Stredná priemyselná škola elektrotechnická
Komenského 44, 040 01 Košice

Forenzná analýza obsahu operačnej pamäte

Marek Horňak

2024

Stredná priemyselná škola elektrotechnická
Komenského 44, 040 01 Košice

Forenzná analýza obsahu operačnej pamäte

Záverečná práca

Praktická časť odbornej zložky maturitnej skúšky
Forma: Obhajoba vlastného projektu

Riešiteľ: Marek Horňak
Ročník štúdia: štvrtý
Konzultant: Ing. Michal Copko
Košice 2024

Obsah

0. Úvod

Prvá myšlienka tohto projektu sa zrodila na jesennom Cybersecurity day, ktorý sa konal 22. októbra 2022 na Prírodovedeckej fakulte Univerzity Pavla Jozefa Šafárika na ulici Jesenná 5 v Košiciach. Súčasťou programu bola aj ponuka zaujímavých workshopov. Jedným z nich bol aj úvod do forenznej analýzy operačnej pamäte, ktorý viedla Mgr. Eva Marková, bezpečnostná analytička firmy ESET. Tam som sa zoznámil s digitálnou forenznou analýzou. Dovtedy mi bol tento pojem celkom cudzí. Vedel som, čo je forenzná analýza, no nijako som si to nespájal s počítačmi a kyberbezpečnosťou. Workshop bol veľmi zaujímavý a inšpiratívny, preto som sa o tému digitálnej forenznej analýzy začal zaujímať stále viac a viac.

Ako som sa pomaly dostával do sveta forenznej analýzy, všímal som si, že bolo dosť ťažké získať potrebné vedomosti a zručnosti jednak o samotnej forenznej analýze a jednak o používaní softvéru Volatility. Zdroje som čerpal aj z video návodov na YouTube, no musel som vidieť množstvo videí, aby som sa k niečomu dostal a aby mi to začalo dávať zmysel. Stretol som sa aj s firmami, ktoré ponúkali platené kurzy digitálnej forenznej analýzy. Preto som dostal nápad vytvoriť bezplatný učebný materiál o forenznej analýze a sprístupniť ho širokej verejnosti, aby sa poznatky o forenznej analýze operačnej pamäte dostali k čo najväčšiemu počtu ľudí a možno pomohli vychovať novú generáciu digitálnych forenzných analytikov. Ako svoj ročníkový projekt som chcel niečo výnimočné, niečo, čo predo mnou ešte nikto neprezentoval a som rád, že som sa dostal práve k forenznej analýze, pretože táto téma je zaujímavá a v dnešnej dobe aktuálna.

Svoje nápady a priebeh projektu som konzultoval nielen so svojimi konzultantmi, Ing. Michalom Copkom a Ing. Radovanom Repovským, ale aj so samotnou Mgr. Evou Markovou, ktorá mi poskytla odborné konzultácie.

Celý projekt je postavený na analýze obrazov operačnej pamäte počítačov s operačným systémom Windows 10 Home a Ubuntu 22.04 LTS pomocou open-source softvéru Volatility3. Projekt sa skladá z troch učebných materiálov o forenznej analýze. Jeden je čisto teoretický a má za úlohu oboznámiť študenta s digitálnou forenznou analýzou. Ďalší materiál obsahuje návod na inštaláciu a nastavenia Volatility3. Ostatné dva sú praktické a sú zhotovené formou návodu ako analyzovať jednotlivé obrazy operačnej pamäte. Jeden je pre Windows a druhý pre Linux. V oboch materiáloch je analýza prvých dvoch obrazov operačnej pamäte podrobne rozpísaná s obrázkami a použitými príkazmi. Na konci každého materiálu je referencia použitých príkazov a úloha na samostatnú prácu s tretím obrazom, pri analýze ktorého študent využije znalosti nadobudnuté analýzou prvých dvoch obrazov.

Chcel by som sa poďakovať svojmu odbornému konzultantovi pánovi Ing. Michalovi Copkovi, svojmu metodickému konzultantovi pánovi Ing. Radovanovi Repovskému a pani Mgr. Eve Markovej za ochotu pomôcť keď som si s niečím nevedel poradiť a za praktické rady a motiváciu.

1. Problematika a prehľad literatúry

1.1. Forenzná analýza vo všeobecnosti

Vo všeobecnosti môžeme forenznú analýzu definovať ako každú činnosť zameranú na hĺbkovú analýzu, vyšetrovanie, ktorého cieľom je objektívne určiť a zdokumentovať vinníkov, dôvody, priebeh a dôsledky bezpečnostného incidentu, priestupku alebo trestného činu. Cieľom forenznej analýzy je zozbierať čo najviac dôkazov o tom, kto, kedy a ako niečo zavinil. Často súvisí so súdnym dokazovaním, najmä v trestných záležitostiach. Zahŕňa využitie širokého spektra vyšetrovacích technológií, postupov a metód. Výsledkom forenznej analýzy je znalecký alebo technický posudok alebo vyjadrenie majúce dôkaznú hodnotu v súdnom konaní. Vychádza z odboru Forenzika (Forenzná veda).

1.2. Forenzná analýza v kyberbezpečnosti

Z hľadiska kyberbezpečnosti je forenzná analýza prostriedkom získavania dôkazov k bezpečnostným incidentom. Väčšina kyberbezpečnostných incidentov sa odohráva na serveroch, počítačoch a iných zariadeniach. Tie sú mnohokrát cieľom útoku. Dá sa povedať, že počítač alebo server je pri vyšetrovaní takýchto incidentov niečo ako miesto činu. Existuje mnoho podkategórií *digitálnej forenznej analýzy*, no my sa špecifikujeme len na jednu – forenzná analýza obsahu operačnej pamäte. Budeme sa teda sústrediť len na operačnú pamäť. Je to pamäť závislá od napätia (volatilná), teda je schopná uchovávať dáta len po dobu, kým je pod napätím. Uchováva dáta, s ktorými počítač aktuálne pracuje. Ak dôjde k nejakému kyberútoku alebo bezpečnostnému incidentu, všetko, čo sa na danom počítači alebo serveri odohrá, ostane v jeho operačnej pamäti. Z nej sa vyhotoví obraz, ktorý sa následne analyzuje nástrojom na forenznú analýzu pamäte a hľadáme potenciálne stopy, ktoré by nás mohli dostať k vinníkovi.

