



## **Analyysityö**

### **Nixu: Phishcap part 1 & Bad memories part 1**

Timo Lehosvuo, TTV18S1

Tuukka Bordi, TTV18S1

Harjoitustyö

Tunkeutumis- ja puolustusmenetelmät, Jarmo Nevala

25.4.2021

Tekniikan ala

## Sisältö

|                                     |           |
|-------------------------------------|-----------|
| <b>1. Phishcap part 1 .....</b>     | <b>3</b>  |
| <b>2. Bad memories part 1 .....</b> | <b>5</b>  |
| 2.1 Flag 3 .....                    | 10        |
| <b>3. Pohdinta.....</b>             | <b>12</b> |
| <b>Lähteet .....</b>                | <b>13</b> |

## Kuviot

|   |    |
|---|----|
| Kuvio 1: Ladattu dokumentti .....                   | 3  |
| Kuvio 2: Ladatut haittaohjelmat .....               | 3  |
| Kuvio 3: Dokumentin lataus .....                    | 4  |
| Kuvio 4: Ladatun dokumentin sisältö.....            | 4  |
| Kuvio 5: Teksti selkokielellä .....                 | 5  |
| Kuvio 6: Imageinfo .....                            | 5  |
| Kuvio 7: pslist tulokset .....                      | 6  |
| Kuvio 8: pslist grep notepad .....                  | 6  |
| Kuvio 9: Prosessien pid .....                       | 7  |
| Kuvio 10: flag.txt tiedoston poistaminen.....       | 7  |
| Kuvio 11: lsadump flag .....                        | 8  |
| Kuvio 12: dumpfiles komento .....                   | 8  |
| Kuvio 13: Strings komento putkitettu greppiin ..... | 9  |
| Kuvio 14: clipboard komennon tuloste.....           | 9  |
| Kuvio 15: notepad flag .....                        | 10 |
| Kuvio 16: mpaint memdump gimpsissä.....             | 11 |
| Kuvio 17: Kuva väärinpäin.....                      | 12 |
| Kuvio 18: mspaint flag.....                         | 12 |

## 1. Phishcap part 1

Aloitimme tehtävän laittamalla challenge.pcap -tiedoston paketit aika järjestykseen painamalla “Time” kohtaa sivun yläalaidassa. Tämän jälkeen aloimme selaamaan paketteja, kunnes löysimme paketin, jossa oli ladattu “invite\_to\_ski\_trip.docx” dokumentti:

|     |           |               |               |      |   |
|-----|-----------|---------------|---------------|------|---|
| 883 | 34.668345 | 10.100.10.100 | 51.15.75.147  | HTTP | 478 GET /invite_to_ski_trip.docx HTTP/1.1         |
| 884 | 34.668345 | 51.15.75.147  | 10.100.10.100 | TCP  | 54 80 → 49213 [ACK] Seq=1 Ack=425 Win=64240 Len=0 |

