Challenge: BankingTroubles Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Hard

Tools Used: Volatility 2, strings, foremost, peepdf, pdf-parser, jsunpack

Summary: This lab involves investigating a compromised Windows XP host. Initial access was achieved through a phishing email containing a link to a malicious PDF that, when opened in Acrobat Reader, executed embedded JavaScript leading to infection. Analysis identified that firefox.exe spawned AcroRd32.exe which in turn, executed the malicious JavaScript. This was a challenging yet enjoyable lab, I highly recommend giving it a go.

Scenario: Company X has contacted you to perform forensics work on a recent incident that occurred. One of their employees had received an e-mail from a co-worker that pointed to a PDF file. Upon opening, the employee did not notice anything; however, they recently had unusual activity in their bank account.

The initial theory is that a user received an e-mail, containing an URL leading to a forged PDF document. Opening that document in Acrobat Reader triggers a malicious Javascript that initiates a sequence of actions to take over the victim's system.

Company X was able to obtain a memory image of the employee's virtual machine upon suspected infection and asked you as a security blue team analyst to analyze the virtual memory and provide answers to the questions.

What was the local IP address of the victim's machine?

TLDR: Use the connections plugin and focus on the local address.

Let's start by identifying the profile and KDBG address for this memory image:

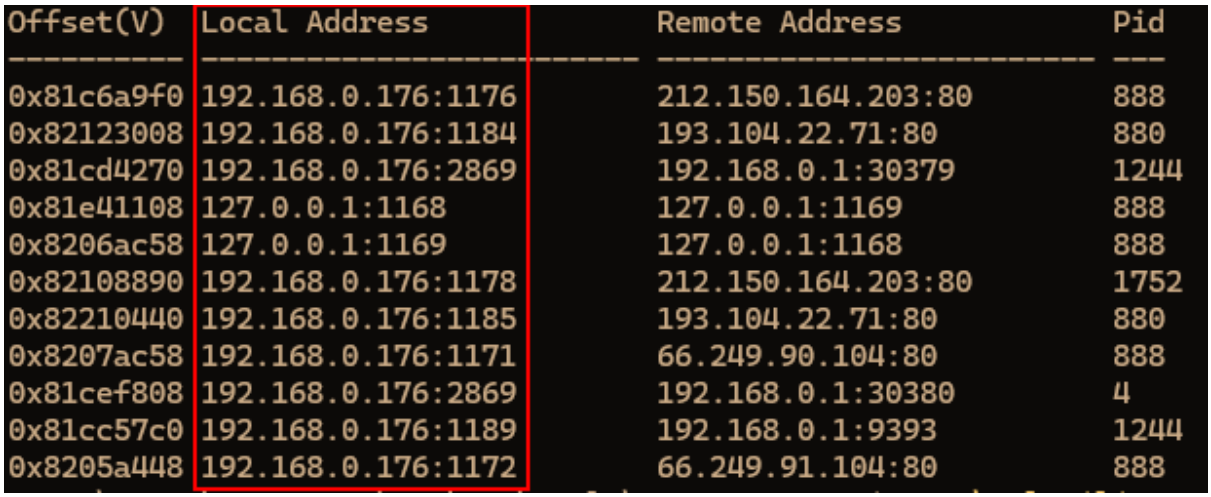
- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" imageinfo`

```
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\DFIR-USER\Desktop\43-banking-troubles\temp_extract_dir\Bob.vmem)
                      PAE type : PAE
                           DTB : 0x319000L
                          KDBG : 0x80544ce0L
          Number of Processors : 1
     Image Type (Service Pack) : 2
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2010-02-27 20:12:38 UTC+0000
     Image local date and time : 2010-02-27 15:12:38 -0500
```

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 kdbgscan`

Unfortunately, this profile does not support the netscan plugin, so we can use a plugin called connections instead. This plugin displays a list of network connections present at the time the memory dump was taken:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 connections`



