Exploit Mitigation using Multi-Variant Execution

Stijn Volckaert

University of California, Irvine

OWASP Belgium Chapter Meeting – 18 October 2016

Adobe Patches 10 Critical Vulnerabilities in Flash Player, Shockwave Player, and ColdFusion

Posted on April 9th, 2013 by Derek Erwin

Zero-Day Vulnerability Bypasses Apple's Security Featu ces: Updat

09 June 2012, 10:59

Adobe Flash update closes several critical holes

28 March 2016, 8:15 am EDT By Horia Ungureanu Tech Times

Adobe Issues Emergency Updates For Zero-Day Flaw in Flash Player

Memory corruption flaw is being exploited in the wild to distribute ransomware samples like Locky and Cerber.

Update Flash now! Adobe releases patch, fixing critical security holes

BY GRAHAM CLULEY POSTED 22 SEP 2015 - 09:24AM

stems.

Critical Adobe Flash bug under active attack currently has no patch

Exploit works against the most recent version; Adobe plans update later this week.

by Dan Goodin - Jun 14, 2016 12:50pm PDT

Firefox 45 browser update patches 22

(Another) Update To Adobe Flash Addresses Latest 0-Day Vulnerability

April 15, 2011

By Charlie Osborne for Zero Day | March 10, 2016 -- 10:01 GMT (02:01 PST) | Topic: Security

Adobe Releases Security Update for 19 'Critical' Vulnerabilities in Flash Player

DAVID BISSON

DEC 29, 2015 | LATEST SECURITY NEWS

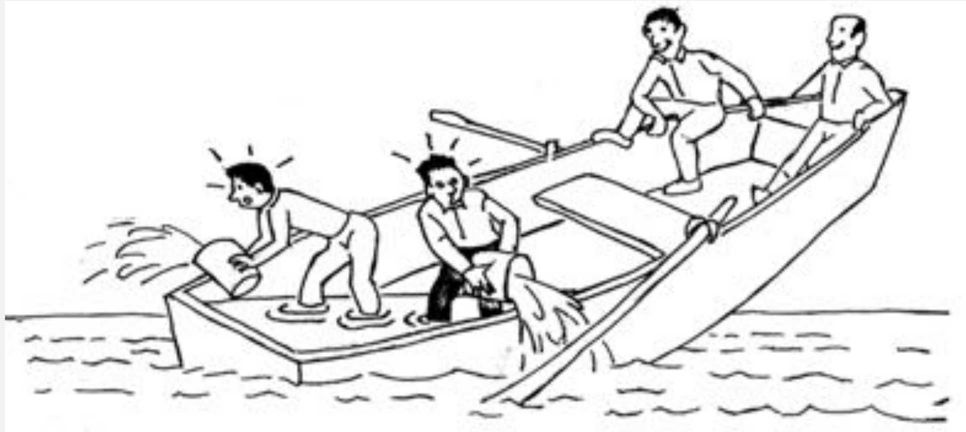
Adobe Fixes 18 Vulnerabilities in Flash Player