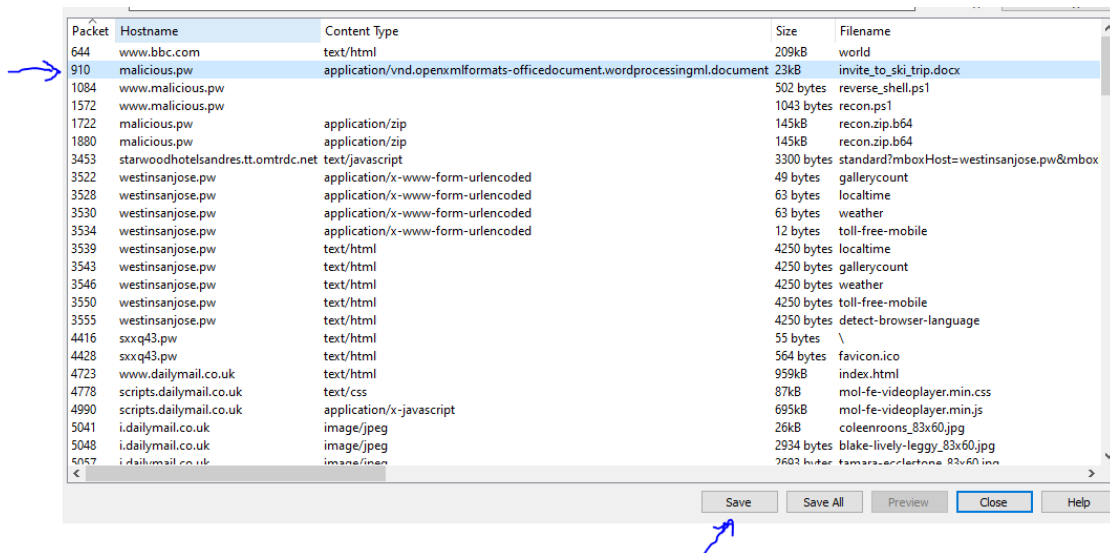
Kuvio 1: Ladattu dokumentti

Tämä liikenne vaikutti mielenkiintoiselta, joten filtteröimme liikenteen kohteen ip-osoitteen mukaan ja aloimme tutkimaan kohdetta mahdollisen muun hälyttävän liikenteen varalta. Huomasimme pian, että kohteesta oli ladattu haittaohjelmia koneelle tämän dokumentin lataamisen jälkeen:

|       |            |               |              |      |   |
|-------|------------|---------------|--------------|------|---|
| 883   | 34.668345  | 10.100.10.100 | 51.15.75.147 | HTTP | 478 GET /invite_to_ski_trip.docx HTTP/1.1   |
| 1082  | 42.865730  | 10.100.10.100 | 51.15.75.147 | HTTP | 143 GET /tools/reverse_shell.ps1 HTTP/1.1   |
| 1570  | 107.914708 | 10.100.10.100 | 51.15.75.147 | HTTP | 117 GET /tools/recon/recon.ps1 HTTP/1.1     |
| 1580  | 108.120254 | 10.100.10.100 | 51.15.75.147 | HTTP | 234 GET /tools/recon/recon.zip.b64 HTTP/1.1 |
| 1726  | 108.532465 | 10.100.10.100 | 51.15.75.147 | HTTP | 187 GET /tools/recon/recon.zip.b64 HTTP/1.1 |
| 12932 | 422.823314 | 10.100.10.100 | 51.15.75.147 | HTTP | 133 GET /tools/PowerUp.ps1 HTTP/1.1         |
| 14041 | 456.119001 | 10.100.10.100 | 51.15.75.147 | HTTP | 142 GET /tools/reverse_shell.cs HTTP/1.1    |
| 14063 | 466.839274 | 10.100.10.100 | 51.15.75.147 | HTTP | 110 GET /tools/data.zip HTTP/1.1            |
| 20518 | 642.477121 | 10.100.10.100 | 51.15.75.147 | HTTP | 145 GET /tools/Invoke-Mimikatz.ps1 HTTP/1.1 |
| 23096 | 704.182224 | 10.100.10.100 | 51.15.75.147 | HTTP | 145 GET /tools/Invoke-Mimikatz.ps1 HTTP/1.1 |

Kuvio 2: Ladatut haittaohjelmat

Varmistuttuamme kohteen haitallisuudesta päätimme ladata “invite\_to\_ski\_trip.docx” -dokumentin mahdollisen lipun varalta. Tämä tapahtui painamalla “File -> Export objects -> HTTP” ja painamalla “save”:



| Packet | Hostname                           | Content Type  | Size       | Filename                                |
|--------|------------------------------------|---|------------|---|
| 644    | www.bbc.com                        | text/html   | 209kB      | world                                   |
| 910    | malicious.pw                       | application/vnd.openxmlformats-officedocument.wordprocessingml.document | 23kB       | invite_to_ski_trip.docx                 |
| 1084   | www.malicious.pw                   |   | 502 bytes  | reverse_shell.ps1                       |
| 1572   | www.malicious.pw                   |   | 1043 bytes | recon.ps1                               |
| 1722   | malicious.pw                       | application/zip   | 145kB      | recon.zip.b64                           |
| 1880   | malicious.pw                       | application/zip   | 145kB      | recon.zip.b64                           |
| 3453   | stanwoodhotelsandres.tt.omtrdc.net | text/javascript   | 3300 bytes | standard?mboxHost=westinsanjose.pw&mbox |
| 3522   | westinsanjose.pw                   | application/x-www-form-urlencoded                                       | 49 bytes   | gallerycount                            |
| 3528   | westinsanjose.pw                   | application/x-www-form-urlencoded                                       | 63 bytes   | localtime                               |
| 3530   | westinsanjose.pw                   | application/x-www-form-urlencoded                                       | 63 bytes   | weather                                 |
| 3534   | westinsanjose.pw                   | application/x-www-form-urlencoded                                       | 12 bytes   | toll-free-mobile                        |
| 3539   | westinsanjose.pw                   | text/html   | 4250 bytes | localtime                               |
| 3543   | westinsanjose.pw                   | text/html   | 4250 bytes | gallerycount                            |
| 3546   | westinsanjose.pw                   | text/html   | 4250 bytes | weather                                 |
| 3550   | westinsanjose.pw                   | text/html   | 4250 bytes | toll-free-mobile                        |
| 3555   | westinsanjose.pw                   | text/html   | 4250 bytes | detect-browser-language                 |
| 4416   | sxq43.pw                           | text/html   | 55 bytes   | \                                       |
| 4428   | sxq43.pw                           | text/html   | 564 bytes  | favicon.ico                             |
| 4723   | www.dailymail.co.uk                | text/html   | 959kB      | index.html                              |
| 4778   | scripts.dailymail.co.uk            | text/css  | 87kB       | mol-fe-videoplayer.min.css              |
| 4990   | scripts.dailymail.co.uk            | application/x-javascript  | 695kB      | mol-fe-videoplayer.min.js               |
| 5041   | i.dailymail.co.uk                  | image/jpeg  | 26kB       | coleenrooms_83x60.jpg                   |
| 5048   | i.dailymail.co.uk                  | image/jpeg  | 2934 bytes | blake-lively-leggy_83x60.jpg            |
| 5057   | i.dailymail.co.uk                  | image/jpeg  | 7603 bytes | tamara-cerletone_83x60.jpg              |

Kuvio 3: Dokumentin lataus

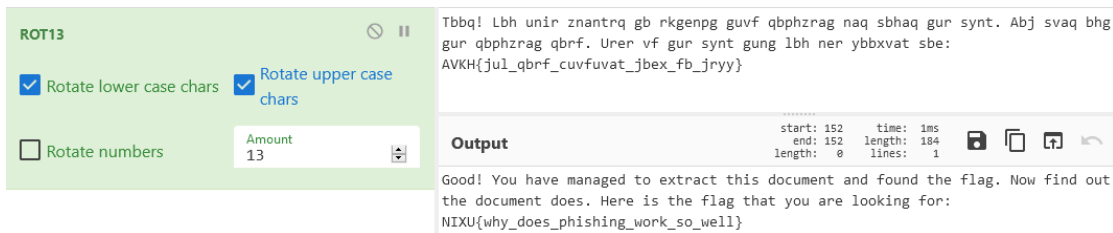
Dokumentti piti sisällään tekstin: "Tbbq! Lbh unir znantrq gb rkgenpg guvf qbphzrag naq sbhaq gur synt. Abj svaq bhg jung gur qbphzrag qbrf. Urer vf gur synt gung lbh ner ybbxvat sbe: AVKH{jul\_qbrf\_cuvfuvat\_jbex\_fb\_jryy}":

Unable to display images

Tbbq! Lbh unir znantrq gb rkgenpg guvf qbphzrag naq sbhaq gur synt. Abj svaq bhg jung gur qbphzrag qbrf. Urer vf gur synt gung lbh ner ybbxvat sbe: AVKH{jul\_qbrf\_cuvfuvat\_jbex\_fb\_jryy}

Kuvio 4: Ladatun dokumentin sisältö

Teksti oli selvästi salattu ja tekstin loppuosa muistutti hyvin paljon lipun syntaksia, joten purimme tekstissä käytetyn salauksen (ROT13) käyttämällä "cyberchef" -sivua:



Kuvio 5: Teksti selkokielellä

Selkoteksti paljasti lipuksemme: “NIXU{why\_does\_phishing\_work\_so\_well}”

