Pro disassembling i dekompilaci je standardem nástroj IDA Pro (Interactive DisAssembler)[h] nebo open-source alternativa Ghidra[i].

2.1.5 API volání

Analýza volání systémového API může poskytovat informace o funkcích vyvolaných malwarem. To může zahrnovat operace manipulující se soubory, registry, procesy, síťové operace atd. Pokud je určité API volání prováděno ve spojení s manipulací souborů na kritických místech (např. v adresářích systému), může to naznačovat podezřelé chování. Tuto analýzu lze provádět staticky i dynamicky. Například volání API funkcí "CryptEncrypt"a "NtWriteFile"jsou během provádění ransomwaru vyvolávána častěji, protože se snaží data zašifrovat a následně zapsat zašifrované informace zpět do souboru. [22]

Velmi dobrý přehled API volání zneužívaných v malware a konkrétně ransomware poskytuje webová stránka MalAPI.io[j].

2.1.6 Graf toku řízení

Control Flow Graph (CFG) je orientovaný graf, který znázorňuje řízení toku programu, kde bloky kódu jsou reprezentovány uzly a cesty jsou reprezentovány přechody mezi jednotlivými bloky kódu podle logiky programu. Každý uzel v CFG představuje základní blok kódu, což je posloupnost instrukcí, které se provádějí za sebou, bez přerušení skokem (např. podmíněným skokem nebo návratem z funkce). CFG umožňuje identifikovat a vizualizovat různé cesty, kterými může program procházet při provádění, a také snadněji analyzovat důležité body v programu, jako jsou volání funkcí, podmínky nebo smyčky.

2.1.7 Opkódy

Opkód (z anglického "opcode") je první částí strojové instrukce, která identifikuje, jaká operace má být vykonána procesorem. Úplná strojová instrukce se skládá z opkódu a volitelně jednoho nebo více operandů. Opkód může být využit jako charakteristika při detekci malwaru pomocí testování frekvence opkódů nebo výpočtu podobnosti mezi sekvencemi opkódů. Tento přístup umožňuje detekovat podobné kusy kódu nebo sledovat evoluci určitého malwaru přes různé verze. Podobnost mezi opkódy může naznačovat společné zdroje nebo autory.

[h] `https://hex-rays.com/ida-pro/`
[i] `https://github.com/NationalSecurityAgency/ghidra/releases`
[j] `https://malapi.io/`

2.2 Dynamická analýza

Dynamická analýza je metoda pro studium chování programu nebo systému za běhu, kdy je aktivně spuštěn a následně je pozorována jeho interakce s prostředím. Jedním z hlavních cílů dynamické analýzy je identifikovat potenciálně škodlivé nebo nebezpečné chování, jako jsou nežádoucí síťové aktivity, tvorba nebo smazání souborů, manipulace s operačním systémem nebo neobvyklé interakce s dalšími aplikacemi.

2.2.1 Windows registry

Registry v systému Windows jsou jeho důležitou součástí sloužící k ukládání a správě konfiguračních informací, jako je nastavení systému, aplikací nebo hardwaru. Jedná se o centrální databázi, která obsahuje hierarchicky uspořádané informace ve formě klíčů a hodnot. Technik zneužití registrů ke škodlivým činnostem existuje celá řada, ať už se jedná o jejich vytvoření, smazání, modifikace nebo pouhé čtení. Příkladem těchto technik může být přístup k přihlašovacím údajům, dosažení perzistence, elevace oprávnění, deaktivace obraných mechanismů nebo přímo spuštění škodlivého kódu. Podrobný přehled většiny technik je možné získat ze znalostní báze MITRE ATT&CK[k].

2.2.2 Síťový provoz

V síťovém provozu je standardem pro detekci a prevenci průniku (IDPS) nástroj Snort[l], který na základě detekčních pravidel dokáže zachytávat podezřelý síťový provoz jako větší množství nestandardních DNS dotazů indikujících např. komunikaci s C&C serverem. Dále komunikace s podezřelými IP adresami nebo na nestandardních portech, komunikace přes I2P (Invisible Internet Project) nebo Tor, případně zneužití známých zranitelností. [23]

Pro analýzu souborů síťového provozu ve formátu .pcap je možné využít online službu DynamiteLab[m]. Vzorky síťového provozu zaznamenávajícího komunikaci malware je možné získat např. z webu Malware-Traffic-Analysis[n].

2.2.3 Sandboxing

Sandboxing představuje důležitou techniku v oblasti analýzy malware, která umožňuje bezpečné spouštění nedůvěryhodného softwaru v izolovaném prostředí za účelem studia jeho chování a funkcí. Tato technika je významná pro bezpečnostní analytiky, kteří se snaží porozumět škodlivým programům a vyvinout účinné metody detekce a ochrany.

[k] `https://attack.mitre.org/datasources/DS0024/`
[l] `https://www.snort.org/`
[m] `https://lab.dynamite.ai/`
[n] `https://www.malware-traffic-analysis.net/`

Princip sandboxingu spočívá v izolaci a omezení provádění škodlivého kódu v prostoru, který je oddělený od hlavního operačního systému. Toho se obvykle dosahuje prostřednictvím virtualizace nebo kontejnerizace, které umožňují simulovat prostředí maximálně blízké reálnému operačnímu systému.

Během analýzy v sandboxu jsou monitorovány různé aspekty chování malware, včetně síťové komunikace, operací souborového systému, manipulace s registrem a dalších systémových aktivit. Sandboxy jsou však schopné provádět i metody statické analýzy a tím doplnit komplexní obraz chování zkoumaného vzorku.

Mezi nejpopulárnější sandboxy s otevřeným zdrojovým kódem (angl. open-source) patří Cuckoo sandbox a jeho nástupce CAPE[o]. Volně dostupnou instanci mírně upraveného Cuckoo snadboxu provozuje CERT-EE[p].

2.3 Obrana proti analýze

2.3.1 Obfuskace

Obfuskováním je myšlen proces znepřehlednění analyzovaného souboru, při zachování jeho funkcionality, s cílem maximálně zkomplikovat a zpomalit jeho analýzu např. pomocí disassemblingu nebo dekompilace.

Metod obfuskace zdrojového kódu je mnoho, následující seznam uvádí pouze některé z nich.

Obfuskace rozložení zahrnuje jednoduché odstranění informací o formátování, názvech identifikátorů nebo samotných komentářů.

Obfuskace řízení toku kódu můžeme rozdělit do kategorií

- **Transformace výpočtu** přidávají nerelevantní kód do programu, který nemusí být nikdy vykonán nebo který komplikuje jeho pochopení (např. v podmínkách). Do této kategorie spadá i tabulková interpretace kódu a paralelizace kódu, které patří k těm nejkomplexnějším obfuskacím, jež jsou zároveň výpočetně náročné, což se projevuje na spotřebě systémových zdrojů.
- **Transformace agregace** obsahuje metodu inliningu, která ruší funkce v programu a kopíruje jejich kód přímo na cílová místa nebo outliningu, který naopak z kódu vytváří funkce. Také sem náleží manipulace se smyčkami, které je rozdělí do bloků nebo rozbalují jejich obsah do kódu za sebe.
- **Transformace pořadí** náhodně mění pořadí provádění kódu.

Obfuskace dat míří na přeuspořádání dat, jako pole, třídy, proměnné atd. Dále na změnu kódování znaků nebo tvorbu falešných tříd. [24]

[o] `https://github.com/kevoreilly/CAPEv2`
[p] `https://cuckoo.cert.ee/`

2.3.2 Anti-debugging

Debuggeru je nástroj umožňující analýzu běhu programu na úrovni strojového kódu nebo assembleru, který umožňuje sledovat a ovládat chování programu, pozastavit jeho běh, sledovat obsah paměti a registru, provádět krokování kódu nebo monitorovat systémové volání.

Většina malware se snaží debugging znesnadnit nebo úplně znemožnit např. tím, že program pozmění své chování a nemusí se projevit škodlivě nebo se úplně odmítne spustit.

