Additional Slides + Examples 2020

From Carminati's Lectures

2. A (Quick) Introduction to Cryptography

Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

The Zip Example

***Algorithm**: C = K* xor *M*

- K(hex) = **AA BB CC DD** .. .. (repeat the key)
- M(hex) = 50 4B 03 04 BA DA 55 55 .. .. .. (and so on)

  XOR
- C(hex) = FA F0 CF D9 10 61 99 88 .. ..

NOT PERFECT -> len(k) < len(M) -> k is reused

The Zip Example

***Algorithm**: C = K* xor *M*

- K(hex) = **AA BB CC DD** .. .. (repeat the key)
- M(hex) = 50 4B 03 04 BA DA 55 55 .. .. .. (and so on)

  XOR
- C(hex) = FA F0 CF D9 10 61 99 88 .. ..

NOT PERFECT -> len(k) < len(M) -> k is reused

- *K = M xor C*
- *K = X X X X xor FA F0 CF D9*

```
michele@starkiller ~/Desktop
➤ xxd test.zip | head
00000000: 504b 0304 1400 0808 0800 bb74 3150 0000  PK.........t1P..
00000010: 0000 0000 0000 0000 0000 1400 0000 6974  ..............it
00000020: 2d36 3931 3439 3678 3038 3931 3635 2e70  -691496x089165.p
00000030: 6466 ccbb 6554 5ccb ba36 8abb bbbb 5be3  df..eT\..6....[.
00000040: eeee ee6e 8dbb bb6b 20b8 8510 9ce0 ee2e  ...n...k .......
00000050: 0182 bb3b c125 b806 bb49 d65e 7baf 7dce  ...;.%...I.^{.}.
00000060: face ddf7 8cef c71d 73d4 acaa 77be 56d2  ........s...w.V.
00000070: b3bb 9ef1 34a5 b2b8 2423 0b13 3b1c e5b7  ....4...$#..;...
00000080: 9dc9 5938 0e12 6612 4753 1b12 387e 7e38  ..Y8..f.GS..8~~8
00000090: 80ba b713 9004 20e1 e526 a5e6 66e2 0684  ...... ..&..f...
```

```
michele@starkiller ~/Desktop
➤ xxd test2.zip | head
00000000: 504b 0304 1400 0808 0000 c051 3050 0000  PK.........Q0P..
00000010: 0000 0000 0000 0000 0000 3200 0000 5241  ..........2...RA
00000020: 4d53 4553 2046 494e 414c 2052 6576 6965  MSES FINAL Revie
00000030: 7720 5054 2050 6572 7370 6563 7469 7665  w PT Perspective
00000040: 204a 616e 2032 3020 2831 292e 7070 7478   Jan 20 (1).pptx
00000050: 504b 0304 1400 0600 0800 0000 2100 4fac  PK..........!.O.
00000060: d3c4 4302 0000 d416 0000 1300 0802 5b43  ..C...........[C
00000070: 6f6e 7465 6e74 5f54 7970 6573 5d2e 786d  ontent_Types].xm
00000080: 6c20 a204 0228 a000 0200 0000 0000 0000  l ...(..........
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

The Zip Example

***Algorithm**: C = K* xor *M*

- K(hex) = AA BB CC DD .. .. (repeat the key)
- M(hex) = 50 4B 03 04 BA DA 55 55 .. .. .. (and so on)

  XOR
- C(hex) = FA F0 CF D9 10 61 99 88 .. ..

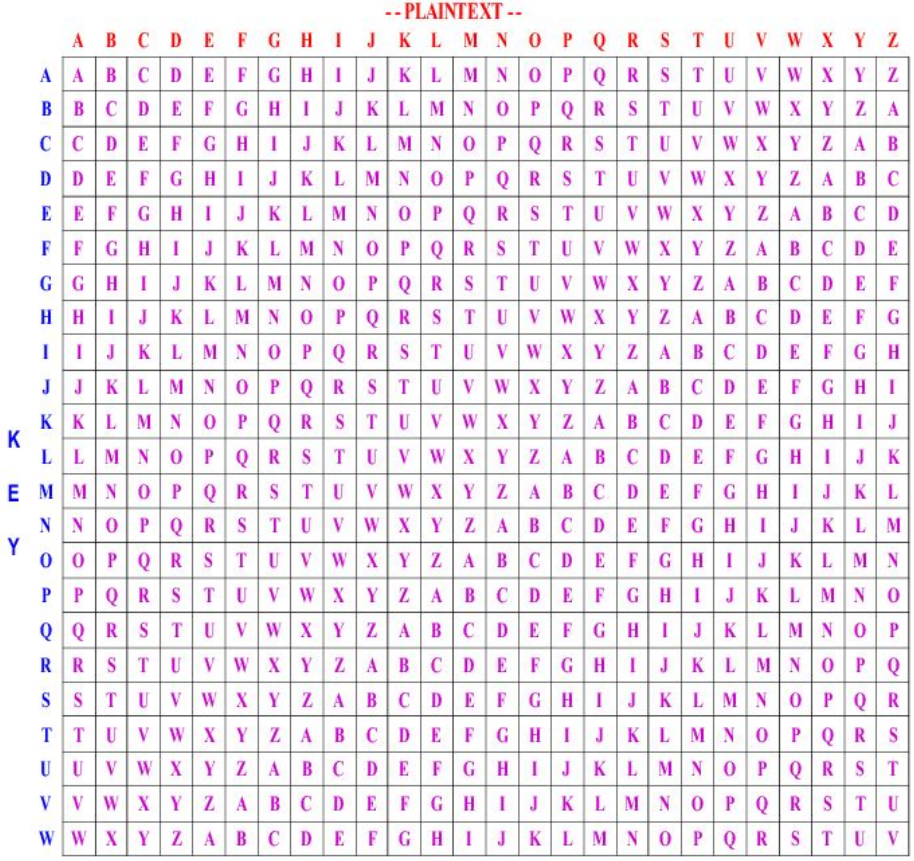
- *K = M xor C*
- *K =* 50 4B 03 04 *xor* FA F0 CF D9 = AA BB CC DD ->
  - KNOWN PLAINTEXT ATTACK

Polyalphabetic ciphers (Vigenere Cipher)

m=SECURE

k=SECRET

c=?

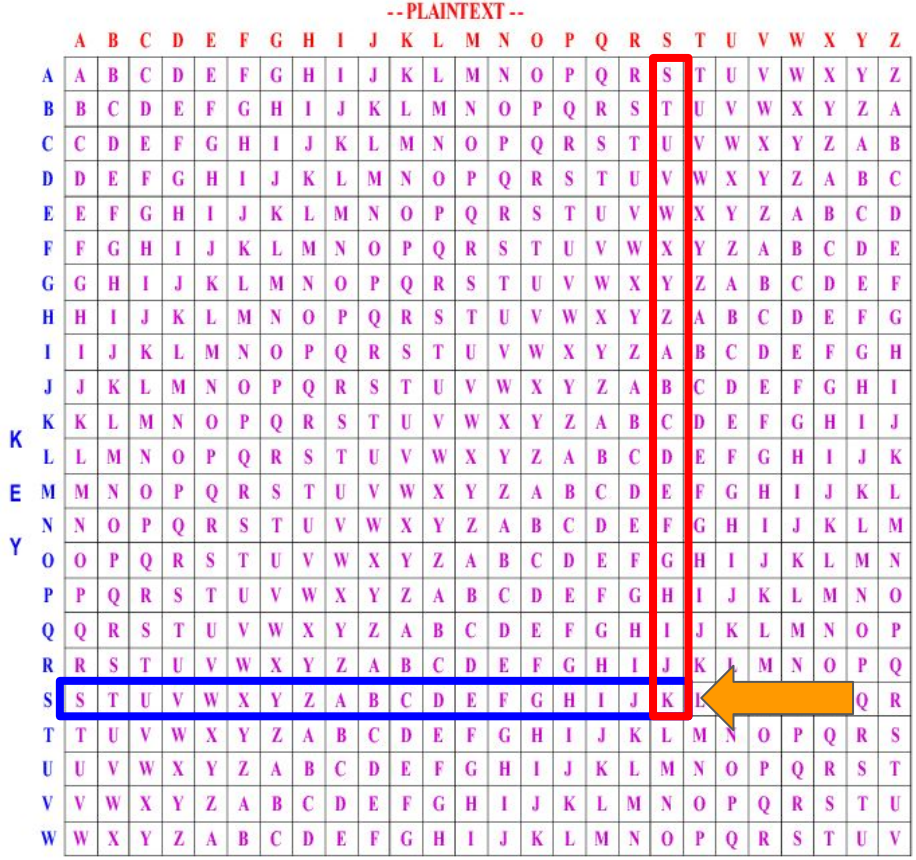


Polyalphabetic ciphers (Vigenere Cipher)

m=SECURE

k=SECRET

c=K

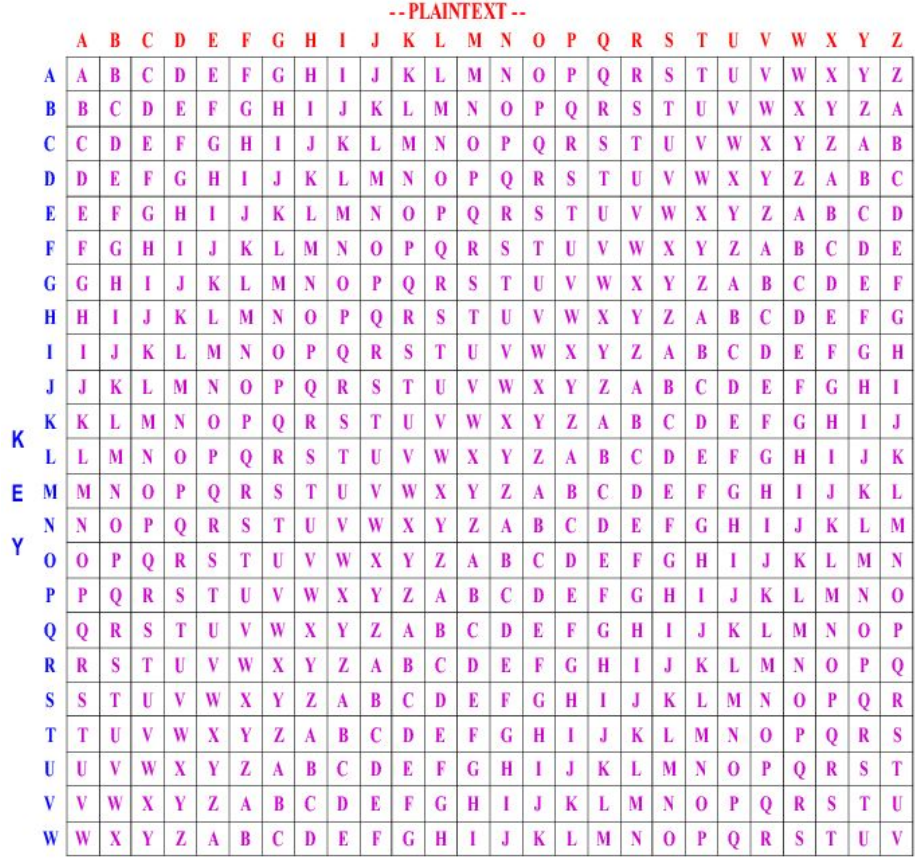


Polyalphabetic ciphers (Vigenere Cipher)

m=SECURE

k=SECRET

c=KIELVX



Example - Diffusion

m= HALLO EVERYONE!

k=(3,5)

c=H YAEOLVNLEEOR!

H	**A**	**L**	**L**	**O**
	E	**V**	**E**	**R**
Y	**O**	**N**	**E**	**!**

Example - Diffusion

m= HALLO

k=(3,5)

c=HALLO

H	A	L	L	O

R * C >> len(msg)

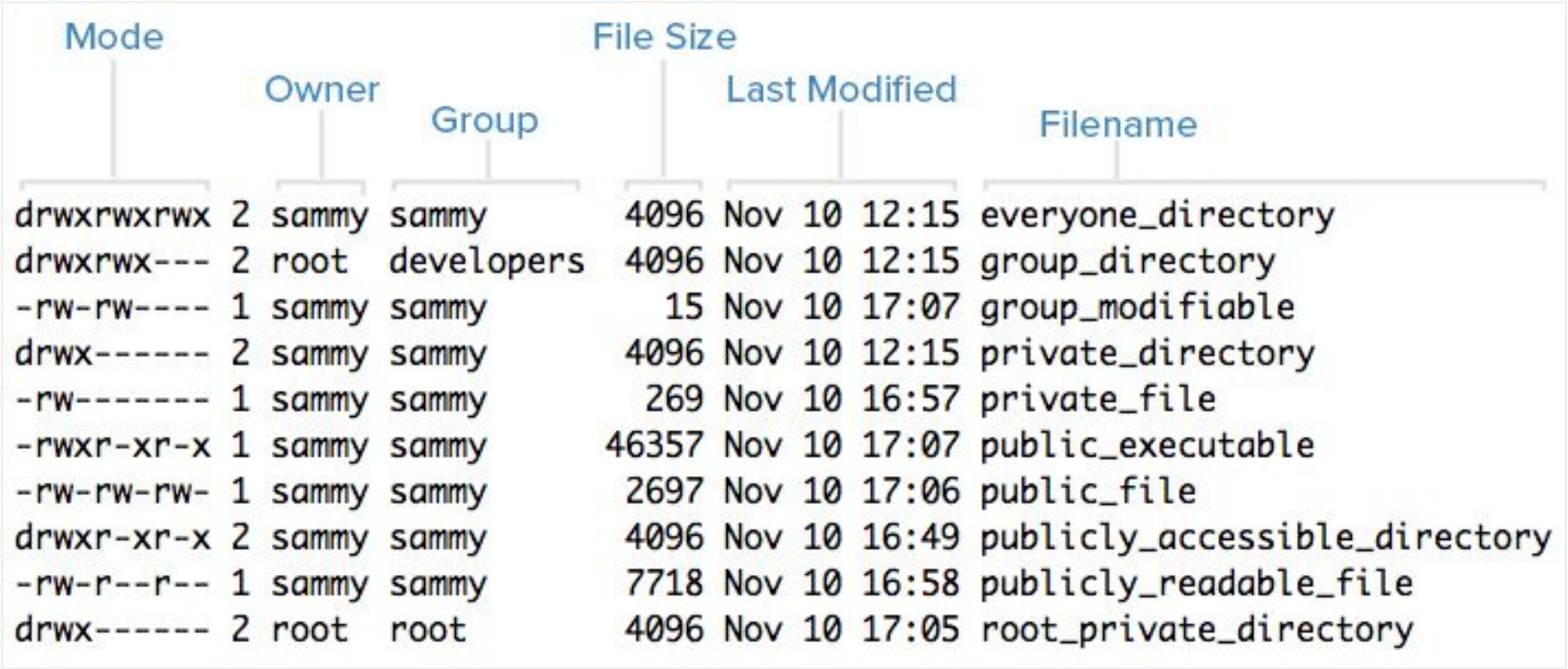
15 != 4

4. Access Control

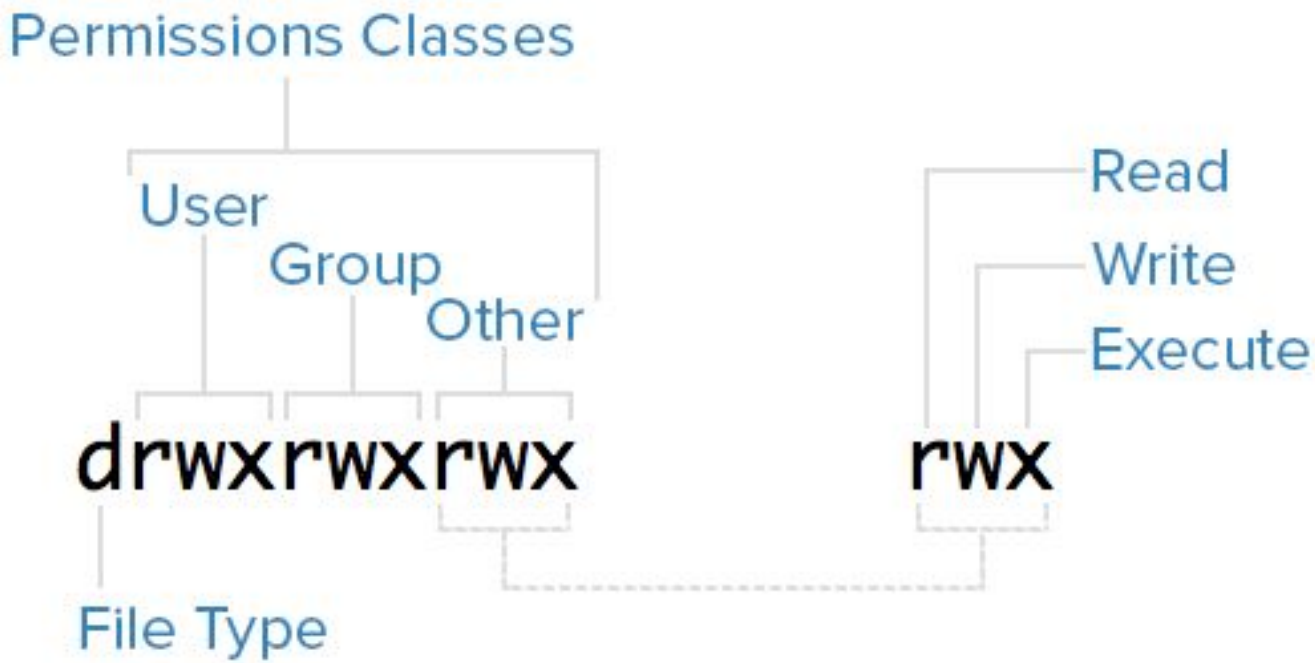
Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

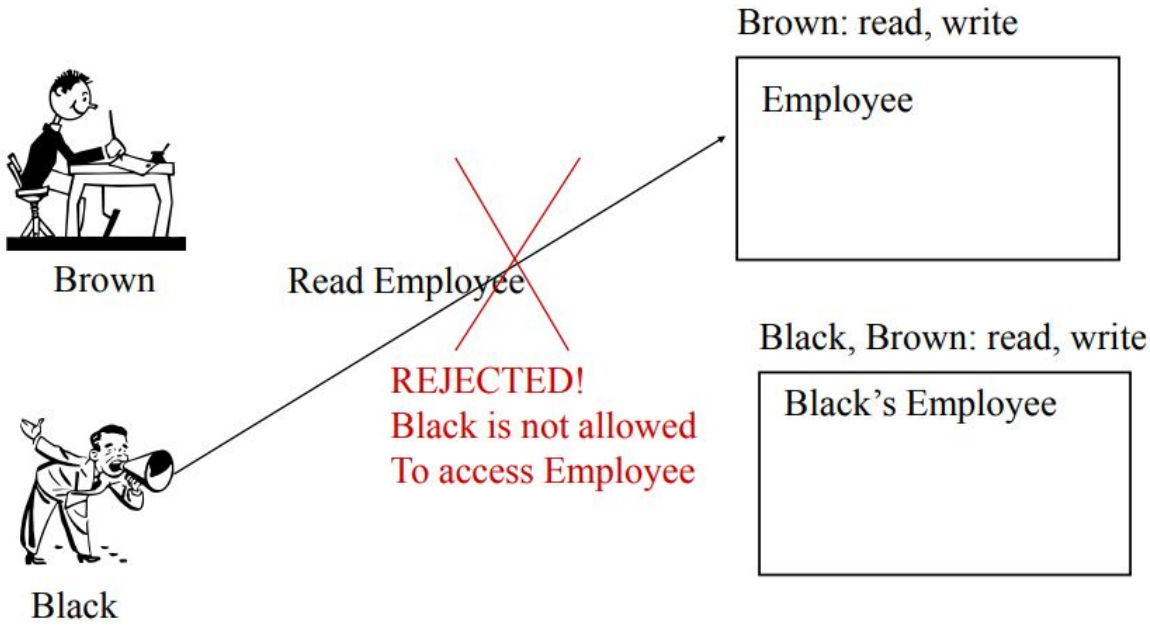
UNIX Permissions



Permissions “Triads”

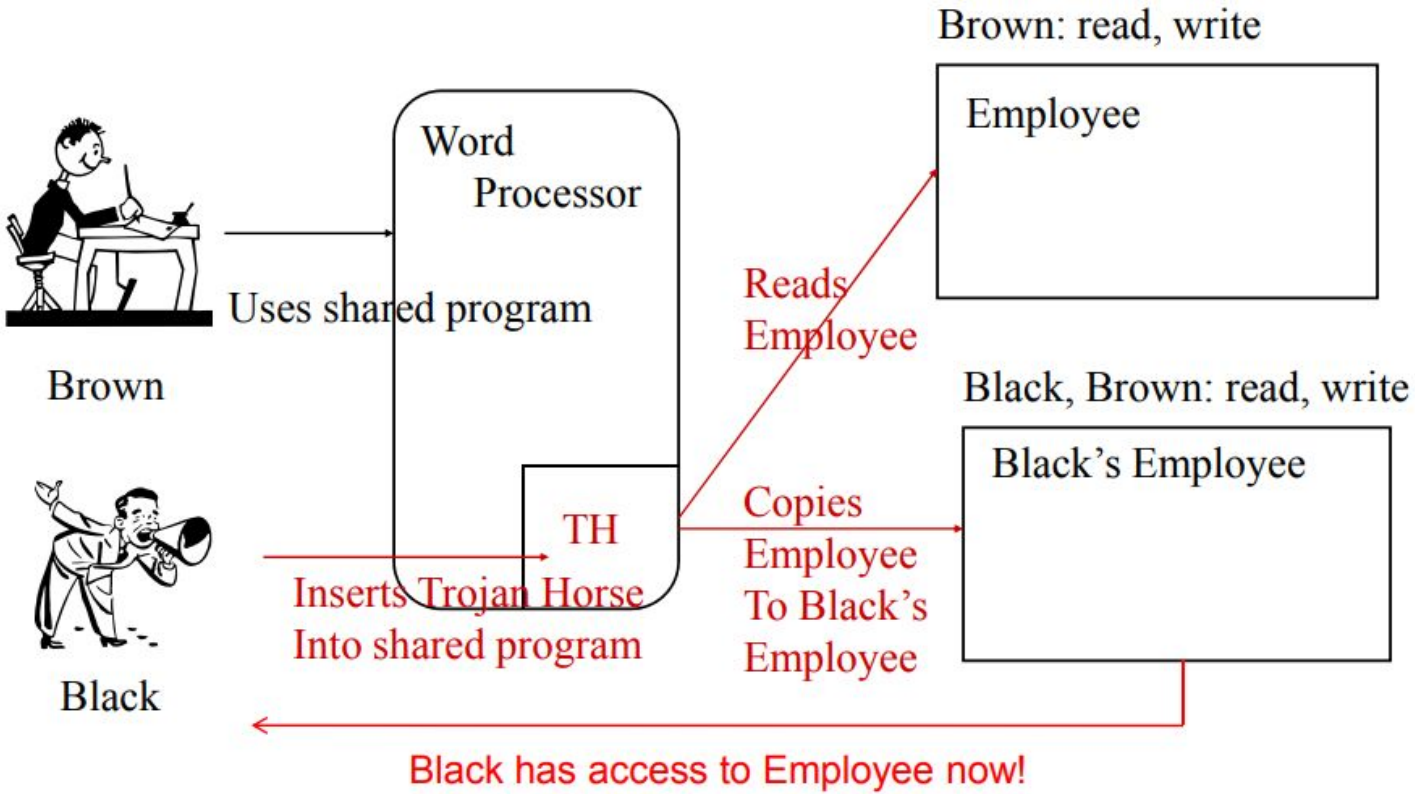


DAC and Trojan Horse



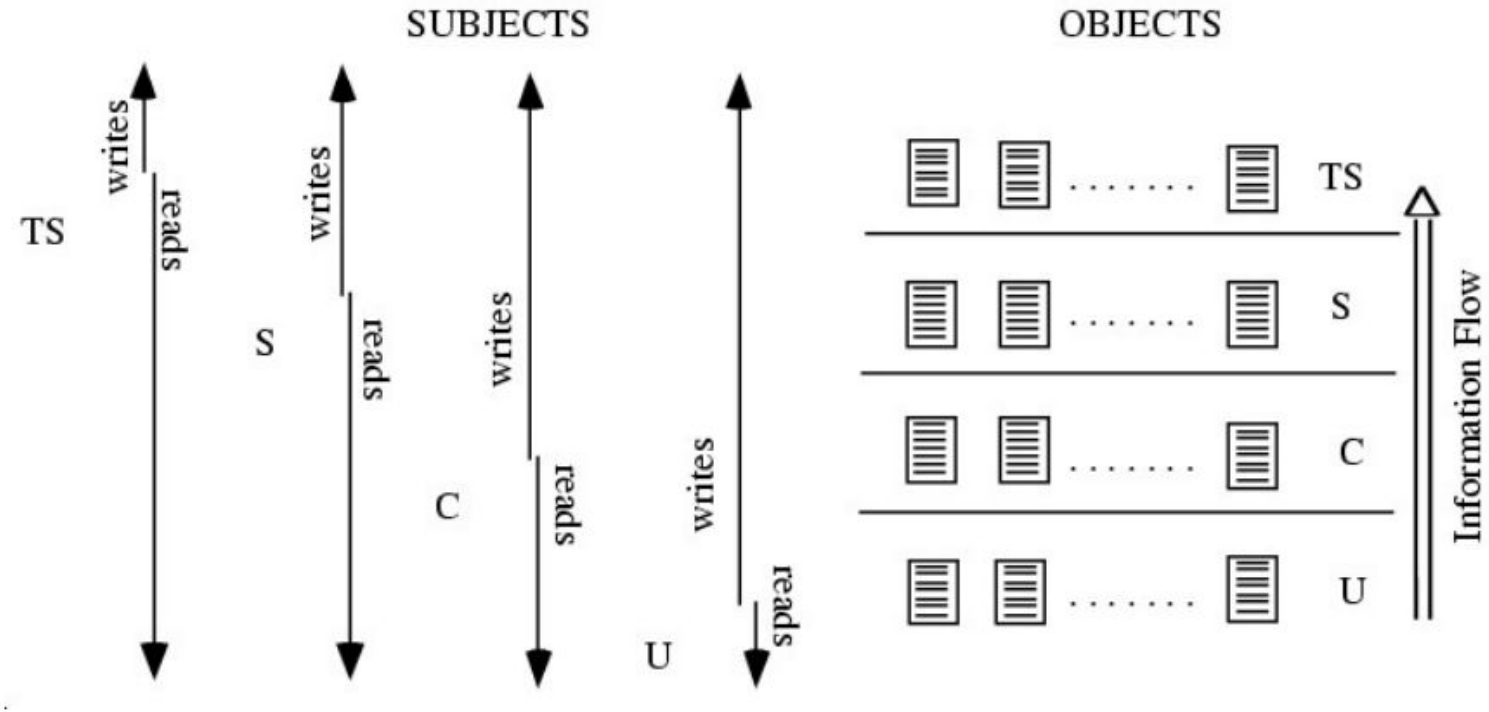
credits: https://www.comp.nus.edu.sg/~tankl/cs5322/slides/mac2.pdf

DAC Trojan Horse Problem



credits: https://www.comp.nus.edu.sg/~tankl/cs5322/slides/mac2.pdf

MAC Information Flow



credits: https://www.comp.nus.edu.sg/~tankl/cs5322/slides/mac2.pdf

6. Buffer Overflows

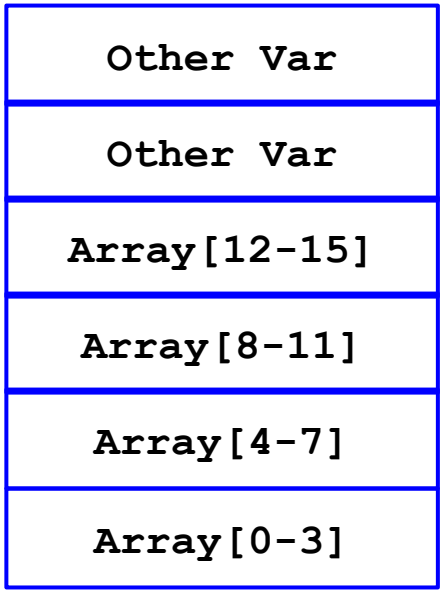
Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

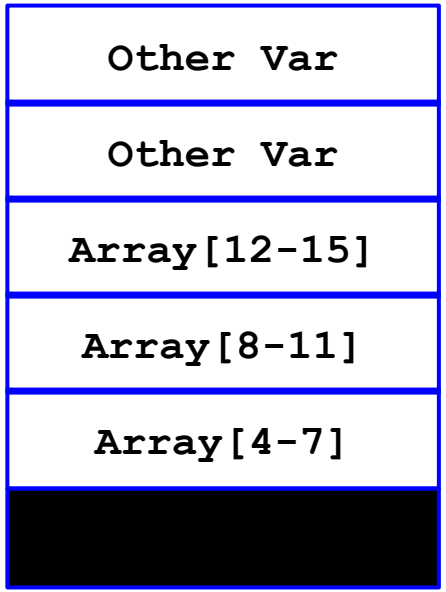
Buffer Overflow

`Int Array[16]`

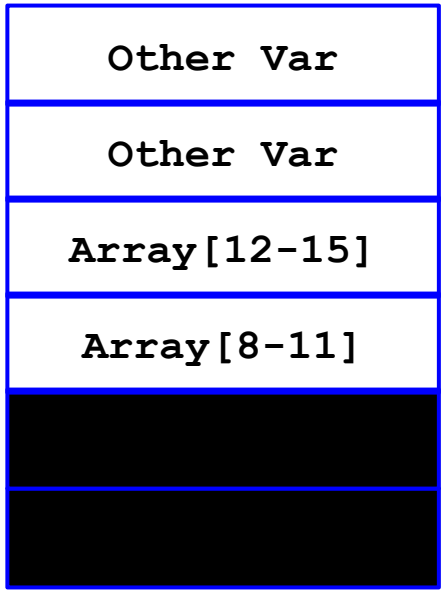
No check for oversized input



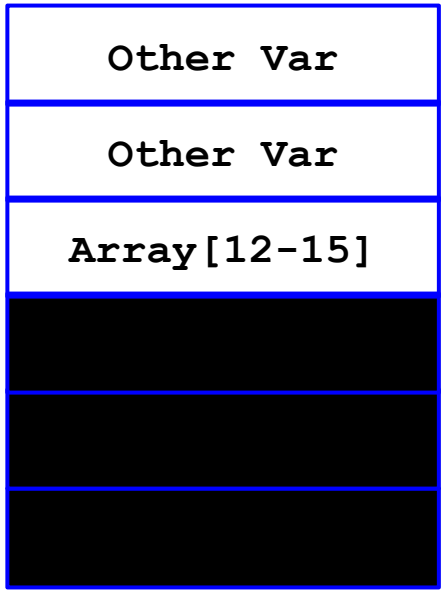
Buffer Overflow



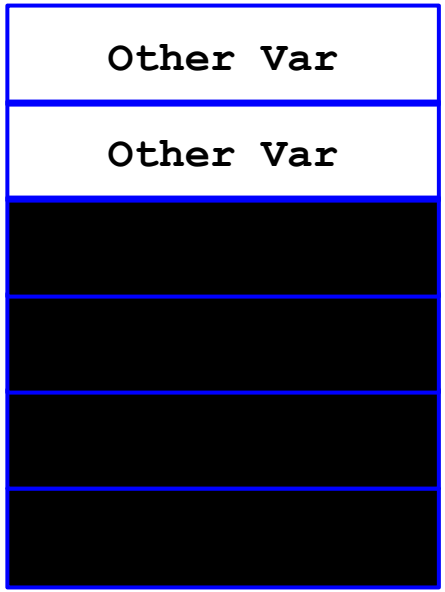
Buffer Overflow



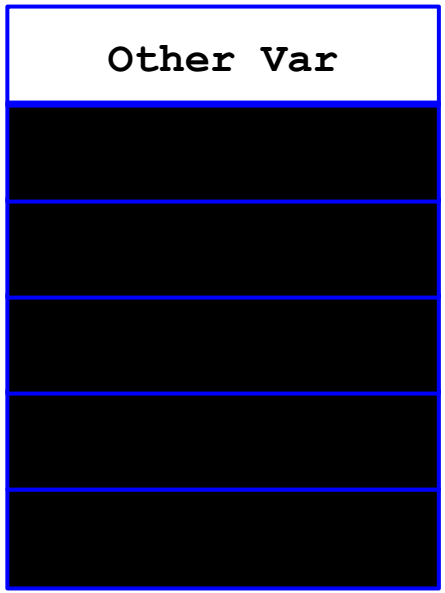
Buffer Overflow



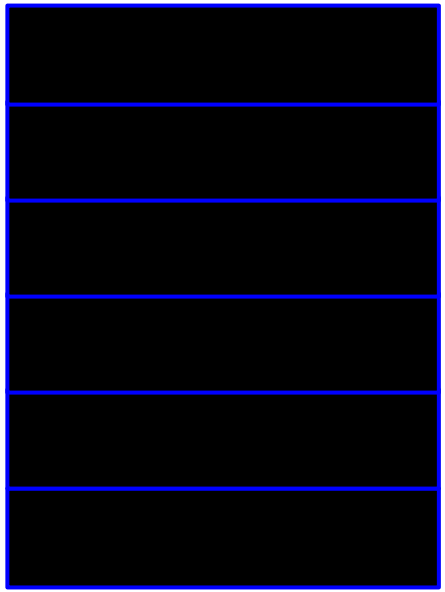
Buffer Overflow



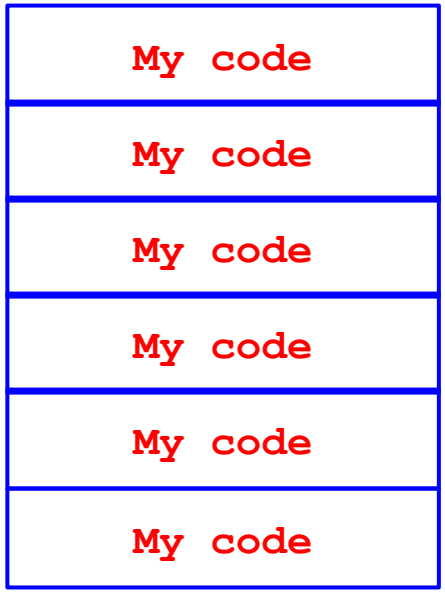
Buffer Overflow



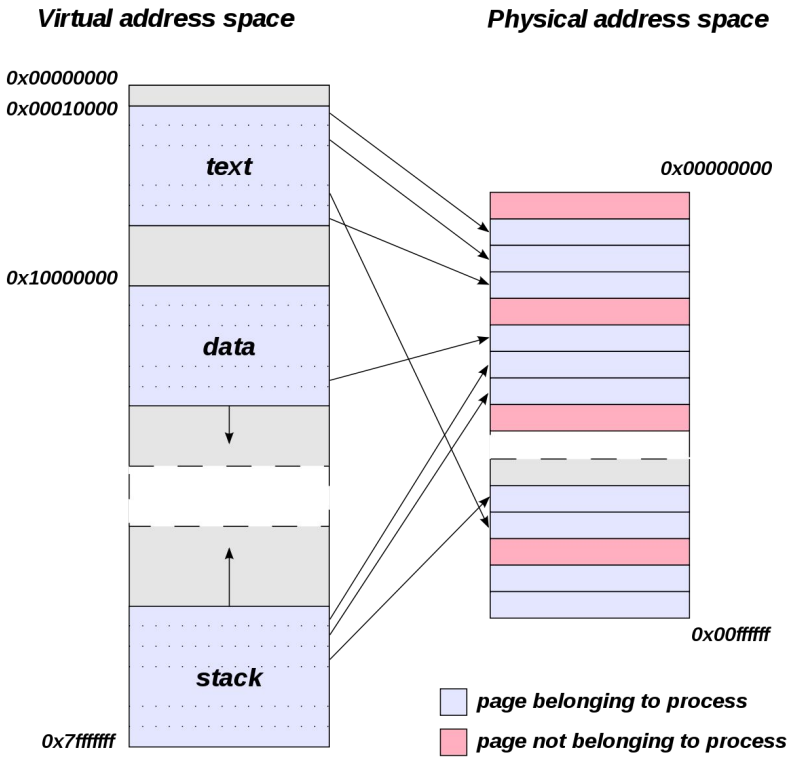
Buffer Overflow



Buffer Overflow

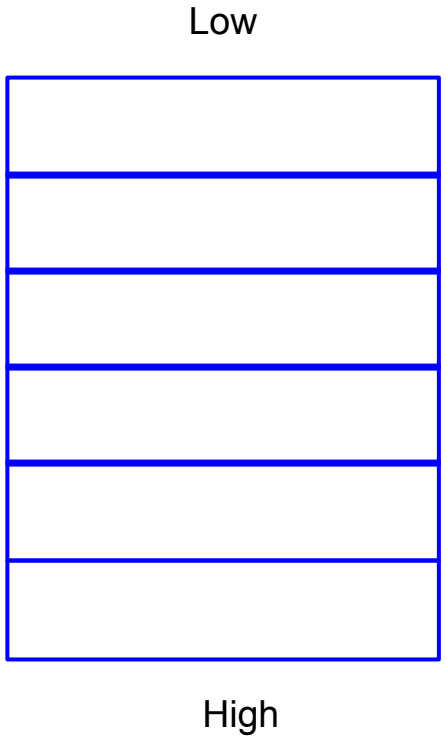


Virtual vs Physical Address Space



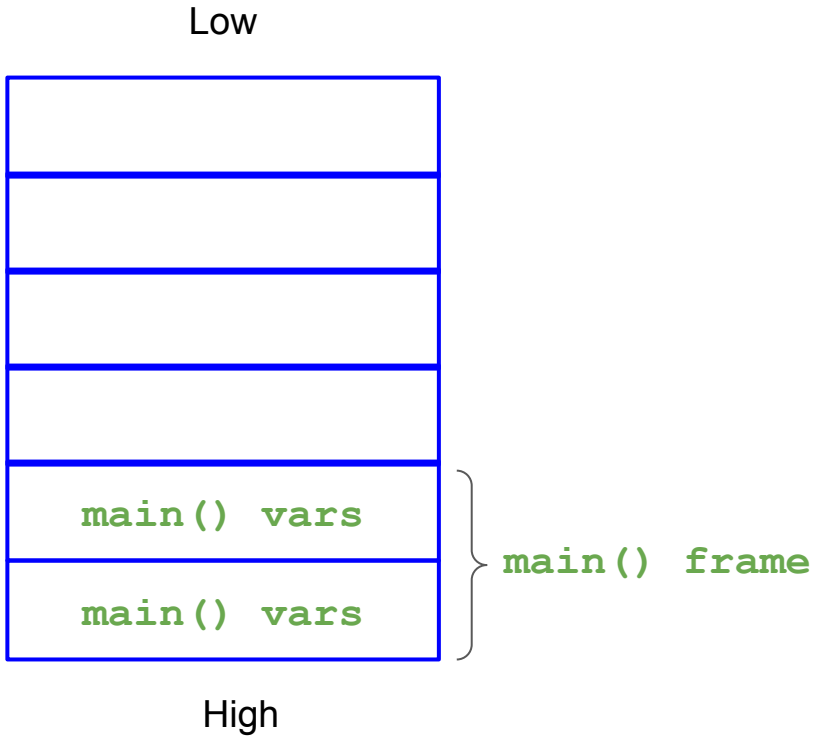
Buffer Overflow