Offset(V)	Local Address	Remote Address	Pid
0x81c6a9f0	192.168.0.176:1176	212.150.164.203:80	888
0x82123008	192.168.0.176:1184	193.104.22.71:80	880
0x81cd4270	192.168.0.176:2869	192.168.0.1:30379	1244
0x81e41108	127.0.0.1:1168	127.0.0.1:1169	888
0x8206ac58	127.0.0.1:1169	127.0.0.1:1168	888
0x82108890	192.168.0.176:1178	212.150.164.203:80	1752
0x82210440	192.168.0.176:1185	193.104.22.71:80	880
0x8207ac58	192.168.0.176:1171	66.249.90.104:80	888
0x81cef808	192.168.0.176:2869	192.168.0.1:30380	4
0x81cc57c0	192.168.0.176:1189	192.168.0.1:9393	1244
0x8205a448	192.168.0.176:1172	66.249.91.104:80	888

Here we can see the local address of “192.168.0.176”, which is the IP of the victim’s machine.

Alternatively, you can find the IP of this host in registry located at:

- `SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces`

Using the printkey plugin, we can print the registry key for an available interface:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 printkey -K "ControlSet001\Services\Tcpip\Parameters\Interfaces\{6A1CDCB9-3F64-4C3B-A932-E55C5FA47352}"`

```
Values:
REG_DWORD     UseZeroBroadcast : (S) 0
REG_DWORD     EnableDeadGWDetect : (S) 1
REG_DWORD     EnableDHCP       : (S) 1
REG_MULTI_SZ  IPAddress        : (S) ['0.0.0.0', '', '']
REG_MULTI_SZ  SubnetMask       : (S) ['0.0.0.0', '', '']
REG_MULTI_SZ  DefaultGateway   : (S) ['', '']
REG_MULTI_SZ  DefaultGatewayMetric : (S) ['', '']
REG_SZ        NameServer       : (S)
REG_SZ        Domain           : (S)
REG_DWORD     RegistrationEnabled : (S) 1
REG_DWORD     RegisterAdapterName : (S) 0
REG_MULTI_SZ  TCPAllowedPorts  : (S) ['0', '', '']
REG_MULTI_SZ  UDPAllowedPorts  : (S) ['0', '', '']
REG_MULTI_SZ  RawIPAllowedProtocols : (S) ['0', '', '']
REG_MULTI_SZ  NTEContextList   : (S) ['0x00000002', '', '']
REG_DWORD     DhcpClassIdBin   : (S) 3
REG_SZ        DhcpServer       : (S) 192.168.0.1
REG_DWORD     Lease            : (S) 86400
REG_DWORD     LeaseObtainedTime : (S) 1267300134
REG_DWORD     T1               : (S) 1267343334
REG_DWORD     T2               : (S) 1267375734
REG_DWORD     LeaseTerminatesTime : (S) 1267386534
REG_SZ        IPAutoconfigurationAddress : (S) 0.0.0.0
REG_SZ        IPAutoconfigurationMask : (S) 255.255.0.0
REG_DWORD     IPAutoconfigurationSeed : (S) 0
REG_DWORD     AddressType      : (S) 0
REG_SZ        DhcpIPAddress    : (S) 192.168.0.176
REG_SZ        DhcpSubnetMask   : (S) 255.255.255.0
REG_DWORD     DhcpRetryTime    : (S) 43197
REG_DWORD     DhcpRetryStatus : (S) 0
REG_MULTI_SZ  DhcpSubnetMaskOpt : (S) ['255.255.255.0', '', '']
REG_SZ        DhcpDomain       : (S) rh.rit.edu.
REG_MULTI_SZ  DhcpDefaultGateway : (S) ['192.168.0.1', '', '']
REG_SZ        DhcpNameServer   : (S) 192.168.0.1
```

Answer: 192.168.0.176

What was the OS environment variable's value?

TLDR: use the envars plugin and grep for “OS” to find the OS environment variable.

For context, an environment variable is a global variable accessible by all the processes running on the OS. To print all the environment variables on this host, we can use the envars plugin and filter for “OS”:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 envars | Select-String -Pattern "OS"`

```
0x00100000 FP_NO_HOST_CHECK              NO
0x00100000 OS                            Windows_NT
0x00010000 FP_NO_HOST_CHECK              NO
0x00010000 OS                            Windows_NT
0x00010000 FP_NO_HOST_CHECK              NO
0x00010000 OS                            Windows_NT
0x00010000 FP_NO_HOST_CHECK              NO
0x00010000 OS                            Windows_NT
0x00010000 FP_NO_HOST_CHECK              NO
0x00010000 OS                            Windows_NT
```

We can see that the OS environment variable is given the value “Windows_NT”.