Metod existuje mnoho, ty méně sofistikované pouze kontrolují přítomnost debbugeru pomocí funkcí WinAPI jako *IsDebuggerPresent, CheckRemoteDebuggerPresent, NtQueryInformationProcess, NtSetInformationThread*. Přítomnost debuggeru s konkrétním názvem je možné zjistit i funkcemi jako *FindWindow* pro vyhledávání okna specifického jména nebo vyhledání názvu procesu pomocí *Process32Next, Process32First*, případně je možné názvy debuggerů vyhledává přímo v paměti. Možné je také pomocí registrů detekovat nastavení bodů přerušení (breakpoints), které se při analýze programu využívají k jeho pozastavení. V neposlední řadě může program měřit čas svého vykonávání a kontrolovat, jestli nedochází ke zpoždění, které by ukazovalo na pokusy o jeho analýzu. [25]

2.3.3 Packing

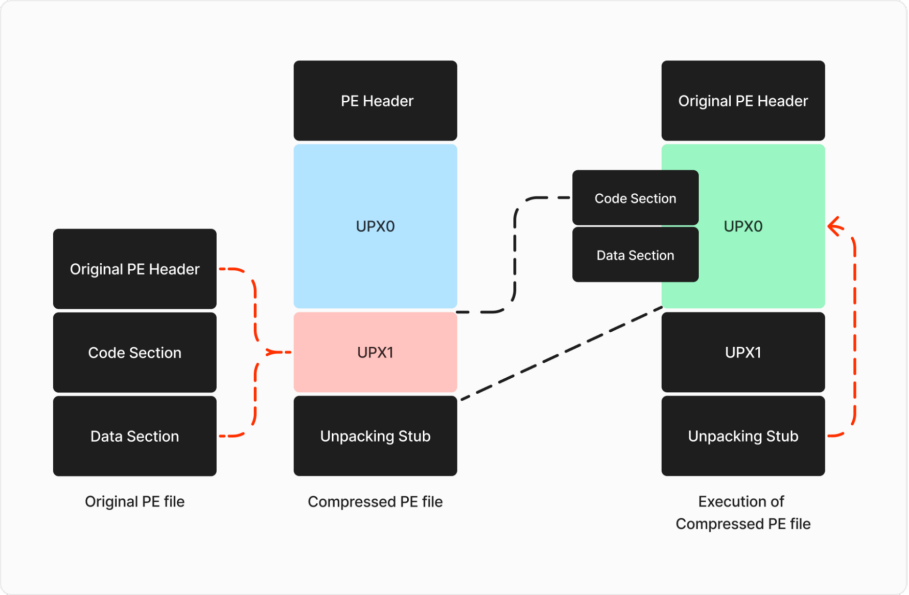
Packing (zabalení) je technika používaná ke komprimaci a případně šifrování binárního kódu škodlivého softwaru za účelem skrytí jeho skutečného obsahu a ztížení jeho detekce. Obecně lze zabalený soubor poznat podle vysoké entropie souboru, minima čitelných řetězců a minima importovaných nebo exportovaných funkcí.

Běžně používané packery jsou většinou snadno rozpoznatelné pomocí známých signatur. Například jeden z nejpoužívanějších packerů UPX[q] je snadno identifikovatelným výskytem řetězců *UPX!, UPX0, UPX1*. Malware však může využívat vlastní packer, který nelze tak snadno detekovat. Nicméně před svým spuštěním musí být škodlivý kód rozbalen do paměti systému, kde už je možné ho detekovat.

Velmi populárním nástrojem pro identifikaci packerů a zároveň i statickou analýzu souborů je Detect It Easy[r]

[q] `https://github.com/upx/upx`

[r] `https://github.com/horsicq/Detect-It-Easy`



Obrázek 2.1: Proces spuštění souboru zabaleného UPX packerem [26]

Kapitola 3

YARA

YARA je open-source nástroj vyvinutý Victorem M. Alvarezem ze společnosti VirusTotal s cílem usnadnit a zefektivnit proces detekce a analýzy škodlivého kódu, který umožňuje definovat pravidla pro identifikaci vzorů v binárním obsahu souborů.

Funguje na základě skenování souborů pomocí definovaných pravidel a následně ohlašuje nalezené shody. Pravidla jsou vytvářena pomocí vlastní syntaxe a mohou zahrnovat řetězce v několika formátech. YARA umožňuje definovat pravidla, které jsou flexibilní, podporuje tedy použití regulárních výrazů a logických operátorů pro kombinování podmínek.

Běžně se používá v bezpečnostních systémech pro detekci a prevenci průniku (IDPS), k identifikaci škodlivého software a dalších potenciálních hrozeb. Dá se využít nejen k analýze statických dat, ale i analýze paměti.

Vývojář YARA Victor M. Alvarez ze společnosti VirusTotal aktuálně pracuje na nové generaci tohoto nástroje YARA-X, který implementuje v jazyku Rust. Ten by měl nabídnout rozšířenou podporu pro moduly, modulární architekturu a větší možnosti přizpůsobení pro uživatele.

3.1 Formát

Pravidlo ve formátu YARA lze rozdělit na tři základní sekce metadata (meta), řetězce (strings) a podmínky (condition). Při popisu formátu bude využívána příručka stylu od Floriana Rotha. [27]

3.1.1 Metadata

Metadata by měla obsahovat minimálně krátký popis pravidla, jméno autora a případně organizace, datum vytvoření pravidla, a také odkaz na výzkum, článek, blog apod., popisující škodlivý software. Volitelně mohou být připsány i další informace jako hash hodnoty souborů, na které byla pravidla cílena,

```
rule Example : OptionalTag
{
    meta:
        description = "Detects ..."
        author = "Author Name / Company / Org"
        date = "YYYY-MM-DD"
        reference = "URL / Internal Research"
    strings:
        $text_string = "My string" wide ascii nocase
        //comment
        $hexa_string = { E2 34 F1 67 }
        $regex = /abc[def]+/

    condition:
        filesize < 100KB and
        $text_string or ($hexa_string and $regex)
}
```

Obrázek 3.1: Ukázkové pravidlo ve formátu YARA

skóre odhadující na škále 0–100 závažnost hrozby, název staršího pravidla, ze kterého je vycházeno nebo licence, pod kterou je pravidlo publikováno.

3.1.2 Řetězce

Do části obsahující řetězce je možné vložit textové řetězce, hexadecimální řetězce nebo regulární výrazy.

Pro textové řetězce i regulární výrazy ve výchozím stavu platí, že jsou vyhodnocovány s přihlédnutím na velikost písmen a na jeden znak náleží jeden bajt. Toto chování je ovšem možné změnit pomocí modifikátorů. Klíčové slovo *wide* umožňuje vyhledávat řetězce kódované dvěma bajty na znak, v kombinaci s *ascii* pak varianty jednobajtové i dvoubajtové najednou. Modifikátor *nocase* umožňuje nerozlišovat velikost písmen. Identifikace řetězců, které jsou odděleny od zbytku textu výhradně nealfanumerickými znaky, je možné vynutit klíčovým slovem *fullword*.

Hexadecimální řetězce umožňují do sekvencí bajtů vkládat čtyři druhy konstrukcí, které je činí flexibilnějšími. Zástupný znak symbolizovaný otazníkem „?“ umožňuje nahradit pozici v sekvenci libovolnou hodnotou. Pokud je třeba popsat pravidlem sekvenci neznámého obsahu a délky, je to možno udělat pomocí tzv. skoků. Zapsáním skoku „[2-3]“ tedy například definujeme, že tuto pozici může obsadit posloupnost dvou až tří libovolných bajtů. Tilda „∼“ znamená v tomto kontextu negaci, pomocí ní je tedy možné určit hodnoty, které nemají být detekovány. Poslední možností je využití svislítka „|“,

které se využívá jako alternativa. Tím je možné docílit detekce dvou a více specifických hodnot na jedné pozici.

Kompletní přehled všech klíčových slov, konstrukcí a jejich možných kombinací je možné naleznout v oficiální dokumentaci. [28]

3.1.3 Podmínky

Jak už název napovídá, poslední sekce umožňuje definovat logické podmínky. Kromě standardních logických, relačních, aritmetických a bitových operátorů je možné porovnávat i velikost souboru nebo hodnoty bajtů na konkrétních pozicích, což je hojně využíváno například pro detekci sekvence „MZ“ ve spustitelných souborech, viz 2.1.1. Možné je zde využívat i iterátory pomocí známého příkazu *for*.

3.2 Moduly

Rozšíření o další funkce nabízejí YARA moduly. Ty jsou implementovány v jazyce C a vzhledem k otevřenosti celého projektu YARA může vytvořit vlastní modul kdokoliv. Ty nejvýznamnější, které jsou s YARA oficiálně distribuovány, budou na základě oficiální dokumentace [29] popisovány v této kapitole. Moduly umožňují např. jednodušší práci s vybranými souborovými formáty, spolupráci se Cuckoo sandboxem nebo podporu nejrůznějších výpočtů. Pro jejich využití je stačí na prvních řádcích pravidla importovat. V případě PE modulu tedy stačí vložit pouze *import "pe"*, obdobně u dalších modulů.