```
void f2(){
    ...
     }
void f1(){
   f2();
     }
void main(){
    ...
   f1();
  }
```



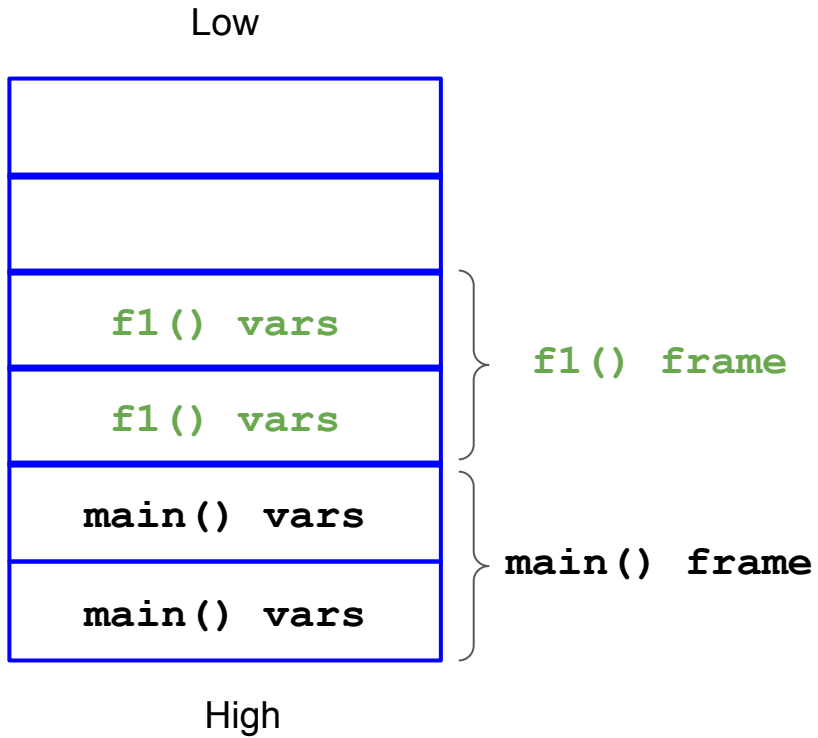
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



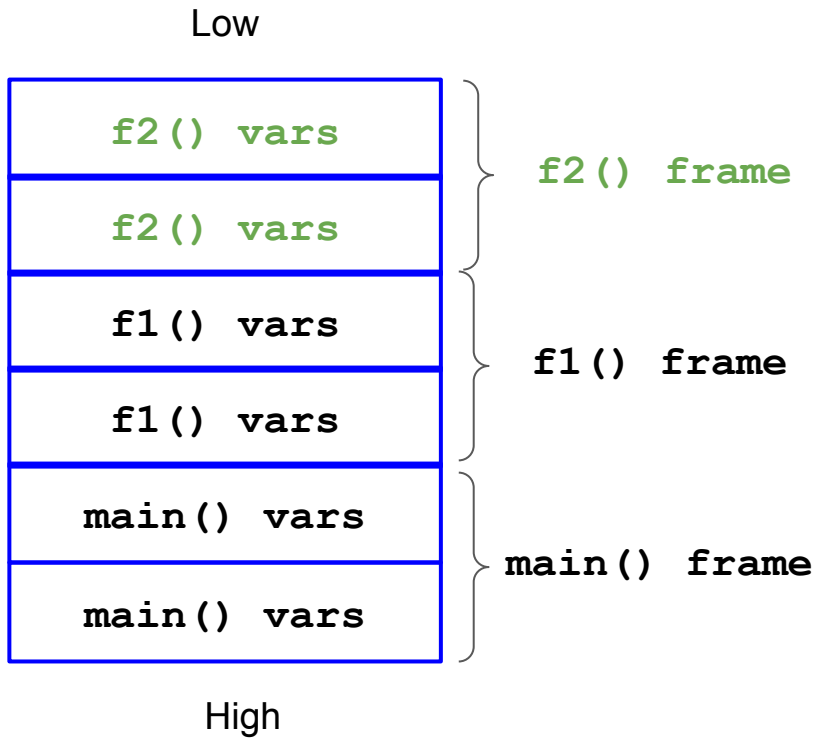
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



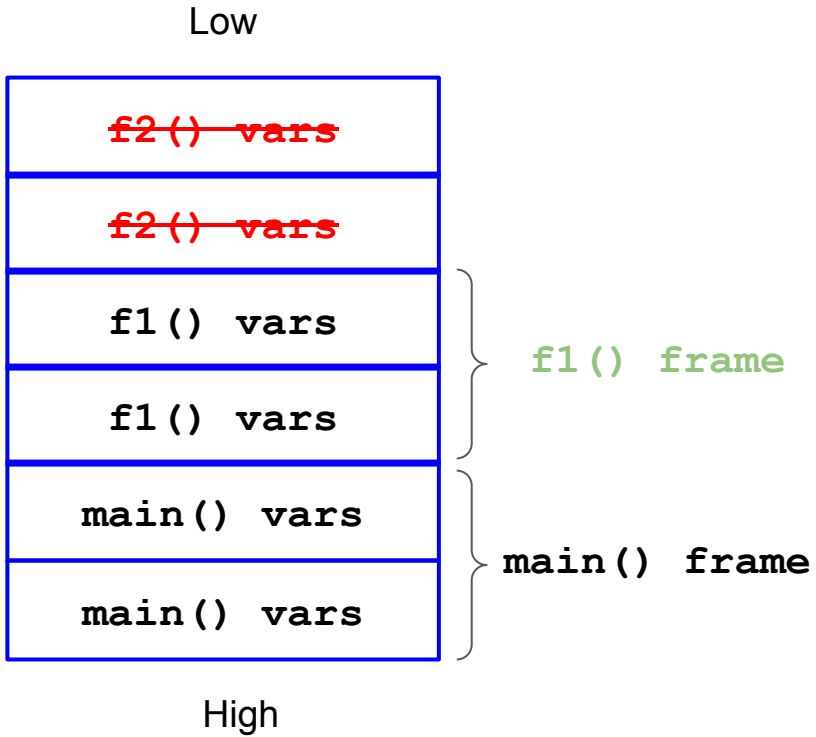
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



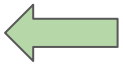
Buffer Overflow

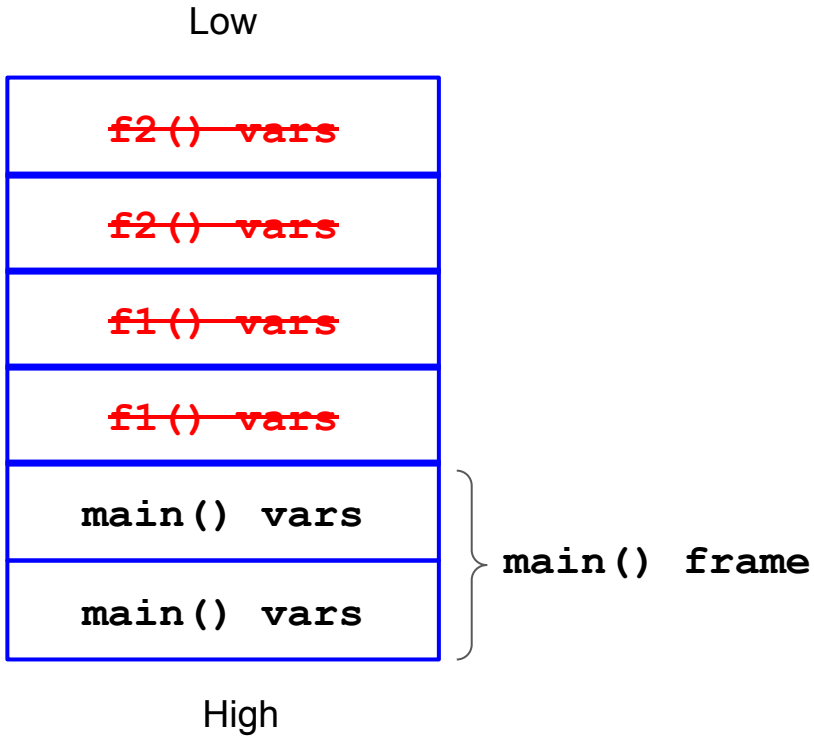
```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