1.3. Prehľad existujúcich riešení

V dnešnej dobe sú asi najpopulárnejšou metódou učenia a tréningu digitálnej forenznej analýzy rôzne školenia a kurzy dostupné na internete alebo ponúkané rôznymi firmami. Avšak veľakrát sú platené a veľmi drahé. Alternatívou sú rôzne internetové videá dostupné hlavne na YouTube, no tie sú väčšinou nekompletné a človek si musí pospájať informácie z viacerých videí. Riešenie, ktorým som sa inšpiroval aj ja priniesol na Github používateľ menom **stuxnet999**

(https://github.com/stuxnet999/MemLabs). Jeho projekt niesol názov Memlabs. Obsahoval šesť voľne dostupných obrazov operačnej pamäte s operačným systémom Windows, na ktorých si ktokoľvek môže precvičiť analýzu operačnej pamäte. K týmto obrazom vytvoril na svojej Github stránke používateľ **n1ghtw0lf** (https://n1ght-w0lf.github.io/categories/#ctf-writeups) tzv. „writeupy“, čiže návody ako dané obrazy analyzovať. Ja som sa rozhodol spojiť práce týchto dvoch ľudí vytvorením vlastných obrazov operačnej pamäte na analýzu a zároveň k nim napísať aj návody ako ich analyzovať aj s potrebnou teóriou. Okrem toho som pridal aj materiály na analýzu obrazov operačnej pamäte s Linuxom

1.4. Použité technológie

1.4.1. Volatility3

V súčasnosti existuje niekoľko populárnych nástrojov na forenznú analýzu obsahu operačnej pamäte. Medzi najpoužívanejšie patria: Varc, Rekall (od Google security teamu) a Volatility.

Volatility je v súčasnosti najrozšírenejší a najpokročilejší nástroj pre forenznú analýzu obsahu operačnej pamäte. Používajú ho aj profesionálni forenzní analytici pri vyšetrovaní bezpečnostných incidentov. Je napísaný v programovacom jazyku Python. Poskytuje širokú škálu možností analýzy. Je dostupný ako open-source pre Windows, Linux aj Mac a taktiež umožňuje analýzu obsahu operačnej pamäte všetkých týchto operačných systémov. Volatility2 je dostupné aj ako standalone executable verzia aj ako zdrojový kód v .zip archíve. Volatility3 je dostupné už len ako zdrojový kód.

Obrázok 1: Volatility logo

Výhody Volatility3:

- rýchlosť
- automatická identifikácia verzie OS

Nevýhody Volatility3:

- nedostatok pluginov
- nedostatočná funkčnosť niektorých pluginov

1.4.2. Rozdiel medzi Volatility2 a Volatility3

Volatility3 je novšia verzia softvéru Volatility2 na analýzu obrazov operačnej pamäte. Volatility3 je napísaný v Python3 a podporuje Python vo verzii 3.7 a vyššie. Je teda aktuálny. Posledné vydanie Volatility2 bolo v decembri 2016, zatiaľčo najnovšia verzia Volatility3 bola vydaná v apríli 2023. Volatility3 poskytuje oproti Volatility2 novšie a rýchlejšie pluginy ako pre Windows, tak aj pre Linux. Hlavnou výhodou Volatility3 oproti Volatility2 je rýchlosť. Volatility3 je výrazne rýchlejšie ako Volatility2. Je to výhodné, pretože obrazy operačnej pamäte majú väčšinou veľkosť 2 GB a viac, takže zrýchlenie celého procesu je veľmi užitočné. Oproti Volatility2 prináša Volatility3 okrem zvýšenej rýchlosti tú výhodu, že nie je potrebné špecifikovať profil operačného systému, pretože Volatility3 pri prvom spustenom príkaze automaticky identifikuje profil operačného systému, uloží ho do svojej vyrovnávacej pamäte a používa ho pri ďalšej analýze. Čo sa týka analýzy operačnej pamäte s Linuxom, nie je Volatility3 až také efektívne z hľadiska absencie niektorých pluginov, ktorými disponovalo Volatility2. Z tohto dôvodu som sa rozhodol trochu pozmeniť formát úloh pre obrazy operačnej pamäte s Linuxom, aby to vyhovovalo dostupnosti pluginov a aby vzdelávací potenciál učebného materiálu ostal na úrovni materiálu pre analýzu Windowsu. Čo sa týka verzií Pythonu, Volatility2 podporuje Python2.6 a novšie, no nie Python3. To prináša určitú nevýhodu, pretože Python2 od 1. januára 2020 už nie je upravovaný a nie sú v ňom opravované ani bezpečnostné chyby a zraniteľnosti. Volatility3 vyžaduje Python 3.7 a vyššie.

Výhody Volatility2:

- Širšia škála pluginov
- Výborná funkčnosť pluginov

Nevýhody Volatility2:

- Zastaralé a stráca podporu
- Výrazne pomalšie ako Volatility3

1.4.3. AVML

AVML (Acquire Volatile Memory for Linux) je softvér od spoločnosti Microsoft vytvorený pre zachytávanie operačnej pamäte počítačov s Linuxovým operačným systémom. Je napísaný v

programovacom jazyku Rust a je dostupný ako binárny spustiteľný súbor. Taktiež ponúka možnosť kompresie obrazu operačnej pamäte pomocou knižnice Snappy. Všetko, čo treba urobiť je stiahnuť ho a upraviť práva na spúšťanie príkazom:

sudo chmod 766 avml

Jeho následné používanie je veľmi jednoduché. Stačí použiť príkaz:

sudo ./avml /path/to/memory.raw

Výhody:

- jednoduché používanie
- nezávislosť na distribúcii alebo na verzii kernelu
- možnosť kompresie obrazov

Obrázok 2: Microsoft logo

1.4.4. FTK Imager

FTK Imager je softvér firmy Exterro, ktorý som použil na zachytenie obrazu operačnej pamäte počítača s Windowsom. Vybral som si ho pre jeho jednoduché použitie a rýchle spracovanie zachyteného obrazu. Okrem tejto funkcie má ešte mnoho ďalších. Dokáže napríklad prezerať súborový systém pevných diskov počítačov vytvorením read-only obrazu celého disku, takže nedôjde k zmenám na samotnom disku a dokáže zobraziť a obnoviť zmazané súbory pokiaľ neboli prepísané na pevnom disku. Taktiež s ním môžeme vytvárať hashe súborov z disku. Používa sa v mnohých odvetviach digitálnej forenznej analýzy.

Obrázok 3: FTK Imager logo

1.4.5. Virtualbox

Virtualbox je virtualizačný softvér od spoločnosti Oracle, ktorý umožňuje vytváranie virtuálnych počítačov na fyzickom hostiteľskom počítači. Ponúka množstvo užitočných funkcií ako napríklad možnosť nastavenia parametrov virtuálneho počítača (veľkosť operačnej pamäte, počet jadier procesora, veľkosť úložiska, nastavenie sieťových adaptérov atď.). Veľmi užitočnou funkciou Virtualboxu je možnosť exportovať a importovať virtuálny počítač. Túto funkciu som často využíval počas vytvárania obrazov operačnej pamäte (na každý obraz som používal osobitný virtuálny počítač). Na začiatku som si vytvoril jeden virtuálny počítač, nainštaloval som doňho potrebný softvér a exportoval som si ho do súboru .ova. Následne som si podľa potreby mohol importovať daný počítač a keď sa náhodou niečo nevydarilo, počítač som zmazal a naimportoval nový bez potreby inštalácie.