3.2.1 PE modul

Jeden z nejobsáhlejších modulů podporuje přístup k většině polím v souborech PE formátu. Je tedy možné provádět vyhledávání importovaných a exportovaných funkcí, detekovat cílovou architekturu atd. Jako příklad může sloužit podmínka pro detekci importované funkce IsDebuggerPresent z knihovny kernel32.dll *pe.imports("Kernel32.dll", "IsDebuggerPresent")*.

3.2.2 ELF modul

Formát ELF je Unixovou alternativou PE formátu. Tento modul je tedy velmi podobný PE modulu, jeho cílem je také snadno zpřístupnit většinu zde obsažených polí. Např. pro detekci spustitelného souboru je možné použít podmínku *elf.type == elf.ET_EXEC*.

3.2.3 LNK modul

Soubory s příponou .LNK představují zástupce (angl. shortcut) v systému Windows, což jsou malé soubory, které slouží k odkazování na jiné soubory

nebo aplikace. Ty jsou často zneužívány ke stažení malware nebo přímo k jeho spuštění. Pro spuštění škodlivého kódu bývá, mimo jiné využívána struktura *COMMAND_LINE_ARGUMENTS*, jejíž detekování lze provést podmínkou *lnk.command_line_arguments == "\x00 \x00 \x00"*.

3.2.4 Cuckoo modul

Jak už název napovídá, tento modul dovoluje vytvářet pravidla na základě informací poskytnutých přímo z Cuckoo sandboxu. To poskytuje poměrně unikátní kombinaci a rozšiřuje schopnosti YARA detekovat nejen na základě obsahu souborů, ale i v závislosti na chování souboru. Aktuálně jsou podporovány funkce detekující síťové chování, jako kontaktování určené IP adresy a DNS nebo HTTP dotazy na hledané webové domény. Nabízí také možnost detekce přístupu k registrům, souborům a mutexům.

3.2.5 Math modul

Matematický modul umožňuje využívat běžné operace jako minima, maxima, absolutní hodnoty, průměry atd. Významná je možnost výpočtu entropie řetězce nebo libovolné části souboru, k čemuž stačí specifikovat počáteční koncový bod souboru. Pro výpočet entropie celého souboru lze tedy použít příkaz *math.entropy(0, filesize)*.

3.2.6 Další moduly

Modul *Time* nabízí hodnotu aktuálního času ve formátu Unix time stamp. Pro přístup k jednotlivým sekcím souborů ve formátu .NET lze využít stejnojmenný modul *Dotnet*. *Hash* modul umožňuje výpočet hashů ve standardních formátech MD5, SHA1 a SHA256 a dále výpočet kontrolních součtů.

3.3 Výkonnost

Při testování velkého objemu dat (např. celého systémového disku) s velkou sadou pravidel začne být významné pravidla optimalizovat pro výkon nebo využití paměti.

Při hlubším pohledu do fungování YARA je možné zjistit, že z řetězců obsažených v pravidlech jsou vytvořeny tzv. *atomy*, což jsou maximálně 4bajtové hodnoty vybírané dle běžnosti jejich výskytu. Ty jsou pak použity v automatu vytvořeném na základě algoritmu Aho-Corasicková, který atomy vyhledává ve zkoumaném souboru. Pokud je atom nalezen, algoritmus pokračuje v ověření, zda se v místě nachází celý hledaný řetězec. Až v posledním kroku jsou vyhodnoceny podmínky. [30]

3.3.1 Moduly

Moduly často nabízejí snadný a dobře čitelný přístup k jednotlivým polím v různých souborových formátech. Pro zvýšení výkonnosti pravidel je však doporučováno přistupovat k těmto polím manuálně. Příkladem je velmi využívaná detekce spustitelného souboru skrz hodnotu „MZ“. Té je možné dosáhnout díky PE modulu skrz podmínku *pe.is_pe*, anebo manuálně porovnat první dva bajty souboru podmínkou *uint16(0) == 0x5A4D*, což povede k rychlejšímu vyhodnocení.

3.3.2 Nevhodné řetězce

Obecně je nevhodné používat řetězce kratší než 4 bajty, protože jejich výskyt v běžných souborech bude značný, což povede k mnoha falešným pozitivitám. Stejný problém nastává u příliš uniformních řetězců. Uniformitou jsou zde myšleny řetězce, kde se vyskytuje opakovaně pouze jeden znak. Taktéž je vhodné vyhnout se nadužívání modifikátoru *nocase*. Ten způsobí, že budou vyhledávány všechny kombinace malých a velkých písmen v řetězci. Pokud existuje rozumné množství variant s odlišnou velikostí písmen, je lepší vypsat je jednotlivě.

3.3.3 Cykly

Ačkoliv YARA nativně podporuje smyčky, není vhodné je nadužívat, obzvlášť v kombinaci s příliš krátkými nebo obecnými řetězci. Počet iterací by měl být smysluplně omezen a jejich počet by neměl záviset na proměnlivých faktorech jako velikost souboru.

3.3.4 Regulární výrazy

Použití regulární výrazů by z hlediska výkonnosti měla být až poslední možnost. Obzvlášť důležité je vyhnout se tzv. greedy konstrukcím, jako „.*“ odpovídajícím libovolně dlouhým řetězcům. V případě takových konstrukcí je nanejvýš vhodné stanovit horní a případně i dolní mez velikosti řetězce. V opačném případě může být výsledkem chybové hlášení oznamující příliš mnoho shod.

3.3.5 Vyhodnocování podmínek

Podmínky jsou vyhodnocovány postupně zleva doprava a pokud je jedna z nich nepravdivá, vyhodnocování je ukončeno. Proto je vhodné na první místa umisťovat podmínky, které jsou buďto nenáročné na vyhodnocení, anebo je vysoká šance, že jejich výsledek bude nepravda. Je tedy například žádoucí jako první porovnávat velikost souboru nebo přítomnost bajtů na určitém místě a až následně počítat entropii celého souboru.

3.4 Nástroje

Významným pomocníkem při tvorbě, testování a správě pravidel mohou být různorodé nástroje, které jsou vytvořené komunitou a tedy volně k použití.

3.4.1 yarGen

Nástroj yarGen[s] nejdřív extrahuje řetězce z předloženého souboru a odstraňuje ty, které se nachází v databázi řetězců nezávadného softwaru, čímž snižuje možnost falešně pozitivních nálezů na minimum.

Řetězce ve výsledném pravidlu jsou rozděleny do tří kategorií na základě pravděpodobnosti, že se jedná o indikátory malware. Ty jsou označené $s, $x a $z. Názvy řetězců začínající $s jsou „vysoce specifické řetězce“, které se v legitimním softwaru neobjeví. Tyto řetězce mohou obsahovat adresy škodlivých serverů, názvy hackerských nástrojů a malwaru, výstupy hackerských nástrojů a překlepy v běžných řetězcích, které jsou někdy cíleně používány ve známých názvech pro oklamání oběti. Řetězce začínající na $x jsou „specifické řetězce“, které jsou pravděpodobně indikátory malwarových souborů, ale mohou se objevit i v legitimních souborech. A konečně řetězce začínající na $z jsou pravděpodobně běžné, ale v současné době nejsou zahrnuty do databáze řetězců nezávadného softwaru. YarGen následně používá v sekci podmínek kombinaci magické hlavičky, velikosti souboru a již zmíněných řetězců. [31]

3.4.2 YARA Forge

YARA Forge[t] automatizuje získávání, zpracování, optimalizaci a distribuci pravidel pro YARA z veřejných repozitářů sdílených různými organizacemi a jednotlivci.

V první fázi jsou pravidla kopírována z vybraných repozitářů. Následně jsou zpracovány do standardizovaného formátu, metadata jsou sjednocena a každému je přiřazen unikátní identifikátor. Taktéž jsou odstraněny duplicitní pravidla. Dále je hodnocena kvalita pravidla na základě jeho nedostatků a závažnosti detekované hrozby. Snížení hodnocení kvality může proběhnout z důvodu nedostatečné výkonnosti, logických chyb nebo vysokého procenta falešně pozitivních nálezů.

