

기술문서 09. 09. 20. 작성

# 스니핑 동작원리 및 공격방법

작성자 : 한서대학교 H.I.S.L 동아리 전정수 silvarows@nate.com

2009. 09. 20.



**한서대학교 정보보호동아리**

Since 2007 Hanseo University Information Security Lab.

※ 본 보고서의 전부나 일부를 인용시 반드시 [자료: 한서대학교 정보보호동아리(H.I.S.L)]를 명시하여 주시기 바랍니다.

## ◎ 목 차

### 1. 개 요

### 2. 스니핑

- 1) 스니핑 정의
- 2) 스니핑 원리

### 3. 허브 환경에서 스니핑

### 4. 스위치 환경에서 스니핑 기법

- 1) Switch Jamming
- 2) ARP Redirect
- 3) ARP spoofing
- 4) ICMP Redirect

### 5. 스니핑 공격방법

- 1) TCP Dump
- 2) Dsniff
- 3) Sniffer Pro(윈도우용)

### 6. 맺음말

## ◎ 참고자료

## 1. 개 요

IT의 발달로 정보화 사회에서 생활하고 있는 우리는 많은 정보를 컴퓨터에 저장하고 활용한다. 또한 인터넷을 이용한 전자 상거래 및 금융 서비스를 제공 받고 있다. 다양한 서비스와 컴퓨터의 활용도가 높은 사회에서 이를 악용하는 사례가 발생하고 있다. 스니핑은 그 중에서 가장 간단하여 자료도 쉽게 구할수 있다. 그럼 스니핑에 대하여 지금부터 알아보도록 하겠다.

## 2. 스니핑

### 1) 스니핑 정의

스니퍼(sniffer)는 원래 Network Associate사의 등록상표였으나 현재는 PC나 kleenex처럼 일반적인 용어로 사용되고 있다. "sniff"라는 단어의 의미(냄새를 맡다, 코를 킁킁거리다)에서도 알 수 있듯이 스니퍼는 "컴퓨터 네트워크상에 흘러다니는 트래픽을 엿듣는 도청장치"라고 말할 수 있다. 그리고 "스니핑"이란 이러한 스니퍼를 이용하여 네트워크상의 데이터를 도청하는 행위를 말한다. 이러한 스니핑 공격은 웹호스팅, 인터넷데이터센터(IDC) 등과 같이 여러 업체가 같은 네트워크를 공유하는 환경에서는 매우 위협적인 공격이 될 수 있다. 하나의 시스템이 공격 당하게 되면 그 시스템을 이용하여 네트워크를 도청하게 되고, 다른 시스템의 User ID/Passwd를 알아내게 된다. 비록 스위칭 환경의 네트워크를 구축하여 스니핑을 어렵게 할 수는 있지만 이를 우회할 수 있는 많은 공격방법이 존재한다. 본 문서는 스위칭 환경에서의 스니핑 공격 기법과 그리고 이에 대한 대책을 설명한다.

### 2) 스니핑의 원리

LAN 상에서 개별 호스트를 구별하기 위한 방법으로 이더넷 인터페이스는 MAC(Media Access Control) 주소를 갖게 되며, 모든 이더넷 인터페이스의 MAC 주소는 서로 다른 값을 갖는다. 따라서 로컬 네트워크상에서 각 각의 호스트는 유일하게 구별될 수 있다. 다음은 이더넷(ethernet) 프레임의 포맷을 나타낸다.

destination MAC addr 6 byte	source MAC addr 6 byte	type 2	data 46-1500 byte	CRC 4
<표 1> 이더넷의 포맷(RFC 894)				

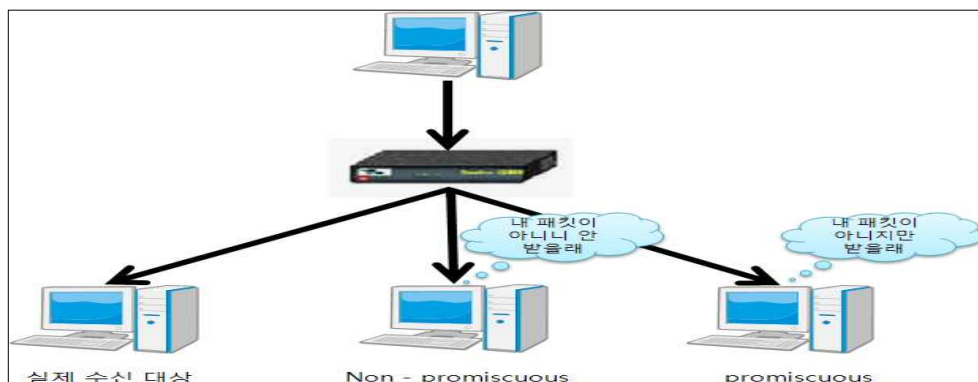
그리고 위의 이더넷 포맷은 type에 따라 다음과 같은 3가지 포맷으로 구성된다.

type 0800	IP datagram 46-1500	
type 0806	ARP request/reply 28	PAD 18
type 8035	RARP request/reply 28	PAD 18

이더넷은 로컬 네트워크내의 모든 호스트가 같은 선(wire)을 공유하도록 되어 있다. 따라서 같은 네트워크내의 컴퓨터는 다른 컴퓨터가 통신하는 모든 트래픽을 볼 수 있다. 하지만 이더넷을 지나는 모든 트래픽을 받아들이면 관계없는 트래픽까지 처리해야 하므로 효율적이지 못하고 네트워크의 성능도 저하될 수 있다. 그래서 이더넷 인터페이스(LAN 카드)는 자신의 MAC address를 갖지 않는 트래픽을 무시하는 필터링 기능을 가지고 있다. 이 필터링 기능은 자신의 MAC address를 가진 트래픽만을 보도록 한다. 또한 이더넷 인터페이스에서 모든 트래픽을 볼 수 있도록 하는 기능을 설정할 수도 있는데 이를 "promiscuous mode"라 한다. 스니퍼는 이더넷 인터페이스를 이러한 promiscuous mode"로 설정하여 로컬 네트워크를 지나는 모든 트래픽을 도청할 수 있게 된다.