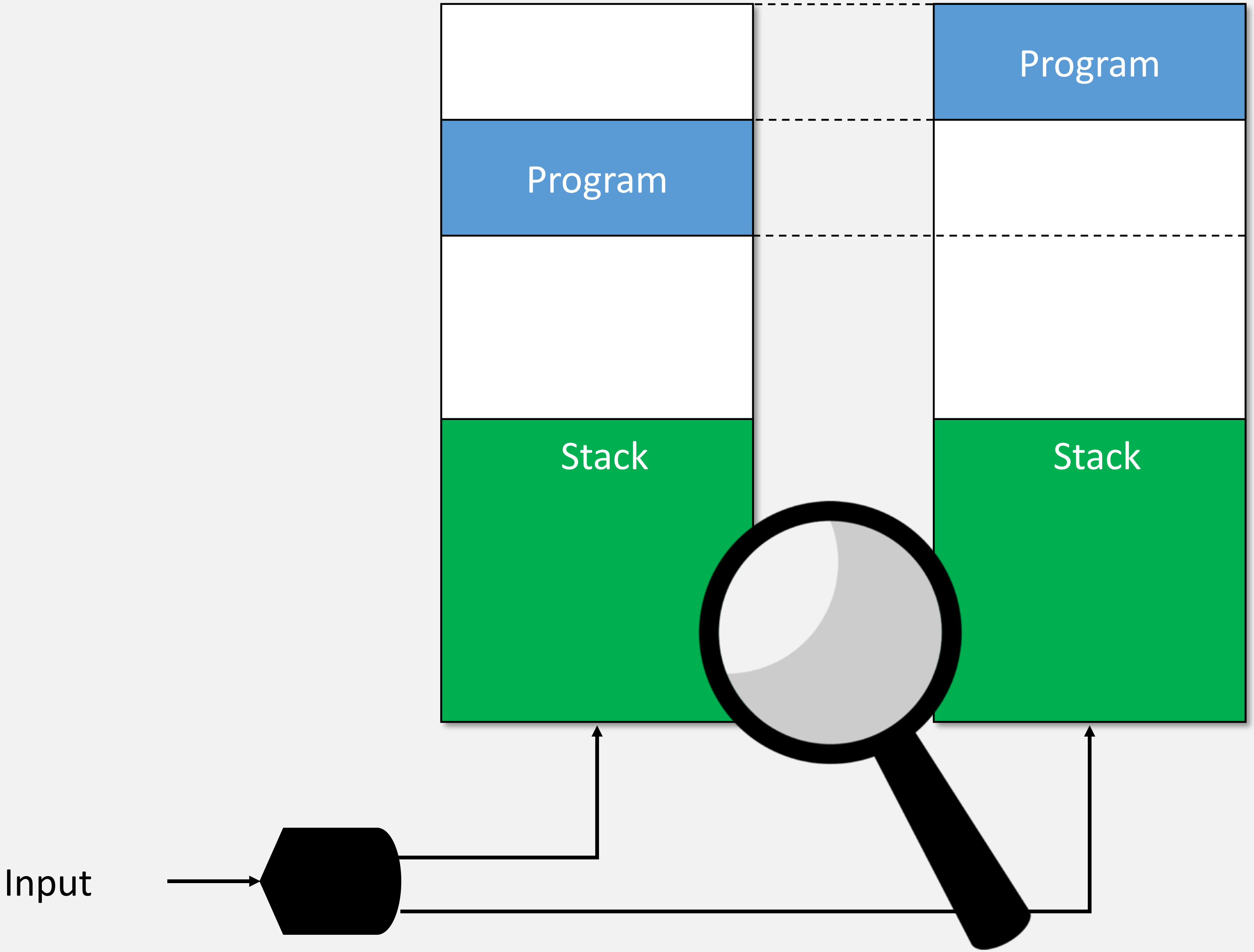
By Eduard Kovacs on November 12, 2014

Possible Solutions

- ~~Type-Safe Languages (e.g. Rust)~~

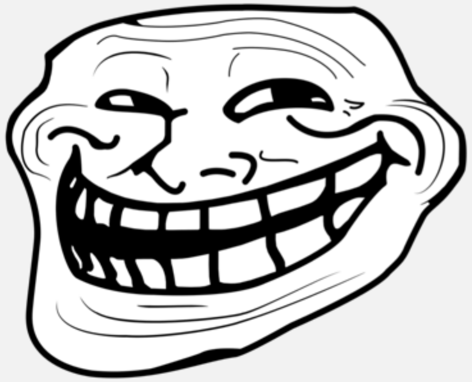
- ~~Mitigations:~~
  - ~~Integrity-Based (e.g. CFI)~~
  - ~~Randomization-Based (e.g. ASLR)~~
- Multi-Variant Execution Environments (MVEEs)