Obrázok 4: Virtualbox logo

1.4.6. Python3

Python je interpretovaný, interaktívny programovací jazyk, ktorý vytvoril Guido van Rossum, pôvodne ako skriptovací jazyk pre Amoeba OS schopný systémových volaní. Python je vyvíjaný ako open source projekt, a je v súčasnosti pri verzii 3.12. Podporuje objektovo orientované, štruktúrované aj funkcionálne programovanie. Je to dynamicky typový jazyk, ktorý podporuje veľké množstvo vysokoúrovňových dátových typov. Aj keď sa Python často označuje ako „skriptovací jazyk“, používa sa na vývoj mnohých veľkých softvérových projektov ako sú aplikačný server Zope a systémy na zdieľanie súborov Mnet a BitTorrent. Tak isto ho široko využíva Google. Zástanci Pythonu ho radšej volajú vysokoúrovňovým dynamickým programovacím jazykom, lebo pojem „skriptovací jazyk“ sa asociuje s jazykmi, ktoré sa používajú len na jednoduché shell skripty alebo s jazykmi ako JavaScript: jednoduchšími a na väčšinu účelov menej spôsobilými ako „skutočné“ programovacie jazyky ako Python. Ďalšou dôležitou vlastnosťou Pythonu je to, že sa dá jednoducho rozširovať. Nové zabudované moduly môžu byť jednoducho napísané v C alebo C++. Python tiež môže byť použitý ako rozširovací jazyk pre existujúce moduly a aplikácie, ktoré potrebujú programovateľné rozhranie. Názov jazyka vôbec nevznikol z názvu druhu hada. Autor nazval jazyk podľa populárneho britského satirického seriálu Monty Python’s Flying Circus. Ale napriek tomu sa názov jazyka často asociuje práve s hadom a nie so seriálom.

Obrázok 5: Python logo

1.5. Finančná analýza

Čo sa týka financií, môj projekt nestál čo sa týka nákladov vôbec nič. Používal som len open-source a voľne dostupné softvéry a svoju prácu som bez akýchkoľvek poplatkov zverejnil na stránke Github. Volatility3 je voľne dostupné na stiahnutie na Github stránke alebo na oficiálnej stránke Volatility foundation . Softvéry na zachytávanie obrazu operačnej pamäte pre oba operačné systémy boli taktiež voľne dostupný. AVML som obstaral z Github stránky firmy Microsoft a FTK Imager pre Windows je po vyplnení krátkeho dotazníka zadarmo dostupný na stránke firmy Exterro. Virtualbox, v ktorom som vytváral všetky obrazy operačnej pamäte je taktiež voľne dostupný. ISO súbory Ubuntu a Windows 10 Enterprise som obdržal od školy, čo mi v prípade Windowsu ušetrilo náklady približne 60,00€. Virtualbox je taktiež voľne dostupný softvér. Jediným nákladom na tento projekt bola moja vlastná práca pri vytváraní obrazov, dokumentov a ostatných súborov.

2. Výsledky práce

2.1. Vytvorenie schémy úloh

Pri vytváraní schémy úloh pre jednotlivé obrazy som sa inšpiroval hrou „Capture the flag“ a taktiež už spomínaným projektom Memlabs od používateľa stuxnet999. Každý obraz operačnej pamäte obsahuje vlajky (ang. flag), buď v obrázkovej alebo textovej podobe, ktoré je treba v obraze operačnej pamäte nájsť. Obrazy operačnej pamäte s Windowsom obsahujú 3 flagy a sú o niečo zložitejšie ako obrazy s Ubuntu, ktoré sú realizované formou pracovných listov a pre každý obraz je vytvorených päť jednoduchých úloh. V úlohe je zadané, aký údaj treba v pamäti nájsť. Tu už študent nehľadá flagy ako také, ale konkrétne dáta. Je to síce odlišné od učebného materiálu pre Windows, no stále to funguje na tom istom princípe. Analýza Windowsu je komplexnejšia a náročnejšia ako pri

Linuxe. Dôvodom je to, že som chcel dať väčší dôraz na Windows, pretože veľká väčšina kyberútokov spôsobených napríklad počítačovými vírusmi alebo iným malvérom sa odohráva práve na Windowse. Tým pádom je častejším predmetom forenzných analýz Windows. Okrem toho pri Linuxe som sa musel prispôsobiť škále pluginov, ktoré Volatility3 ponúkalo. Z môjho pohľadu táto ponuka pluginov nebola dostačujúca na vytvorenie kvalitnejších úloh pre analýzu obrazov operačnej pamäte s Linuxom a veľakrát sa stalo, že som musel pôvodnú schému úloh pre Linux prerobiť, pretože sa nedala dostupnými pluginmi pre Linux analyzovať.

2.2. Vytvorenie potrebných súborov

Keď som mal schému úloh hotovú, zameral som sa na vytvorenie potrebných súborov. Do tejto kategórie spadá vytvorenie obrázkových flagov, ktoré som vytváral v programe Inkscape, s ktorým sme sa učili pracovať v druhom ročníku na predmete Grafika a multimédiá. Ďalej som vytváral WinRAR archívy chránené heslom, ktoré obsahovali dané flagy. Archívy obsahovali aj komentár, kde bola pomôcka na zistenie hesla. Niektoré archívy a textové súbory som nahrával aj na cloud. Používal som na to cloudové služby Mega, Google drive, Github a Pastebin.