Answer: Windows_NT

What was the Administrator's password?

TLDR: Use hashdump to extract password hashes from memory, crack the NT hash for the Administrator user using a tool like CrackStation.

Using the hashdump plugin, we can extract user password hashes from the memory dump:

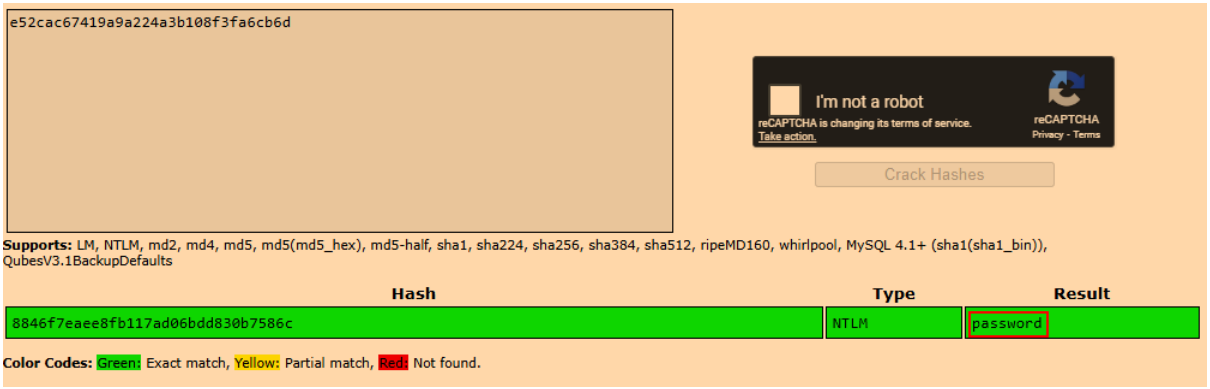
- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 hashdump`

```
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:9f8ac2eaebcd2e3a6f94d53c19803662:d95e38a172b3ddaa1ce0b63bb1f5e1fb:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:ad052c1cbab3ec2502df165cd25d95bd:::
```

The format is as follows:

- Username
- Relative Identifier (RID)
- LM hash
- NT hash

Each separated by a semicolon (:). If you copy the NT hash, we can use a tool like CrackStation to crack the hash:



Answer: password

Which process was most likely responsible for the initial exploit?

Let's start by using the pstree plugin to list the parent-child relationships of the processes running at the time of the memory dump:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 pstree`

If you recall in the scenario, an employee received an email that contained a URL leading to a forged PDF. Opening that document in Acrobat Reader triggered a malicious JavaScript payload.