Výstupem jsou balíčky pravidel distribuovaných ve třech sadách. Základní (Core) obsahující pravidla s vysokou přesností a nízkým procentem falešných pozitivit. Rozšířená (Extended) přidávající do základní sady obecnější pravidla pro širší pokrytí detekcemi. A plná (Full) sada, kde je upřednostňována šířka pokrytí na úkor falešných pozitivit. [32]

[s] `https://github.com/Neo23x0/yarGen`
[t] `https://github.com/YARAHQ/yara-forge`

3.4.3 YARA-CI

YARA-CI je rozšíření pro kontinuální integraci (continuous integration), které je možné velmi snadno nainstalovat do GitHub repozitáře. To po vložení nového kódu do repozitáře validuje všechna přítomná YARA pravidla z hlediska syntaxe a výkonnosti. Následně jsou pravidla vyhodnocena vůči referenční databázi více než milionu validních souborů poskytovaných projektem NSRL[u], pro nalezení případných falešných pozitivit. Zároveň jsou pravidla spuštěna proti vzorkům malware stažených z databáze VirusTotal. Hash hodnoty vzorků, které mají být staženy, postačí vložit do sekce metadat se jménem záznamu začínajícím na „hash“, „sha1“, „sha256“ nebo „md5“. [33] Nevýhodou ovšem je nemožnost získat jednotlivé vzorky pro manuální testování, neboť VirusTotal ani NSRL neumožňují jejich volné stažení. Jedinou možností je tedy využít alternativní zdroje, které budou popisovány v části 5.1.

3.4.4 Yara Toolkit

Nejnovější sadou nástrojů je Yara Toolkit[v], která začala vznikat teprve na začátku roku 2024 v rámci iniciativy „#100DaysOfYara“[w], ve které se členové komunity snaží prvních sto dnů v roce libovolným způsobem přispět k vylepšení projektu YARA.

Součástí sady nástrojů je snadný a funkční editor schopný kontroly syntaxe. Dále nástroj pro zjednodušenou tvorbu pravidel a skener, který dokáže vytvořená pravidla vyhodnotit vůči nahranému souboru. V neposlední řadě je možné využít vyhledávač pravidel z několika veřejných repozitářů a také několik dalších přejatých nástrojů určených k práci s řetězci, nebo ke generování specifických druhů pravidel.

[u] https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl

[v] https://yaratoolkit.securitybreak.io/

[w] https://twitter.com/hashtag/100DaysOfYara

Kapitola 4

Sigma

„Sigma je pro logy to, co Snort pro síťový provoz a YARA pro soubory.“ [34]

Sigma je standardizovaný formát používaný k převodu indikátorů kompromitace (IOC) na detekční pravidla pro analýzu logů používaný v SIEM nebo EDR systémech.

Formát vydal v roce 2017 Florian Roth[x] ze společnosti Nextron Systems[y] s cílem otevřeného sdílení pravidel detekujících chování malware v souborech protokolu, které budou dále nazývány *logy* (z angl. logs).

Sigma umožňuje vytvářet pravidla na základě různých systémů generujících logy. Ty následně pomáhají identifikovat specifické vzory nebo chování svědčící o škodlivých aktivitách. Pravidla mohou obsahovat podmínky, filtry a další parametry, které určují, kdy má událost vyvolat upozornění. Jsou zapisována ve formátu YAML a lze je používat s různými bezpečnostními nástroji a platformami. To je jedna z největších výhod tohoto formátu, neboť pomocí nástroje *sigma-cli*[z] je možné pravidla velmi snadno převádět do specifických formátů používaných v jednotlivých systémech. Těmi může být např. ElasticSearch, Splunk, QRadar a mnoho dalších.

4.1 Formát

Sigma pravidlo je rozděleno do třech základní sekcí, které obsahují různá pole. Sekce detekce (detection) popisuje chování malware, dle kterého probíhá vyhledávání. Zdroj logu (logsource) definuje systém a případně nástroj, ze kterého soubor událostí pochází. Metadata pak obsahují všechny identifikační i jiné informace o pravidle a jeho autorovi. Všechny popisy v této podkapitole vychází z oficiální dokumentace. [35]

[x] `https://github.com/Neo23x0`

[y] `https://www.nextron-systems.com/`

[z] `https://github.com/SigmaHQ/sigma-cli`

```
title: Example rule in Sigma format
id: f5d09a4e-9a9b-4d3e-b1fa-92559c87d323
status: test
description: Detects ...
references:
    - https://github.com/SigmaHQ/sigma-specification/tree/main
author: Name
date: 2024-04-20
tags:
    - attack.defense_evasion
    - attack.t1218.005
    - car-2013-04-002
logsource:
    product: windows
detection:
    keywords:
        - 'first string' #or
        - 'another string' #or
        - 'test string'
    field:
        EventID: 6416  # and where
        ClassName: 'DiskDrive'
    fieldList:
        EventID:
            - 4728  # or where
            - 4729  # or where
            - 4730
    modif:
        ParentImage|endswith: '\svchost.exe'
        Image|endswith: '\mshta.exe'
    condition: (keywords or field or fieldList) and not modif
falsepositives:
    - Unknown
level: high
```

Obrázek 4.1: Ukázkové pravidlo ve formátu Sigma

4.1.1 Detekce

Definice detekce jsou rozděleny do tzv. *výběrů* (selections), kdy každý výběr popisuje jeden konkrétní indikátor kompromitace, kterých může pravidlo obsahovat více najednou. Výběry je možné zapsat hned třemi různými způsoby.

Nejjednodušší z nich je pomocí *klíčových slov* (keyword), kdy jsou pouze porovnávány hledané řetězce se všemi daty v cílovém logu. Ve výběru může být klíčových slov více, stačí je umístit do seznamu, ve kterém každá položka začíná pomlčkou na novém řádku. K vyvolání upozornění pak stačí, aby byl nalezen pouze jediný z nich.

Druhým způsobem je zápis dle hodnoty *pole* (field), odkazující na konkrétní hodnoty ve struktuře cílového logu. V ukázkovém pravidle se jedná o připojení externího zařízení s konkrétním názvem zaznamenaného do Windows logu událostí.

Poslední způsob zápisu *seznamem polí* (field list) je velmi podobný tomu předchozímu, umožňuje však sledovat více událostí najednou. Ty jsou opět vypsány jako odrážkový seznam s každou položkou na novém řádku, kde stačí výskyt pouze jedné z nich. V ukázkovém příkladě je možné vidět zápis událostí v systému Windows dle jejich tzv. event ID.

Jednotlivé výběry je pak možné mezi sebou vyhodnocovat logickými operátory v části určené pro podmínky (condition). [35]

4.1.2 Zdroj logu

Aby bylo každé pravidlo Sigma účinné při detekci, je důležité určit, jaký typ protokolů má SIEM prohledávat. To je důležité nejen pro efektivitu detekce, ale také pro zajištění toho, aby detekce probíhala podle správné sady polí a použitých hodnot.

Každá definice zdroje logů v rámci pravidla Sigma se skládá ze tří samostatných polí, která mohou být kombinována, čímž přesněji definují daný zdroj logu. Tím je zajištěno, že detekce bude zaměřena pouze na konkrétní sadu logů relevantních pro danou detekci.

Pole *kategorie* (category) slouží k výběru všech logů zapisovaných určitou skupinou produktů. Může se tedy jedna o logy z firewallů, webových serverů, z práce se soubory nebo s registry apod.

Produkt (product) je pole určené k výběru všech logů jednoho konkrétního produktu. Tím můžou být například všechny typy logů systému Windows, Linux nebo různých produktů společnosti Microsoft.

Pole *služba* (service) dokáže vybrat pouze podmnožinu logů z konkrétního produktu. V logu událostí systému Windows to tak může být například „Zabezpečení“, „Systém“, „Aplikace“, anebo nové typy logů, jako jsou „AppLocker“, „Windows Defender“ atd. V systému Linux se může jednat například o „sshd“, „syslog“ apod.

Ačkoliv je možné používat jakýkoliv zdroj logu, existuje seznam standardních zdrojů, které jsou obecně podporované. Ten je možné také naleznout v příslušné části oficiální dokumentace. [36]

Modifikátory

Pomocí modifikátorů je možné přesněji specifikovat chování a vlastnosti detekce v kontextu hodnot jednotlivých polí. Jak je vidět v ukázkovém pravidle, za názvem konkrétního pole lze vložit svislítko „|“ a za něj pak samotný modifikátor. Zde například *endswith*, určující, že řetězec má být hledaný na konci zadaného pole. Kromě opačného modifikátoru *startswith*, lze využít i univerzálnější *contains* hledající hodnotu kdekoliv v cílovém poli. Pro přímý zápis regulárního výrazu je pak možné využít modifikátor *re*.