Buffer Overflow

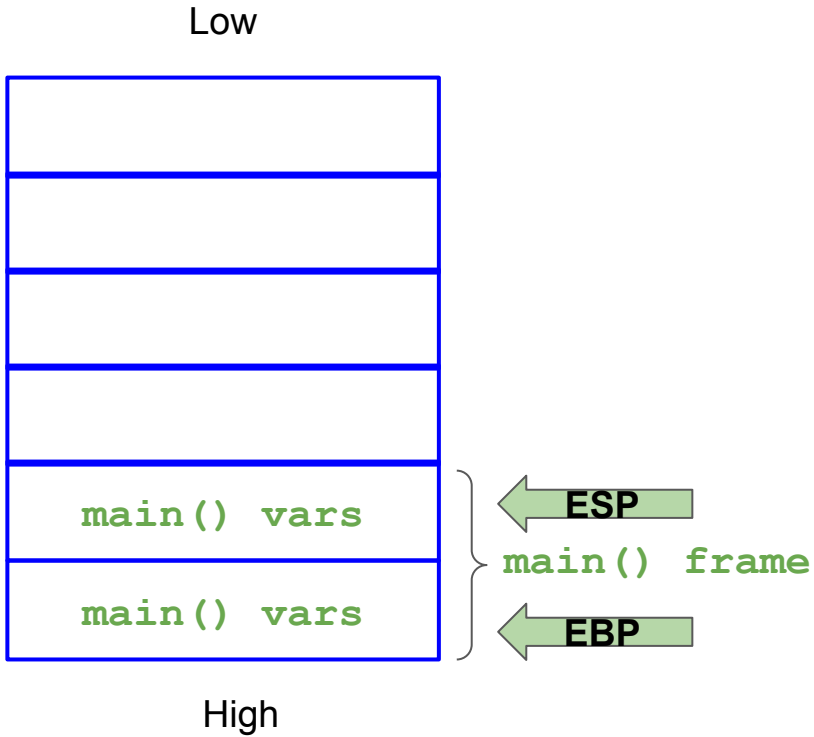
```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
 }
```





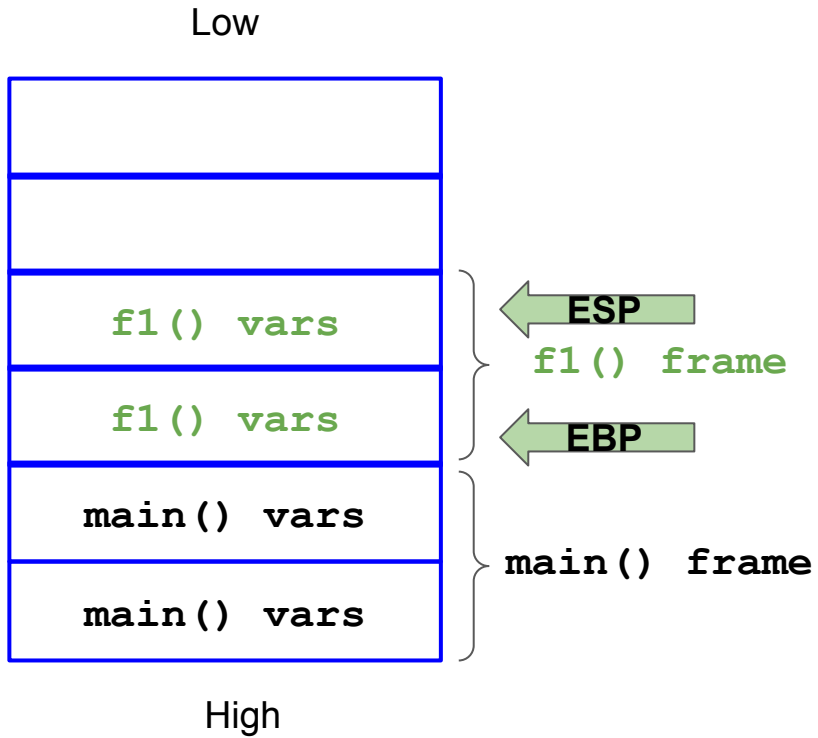
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
 }
```



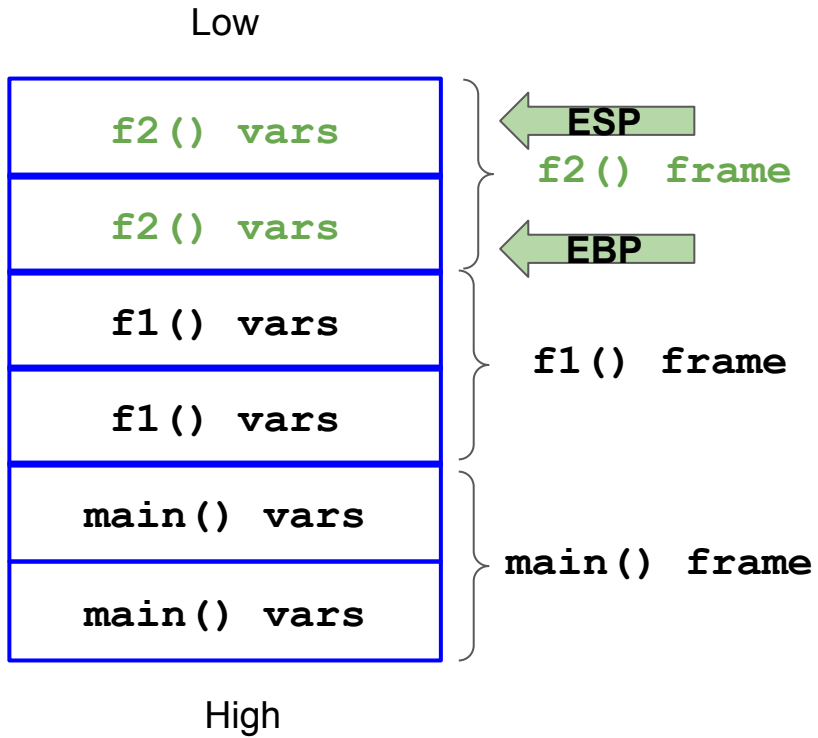
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



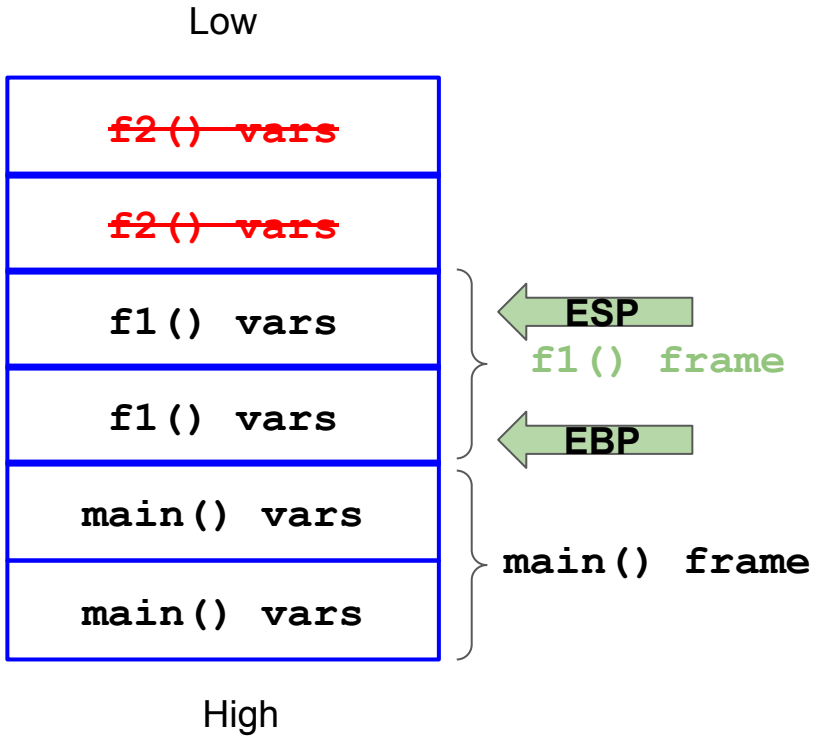
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```



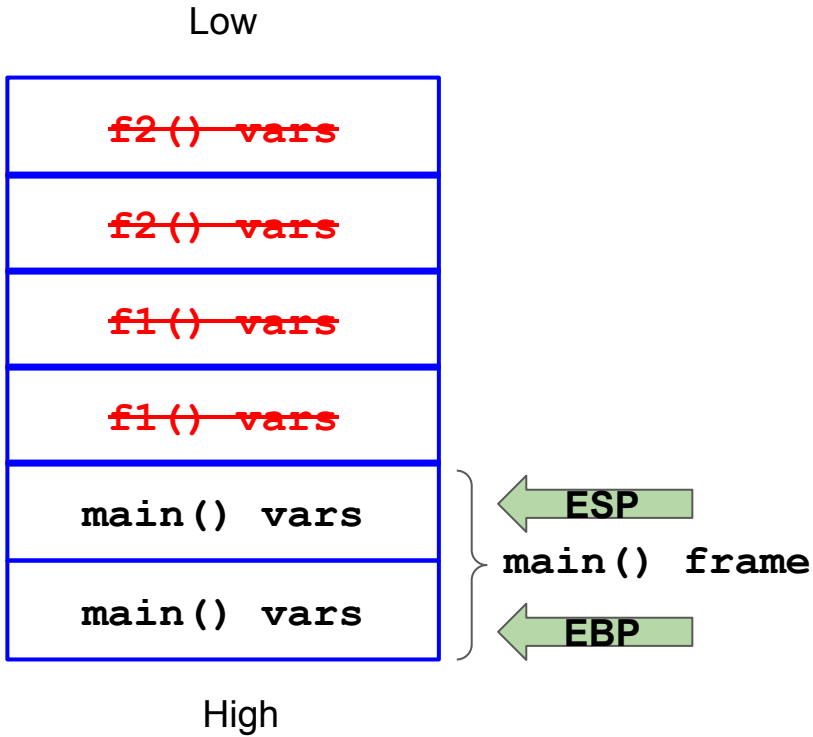
Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
  }
```

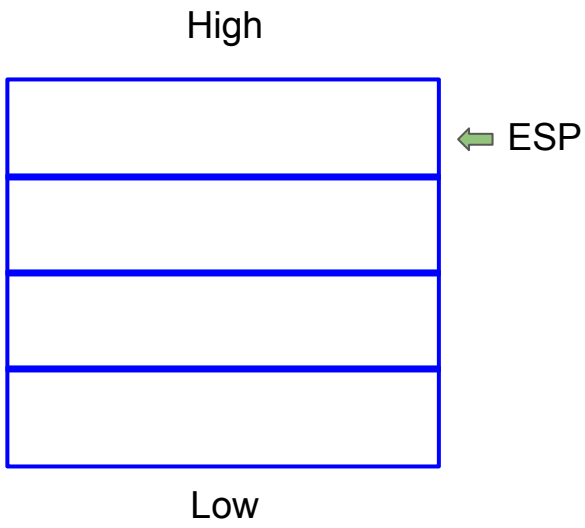


Buffer Overflow

```
void f2(){
    ...
    }
void f1(){
   f2();
    }
void main(){
    ...
   f1();
 }
```