In the output, we can see firefox.exe (PID 1108) spawned AcroRd32.exe (PID 1752), which matches the scenario infection chain:

```
Name                                               Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x81cdd790:explorer.exe                          1756   1660     14    345 2010-02-26 03:34:38 UTC+0000
. 0x820cd5c8:VMwareUser.exe                       1116   1756      4    179 2010-02-26 03:34:39 UTC+0000
. 0x81ca96f0:VMwareTray.exe                       1108   1756      1     59 2010-02-26 03:34:39 UTC+0000
. 0x82068020:firefox.exe                           888   1756      9    172 2010-02-27 20:11:53 UTC+0000
.. 0x820618c8:AcroRd32.exe                        1752    888      8    184 2010-02-27 20:12:23 UTC+0000
 0x823c8830:System                                   4      0     58    573 1970-01-01 00:00:00 UTC+0000
. 0x81f04228:smss.exe                             548      4      3     21 2010-02-26 03:34:02 UTC+0000
.. 0x81e5b2e8:winlogon.exe                        644    548     21    521 2010-02-26 03:34:04 UTC+0000
... 0x82256da0:services.exe                       688    644     16    293 2010-02-26 03:34:05 UTC+0000
.... 0x822ea020:svchost.exe                      1040    688     83   1515 2010-02-26 03:34:07 UTC+0000
..... 0x81c80c78:wuauclt.exe                      440   1040      8    188 2010-02-27 19:48:49 UTC+0000
..... 0x81cee5f8:wscntfy.exe                     1132   1040      1     38 2010-02-26 03:34:40 UTC+0000
..... 0x8221a020:wuauclt.exe                      232   1040      4    136 2010-02-27 19:49:11 UTC+0000
.... 0x81de55f0:svchost.exe                      1244    688     19    239 2010-02-26 03:34:08 UTC+0000
.... 0x81ddd8d0:VMUpgradeHelper                  1836    688      4    108 2010-02-26 03:34:34 UTC+0000
.... 0x81dde568:spoolsv.exe                      1460    688     11    129 2010-02-26 03:34:10 UTC+0000
.... 0x822e1da0:svchost.exe                       948    688     10    276 2010-02-26 03:34:07 UTC+0000
.... 0x81dea020:svchost.exe                      1100    688      6     96 2010-02-26 03:34:07 UTC+0000
.... 0x821018b0:vmtoolsd.exe                     1628    688      5    220 2010-02-26 03:34:25 UTC+0000
.... 0x82209640:svchost.exe                      1384    688      9    101 2010-02-27 20:12:36 UTC+0000
.... 0x820d6b88:alg.exe                          2024    688      7    130 2010-02-26 03:34:35 UTC+0000
.... 0x82266870:svchost.exe                       880    688     28    340 2010-02-26 03:34:07 UTC+0000
.... 0x82333620:msiexec.exe                       244    688      5    181 2010-02-26 03:46:06 UTC+0000
..... 0x81ce1af8:msiexec.exe                      452    244      0 ------ 2010-02-26 03:46:07 UTC+0000
.... 0x81d3f020:vmacthlp.exe                      852    688      1     35 2010-02-26 03:34:06 UTC+0000
... 0x82129da0:lsass.exe                          700    644     22    416 2010-02-26 03:34:06 UTC+0000
.. 0x822eeda0:csrss.exe                           612    548     12    423 2010-02-26 03:34:04 UTC+0000
```

Answer: AcroRd32.exe

What is the extension of the malicious file retrieved from the process responsible for the initial exploit?

Let's start by using the filescan plugin to search for PDF files:

- `. \volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 filescan | Select-String -Pattern ".pdf"`

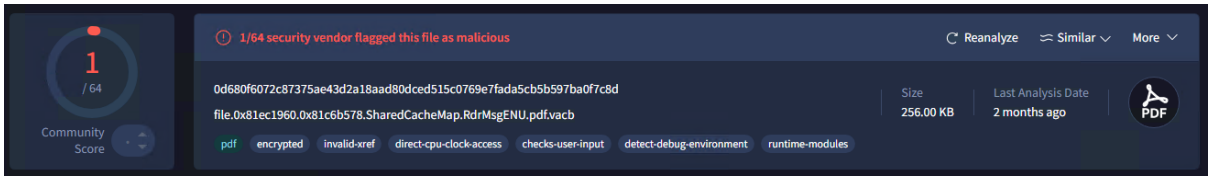
One file really stands out due to its odd filename:

```
R--r-d \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\ActiveX\pdf.ocx
R--r-- \Device\HarddiskVolume1\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
R--r-- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\Messages\ENU\RdrMsgENU.pdf
R--r-- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\ActiveX\pdf.ocx
R--r-d \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\Browser\nppdf32.dll
-W---- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\plug_ins\PictureTasks\Templates\legal_46.pdf
-W---- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\plug_ins\PictureTasks\Templates\B5_fit.pdf
-W---- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\plug_ins\PictureTasks\Templates\A4_fit.pdf
R--r-- \Device\HarddiskVolume1\Program Files\Adobe\Acrobat 6.0\Reader\Messages\ENU\RdrMsgENU.pdf
```

If you dump this file using the dumpfiles plugin:

- `\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 dumpfiles -Q 0x00000000020c1960 -D .`

And hash it, we can see that it receives one detection on VirusTotal (which isnt a lot):



However, in the comments, we can see that multiple sandbox environments have detected this file as suspicious.

