Harden your program the hard way

by Jhe & Eddy@HITCON-CMT

Who am I ?

- Jhe
- co-founder of UCCU
- know a little
  - Web security
  - Linux exploitation
  - Python

Who are we ?

Kuon : PM

Jhe : Exploit PoC

Eddy : Solution implementation

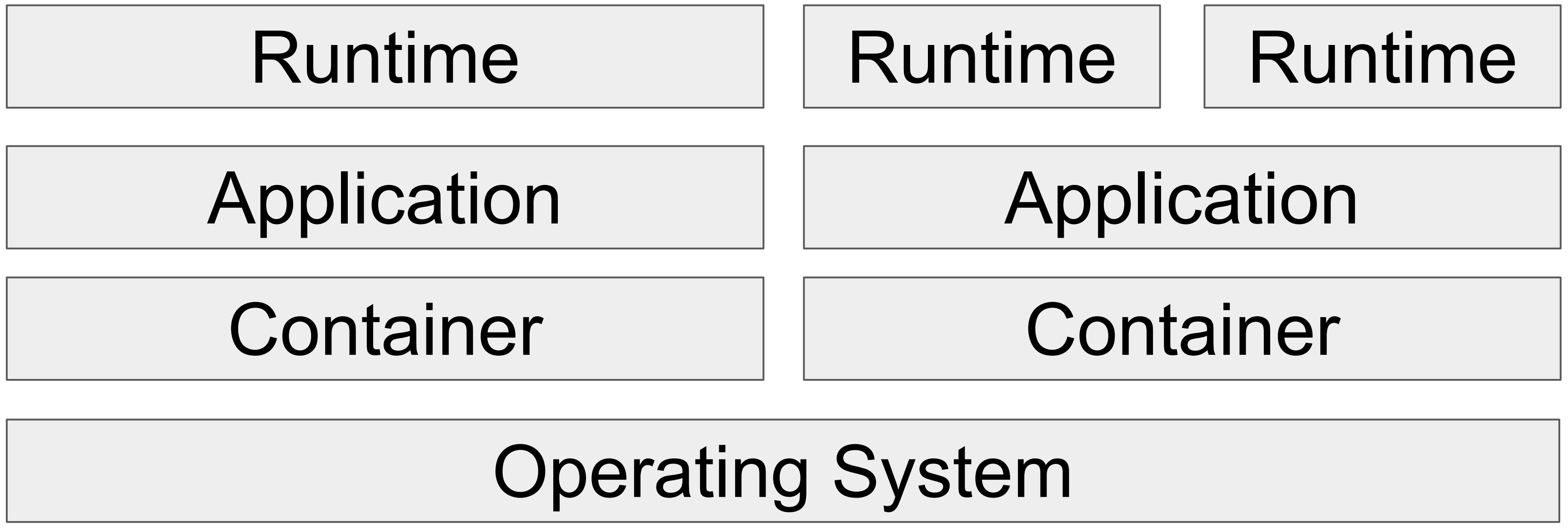
AJ : Solution implementation

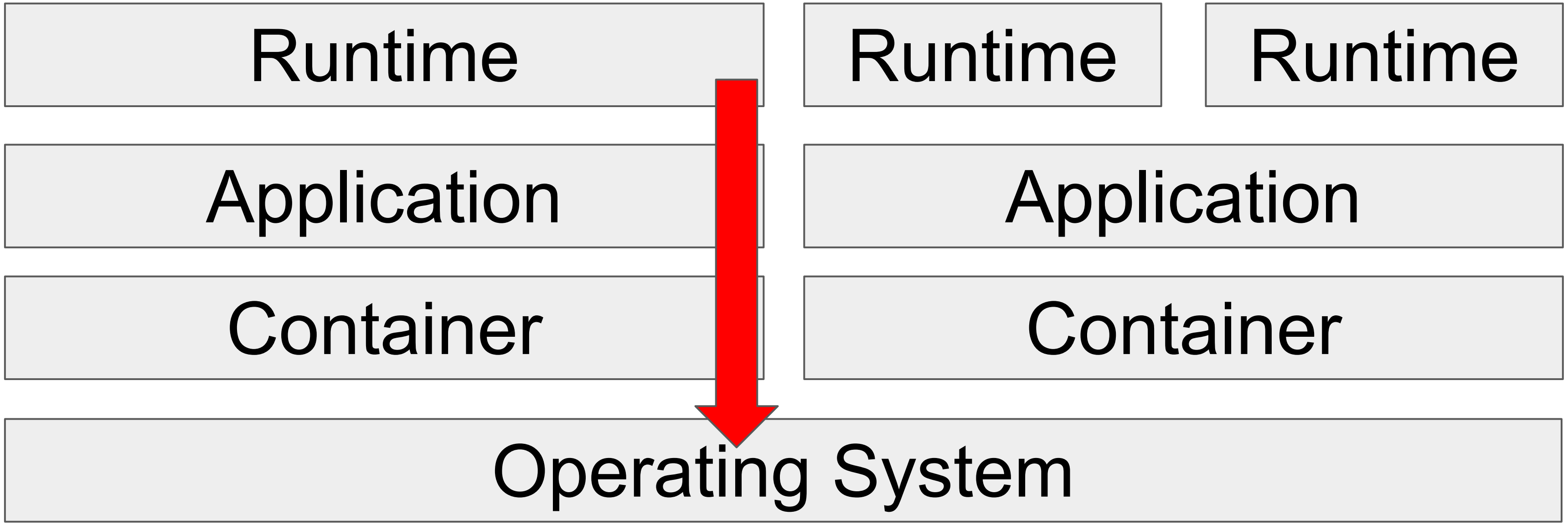
工業技術研究院
Industrial Technology
Research Institute

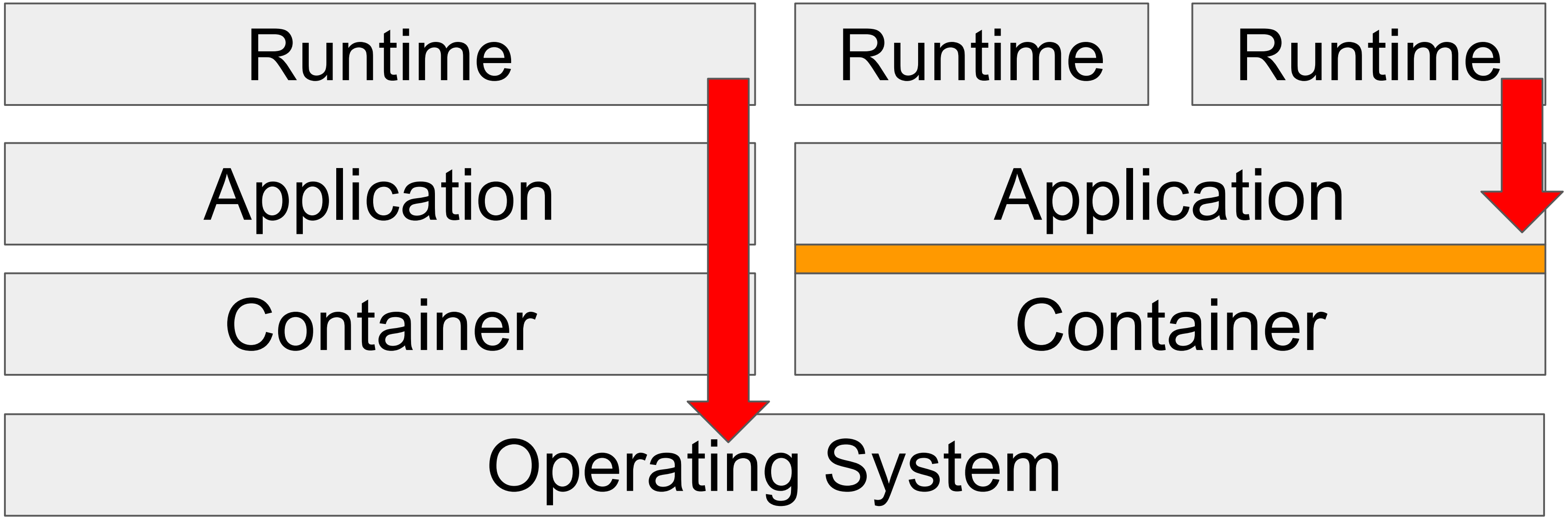
Funny
·
Systems

UCCU

Why ?


Runtime
Runtime
Runtime
Application
Application
Container
Container
Operating System


Runtime
Runtime
Runtime
Application
Application
Container
Container
Operating System


Runtime
Runtime
Runtime
Application
Application
Container
Container
Operating System

Compiler-based approach security solution

In a nutshell

Harden your program after compiled

Prerequisites

1. Terms

Prerequisites

1. Terms
2. Buffer overflow attack

Prerequisites

1. Terms
2. Buffer overflow attack
3. Use after free

Vulnerability
vs
Exploit

Proof of Concept (PoC)

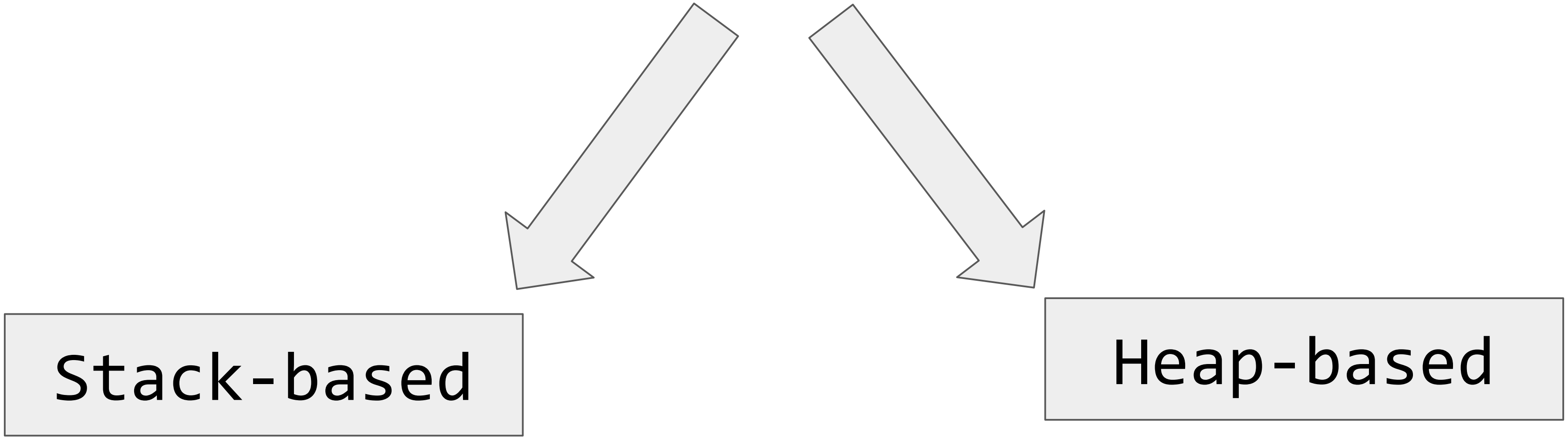
Mitigation

Buffer overflow (Bof)

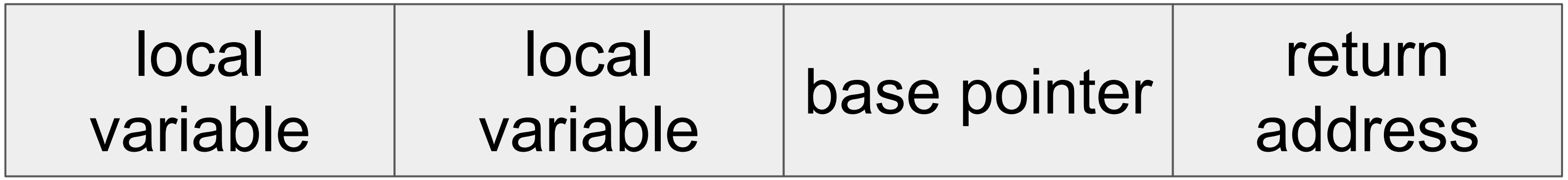
Moving Target Defense (MTD)

MTD = confuse your enemie

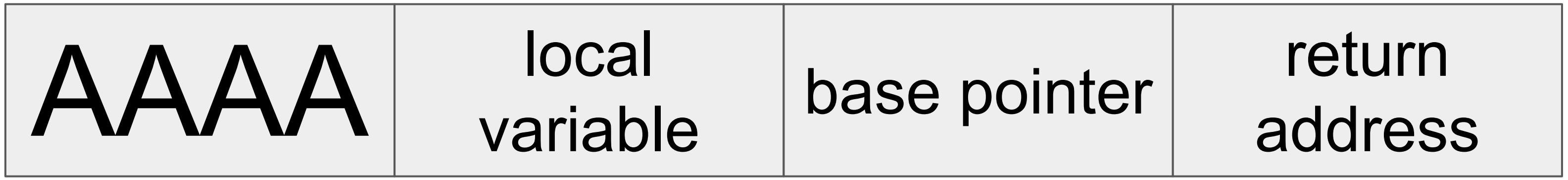
Terms | **Buffer overflow** | Use after free