Kromě standardních relačních operátorů a možnosti porovnávat řetězce zakódované do formátu base64 je k dispozici i modifikátor *cidr*, umožňující zápis IP adres včetně masky sítě za lomítkem. Následná detekce je vyhodnocována oproti všem IP adresám z vymezeného rozsahu. [45]

4.1.3 Metadata

Všechna ostatní data vyskytující se v Sigma pravidlech jsou součástí sekce metadata. Jediný povinný atribut je jeho *název* (title), který by měl velmi krátce shrnout čeho se pravidlo snaží dosáhnout. Více informací je možné vložit do pole *popisu* (description). Odkazy na blogy, výzkumné články nebo dokumentaci je vhodné vkládat do atributu *reference*. Jméno autora, typ licence a datum vytvoření nebo modifikace pravidla je taktéž možné vložit do příslušných atributů. *Level* určuje závažnost detekované hrozby. Pro popis situací, ve kterých může pravidlo vyvolávat falešně pozitivní náhledy, slouží atribut *falsepositives*. Každé pravidlo by také mělo být opatřeno unikátním identifikátorem označeným jako *id*, ve formátu UUIDv4, který lze náhodně vygenerovat.

Pravidlo lze také opatřit *tagy* (tags), které ho kategorizují do standardizovaných rámců zabezpečení. Konkrétně se jedná o frameworky:

MITRE ATT&CK popisující taktiky a techniky používané útočníky při kybernetických útocích.

MITRE Cyber Analytics Repository je platforma poskytující kolekci analytických technik a pravidel pro detekci kybernetických hrozeb.

Traffic Light Protocol je systém označování citlivosti informací, který pomáhá určit úroveň citlivosti a vhodnost pro sdílení těchto informací s ostatními subjekty.

Common Vulnerabilities and Exposures je standardizovaný identifikátor používaný k jednoznačné identifikaci a popisu zranitelností v softwaru a hardwaru. [35]

Kapitola 5

Návrh pravidel

5.1 Popis vzorků ransomware

K analýze bylo vybráno 195 vzorků celkem 24 variant ransomware, které byly v posledním roce pozorovány jako aktivní. Vzorky ransomware byly stahovány z webů tri.age[aa] a any.run[ab]. Při vytváření YARA pravidel s pomocí YARA-CI byly testovány i vzorky, které není možné volně získat k ruční analýze. V takových případech bylo vycházeno z podrobných analýz databáze Hybrid Analysis[ac] využívající Falcon Sandbox.

Největší analyzovaný vzorek měl velikost 12.54 MB, nicméně vzhledem k tomu, že cílem byla obecná detekce ransomware, nebyla ve vytvořených pravidlech nijak omezována velikost skenovaných souborů.

Následující seznam nabízí krátký popis některých z rodin ransomware. Kompletní seznam všech vzorků je možné nalézt v přiloženém archivu.

Hitobito Vzorky ransomwaru Hitobito jsou k dispozici od konce března 2024. Stejně jako většina ransomware i tento ransomware šifruje soubory a za jejich dešifrování požaduje výkupné prostřednictvím ransom note. Skupina používá ke komunikaci s obětmi TOR a pravděpodobně, nekrade žádná data. [37]

Abyss Locker Ačkoli první vzorek Abyss Locker byl veřejně dostupný v červenci 2023, první varianta ransomwaru může pocházet z dřívější doby, protože ransomware vychází ze zdrojového kódu ransomwaru HelloKitty. Abyss Locker před svým spuštěním krade data obětí. Tento ransomware je také schopen odstranit stínové kopie a zálohy systému. [38]

RA World Ransomware RA World byl poprvé zveřejněn na začátku prosince 2023. Útočník před nasazením a spuštěním svého ransomware nejprve

[aa] `https://tria.ge`
[ab] `https://app.any.run`
[ac] `https://hybrid-analysis.com`

ukradne data obětí.Skupina provozuje weby na síti TOR i na standardních webech. Ransomware je také navržen tak, aby mazal stínové kopie. [39]

Albabat Albabat, známý také jako White Bat, je finančně motivovaná varianta ransomwaru napsaná v jazyce Rust, která identifikuje a zašifruje soubory důležité pro uživatele a požaduje výkupné za jejich uvolnění. Poprvé se objevil v listopadu 2023 s variantou verze 0.1.0. Koncem prosince byla vydána verze 0.3.0 a v polovině ledna 2024 následovala verze 0.3.3. [40]

Faust Faust je ransomware z rodiny Phobos, který se šíří skrze dokument Office obsahující skript VBA. Útočníci využili službu Gitea k uložení několika souborů zakódovaných v Base64, z nichž každý nese škodlivou binárku. Po vložení těchto souborů do paměti systému zahájí šifrování souborů. [41]

NoEscape NoEscape je finančně motivovaná skupina provozující ransomware jako službu. Skupina má na svědomí mnoho obětí v různých odvětvích, včetně vládního sektoru, energetiky, nemocnic a lékařských klinik. Předpokládá se, že skupina ransomwaru NoEscape souvisí s již neexistující skupinou ransomwaru Avaddon. [42]

Knight Knight je skupina, která se objevila v srpnu 2023. Stejně jako mnoho útočníků používá gang stojící za touto variantou taktiku dvojitého vydírání, kdy ransomware Knight šifruje soubory na počítačích obětí a exfiltruje data pro účely vydírání. Předchůdce Knight byl ransomware Cyclops. [43]

Akira Akira je varianta ransomware s verzemi pro Windows a Linux, která vyšla v dubnu 2023. Používá také taktiku dvojitého vydírání, požadující od obětí výkupné výměnou za dešifrování souborů a neprozrazení ukradených informací veřejnosti. [44]

5.2 Testovací prostředí

K vývoji pravidel ve formátu YARA bylo využito rozšíření *YARA-CI* a online editor ze sady *Yara toolkit*. Pro statickou analýzu a detekci zabalení nebo obfuskací souborů byl využit nástroj *Detect It Easy*.

K vytvoření testovacích dat pro vývoj Sigma pravidel byl z každé rodiny vybrán jeden vzorek, který byl spuštěn v laboratorním prostředí. Ke spuštění bylo využíváno virtuální prostředí VMware Workstation s nainstalovaným OS Windows 10 v sestavení 17763. Chování bylo sledováno pomocí nástroje

Sysmon[ad] s konfiguračním souborem[ae], zaměřeným na maximální zaprotokolování všech událostí vyvolaných ransomware. Extrakce protokolů ve formátu .evtx probíhala nejprve spuštěním skriptu[af], kterým byly logy zabaleny do jednoho archivu a následně manuálním vyjmutím z virtuálního stroje. Testování pravidel Sigma nad extrahovanými soubory bylo následně prováděno pomocí nástroje *Chainsaw*[ag].

K vyhledání falešně pozitivních nálezů byly využity vzorové logy systému Windows 10 z projektu *evtx-baseline*[ah].

Vzorky ransomware Faust se v testovacím prostředí nepodařilo spustit. Důvodem jsou zřejmě specifické vazby na infikovaný soubor maker ve formátu *xlam*.

5.3 Ransom note

Podrobnou analýzou vzorků bylo zjištěno, že kromě informací popsaných v kapitole 1.2.6 patří mezi pokyny sdělované obětem výzva ke komunikaci skrze open-source službu pro zasílání zpráv qTox[ai]. Výjimkou však není ani využití komunikační platformy Telegram. Ačkoliv všechny řetězce jsou cíleně vybírány z obsahu ransom note, detekce je někdy úspěšná na základě nálezů např. v metadatech souboru, které byly v programu útočníkem zapomenuty. Příkladem zde může být ransomware S.H.O., který obsahuje cestu k debbugovacímu souboru obsahující řetězec „ransomware-master“.

Řetězce byly rozděleny do dvou skupin. První skupina „x“ obsahuje specifické řetězce vyskytující se v ransom note, z nichž stačí, aby cílový soubor obsahoval jeden z nich. Druhá skupina „s“ jsou pak obecnější řetězce, ze kterých musí soubor obsahovat alespoň tři.

Úspěšně detekovány byly vzorky celkem 14 rodin, konkrétně: Hitobito, Abyss Locker, RA World, Albabat, Akira, Retch, S.H.O., NoCry, DoDo, Rancoz, Buddy, Big Head, UNIZA a Kadavro.