2.3. Vytvorenie obrazov operačnej pamäte

Táto časť bola časovo neľmi náročná. Všetky obrazy operačnej pamäte som vytváral tak, že som si vytvoril virtuálny počítač a v závislosti od schémy úloh pre daný obraz som na počítači spravil potrebné úkony. V tomto štádiu mojej práce som sa dostával stále hlbšie do celého procesu fungovania operačnej pamäte. Potreboval som potrebné súbory dostať do operačnej pamäte, aby následne boli v jej obraze dostupné. Napríklad čo sa týka súborov ako sú obrázky a archívy, tie stačilo iba otvoriť, aby sa ich obsah nakopíroval do operačnej pamäte. Aby som docielil zapísanie linkov na webové stránky, otvoril som prehliadač a navštívil danú stránku. Jednoducho povedané, operačná pamäť zachytí všetko, čo používateľ na danom počítači robil od jeho zapnutia až po vytvorenie samotného obrazu operačnej pamäte. Zaujímavým bol pre mňa Windows skicár. Ak si len otvoríme skicár a začneme niečo kresliť bez toho, aby sme daný súbor uložili, jeho obsah sa dynamicky s každou zmenou ukladá do operačnej pamäte. Preto, ak zachytíme operačnú pamäť v čase, kedy bol skicár otvorený a niečo v ňom bolo, vieme si jeho obsah stiahnuť do súboru s nespracovanými údajmi a premenovaním prípony na .data si tento súbor vieme otvoriť v Gimpe. Tu už potrebujeme len trochu upraviť šírku a offset a uvidíme, čo daný používateľ nakreslil. Čo sa týka Linuxu, zaujímavým bol pre mňa plugin *linux.bash*. Ten umožňuje vypísanie histórie bashu. No nielen príkazy zadané počas behu počítača, dá sa ísť ešte hlbšie. Linux si uchováva históriu príkazov v súbore a pri každom otvorení terminálu sa obsah tohto súboru nahrá do operačnej pamäte. Takto je možné dostať sa aj k starším príkazom. Okrem šiestich obrazov na analýzu som vytvoril jeden čistý obraz operačnej

pamäte, ktorý patrí k učebnému materiálu o inštalácii a nastaveniach Volatilit3. Nakoniec som všetky obrazy pamäte nahral na cloud. Keďže Github neuchováva tak veľké súbory (veľkosť obrazov bola 2 – 4GB), využil som službu Mega.

2.4. Používanie Volatility3

Používanie Volatility3 je jednoduché a ľahko sa dá naučiť, preto si myslím, že použitie Volatility3 ako softvéru na analýzu do učebných materiálov pre širokú verejnosť bola dobrá voľba. Výhodou Pythonu, v ktorom je Volatility3 napísané je, že je rýchly a dobre pracuje s dátami. Koniec-koncov obrazy operačnej pamäte sú iba veľká kopa dát, takže použitie Pythonu je veľmi efektívnou metódou analýzy. Volatility3 je obyčajný Python skript a spúšťa sa tak isto ako iné Python skripty.

python3 vol.py -h.

Tento príkaz vypíše všetky informácie o dostupných profiloch operačných systémov a dostupných pluginoch. Ak chceme analyzovať nejaký obraz operačnej pamäte, musíme to špecifikovať prepínačom -f.

python3 vol.py -f memdump.raw

Ďalej musíme špecifikovať aký plugin chceme používať. Systém názvov jednotlivých pluginov je takýto: *operačnýsystém.plugin.Plugin*, ale môže sa používať aj: *operačnýsystém.plugin*.

python3 vol.py -f memdump.raw windows.pslist.Pslist

Ak chceme výstup filtrovať na základe nejakých parametrov, v závislosti od pluginu Volatility3 ponúka prepínače ako napríklad: --pid na filtrovanie podľa procesu. Taktiež je možné použiť vlastné doplnkové skripty alebo filtrovať výstup pomocou programu grep.

2.5. Praktická analýza

Môžeme začať napríklad tak, že si vypíšeme zoznam bežiacich procesov z pamäte pomocou pluginu *windows.pslist*.

Výstup z príkazu je v Prílohách C v prílohe 1

Uvidíme, že bol spustený WinRAR.exe. To značí, že používateľ otváral nejaké archívy. Volatility ponúka plugin *windows.cmdline*, ktorý dokáže vypísať, ktorý vypíše spustený proces a aj súbor, ktorý otvoril. Použijeme ho a výstup si pomocou programu grep vyfiltrujeme len na WinRAR.

Výstup z príkazu je v Prílohách C v prílohe 2

Vidíme, že WinRAR otvoril súbor Important.rar. Teraz potrebujeme zistiť jeho adresu v pamäti, aby sme ho mohli extrahovať. Použujeme na to plugin *windows.filescan* a pomocou programu grep si vyfiltujeme výstup len pre súbor Important.rar.

Výstup z príkazu je v Prílohách C v prílohe 3

Teraz, keď vieme jeho pamäťový offset, môžeme si archív extrahovať z obrazu operačnej pamäte. Použjeme plugin *windows.dumpfiles*. Prepínačom -o špecifikujeme, kam chceme súbor extrahovať a prepínačom --virtaddr špecifikujeme pamäťový offset súboru.

Výstup z príkazu je v Prílohách C v prílohe 4

Súbor máme extrahovaný v našom počítači. Volatility mu priradilo trochu zvláštny názov, ktorý obsahuje aj jeho pamäťový offset, no stačí ho premenovať späť na Important.rar a otvoriť. Po otvorení zistíme, že súbor je chránený heslom. Komentár k archívu hovorí, že heslo je ukryté medzi premennými prostredia (environment variables).

Okno WinRARu je v Prílohách C v prílohe 5

Vrátime sa teda späť do analytického počítača a spustíme plugin *windows.envars*, ktorý vypíše všetky premenné prostredia. Keďže je ich pomerne veľa, programom grep si vyfiltujeme výstup len na premenné programu WinRAR. Pozor na veľké a malé písmená.

Výstup z príkazu je v Prílohách C v prílohe 6

Použijeme získané heslo na odomknutie archívu a archív rozbalíme. Dostaneme obrázok s flagom.

Obrázok s flagom je v Prílohách C v prílohe 7

2.6. Použité príkazy Volatility3

Tu je zornam všetkých Volatility3 pluginov, ktoré som použil pri analýze obrazov operačnej pamäte s Windowsom a Linuxom spolu s krátkym popisom ich funkcie.

2.6.1. Windows

windows.pslist	vypíše zoznam bežiacich procesov a ich PID a PPID
windows.cmdline	vypíše postupne aktiváciu procesov a ich manipuláciu so súbormi
windows.envars	vypíše zoznam premenných prostredia
windows.hashdump	vypíše zoznam používateľov a hash hodnoty ich hesiel
windows.filescan	vypíše zoznam všetkých súborov v systéme
windows.vadyarascan	vyhľadá zadaný reťazec v yara-rules v pamäti daného procesu

windows.memmap	extrahuje celý pamäťový úsek patriaci danému procesu do súboru
windows.dumpfiles	extrahuje celý pamäťový úsek patriaci virtuálnej adresy v pamäti do súboru

2.6.2. Linux

linux.pslist	vypíše zoznam bežiacich procesov a ich PID
linux.bash	vypíše históriu bash commandov
linux.lsof	vypíše zoznam otvorených súborov aj procesov, ktoré ich používali
linux.tty_check	vypíše zoznam spustených terminálov
linux.sockstat	vypíše zoznam aktívnych sieťových spojení
linux.elfs	vypíše zoznam elf súborov a procesov, ktoré ich používali
linux.mountinfo	vypíše zoznam mountovacích bodov, ich mount ID a Major:Minor index

2.6.3. Skript

Tento skript som používal súbežne s pluginom *windows.vadyarascan.VadYaraScan*. Úlohou tohto doplnkového kódu je upraviť výpis z tohto pluginu, aby sme videli potrebné informácie, ktoré plugin sám o sebe nevypisuje. Samostatný plugin vypíše iba miesto, kde našiel konkrétny zadaný reťazec a jeho hexadecimálnu hodnotu. Skript robí to, že tieto hexadecimálne hodnoty mení na znaky a vypíše nie len konkrétny zadaný reťazec, ale aj niekoľko znakov za ním. Ich počet si vieme zvoliť.