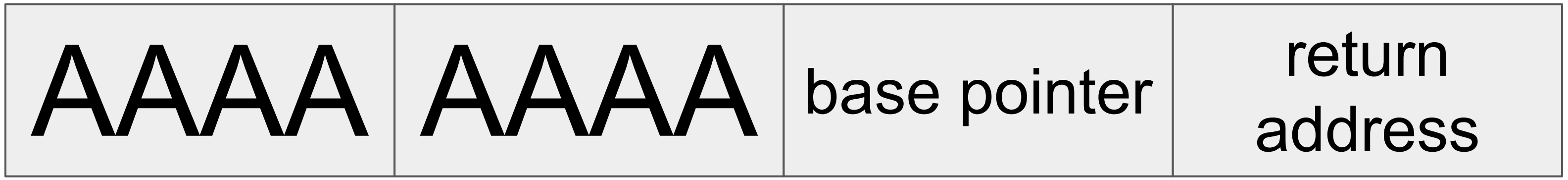
Terms	Buffer overflow	Use after free

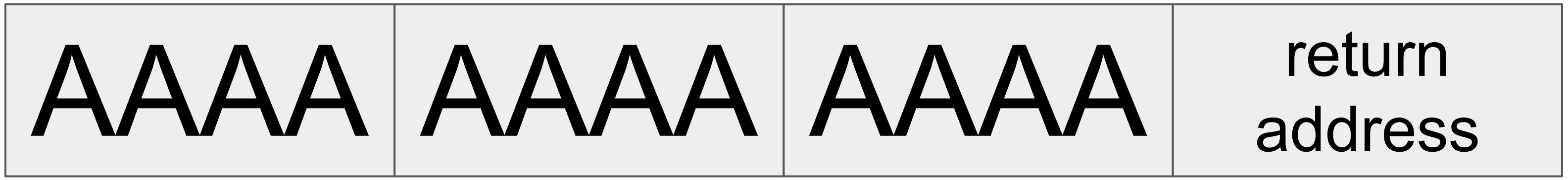


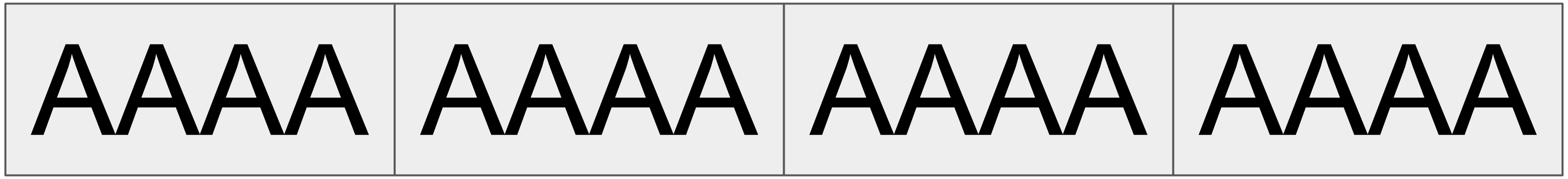
Terms	Buffer overflow	Use after free



Terms	**Buffer overflow**	Use after free




AAAA
AAAA
AAAA
return address


AAAA
AAAA
AAAA
AAAA

Terms	Buffer overflow	Use after free

AAAA AA

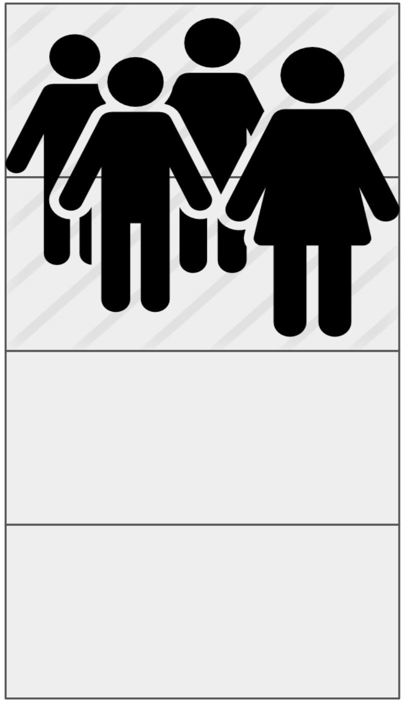
```
>Segmentation Fault
(Core Dumped)
```

Terms	Buffer overflow	Use after free

```
malloc(TWs)
TWs->say()
free(TWs)
```

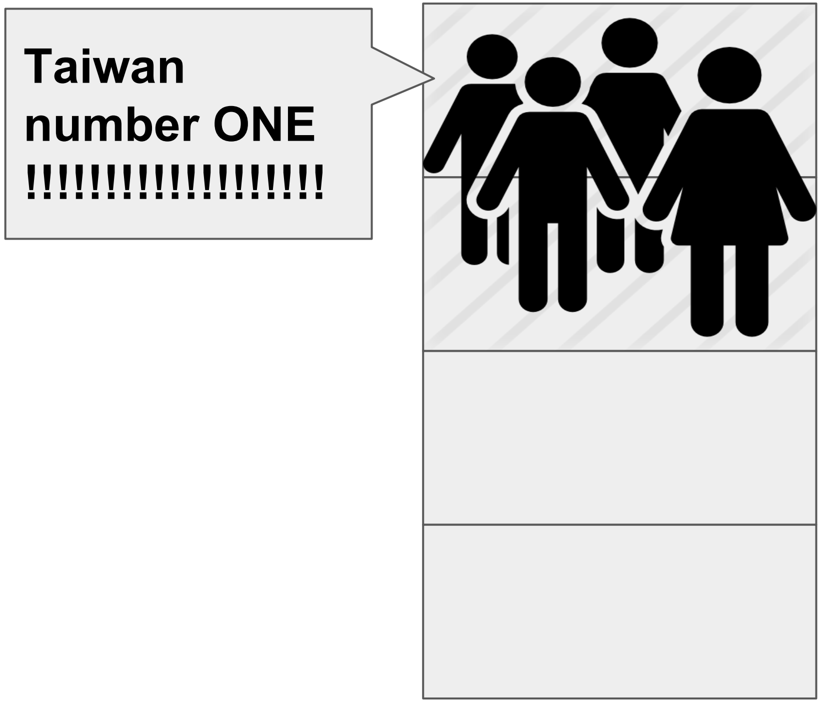
Terms	Buffer overflow	Use after free

```
malloc(TWs)
TWs->say()
free(TWs)
```



Terms | Buffer overflow | **Use after free**

```
malloc(TWs)
TWs->say()
free(TWs)
```



Terms | Buffer overflow | **Use after free**

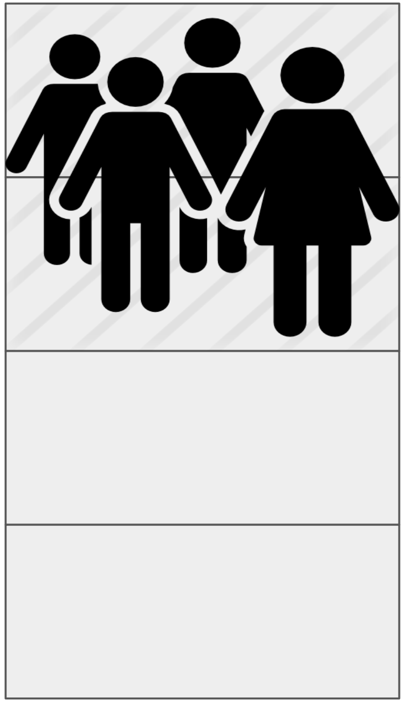
```
malloc(TWs)
TWs->say()
free(TWs)
```

Terms	Buffer overflow	Use after free

malloc(TWs)
free(TWs)
malloc(Xs)
TWs->say()

Terms	Buffer overflow	Use after free

malloc(TWs)
free(TWs)
malloc(Xs)
TWs->say()



Terms | Buffer overflow | **Use after free**

malloc(TWs)

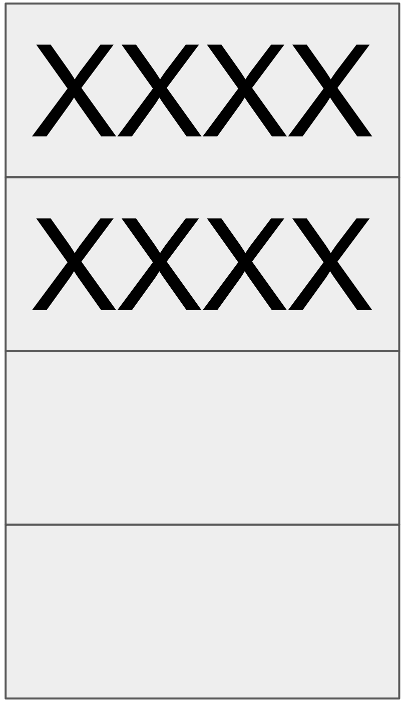
free(TWs)

malloc(Xs)

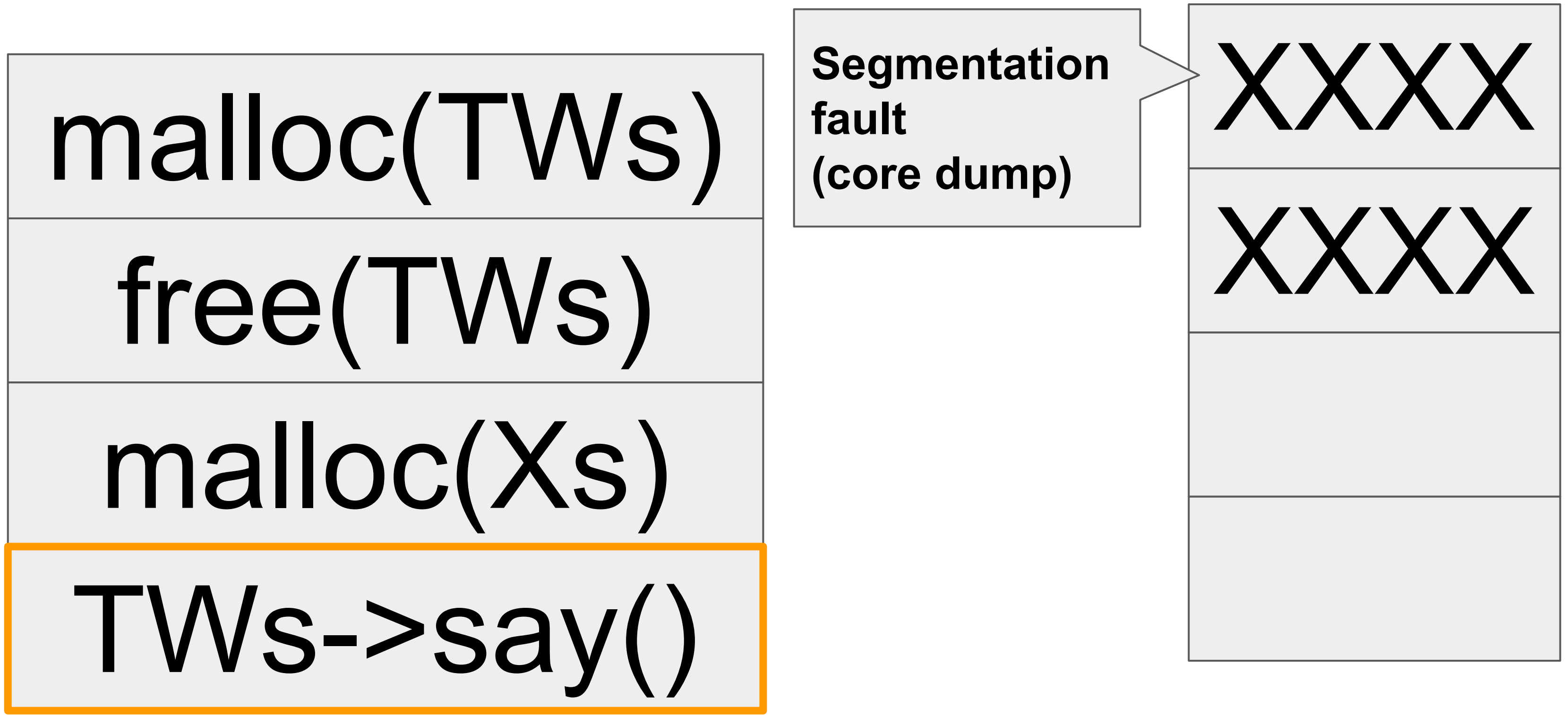
TWs->say()

Terms	Buffer overflow	Use after free

malloc(TWs)
free(TWs)
malloc(Xs)
TWs->say()



Terms | Buffer overflow | Use after free



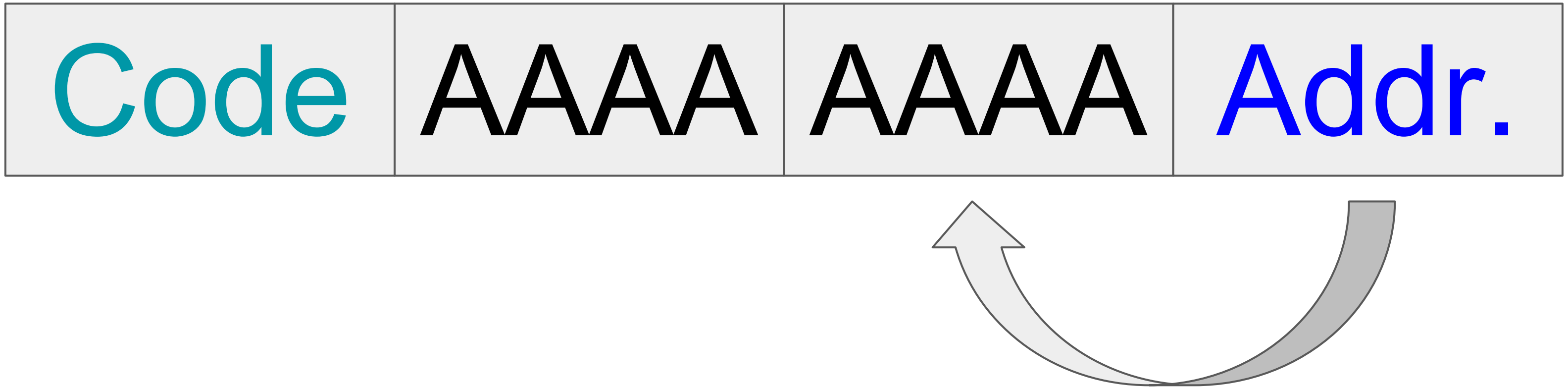
ASLR	DEP	Stack guard

Address Space Layout Randomization

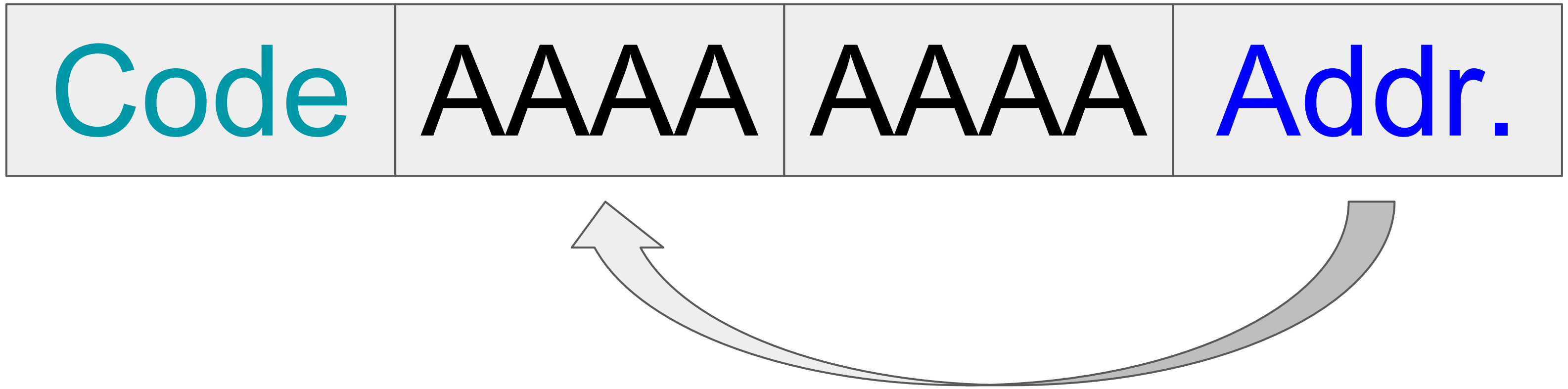
ASLR	DEP	Stack guard

Code	AAAA	AAAA	Addr.