Porovnáním s databází NSRL byly nalezeny 2 falešně pozitivní soubory. Přestože se jedná o validní soubory, jeden z nich byl i analýzou sandboxem Falcon vyhodnocen jako podezřelý[aj], k druhému se nepodařilo získat další podrobnosti.

[ad] `https://learn.microsoft.com/cs-cz/sysinternals/downloads/sysmon`

[ae] `https://github.com/bobby-tablez/Enable-All-The-Logs`

[af] `https://github.com/NextronSystems/evtx-baseline?tab=readme-ov-file#export-the-event-logs`

[ag] `https://github.com/WithSecureLabs/chainsaw`

[ah] `https://github.com/NextronSystems/evtx-baseline`

[ai] `https://qtox.github.io/`

[aj] `https://hybrid-analysis.com/sample/daa61c42d53be45c7709a0b0f66a51a0a47ca84eab787e0627f6da255c96ddff/6638e1069be5ef74bc0abbd0`

```
rule SUSP_RANSOM_Generic_ransom_note
{
    meta:
        description = "Detects the content of ransom notes"
        author = "Stanislav Lepic"
        date = "2024-05-05"
        reference = "Internal Research"
    strings:
        $x1 = "https://www.torproject.org/" ascii wide
        $x2 = "https://tox.chat/" ascii wide
        $x3 = ".onion" ascii wide
        $x4 = "ransomware" ascii wide
        $x5 = "Ransomware" ascii wide
        $x6 = "RANSOMWARE" ascii wide
        $x7 = "qTox" fullword ascii wide
        $x8 = "https://t.me/" ascii wide
        $x9 = "hacked" fullword ascii wide
        $x10 = "decrypt your files" ascii wide
        $s1 = "Bitcoin" fullword ascii wide
        $s2 = "BTC" fullword ascii wide
        $s3 = "Btc" fullword ascii wide
        $s4 = "btc" fullword ascii wide
        $s5 = "TOR" fullword ascii wide
        $s6 = "Tor" fullword ascii wide
        $s7 = "tor" fullword ascii wide
        $s8 = "tor browser" ascii wide
        $s9 = "Tor Browser" ascii wide
        $s10 = "TOR Browser" ascii wide
        $s11 = "TOX" fullword ascii wide
        $s12 = "Tox" fullword ascii wide
        $s13 = "tox" fullword ascii wide
        $s14 = "recover" ascii wide
        $s15 = "pay" fullword ascii wide
        $s16 = "crypt" ascii wide
        $s17 = "CRYPT" ascii wide
        $s18 = "leak" ascii wide
        $s19 = "ransom" ascii wide
        $s20 = "Ransom" ascii wide
        $s21 = "RANSOM" ascii wide
    condition:
       uint16(0) == 0x5a4d and (1 of ($x*) and 3 of ($s*))
}
```

Obrázek 5.1: Pravidlo detekující obsah ransom note

```
rule SUSP_RANSOM_Generic_ransom_note_xored
{
    meta:
        description = "Detects the content of ransom notes"
        author = "Stanislav Lepic"
        date = "2024-05-06"
        reference = "Internal Research"
    strings:
        $x1 = "https://www.torproject.org/" ascii wide xor...
        $x2 = "https://tox.chat/" ascii wide xor(0x01-0xff)
        $x3 = "https://t.me/" ascii wide xor(0x01-0xff)
        $x4 = "decrypt your files" ascii wide xor(0x01-0xff)

    condition:
  uint16(0) == 0x5a4d and (1 of ($x*))
}
```

Obrázek 5.2: Pravidlo detekující XORovaný obsah ransom note

5.3.1 Obfuskace xorováním

Mezi nejjednodušší obfuskace patří tzv. xorování, při kterém je na zdrojové řetězce, jež mají být zatemněny, aplikována logická operace XOR s 1bajtovým klíčem. YARA umožňuje takto upravené řetězce vyhledat klíčovým slovem *xor*, za které je možné dodat rozsah všech klíčů, které mají být při detekci otestovány.

Jako základ návrhu bylo využito pravidlo 5.1. Vzhledem k velkému množství variant, které jsou zde testovány, však není možné používat příliš krátké nebo obecné řetězce, neboť vedou k velmi vysokému počtu falešně pozitivních nálezů. Proto bylo využito jen několik unikátních řetězců dostatečné délky. Pravidlo je účinné pouze na vzorky obfuskovány touto konkrétní technikou.

Detekovány byly pouze vzorky ransomware Darkrace. Nebyly nalezeny žádné falešně pozitivní soubory.

5.4 Zabránění obnovení systému

Jak již bylo naznačeno v kapitole 1.2.4 jedním z hlavních způsobů obnovy, na který spoléhá operační systém Windows jsou *stínové kopie*. Ty lze jednoduše smazat pomocí příkazů „vssadmin.exe delete shadows“ nebo „wmic.exe shadowcopy delete“. Využít lze i změnu velikosti těchto kopií, čímž jsou de facto znefunkčněny.

Smazání kopií lze dosáhnout i přímým přístupem k WMI objektům pomocí

příkazů v prostředí PowerShell. Jedním z příkladů zde může být spuštění „Get-WmiObject Win32_ShadowCopy | Remove-WmiObject“ [46]

Velmi populární technikou, které je možné dosáhnout příkazem „bcdedit.exe /set default bootstatuspolicy ignoreallfailures“, je změna nastavení zásad zavádění systému tak, aby byly ignorovány chyby při procesu spuštění. To vede k nemožnosti vstoupit do prostředí obnovy (Windows Recovery Environment). Velmi podobně pak funguje příkaz „bcdedit.exe /set default recoveryenabled no“, který systém obnovy zakazuje zcela.

Další z metod je deaktivace pravidelně spouštěné úlohy Obnovy systému, která je standardně využívána k vytvoření bodu obnovy, sloužícímu k návratu do stavu před určitou událostí. Standardně zneužívaným příkazem je zde „schtasks.exe /Change /TN "\Microsoft\Windows\SystemRestore\SR" /disable“.

Systém obnovy je také možné deaktivovat pomocí registrů. Konkrétně „HKLM\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore“ a HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestore je nutné nastavit, aby obsahovaly hodnoty „DisableConfig“ a „DisableSR“ s nastavenou jedničkou. [47]

Pravidlo úspěšně nalezlo shodu u vzorků ransomware Abyss Locker, RA World, Akira, DoDo, Rancoz, Buddy a Big Head. V porovnání s databází NSRL bylo nalezeno 8 falešně pozitivních souborů. Vzhledem k tomu, že jde o soubory jazykové lokalizace VSSADMIN.EXE.MUI a WBADMIN.EXE.MUI, lze to považovat za očekávatelné chování.

5.5 Deaktivace logování

Nevýhodou zvoleného testovací přístupu je situace, kdy ransomware potlačí schopnost systému Windows zaznamenávat proběhlé události. V takový moment již dále není možné sledovat chování škodlivého souboru pouze na základě extrahovaných logů. Samotné vypnutí, kterého může být docíleno mnoha způsoby, však zaznamenáno je. Pravidlo 5.4 detekuje několik způsobů, které k deaktivaci využívají registry systému Windows.

Prvním z nich je velmi jednoduchá metoda vytvoření prázdného klíče registru HKLM\SYSTEM\CurrentControlSet\Control\MiniNt, díky kterému bude po restartování systému pozastaveno zaznamenávání všech událostí a zároveň znepřístupněn samotný *Prohlížeč událostí*. To je způsobeno tím, že systém Windows je přesvědčen o svém spuštění v režimu Windows Preinstallation Environment.