### 3. 허브 환경에서의 스니핑

허브(Hub)는 기본적으로 들어온 패킷에 대해 ozt이 들어온 포트를 제외한 모든 포트에 대한 패킷을 보내는 리피터(Repeater)장비이다. 사실 여러 기업에서 허브를 사용하고 있고 여러 시스템이 그 허브에 연결되어 있다면 원하던 원치 않던 간에 계속하여 다른 사람의 패킷들을 받아보고 있었던 것이다. 물론 네트워크 다리이버, OS커널 등의 수준에서 MAC주소를 보아 자신이 아닌 다른 이들의 패킷은 버려지기 때문에 그것을 쉽게 느낄수 없었을 것이다. 하지만 여러 시스템 NIC를 promisuous모드로 동작하게 한다면 다른 이들의 패킷 또한 버리지 않고 받아볼 수 있다. 이제 스니핑 도구를 통해 해당 패킷을 저장하고 분석하기만 하면 된다.



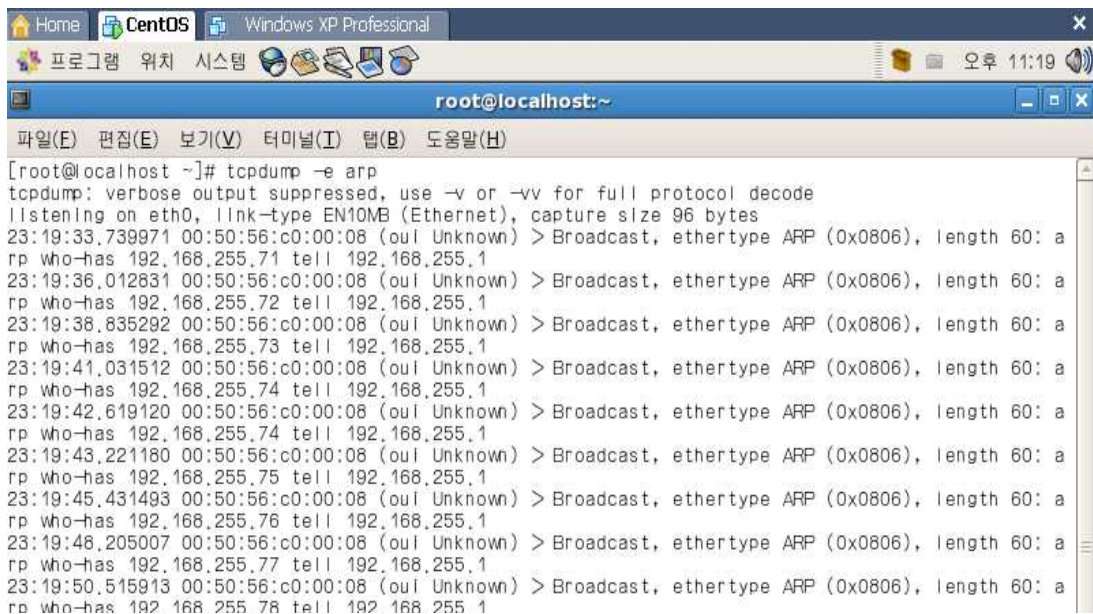
< 허브환경에서의 패킷 송신 >

## 4. 스위치 환경에서의 스니핑

일반적으로 앞서 설명한 스니핑을 방지하는 방법으로 스위칭 허브를 사용하게 된다. 스위칭 허브는 로컬 네트워크를 여러개의 세그먼트로 나누어 쓸 수 있도록 하는데, 각 세그먼트내의 트래픽은 다른 세그먼트로 전달되지 않는다. 따라서 스위칭 허브를 이용하여 업무별로 또는 독립적인 사이트별로 네트워크를 나누어 놓으면 다른 네트워크 세그먼트내의 네트워크 트래픽을 도청할 수 없게 된다. 하지만 Switch Jamming, ARP Redirect나 ICMP Redirect 등의 기법을 이용하여 다른 네트워크 세그먼트의 데이터를 스니핑 할 수 있는 방법도 있다.

### 1) Switch Jamming

많은 종류의 스위치들은 주소 테이블이 가득차게 되면(Full) 모든 네트워크 세그먼트로 트래픽을 브로드캐스팅하게 된다. 따라서 공격자는 위조된 MAC 주소를 지속적으로 네트워크에 흘림으로서 스위칭 허브의 주소 테이블을 오버플로우 시켜 다른 네트워크 세그먼트의 데이터를 스니핑 할 수 있게 된다. 이는 보안 원리의 하나인 "Fail close (시스템에 이상이 있을 경우 보안기능이 무력화되는 것을 방지하는 원리)"를 따르지 않기 때문에 발생한다. 스위치들은 사실상 보안보다는 기능과 성능 위주로 디자인 되어있다. 다음은 arp flooding 공격을 할 때 발생하는 임의의 arp 패킷을 tcpdump를 이용하여 잡은것이다. 공격자가 만들어낸 이러한 임의의 arp 패킷의 MAC 주소는 스위치의 주소 테이블을 오버플로우 시키게 된다.