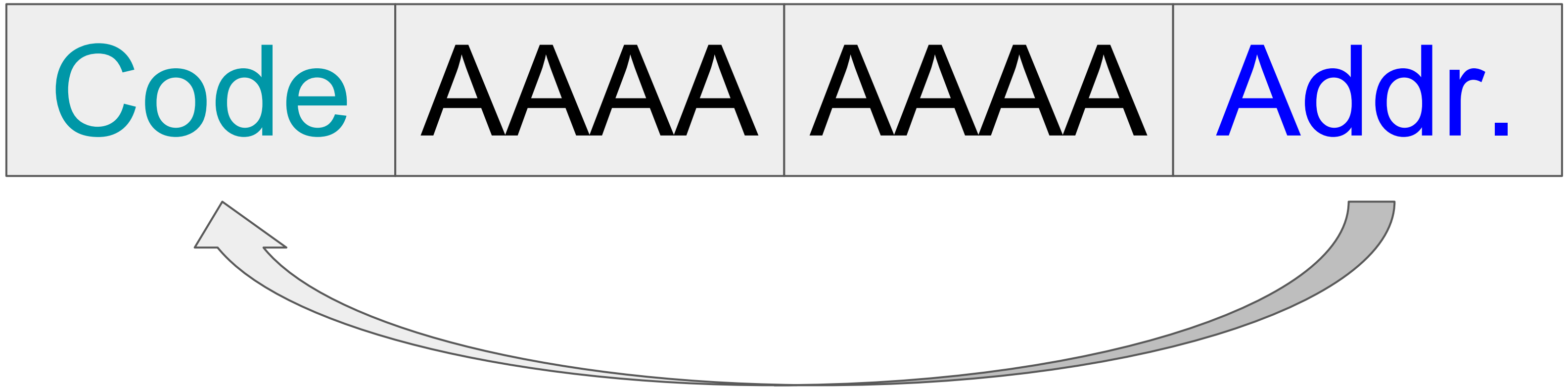
ASLR | DEP | Stack guard



ASLR	DEP	Stack guard

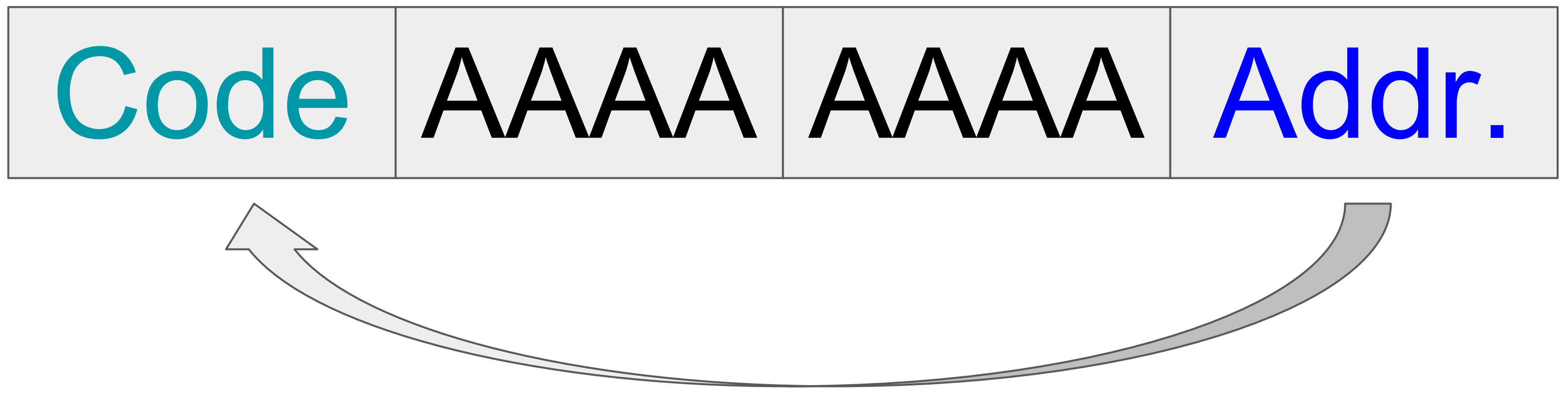


ASLR	DEP	Stack guard

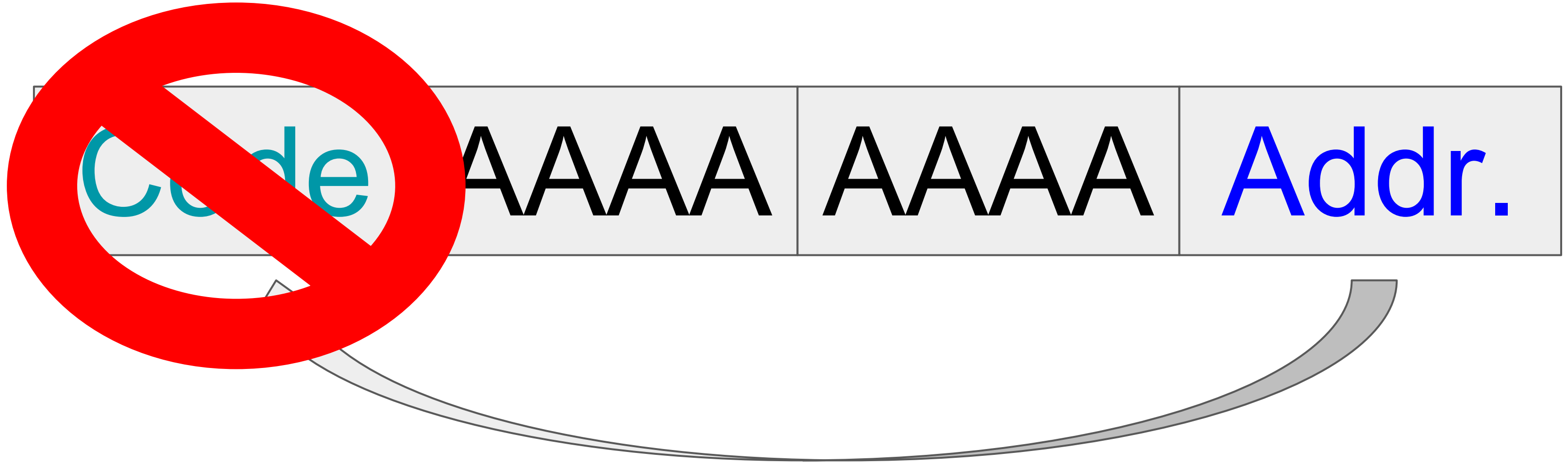


Data Execution Prevention

ASLR	DEP	Stack guard



ASLR	DEP	Stack guard



Stack guard

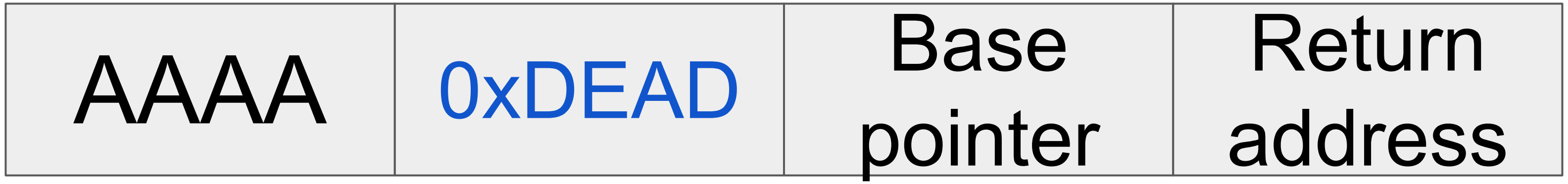
ASLR	DEP	**Stack guard**

Local variable	Stack guard	Base pointer	Return address

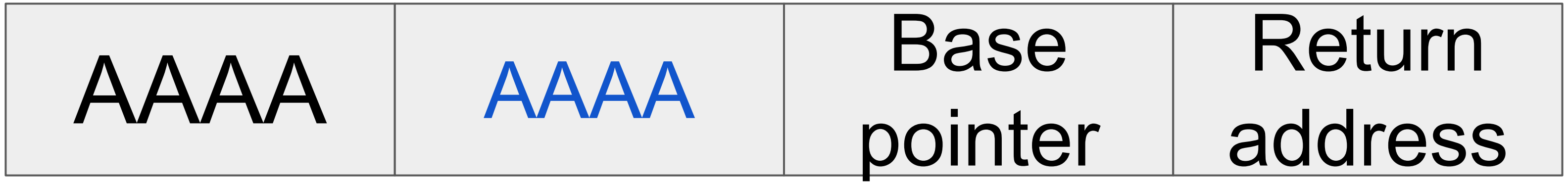
ASLR	DEP	**Stack guard**

Local variable	Base pointer	Return address

ASLR	DEP	**Stack guard**



ASLR | DEP | **Stack guard**



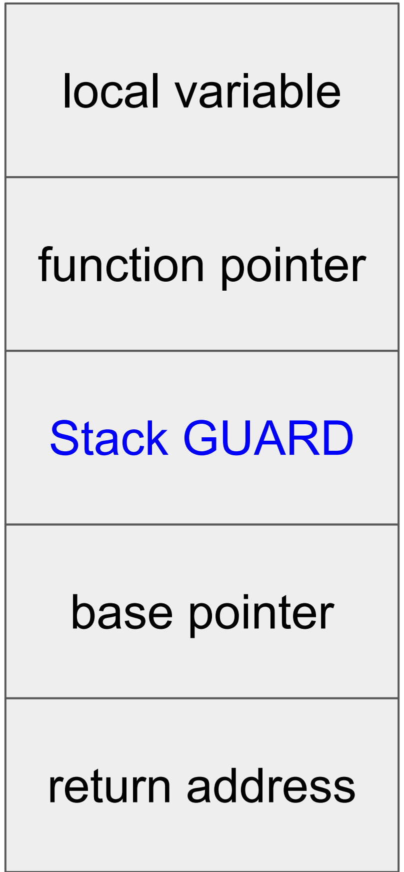
ASLR	DEP	Stack guard

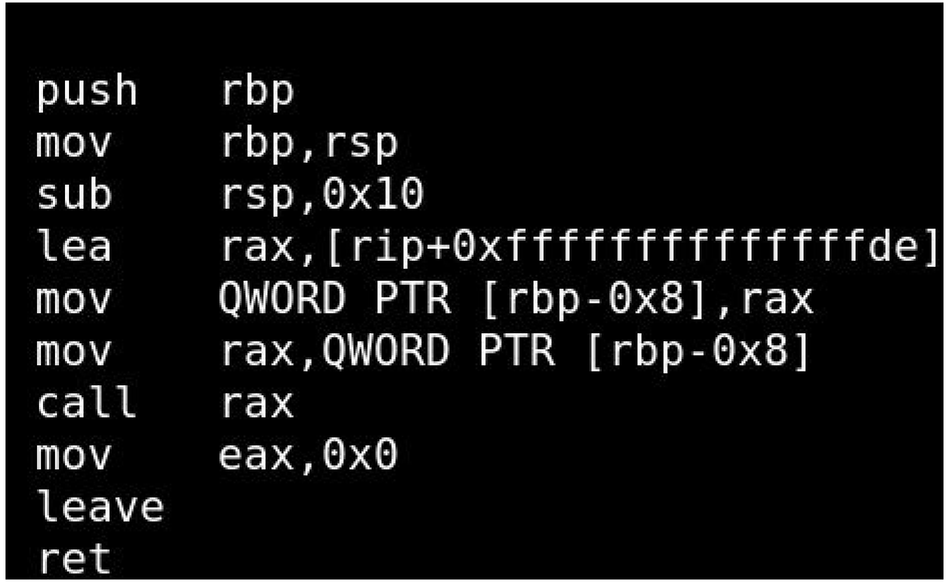
AAAA
AAAA
AAAA
AAAA

ASLR	DEP	Stack guard

AAA AAA

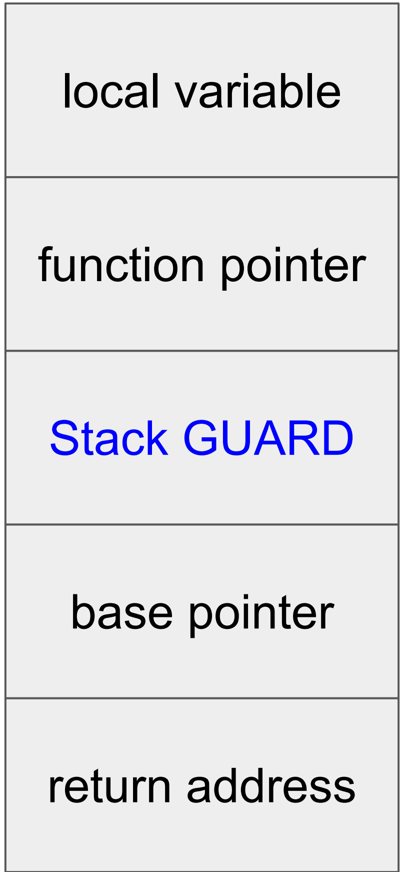
Function Pointer overwrite

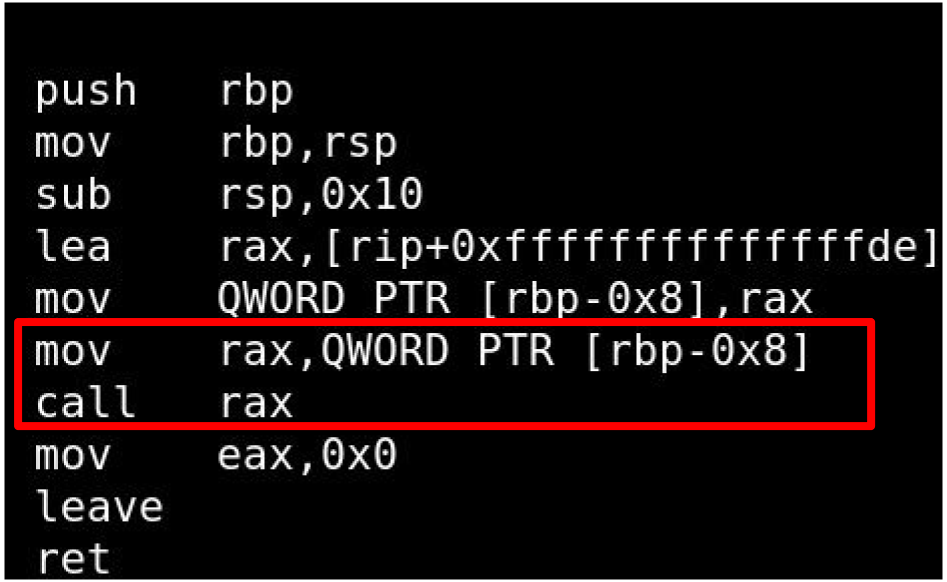


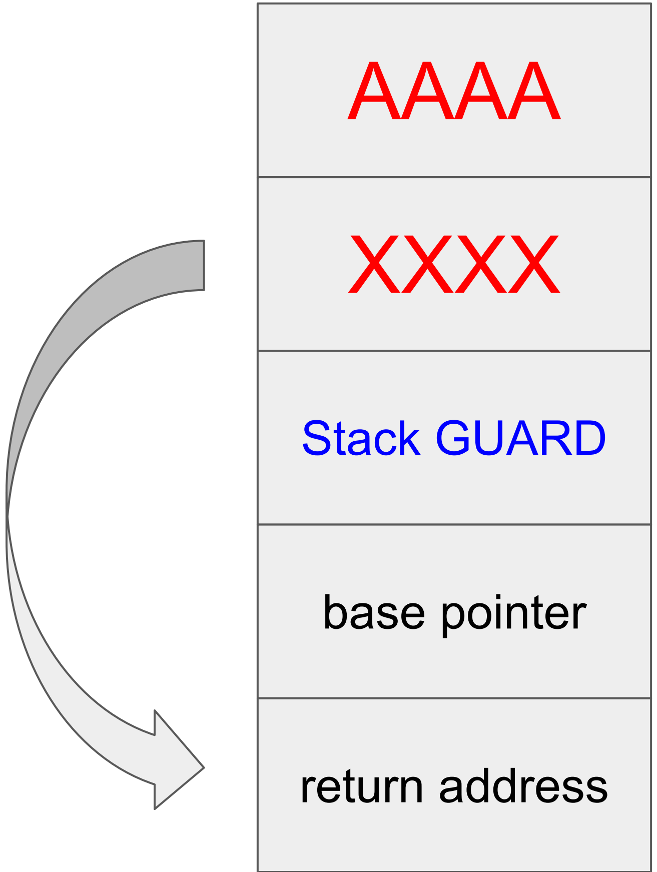