## 2. Bad memories part 1

Aloitimme tehtävän ajamalla tiedoston komennolla “vol.py -f mem.dmp imageinfo” (P4N4Rd1 2019), tämän avulla saimme tarvitsemamme profiilin millä pystyimme ajamaan plugineja:

```
root@kali:~/volatility# vol.py -f ../Desktop/mem.dmp imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000, Win7SP1x64_23418
      AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/root/Desktop/mem.dmp)
      PAE type : No PAE
      DTB : 0x187000L
      KDBG : 0xf80002a03110L
      Number of Processors : 4
      Image Type (Service Pack) : 1
      KPCR for CPU 0 : 0xffffffff80002a04d00L
      KPCR for CPU 1 : 0xffffffff800009ee000L
      KPCR for CPU 2 : 0xffffffff80002f69000L
      KPCR for CPU 3 : 0xffffffff80002fdf000L
      KUSER_SHARED_DATA : 0xffffffff78000000000L
      Image date and time : 2018-12-20 05:30:11 UTC+0000
      Image local date and time : 2018-12-19 21:30:11 -0800
root@kali:~/volatility#
```

Kuvio 6: Imageinfo

Päättelimme komennon tulosteesta, että oikea profiili dumpille olisi “Win7SP1x64”, tai ainakin se voisi olla lähellä oikeaa. Profiilin selvittämisen jälkeen ajoimme komennon “pslist” saamallamme profiililla (P4N4Rd1 2019), komento näytti järjestelmän prosessit:

```

0xffffffff80014d2060 mspaint.exe          2816    1840      8      184      1
0 2018-12-20 05:29:18 UTC+0000
0xffffffff800173eb10 svchost.exe          2724     428      9      113      0
0 2018-12-20 05:29:18 UTC+0000
0xffffffff8003caf060 notepad.exe           700    1840      2       57      1
0 2018-12-20 05:29:22 UTC+0000
0xffffffff8003be0060 dllhost.exe          3268     604     11     240      1
0 2018-12-20 05:30:05 UTC+0000
0xffffffff8003bda930 winpmem-2.1.po        3408    1840      1      47      1
1 2018-12-20 05:30:11 UTC+0000
0xffffffff8000d89b10 conhost.exe          3420     384      2      53      1
0 2018-12-20 05:30:11 UTC+0000
0xffffffff8003bfa660 svchost.exe          3536     428     12     260      0
0 2018-12-20 05:30:22 UTC+0000
root@kali:~/volatility#

```

Kuvio 7: pslist tulokset

Silmiimme pisti useampikin prosessit, mutta vahvimpana oli “notepad.exe”. Alla sama komento, mutta tuloksista haettu notepad.exe:

```

root@kali:~/volatility# vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 pslist
| grep notepad
Volatility Foundation Volatility Framework 2.6.1
0xffffffff8003caf060 notepad.exe          700    1840      2       57      1
0 2018-12-20 05:29:22 UTC+0000
root@kali:~/volatility#

```

Kuvio 8: pslist grep notepad

Emme kuitenkaan vielä aloittaneet syvempää notepadin tutkimista vaan jatkoimme tiedoston tutkimista muilla komennoilla. Ajoimme seuraavaksi komennon “vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 cmdline”(Command Reference: volatilityfoundation/volatility Wiki 2020). Tämä osoittautui hyödylliseksi sillä niistä selvisi prosessien pid, joista oli hyötyä lippujen selvittämisessä sekä esimerkiksi sen, että paintissa on ollut auki “flag.bmp” tiedosto:

```

*****
mspaint.exe pid: 2816
Command line : "C:\Windows\system32\mspaint.exe" "C:\Users\Alice\Pictures\flag.b
mp"
*****
svchost.exe pid: 2724
Command line : C:\Windows\system32\svchost.exe -k imgsvc
*****
notepad.exe pid: 700
Command line : "C:\Windows\system32\notepad.exe"
*****
dllhost.exe pid: 3268
Command line : C:\Windows\system32\DllHost.exe /Processid:{76D0CB12-7604-4048-B8
3C-1005C7DDC503}
*****
winpmem-2.1.po pid: 3408
Command line : "C:\Users\Alice\Downloads\winpmem-2.1.post4.exe" -o mem.aff4
*****
conhost.exe pid: 3420
Command line : \??\C:\Windows\system32\conhost.exe "-138943212138372032818097864
32-1020953099-987546483-771985901049490496-1582483814
*****
svchost.exe pid: 3536
root@kali:~/volatility#