Answer: pdf

Suspicious processes opened network connections to external IPs. One of them starts with "2". Provide the full IP.

Using the connections plugin, we can see two connections made to a remote address starting with “2”:

Offset(V)	Local Address	Remote Address	Pid
0x81c6a9f0	192.168.0.176:1176	212.150.164.203:80	888
0x82123008	192.168.0.176:1184	193.104.22.71:80	880
0x81cd4270	192.168.0.176:2869	192.168.0.1:30379	1244
0x81e41108	127.0.0.1:1168	127.0.0.1:1169	888
0x8206ac58	127.0.0.1:1169	127.0.0.1:1168	888
0x82108890	192.168.0.176:1178	212.150.164.203:80	1752
0x82210440	192.168.0.176:1185	193.104.22.71:80	880
0x8207ac58	192.168.0.176:1171	66.249.90.104:80	888
0x81cef808	192.168.0.176:2869	192.168.0.1:30380	4
0x81cc57c0	192.168.0.176:1189	192.168.0.1:9393	1244
0x8205a448	192.168.0.176:1172	66.249.91.104:80	888

The PIDs resolve to firefox.exe and AcroRd32.exe respectively.

Answer: 212.150.164.203:80

A suspicious URL was present in process svchost.exe memory. Provide the full URL that points to a PHP page hosted over a public IP (no FQDN).

TLDR: Run strings against the memory dump and grep for URLs, focusing on URLs that include an IP rather than a domain name.

Using the pstree plugin, we can see that all svchost.exe processes are legitimate (i.e., there is no process masquerading as svchost.exe):

```
0x82256da0:services.exe                                688
```

```
.... 0x822ea020:svchost.exe                  1040    688     83   1515 2010-02-26 03:34:07 UTC+0000
.... 0x81de55f0:svchost.exe                  1244    688     19    239 2010-02-26 03:34:08 UTC+0000
.... 0x822e1da0:svchost.exe                   948    688     10    276 2010-02-26 03:34:07 UTC+0000
.... 0x81dea020:svchost.exe                  1100    688      6     96 2010-02-26 03:34:07 UTC+0000
.... 0x82209640:svchost.exe                  1384    688      9    101 2010-02-27 20:12:36 UTC+0000
.... 0x82266870:svchost.exe                   880    688     28    340 2010-02-26 03:34:07 UTC+0000
```

Using the malfind plugin, we can look for injected code:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 malfind | Select-String -Pattern "Process:"`

In the output, we can see many results, which is extremely suspicious:

```
Process: System Pid: 4 Address: 0x40000
Process: System Pid: 4 Address: 0x170000
Process: System Pid: 4 Address: 0x190000
Process: csrss.exe Pid: 612 Address: 0x7f6f0000
Process: winlogon.exe Pid: 644 Address: 0xa10000
Process: winlogon.exe Pid: 644 Address: 0x24990000
Process: winlogon.exe Pid: 644 Address: 0x42e60000
Process: winlogon.exe Pid: 644 Address: 0x26200000
Process: winlogon.exe Pid: 644 Address: 0x7a330000
Process: winlogon.exe Pid: 644 Address: 0x7fc00000
Process: services.exe Pid: 688 Address: 0x750000
Process: lsass.exe Pid: 700 Address: 0xa10000
Process: vmacthlp.exe Pid: 852 Address: 0x640000
Process: svchost.exe Pid: 880 Address: 0x720000
Process: svchost.exe Pid: 948 Address: 0x850000
Process: svchost.exe Pid: 1040 Address: 0x20d0000
Process: svchost.exe Pid: 1100 Address: 0x870000
Process: svchost.exe Pid: 1244 Address: 0xa30000
Process: spoolsv.exe Pid: 1460 Address: 0x920000
Process: vmtoolsd.exe Pid: 1628 Address: 0x15b0000
Process: VMUpgradeHelper Pid: 1836 Address: 0xa30000
Process: alg.exe Pid: 2024 Address: 0x6b0000
Process: explorer.exe Pid: 1756 Address: 0x1830000
Process: explorer.exe Pid: 1756 Address: 0xac0000
Process: VMwareTray.exe Pid: 1108 Address: 0xd70000
Process: VMwareUser.exe Pid: 1116 Address: 0x1630000
Process: wscntfy.exe Pid: 1132 Address: 0x800000
Process: msiexec.exe Pid: 244 Address: 0x890000
Process: wuauclt.exe Pid: 440 Address: 0x1000000
Process: wuauclt.exe Pid: 232 Address: 0x12d0000
Process: firefox.exe Pid: 888 Address: 0x1e80000
Process: AcroRd32.exe Pid: 1752 Address: 0x30000
Process: svchost.exe Pid: 1384 Address: 0x80000
```