Klíč HKLM\SYSTEM\CurrentControlSet\Services\EventLog\start indikuje, zda Prohlížeč událostí aktivně zaznamenává informace. Nastavení hodnoty na „0x00000004“, které může proběhnout pomocí nástrojů jako *Wevtutil* nebo *auditpol*, případně ručním zásahem, signalizuje vypnutí logování.

```
rule SUSP_Backups_del
{
    meta:
        description = "Detects deletion of system backups"
        author = "Stanislav Lepic"
        date = "2024-05-06"
        reference = "https://atomicredteam.io/impact/T1490/"
        reference = "https://www.fortinet.com/blog/threat-research/stomping-
        shadow-copies-a-second-look-into-deletion-methods"
    strings:
        //vssadmin
        $vss1 = "vssadmin delete shadows /all" ascii wide nocase
        $vss2 = "vssadmin.exe delete shadows /all" ascii wide nocase
        $vss3 = "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=" ...
        $vss4 = "vssadmin.exe resize shadowstorage /for=c: /on=c: /maxsize=" ...
        //wmic
        $wmic1 = "wmic shadowcopy delete" ascii wide nocase
        $wmic2 = "wmic.exe shadowcopy delete" ascii wide nocase
        $wmic3 = "wmic shadowcopy /nointeractive" ascii wide nocase
        $wmic4 = "wmic.exe shadowcopy /nointeractive" ascii wide nocase
        $wmic5 = "shadowcopy where \"ID='%s'\" delete" ascii wide nocase
        //wbadmin
        $wba1 = "wbadmin delete backup" ascii wide nocase
        $wba2 = "wbadmin.exe delete backup" ascii wide nocase
        $wba3 = "delete catalog -quiet" ascii wide nocase
        $wba4 = "delete systemstatebackup" ascii wide nocase
        //bcdedit
        $bcd1 = "} bootstatuspolicy ignoreallfailures" ascii wide nocase
        $bcd2 = "} recoveryenabled no" ascii wide nocase
        //schtasks
        $scht1 = "\\Microsoft\\Windows\\SystemRestore\\SR\" /disable" ...
        //powershell
        $ps1 = "Win32_ShadowCopy | % { $_.Delete() }" ascii wide nocase
        $ps2 = "Win32_Shadowcopy | ForEach-Object {$_.Delete();}" ...
        $ps3 = "Win32_ShadowCopy | Remove-WmiObject" ascii wide nocase
        $ps4 = "Win32_ShadowCopy | Remove-CimInstance" ascii wide nocase
        //regkeys
        $reg1 = "SOFTWARE\\Policies\\Microsoft\\Windows NT\\SystemRestore\"
        /v \"Disable" ascii wide nocase
        $reg2 = "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\
        SystemRestore\" /v \"Disable" ascii wide nocase

    condition:
uint16(0) == 0x5a4d and 1 of them
}
```

Obrázek 5.3: Pravidlo detekující zabránění obnovení systému

```
title: Disable Windows Logging Using the Registry
id: fb9570a2-3aba-4721-bdde-0d84bc9174d3
status: test
description: Detect various methods of deactivating windows logs using registry.
references:
    - https://ptylu.github.io/content/report/report.html?report=25
author: Stanislav Lepic
date: 2024/05/08
tags:
    - attack.defense_evasion
    - attack.t1562.002
    - CAR.2022.03.001
logsource:
    category: registry_event
    product: windows
detection:
    minint:
        TargetObject|endswith: '\CurrentControlSet\Control\MiniNt'
        EventType: 'CreateKey'
    evenlog:
        TargetObject|endswith: '\CurrentControlSet\Services\EventLog'
        Details: 'DWORD (0x00000004)'
        EventType: 'SetValue'
    channels:
        TargetObject|contains: '\Microsoft\Windows\CurrentVersion\WINEVT\Channels\'
        TargetObject|endswith: '\Enabled'
        Details: 'DWORD (0x00000000)'
        EventType: 'SetValue'
    filter_wevutil:
        Image: 'C:\Windows\system32\wevtutil.exe'

    condition: (minint or evenlog or channels) and not filter_wevutil

level: medium
```

Obrázek 5.4: Pravidlo detekující vypnutí logování

Další metodou je deaktivace jednotlivých kanálů, ze kterých Windows shromažďují logy. Těch existují stovky a ve struktuře registrů jsou umístěné pod HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels*. [48]

Detekovány byly dle očekávání vzorky ransomware Lockbit 3.0. Falešně pozitivních nálezů bylo zjištěno 5. Všechny se týkaly použití legitimního nástroje *wevtutil.exe*. Na základě této skutečnosti byla do pravidla přidána výjimka ignorující akce z tohoto nástroje.

Závěr

Cílem práce bylo analyzovat různorodé vzorky ransomware a identifikovat jejich typické chování, na základě kterého by bylo možné vybudovat efektivní detekce. Obecný popis chování ransomware není však jednoduchý, neboť každá rodina používá mírně odlišný princip svého fungování nebo využívá pouze vybrané metody.

V teoretické části byly nejdříve obecně popsány typy ransomware, způsoby jeho analýzy a práce s formáty YARA a Sigma. Praktická část pak ukázala volně dostupné nástroje a služby, pomocí kterých je možné s analýzou a vývojem pravidel začít. V těchto ohledech se ukázala značná nevyspělost formátu Sigma, pro který neexistuje mnoho nekomerčních nástrojů, které by usnadňovaly vývoj. Výsledkem je několik pravidel ve standardizovaných formátech, která poskytují detekce nezávislé na konkrétní rodině nebo druhu ransomware. Ty detekují obsah ransom note, pokusy o znefunkčnění systémových záloh nebo vypnutí logování.

Na práci by bylo možné navázat rozšířením o další pravidla zaměřující se na typické ukazatele, jako je snaha ransomware o dosažení perzistence, vysoká entropie zašifrovaných souborů nebo použití specifických funkcí a knihoven. Taktéž by bylo možné zaměřit se na síťovou komunikaci a případná detekční pravidla pro nástroj Snort.

Literatura

[1] RAZAULLA, Salva, Claudie FACHKHA, Christine MARKARIAN, Amjad GAWANMEH, Wathiq MANSOOR, Benjamin C. M. FUNG a Chadi ASSI. The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions [online]. 2023 [cit. 2024-03-31]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10105244

[2] LOGRYTHM LABS. NOTPETYA TECHNICAL ANALYSIS [online]. 2017 [cit. 2024-03-31]. Dostupné z: https://gallery.logrhythm.com/threat-intelligence-reports/notpetya-technical-analysis-logrhythm-labs-threat-intelligence-report.pdf

[3] BAHRAMI, Pooneh Nikkhah, Ali DEHGHANTANHA, Tooska DARGAHI, Reza M. PARIZI, Kim-Kwang Raymond CHOO a Hamid H. S. JAVADI. Cyber Kill Chain-Based Taxonomy of Advanced Persistent Threat Actors: Analogy of Tactics, Techniques, and Procedures [online]. 2019 [cit. 2024-04-05]. Dostupné z: https://koreascience.kr/article/JAKO201925462478086.pdf

[4] MARCO, Lucia Di. Persistence deployment automation [online]. 2022 [cit. 2024-04-05]. Dostupné z: https://upcommons.upc.edu/bitstream/handle/2117/363670/164219.pdf

[5] LOUI, Eric, Karl SCHEUERMAN, Aaron PICKETT a Brendon FEELEY. Targeted Dharma Ransomware Intrusions Exhibit Consistent Techniques [online]. 2020 [cit. 2024-04-05]. Dostupné z: https://www.crowdstrike.com/blog/targeted-dharma-ransomware-intrusions-exhibit-consistent-techniques/

[6] AKBANOV, Maxat, Vassilios G. VASSILAKIS a Michael D. LOGOTHETIS. WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms [online]. 2019 [cit. 2024-

04-05]. Dostupné z: https://www.researchgate.net/publication/332088162_WannaCry_Ransomware_Analysis_of_Infection_Persistence_Recovery_Prevention_and_Propagation_Mechanisms

[7] SYMANTEC CORPORATION. An ISTR Special Report: Ransomware and Businesses 2016 [online]. 2016 [cit. 2024-04-05]. Dostupné z: https://conferences.law.stanford.edu/cyberday/wp-content/uploads/sites/10/2016/10/5c_ISTR2016_Ransomware_and_Businesses.pdf

[8] LI, Adrian. An Analysis of the Recent Ransomware Families [online]. 2021 [cit. 2024-03-29]. Dostupné z: https://www.cs.purdue.edu/homes/li3944/blog/Project%20report_Adrian%20Li.pdf

[9] SYMANTEC CORPORATION. Data Exfiltration: Increasing Number of Tools Leveraged by Ransomware Attackers [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/ransomware-data-exfiltration

[10] OZ, Harun, Ahmet AIRS, Albert LEVI a Selcuk ULUAGAC. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions [online]. 2022 [cit. 2024-04-07]. Dostupné z: https://arxiv.org/pdf/2102.06249.pdf

[11] LEMMOU, Yassine, Jean-Louis LANET a El Mamoun SOUIDI. A behavioural in-depth analysis of ransomware infection [online]. 2020 [cit. 2024-04-07]. Dostupné z: https://doi.org/10.1049/ise2.12004