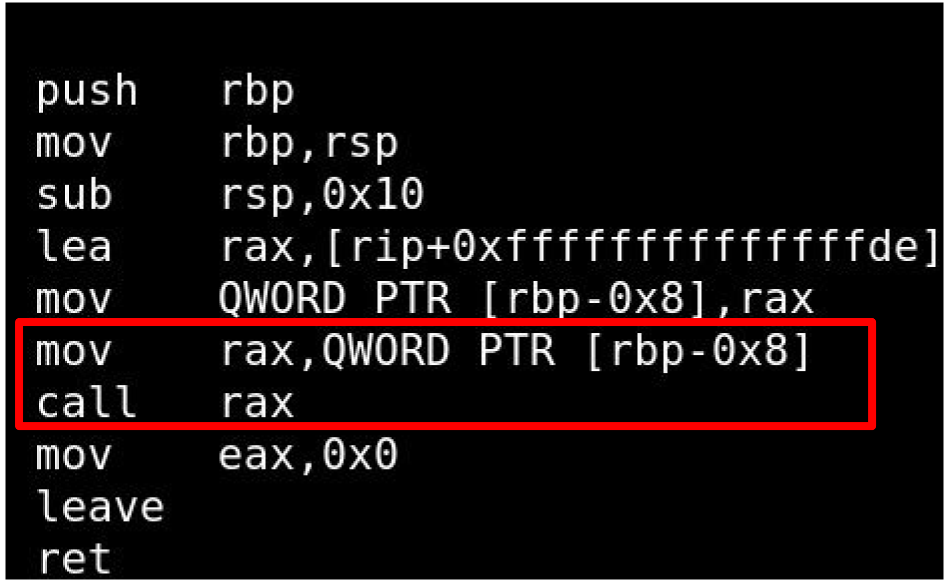
```
push    rbp
mov     rbp,rsp
sub     rsp,0x10
lea     rax,[rip+0xffffffffffffffde]
mov     QWORD PTR [rbp-0x8],rax
mov     rax,QWORD PTR [rbp-0x8]
call    rax
mov     eax,0x0
leave
ret
```

FP overwrite	ROP	BROP	offset2lib




```
push    rbp
mov     rbp,rsp
sub     rsp,0x10
lea     rax,[rip+0xffffffffffffffde]
mov     QWORD PTR [rbp-0x8],rax
mov     rax,QWORD PTR [rbp-0x8]
call    rax
mov     eax,0x0
leave
ret
```





```
push    rbp
mov     rbp,rsp
sub     rsp,0x10
lea     rax,[rip+0xffffffffffffffde]
mov     QWORD PTR [rbp-0x8],rax
mov     rax,QWORD PTR [rbp-0x8]
call    rax
mov     eax,0x0
leave
ret
```

Return Oriented Programming

FP overwrite	**ROP**	BROP	offset2lib

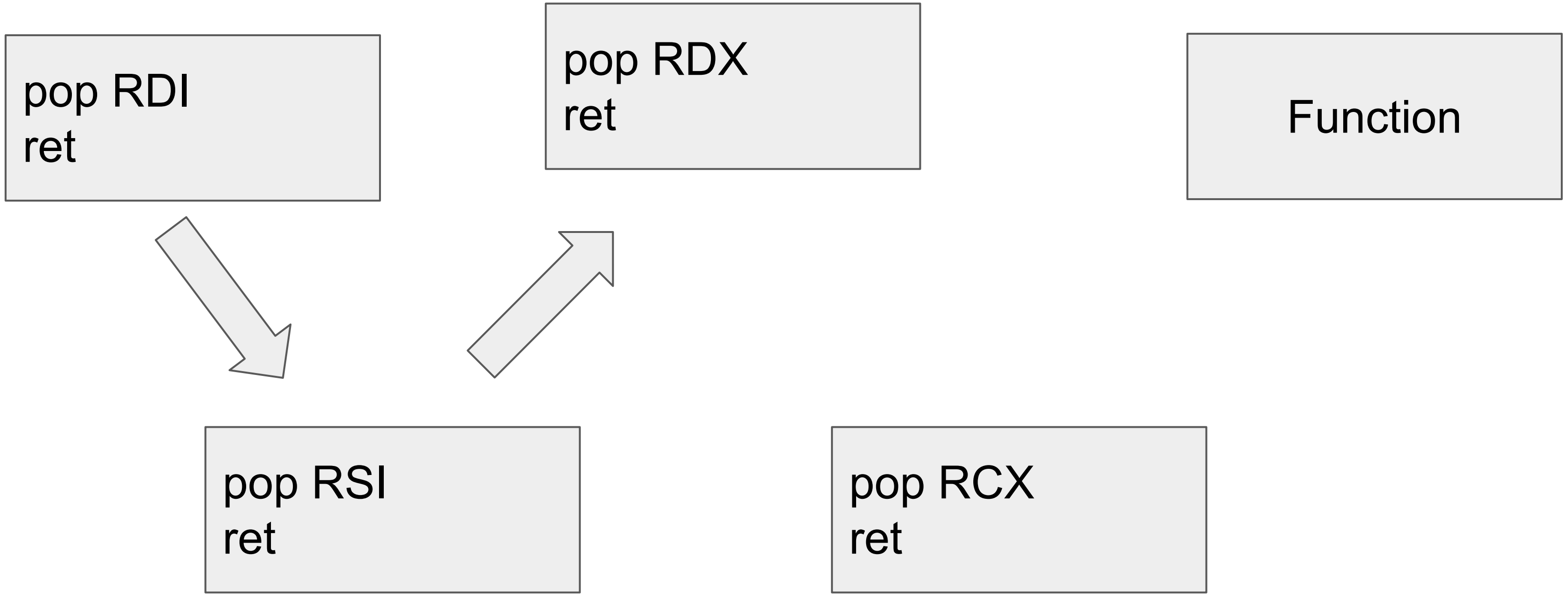
```
pop RDI
ret
```

```
pop RDX
ret
```

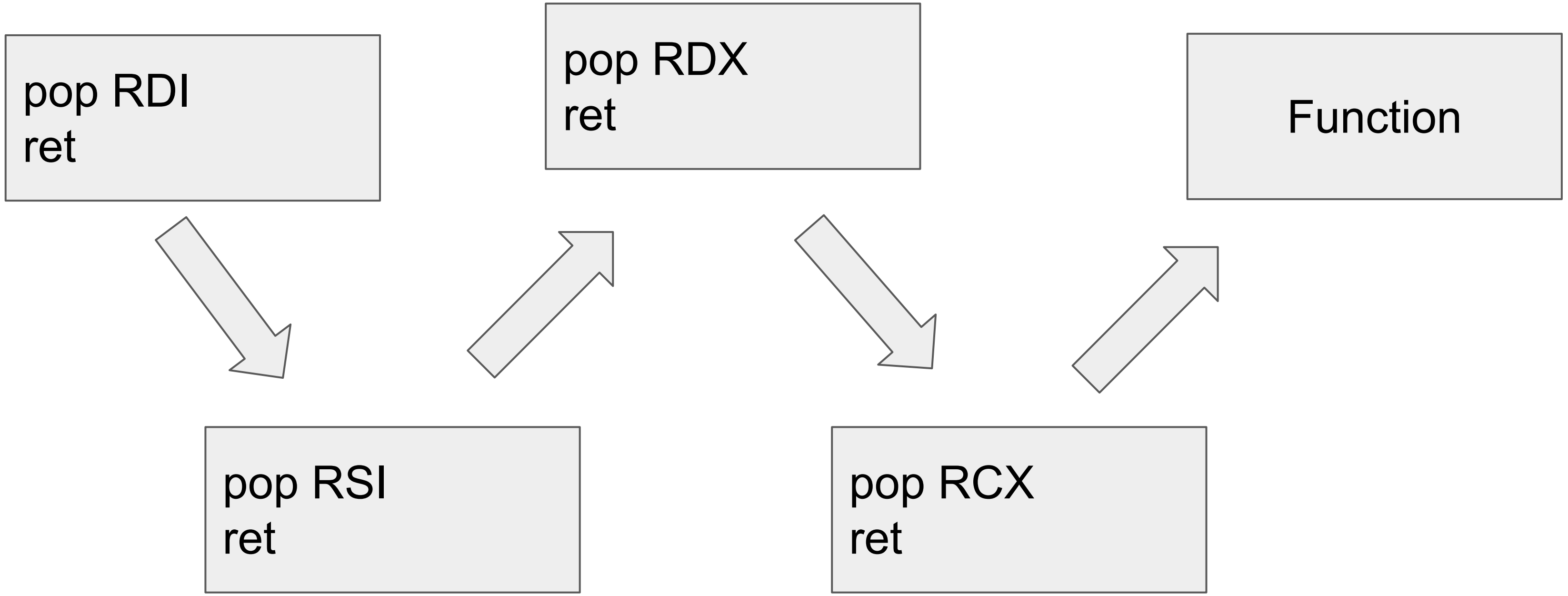
Function

```
pop RSI
ret
```

```
pop RCX
ret
```