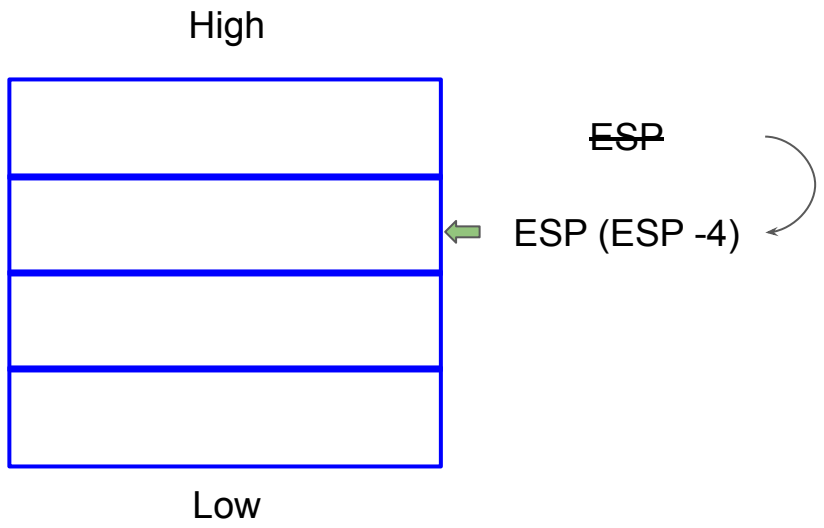
Stack Instructions



Stack Instructions: PUSH

Push $1 ⇦

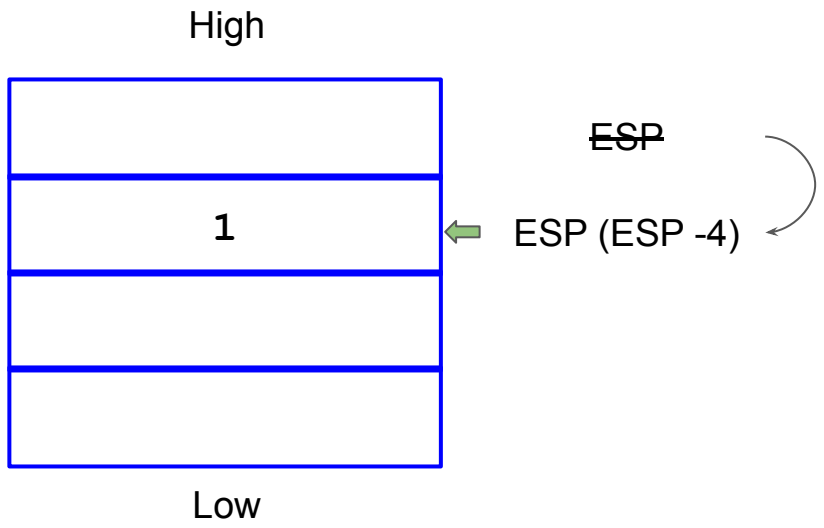
Push $2



Stack Instructions: PUSH

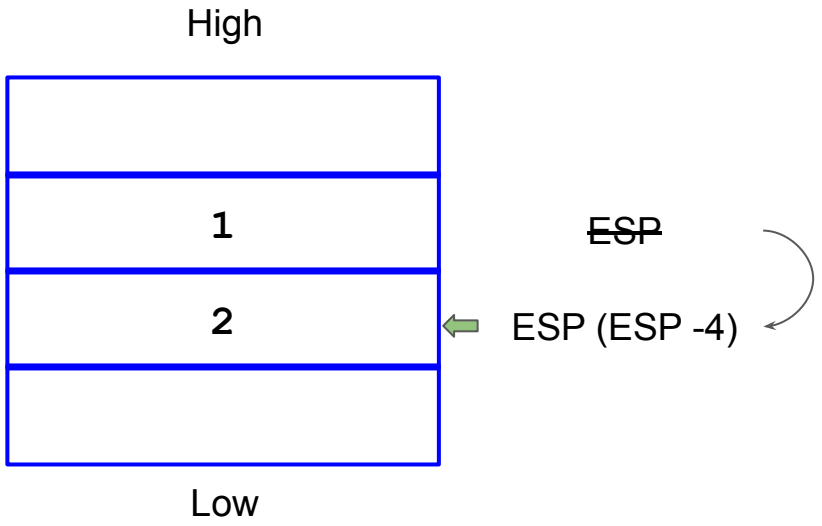
Push $1 ⇦

Push $2



Stack Instructions: PUSH

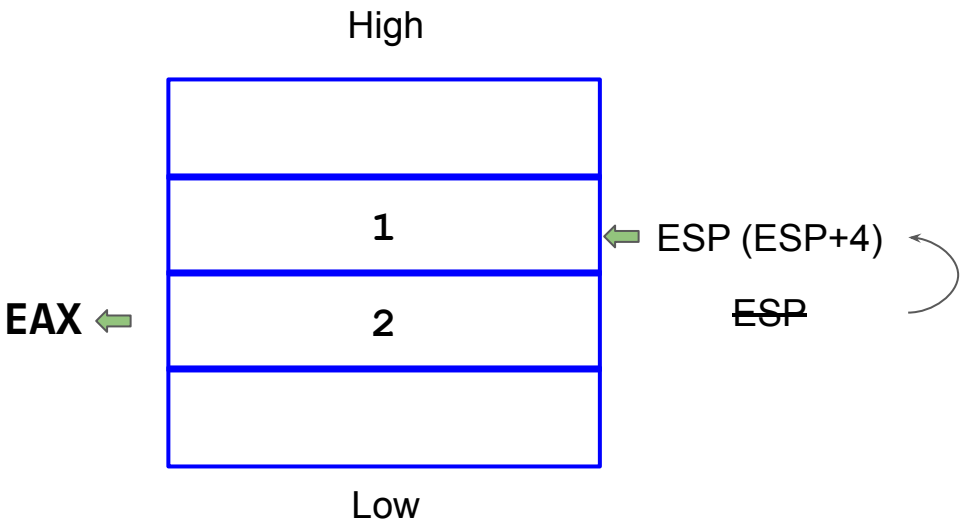
```
Push $1

Push $2
```



Stack Instructions: POP

POP %EAX

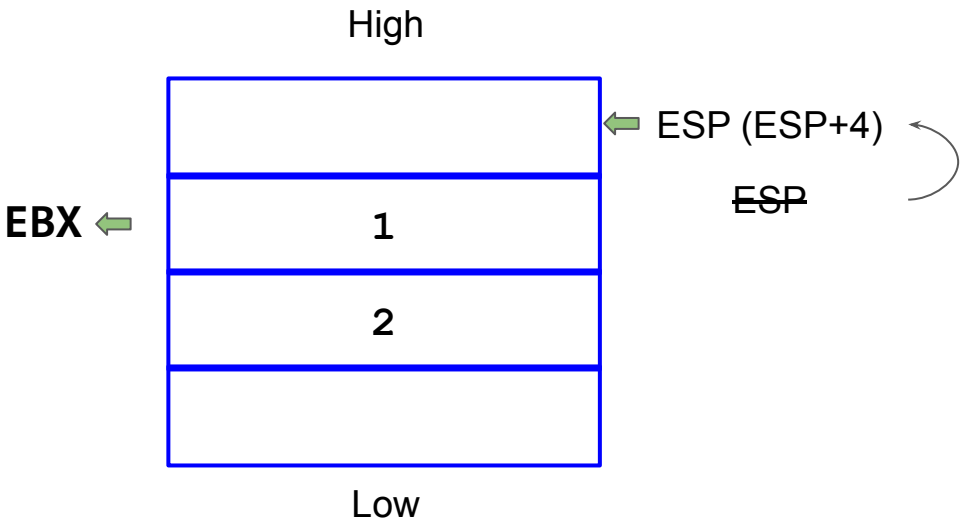
POP %EBX



Stack Instructions: POP

```
POP %EAX
```

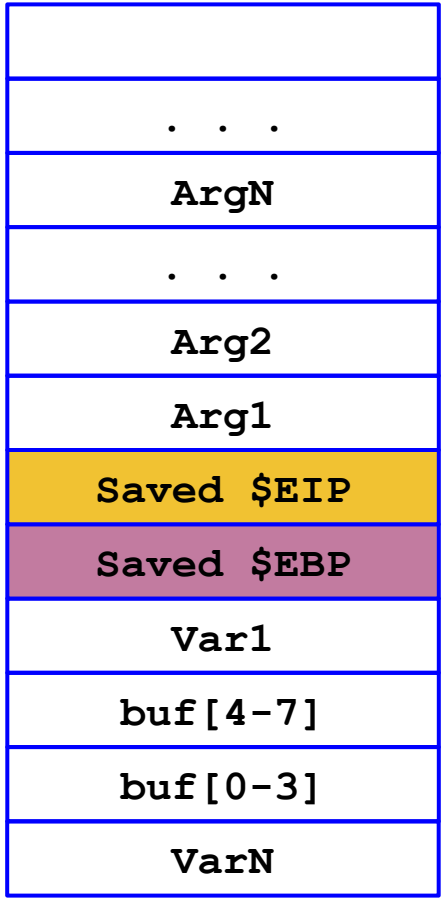
POP %EBX



High addresses (0xC0000000)

```
foo(arg1, arg2,
          ..., argN) {

     var1;
     buf[8];
     ...
     varN;

}
```

MEMORY ALLOCATION



EBP + "(N+1)*4" in hex

EBP+0x12

EBP+0x8

EBP+0x4

EBP

EBP-0x4

EBP-0x8

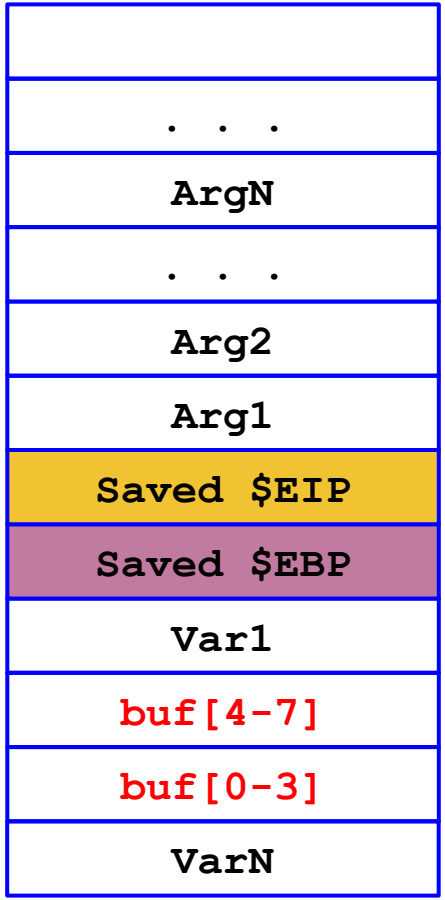
EBP - "N*4" in hex

Low addresses (0xBFFDF000)

High addresses (0xC0000000)

```
foo(arg1, arg2,
          ..., argN) {

     var1;
     buf[8];
     ...
     varN;

}
```

MEMORY ALLOCATION



EBP + "(N+1)*4" in hex

EBP+0x12

EBP+0x8

EBP+0x4

EBP

EBP-0x4

EBP-0x8

EBP - "N*4" in hex

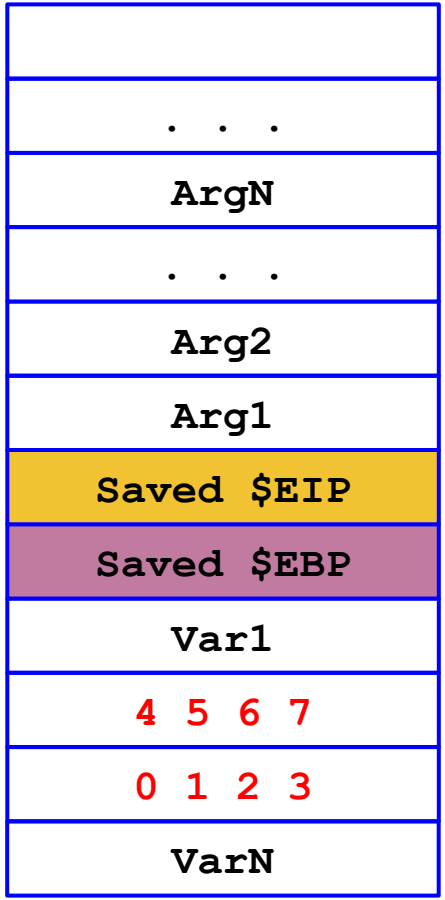
```
{
     ...

     gets(buf);

}
```

Low addresses (0xBFFDF000)

High addresses (0xC0000000)

```
foo(arg1, arg2,
          ..., argN) {

     var1;
     buf[8];
     ...
     varN;

}
```

MEMORY ALLOCATION



EBP + "(N+1)*4" in hex

EBP+0x12

EBP+0x8

EBP+0x4

EBP

EBP-0x4

EBP-0x8

EBP - "N*4" in hex

MEMORY WRITING

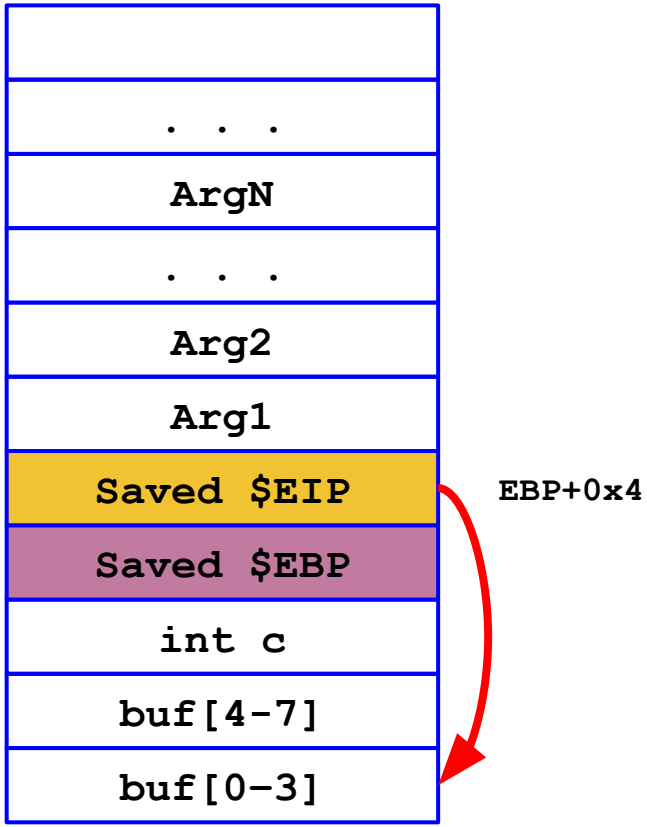
```
{
     ...
     gets(buf);
}
```

Low addresses (0xBFFDF000)

```
$ ./executable-vuln
    XXXXXXXXXXXXXXXXAddressofbuf[]
```

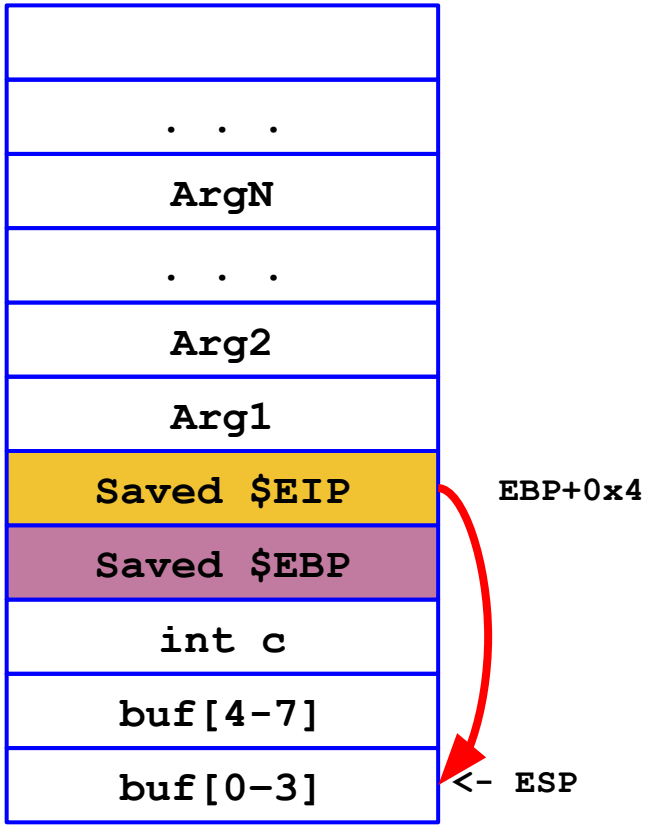
Addressofbuf[]

X X X X
X X X X
X X X X
X X X X



```
$ ./executable-vuln
    validmachinecodeaddressofbuf[]
```

addressofbuf[]
code
hine
dmac
vali



$ man execve

```
int execve(const char *filename, char *const argv[],char *const envp[]);
```

filename = path to binary executable we want to execute

argv[]= array of pointers to strings passed to the new program as its command-line arguments. By convention, the first of these strings (i.e., argv[0]) should contain the filename associated with the file being executed. The argv array must be terminated by a NULL pointer

envp[]= array of pointers to strings, conventionally of the form key=value, which are passed as the environment of the new program. The envp array must be terminated by a NULL pointer

SysCall calling convention

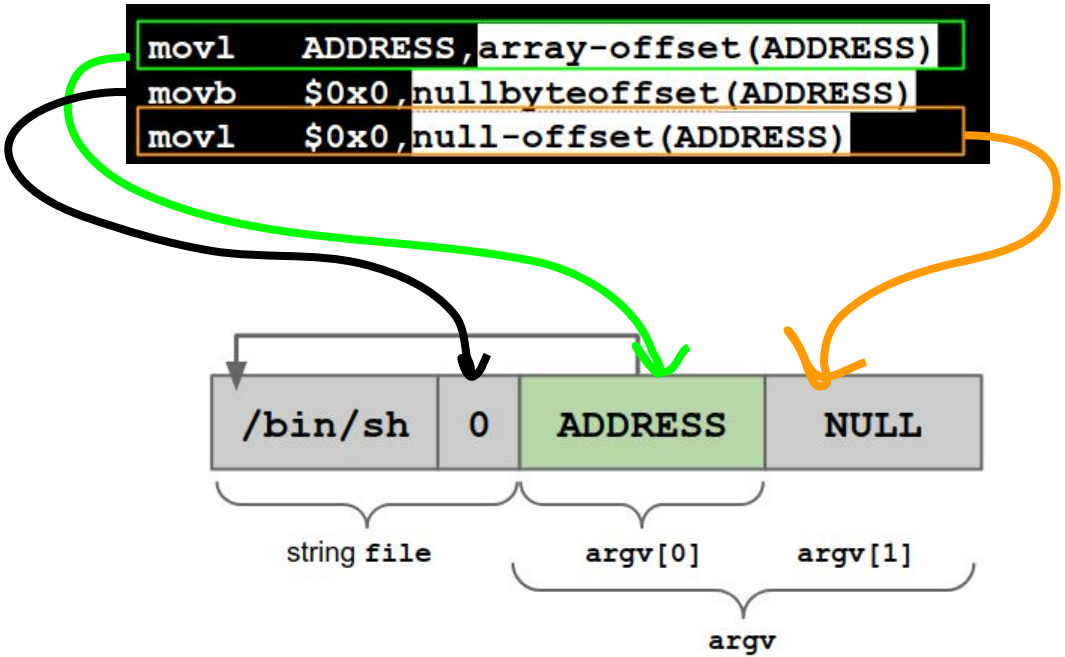
1. `movl $syscall_number, eax`
2. Syscall arguments -> general purpose registers (ebc, ecx, edx)
   a. `mov arg1, %ebx`
   b. `mov arg2, %ecx`
   c. `mov arg3, %edx`
3. `int 0x80` -> Switch to kernel mode
4. Syscall is executed

Disassemble execve

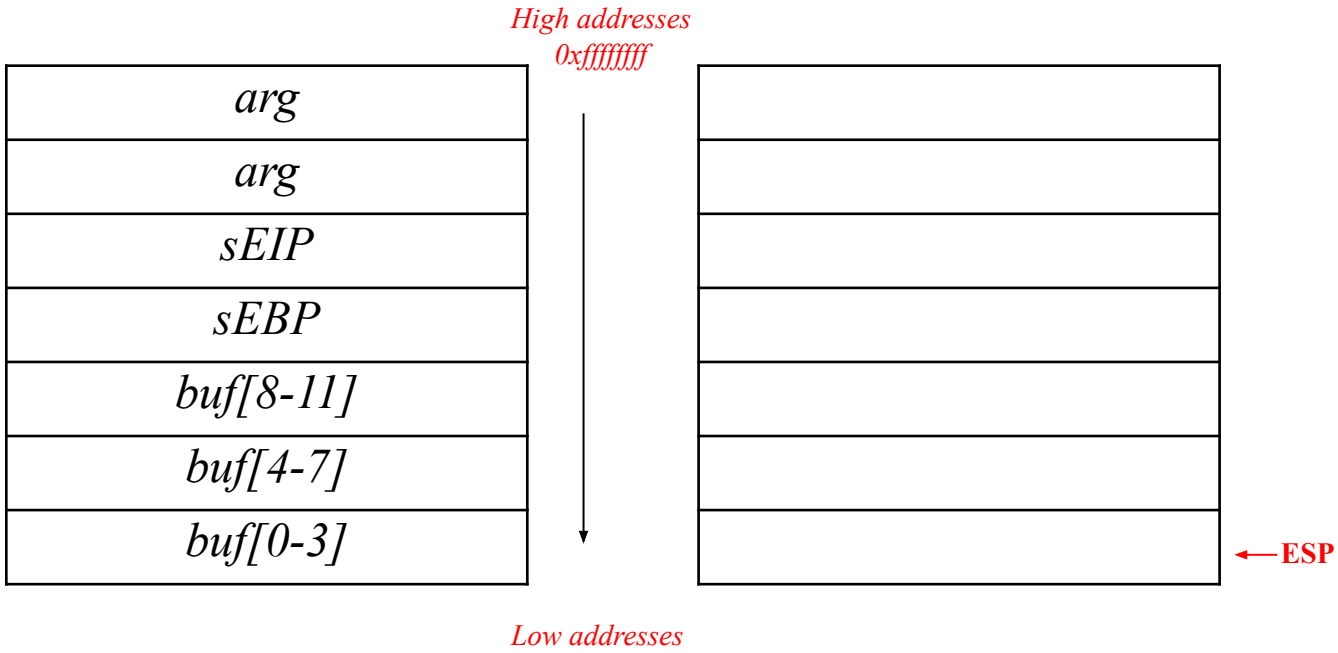
```
int execve(char *file, char *argv[], char *env[])
```

move **$0xb** into EAX registry
move EBP+8 (i.e., ***file**) into EBX
move EBP+12 (i.e., ***argv[0]**) into ECX
move EBP+16 (i.e., ***env[0]**) into EDX
invoke the system call found in EAX