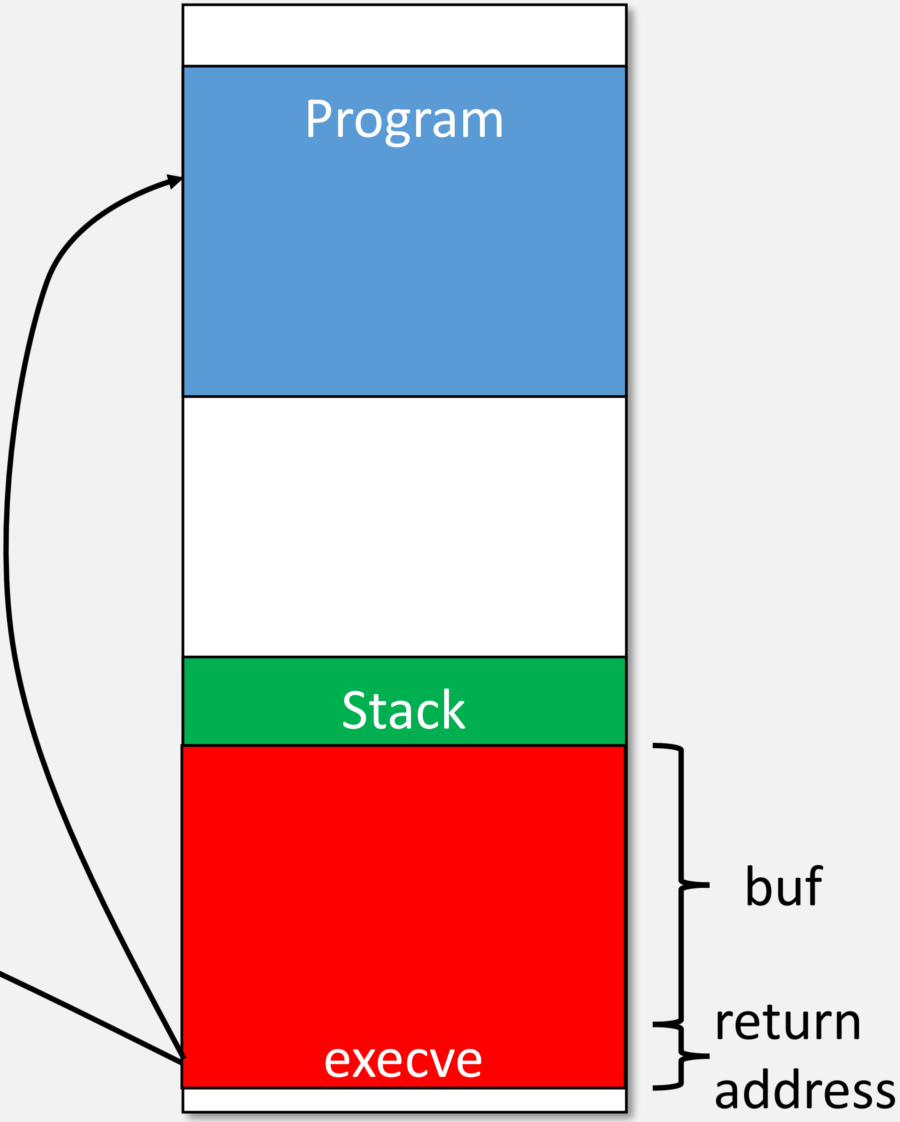


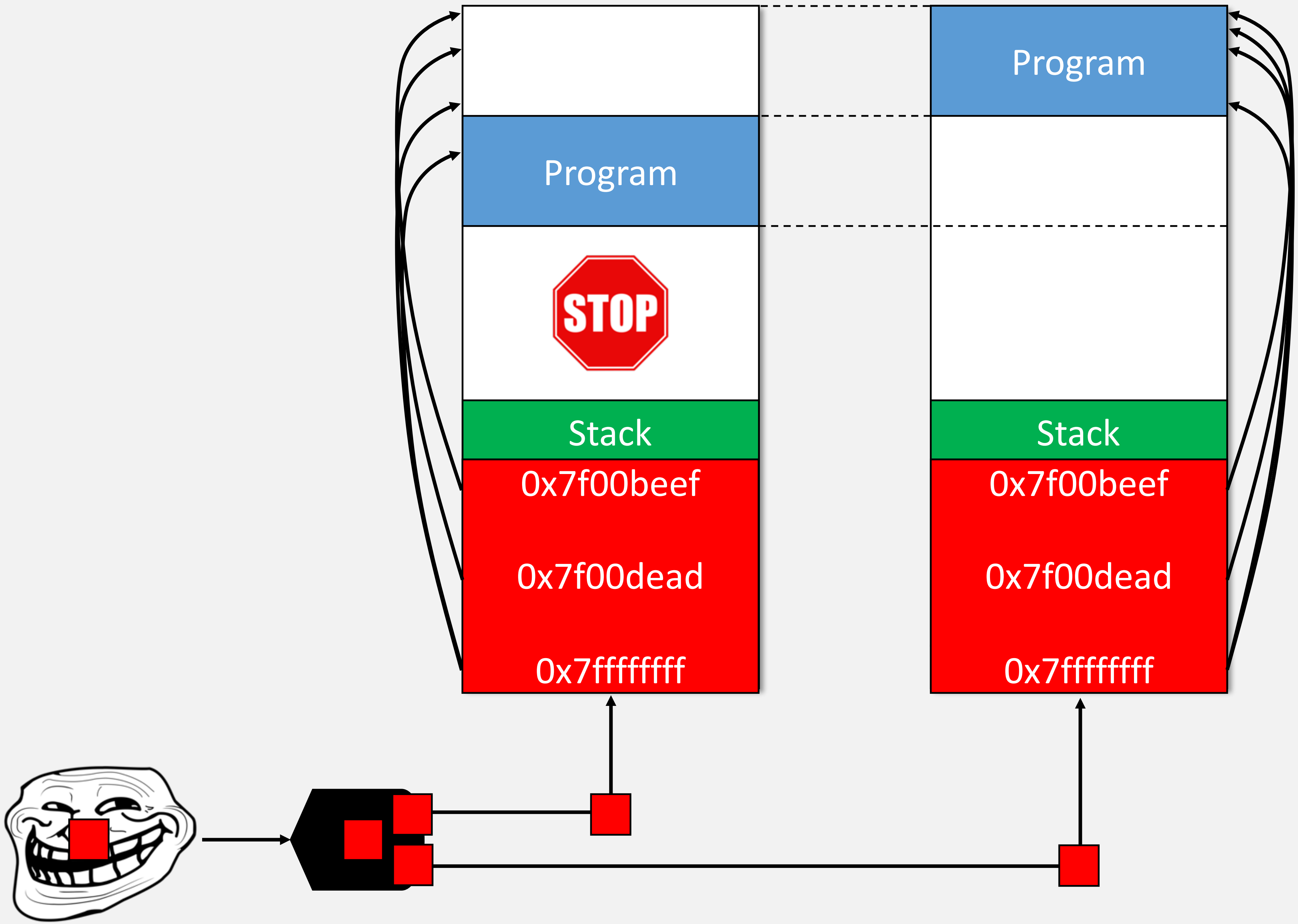

Program
Program
Stack
Stack
Input

Memory Corruption Attacks

```c
void foo() {
    char buf[256];
    gets(buf);
    printf(“%s”, buf);
}

int main(int argc, char** argv) {
    foo();
    return 0;
}
```

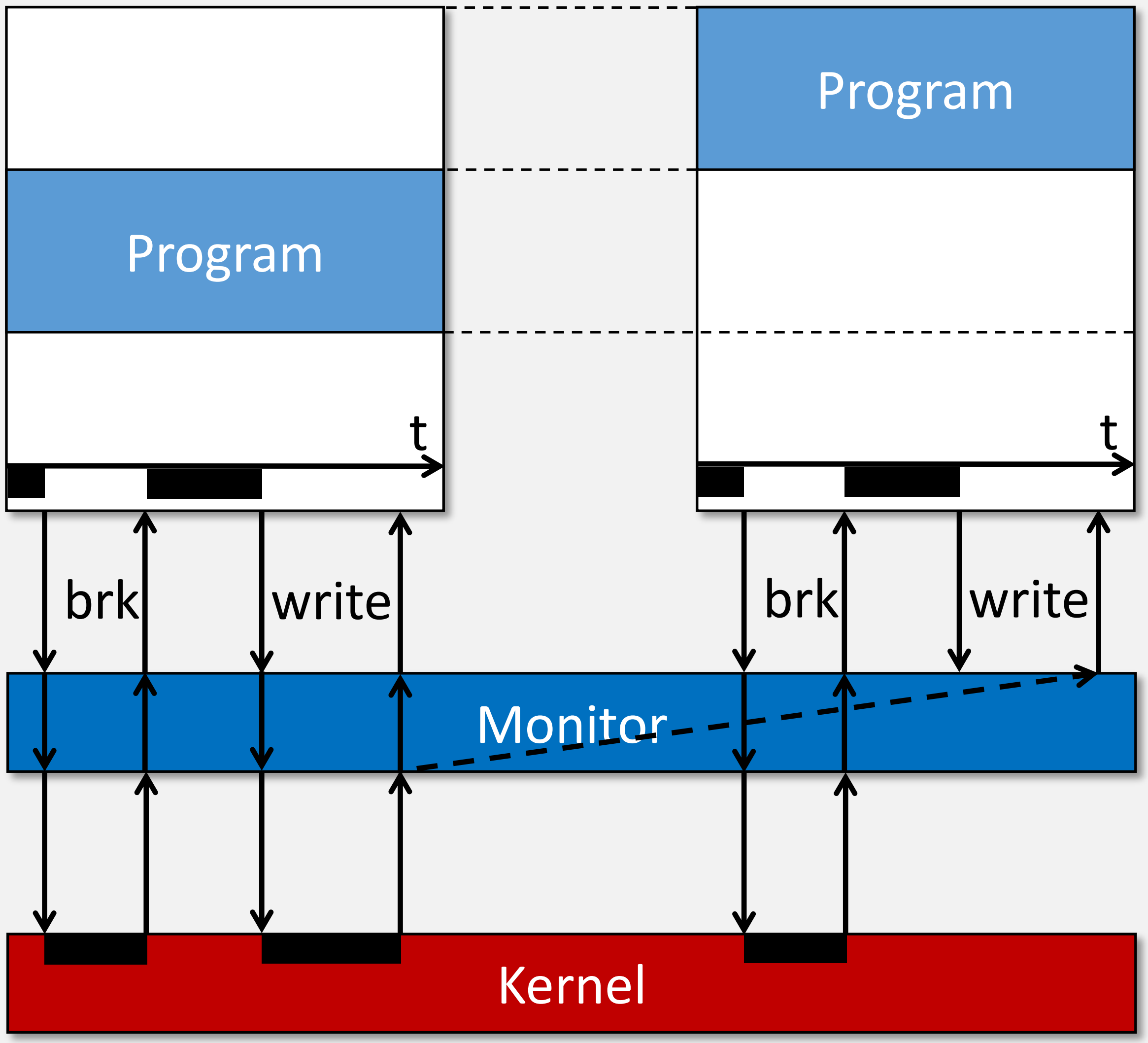






Program
STOP
Stack
0x7f00beef
0x7f00dead
0x7fffffff
Program
Stack
0x7f00beef
0x7f00dead
0x7fffffff