pop RDI
ret
pop RDX
ret
Function
pop RSI
ret
pop RCX
ret

FP overwrite	ROP	BROP	offset2lib



Blind ROP

http://www.scs.stanford.edu/brop/

Stack reading

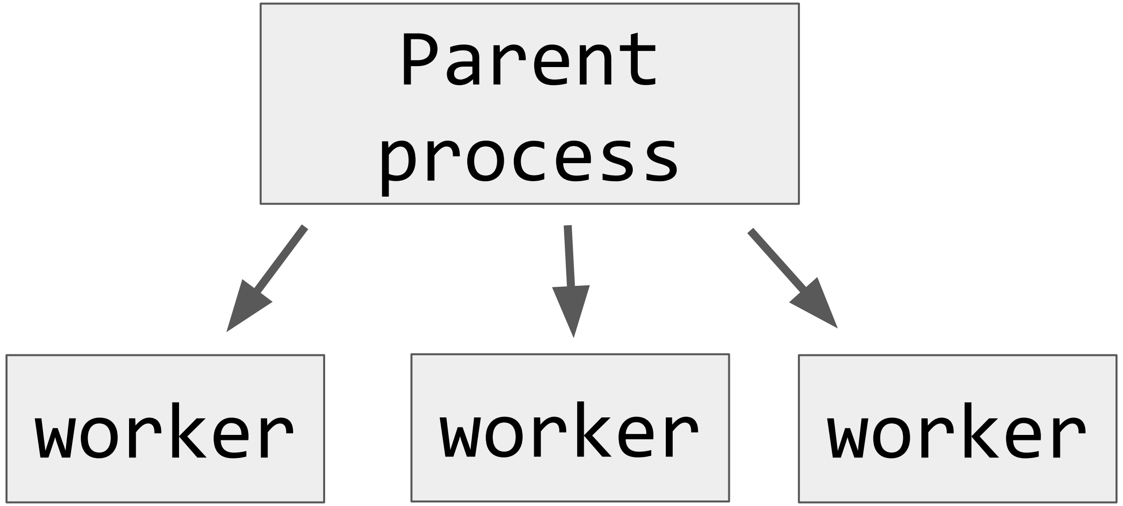
FP overwrite	ROP	BROP	offset2lib

Apache

Nginx

Samba

OpenSSH



CVE 2013-2028

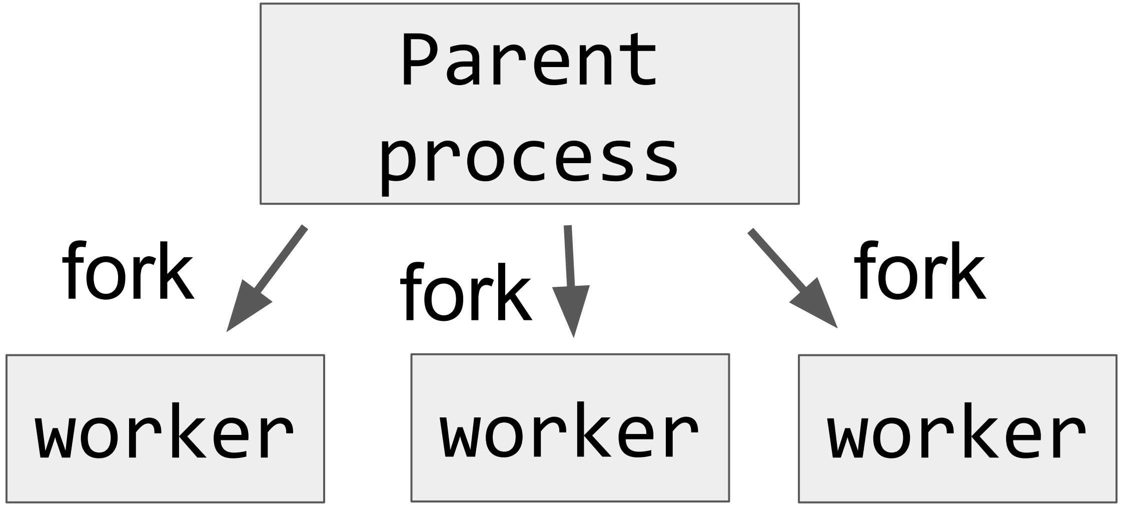
FP overwrite	ROP	BROP	offset2lib

Apache

Nginx

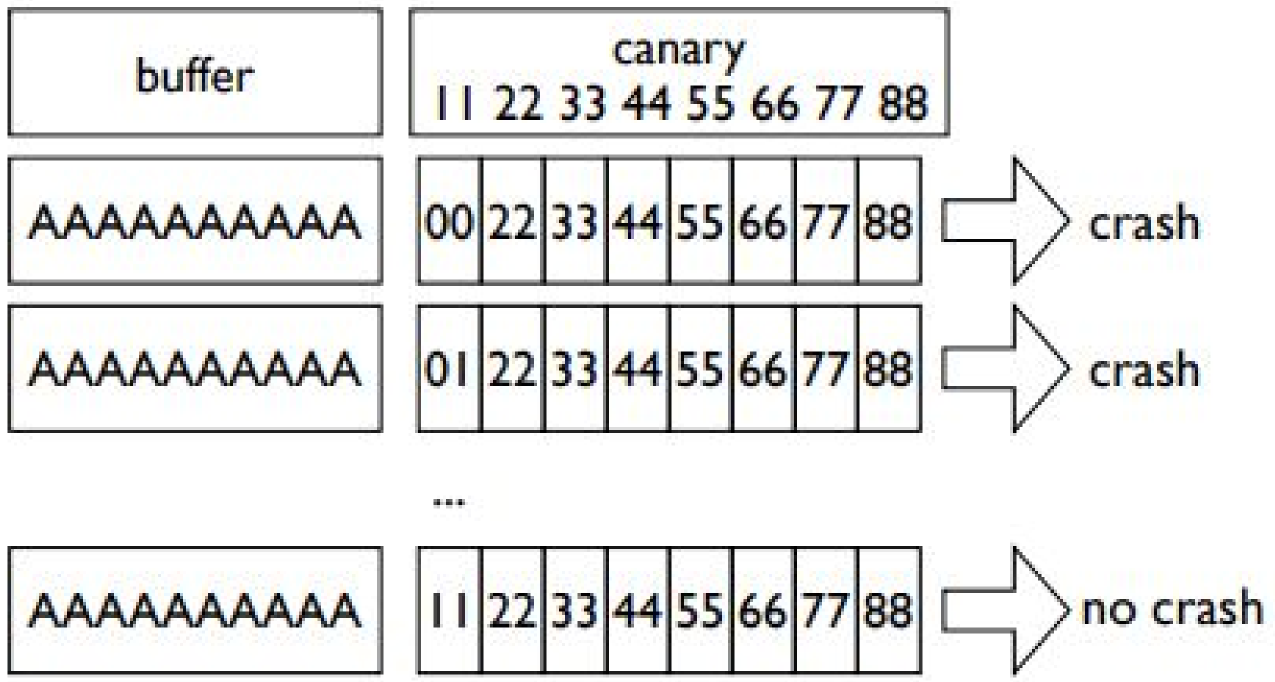
Samba

OpenSSH



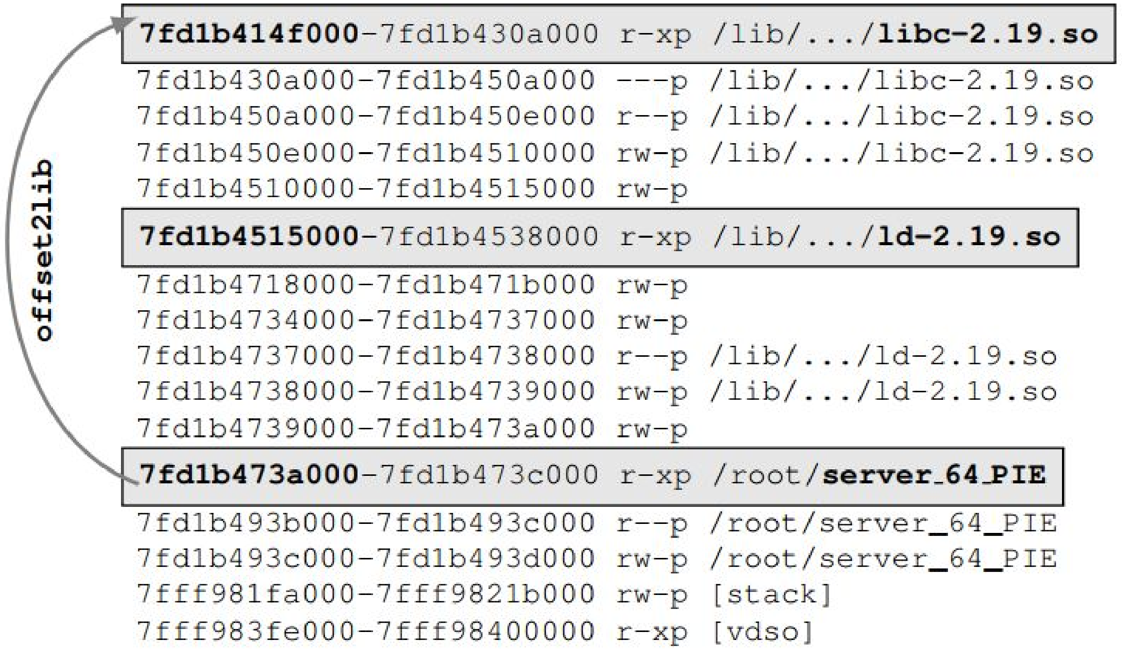
CVE 2013-2028

FP overwrite	ROP	BROP	offset2lib



Offset to library

https://cybersecurity.upv.es/attacks/offset2lib/offset2lib.html



```
7fd1b414f000-7fd1b430a000 r-xp /lib/.../libc-2.19.so
7fd1b430a000-7fd1b450a000 ---p /lib/.../libc-2.19.so
7fd1b450a000-7fd1b450e000 r--p /lib/.../libc-2.19.so
7fd1b450e000-7fd1b4510000 rw-p /lib/.../libc-2.19.so
7fd1b4510000-7fd1b4515000 rw-p
7fd1b4515000-7fd1b4538000 r-xp /lib/.../ld-2.19.so
7fd1b4718000-7fd1b471b000 rw-p
7fd1b4734000-7fd1b4737000 rw-p
7fd1b4737000-7fd1b4738000 r--p /lib/.../ld-2.19.so
7fd1b4738000-7fd1b4739000 rw-p /lib/.../ld-2.19.so
7fd1b4739000-7fd1b473a000 rw-p
7fd1b473a000-7fd1b473c000 r-xp /root/server_64_PIE
7fd1b493b000-7fd1b493c000 r--p /root/server_64_PIE
7fd1b493c000-7fd1b493d000 rw-p /root/server_64_PIE
7fff981fa000-7fff9821b000 rw-p [stack]
7fff983fe000-7fff98400000 r-xp [vdso]
```

Distribution	Libc version	Offset2lib (bytes)
CentOS 6.5	2.12	0x5b6000
Debian 7.1	2.13	0x5ac000
Ubuntu 12.04 LTS	2.15	0x5e4000
Ubuntu 12.10	2.15	0x5e4000
Ubuntu 13.10	2.17	0x5ed000
openSUSE 13.1	2.18	0x5d1000
Ubuntu 14.04.1 LTS	2.19	0x5eb000

Compiler-based =

Front-end +

IR +

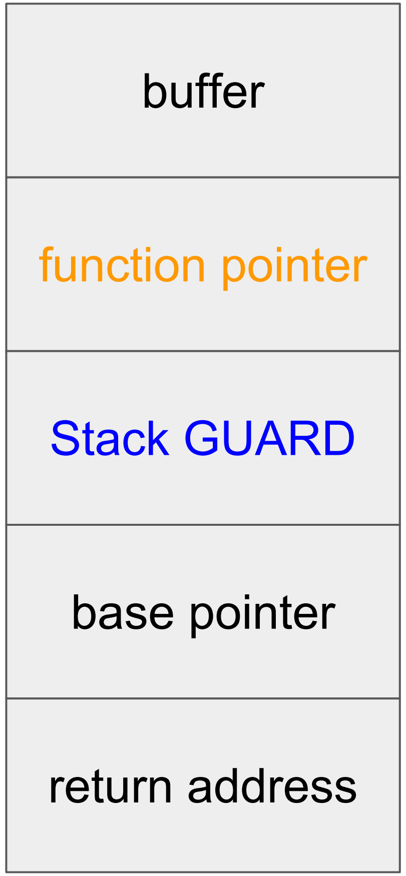
Back-end

FP protection	Function padding	Variable re-order	Two birds

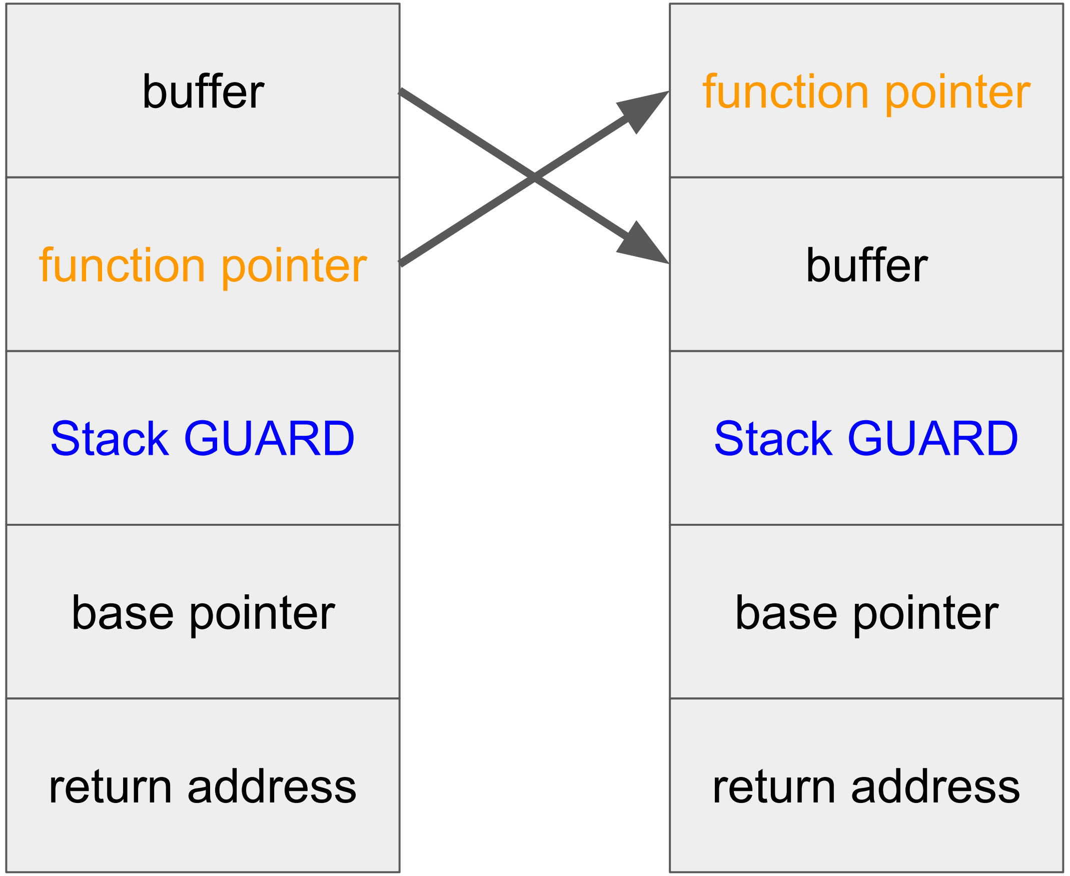
FP protection	Function padding	Variable re-order	Two birds