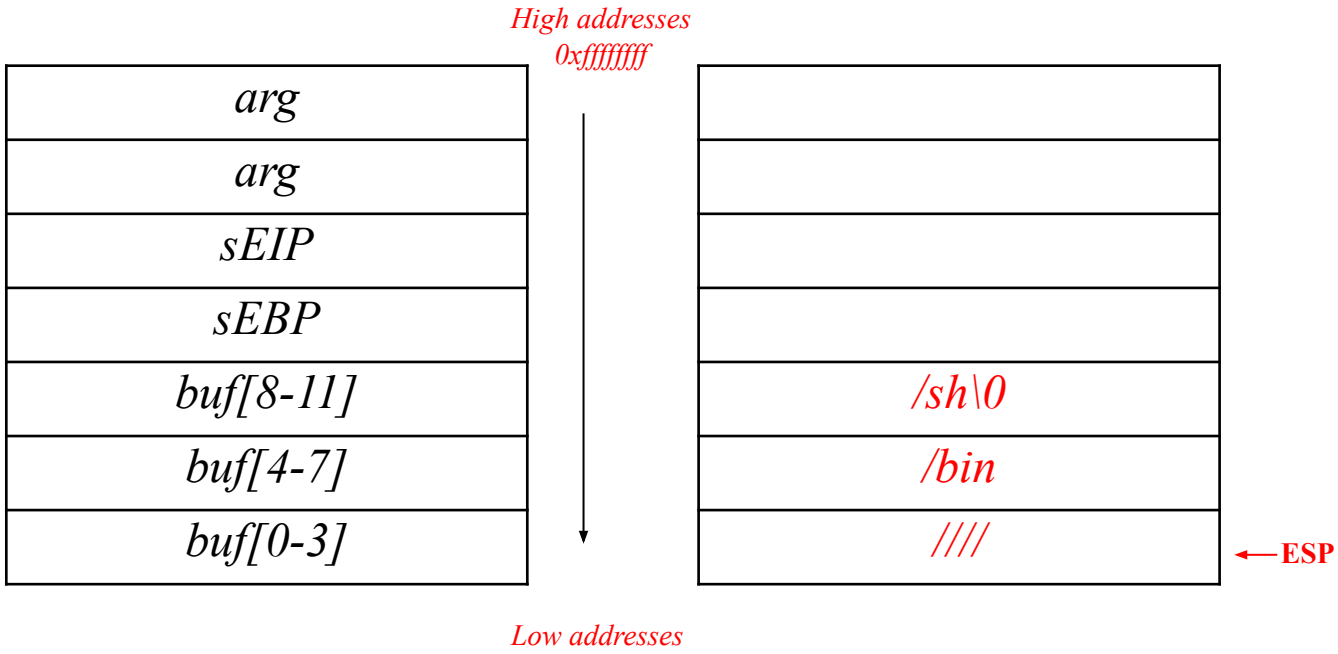
```
(gdb) disassemble  execve
...
movl   $0xb,%eax       //0xb is "execve"
movl   0x8(%ebp),%ebx
movl   0xc(%ebp),%ecx
movl   0x10(%ebp),%edx
int    $0x80
```


movl ADDRESS,array-offset(ADDRESS)
movb $0x0,nullbyteoffset(ADDRESS)
movl $0x0,null-offset(ADDRESS)
/bin/sh
0
ADDRESS
NULL
string file
argv[0]
argv[1]
argv

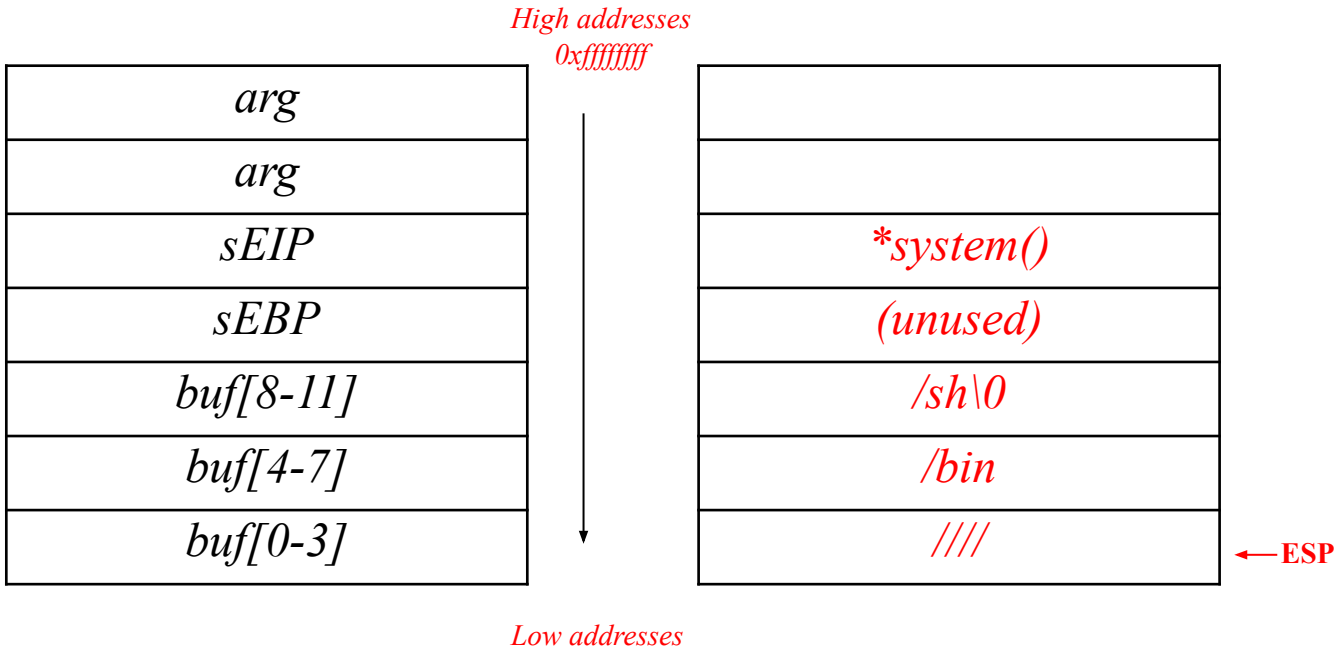
Return to libc `system("/bin/sh")`



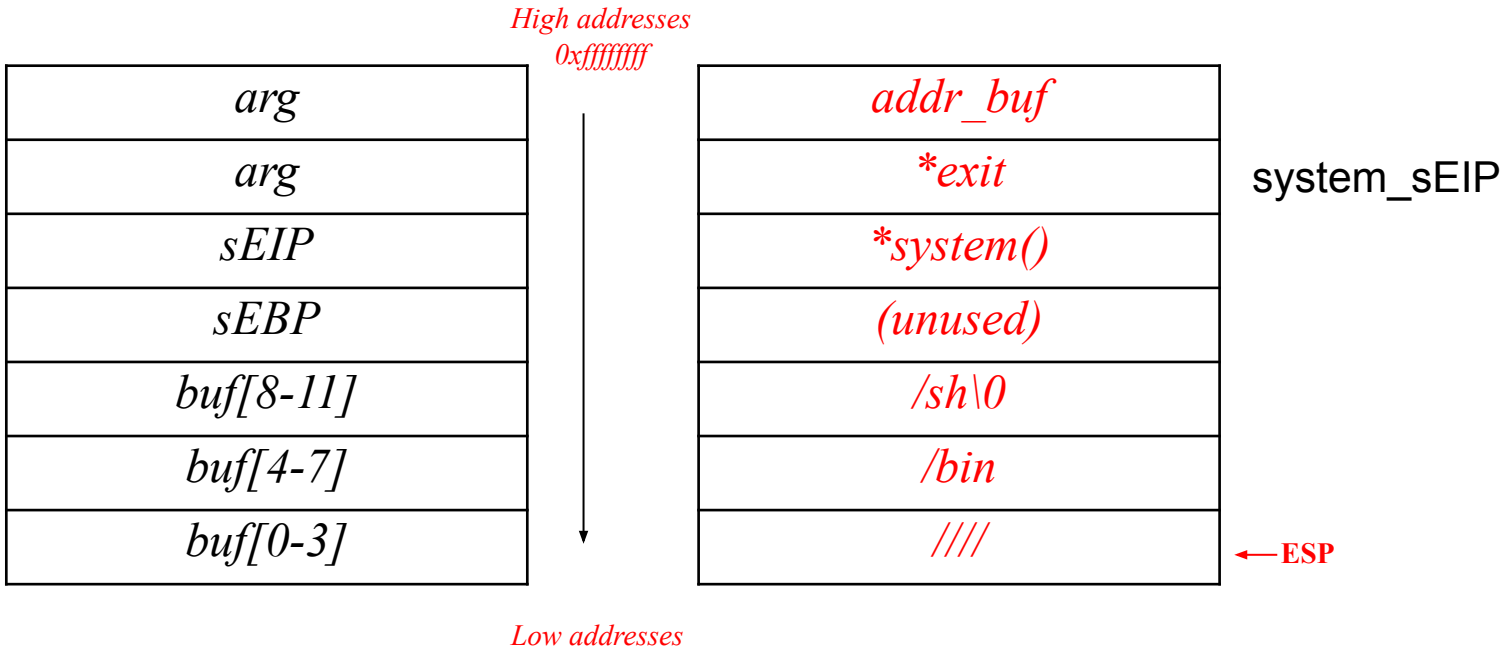
`system("/bin/sh")`



```
system("/bin/sh")
```



```
system("/bin/sh")
```

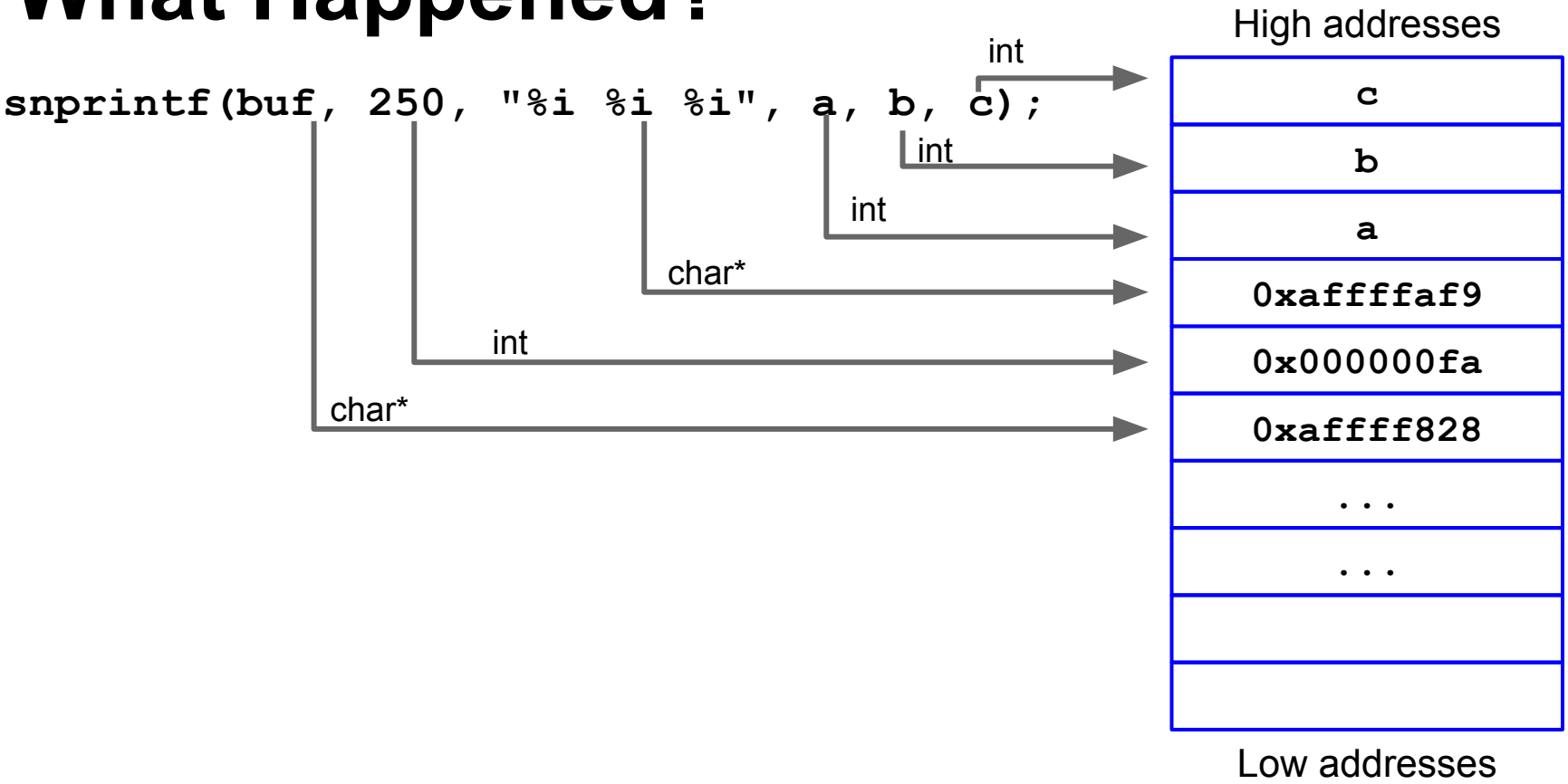


7. Format String Bugs

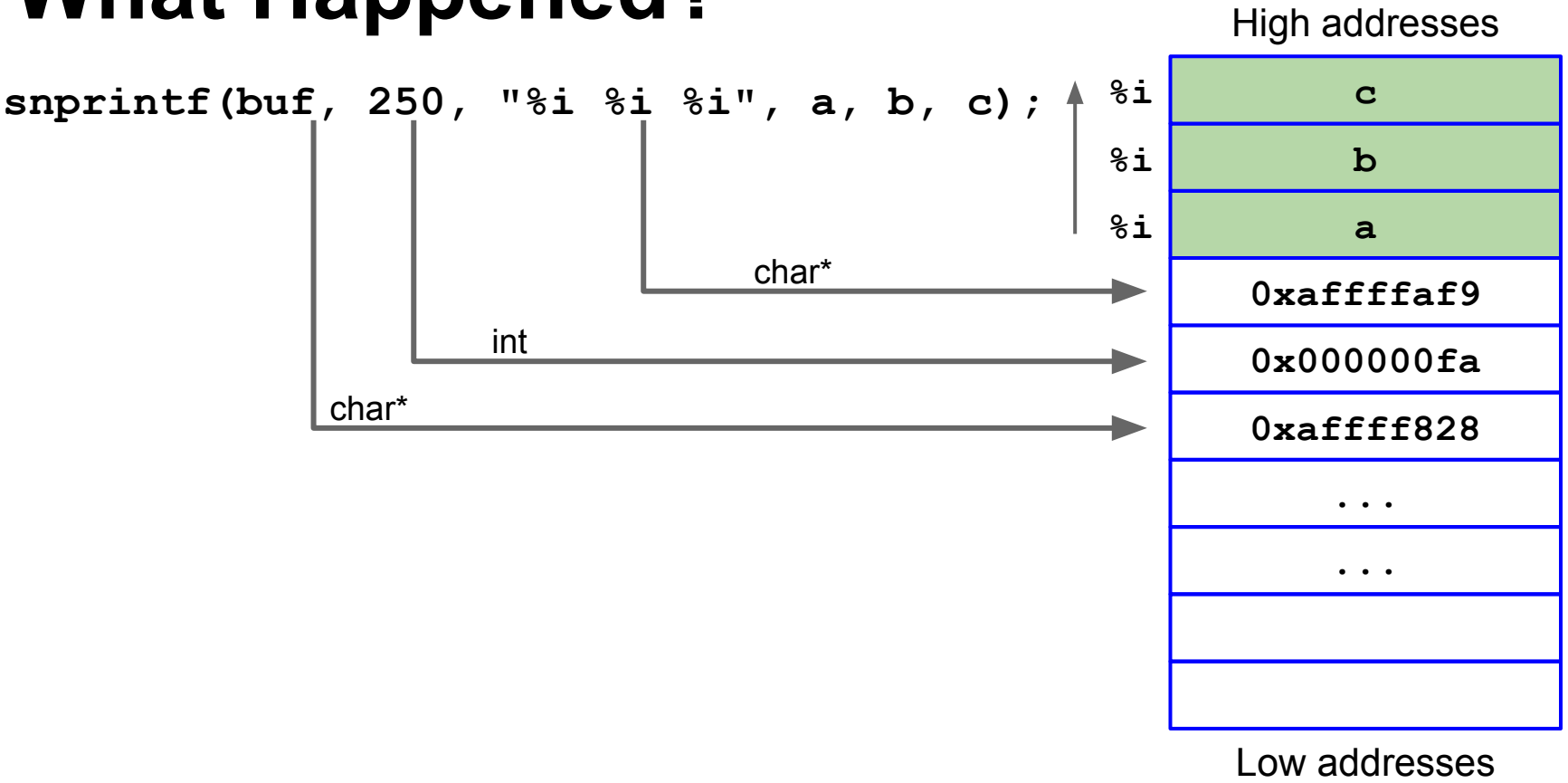
Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

What Happened?



What Happened?

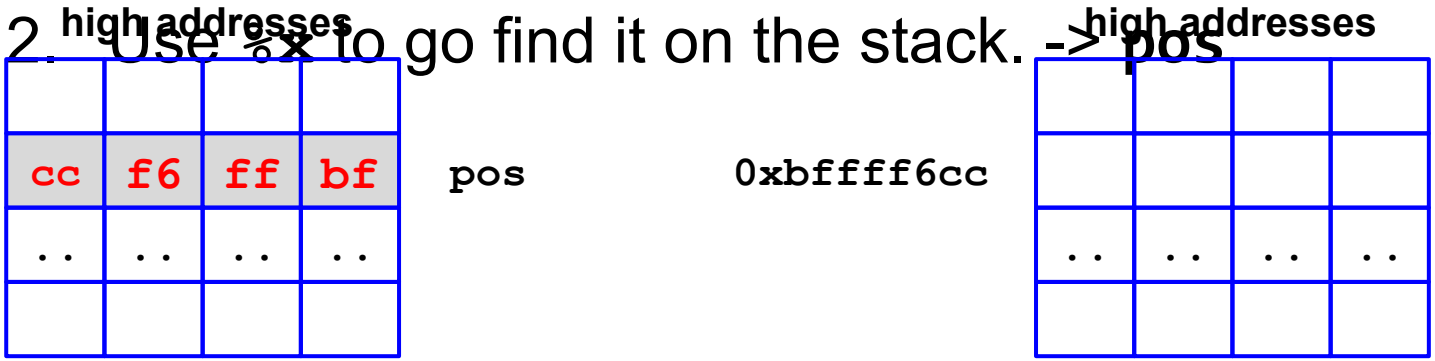


`Target addr = bffff6cc`

1. Put, on the stack, the **`target addr`** of the memory cell to modify.

`Target addr = bffff6cc`

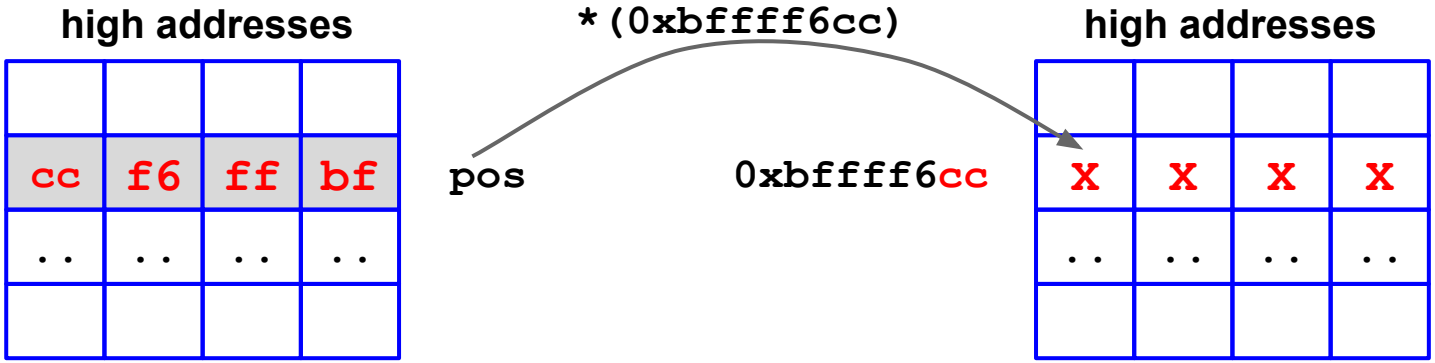
1. Put, on the stack, the **`target addr`** of the memory cell to modify.
2. Use **`%x`** to go find it on the stack. -> **`pos`**



<target>%XXXXc%pos$n

Target addr = bffff6cc

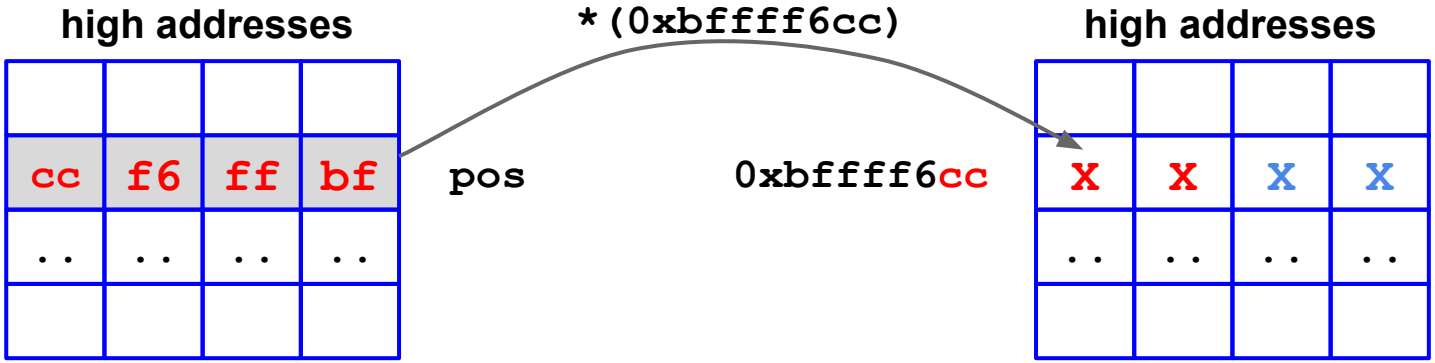
1. Put, on the stack, the **target addr** of the memory cell to modify.
2. Use %x to go find it on the stack.