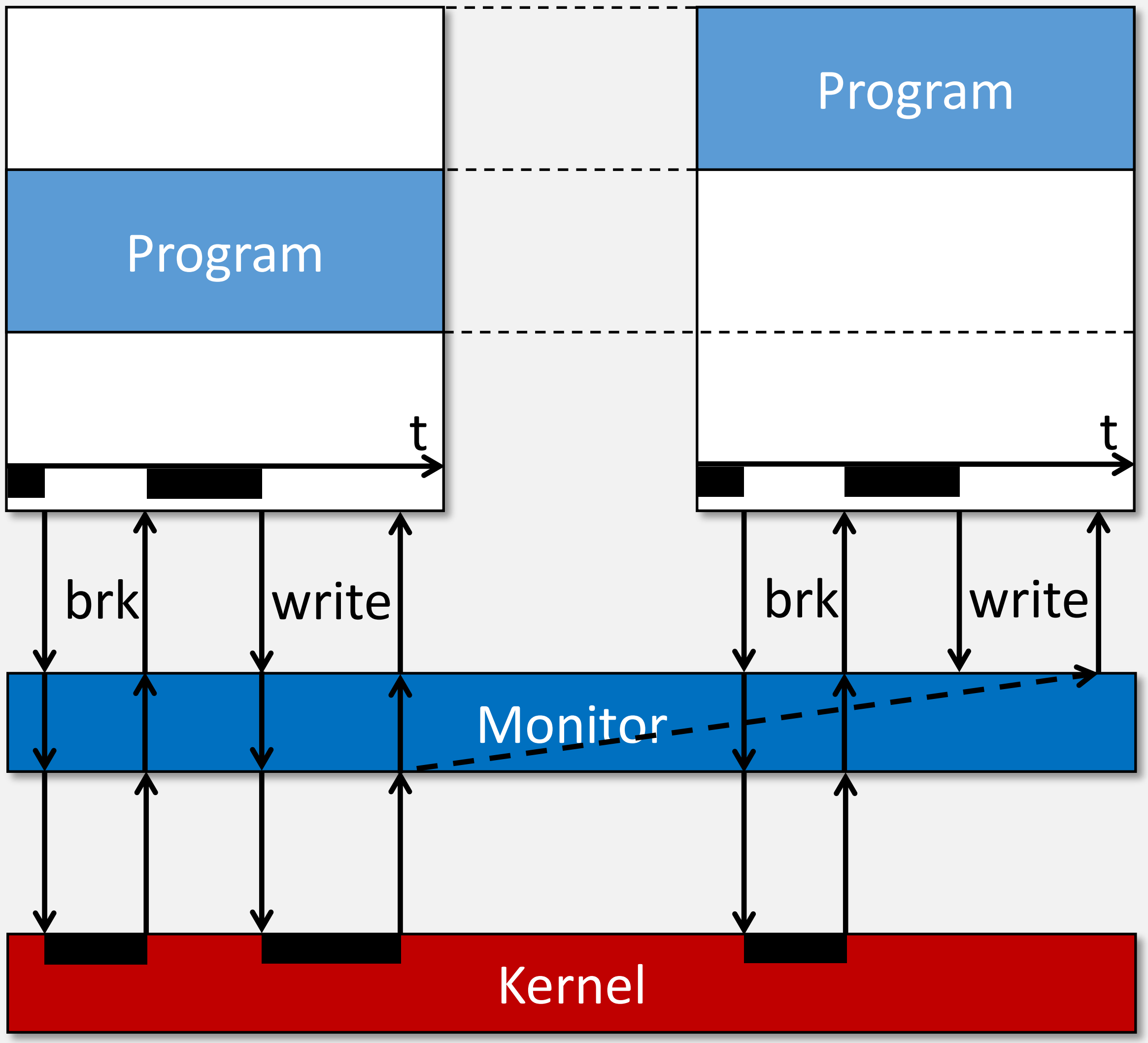
Multi-Variant Execution Environments (MVEEs)



In a nutshell:

- Run multiple program variants in parallel
- Variant system calls executed in lock-step
- Suspend them at every system call
- Compare system call numbers/arguments
- Master/Slave replication for I/O