```

Kuvio 9: Prosessien pid

Seuraavasta komennosta “cmdscan” saimme ulos jotain tietoa, mitä komentorivillä on tehty (Com-  
mand Reference: volatilityfoundation/volatility Wiki 2020). Kuvasta näkyy, että tiedosto nimeltä  
“flag.txt” on poistettu.

```

root@kali:~/volatility# !190
vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 cmdscan
Volatility Foundation Volatility Framework 2.6.1
*****
CommandProcess: conhost.exe Pid: 1440
CommandHistory: 0x2215a0 Application: sshd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x58
*****
CommandProcess: conhost.exe Pid: 1144
CommandHistory: 0x208e00 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 5 LastAdded: 4 LastDisplayed: 4
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x10
Cmd #0 @ 0x1fd210: dir Documents
Cmd #1 @ 0x1fd240: dir Pictures
Cmd #2 @ 0x1fd270: cd Documents
Cmd #3 @ 0x20d230: del flag.txt
Cmd #4 @ 0x2076f0: dir
Cmd #15 @ 0x1a0158:
Cmd #16 @ 0x207790:
*****
CommandProcess: conhost.exe Pid: 3420
CommandHistory: 0x1c8e50 Application: winpmem-2.1.post4.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x64
Cmd #15 @ 0x160158:
Cmd #16 @ 0x1c77e0:
root@kali:~/volatility#

```

Kuvio 10: flag.txt tiedoston poistaminen

Ensimmäinen lippu, jonka löysimme, oli kuva nimeltä “flag.bmp” joka sijaitsi Alicen profiilissa ja  
toinen “flag.txt”, joka ilmeisesti sijaitsi roskakorissa. Jatkoimme tutkimista ja ajoimme komennon

“lsadump” (Command Reference: [volatilityfoundation.org/volatility](https://volatilityfoundation.org/volatility) Wiki 2020), josta saimmekin ensimmäisen lippumme ulos “NIXU{was\_it\_even\_hard\_for\_you?}”:

```
root@kali:~/volatility# !230
vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 lsadump
Volatility Foundation Volatility Framework 2.6.1
DefaultPassword
0x00000000 3e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 >.....
0x00000010 4e 00 49 00 58 00 55 00 7b 00 77 00 61 00 73 00 N.I.X.U.{w.a.s.
0x00000020 5f 00 69 00 74 00 5f 00 65 00 76 00 65 00 6e 00 _i.t._e.v.e.n.
0x00000030 5f 00 68 00 61 00 72 00 64 00 5f 00 66 00 6f 00 _h.a.r.d._f.o.
0x00000040 72 00 5f 00 79 00 6f 00 75 00 3f 00 7d 00 00 00 r._y.o.u.?}...

SC_OpenSSHd
0x00000000 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000010 44 00 40 00 72 00 6a 00 33 00 33 00 6c 00 31 00 D.@.r.j.3.3.l.l.
0x00000020 6e 00 67 00 00 00 00 00 00 00 00 00 00 00 00 n.g.....

DPAPI_SYSTEM
0x00000000 2c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x00000010 01 00 00 00 d7 e7 10 e0 e1 fe e9 a6 fa 72 6c e5 .....rl.
0x00000020 6d bd f2 fb b3 20 2d 1e ac 17 fe 50 74 dd ae a2 m.....Pt...
0x00000030 1a 32 dc d9 18 b6 5f 26 91 b1 dd c4 00 00 00 00 .2...._&.....

root@kali:~/volatility#
```

Kuvio 11: lsadump flag

Selvisi, että tämä lippu ei ollutkaan se mitä haimme, joten jatkoimme tutkintaa (lippu oli ratkaisu Bad memories part 5). Tämän jälkeen ajoimme komennon “dumpfiles”, joka tulosti kaikki muistidumpin sisältämät tiedostot. Latasimme monia komennon paljastamia tiedostoja (Whiteheart 2019), mutta yksikään niistä ei paljastanut mitään hyödyllistä. Esimerkkikomento, josta ei ollut hyötyä (ei antanut tiedostoa ulos):

```
280 vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 dumpfiles -Q 0x00000000
15cb5c80 --name file -D pictures/
```