Each process contains an injected PE file, as identified through the PE file header:

```
Process: svchost.exe Pid: 1384 Address: 0x80000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 29, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00080000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00080010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00080020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00080030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00   ................
```

Let's run the strings command against the memory dump, and grep for .php and http:

- `strings Bob.vmem | grep -F '.php' | grep '^http:'`

Here we can find a php page hosted on a public IP:

```
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
```

Answer: http://193.104.22.71/~produkt/9j856f_4m9y8urb.php

Extract files from the initial process. One file has an MD5 hash ending with "528afe08e437765cc". When was this file first submitted for analysis on VirusTotal?

Let's start by dumping the memory for AcroRd32.exe (PID 1752) using the memdump plugin:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 memdump -p 1752 -D .`

We can then use a file carving tool called foremost to extract files from the memory dump:

- `foremost -i 1752.dmp -o banking_troubles/`

I started by generating the MD5 hashes for all extracted pdf files, grepping for the ending string given in the question:

- `md5sum * | grep "528afe08e437765cc"`

```
f32aa81676c7391528afe08e437765cc  00601560.pdf
```

Answer: 2010-03-29 19:31:45

What was the PID of the process that loaded the file PDF.php?

If you run strings against the AcroRd32.exe (PID 1752) dump, we can see mentions of PDF.php:

- `strings 1752.dmp | grep "PDF.php"`

```
PDF.php (application/pdf Object) - Mozilla Firefox
OCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
che/PDF.php?st=
C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
PDF.php (application/pdf Object) - Mozilla Firefox
OCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
PDF.php
PDF.php
HTTP:http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
    =http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0)
PDF.php
PDF.php
1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
PDF.php
DOCUME~1\ADMINI~1\LOCALS~1\Temp\plugtmp\PDF.php
search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
```

Answer: 1752

The JS includes a function meant to hide the call to function eval(). Provide the name of that function.

Let's start by using peepdf to analyse "00601560.pdf". Peepdf is a Python tool used to examine PDF files:

- `peepdf 00601560.pdf`

```
File: 00601560.pdf
MD5: f32aa81676c7391528afe08e437765cc
SHA1: 6045554853a61681d7264260cdd1072bbdc113ac
SHA256: 0bd1a5731f70dbf77c03e09822e2b3d68a4f25064baff7371f281410114fc936
Size: 607083 bytes
Version: 1.3
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 5
Streams: 1
URIs: 0
Comments: 0
Errors: 0

Version 0:
        Catalog: No
        Info: No
        Objects (5): [11, 112, 787, 1054, 1847]
        Streams (1): [1054]
                Encoded (1): [1054]
        Objects with JS code (1): [1054]
        Suspicious elements:
                /AA (1): [1847]
                /JS (1): [11]
                /JavaScript (1): [11]
```

Here we can see that it detects suspicious JavaScript code. We can use pdf-parser.py to extract the JavaScript:

- `pdf-parser.py --raw -o 1054 -f 00601560.pdf -d mal.js`
    - This command extracts the object "1054" which contains JavaScript code as identified by peepdf, saving the output to mal.js.

Unfortunately, my pdf-parser was just not working so refer to other writeups.

Answer: HNQYxrFW

The payload includes 3 shellcodes for different versions of Acrobat reader. Provide the function name that corresponds to Acrobat v9.

Answer: XiIHG

Process winlogon.exe hosted a popular malware that was first submitted for analysis at VirusTotal on 2010-03-29 11:34:01. Provide the MD5 hash of that malware.