1. *python3 vol.py -f /media/sf_VMshare/lab1_win.raw windows.vadyarascan.VadYaraScan --pid 3876 --yara-rules "/http.{100}?/" | awk '*
2. * /^0x/{*
3. * S=""*
4. * for(i=5; i<NF; i++) {*
5. * S=S sprintf("%s", $i)*
6. * }*
7. * print "PID:" $2 "\tRule:" $3 "\tComponent:" $4*
8. * system("echo " S " | xxd -r -ps | xxd -o " $1 " | sed \"s/^/0x/\"")*
9. * print ""*
10. * next*

11. *}*
12. */^Offset/{next}*
13. *{print}*
14. *'*

Funkcia jednotlivých riadkov:

1. Spustí program Volatility3 s pluginom vadyarascan. Výstup obmedzí len na proces číslo 3876 (chrome) a vyhľadá v jeho pamäti string špecifikovaný v yara-rules. Výstup pošle programu awk, čo je program na spracovanie bash skriptov, v apostrofoch je text, ktorý program awk musí spracovať.
2. Definícia podmienky, aby program spracoval len riadky, ktoré sa začínajú s „0x“
3. Inicializácia prázdneho reťazca S
4. Každý spracovaný riadok textu, ktorý awk preberá od pluginu vadyarascan je rozdelený na jednotlivé polia, kde sú offset a hexadecimálne hodnoty reťazcov z pamäte. Tento for cyklus má z úlohu všetky tieto polia zapísať do premennej S
5. Zápis vybraných polí do premennej S
6. Koniec for cyklu
7. Výpis premenných PID, Rule a Component
8. Reťazec S prekonvertuje z hexdumpu do binárneho kódu, pridá na začiatok reťazca „0x“ a na koniec reťazca pridá pole $1, čo je vlastne text, ktorý program vadyarascan hľadal v pamäti
9. Prázdny print, vynechanie jedného riadku
10. Presun, spracovanie ďalšieho riadku
11. Koniec podmienky
12. Definícia podmienky na preskočenie riadkov, ktoré sa začínajú s „Offset“
13. Vypíše všetky riadky, ktoré nezačínajú ani s „0x“ ani s „Offset“
14. Koniec reťazca, ktorý je spracovaný programom awk

Výstup skriptu je v Prílohách C v prílohe 8

2.7. Analýza vytvorených obrazov

Počas analýzy vytvorených obrazov operačnej pamäte som musel riešiť najviac problémov. Dalo by sa povedať, že okrem tohto bodu tvorenia môjho projektu išlo všetko bezchybne. S analýzou obrazov operačnej pamäte s Windows 10 žiadny výrazný problém nebol, keďže Volatility má prednastavené viaceré Windowsové profily, na základe ktorých dokáže dané obrazy operačnej pamäte prečítať. No obrazy s Ubuntu ma poriadne potrápili. Bolo to preto, lebo Linuxových distribúcií je príliš veľa na to, aby Volatility mohlo mať všetky profily uložené. Preto vznikli Linuxové profily pre Volatility, ktoré sú dostupné na internete, alebo sa dajú manuálne vygenerovať. Vytvorenie obrazov pre Ubuntu nebolo o nič ťažšie ako pre Windows, no s Volatility3 boli problémy. Zo začiatku som plánoval použiť na analýzu obrazov operačnej pamäte s Ubuntu Volatility3 rovnako ako pri Windowse. Volatility3 používa na určenie verzie Linuxu tzv. symbolové tabuľky (ang. symbol tables), teda .json súbor komprimovaný do súboru .json.xz. Tento súbor je treba vytvoriť. Skúšal som rôzne spôsoby vytvorenia symbolovej tabuľky, no ani jeden nefungoval. Po niekoľkých hodinách troubleshootingu som našiel príčinu. Nefungovalo to preto, lebo bolo ešte potrebné vytvoriť konfiguračný súbor pre Volatility3 a zapísať doňho cestu k spomínanému súboru. Po tomto zistení som úspešne vytvoril potrebnú symbolovú tabuľku. Vtedy som už bez problémov mohol začať analýzu obrazov s Ubuntu. No po chvíli som zistil, že ponuka pluginov vo Volatility3 nie je dosť široká a neobsahuje potrebné pluginy pre moje úlohy. Rozhodol som sa preto pre Volatility2. Vyskúšal som aj verziu standalone aj zdrojový kód, no ani jedno nefungovalo. Volatility2 používa na určenie verzie Linuxu nie symbolové tabuľky, ale tzv. linuxový profil, .zip súbor, ktorý obsahuje kernelové informácie. Vytvára sa pomocou programu dwarfdump. Profil som vytvoril, no Volatility nebol schopný ho použiť. Napriek svojej zastaranosti je Volatility2 stále široko používané na forenznú analýzu, no po všetkých tých problémoch som sa rozhodol, že pre univerzálnosť môjho učebného materiálu bude lepšie ak Volatility2 nepoužijem. Keď som pochopil, že s Volatility2 sa ďalej nepohnem, rozhodol som sa vrátiť k pôvodnému plánu s Volatility3. Avšak pôvodné obrazy určené pre Volatility2 sa kvôli nedostatku a nedostatočnej funkčnosti pluginov nedali analyzovať. To znamenalo, že som musel kompletne prerobiť úlohy k obrazom a nanovo vytvoriť všetky tri obrazy operačnej pamäte. Práca však išla veľmi dobre a tieto úlohy som zvládol v priebehu niekoľkých hodín aj s tým, že som pre ne zároveň vytvoril aj príslušné učebné materiály.