return address is also pointer

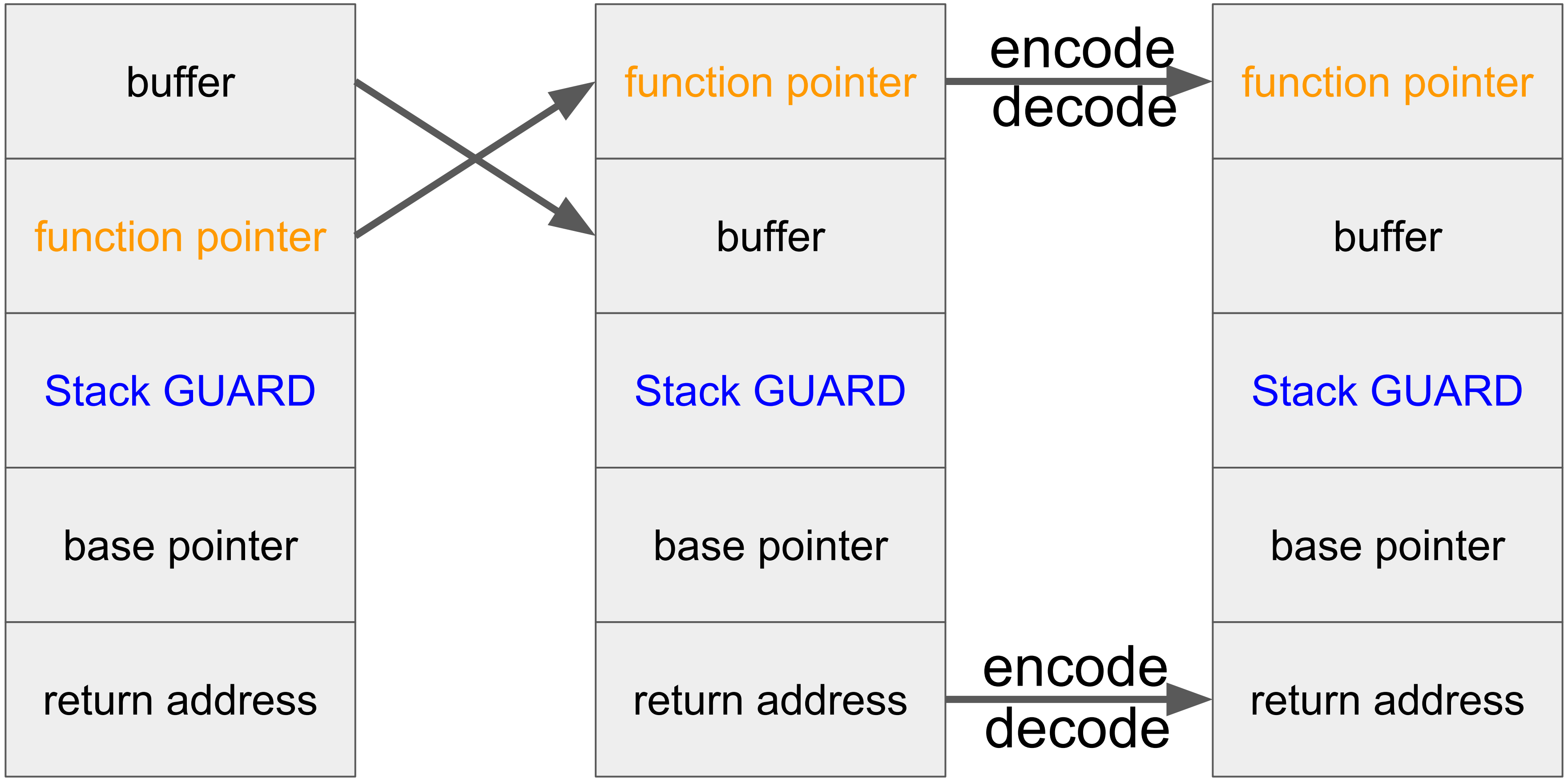
FP protection	Function padding	Variable re-order	Two birds



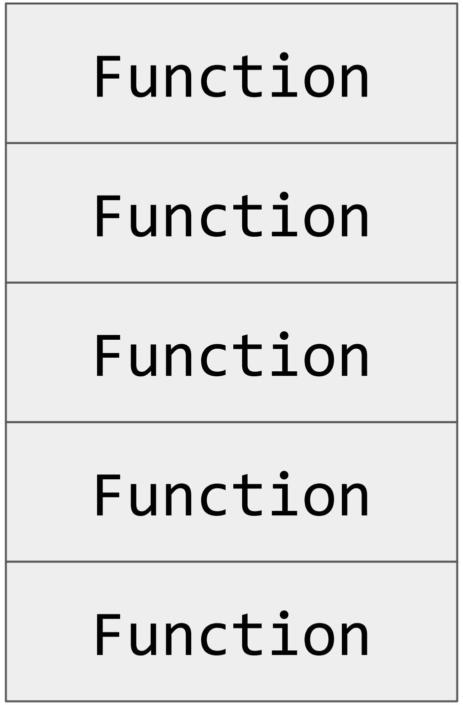
FP protection	Function padding	Variable re-order	Two birds