Performance Considerations

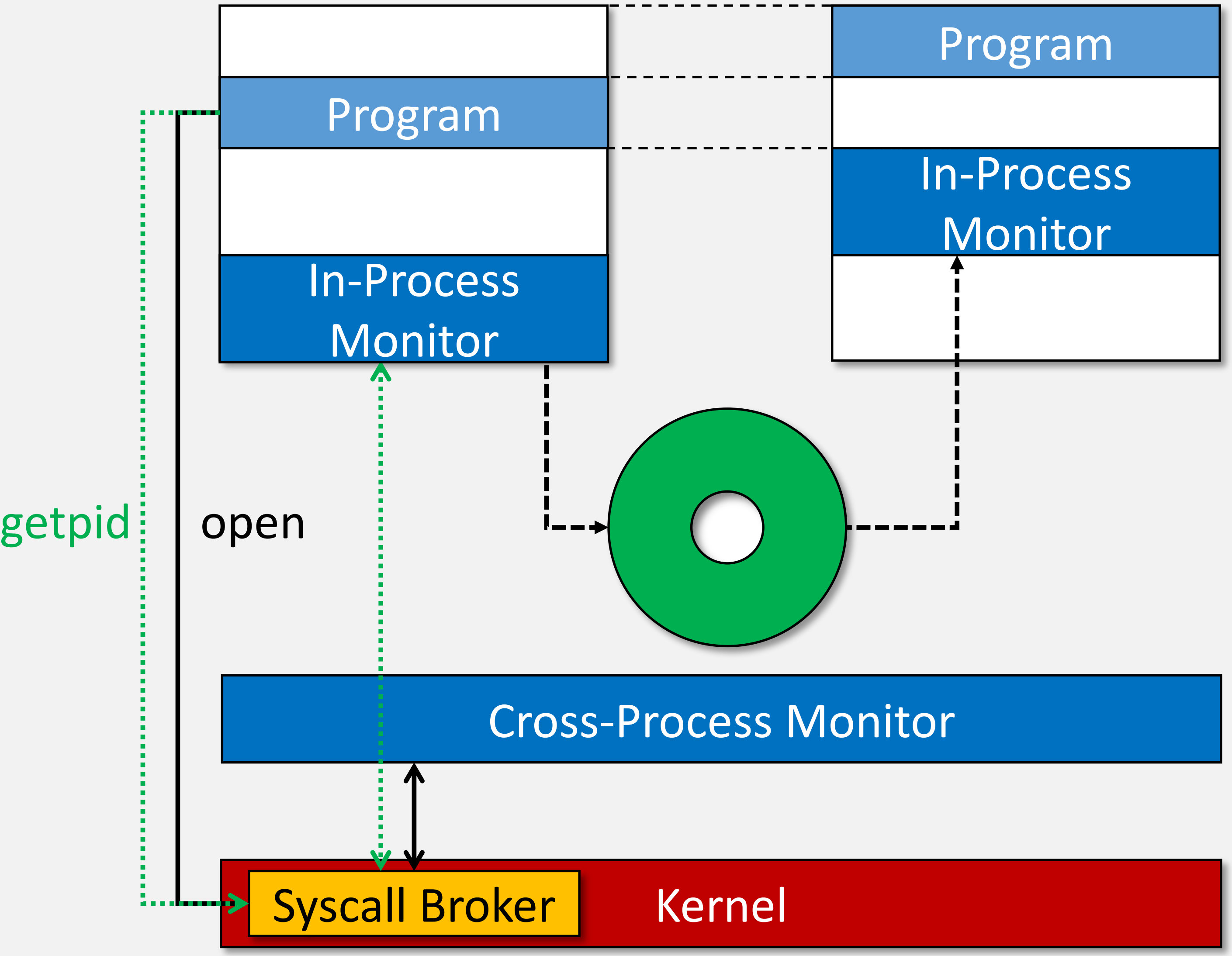


Programs can execute at **native speed** (assuming you have enough idle CPU cores and memory bandwidth)

The total system load does **not** increase by a factor of **n**
(with n the number of variants)

BUT there are some problems!

Slow System Call Interception



Split-Monitor Design:

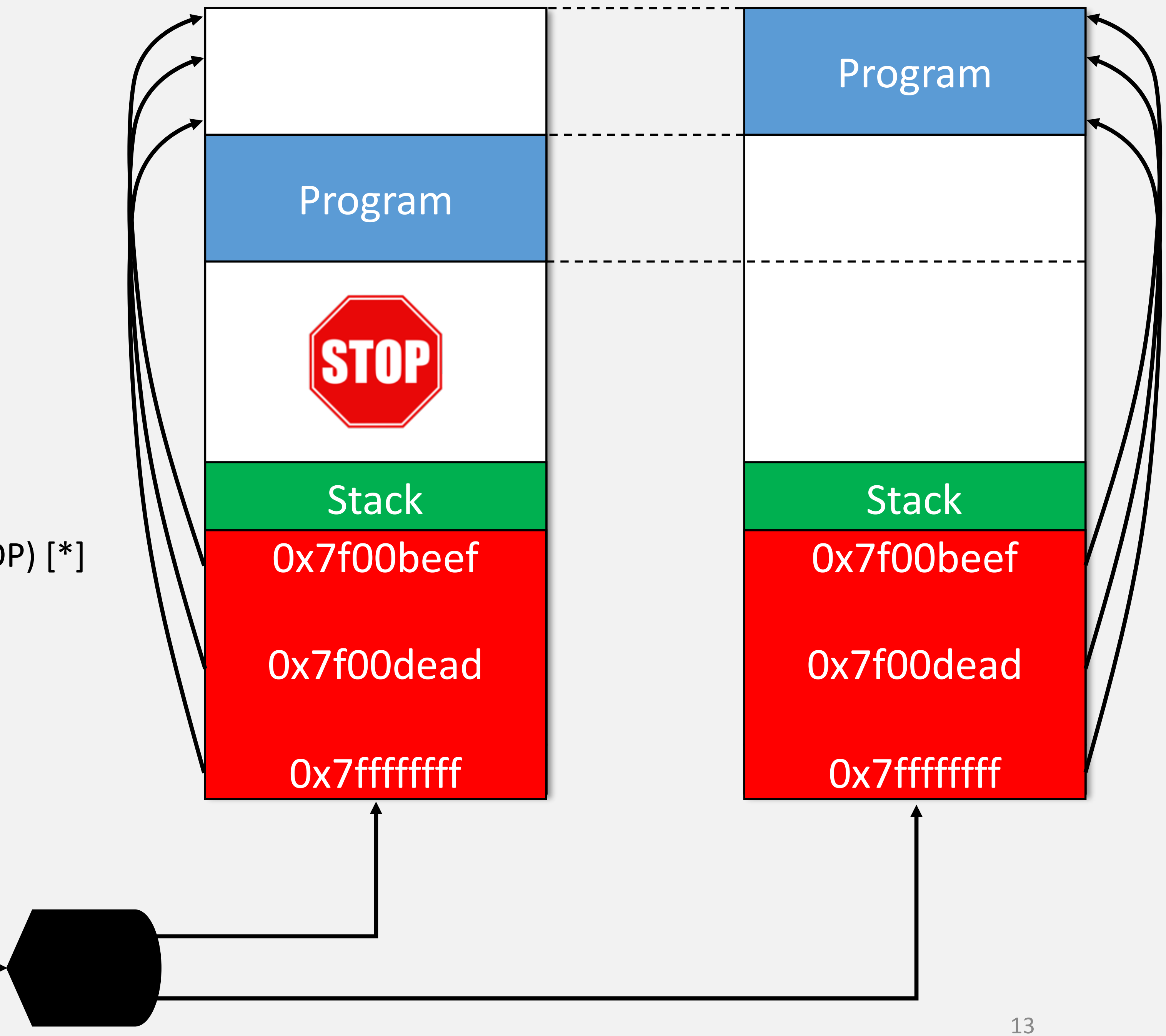
- Handle security-sensitive system calls in **Cross-Process Monitor (CP-MON)**
- Handle non-sensitive system calls in **In-Process Monitor (IP-MON)**