Kuvio 12: dumpfiles komento

Seuraavaksi palasimme tutkimaan alussa löytäämämme “notepad.exe” prosessia. Otimme “notepad.exe” prosessin sisältämän muistin talteen komennolla “vol.py -f mem.dmp --profile=Win7SP1x64 memdump -p 700 -dump-dir=./” (Command Reference: [volatilityfoundation.org/volatility](https://volatilityfoundation.org/volatility) Wiki 2020) ja tutkimme komennon luomaa “700.dmp” tiedostoa “strings” ohjelmalla (Linux strings command – javatpoint n.d). Testailimme useaa eri vaihtoehtoa mahdolliselle lipulle mutta tuloksetta:



```

root@kali:~/Desktop/vt# strings 700.dmp | grep flag_1
root@kali:~/Desktop/vt# strings 700.dmp | grep NIXU
root@kali:~/Desktop/vt# strings 700.dmp | grep AVKH
root@kali:~/Desktop/vt#

```

Kuvio 13: Strings komento putkitettu greppiin

Koska yksikään string komento ei tuottanut tulosta aloimme tutkimaan tiedostoa käsin komenolla “strings 700.dmp | grep {”, koska tiesimme lipun syntaksin sisältävän merkin “{”. Reilun tunnin tutkimisen jälkeen emme olleet vielä tutkineet koko tiedostoa, joten aloimme etsimään tietoa netistä. Verkosta selvisi että “notepad” käyttää eri tavujärjestystä ja “volatilessa” on komento “clipboard” millä näkee leikepöydän (Command Reference Gui [clipboard], Andrea Fortuna 2018): volatilityfoundation/volatility Wiki 2017). Jatkoimme tutkimista näitä hyväksikäyttäen ja ajoimme “clipboard” komennon:

```

root@kali:~/Desktop/vt# vol.py -f /root/Desktop/vt/mem.dmp --profile=Win7SP1x64
clipboard
Volatility Foundation Volatility Framework 2.6.1
Session WindowStation Format Handle Object
Data
-----
1 WinSta0 CF_UNICODETEXT 0x40127 0xffffffff900c0792dc
0 you're on the right track! try harder
1 WinSta0 CF_TEXT 0x7400000000 -----
-
1 WinSta0 CF_LOCALE 0x30129 0xffffffff900c1d5ce7
0
1 WinSta0 0x0L 0x0 -----
-
root@kali:~/Desktop/vt#

```

Kuvio 14: clipboard komennon tuloste

Komento paljasti tekstin “you’re on the right track! try harder”. Palasimme takaisin “notepadin” tutkimiseen ja kokeilimme “string” komentoa “grep” arvolla “try harder” käyttäen eri tavujärjestystä ja komento tulosti tekstin, joka muistutti hyvin paljon lippua:

```

root@kali:~/Desktop/vt# strings -e l 700.dmp | grep 'try harder'
you're on the right track! try harder
n the right track! try harder
AVKH{guvf_j4f_gu3_rnfl_bar}try harder
you're on the right track! try harder
you're on the right track! try harder
root@kali:~/Desktop/vt#

```

Kuvio 15: notepad flag

Purimme tekstin salauksen käyttäen ROT13 salauksen purkua ja saimme lipun

“NIXU{this\_w4s\_th3\_easy\_one}”. Testasimme lippua ja se paljastui “Bad memory part 1” lipuksi.

## 2.1 Flag 3

Löysimme flag 3:n vain muutama minuutti sen jälkeen, kun saimme flag 1:n, joten päätimme dokumentoida prosessin. Tutkimme tosiaan eri osa-alueita säästääksemme aikaa lipun löytämisessä. Netistä löydetyn materiaalin perusteella selvisi, että mspaint-prosessista voi saada ohjelmassa muokattavana olevan kuvan ulos, jos ottaa ensiksi datadumpin prosessista komennolla “mem-dump” ja aukaisee sen jälkeen tiedoston gimp-sovelluksella. Gimpissä voi prosessoida tätä dumpia ja kokeilla eri offseteilla, sisältääkö raakadata kuvatiedoston (näkyv esikatselussa). (Rodrigues 2015.)

```

root@kali:~/volatility# vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 pslist
| grep mspaint
Volatility Foundation Volatility Framework 2.6.1
0xfffffa80014d2060 mspaint.exe          2816    1840         8      184      1
  0 2018-12-20 05:29:18 UTC+0000
root@kali:~/volatility#