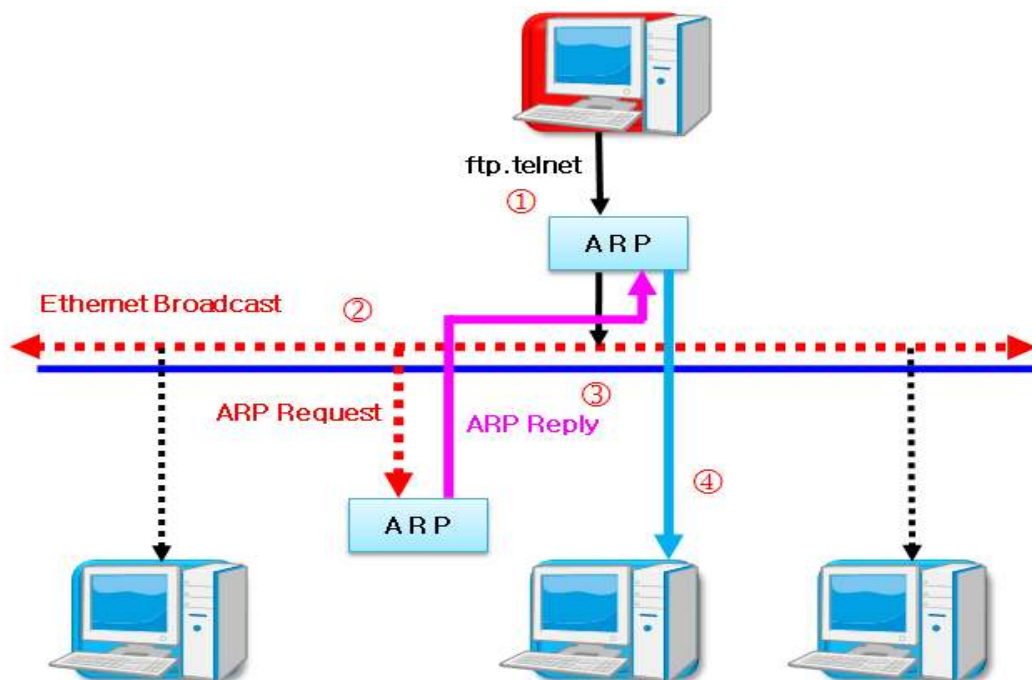
```
[root@localhost ~]# tcpdump -e arp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
23:19:33.739971 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.71 tell 192.168.255.1
23:19:36.012831 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.72 tell 192.168.255.1
23:19:36.835292 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.73 tell 192.168.255.1
23:19:41.031512 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.74 tell 192.168.255.1
23:19:42.619120 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.74 tell 192.168.255.1
23:19:43.221180 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.75 tell 192.168.255.1
23:19:45.431493 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.76 tell 192.168.255.1
23:19:48.205007 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.77 tell 192.168.255.1
23:19:50.515913 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.78 tell 192.168.255.1
```

## 2) ARP Redirect 공격

먼저 정상적인 ARP Protocol에 대하여 설명한다. IP 데이터그램에서 IP 주소는 32 bit 구조로 되어 있고 이더넷 주소(MAC 주소)는 48 bit의 크기를 갖는다. 다른 호스트로 ftp나 telnet 등과 같은 네트워크 연결을 하기 위해서는 상대방 호스트의 이더넷 주소를 알아야 한다. 즉, 사용자는 IP 주소를 이용하여 연결을 하지만 이더넷상에서는 이더넷 주소를 이용하게 된다. 이를 위하여 IP주소를 이더넷 주소로 변환시켜 주어야 하는데 이를 ARP(Address Resolution Protocol)라 한다. 그리고 그 역 과정을 RARP(Reverse Address resolution Protocol)라 한다. ARP를 이용하여 상대 호스트의 이더넷 주소를 알아내는 과정은 다음과 같다.

- ① 먼저 네트워크내의 모든 호스트에 "ARP Request"라고 불리는 이더넷 프레임을 보낸다. 연결하고자 하는 호스트의 IP 주소를 포함한 ARP Request는 이더넷상의 모든 다른 호스트들에게 "이 IP 주소를 사용하는 호스트는 나에게 하드웨어 주소(이더넷 주소)를 알려주세요"라는 의미를 갖는다.
- ② ARP Request를 받은 호스트 중 해당 IP를 사용하는 호스트는 자신의 하드웨어 주소(이더넷 주소)를 ARP Request를 보낸 호스트에게만 보내주게 되는데 이를 ARP Reply라고 한다.
- ③ 이후 두 호스트간의 통신(ftp, telnet 등)을 위하여 상대방의 이더넷 주소를 사용하게 되며, IP datagram을 송수신할 수 있게 된다.

다음 그림은 ARP Request와 ARP Reply의 과정을 보여주고 있다.