2.8. Rozdiel medzi analýzou Windowsu a Linuxu

Kým analýza pamäte s Windowsom sa pri prechode z verzie 2 na verziu 3 takmer nezmenila, pri Linuxe sú podstatné rozdiely. Verzia 2 používala na zistenie verzie operačného systému tzv. linuxový profil, čiže .zip archív, ktorý obsahuje údaje na identifikáciu operačného systému. Bez toho

Volatility2 nie je schopné analyzovať obraz operačnej pamäte s Linuxom. Volatility3 už nepoužíva Linuxové profily, ale tzv. symbolové tabuľky. V niektorých ohľadoch sú podobné s profilmi. Taktiež obsahujú informácie potrebné na rozpoznanie verzie operačného systému, no už to nie je klasický .zip súbor, ale komprimovaný .json súbor s príponou .json.xz. Čo sa týka Windowsu, vo Volatility2 bolo potrebné ešte pred samotnou analýzou oskenovať obraz operačnej pamäte pomocou pluginu *imageinfo* a vyhodnotiť verziu operačného systému. Tento profil bolo nutné následne špecifikovať zakaždým, keď sme používali nejaký plugin. Volatility3 to celé o niečo zjednodušilo, pretože netreba vôbec riešiť verziu operačného systému. Pri behu prvého pluginu sa automaticky oskenuje celý obraz operačnej pamäte (preto aj vykonanie prvého pluginu trvá dlhšie) a profil operačného systému uloží do svojej vyrovnávacej pamäte.

2.9. Vytvorenie učebných materiálov

Počas kontrolnej analýzy všetkých obrazov operačnej pamäte som si robil postupne screenshoty každého príkazu a ukladal som si ich do priečinkov ku každému obrazu osobitne. Následne som pri vytváraní učebných materiálov tieto screenshoty použil. Dbal som na to, aby textu nebolo priveľa a zároveň aby bolo všetko vysvetlené. Bolo pre mňa dôležité, aby materiály boli písané zrozumiteľne a zaujímavo a aby vzbudili v študentoch záujem o hlbšie štúdium forenznej analýzy či už ako samoštúdium alebo na vysokej škole. Ku každému screenshotu som napísal aj vzor príkazu, ktorý treba použiť. Na konci materiálu som zhrnul všetky príkazy použité pri analýze obrazov spolu s ich syntaxom, prepínačmi a aj krátkym popisom funkcie. Na konci práce som všetky učebné materiály zverejnil na mojej Github stránke aj s odkazmi na obrazy operačnej pamäte uložené na Mega.

3. Závery a zhrnutie

Realizáciou môjho projektu som priniesol prehľadný učebný materiál, ktorý je vďaka platformám ako je Github a Mega dostupný pre širokú verejnosť zadarmo pre študentov stredných škôl ako aj pre obyčajných ľudí so záujmom o kyberbezpečnosť a vyšetrovanie kyberzločinu. Učebný materiál obsahuje teoretickú časť, ktorá zásobí študenta potrebnými vedomosťami o forenznej analýze a o používaní softvéru Volatility, ktorý je v súčasnosti najrozšírenejší, najpoužívanejší a zároveň voľne dostupný open-source softvér na analýzu obsahu operačnej pamäte. Ďalej vďaka prehľadným návodom sa študent hravou a zaujímavou formou naučí základy forenznej analýzy obsahu operačnej pamäte, ktoré budú tvoriť základ jeho ďalšieho poznávania tejto oblasti.

Môj projekt má veľký potenciál na zlepšovanie, pretože zatiaľ pokrýva len jednu podmnožinu celkovej digitálnej forenznej analýzy. Pojem digitálna forenzná analýza je veľmi široký a môj projekt

by sa dal určite ešte niekoľkonásobne rozšíriť. Ďalšími prvkami, o ktoré je možné môj vzdelávací projekt rozšíriť sú napríklad Forenzná analýza mobilných zariadení, Forenzná analýza počítačovej siete, Forenzná dátová analýza, Forenzná databázová analýza alebo Forenzná analýza IoT zariadení. Taktiež by bolo možné vytvoriť blog alebo komunitné fórum, kam by nadšenci forenznej analýzy mohli prispievať svojimi tipmi a trikmi na zlepšenie efektivity forenznej analýzy.

Počas realizácie tohto projektu som sa toho veľa naučil. Nepochybne som sa zlepšil v práci s Linuxovým operačným systémom a začalo ma to viac baviť. Taktiež som sa naučil sa nové veci tým, že som sa musel od základu naučiť pracovať s Volatility3. Okrem týchto technických zručností v odbore som si určite zlepšil trpezlivosť pri riešení problémov ako aj prácu s vyhľadávaním informácií na internete. Forenzná analýza ma naučila všímať si detaily ukryté ako „ihla v kope sena“ a pracovať s nimi, aby som sa dostal do stanoveného cieľa. A v neposlednom rade ma tento projekt naučil disciplíne pri riešení projektových úloh, čo mi veľmi pomôže počas štúdia na vysokej škole a aj v živote.

4. Resumé

My project, educational material for memory forencics, is meant for everyone who is interested in forensic analysis of volatile memory and wants to gain experience and knowledge in that field.

My project consists of three memory images for Windows and three memory images for Ubuntu. Then it has one teorethical material containing necessary knowledge about forensic analysis and use of Volatility. There is also one material as a guide to Volatility3 setup and two practical documents with tutorials on how to analyse memory images.

For my project I used only free and open-source software in order to make my project available for wide public. For memory capture I used Microsoft’s AVML (Acquire Volatile Memory for Linux) and for Windows I used FTK Imager, both available for free. For memory analysis I used Volatility3, an open-source software for memory analysis often used by professionals.

Each Windows memory image contains three flags. Linux material is created as a worksheet and for each Linux memory image are given five simple tasks. Common feature for both materials is, that first two images are for guided analysis described in the educational material. The last image is for students to analyze it on their own to test their skills gained by analysing previous two.

I had some problems with this project of course. The main problem was analysis of memory images with Ubuntu. Volatility uses zip files containig Linux profiles to determine the kernel version of analysed image. I had problems with creating and using those profiles, because I had to create them

manually, but after some troubleshooting I managed to solve this problem by changing versions from Volatility2 to Volatility3.

I published my project on my Github page to be available for everyone with internt access.