[12] DOSUMU, Richard. Evolving Trends In Ransomware: An Analysis of Attack Vectors, Mitigation Strategies, and Future Trends [online]. 2023 [cit. 2024-04-07]. Dostupné z: https://www.researchgate.net/publication/376523141_Evolving_Trends_In_Ransomware_An_Analysis_of_Attack_Vectors_Mitigation_Strategies_and_Future_Trends

[13] KHAN, Mohammed Rauf Ali. Understanding impacts of a ransomware on medical and health facilities by utilizing LockBit as a case study [online]. 2023 [cit. 2024-04-07]. Dostupné z: https://doi.org/10.1002/spy2.328

[14] MELAND, Per Håkon, Yara Fareed Fahmy BAYOUMY a Guttorm SINDRE. The Ransomware-as-a-Service economy within the darknet [online]. 2020 [cit. 2024-03-29]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167404820300468

[15] KEIJZER, Noël. The new generation of ransomware - An in depth study of Ransomware-as-a-Service [online]. 2020 [cit. 2024-03-29]. Dostupné z: http://essay.utwente.nl/81595/1/Keijzer_MA_EEMCS.pdf

[16] KREBS, Brian. Inside a 'Reveton' Ransomware Operation [online]. 2012 [cit. 2024-03-29]. Dostupné z: https://krebsonsecurity.com/2012/08/inside-a-reveton-ransomware-operation/

[17] BAKER, Kurt. Ransomware as a Service (RaaS) Explained How It Works & Examples [online]. 2023 [cit. 2024-03-29]. Dostupné z: https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-as-a-service-raas/

[18] EUROPOL. Law enforcement disrupt world's biggest ransomware operation [online]. 2024 [cit. 2024-03-30]. Dostupné z: https://www.europol.europa.eu/media-press/newsroom/news/law-enforcement-disrupt-worlds-biggest-ransomware-operation

[19] MICROSOFT. PE Format. 2024 [cit. 2024-04-12]. Dostupné z: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

[20] HAHN, Karsten. All your hashes are belong to us: An overview of malware hashing algorithms [online]. 2021 [cit. 2024-04-10]. Dostupné z: https://www.gdatasoftware.com/blog/2021/09/an-overview-of-malware-hashing-algorithms

[21] KRÁL, Benjamin. Forenzní analýza malware. 2018 [cit. 2024-04-12]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=180629

[22] ANAND, P. Mohan, P.V. Sai CHARAN a Sandeep K. SHUKLA. A Comprehensive API Call Analysis for Detecting Windows-Based Ransomware. 2022 [cit. 2024-04-14]. Dostupné z: https://ieeexplore.ieee.org/document/9850320

[23] ERMA, Mayank, Ponnurangam KUMARGURU, Shuva Brata DEB a Anuradha GUPTA. Analysing Indicator of Compromises for Ransomware: Leveraging IOCs with Machine Learning Techniques. 2018 [cit. 2024-04-14]. Dostupné z: https://ieeexplore.ieee.org/document/8587409

[24] COLLBERG, Christian, Clark David THOMBORSON a Douglas LOW. A Taxonomy of Obfuscating Transformations. 1997 [cit. 2024-04-16]. Dostupné z: https://researchspace.auckland.ac.nz/bitstream/handle/2292/3491/TR148.pdf

[25] FERRIE, Peter. The "Ultimate"Anti-Debugging Reference. 2011 [cit. 2024-04-15]. Dostupné z: https://anti-reversing.com/Downloads/Anti-Reversing/The_Ultimate_Anti-Reversing_Reference.pdf

[26] BALAJI, Priyadharshini. How to Perform Static Code Analysis on Packed Malware?. 2023 [cit. 2024-04-20]. Dostupné z: https://www.socinvestigation.com/how-to-perform-static-code-analysis-on-packed-malware/

[27] ROTH, Florian. YARA-Style-Guide. 2024 [cit. 2024-04-20]. Dostupné z: https://github.com/Neo23x0/YARA-Style-Guide

[28] VIRUSTOTAL. Writing YARA rules. 2024 [cit. 2024-04-20]. Dostupné z: https://yara.readthedocs.io/en/latest/writingrules.html

[29] VIRUSTOTAL. YARA modules. 2024 [cit. 2024-04-20]. Dostupné z: https://yara.readthedocs.io/en/stable/modules.html

[30] ROTH, Florian. YARA Performance Guidelines. 2023 [cit. 2024-04-21]. Dostupné z: https://github.com/Neo23x0/YARA-Performance-Guidelines/

[31] LI, Vickie. Malware Detection Using Yara And YarGen. 2021 [cit. 2024-04-21]. Dostupné z: https://sec.okta.com/articles/2021/08/malware-detection-using-yara-and-yargen

[32] ROTH, Florian. YARA Forge. [cit. 2024-04-21]. Dostupné z: https://yarahq.github.io/

[33] VIRUSTOTAL. What's YARA-CI. [cit. 2024-04-21]. Dostupné z: https://yara-ci.cloud.virustotal.com/

[34] SIGMAHQ. Sigma - Generic Signature Format for SIEM Systems. [cit. 2024-04-24]. Dostupné z: https://github.com/SigmaHQ/sigma

[35] SIGMAHQ. Sigma Rules. 2024 [cit. 2024-04-27]. Dostupné z: https://sigmahq.io/docs/basics/rules.html

[36] SIGMAHQ. Sigma Logsources. 2024 [cit. 2024-04-27]. Dostupné z: https://sigmahq.io/docs/basics/log-sources.html

[37] IMANO, Shunichi. Ransomware Roundup - KageNoHitobito and DoNex. 2024 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-keganohitobito-and-donex

[38] IMANO, Shunichi.Ransomware Roundup – Abyss Locker. 2024 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-abyss-locker

[39] IMANO, Shunichi. Ransomware Roundup – RA World. 2024 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-ra-world

[40] IMANO, Shunichi a Fred Gutierrez. Ransomware Roundup - Albabat. 2024 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-albabat

[41] LIN, Cara. Another Phobos Ransomware Variant Launches Attack – FAUST. 2024 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/phobos-ransomware-variant-launches-attack-faust

[42] IMANO, Shunichi a Fred Gutierrez. Ransomware Roundup – NoEscape. 2023 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-noescape

[43] IMANO, Shunichi. Ransomware Roundup - Knight. 2023 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-knight

[44] IMANO, Shunichi. Ransomware Roundup - Akira. 2023 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/ransomware-roundup-akira

[45] SIGMAHQ. Sigma Modifiers. 2024 [cit. 2024-04-29]. Dostupné z: https://sigmahq.io/docs/basics/modifiers.html

[46] HUNTER, Ben. Stomping Shadow Copies - A Second Look Into Deletion Methods. 2020 [cit. 2024-05-08]. Dostupné z: https://www.fortinet.com/blog/threat-research/stomping-shadow-copies-a-second-look-into-deletion-methods

[47] ATOMIC READ TEAM. Inhibit System Recovery. 2023 [cit. 2024-05-08]. Dostupné z: https://atomicredteam.io/impact/T1490/

[48] HEILIGENSTEIN, Lucas. REP-25: Disable Windows Event Logging. [cit. 2024-05-08]. Dostupné z: https://sigmahq.io/docs/basics/modifiers.html

Příloha A

Seznam použitých zkratek

API Application programming interface

ASCII American Standard Code for Information Interchange

C&C Command and control

CERT Computer Emergency Response Team

CFG Control Flow Graph

CI Continuous integration

CUI Character user interface

DLL Dynamic-link library

DNS Domain Name System

DOS Disk Operating System

EDR Endpoint Detection and Response

FTP File Transfer Protocol

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

IDPS Intrusion Detection and Prevention Systems

NSA National Security Agency

MD5 Message-Digest Algorithm

MBR Master boot record

NSRL National Software Reference Library

PE Portable Executable

RaaS Ransomware as a Service

RDP Remote Desktop Protocol

SHA Secure Hash Algorithm

SIEM Security Information and Event Management

SMB Server Message Block

SSH Secure Shell

URL Uniform Resource Locator

UTF16 Unicode Transformation Format

UUID Universally unique identifier

VNC Virtual Network Computing

WMI Windows Management Instrumentation

YAML YAML Ain't Markup Language

YARA Yet Another Ridiculous Acronym

Příloha B

Obsah příloh

- `DP-Lepic-Stanislav-2024.pdf` ..... text práce ve formátu PDF
- `ctiMe.txt` ..... stručný popis obsahu
- `iocs.txt` ..... hash hodnoty testovaných vzorků
- `src`
  - `rules` ..... vytvořená pravidla
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX