은 상

박종운 ((주) 시큐브 정보보호기술연구소)
이준호 ((주) 시큐브 정보보호기술연구소)
최홍민 ((주) 시큐브 정보보호기술연구소)

요약 보고서

□ 침해사고 분석 1 요약

침해사고가 발생한 와우 리눅스 7.3 운영체제를 사용하는 시스템은 외부에서 접속 가능한 웹 및 SSH 서비스를 제공하였다. 따라서 침해사고는 시스템 설정의 오류 및 두 서비스의 취약성을 이용해 발생했을 것이라는 가정을 토대로 다음과 같은 분석을 수행하였다.

- 시스템 및 웹 서버 감사로그 조사
- 시스템 설정 파일 변경 내역 조사
- SetUID 설정 프로그램의 조사
- 숨김 속성을 가진 파일의 조사

이와 같은 조사를 토대로 우리는 Linux Rootkit 유형의 백도어가 설치된 것을 확인할 수 있었고, 이를 토대로 Rootkit을 구성하는 각각의 파일에 대한 분석을 수행하였다. 분석 결과 침해사고는 SSH CRC32 취약성을 이용한 공격에 의해 일어났고, 다음과 같은 절차로 수행되었음을 확인할 수 있었다.

- 공격자는 원격에서 SSH 취약성을 이용한 공격 코드 “x3”를 수행하여, Root 권한을 획득하고 Rootkit을 /usr/lib/librk/rootkit/ 디렉터리에 복사.
- install 스크립트의 실행
- 기존 시스템 명령어 ls, ps, netstat 명령어의 대체
- bshell, srload, cgihack.cgi 백도어 프로그램의 설치
- bshell, srload 백도어의 부팅 시 자동 실행 등록
- cgihack.cgi 프로그램에 SetUID 설정
- bshell, srload 백도어 프로그램의 실행
- hideit 프로그램을 통한 Rootkit 흔적의 제거
- logclean 프로그램을 통한 감사로그 흔적의 제거

□ 악성코드 분석 2 요약

SISWorm.exe 웜의 분석을 수행하기 위해, 우리는 직접 웜을 다음과 같은 절차로 수행하여 시스템의 변경 증상을 분석하였다.

- 감염 후 시스템 레지스트리 변경 조사
- 감염 후 시스템에 설치된 파일 조사
- 감염 후 프로세스의 네트워크 사용 변경 조사
- 감염 후 네트워크 트래픽 변경 조사
- 기타사항

이와 같은 조사를 토대로 우리는 SISWorm.exe 웜이 LSASS Buffer Overrun 취약성을 이용한 공격을 토대로 전파되는 것을 확인할 수 있었으나, 대량의 메일을 발송하는 것은 확인할 수 없었다.

문제 1. 침해사고 분석

□ 개요

침해사고가 발생한 시스템은 Wow Linux 7.3 운영체제를 사용하며, Apache 웹 서버를 통한 서비스를 제공한다. 또한 원격 시스템 관리를 위해 SSH를 사용한다. 따라서 발생된 침해사고는 세 가지 환경(Wow Linux 7.3, 웹 서비스, SSH 서비스)과 연관된 취약성 및 설정의 오류를 통해 일어난 것으로 판단하고, 여기에 초점을 맞추어 침해사고 분석을 수행하였다.

□ 환경·자산 분석

우리는 시스템에 존재하는 침해사고의 흔적을 조사하기 위해 다음과 같은 5 단계의 환경 및 자산 분석 과정을 수행하였다.

▸ 1단계: 시스템 자원 조사

이는 피해가 발생한 시스템에 어떤 자원이 존재하며, 이를 통해 알려진 취약성 및 위협원은 어떤 것이 존재하는지를 파악한다. 이를 통해 침해사고가 발생한 방법을 유추하는데 기초적인 정보를 제공한다.

Wow Linux	버전	7.3
	알려진 취약성	▪ RootKit 유형의 백도어 ▪ 시스템 설정의 오류
Apache	버전	
	알려진 취약성	▪ 취약성이 있는 CGI 스크립트 공격
SSH	버전	
	알려진 취약성	▪ 소스코드 취약성을 통한 BOF 공격

▸ 2단계 - 감사로그 파일 조사

이는 시스템 및 특정 응용 프로그램에서 제공하는 감사로그 정보의 분석을 통해 사용자의 활동 내역, 이벤트 처리 결과 등 시스템 및 응용 프로그램 사용의 비정상적인 내역에 대한 정보를 제공한다. 이를 위해 우리는 /mnt/etc/syslog.conf 파일에 설정된 감사로그 내역을 통해 시스템에서 기록하는 정보와 Apache 웹 서버에서 설정된 감사로그 내역을 통해 웹 서비스를 통해 기록하는 정보를 조사하였다.

대상	/mnt/var/log/messages
역할	*.info;mail.none;authpriv.none;cron.none
감사로그	fig1 - messages 참조
분석결과	fig1에서 흰색으로 반전된 레코드를 확인해 보면 네트워크 인터페이스가 promiscuous mode로 설정된 것을 확인할 수 있다. 이는 네트워크를 통과하는 모든 트래픽을 감시하기 위해서 사용되고, Root 권한을 이용해야 하므로 해당 시스템에서는 비정상적인 내역으로 보인다.

대상	/mnt/var/log/secure
역할	authpriv.*
감사로그	fig2 - secure 참조
분석결과	정상적인 사용 내역으로 보인다.

대상	/mnt/var/log/maillog
역할	mail.^
감사로그	fig3 - maillog 참조
분석결과	정상적인 사용 내역으로 보인다.

대상	/mnt/var/log/spooler
역할	uucp, news.crit
감사로그	fig4 - spooler 참조
분석결과	정상적인 사용 내역으로 보인다.

대상	/mnt/var/log/wtmp
역할	사용자 접속 시 인증 관련 감사로그
감사로그	fig5 - wtmp 참조
분석결과	정상적인 사용 내역으로 보인다.

대상	/mnt/var/log/httpd/error_log
역할	웹 서버의 에러 관련 감사로그
감사로그	fig6 - error_log 참조
분석결과	정상적인 사용 내역으로 보인다.

대상	/mnt/var/log/httpd/access_log
역할	웹 서버 접근 관련 감사로그
감사로그	fig7 - access_log 참조
분석결과	정상적인 사용 내역으로 보인다.

▸ 3단계: 시스템 설정 파일 조사

이는 시스템 운영 및 특정 서비스 운영을 위해 필요한 설정이 관리자가 의도한 데로 정확히 설정되어 있는지에 대한 정보를 제공한다. 일반적으로 침해사고가 발생한 경우, 공격자는 백도어를 자동으로 수행하기 위해 시스템 설정 파일을 임의로 조작하거나, 관리자가 알지 못하는 파일을 추가한다.

대상	/mnt/etc/rc.d/xinetd.d
역할	시스템 부팅 시 초기에 자동으로 실행되는 프로그램 등록
감사로그	fig8 - xined.d 참조
분석결과	시스템에서 사용되지 않는 telnet 서비스가 enable 되어 있으나, 해킹에 이용된 흔적을 찾을 수 없어, 관리자의 설정 오류로 보임.

대상	/mnt/etc/rc.d/rc.local

역할	시스템 로그인 전에 실행되어야 하는 프로그램 등록
감사로그	fig9 - rc.local 참조
분석결과	정상적인 설정 내용으로 보인다.

대상	/mnt/etc/rc.d/rc.sysinit
역할	시스템 부팅 시 초기화를 위해 필요한 프로그램 등록
감사로그	fig10 - rc.sysinit 참조
분석결과	fig10에서 붉은색 선으로 구분된 파일들은 시스템 운영에 필요한 파일로 보이지 않으며, 관리자가 알 수 없는 제 3의 사용자에 의해서 추가된 파일로 보인다. 이는 외부의 공격자가 백도어 형태의 프로그램을 부팅 시 자동을 실행하기 위해 등록한 것으로 보인다.

대상	/mnt/etc/passwd
역할	시스템 사용자 계정에 대한 정보
감사로그	fig11 - passwd 참조
분석결과	정상적인 설정 내용으로 보인다.

대상	/mnt/etc/shadow
역할	시스템 사용자 계정에 대한 정보
감사로그	fig12 - shadow 참조
분석결과	정상적인 설정 내용으로 보인다.

대상	/mnt/etc/services
역할	시스템에서 제공하는 서비스 관련 설정
감사로그	fig13 - services 참조
분석결과	정상적인 설정 내용으로 보인다.

대상	/mnt/etc/skel
역할	
감사로그	fig14 - skel 참조
분석결과	정상적인 설정 내용으로 보인다.

▸ 4단계: SetUID 설정 파일 조사

SetUID 설정을 가진 프로그램은 Root 권한으로 실행될 수 있는 특권을 가지므로 침해사고에 많이 이용되는 것이다. 따라서 시스템에서 사용하는 또는 관리자가 임의로 설정한 SetUID 설정 프로그램을 제외한 나머지 파일들은 비정상적인 행위에 사용될 수 있는 여지를 제공한다.

방법	find /mnt -user 0 -perm -4000 -print
실행결과	fig15 - setuid 참조
분석결과	fig15에서 흰색으로 반전된 파일은 일반적으로 웹 서비스를 위해 사용되지 않으며, 관리자가 확인 할 수 없는 용도를 가진 것으로 보인다. 따라서 이는 침해사고로 발생한 백도어 혹은 외부에서 악용될 소지가 있는 CGI 파일로 보인다.

▸ 5단계: 숨김(Hidden) 속성을 가진 파일 조사

일반적으로 시스템 혹은 특정 프로그램의 안전한 운영을 위해 숨김 속성을 가진 파일들이 존재한다. 그러나 시스템에서 사용하지 않거나 관리자가 임의로 설정한 숨김 속성을 가진 파일이 아니라면, 외부의 제 3의 사용자에 의해 관리자에게 숨기기 위해서 설정된 것이라는 정보를 제공한다.

방법	find /mnt -name ".*" -print
실행결과	fig16 - hidden 참조
분석결과	fig16에서 붉은선으로 구분된 파일들은 시스템 운영에 사용되지 않으며, 관리자가 확인 할 수 없는 용도를 가진 것으로 보인다. 따라서 이는 제 3의 백도어 혹은 외부에서 악용하기 위해 숨긴 파일로 보인다.

□ 침해사고 탐지·분석

『환경·자산 분석』 단계에서 찾은 4가지의 침해사고 흔적을 정리해 보면 다음과 같다.

1	네트워크 인터페이스를 통해 패킷 수집을 수행하는 프로세스의 존재 [증거] /mnt/var/log/messages 감사로그에 남은 promiscuous mode를 실행한 srload 프로세스의 이름 확인
2	시스템 부팅 시 자동으로 실행되도록 설정된 프로세스의 존재 [증거] /mnt/etc/rc.d/rc.sysinit 설정 파일에 등록된 /sbin/bshell, /sbin/srload 프로세스 확인
3	관리자 및 감시자의 눈을 속이기 위한 숨김 속성 파일의 존재 [증거] 숨김 속성을 가진 파일 이름의 검색을 통해 찾은 /mnt/dev/ptyxx/.file, /mnt/dev/ptyxx/.proc, /mnt/dev/ptyxx/.addr, /mnt/usr/lib/..., /mnt/usr/lib/librk/rootkit/.snifflogsk, /mnt/usr/share/man/man.1/..1.gz 파일 확인
4	알 수 없는 SetUID 설정을 가진 CGI 프로그램의 존재 [증거] /mnt/var/www/cgi-bin/cgihack.cgi 확인

우리는 이상의 4가지 흔적의 역추적 과정을 거치면서 /mnt/usr/lib/librk/ 디렉터리에 설치된 Rootkit의 정보를 확인할 수 있었다.

▸ messages 감사로그에 남은 promiscuous mode를 실행한 srload 프로세스 이름과 rc.sysinit 설정 파일에 등록된 /sbin/srload 등록 정보를 통해 /mnt/sbin/srload 파일 위치 파악 및 바이너리 데이터 분석

1단계	방법	vi /mnt/sbin/srload 명령어 실행을 통해 바이너리 데이터 분석 시도
	실행 결과	fig17 - srload 참조
	분석 결과	fig17에서 흰색으로 반전된 ".snifflogsk" 파일이 숨김 속성으로 되어 있고 srload 프로그램에서 사용하는 감사로그 파일로 유추
2단계	방법	find /mnt -name ".snifflogsk" -print 명령어 실행을 통해 해당 시

		스템에 관련 파일이 존재하는지 분석 시도
	실행 결과	fig18 - .snifflogsk 참조
	분석 결과	fig18에서 흰색으로 반전된 부분에서 확인할 수 있듯이, /mnt/usr/lib/librk/rootkit/ 디렉터리 밑에 ".snifflogsk" 파일이 존재함을 확인.
3단계	방법	ls -alF /mnt/usr/lib/librk/rootkit 명령어 실행을 통해
	실행 결과	fig19 - rootkit 참조
	분석 결과	fig19에서 확인할 수 있듯이 잠정적으로 침해사고가 발생한 rootkit 도구의 12개 구성요소 확인

▸ rc.sysinit 설정 파일에 등록된 /sbin/bshell 정보를 통해 파악하여 /mnt/sbin/bshell 파일의 바이너리 데이터 분석

1단계	방법	vi /mnt/sbin/bshell 명령어 실행을 통해 바이너리 데이터 분석 시도
	실행 결과	fig20 - bshell 참조
	분석 결과	fig20에서 흰색으로 반전된 "/bin/sh"에서 확인할 수 있듯이 bshell 프로그램은 특정 쉘을 Root의 권한으로 실행할 것임을 유추

▸ SetUID 설정을 가진 cgihack.cgi 파일의 바이너리 데이터 분석

1단계	방법	vi /mnt/var/www/cgi-bin/cgihack.cgi 명령어 실행을 통해 바이너리 데이터 분석 시도
	실행 결과	fig21 - cgihack.cgi 참조
	분석 결과	fig21에서 붉은색으로 구분된 세 부분은 해당 CGI 프로그램을 역할을 유추하게 해 준다. 해당 CGI는 원격에서의 접속을 통해 아래 3가지 역할을 수행하는 것으로 유추 ① 웹 서비스 감사로그 변조 기능 ② 원격에서 임의의 명령어 수행 기능 ③ 원격에서 Root 권한을 가진 사용자 계정 생성 기능

위와 같은 단계를 거쳐 분석된 침해사고의 흔적은 백도어 형태의 Rootkit 존재를 확실히 증명해 준다고 생각된다. 그러나 우리는 Rootkit을 통한 침해사고 가상 시나리오를 완성하기 위하여 다음과 같이 추가적으로 /mnt/usr/lib/librk/rootkit/ 디렉터리의 구성 요소를 분석하였다.

▸ install 파일 분석

1단계	방법	vi /mnt/usr/lib/librk/rootkit/install 명령어 실행을 통해 스크립트 데이터 분석 시도
	실행 결과	fig22 - install 참조
	분석	fig22에서 색깔별로 구분된 부분은 install 스크립트가 다음과 같은 기

	결과	능을 수행하는 것을 설명한다. ① 기존에 시스템에 존재하던 ls, ps, netstat 명령어 대체 (이는 cmp 명령어를 통해 확인 가능) ② bshell, srload, cgihack.cgi 백도어 프로그램의 설치 ③ bshell, srload 프로그램의 부팅 시 자동 실행 설정 ④ cgihack.cgi 프로그램에 SetUID 설정 ⑤ bshell, srload 백도어 프로그램 실행

▸ bindshell(or /sbin/bshell) 파일의 바이너리 데이터 분석

1단계	방법	grep "socket" /mnt/usr/lib/librk/rootkit/bindshell 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig23 - bindshell 참조
	분석 결과	fig23에서 색깔로 구분된 bindshell 매칭 결과는 실제 socket 함수가 사용되고 있음을 유추
2단계	방법	grep "bind" /mnt/usr/lib/librk/rootkit/bindshell 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig23 - bindshell 참조
	분석 결과	fig23에서 색깔로 구분된 bindshell 매칭 결과는 실제 bind 함수가 사용되고 있음을 유추
3단계	방법	grep "listen" /mnt/usr/lib/librk/rootkit/bindshell 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig23 - bindshell 참조
	분석 결과	fig23에서 색깔로 구분된 bindshell 매칭 결과는 실제 listen 함수가 사용되고 있음을 유추
4단계	방법	grep "accept" /mnt/usr/lib/librk/rootkit/bindshell 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig23 - bindshell 참조
	분석 결과	fig23에서 색깔로 구분된 bindshell 매칭 결과는 실제 accept 함수가 사용되고 있음을 유추
종합분석 결과		bindshell 프로그램은 일반적으로 서버 프로그램에서 사용하는 socket, bind, listen, accpet 라이브러리를 사용하며, /bin/sh 쉘을 실행하는 것으로 보아, 외부의 접속을 기다리며 원격에서 쉘 명령어를 수행할 수 있도록 해주는 백도어의 역할을 해주는 것으로 보임

▸ sniffer(or /sbin/srload) 파일의 바이너리 데이터 분석

1단계	방법	grep "socket" /mnt/usr/lib/librk/rootkit/sniffer 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig24 - sniffer 참조

	분석 결과	fig24에서 색깔로 구분된 sniffer 매칭 결과는 실제 socket 함수가 사용되고 있음을 유추
2단계	방법	grep "bind" /mnt/usr/lib/librk/rootkit/sniffer and grep "listen" /mnt/usr/lib/librk/rootkit/sniffer and grep "connect" /mnt/usr/lib/librk/rootkit/sniffer 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig24 - sniffer 참조
	분석 결과	fig24에서 색깔로 구분된 sniffer 매칭 결과는 실제 bind and listen and connect 함수가 사용되지 않음을 유추
3단계	방법	grep "read" /mnt/usr/lib/librk/rootkit/sniffer and grep "write" /mnt/usr/lib/librk/rootkit/sniffer 명령어 실행을 통해 해당 함수의 존재 분석 시도
	실행 결과	fig24 - sniffer 참조
	분석 결과	fig24에서 색깔로 구분된 sniffer 매칭 결과는 실제 read 함수는 사용되고 있음을, write 함수는 사용되지 않음을 유추
4단계	방법	grep "promiscuous" /mnt/usr/lib/librk/rootkit/sniffer 명령어 실행을 통해 promiscuous mode 사용 여부 분석 시도
	실행 결과	fig24 - sniffer 참조
	분석 결과	fig24에서 색깔로 구분된 sniffer 매칭 결과는 실제 promiscuous 모드가 사용되고 있음을 유추
종합분석 결과		sniffer 프로그램은 socket, read 기능을 수행하며 listen, write 기능을 수행하지 않는 특성을 보이고, promiscuous mode를 사용한다. 이는 일반적인 네트워크 트래픽 모니터링 도구와 동일한 기능을 역할을 해주는 것으로 보임.

▸ hideit 파일 분석

1단계	방법	vi /mnt/usr/lib/librk/rootkit/hideit 명령어 실행을 통해 스크립트 데이터 분석 시도
	실행 결과	fig25 - hideit 참조
	분석 결과	fig25에서 색깔별로 구분된 부분은 hideit 스크립트가 전형적인 Rootkit의 행동처럼 백도어의 구성파일, 프로세스, 네트워크 관련 정보를 숨기는 역할을 수행함을 보여준다. ① sniffer, bindshell, rootkit, .snifflogsk의 백도어 구성 파일 정보를 /dev/ptyxx/.file 에 숨김으로 설정 ② bshell, srload, cgihack.cgi의 프로세스 파일 정보를 /dev/ptyxx/.proc 에 숨김으로 설정 ③ 백도어 프로그램이 사용할 것으로 예상되는 특정 IP(192.168.131.136)와 특정 포트(2222) 정보를 /dev/ptyxx/.addr 에 숨김으로 설정

		④ 히스토리(History) 정보 숨김

▸ logclean 파일의 바이너리 데이터 분석

1단계	방법	vi /mnt/usr/lib/librk/rootkit/logclean 명령어 실행을 통해 스크립트 데이터 분석 시도
	실행 결과	fig26 - logclean 참조
	분석 결과	fig26에서 흰색으로 반전된 내용은 시스템 감사로그 관련 파일 위치 정보를 나타낸다. 이는 logclean이 침해사고 발생 후, 시스템 감사로그 위변조를 위한 역할을 수행한다고 보임. 연관된 감사로그 파일 목록은 다음과 같다. ① /var/log/http/error_log ② /var/log/httpd/access_log ③ /var/log/mail ④ /var/log/warn ⑤ /var/log/maillog ⑥ /var/log/xferlog ⑦ /var/log/secure ⑧ /var/log/messages ⑨ /var/run/utmp ⑩ /var/log/wtmp ⑪ /var/log/lastlog