다음은 실제로 192.168.255.1 번 호스트에서 192.168.255.129번으로 ping을 했을 경우 나타나는 arp 트래픽이다. arp request/reply를 교환한 두 호스트는 상대방의 MAC주소를 각각의 arp cache에 저장하게 된다. 따라서 마지막 라인에서 172.16.2.26번 호스트가 15번 호스트로 echo reply를 보낼때는 arp request/reply 과정을 거치지 않아도 된다.

```
[root@localhost ~]# tcpdump -e host 192.168.255.129
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
23:28:00.873135 00:50:56:c0:00:08 (oui Unknown) > Broadcast, ethertype ARP (0x0806), length 60: a
rp who-has 192.168.255.129 tell 192.168.255.1
23:28:00.941908 00:0c:29:e6:53:87 (oui Unknown) > 00:50:56:c0:00:08 (oui Unknown), ethertype ARP
(0x0806), length 42: arp reply 192.168.255.129 is-at 00:0c:29:e6:53:87 (oui Unknown)
23:28:00.873165 00:50:56:c0:00:08 (oui Unknown) > 00:0c:29:e6:53:87 (oui Unknown), ethertype IPv4
(0x0800), length 74: 192.168.255.1 > 192.168.255.129: ICMP echo request, id 512, seq 2304, lengt
h 40
23:28:00.873333 00:0c:29:e6:53:87 (oui Unknown) > 00:50:56:c0:00:08 (oui Unknown), ethertype IPv4
(0x0800), length 74: 192.168.255.129 > 192.168.255.1: ICMP echo reply, id 512, seq 2304, length
40
```

"ARP Redirect" 공격은 위조된 arp reply를 보내는 방법을 사용한다. 즉 공격자 호스트가 "나의 MAC 주소가 라우터의 MAC 주소이다"라는 위조된 arp reply를 브로드캐스트로 네트워크에 주기적으로 보내어, 스위칭 네트워크상의 다른 모든 호스트들이 공격자 호스트를 라우터로 믿게끔한다. 결국 외부 네트워크와의 모든 트래픽은 공격자 호스트를 통하여 지나가게 되고 공격자는 스니퍼를 통하여 필요한 정보를 도청할 수 있게 된다.

※ ARP Protocol specification에 의하면 이미 cache에 저장하고 있는 IP에 대 의하면 이request를 받게되면 호스트는 하면 이request를 보낸 호스트의 MAC 주소를 cahe에 업데이트 하게 된다고 나와 있다. 그리고 이러 의cache의 업데이트 기능은 arp reply에도 적용되는 것으로 보이며, 위의 공격이 성공할 수 있는 요인이 된다.

하지만 시스템에 따라 다를 수도 있다.

이때 공격 호스트는 IP Forwarding 기능을 설정하여야 공격 호스트로 오는 모든 트래픽을 원래의 게이트웨이로 Forwarding 해줄 수 있다. 그렇지 않으면 외부로 나가는 모든 네트워크 연결이 끊어지게 된다. 다음은 "arpredirect"라는 공격 프로그램으로 공격했을 때 네트워크 상에 나타나는 arp 패킷을 tcpdump를 이용하여 잡은 모습이다. 공격이 끝날때는 원래의 arp 매핑을 복원하여 네트워크 연결이 끊어지지 않도록 하고 있다.

### 3) ARP Spoofing 공격

ARP redirect와 비슷한 공격 방법으로 다른 세그먼트에 존재하는 호스트간의 트래픽을 스니핑하고자 할 때 사용된다. 공격자는 자신의 MAC 주소를 스니핑하고자 하는 두 호스트의 MAC 주소로 위장하는 arp reply(또는 request) 패킷을 네트워크에 뿌린다. 즉 "나의(공격자의) MAC 주소가 스니핑하고자 하는 호스트의 MAC 주소이다"라는 arp reply를 각 각의 호스트에게 보내게 된다. 이러한 arp reply를 받은 두 호스트는 자신의 arp cache를 업데이트하게 되고, 두 호스트간에 연결이 일어날 때 공격자 호스트의 MAC 주소를 사용하게 된다. 결국 두 호스트간의 모든 트래픽은 공격자가 위치한 세그먼트로 들어오게 된다.

이러한 경우 arp redirect 공격과 마찬가지로 공격자 호스트로 넘어오는 트래픽을 본래의 호스트로 relay 해주어야만 두 호스트 간에 정상적인 연결을 할 수 있게 되고 스니핑도 할 수 있다. 그렇지 않으면 두 호스트간의 연결은 이루어 질 수 없게 되고 결국 스니핑도 할 수 없게 된다.

다음은 "arpmitm"이라는 공격 프로그램을 이용하여 172.16.2.15와 172.16.2.18번 호스트 간의 트래픽을 스니핑 하기 위한 공격했을 때 네트워크상에 나타나는 arp 패킷을 tcpdump를 이용하여 잡은 모습이다.

### 4) ICMP Redirect 공격

ICMP(Internet Control Message Protocol)는 네트워크 에러 메시지를 전송하거나 네트워크 흐름을 통제하기 위한 프로토콜인데 ICMP Redirect를 이용해서 스니핑 할 수 있는 방법이 존재한다. ICMP Redirect 메시지는 하나의 네트워크에 여러개의 라우터가 있을 경우, 호스트가 패킷을 올바른 라우터에게 보내도록 알려주는 역할을 한다. 공격자는 이를 악용하여 다른 세그먼트에 있는 호스트에게 위조된 ICMP Redirect 메시지를 보내 공격자의 호스트로 패킷을 보내도록하여 패킷을 스니핑하는 방법이다.