3. Use **%c%n** to write XXXX in the cell pointed to by **target->** 32 bit address problem

Target addr = bffff6cc

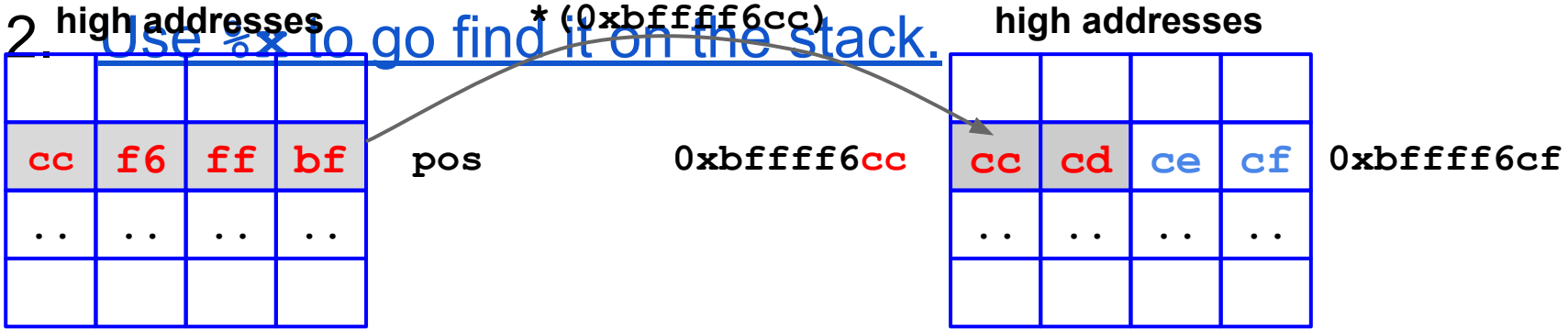
1. Put, on the stack, the **target addr** of the memory cell to modify.
2. Use **%x** to go find it on the stack.



3. Use **%n** to write XX in the cell pointed to by **target**.
4. Use **%n** to write XX in the cell pointed to by **??**.

Target addr = bffff6cc

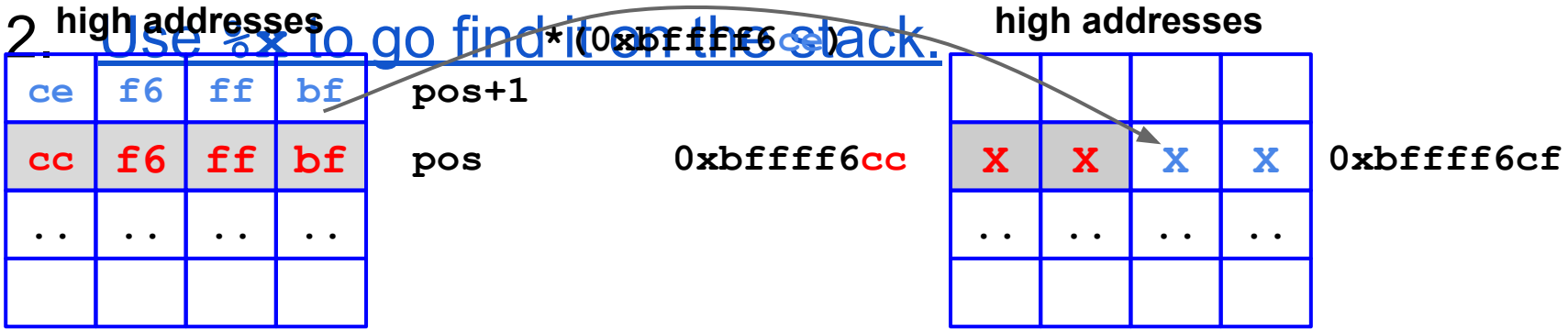
1. Put, on the stack, the **target addr** of the memory cell to modify.
2. Use %x to go find it on the stack.



<target><target+2>%XXc%pos$n%XXc%pos+1$n

Target addr = bffff6cc

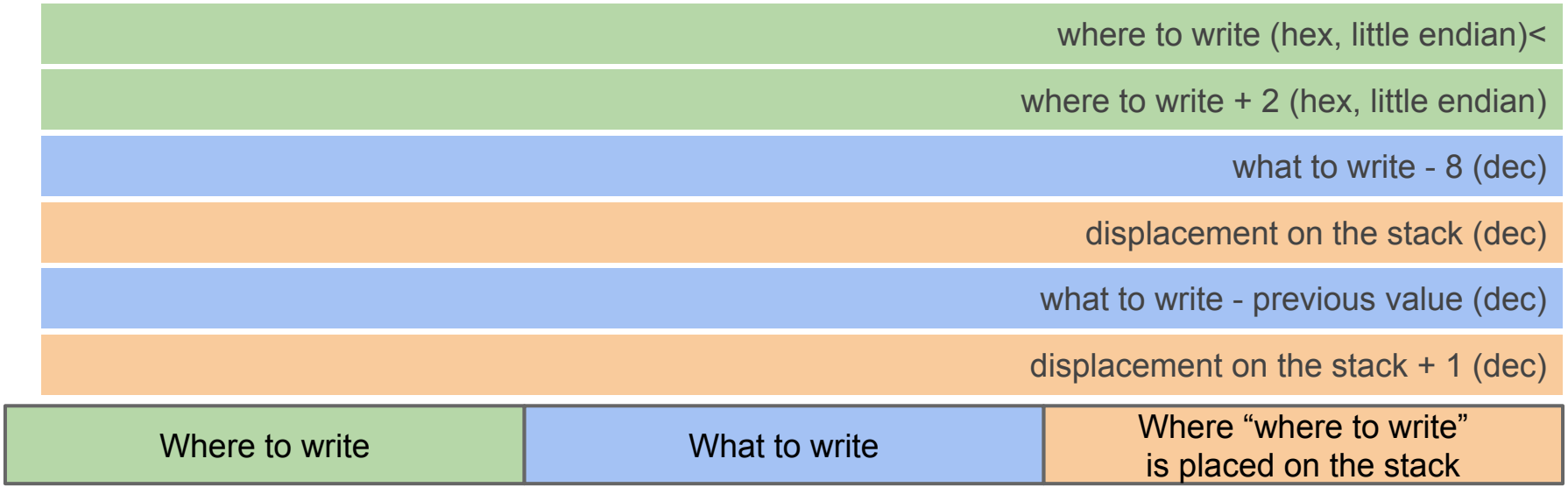
1. Put, on the stack, the **target addr** of the memory cell to modify.
2. Use %x to go find it on the stack.



Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st



Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08 where to write (hex, little endian)<

where to write + 2 (hex, little endian)

what to write - 8 (dec)

displacement on the stack (dec)

what to write - previous value (dec)

displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08 where to write (hex, little endian)<

\x9a\x96\x04\x08 where to write + 2 (hex, little endian)

what to write - 8 (dec)

displacement on the stack (dec)

what to write - previous value (dec)

displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08 where to write (hex, little endian)<

\x9a\x96\x04\x08 where to write + 2 (hex, little endian)

%(7952-8)c what to write - 8 (dec)

displacement on the stack (dec)

what to write - previous value (dec)

displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08	where to write (hex, little endian)<
\x9a\x96\x04\x08	where to write + 2 (hex, little endian)
%(7952-8)c	what to write - 8 (dec)
%<pos>$hn	displacement on the stack (dec)
	what to write - previous value (dec)
	displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08	where to write (hex, little endian)<
\x9a\x96\x04\x08	where to write + 2 (hex, little endian)
%(7952-8)c	what to write - 8 (dec)
%<pos>$hn	displacement on the stack (dec)
%(47083-7952)c	what to write - previous value (dec)
	displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

Example:

Let's write **0xb7eb1f10** to **0x08049698**

0xb7eb = 47083 > 7952 = 0x1f10 ~> 7952 must be written 1st

\x98\x96\x04\x08	where to write (hex, little endian)<
\x9a\x96\x04\x08	where to write + 2 (hex, little endian)
%(7952-8)c	what to write - 8 (dec)
%<pos>$hn	displacement on the stack (dec)
%(47083-7952)c	what to write - previous value (dec)
%<pos+1>$hn	displacement on the stack + 1 (dec)

Where to write	What to write	Where “where to write” is placed on the stack

9. Network Protocol Attacks

Computer Security Courses @ POLIMI

Prof. Carminati & Prof. Zanero

Basic Concept of Domain Name System

DNS Cache Poisoning Attack

Addressing So Far

- Port numbers for applications
- MAC addresses for hardware
- IP addresses for Internet routing

Problems

- Humans are bad at remembering strings of numbers
- Need of a human-friendly naming system

Requirements for Naming System

- As short as possible
- Easy to memorize (i.e., not arbitrary)
- Unique
- Customizable
- Reflect organizational structure (Hierarchy)
- Quickly translate to and from the existing, “computer-friendly” addressing systems
- Address specific resources/services

Domain Names System (DNS) (1/2)

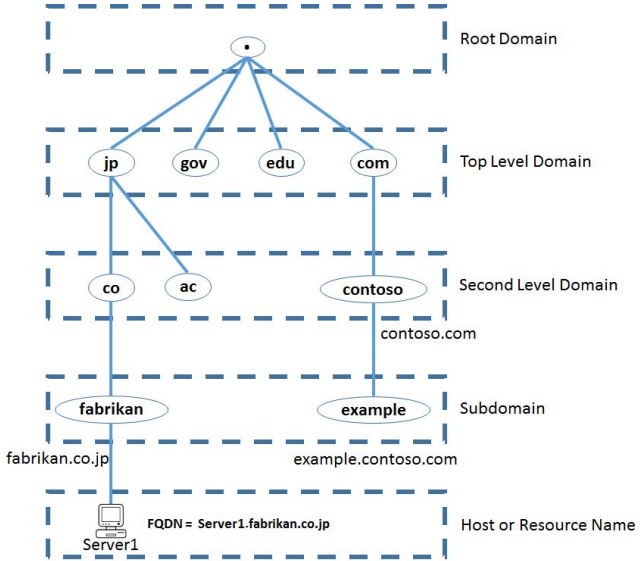
- Maps/Translates “domain names” to numerical IP addresses
  - You can type www.google.com into the browser, and the browser will know to go to 173.194.33.179
- But how might this be done?
  - Some sort of hash (not really practical)
  - A file of all of the mappings (not really practical)

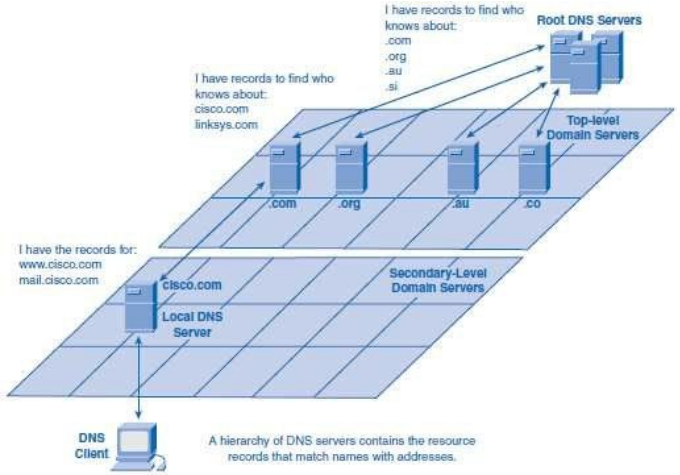
Domain Names System (2/2)

- **Distributed Database**
- **Hierarchy** of servers that provide the mappings
  - Each server keeps a small cache of the mappings
- Based on UDP (Port 53)
- Messages are not authenticated.
- When a domain name is used/requested and isn't in the cache, the system queries a DNS server

Hierarchical DNS Servers

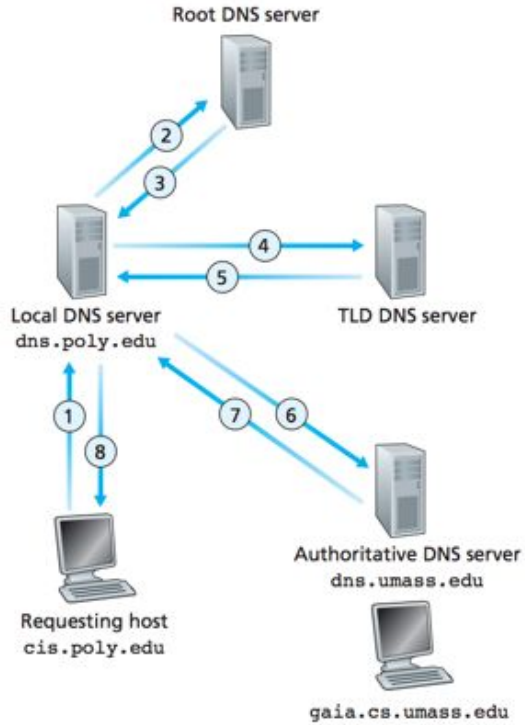
A hierarchy of DNS servers that contains the resource records to match DN with IP





Resolving a Domain Name (1/2)

- If I type sports.polimi.com, what happens?
  - Check /etc/hosts
  - Check DNS cache
  - Check local DNS server
  - Go through the hierarchy:
    - Ask . DNS root server
    - Ask .com TLD/SLD (Top/Second Level Domain) server
    - Ask the Authoritative polimi.com's NS
  - Send HTTP request to the IP address obtained



Basic Concept of Dynamic Host Configuration Protocol

DHCP Poisoning Attack

Dynamic Host Configuration Protocol (DHCP)

Protocol that dynamically assigns IP addresses (and network parameters) to each device in a network:

- It automatically assigns a new IP address when a computer is plugged into the network

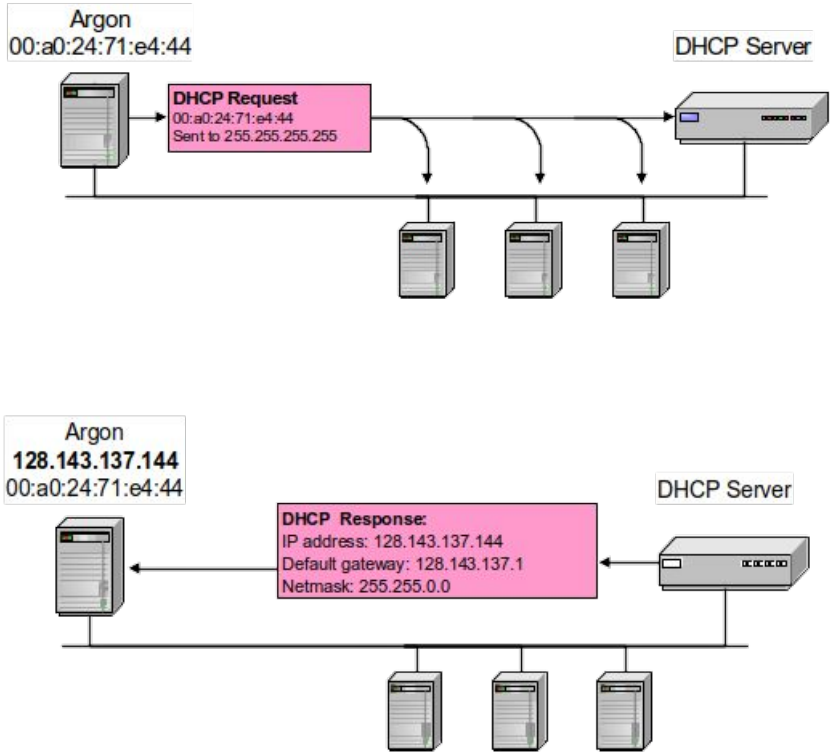
It allows network administrators to supervise and distribute configuration parameters for network hosts from a central point:

- IP address
- Router
- Subnet Mask
-

Limitations of DHCP

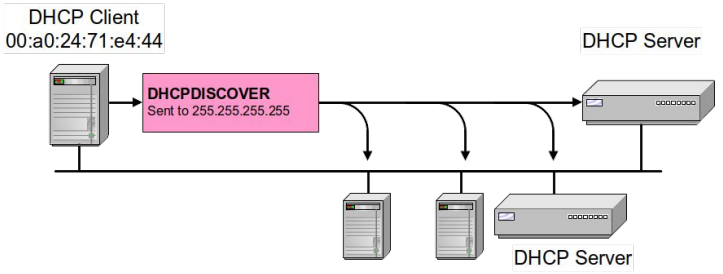
- Again. DHCP is **not authenticated** (not for performance reasons).
- Still based on UDP.
- Some machines on the network must have static addresses (e.g., servers and routers)
- DHCP server must run continually: must be available at all times when clients need IP access.
    - When DHCP server is unavailable, client is unable to access enterprises network.

DHCP Interaction (Simplified)

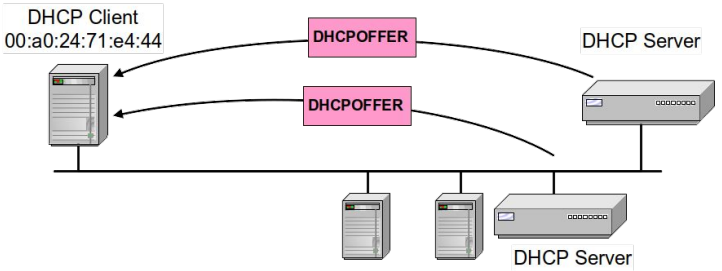


DHCP Operation (1/3)

- **DCHP DISCOVER**



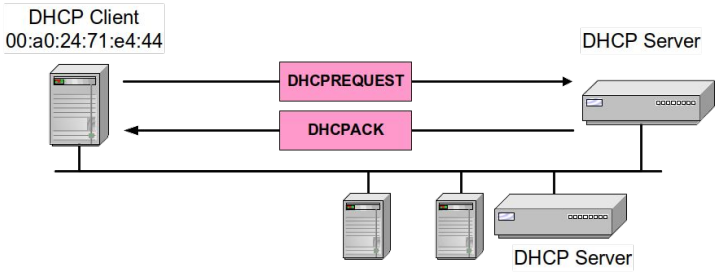
- **DCHP OFFER**



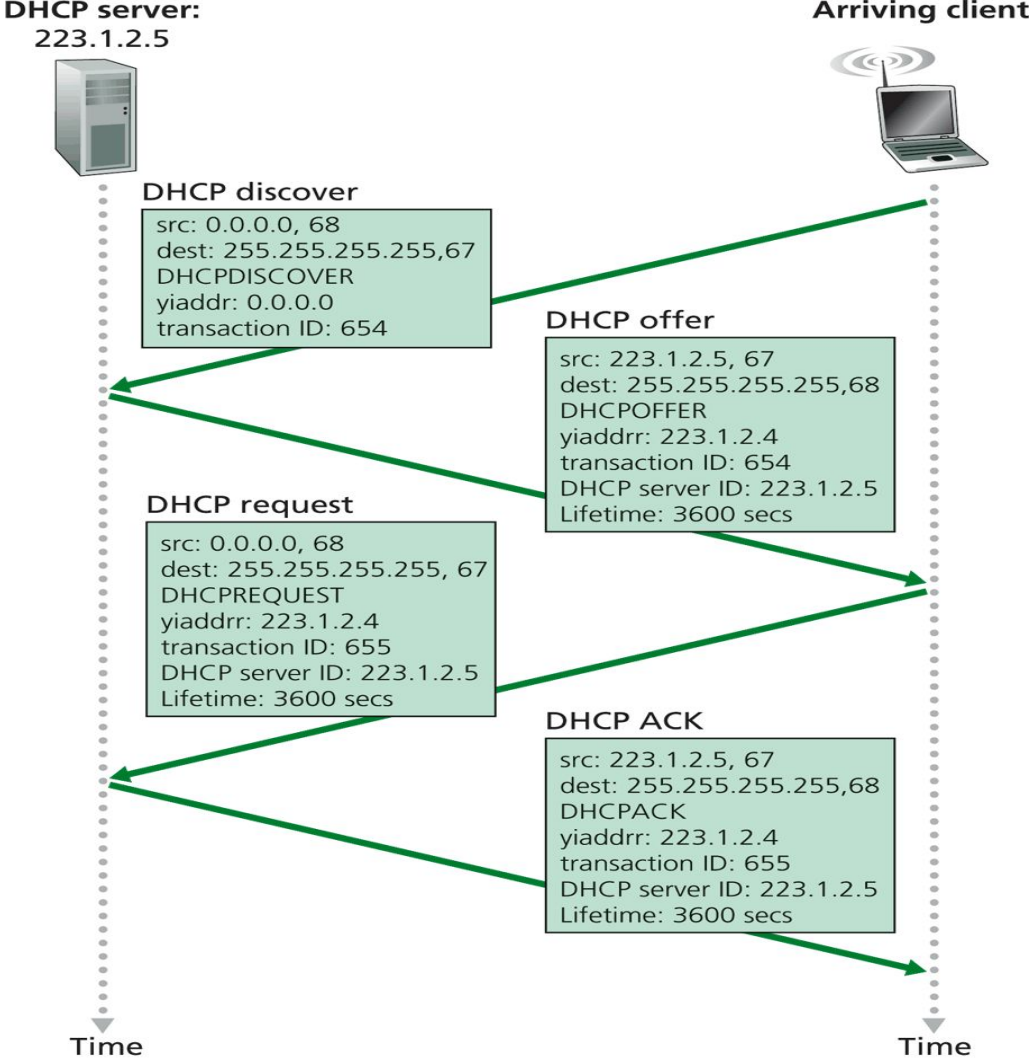
DHCP Operation (2/3)

DCHP DISCOVER

- At this time, the DHCP client can start to use the IP address



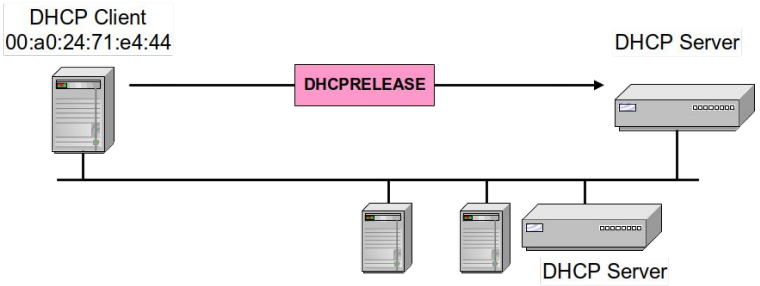
If DHCP server sends **DHCP ACK**, then address is assigned.