```

Figure 1: mspaint pid

Ylhäällä selvitimme mspaint.exe:n prosessi-id:n (P4N4Rd1 2019), ja tämän avulla pystyimme dumpata prosessin muistin (vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 memdump -p 2816 --dump-dir paint/) (Command Reference: volatilityfoundation/volatility Wiki 2020):

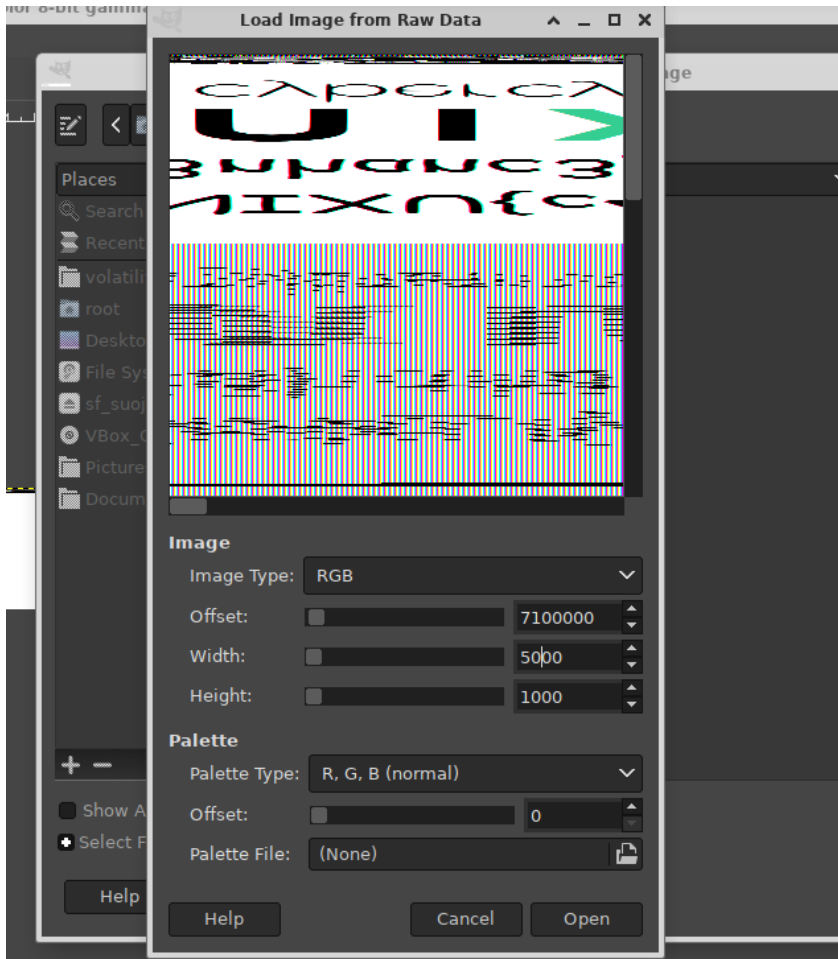
```

root@kali:~/volatility# vol.py -f ../Desktop/mem.dmp --profile=Win7SP1x64 memdump
-p 2816 --dump-dir paint/
Volatility Foundation Volatility Framework 2.6.1
*****
Writing mspaint.exe [ 2816] to 2816.dmp
root@kali:~/volatility#