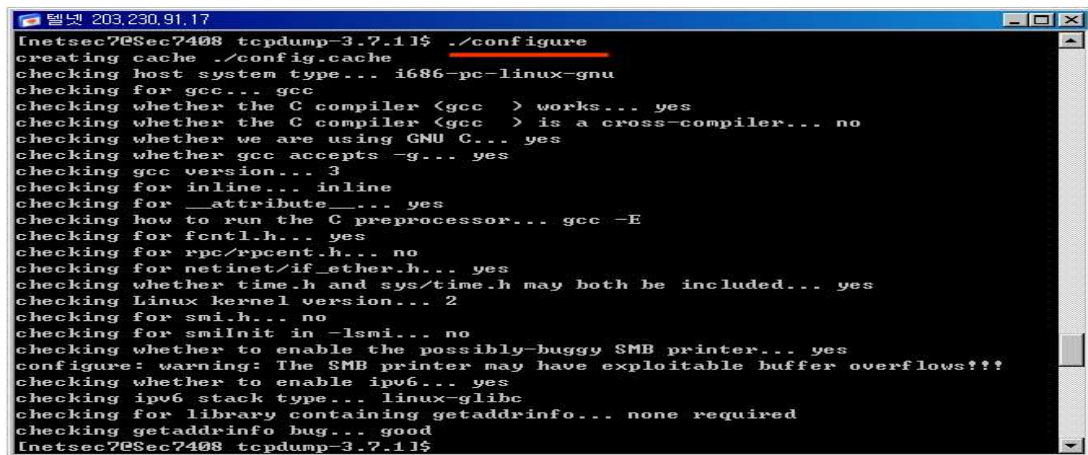


## 5. 스니핑 공격 툴

### 1) TCP Dump

가장 일반적인 스니핑 툴이다. 관리자적인 느낌이 강한 스니퍼이며, 네트워크 관리를 위해 개발되었다.

- ① TCP Dump 소스를 압축에서 풀고 ./configure를 입력한다.

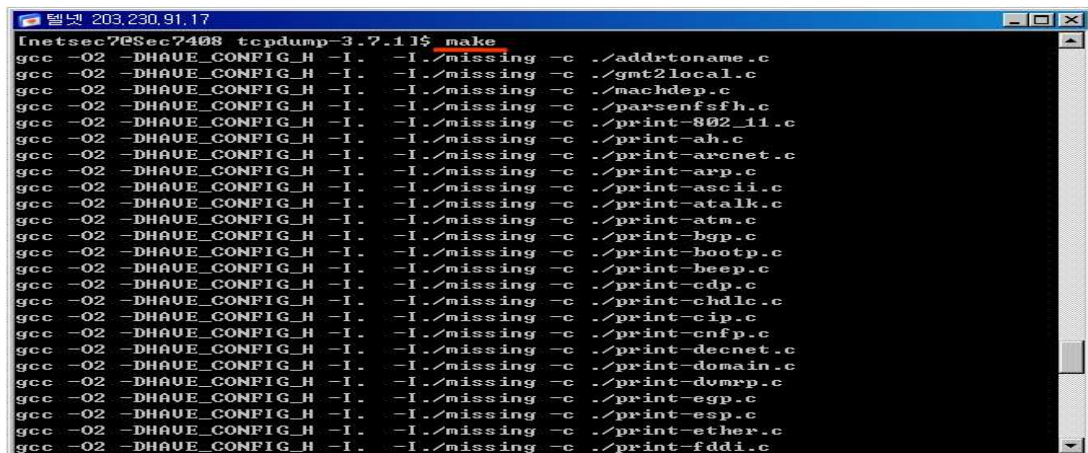


```

Inetsec7@Sec7408 tcpdump-3.7.1$ ./configure
creating cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler (gcc) works... yes
checking whether the C compiler (gcc) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking gcc version... 3
checking for inline... inline
checking for __attribute__... yes
checking how to run the C preprocessor... gcc -E
checking for fcntl.h... yes
checking for rpc/rpc.h... no
checking for netinet/if_ether.h... yes
checking whether time.h and sys/time.h may both be included... yes
checking Linux kernel version... 2
checking for smi.h... no
checking for smilnit in -lsmi... no
checking whether to enable the possibly-buggy SMB printer... yes
configure: warning: The SMB printer may have exploitable buffer overflows!!!
checking whether to enable ipv6... yes
checking ipv6 stack type... linux-glibc
checking for library containing getaddrinfo... none required
checking getaddrinfo bug... good
Inetsec7@Sec7408 tcpdump-3.7.1$

```

- ② 오브젝트 코드를 만들기 위해 make를 입력하고, make install로 컴퓨터에 인스톨한다.



```

Inetsec7@Sec7408 tcpdump-3.7.1$ make
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./addrtoname.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./gmt2local.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./machdep.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./parsenfsfh.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-802.11.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-ah.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-arcnet.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-arp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-ascii.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-atalc.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-atm.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-bgp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-bootp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-beep.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-cdp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-chdlc.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-cip.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-cnfp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-decnet.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-domain.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-dvmrp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-egp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-esp.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-ether.c
gcc -O2 -DHAVE_CONFIG_H -I. -I./missing -c ./print-fddi.c