5. Použitá literatúra

5.1. Popis použitých technológií

[1] Popis Volatility2 <https://github.com/volatilityfoundation/volatility/>

[2] Popis Volatility3 <https://github.com/volatilityfoundation/volatility3>

[3] Popis AVML <https://github.com/microsoft/avml>

[4] Popis FTK Imager <https://www.exterro.com/ftk-imager>

[5] Popis Virtualboxu <https://en.wikipedia.org/wiki/VirtualBox>

[6] Popis Python3 <https://en.wikipedia.org/wiki/Python_(programming_language)>

5.2. Videá o analýze

[7] Volatility3 – používanie <https://www.youtube.com/watch?v=Uk3DEgY5Ue8>

[8] Volatility3 – Windows <https://www.youtube.com/watch?v=EqGoGwVCVwM>

[9] Volatility3 – Linux <https://www.youtube.com/watch?v=6Frec5cGzOg>

6. Prílohy

6.1. Prílohy A - Učebné materiály

[Teoretický materiál]

https://github.com/marekhornak/Forenzna-analyza-operacnej-pamate/blob/main/Forenzn%C3%A1%20anal%C3%BDza%20-%20te%C3%B3ria.pdf

[Volatility3 setup]

https://github.com/marekhornak/Forenzna-analyza-operacnej-pamate/blob/main/Volatility%20-%20setup.pdf

[Praktický materiál Windows]

https://github.com/marekhornak/Forenzna-analyza-operacnej-pamate/blob/main/Forenzna_analyza_Windows/Forenzna_analyza_operacnej_pamate_windows.pdf

[Praktický materiál Linux]

https://github.com/marekhornak/Forenzna-analyza-operacnej-pamate/blob/main/Forenzna_analyza_Linux/Forenzna_analyza_operacnej_pamate_linux.pdf

6.2. Prílohy B - Obrazy pamäte

[Úvodný obraz]

https://mega.nz/file/dbEwlAST#lVMXtspZokdGJPJ_OFDqLZFkLlMtD64w7DQ8b8rTFPc

[Lab 1 – Windows]

https://mega.nz/file/ULdWhCqY#0aYgOrE1uNiRZN94Q5FW5pmzOKkvsnsw0CCxVSq2TiY

[Lab 2 – Windows]

https://mega.nz/file/Jbs2yRYQ#eOcYWsSEMd0p2M7ExdKVOUTtKWCMayrGbAO38QK12I4

[Lab 3 – Windows]

https://mega.nz/file/5Ks0naaT#pVjJ4vawFkei2MCLXjqtO-fQ9TNiQZwtMph7Bn_wfpo

[Lab 1 – Linux]

https://mega.nz/file/cO0CRChY#gM98H0E11FpcaRDfTBeoL2SbyZip5wo6dnSqSsF7u84

[Lab 2 – Linux]

https://mega.nz/file/4WFE0LZK#rQPpKR3RFLaOX9SM3A4WFrXku-JnRgTDZriV6oQCaRY

[Lab 3 – Linux]

https://mega.nz/file/hHE3QKRB#RALQaaQAgMFc7hgWhWzfe8C88YpC4ONhy6bO0k8kmO4

6.3. Prílohy C - Výpisy pluginov

1996	496	NisSrv.exe	0xe000cfdf6780	5	-	0
720	884	dasHost.exe	0xe000cff07080	3	-	0
2340	796	sihost.exe	0xe000cfed0300	10	-	1
2380	796	taskhostw.exe	0xe000d006e780	10	-	1
2488	472	userinit.exe	0xe000d009f780	0	-	1
2504	2488	explorer.exe	0xe000d00cd780	51	-	1
2608	592	RuntimeBroker.	0xe000d0115440	11	-	1
2676	496	SearchIndexer.	0xe000ce11d440	17	-	0
3032	592	ShellExperienc	0xe000cffd5780	51	-	1
2288	592	SearchUI.exe	0xe000cffe4780	36	-	1
3652	2504	VBoxTray.exe	0xe000ce12c780	11	-	1
3716	2504	OneDrive.exe	0xe000d0061780	17	-	1
3516	2504	WinRAR.exe	0xe000d0040680	2	-	1
4060	2504	chrome.exe	0xe000cfa46200	39	-	1
3432	4060	chrome.exe	0xe000cd244780	8	-	1
2216	4060	chrome.exe	0xe000cd25a780	13	-	1
2224	4060	chrome.exe	0xe000cd30a780	15	-	1
2220	4060	chrome.exe	0xe000cd30c780	9	-	1
2348	4060	chrome.exe	0xe000ccf13080	8	-	1
2400	4060	chrome.exe	0xe000cceff080	18	-	1

Príloha 1: Bežiace procesy

```
marek@marek-ubuntu:~/volatility3$ python3 vol.py -f /media/sf_VMshare/lab1_win.raw windows.cmdline | grep WinRAR
3516ressWinRAR.exe      "C:\Program Files\WinRAR\WinRAR.exe" -iext "C:\Users\Marek\Documents\Important.rar"
marek@marek-ubuntu:~/volatility3$
```

Príloha 2: WinRAR archív

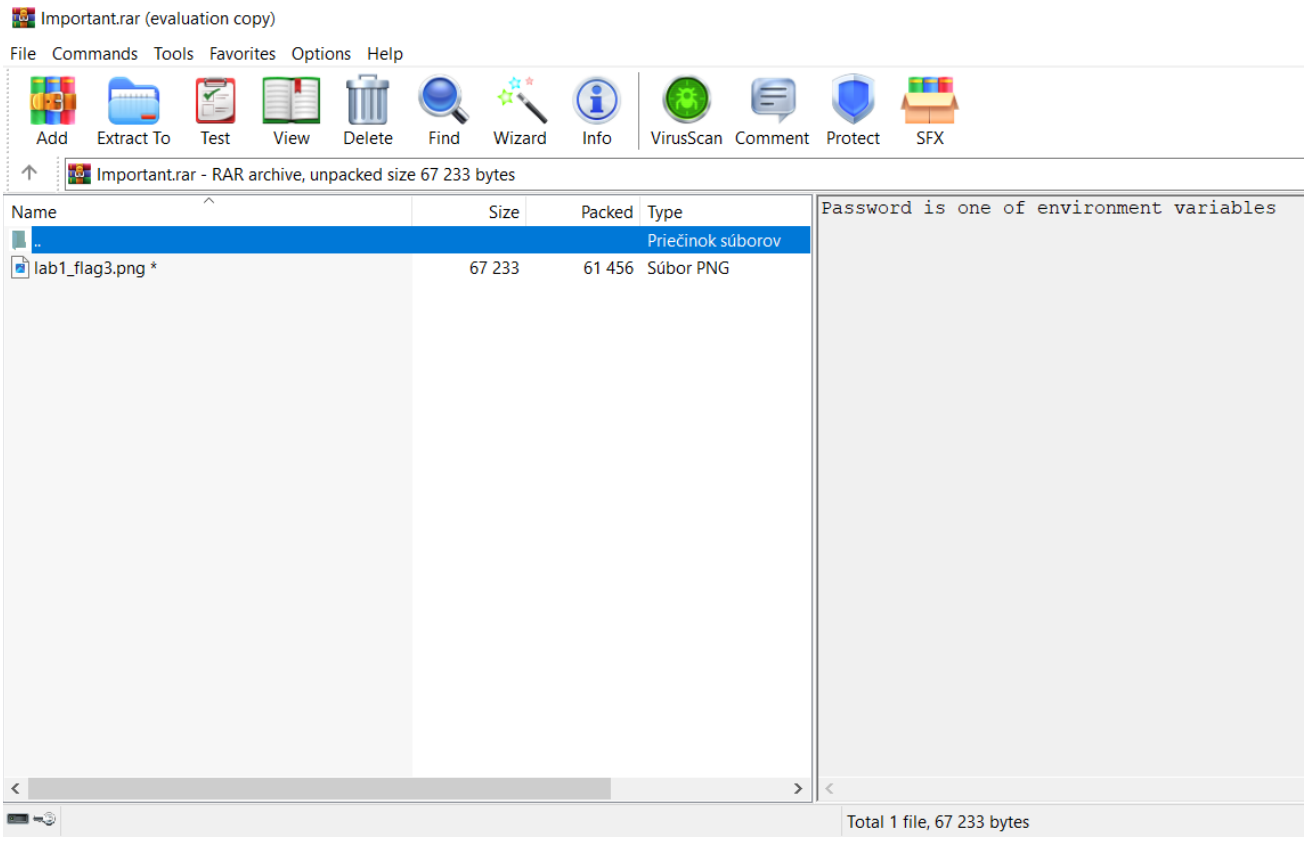
```
marek@marek-ubuntu:~/volatility3$ python3 vol.py -f /media/sf_VMshare/lab1_win.raw windows.filescan | grep Important.rar
0xe000cdd5a800.0\Users\Marek\Documents\Important.rar    216
marek@marek-ubuntu:~/volatility3$
```