▸ x3 파일의 바이너리 데이터 분석

1단계	방법	vi /mnt/usr/lib/librk/rootkit/x3 명령어 실행을 통해 바이너리 데이터 분석 시도
	실행 결과	fig27 - x3 참조
	분석 결과	fig27에서 흰색으로 반전된 부분에서 다음의 내용을 확인할 수 있듯이,"sshd-exploit -t (options) host [port]", 이는 해당 시스템이 사용하는 SSH 서비스의 취약성을 이용한 원격 공격으로 보임. 또한 대상 정보를 찾기 위해 target 파일을 이용하는 것으로 보임. 또한 두 번째 흰색으로 반전된 부분, "deattack exploit By Dvorak with Code from teso"은 SSH CRC32 Overflow 취약성을 이용하는 알려진 공격코드와 동일하다고 판단. 즉 해당 시스템은 이 취약성을 이용하여 공격 당한 것으로 보임.

▸ target 파일 분석

1단계	방법	vi /mnt/usr/lib/librk/rootkit/target 명령어 실행을 통해 데이터 분석 시도
	실행 결과	fig28 - target 참조

	분석 결과	fig28에서 확인할 수 있듯이, target 파일은 SSH 버전별 공격에 필요한 정보를 가지고 있는 것으로 보임. 공격 범위는 다음과 같다. ① SSH-1.5-1.2.26 ~ SSH-1.5-1.3.10 ② SSH-1.5-OpenSSH-1.2 ~ SSH-1.5-OpenSSH-2.1 ③ SSH-1.99-2.0.11 ~ SSH-1.99-3.0.1 ④ SSH-1.99-OpenSSH-2.1 ~ SSH-2.99-OpenSSH_2.2.0p1 ⑤ SSH-2.0-2.3.0 ~ SSH-2.0-3.0.1

이와 같은 Rootkit을 구성하는 각 요소의 분석 결과를 토대로 우리는 다음과 같은 침해사고 발생 시나리오가 발생했음을 요약할 수 있다.

1. 공격자는 원격에서 SSH 취약성을 이용한 공격 코드 “x3”를 수행하여, Root 권한을 획득하고 Rootkit을 /usr/lib/librk/rootkit/ 디렉터리에 복사.
2. install 스크립트의 실행
3. 기존 시스템 명령어 ls, ps, netstat 명령어의 대체
4. bshell, srload, cgihack.cgi 백도어 프로그램의 설치
5. bshell, srload 백도어의 부팅 시 자동 실행 등록
6. cgihack.cgi 프로그램에 SetUID 설정
7. bshell, srload 백도어 프로그램의 실행
8. hideit 프로그램을 통한 Rootkit 흔적의 제거
9. logclean 프로그램을 통한 감사로그 흔적의 제거

이후부터 공격자는 bshell을 통한 원격 쉘 명령어 실행을 시도하거나, 웹 서버 접속을 통해 cgihack.cgi 프로그램을 실행하여 임의의 명령어를 수행할 수 있다. 또한 제 3의 피해 시스템을 공격하기 위해 다시 “x3“ 공격을 시도할 수 있다.

□ 침입사고 대응·결론

우리는 위에서 분석한 침해사고의 분석 결과를 토대로 시스템을 원상태로 복구하고 침해사고의 원인을 제거하기 위해 다음과 같은 절차를 수행한다.

- Rootkit 백도어의 제거
- Rootkit 백도어 설정 파일의 제거
- 시스템 명령어 및 설정 파일의 복구
- SSH CRC32 취약성 패치

문제 2. 악성코드(웜) 분석

□ 개요

침해사고의 원인으로 찾아낸 SISWorm.exe 웜은 사내 개인 PC에 감염되어 외부의 445번 포트로 접속 시도를 하는 트래픽을 생성한다. 또한 웜에 감염된 PC는 ISP에 설치된 메일 서버를 통해 스팸 메일을 보내는 것으로 확인됐다. 우리는 SISWorm.exe 웜에 의해 발생하는 이러한 현상을 확인하기 위해 직접 웜을 실행하여 어떤 현상이 발생하는지를 분석하였으며, 이를 토대로 웜의 전파 방법 및 스팸 메일의 발생 원인을 추론하였다.