TLDR: Using malfind, dump the memory region containing the PE file header, hash the output and submit it to VirusTotal.

If you recall earlier, we found a PE file injected into winlogon.exe (PID 644):

```
Process: winlogon.exe Pid: 644 Address: 0xa10000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 29, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x00a10000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00a10010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00a10020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00a10030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00   ................
```

Using the malfind plugin, we can dump the injected memory regions:

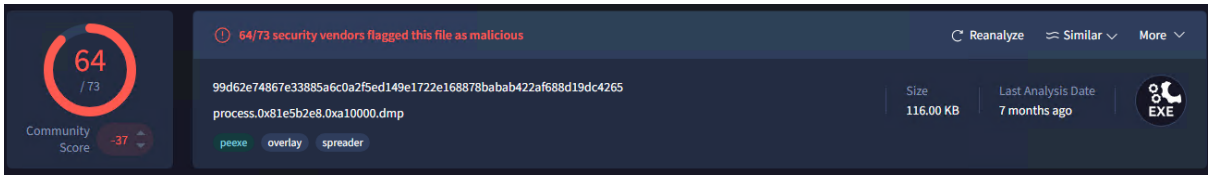
- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 malfind -p 644 --dump-dir=.`

We want to focus on the address 0xa10000 as this contained the PE file header. We can then use the Get-FileHash cmdlet to generate the MD5 hash for this file:

- `Get-FileHash -Algorithm MD5 -Path process.0x81e5b2e8.0xa10000.dmp`

```
Algorithm       Hash
---------       ----
MD5             066F61950BDD31DB4BA95959B86B5269
```

Upon submitting this hash to VirusTotal, it receives a significant number of detections:



Answer: 066F61950BDD31DB4BA95959B86B5269

What is the name of the malicious executable referenced in registry hive '\WINDOWS\system32\config\software', and is variant of ZeuS trojan?

After reading this report, you can determine that the Zeus trojan would modify the Winlogon key for persistence:

> ZeuS also makes registry changes to ensure that it starts up with Administrator privileges:
>
> *HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon*

We can use the printkey plugin to dump this key:

- `.\volatility_2.6_win64_standalone.exe -f "Bob.vmem" --profile=WinXPSP2x86 -g 0x80544ce0 printkey -K "Microsoft\Windows NT\CurrentVersion\Winlogon"`

We can see that “sdra64.exe” was added to the UserInit subkey for persistence.

```
Registry: \Device\HarddiskVolume1\WINDOWS\system32\config\software
Key name: Winlogon (S)
Last updated: 2010-02-27 20:12:34 UTC+0000

Subkeys:
  (S) GPExtensions
  (S) Notify
  (S) SpecialAccounts
  (V) Credentials

Values:
REG_DWORD     AutoRestartShell : (S) 1
REG_SZ        DefaultDomainName : (S) BOB-DCADFEDC55C
REG_SZ        DefaultUserName : (S) Administrator
REG_SZ        LegalNoticeCaption : (S)
REG_SZ        LegalNoticeText : (S)
REG_SZ        PowerdownAfterShutdown : (S) 0
REG_SZ        ReportBootOk    : (S) 1
REG_SZ        Shell           : (S) Explorer.exe
REG_SZ        ShutdownWithoutLogon : (S) 0
REG_SZ        System          : (S)
REG_SZ        Userinit        : (S) C:\WINDOWS\system32\userinit.exe,C:\WINDOWS\system32\sdra64.exe,
REG_SZ        VmApplet        : (S) rundll32 shell32,Control_RunDLL "sysdm.cpl"
REG_DWORD     SfcQuota        : (S) 4294967295
```

Answer: sdra64.exe

The shellcode for Acrobat v7 downloads a file named e.exe from a specific URL. Provide the URL.

Using the following command:

- `strings Bob.vmem | grep "^http:"`

We can hunt for all URLs within the memory dump. I came across the following interesting URL:

```
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2
```

Answer: http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2

The shellcode for Acrobat v8 exploits a specific vulnerability. Provide the CVE number.

Unfortunately, due to pdf-parser.py not working for me, I can't analyse the malicious JavaScript using jsunpackn.

Answer: CVE-2008-2992