Príloha 3: Adresa v pamäti

```
marek@marek-ubuntu:~/volatility3$ python3 vol.py -f /media/sf_VMshare/lab1_win.raw -o /media/sf_VMshare/ windows.dumpfiles --virtaddr 0xe000cdd5a800
Volatility 3 Framework 2.5.2
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xe000cdd5a800  Important.rar   file.0xe000cdd5a800.0xe000d02e97c0.DataSectionObject.Important.rar.dat
marek@marek-ubuntu:~/volatility3$
```

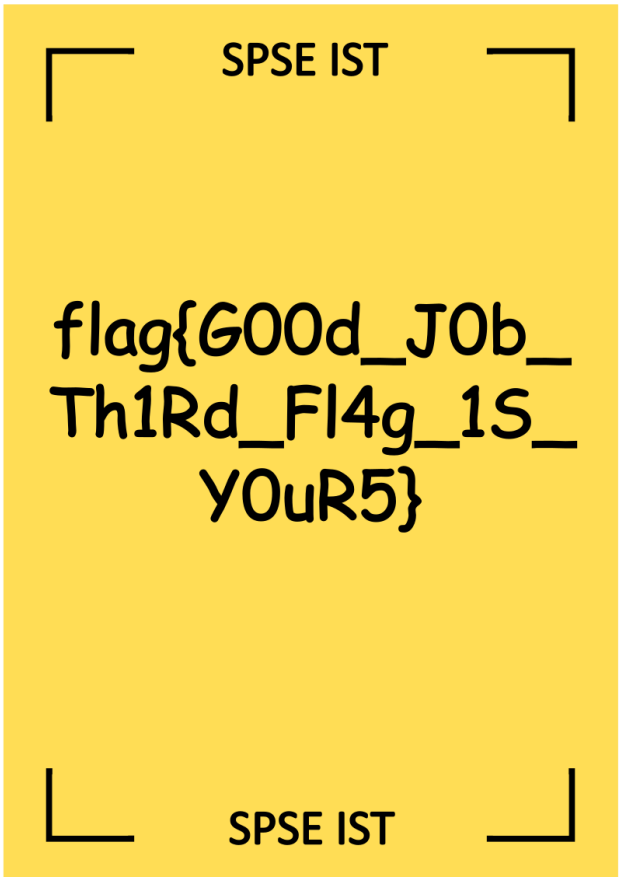
Príloha 4: Extrakcia súboru



Príloha 5: Okno WinRARu

```
3516    WinRAR.exe      0xa7d51013f0    USERNAME        Marek
3516    WinRAR.exe      0xa7d51013f0    USERPROFILE     C:\Users\Marek
3516    WinRAR.exe      0xa7d51013f0    windir  C:\Windows
3516    WinRAR.exe      0xa7d51013f0    WinRAR password Flag3Pa$$w0rd
4060    chrome.exe      0xa87db1420     WinRAR password Flag3Pa$$w0rd
3432    chrome.exe      0x6aee111420    WinRAR password Flag3Pa$$w0rd
2216    chrome.exe      0x7e4a091550    WinRAR password Flag3Pa$$w0rd
2216    chrome.exe      0x7e4a091550    WinRAR password Flag3Pa$$w0rd
2224    chrome.exe      0x51dd351550    WinRAR password Flag3Pa$$w0rd
2220    chrome.exe      0x2fbd101550    WinRAR password Flag3Pa$$w0rd
2348    chrome.exe      0x1526a71550    WinRAR password Flag3Pa$$w0rd
1552    svchost.exe     0x8b9fa02a50    WinRAR password Flag3Pa$$w0rd
4304    svchost.exe     0xc52fc02b00    WinRAR password Flag3Pa$$w0rd
4652    mspaint.exe     0x5642e213c0    WinRAR password Flag3Pa$$w0rd
4680    svchost.exe     0x21df502ac0    WinRAR password Flag3Pa$$w0rd
5024    WmiPrvSE.exe    0x4eea341220    WinRAR password Flag3Pa$$w0rd
4216    WmiPrvSE.exe    0x755e91220     WinRAR password Flag3Pa$$w0rd
2900    FTK Imager.exe  0x1a12e0        WinRAR password Flag3Pa$$w0rd
marek@marek-ubuntu:~/volatility3$
```

Príloha 6: Heslo v premenných prostredia



Príloha 7: Flag z archívu

```
PID:4060        Rule:r1 Component:$a
0x7ffe0f3c8b40: 6874 7470 733a 2f2f 6d65 6761 2e6e 7a2f  https://mega.nz/
0x7ffe0f3c8b50: 6669 6c65 2f51 756b 7953 5379 4b23 546a  file/QukySSyK#Tj
0x7ffe0f3c8b60: 6732 3277 4d6a 7953 3266 4857 5156 3764  g22wMjyS2fHWQV7d
0x7ffe0f3c8b70: 7668 724e 6669 7675 4239 7542 6263 6b59  vhrNfivuB9uBbckY
0x7ffe0f3c8b80: 5431 6662 7149 7073 3400 0000 0000 0000  T1fbqIps4.......
0x7ffe0f3c8b90: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8ba0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8bb0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8bc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8bd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8be0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8bf0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c00: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c10: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c20: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c30: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c50: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c60: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffe0f3c8c70: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Príloha 8: Výpis pluginu vadyarascan