```

④ 위 과정을 마쳤으면 실행명령을 통해 실행해 본다.

```

root@Sec7408 netsec71# /usr/sbin/tcpdump -a
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
21:35:46.834663 arp who-has 203.230.86.232 tell 203.230.86.4
...
21:35:46.835744 IP 203.230.91.17.32770 > ns2.kiu.ac.kr.domain: 2161+ PTR? 232.8
6.230.203.in-addr.arpa. <45>
...
21:35:46.836522 IP ns2.kiu.ac.kr.domain > 203.230.91.17.32770: 2161 NXDomain* 0
/1/0 <99>
...
21:35:46.836819 IP 203.230.91.17.32770 > ns2.kiu.ac.kr.domain: 33943+ PTR? 4.8
.230.203.in-addr.arpa. <43>
...
21:35:46.837489 IP ns2.kiu.ac.kr.domain > 203.230.91.17.32770: 33943 NXDomain*
0/1/0 <97>
...
21:35:46.838024 IP 203.230.91.17.32770 > ns2.kiu.ac.kr.domain: 18989+ PTR? 12.
0.230.203.in-addr.arpa. <44>

```

⑤ Telnet 계정 ID의 경우

```

root@wishfree:~#
22:58:35.780875 172.16.0.2.32852 > 172.16.0.3.23: P [tcp sum ok] 145:146(1) ack 127 win 5840 <nop,nop,timestamp 49678 40>
...
22:58:35.781162 172.16.0.3.23 > 172.16.0.2.32852: P [tcp sum ok] 127:128(1) ack 146 win 5792 <nop,nop,timestamp 406136 4>
...
22:58:35.781212 172.16.0.2.32852 > 172.16.0.3.23: . [tcp sum ok] 146:146(0) ack 128 win 5840 <nop,nop,timestamp 49678 40>
...
22:58:35.781212 172.16.0.2.32852 > 172.16.0.3.23: P [tcp sum ok] 146:147(1) ack 128 win 5840 <nop,nop,timestamp 49689 40>
...
22:58:35.781212 172.16.0.2.32852 > 172.16.0.3.23: . [tcp sum ok] 147:147(0) ack 129 win 5840 <nop,nop,timestamp 49689 40>

```

⑥ Telnet 계정 PW의 경우

```

root@wishfree:~#
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: P [tcp sum ok] 155:156(1) ack 147 win 5840 <nop,nop,timestamp 61101 41>
...
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: . [tcp sum ok] 147:147(0) ack 156 win 5792 <nop,nop,timestamp 417564 6>
...
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: P [tcp sum ok] 156:157(1) ack 147 win 5840 <nop,nop,timestamp 61119 41>
...
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: . [tcp sum ok] 147:147(0) ack 157 win 5792 <nop,nop,timestamp 417578 6>
...
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: P [tcp sum ok] 157:158(1) ack 147 win 5840 <nop,nop,timestamp 61138 41>
...
23:00:30.010924 172.16.0.2.32876 > 172.16.0.3.23: . [tcp sum ok] 147:147(0) ack 158 win 5792 <nop,nop,timestamp 417597 6>

```

## 2) Dsniff

DSniff는 스니핑을 위한 자동화 툴이다. 많은 이들이 SSL과 같은 암호화를 쓰는 통신이 안전하다고 생각하나, DSniff는 이렇게 암호화된 계정과 패스워드까지 읽어내는 능력이 있다. DSniff가 읽어낼 수 있는 패킷은 다음과 같다. ftp, telnet, http, pop, nntp, imap, snmp, ldap, rlogin, rip, ospf, pptp, ms-chap, nfs, yp/nis+, socks, x11, cvs, IRC, ATM, ICQ, PostageSQL, Citrix ICA, Symantec pcAnywhere, M.S. SQL, auth, info 등이 있다.

툴	기능
filesnarf	NFS 트래픽에서 스니프한 파일을 현재 디렉토리에 저장한다.
macof	스위치를 허브와 같이 작동하게 하기 위하여 임의의 MAC 주소로 스위치의 MAC 테이블을 오버플로우(Overflow)시킨다.
mailsnarf	SNMP와 POP 을 스니프하여 이메일을 볼 수 있게 해준다.
msgsnarf	채팅 메시지를 스니핑한다.
tcpkill	탐지할 수 있는 TCP 세션을 모두 끊는다.
tcprace	ICMP source quench 메시지를 보내 특정 TCP 연결을 느리게 만든다. 속도가 빠른 네트워크에서 스니프할 때
arp spoof	ARP 스푸핑 공격을 실행한다.
dns spoof	DNS 스푸핑 공격을 실행한다.
urlsnarf	CLF(Common Log Format)에서 HTTP 트래픽을 스니핑하여 선택된 URL을 알려준다.