```

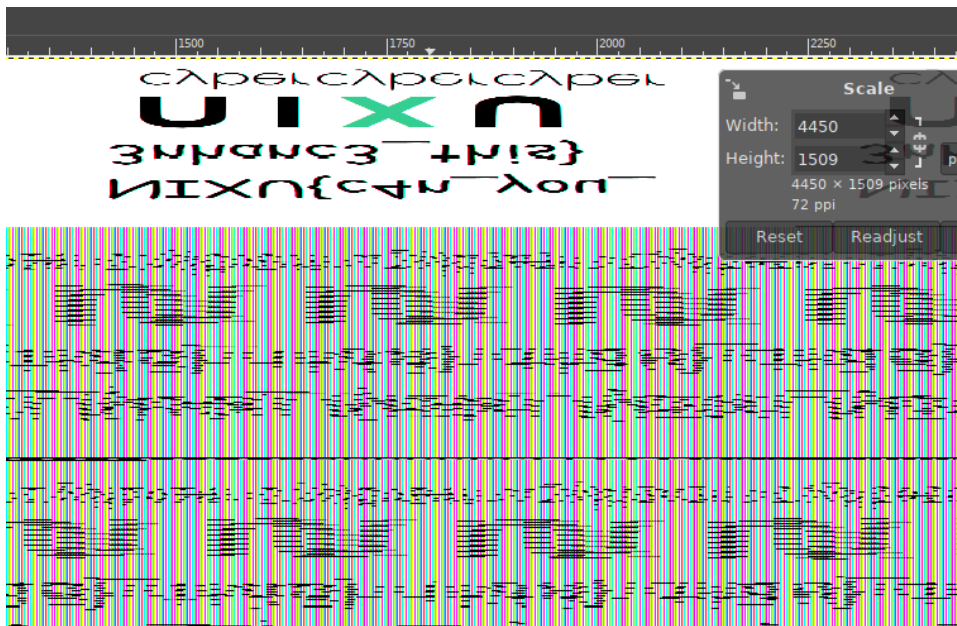
Figure 2: mspaint memdump

Dump-tiedoston tiedostopääte täytyy sen jälkeen muuttaa ".dump" päätteestä ".data":ksi, jotta sen voi avata gimpillä (Rodrigues 2015). Sen jälkeen pitää vain kasvattaa offsettiä kunnes kuva ilmestyy esikatseluun (Rodrigues 2015):



Kuvio 16: mpaint memdump gimpsissä

Lopulta kuva ilmestyi offsetissä 7100000. Kuvan leveyttä piti kasvattaa 5000:een ja korkeutta 1000:een, että esikatselu kertoi enemmän. Kuva oli kuitenkin gimpissä väärin päin:



Kuvio 17: Kuva väärinpäin

Parin gimp-operaation jälkeen saimme kuvan kuitenkin oikein päin, ja saimme ulos lipun “NIXU{c4n\_you\_3nhanc3\_this}”.



Kuvio 18: mspaint flag

### 3. Pohdinta

Oli sinänsä aika huvittavaa, että saimme lippuja, mutta flag 1 jäi melkein viimeiseksi. Hauskaa kuitenkin oli, ja opimme paljon muistianalyysistä. Kuvan saaminen ulos paint-sovelluksesta oli kiinnostavaa.

## Lähteet

Command Reference Gui [clipboard]: volatilityfoundation/volatility Wiki. 2017. Volatility-työkalun Github –sivut. Viitattu 24.4.2021. <https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Gui#clipboard>.

Command Reference: volatilityfoundation/volatility Wiki. 2020. Volatility-työkalun Github –sivut. Viitattu 24.4.2021. <https://github.com/volatilityfoundation/volatility/wiki/Command-Reference>.

Fortuna, A. 2018. Article on Andrea Fortuna’s homepage. Viitattu 24.4.2021. <https://www.andreafortuna.org/2018/03/02/volatility-tips-extract-text-typed-in-a-notepad-window-from-a-windows-memory-dump/>.

Linux strings command – javatpoint. N.d. Artikkelin Javatpoint –sivustolla. Viitattu 24.4.2021. <https://www.javatpoint.com/linux-strings-command>.

P4N4Rd1 [nimimerkki]. 2019. Artikkelin Medium-sivustolla. Viitattu 24.4.2021. <https://medium.com/@zemelusa/first-steps-to-volatile-memory-analysis-dcbd4d2d56a1>.

Rodrigues, B. 2015. Artikkelin Andrea Rodriguesin kotisivuilla. Viitattu 24.4.2021. <https://w00tsec.blogspot.com/2015/02/extracting-raw-pictures-from-memory.html>.

Whiteheart [nimimerkki]. 2019. Artikkelin Mediumin sivustolla. Viitattu 24.4.2021. <https://whiteheart0.medium.com/retrieving-files-from-memory-dump-34d9fa573033>.