DHCP server:
223.1.2.5
Arriving client
DHCP discover
src: 0.0.0.0, 68
dest: 255.255.255.255,67
DHCPDISCOVER
yiaddr: 0.0.0.0
transaction ID: 654
DHCP offer
src: 223.1.2.5, 67
dest: 255.255.255.255,68
DHCPOFFER
yiaddrr: 223.1.2.4
transaction ID: 654
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
DHCP request
src: 0.0.0.0, 68
dest: 255.255.255.255, 67
DHCPREQUEST
yiaddrr: 223.1.2.4
transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
DHCP ACK
src: 223.1.2.5, 67
dest: 255.255.255.255,68
DHCPACK
yiaddrr: 223.1.2.4
transaction ID: 655
DHCP server ID: 223.1.2.5
Lifetime: 3600 secs
Time
Time

DHCP Operation (3/3)

DCHP RELEASE

The DHCP client releases the IP address



Security problem (1/2)

DHCP is an **unauthenticated protocol**

- When connecting to a network, the user is not required to provide credentials to obtain a lease
- Malicious users with physical access to the DHCP-enabled network can instigate a denial-of-service attack on DHCP servers by requesting many leases from the server, thereby depleting the number of leases that are available to other DHCP clients.

Basic Concept of Internet Control Message Protocol (ICMP)

ICMP Redirect Attack

Internet Control Message Protocol

ICMP is used to send debugging information and error reports between hosts, routers and other network devices at IP level.

ICMP messages can be:

- Requests
- Responses
- Error messages

ICMP Messages

- Address mask request/reply:
  - used by diskless systems to obtain the network mask at boot time.
- Timestamp request/reply:
  - used to synchronize clocks.
- Source quench:
  - used to inform about traffic overloads.
- Parameter problem:
  - used to inform about errors in the IP datagram fields.
- Time exceeded:
  - used to report expired datagrams (TTL = 0).
- Echo request/reply:
  - used to test connectivity (ping).

ICMP Echo Request/Reply

Used by the ping program (return to Ping of Death)

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.100 : 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=1.049 msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=660 usec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=597 usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=548 usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=601 usec
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=592 usec
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=547 usec