□ 감염 증상 분석

우리는 SISWorm.exe 웜을 직접 실행하여 시스템에서 발생하는 아래와 같은 증상을 분석하였다.

▸ 감염 후 시스템 레지스트리 변경 조사

방법	ARM(Active Registry Monitory) 도구를 이용하여 SISWorm.exe 웜 실행 전의 레지스트리 정보를 미리 저장하고, SISWorm.exe 실행 후의 변경된 레지스트리 정보를 비교
실행결과	fig29 - arm 참조
분석결과	fig29에서 확인할 수 있듯이, 아래와 같이 정보는 일반적으로 웜이 레지스트리에 자신을 등록하여 부팅 시마다 자동으로 실행되도록 하는 것과 동일하다. HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Run winsystemm.exe= C:\Windows\System32\winsystemm.exe

▸ 감염 후 시스템에 설치된 파일 조사

방법	레지스트리에 등록된 "winsystemm.exe" 파일의 실제 존재를 검색하기 위해 윈도우 익스플로우의 파일검색 메뉴 이용
실행결과	fig30 - search 참조
분석결과	fig30에서 확인할 수 있듯이, 해당 웜이 사용하는 파일이 실제 시스템에 설치되었음을 검증할 수 있다.

▸ 감염 후 프로세스의 네트워크 사용 변경 조사

방법	FPort 도구를 사용해 SISWorm.exe 웜 실행 전과 실행 후, 프로세스의 네트워크 사용 상태 변화 비교
실행결과	fig31 - fport 참조
분석결과	fig31에서 확인할 수 있듯이, Explorer 프로세스가 특정 포트를 Listening 하고 , 웜이 설치 한 winsystemm 프로세스도 복수 개의 포트를 Listening 하고 있는 것을 알 수 있다. 이는 일반적으로 웜이 전파되는 방법인 특정 명령을 전송하기 위해 포트와 웜을 복제하기 위한 ftp 채널을 만드는 포트로 유추된다.

▸ 감염 후 네트워크 트래픽 변경 조사

방법	Ethereal 네트워크 트래픽 모니터링 도구를 사용하여 SISWorm.exe 웜 실행 전과 실행 후에 트래픽 변화 비교
실행결과	fig32 - ethereal 참조
분석결과	fig32에서 확인할 수 있듯이 웜이 실행되는 순간, microsoft-ds 서비스가 사용하는 445번 포트로의 트래픽이 발생하는 것을 확인할 수 있다. 이는 LSASS Buffer Overrun 취약성을 이용하는 공격으로 유추된다.

▸ 기타사항

SISWorm.exe 웜이 사용하는 트래픽의 기타 특성으로 445번 포트로의 접속 시도를 발생하는 부분이 6월 15일 까지로 제한되어 있었고, 또한 감염대상 IP를 “211.241.82.124” 하나로 제한한 것을 확인할 수 있었다. 대회 참가자들의 피해를 우려한 배려로 생각된다.

□ 감염 시나리오 추론

이와 같은 웜 실행 후 발견된 증상들을 토대로 웜이 감염되는 시나리오를 다음과 같이 요약할 수 있다. 우리는 실제 웜이 감염되어 전파되는 결과를 확인하지 못하여 실제 메일이 대량으로 발송되는 과정까지는 확인하지 못하였다.

▸ LSASS 취약성을 통한(TCP 445번 포트를 통한) 시스템 권한 획득
▸ 웜 감염을 위한 명령 수행
▸ 웜 파일 다운로드 및 실행

이 후부터는 다시 웜이 감염되는 동일한 과정을 반복한다.

□ 대응 방안

우리는 위에서 분석한 웜의 증상 및 감염 시나리오를 제거하기 위해 다음과 같은 절차를 수행한다.

- 웜 실행 후 변경된 레지스트리 제거
- 웜 실행 후 변경된 파일 목록 제거
- 웜 실행 후 실행된 프로세스 목록 종료
- LSASS(MS04-011) 취약성 패치