FP protection	Function padding	Variable re-order	Two birds

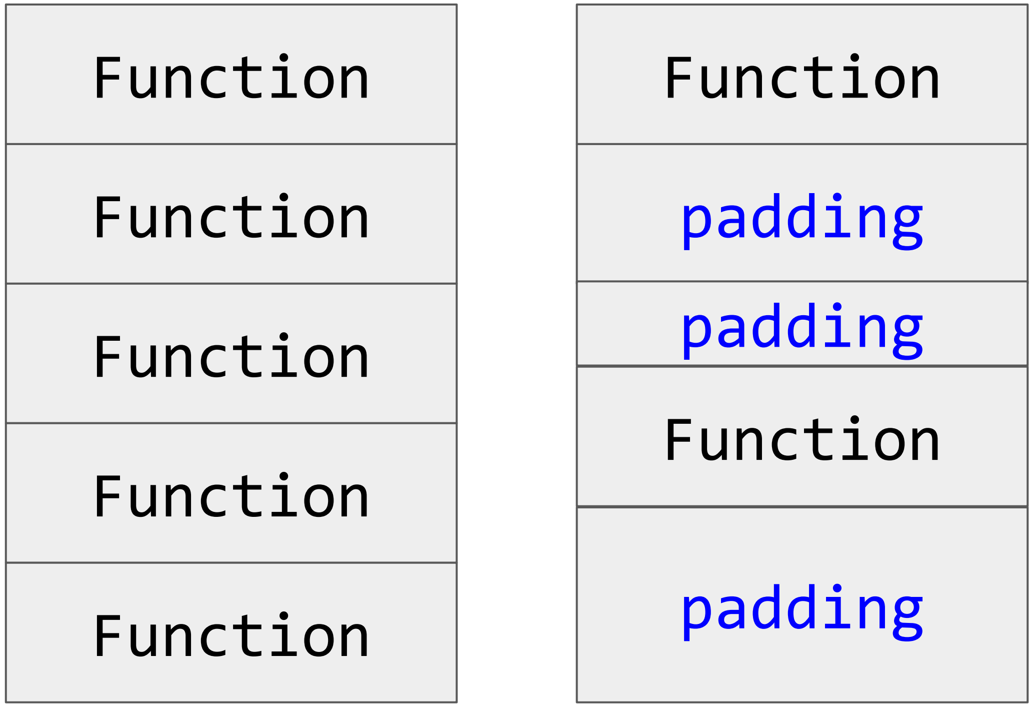


FP protection | **Function padding** | Variable re-order | Two birds

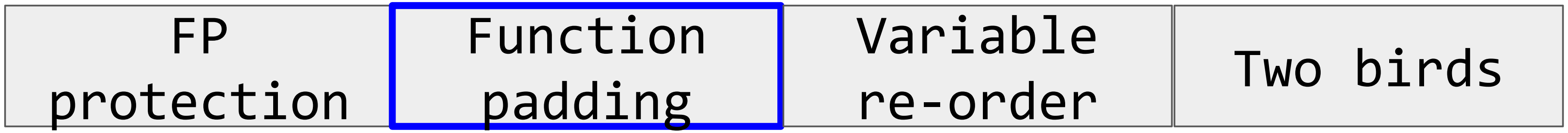


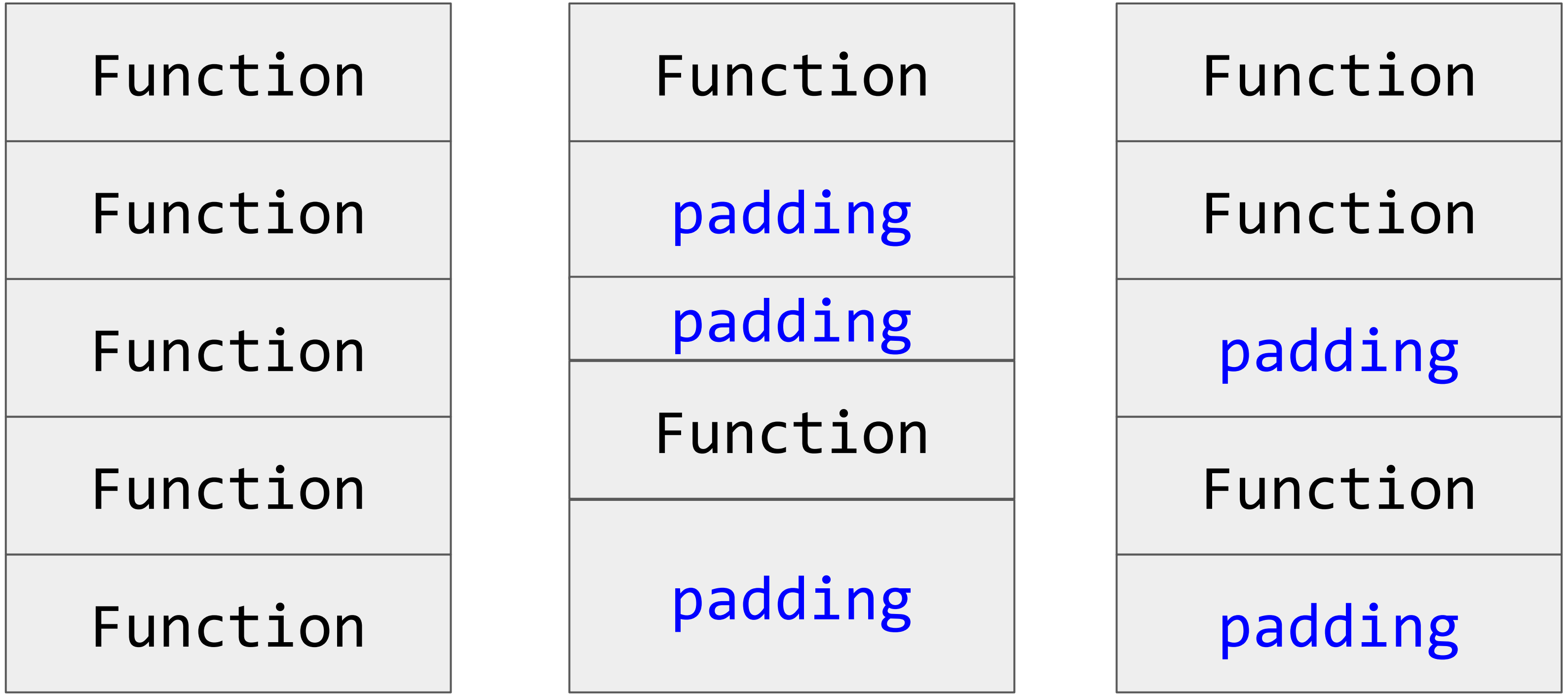
CVE 2012-4221

FP protection	**Function padding**	Variable re-order	Two birds



CVE 2012-4221

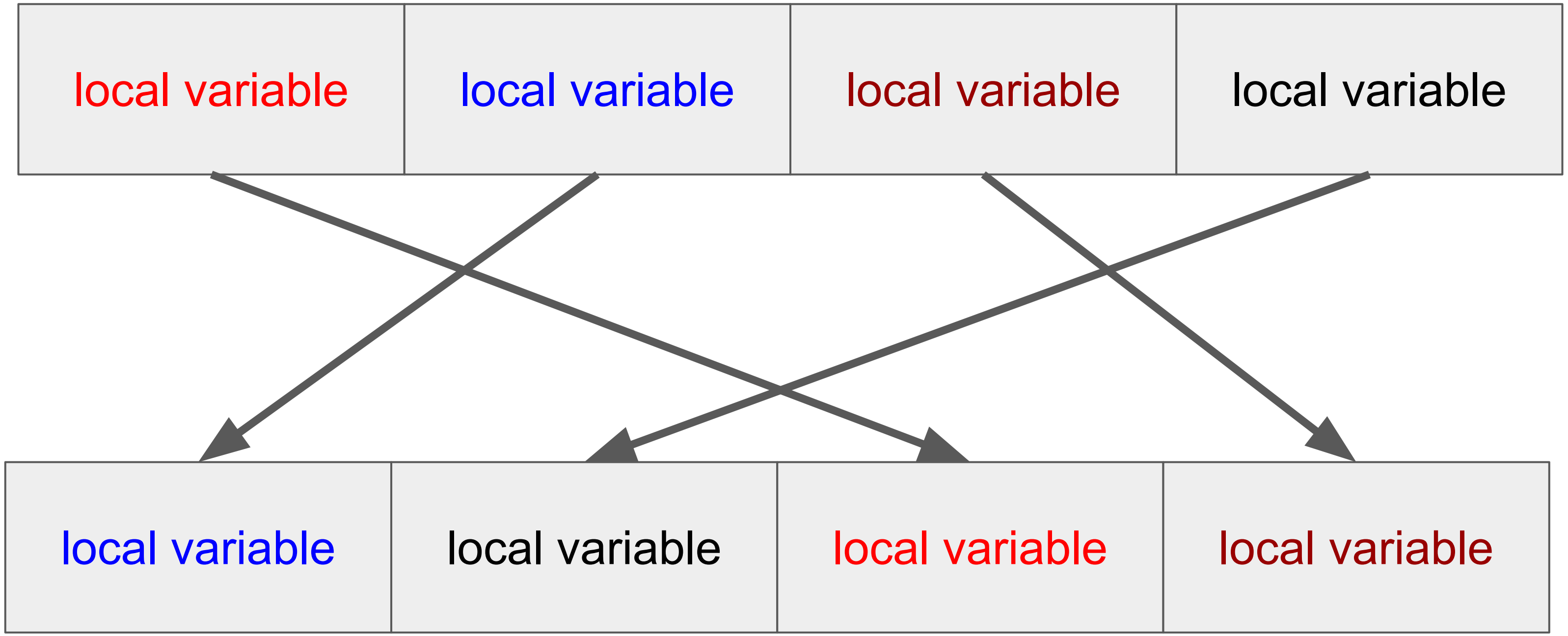




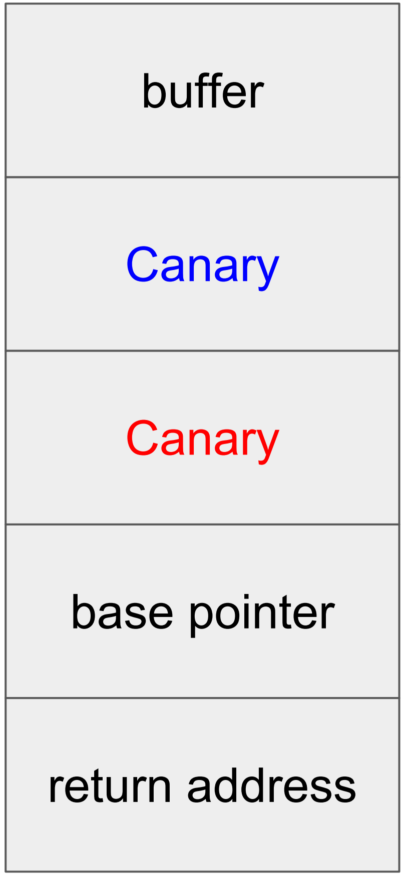
CVE 2012-4221

local variable	local variable	local variable	local variable

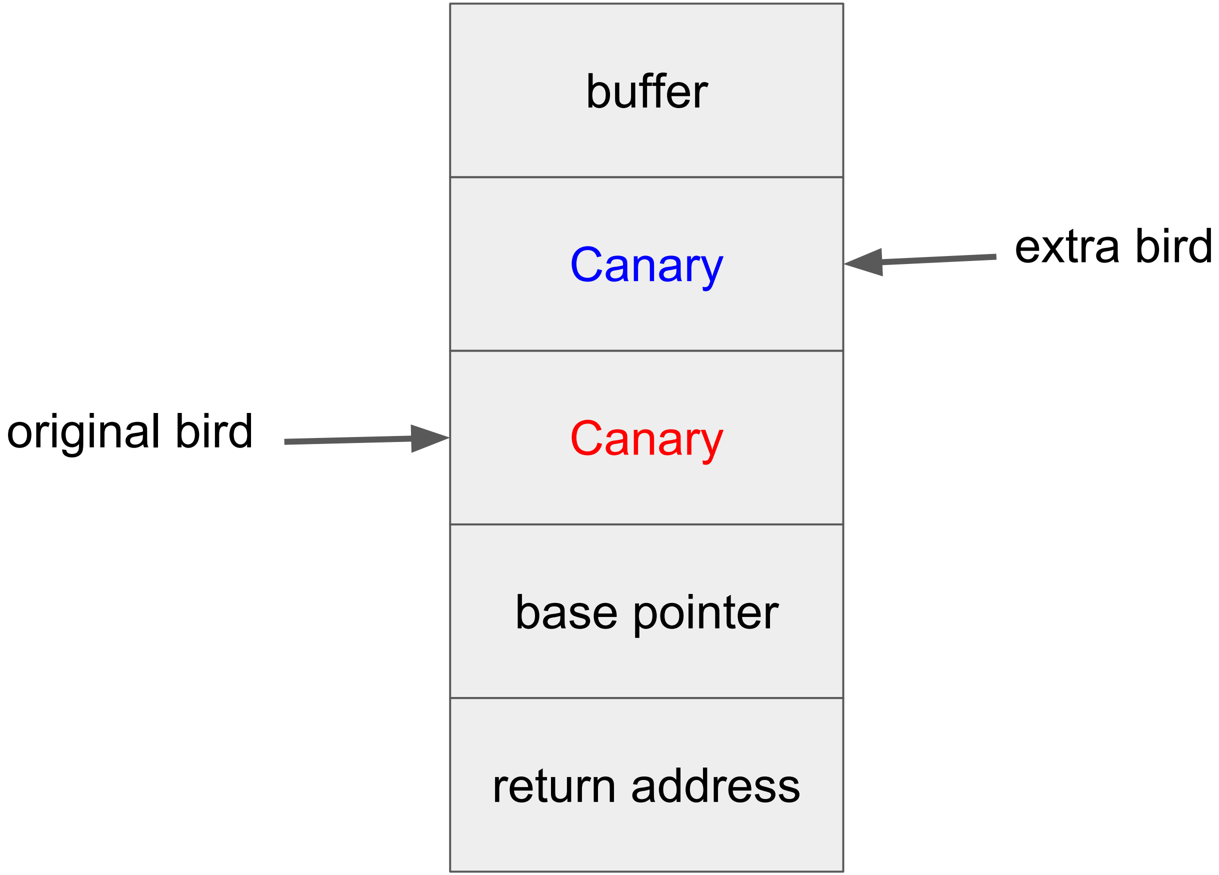
FP protection	Function padding	**Variable re-order**	Two birds



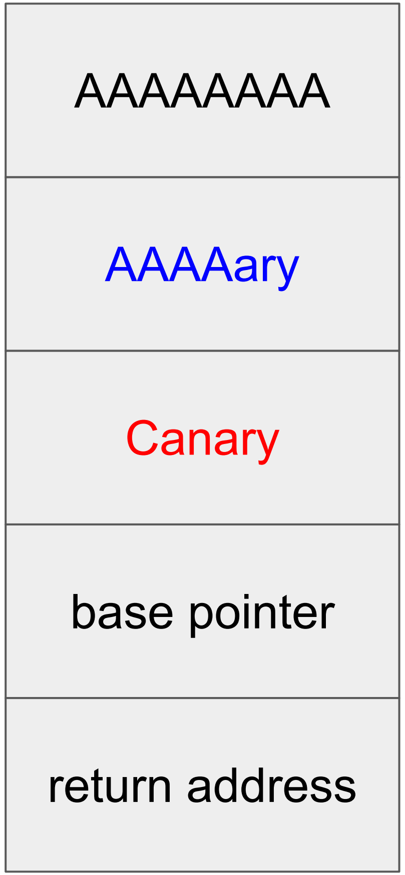
FP protection	Function padding	Variable re-order	**Two birds**



FP protection	Function padding	Variable re-order	**Two birds**



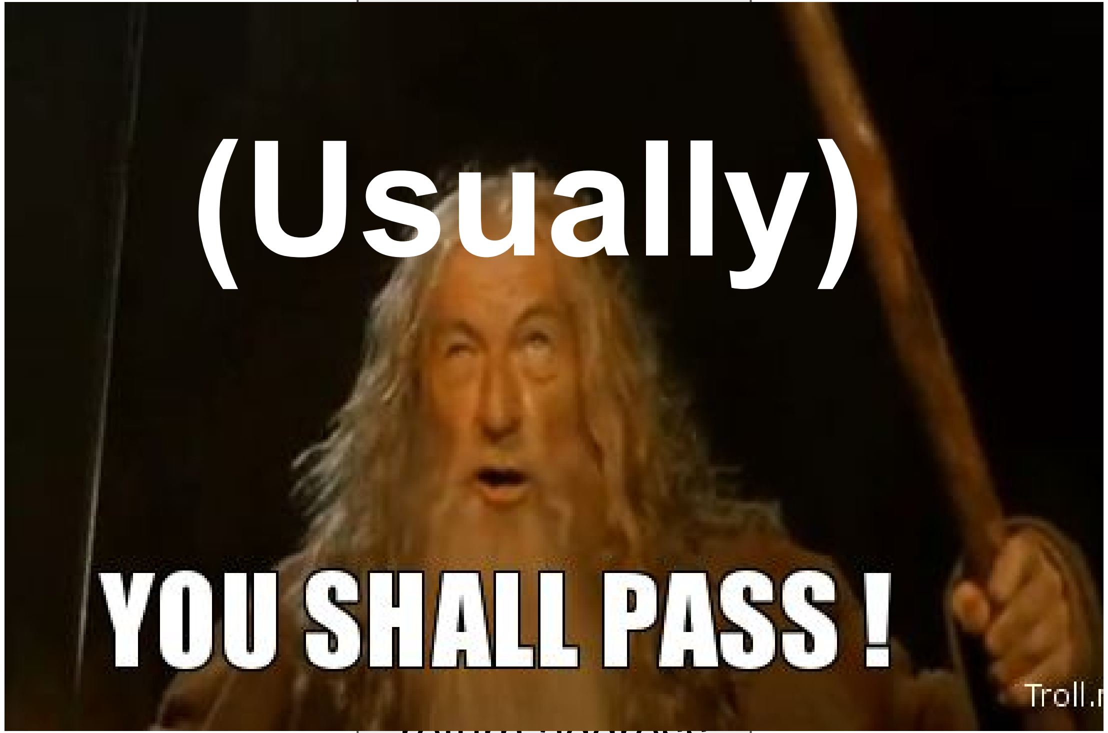
FP protection	Function padding	Variable re-order	Two birds