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.547/0.656/1.049/0.165 ms
```

ICMP Messages

- Address mask request/reply:
  - used by diskless systems to obtain the network mask at boot time.
- Timestamp request/reply:
  - used to synchronize clocks.
- Source quench:
  - used to inform about traffic overloads.
- Parameter problem:
  - used to inform about errors in the IP datagram fields.
- Echo request/reply:
  - used to test connectivity (ping).
- Time exceeded:
  - used to report expired datagrams (TTL = 0).
- Redirect:
  - used to inform hosts about better routes (gateways).
- Destination unreachable:
  - used to inform a host of the impossibility to deliver traffic to a specific destination

ICMP Messages

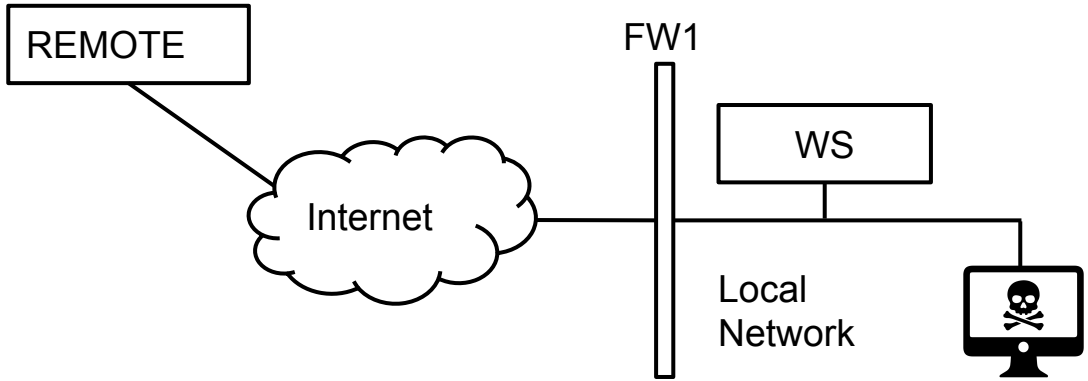
- Address mask request/reply:
  - used by diskless systems to obtain the network mask at boot time.
- Timestamp request/reply:
  - used to synchronize clocks.
- Source quench:
  - used to inform about traffic overloads.
- Parameter problem:
  - used to inform about errors in the IP datagram fields.
- Echo request/reply:
  - used to test connectivity (ping).
- Time exceeded:
  - used to report expired datagrams (TTL = 0).
- **Redirect:**
  - **used to inform hosts about better routes (gateways).**
- Destination unreachable:
  - used to inform a host of the impossibility to deliver traffic to a specific destination

Route Change Requests

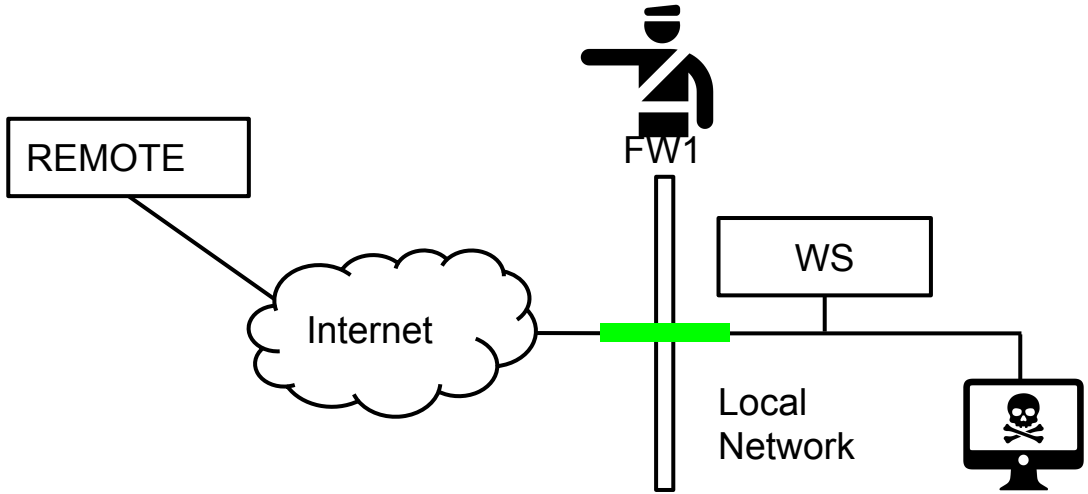
- Routers (not hosts) are responsible for keeping routing information up-to-date.
- Routers are assumed to discover best routes for every destination.
- Hosts begin with minimal routing information and learn new routes from routers.
- A host may boot up knowing the address of only one router – but that may not be the best route.

10. Secure Network Architectures

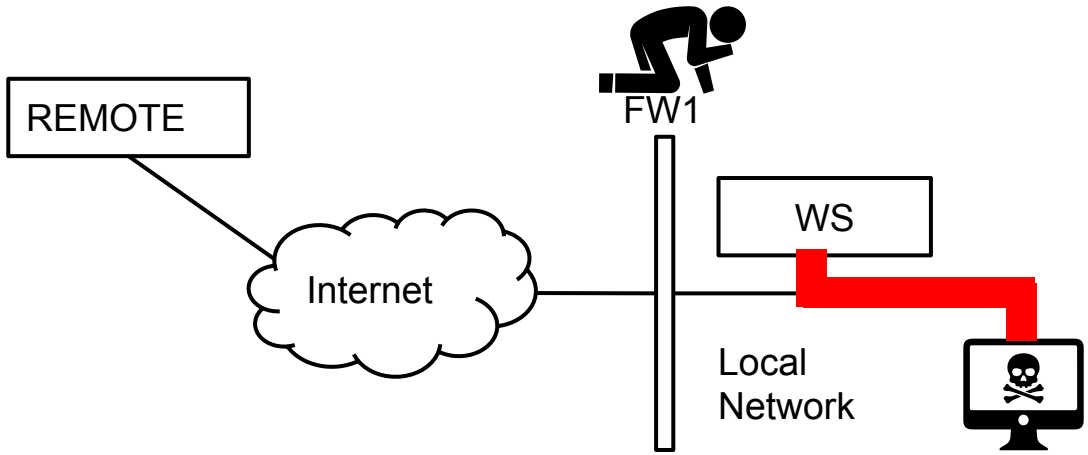
Insider Attacks/Unchecked Paths



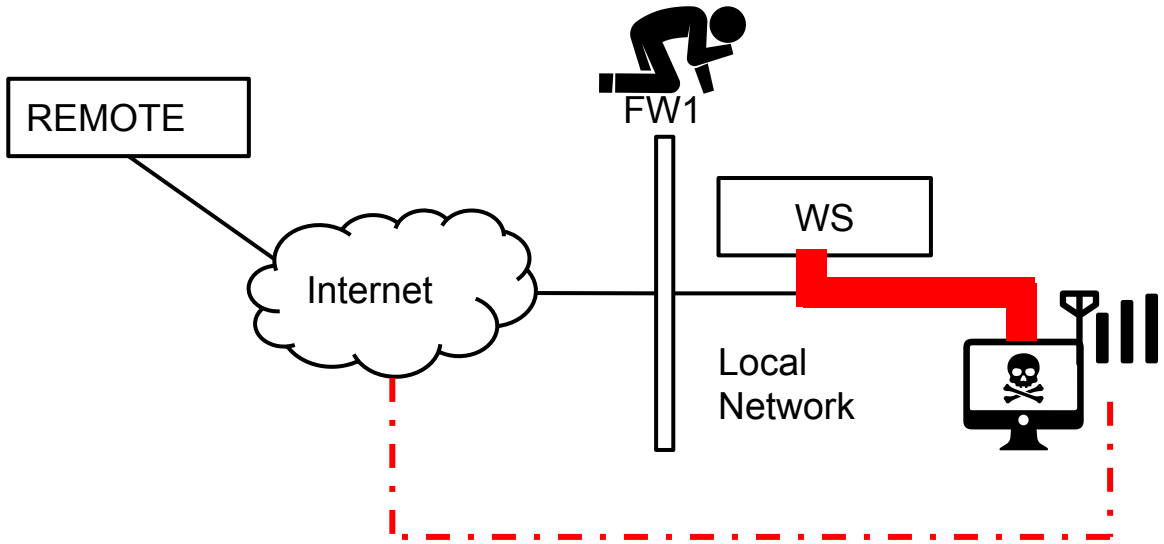
Insider Attacks/Unchecked Paths



Insider Attacks/Unchecked Paths

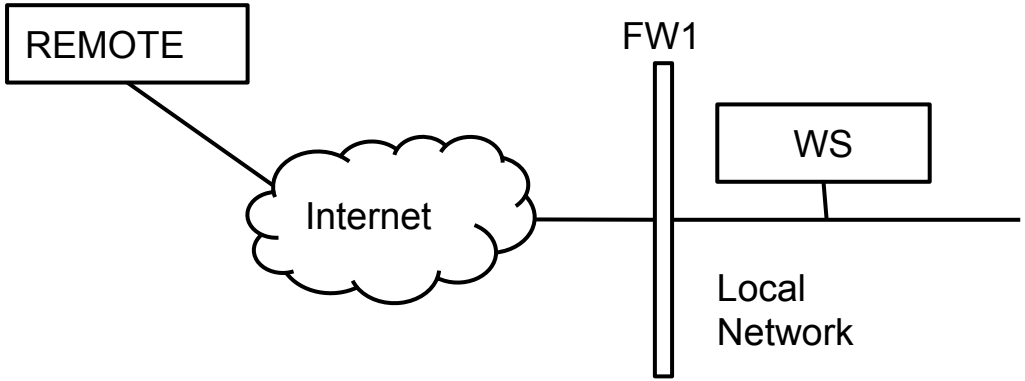


Insider Attacks/Unchecked Paths



Write the firewall rules, assuming firewalls to be <u>stateless</u> packet filters

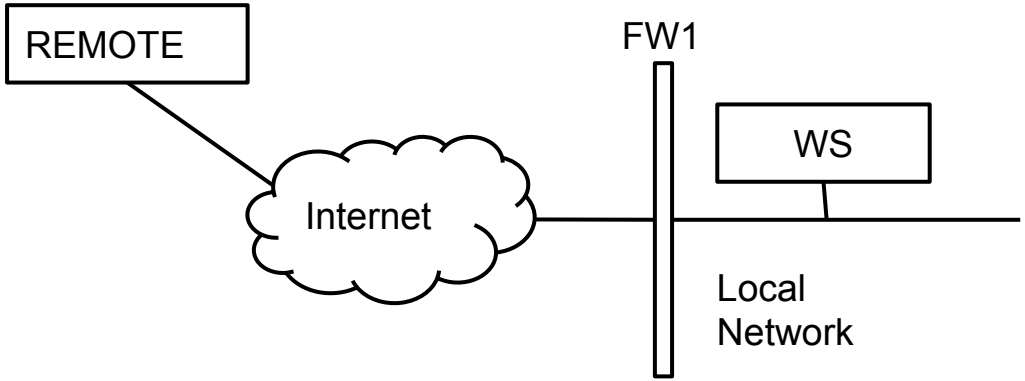
- The WS must be reachable from the Web on Port 80.



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)

Write the firewall rules, assuming firewalls to be <u>stateless</u> packet filters

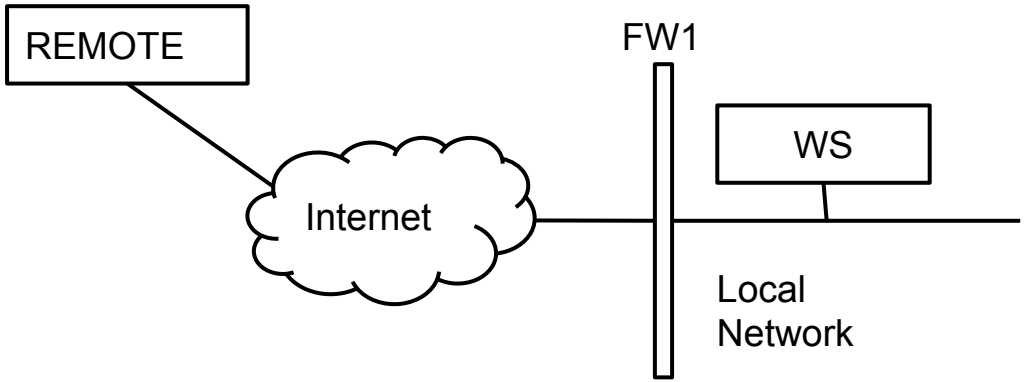
- The WS must be reachable from the Web on Port 80.



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*

Write the firewall rules, assuming firewalls to be <u>stateless</u> packet filters

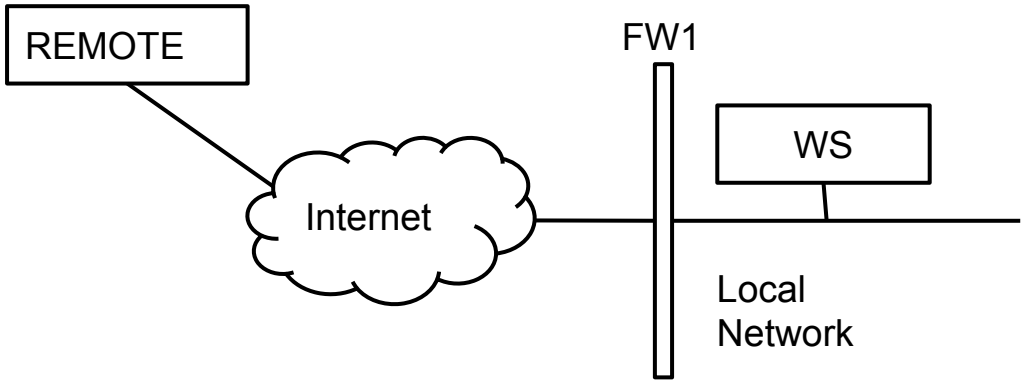
- The WS must be reachable from the Web on Port 80.



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*

Write the firewall rules, assuming firewalls to be <u>stateless</u> packet filters

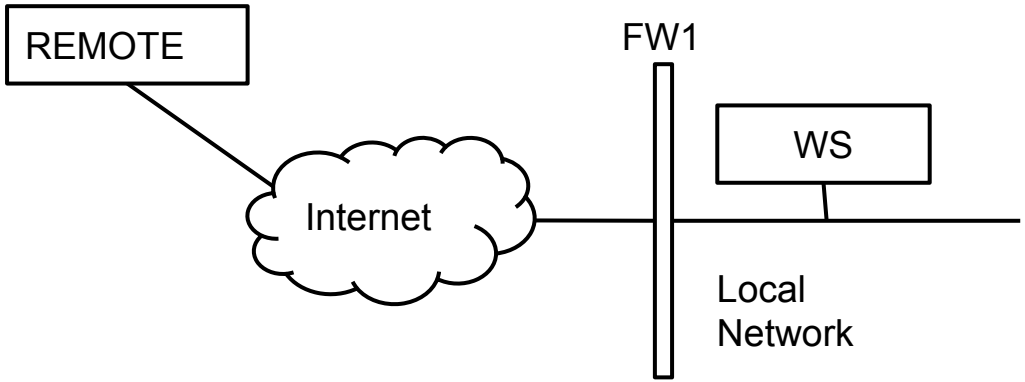
- The WS must be reachable from the Web on Port 80.



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*WWW -> Local*	*WS_IP*	*80*	*ALLOW*	*Allow incoming connection to the WS on PORT 80*

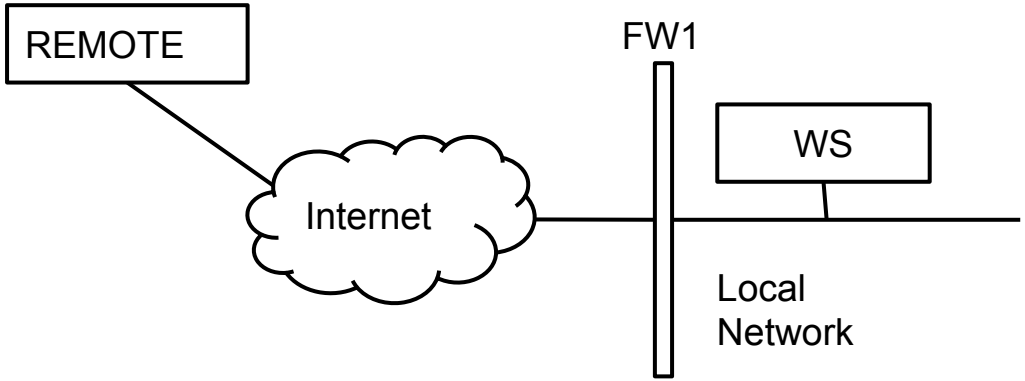
Write the firewall rules, assuming firewalls to be <u>stateless</u> packet filters

- The WS must be reachable from the Web on Port 80.



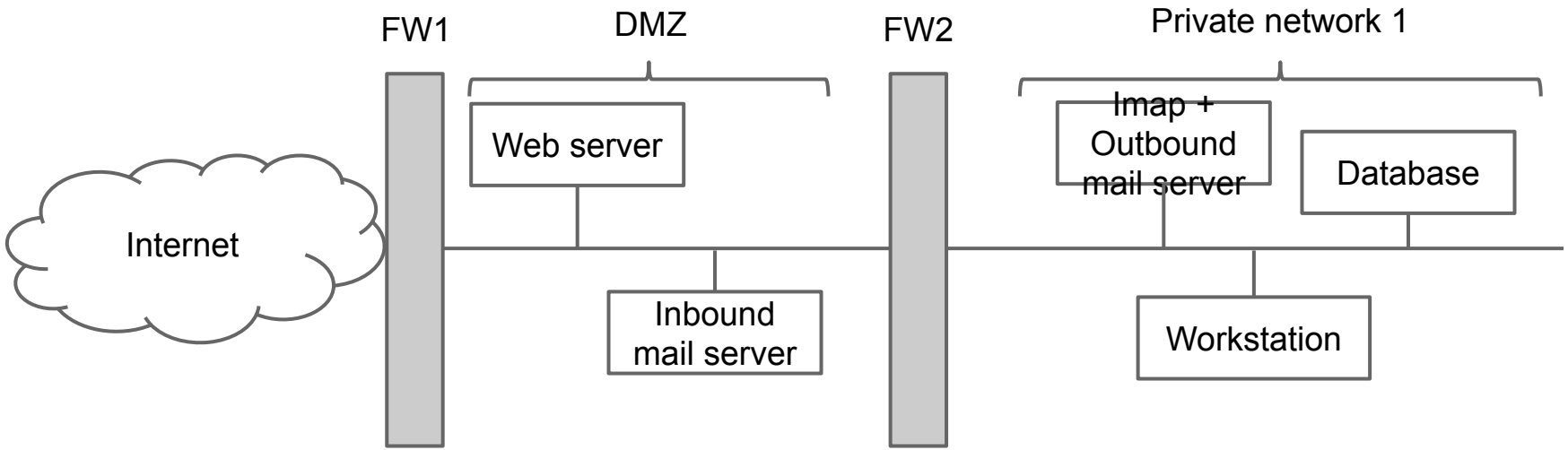
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*WWW -> Local*	*WS_IP*	*80*	*ALLOW*	*Allow incoming connection to the WS on PORT 80*
FW1	*WS_IP*	*80*	*Local -> WWW*	*ANY*	*ANY*	*ALLOW*	*Allow outgoing connection from the WS*

Write the firewall rules, assuming firewalls to be stateful packet filters

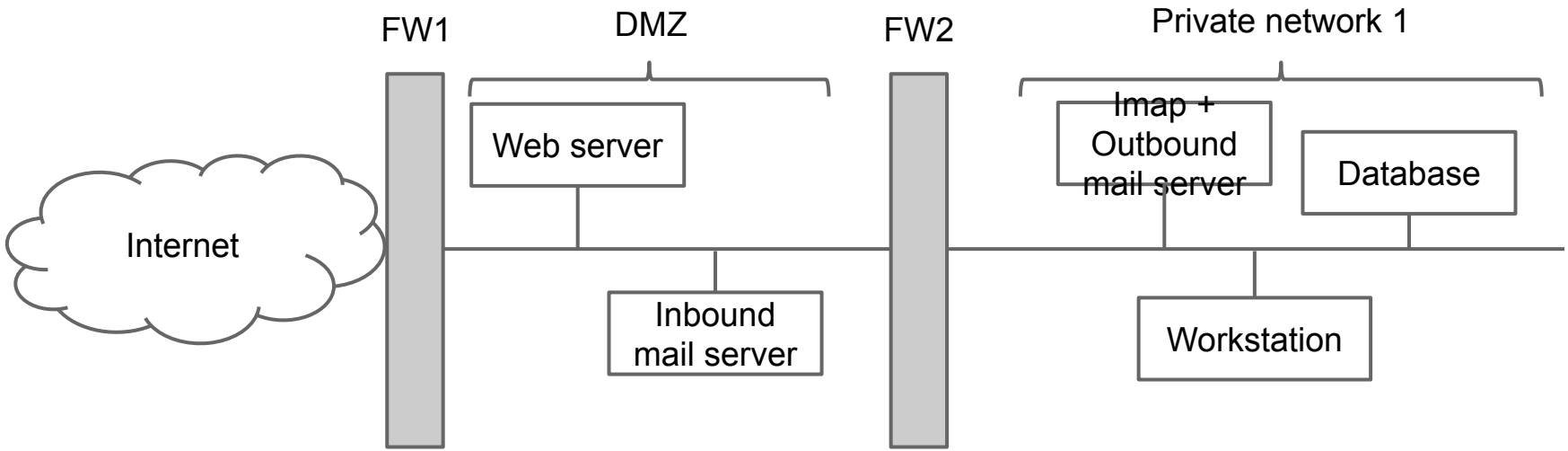


Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*WWW -> Local*	*WS_IP*	*80*	*ALLOW*	*Allow incoming connection to the WS on PORT 80*
~~*FW1*~~	~~*WS_IP*~~	~~*80*~~	~~*Local -> WWW*~~	~~*ANY*~~	~~*ANY*~~	~~*ALLOW*~~	~~*Allow outgoing connection from the WS*~~

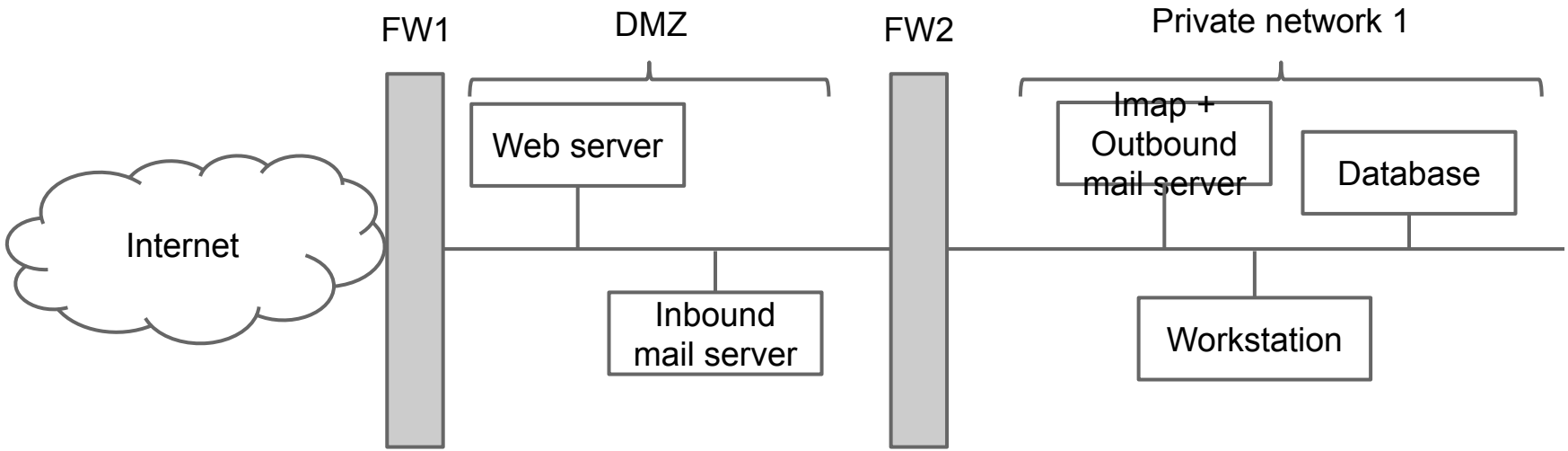
NO NEED TO ADD THE RESPONSE RULE



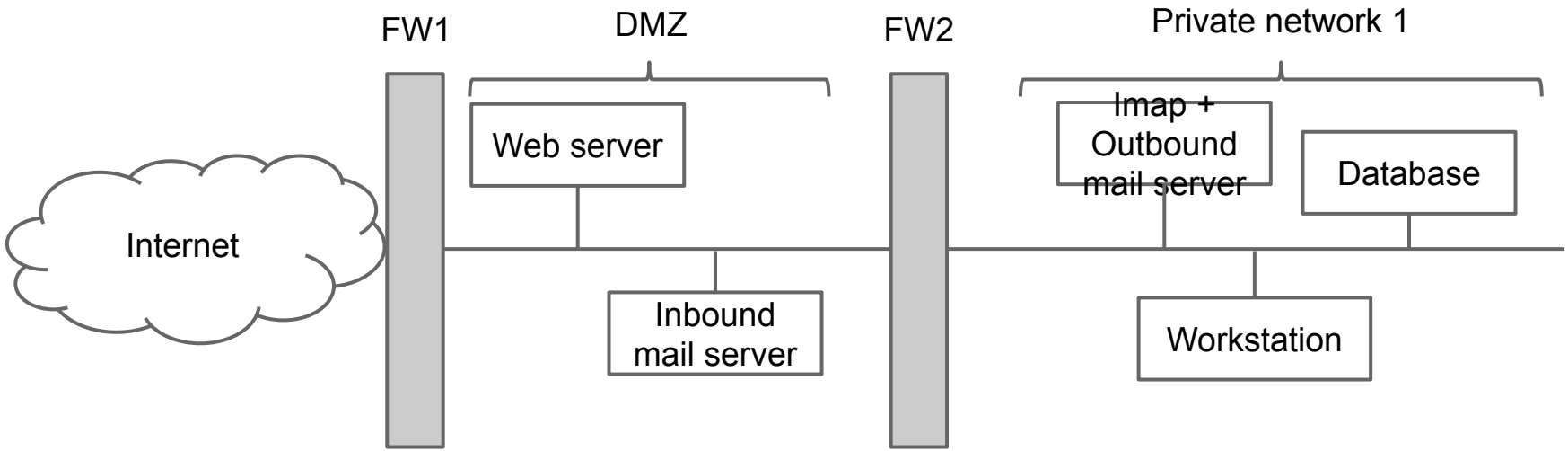
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)



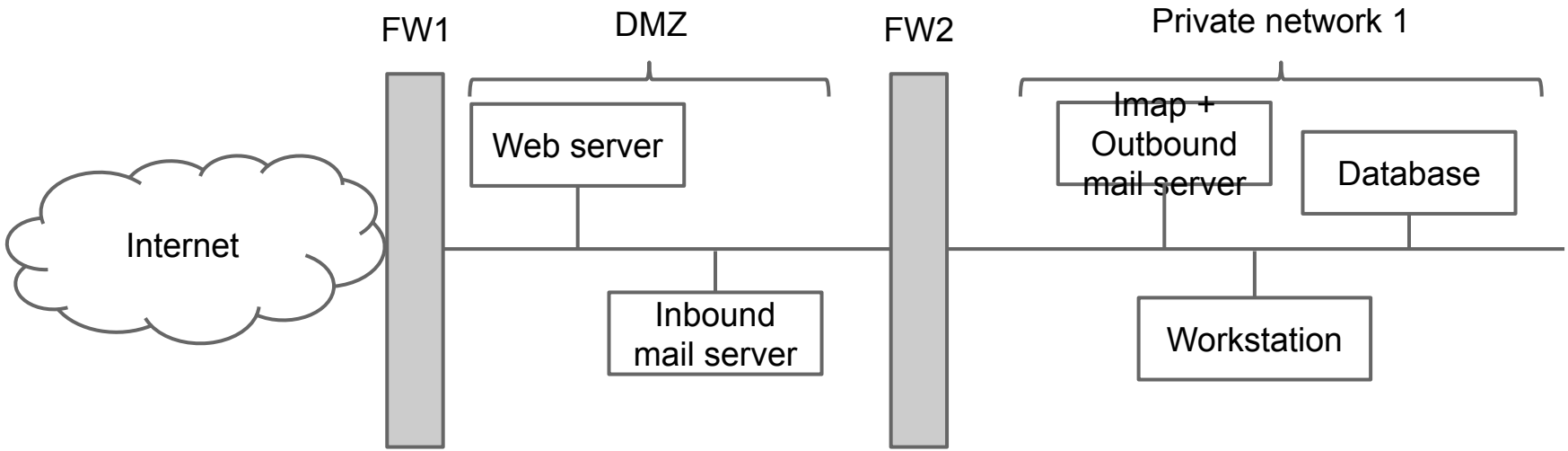
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*



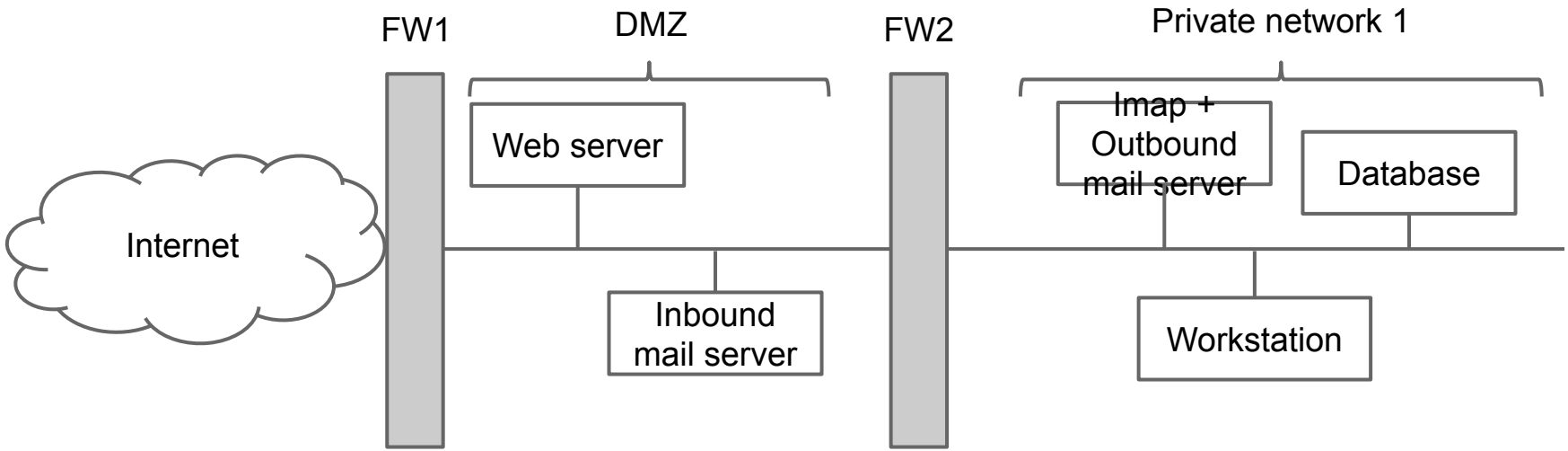
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*



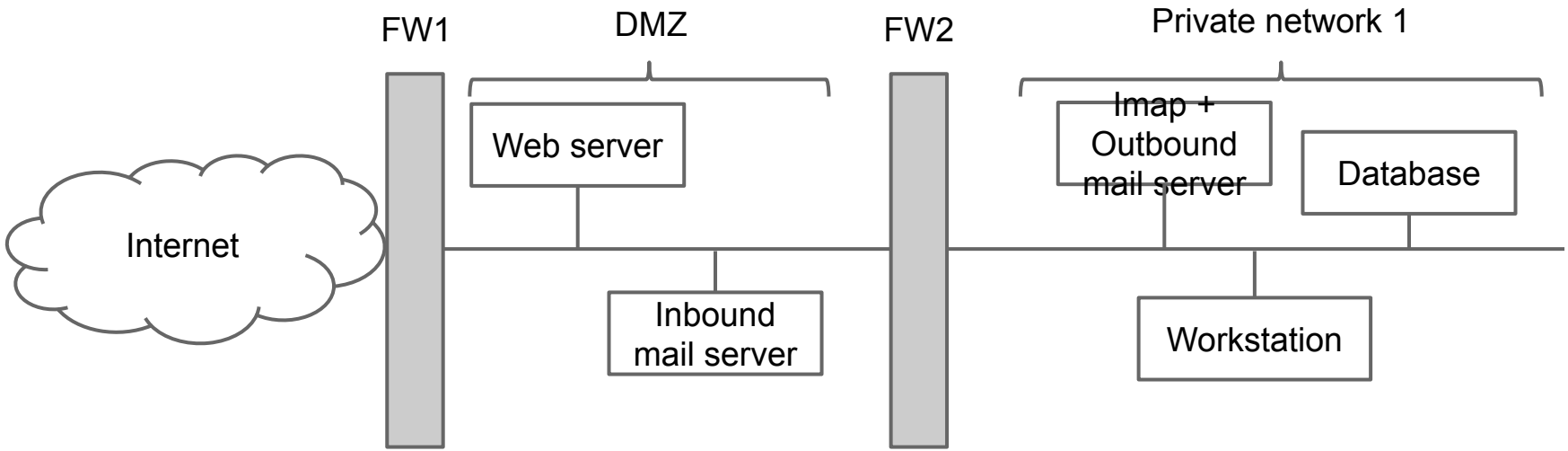
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*



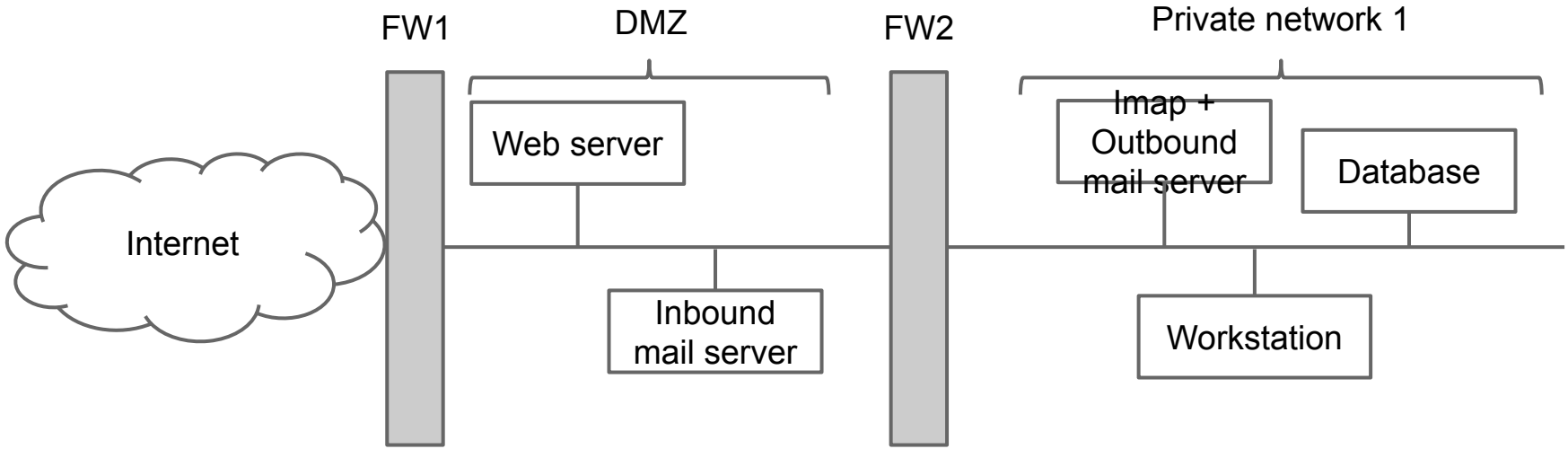
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*587*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*587*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*
FW2	*Workstation_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW1	*Workstation_IP*	*ANY*	*DMZ → Internet*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*587*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*
FW2	*Workstation_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW1	*Workstation_IP*	*ANY*	*DMZ → Internet*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW2	*SMTPOUT_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*25*	*ALLOW*	*The application server sends email (relayed by the SMTPOut server)*
FW1	*SMTPOUT_IP*	*ANY*	*DMZ → Internet*	*ANY*	*25*	*ALLOW*	*The application server sends email (relayed by the SMTPOut server)*