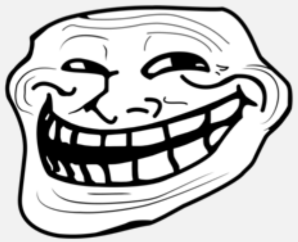
Code Reuse

Protects against:

- return-to-libc (RILC)
- return-oriented programming (ROP)
- jump-oriented programming (JOP)
- just-in-time code reuse (JIT-ROP) [*]
- counterfeit object-oriented programming (COOP) [*]
- …

[*] Requires eXecute-only memory support





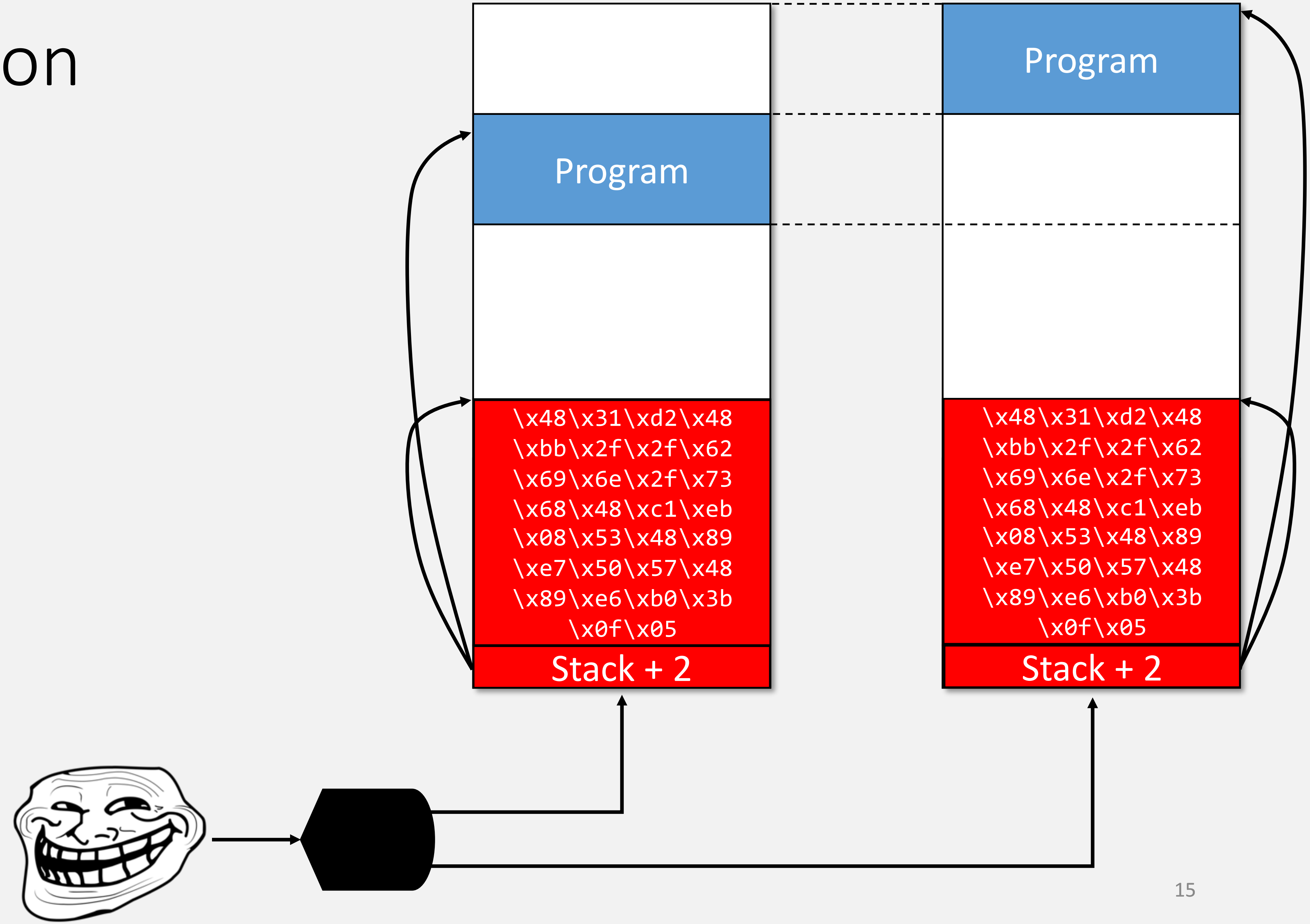
Code Injection

```
\x48\x31\xd2\x48\xbb
\x2f\x2f\x62\x69\x6e
\x2f\x73\x68\x48\xc1
\xeb\x08\x53\x48\x89
\xe7\x50\x57\x48\x89
\xe6\xb0\x3b\x0f\x05
```