### (1) Dsniff를 이용하여 여러 가지 공격

#### ① Dsniff 공격

```

root@wishfree:~
[root@wishfree root]# dsniff
dsniff: listening on eth0

07/20/03 23:10:06 tcp 172.16.0.2.32994 -> 172.16.0.3.23 (telnet)
wishfree
qwer1234
ls
mkdir test2
cd test2
exit

07/20/03 23:10:32 tcp 172.16.0.2.33007 -> 172.16.0.3.21 (ftp)
USER wishfree
PASS qwer1234
  
```

## ② TcpKill을 이용한 세션 끊기

```

root@wishfree:~# tcpkill
Version: 2.3
Usage: tcpkill [-i interface] [-l..9] expression
[root@wishfree root]# tcpkill -i eth0 -4 tcp
tcpkill: listening on eth0 [tcp]
207.46.107.9:1863 > 204.168.1.104:1153: R 767390011:767390011(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767407023:767407023(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767424035:767424035(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767441047:767441047(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620412773:1620412773(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620429815:1620429815(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620446857:1620446857(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620463899:1620463899(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767390011:767390011(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767407023:767407023(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767424035:767424035(0) win 0
207.46.107.9:1863 > 204.168.1.104:1153: R 767441047:767441047(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620412773:1620412773(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620429815:1620429815(0) win 0
204.168.1.104:1153 > 207.46.107.9:1863: R 1620446857:1620446857(0) win 0

```

## ③ Tcpnice을 이용한 트래픽 속도 느리게 만들기

```

root@wishfree:~# tcptnice
Version: 2.3
Usage: tcptnice [-i interface] [-l] [-n increment] expression
[root@wishfree root]# tcptnice -i eth0 -n 20 tcp
tcptnice: listening on eth0 [tcp]
204.168.1.104:1203 > 211.115.210.8:80: . ack 3730823487 win 8
204.168.1.104:1203 > 211.115.210.8:80: . ack 3730823487 win 8
211.115.210.8:80 > 204.168.1.104:1203: . ack 476143015 win 8
211.115.210.8:80 > 204.168.1.104:1203: . ack 476143015 win 8
204.168.1.104:1203 > 211.115.210.8:80: . ack 3730824025 win 8
211.115.210.8:80 > 204.168.1.104:1203: . ack 476143016 win 8
204.168.1.104:1204 > 211.115.210.8:80: . ack 3748352213 win 8
204.168.1.104:1204 > 211.115.210.8:80: . ack 3748352213 win 8
211.115.210.8:80 > 204.168.1.104:1204: . ack 1951193540 win 8
211.115.210.8:80 > 204.168.1.104:1204: . ack 1951193540 win 8
204.168.1.104:1204 > 211.115.210.8:80: . ack 3748352418 win 8
211.115.210.8:80 > 204.168.1.104:1204: . ack 1951193541 win 8
204.168.1.104:1205 > 211.115.210.8:80: . ack 3734158840 win 8
204.168.1.104:1205 > 211.115.210.8:80: . ack 3734158840 win 8
211.115.210.8:80 > 204.168.1.104:1205: . ack 2972077883 win 8

```



## ④ Urlnarf를 이용한 웹 서핑 감시

```

root@WishFree:~/dsniff-2.3
[root@MshFree dsniff-2.3]# urlnarf
urlnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
210.92.110.59 -- [17/Aug/2003:20:39:18 +0900] "GET http://www.daum.net/ HTTP/1.1" -- "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"
210.92.110.59 -- [17/Aug/2003:20:39:18 +0900] "GET http://www.daum.net/daum2.ico HTTP/1.1" -- "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"
210.92.110.59 -- [17/Aug/2003:20:39:19 +0900] "GET http://amsv2.daum.net/cgi-bin/adcgi?corpId=46&secId=00122&type=cpm&tag=javascript&adcnt=1&ord=0546941446395694 HTTP/1.1" -- "http://www.daum.net/" "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"
210.92.110.59 -- [17/Aug/2003:20:39:19 +0900] "GET http://amsv2.daum.net/cgi-bin/adcgi?corpId=46&secId=00113&type=cpm&tag=javascript&ord=05386776241538721 HTTP/1.1" -- "http://www.daum.net/" "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"
210.92.110.59 -- [17/Aug/2003:20:39:19 +0900] "GET http://image.hanmail.net/hanmail/top/highlight/avata_0814.gif HTTP/1.1" -- "http://www.daum.net/" "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"
210.92.110.59 -- [17/Aug/2003:20:39:19 +0900] "GET http://amsv2.daum.net/cgi-bin/adcgi?corpId=46&secId=00115&type=cpm&tag=javascript&ord=7084761396678829 HTTP/1.1" -- "http://www.daum.net/" "Mozilla/5.0 (X11; U; Linux i686; ko-KR; rv:1.2.1) Gecko/20030225"

```

## ⑤ mailsnarf를 이용한 메일 스니핑

```

root@WishFree:~/dsniff-2.3
[root@MshFree dsniff-2.3]# mailsnarf
mailsnarf: listening on eth0
From root@127.0.0.1 Sun Aug 17 21:01:52 2003
Received: from localhost.localdomain (localhost.localdomain [127.0.0.1])
        by WishFree.localdomain (8.12.8/8.12.8) with SMTP id h7HC1KL3004335
        for wishfree@empal.com; Sun, 17 Aug 2003 21:01:43 +0900
Date: Sun, 17 Aug 2003 21:01:20 +0900
From: root@127.0.0.1
Message-Id: <200308171201.h7HC1KL3004335@MshFree.localdomain>
X-Authentication-Warning: WishFree.localdomain: localhost.localdomain [127.0.0.1]
        didn't use HELO protocol

How are you!!