FP protection	Function padding	Variable re-order	Two birds

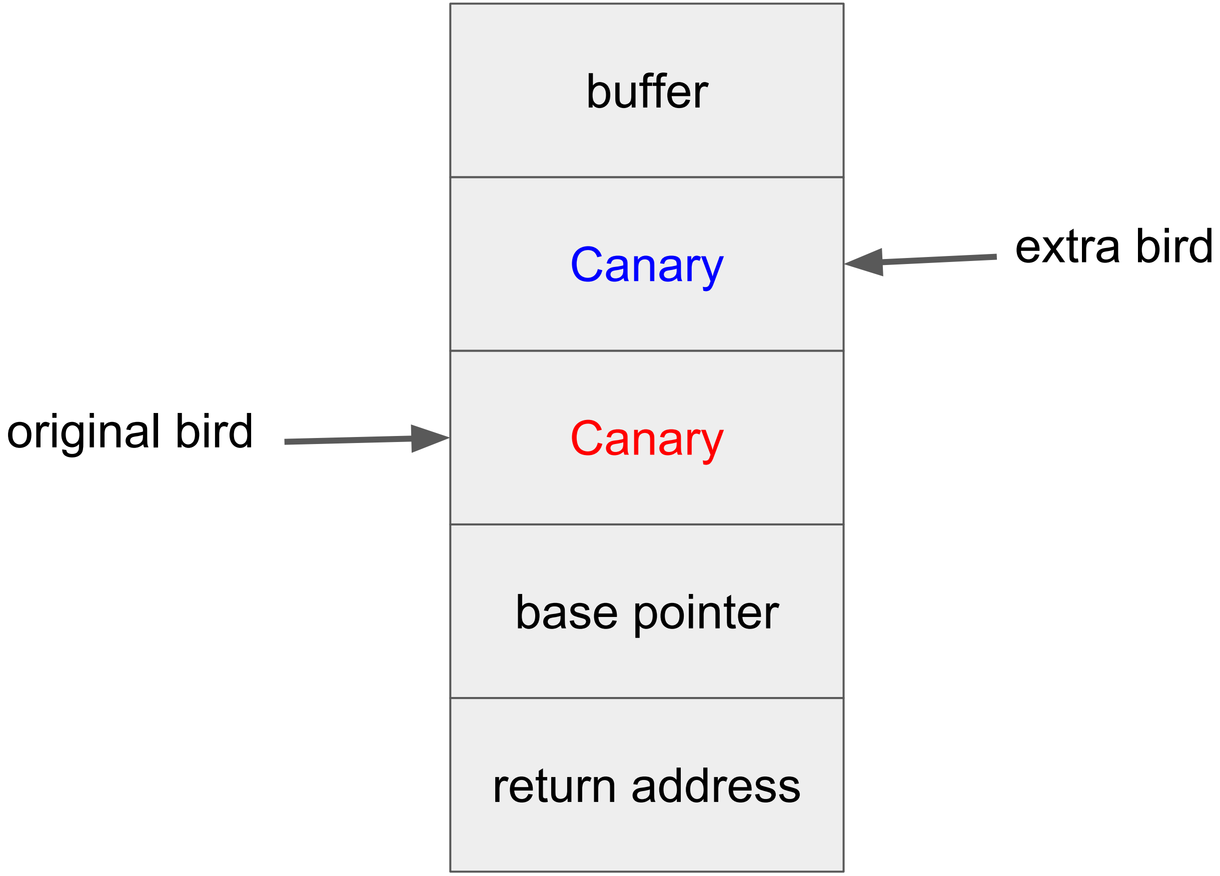
return address

FP protection	Function padding	Variable re-order	Two birds



return address

FP protection	Function padding	Variable re-order	Two birds



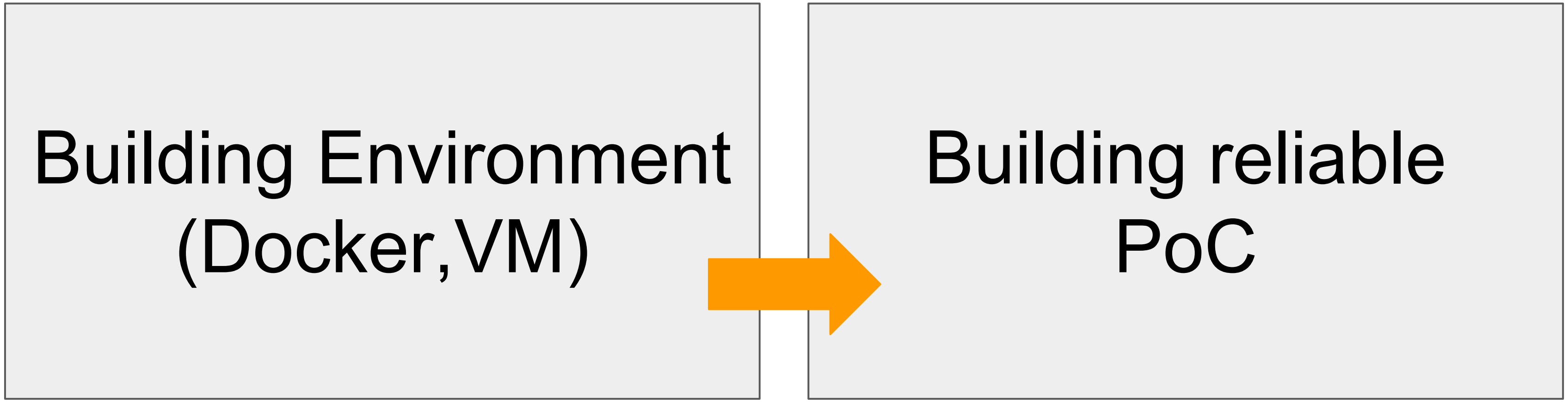
Summary & discussion

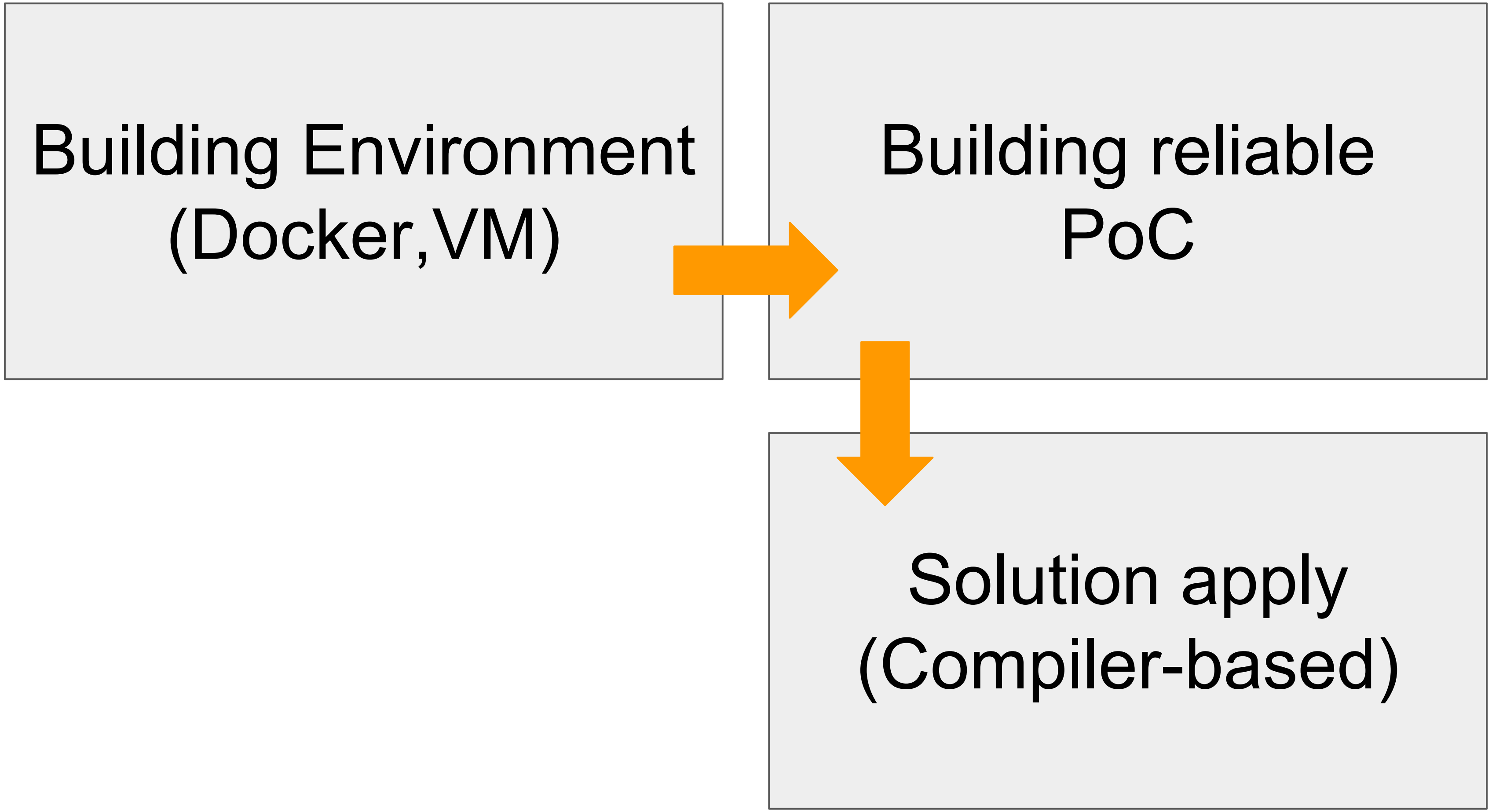
1. Any trade-off ?

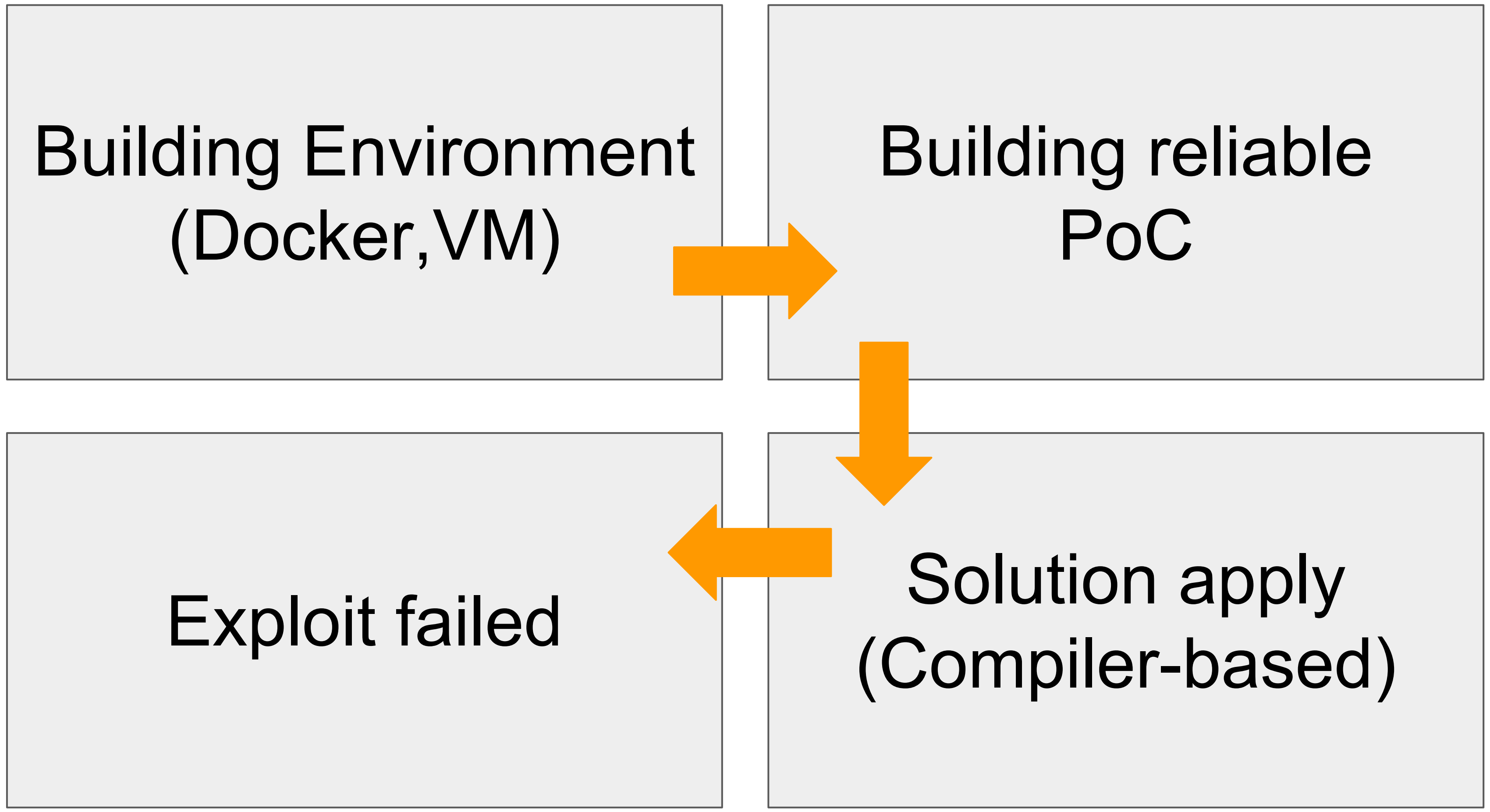
Summary & discussion

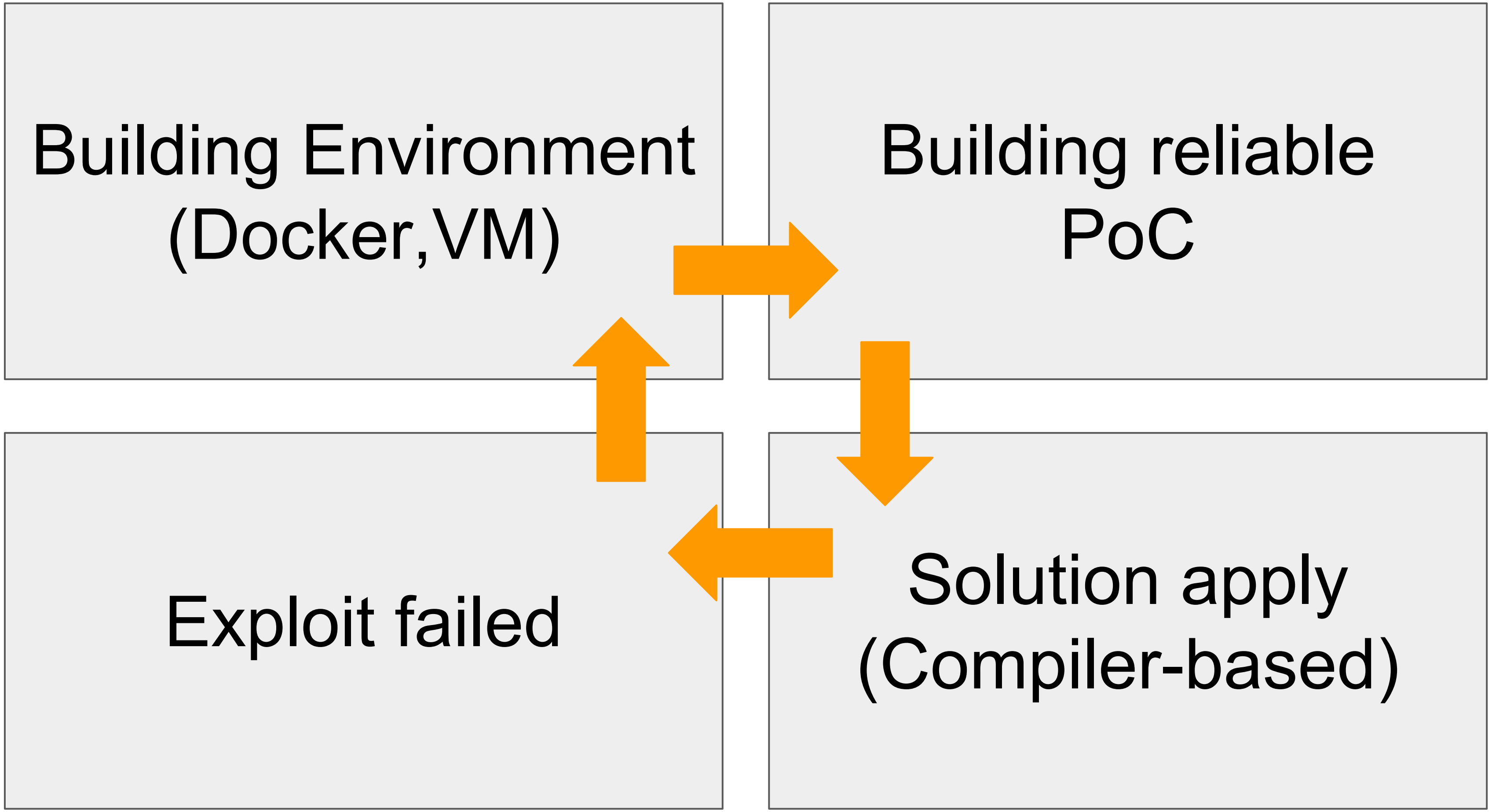
1. Any trade-off ?
2. Does it work ? How to proof ?

Building Environment
(Docker,VM)


Building Environment (Docker,VM)
Building reliable PoC


Building Environment (Docker,VM)
Building reliable PoC
Solution apply (Compiler-based)


Building Environment (Docker,VM)
Building reliable PoC
Solution apply (Compiler-based)
Exploit failed


Building Environment
(Docker,VM)
Building reliable
PoC
Solution apply
(Compiler-based)
Exploit failed

Summary & discussion

1. Any trade-off ?
2. Does it work ? How to proof ?
3. Seems perfect ?

Questions ?

https://fb.com/UCCU.Hacker