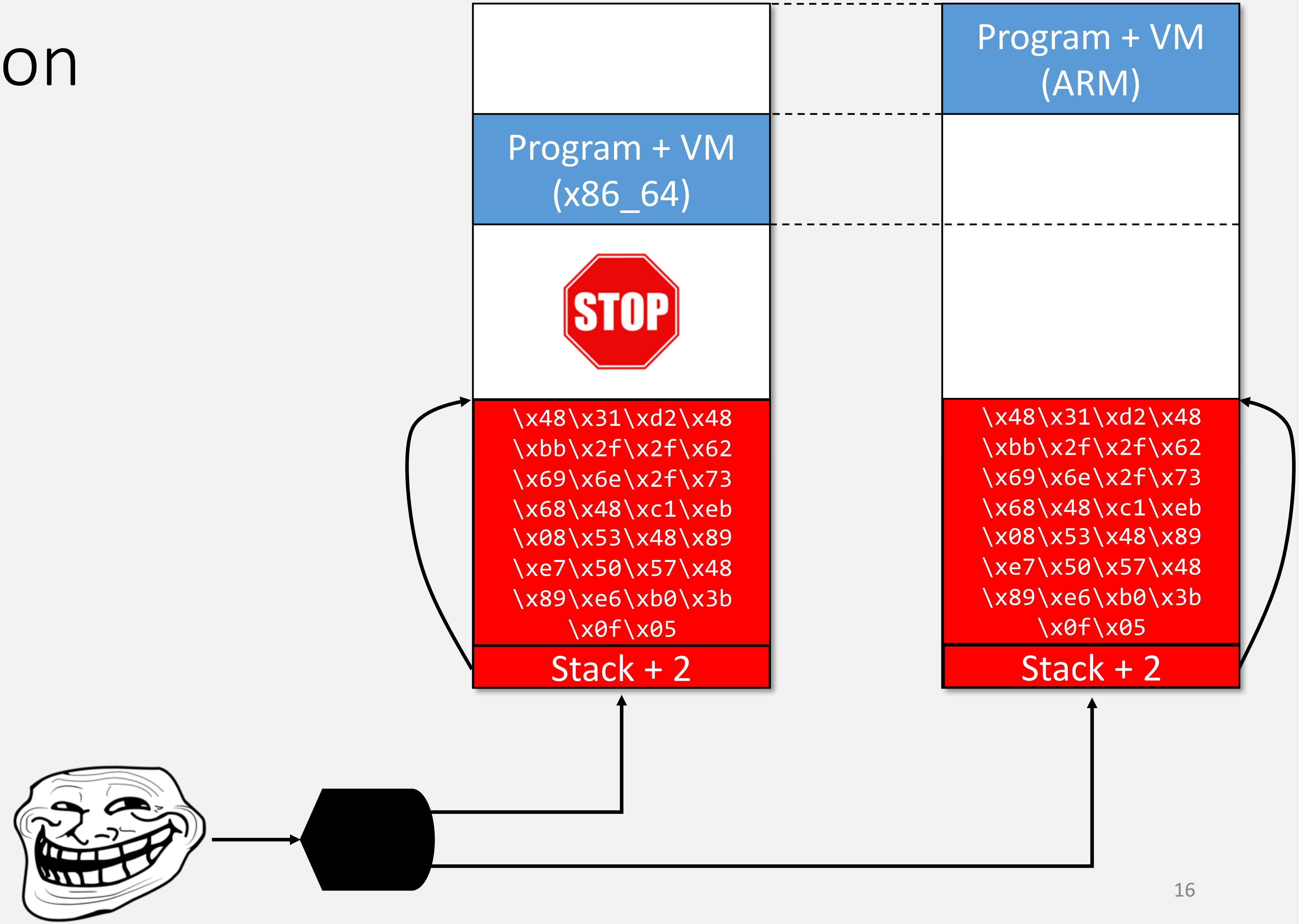
```
xor     rdx, rdx
mov     qword rbx, '//bin/sh'
shr     rbx, 0x8
push    rbx
mov     rdi, rsp
push    rax
push    rdi
mov     rsi, rsp
mov     al, 0x3b
syscall
```

execve("/bin/sh", ["/bin/sh"], NULL)