```

## ⑥ Msgsnarf를 이용한 메신저 통신 내용 스니핑

```

root@WishFree:~/dsniff-2.3
[root@MshFree dsniff-2.3]# msgsnarf
msgsnarf: listening on eth0
Aug 17 20:55:05 MSN unknown > : babo tange
Aug 17 20:55:13 MSN unknown > : mal som hae ba
Aug 17 20:55:25 MSN unknown > : ya!!!
Aug 17 20:55:44 MSN unknown > : hey mung rul

```

### 3) Sniffer Pro

기본적으로 스니핑이 지원되지 않으나, WinPCAP과 같은 라이브러리를 이용해서 스니핑이 가능하다. 윈도우 스니퍼는 뛰어난 GUI를 이용한 네트워크 상태를 점검하거나 패킷의 통계를 내기 위한 목적으로 많이 쓰인다.

#### ① 스니퍼 프로를 이용한 프로토콜 분석

No.	Status	Source Address	Dest Address	Summary	Len(B)	Re
1	M	Samsng13B380	Bridge_Group_Ad	BPDU: S: Pri=8000 Port=8004 Root: Pri=8000 Addr=0000	64	
2		Samsng13B380	Bridge_Group_Ad	BPDU: S: Pri=8000 Port=8004 Root: Pri=8000 Addr=0000	64	
3		Cisco D34485	01000CCCCC	CDP: Type=0x0001 (Device ID): Type=0x0002 (Address	381	
4		TYRANNUS	[192.168.0.255]	BROWSER: Local Master TYRANNUS Announce	243	
5		Samsng13B380	Bridge_Group_Ad	BPDU: S: Pri=8000 Port=8004 Root: Pri=8000 Addr=0000	64	
6		Samsng13B380	Bridge_Group_Ad	BPDU: S: Pri=8000 Port=8004 Root: Pri=8000 Addr=0000	64	
7		[192.168.0.152]	[211.115.210.8]	TCP: D=80 S=1929 SYN SEQ=2511461962 LEN=0 WIN=1638	62	
8		[211.115.210.8]	[192.168.0.152]	TCP: D=1929 S=80 SYN ACK=2511461963 SEQ=2295176152	62	
9		[192.168.0.152]	[211.115.210.8]	TCP: D=80 S=1929 ACK=2295176153 WIN=17520	60	
10		[192.168.0.152]	[211.115.210.8]	HTTP: C Port=1929 GET / HTTP/1.1	427	
11		[211.115.210.8]	[192.168.0.152]	HTTP: R Port=1929 HTTP/1.1 Status=Moved Permanent	594	
12		[192.168.0.152]	[211.115.210.8]	TCP: D=80 S=1929 FIN ACK=2295176693 SEQ=2511462336	60	
13		[211.115.210.8]	[192.168.0.152]	TCP: D=1929 S=80 FIN ACK=2511462336 SEQ=2295176693	60	
14		[192.168.0.152]	[211.115.210.8]	TCP: D=80 S=1929 ACK=2295176694 WIN=16980	60	
15		[211.115.210.8]	[192.168.0.152]	TCP: D=1929 S=80 ACK=2511462337 WIN=65162	60	
16		[192.168.0.152]	[211.115.210.9]	TCP: D=80 S=1930 SYN SEQ=1193271015 LEN=0 WIN=1638	62	
17		[211.115.210.9]	[192.168.0.152]	TCP: D=1930 S=80 SYN ACK=1193271016 SEQ=2295392190	62	
18		[192.168.0.152]	[211.115.210.9]	TCP: D=80 S=1930 ACK=2295392191 WIN=17520	60	
19		[192.168.0.152]	[211.115.210.9]	HTTP: C Port=1930 GET / HTTP/1.1	475	
20		[211.115.210.9]	[192.168.0.152]	HTTP: R Port=1930 HTTP/1.1 Status=Moved Temporary	268	
21		[211.115.210.9]	[192.168.0.152]	TCP: D=1930 S=80 FIN ACK=1193271437 SEQ=2295392405	60	
22		[192.168.0.152]	[211.115.210.9]	TCP: D=80 S=1930 ACK=2295392406 WIN=17306	60	
23		[192.168.0.152]	[211.115.210.9]	TCP: D=80 S=1930 FIN ACK=2295392406 SEQ=1193271437	60	

## 6. 맺음말

스니핑이란 해커들이 가장 많이 사용하는 기술이라고 한다. Sniffer라는 이름을 가진 Tool들이 인터넷상으로 널리 배포되어 패킷들을 재조합하고 원래 데이터를 충분히 복제할 수 있다. 이러한 공격을 방지하기 위하여 취할 수 있는 여러 가지 방법 등으로 예방할 수 있으나 모두 완벽할 수는 없고, 또 해커들의 능력이 날로 향상되고 있으므로 관리자 들은 자신의 시스템 환경에 맞는 대책을 강구해야만 할 것 이다.

## ◎ 참고자료

1. TCP/IP Illustrated, Volume1, The Protocols, W.Richard Stevens, ADDISON-WESLEY
2. dsniiff, <http://www.monkey.org/~dugsong/dsniiff/>
3. arpwatch, <ftp://ftp.ee.lbl.gov/>
4. arpmitm, <http://teso.scene.at/releases.php3>
5. Sniffing FAQ, <http://www.securitymap.net/docs/faq/sniffing-faq.htm>
6. Sniffing tool, [http://kmh.yeungnam-c.ac.kr/Hacking/Programs/stm/stm\\_sniffer.html](http://kmh.yeungnam-c.ac.kr/Hacking/Programs/stm/stm_sniffer.html)