Code Injection

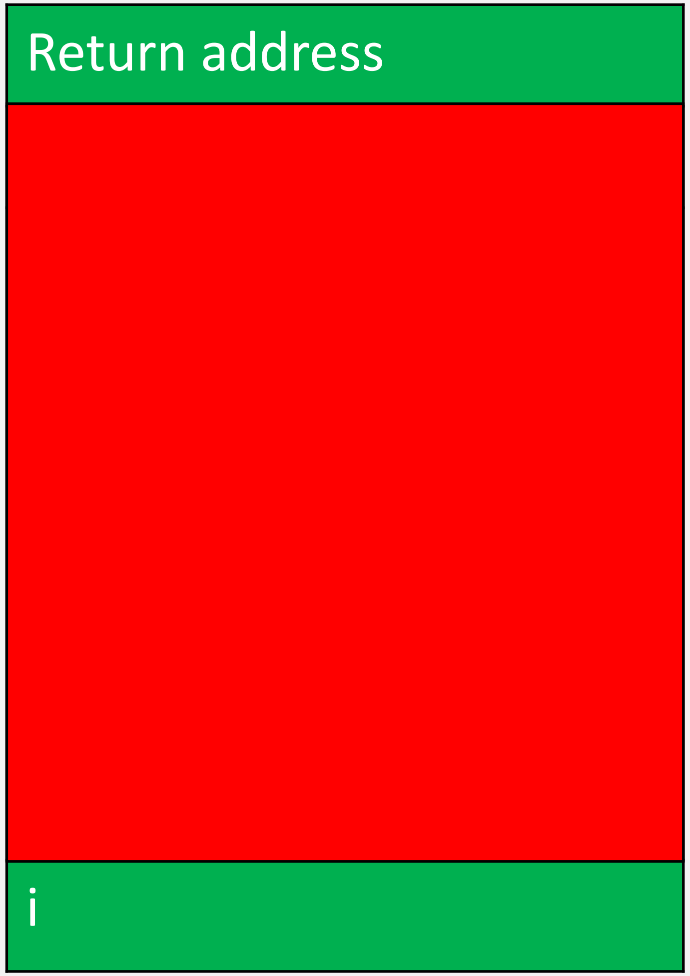


Code Injection



Non-Control Data Attacks

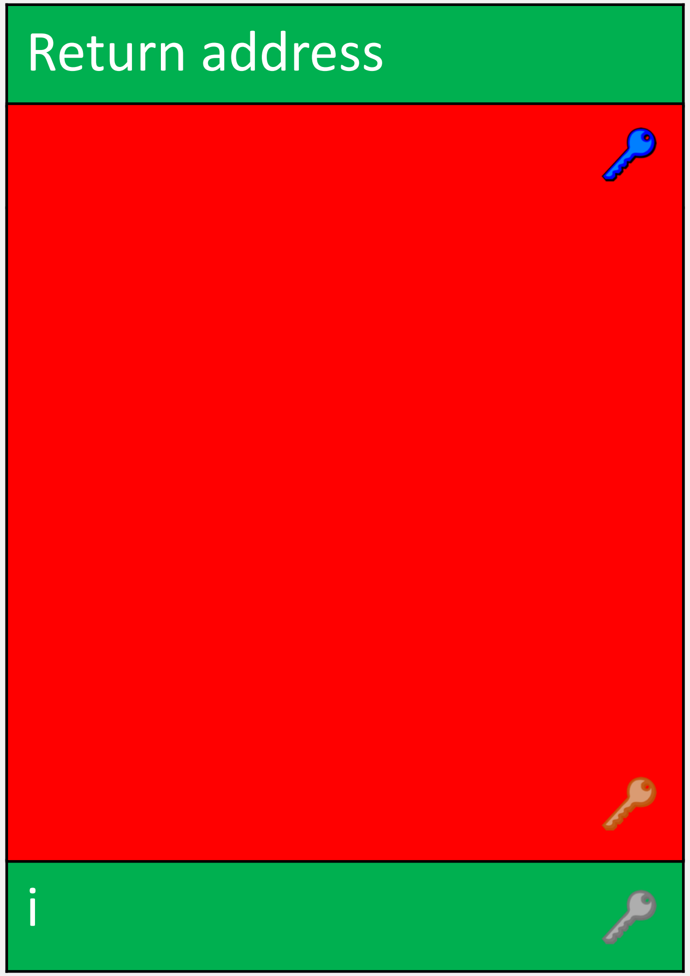
```
void ProcessConnection(connection* c) {
  cred_t user;
  char message[1024];
  int i = 0;

  auth_user(&user, c);
  while (!end_of_message(c))
   message[i++] = get_next_char(c);

  seteuid(user.user_id);
  ExecuteRequest(message);
}
```



Non-Control Data Attacks

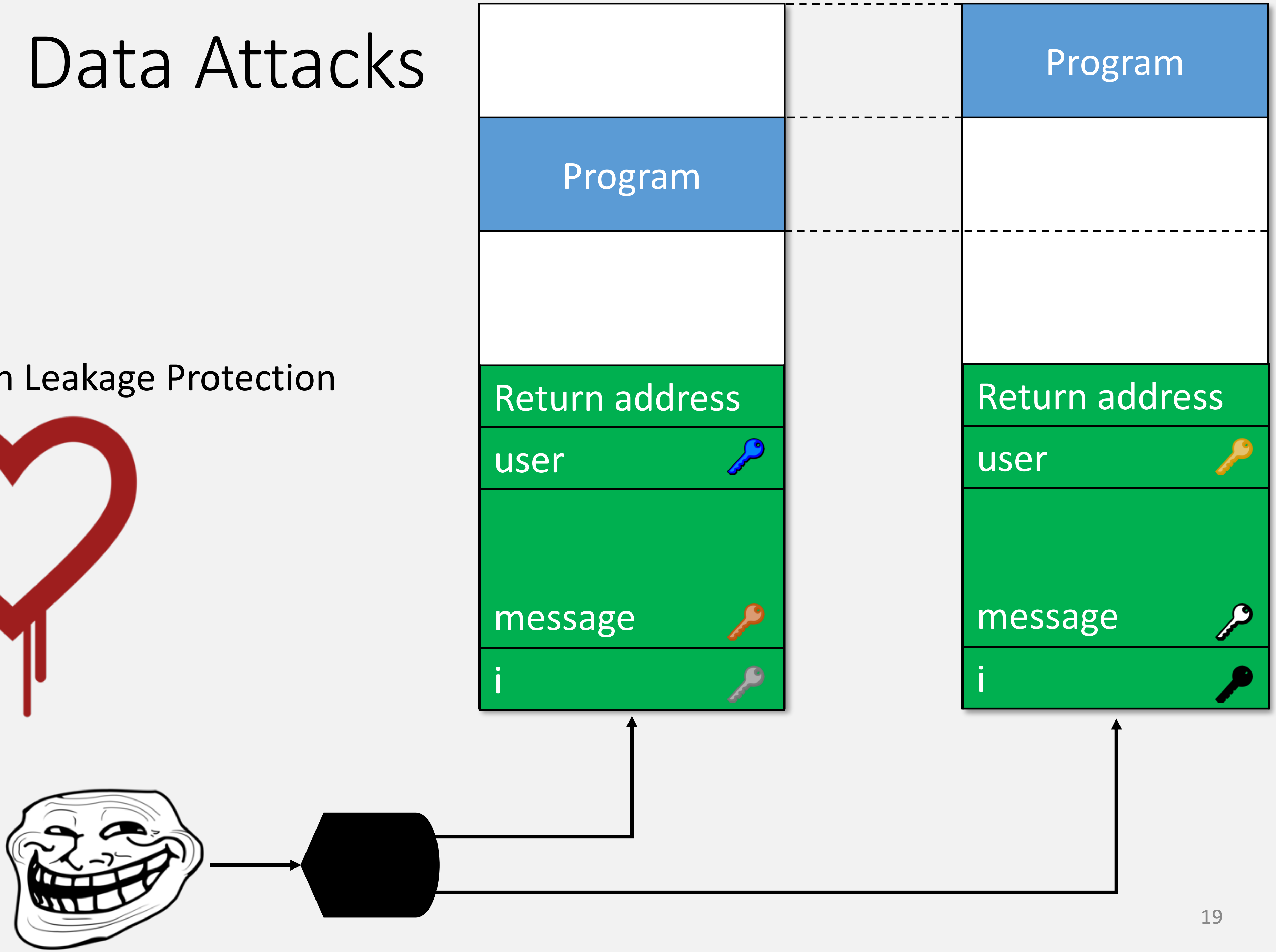
```
void ProcessConnection(connection* c) {
  cred_t user;
  char message[1024];
  int i = encrypt(0, );

  auth_user(&user, c);
  while (!end_of_message(c)) {
    register int tmp = decrypt(i, );
    message[tmp] = encrypt(get_next_char(c), );
    i = encrypt(tmp + 1, );
  }

  seteuid(decrypt(user.user_id, ));
  ExecuteRequest(message);
}
```

```
void ProcessConnection(connection* c) {
  cred_t user;
  char message[1024];
  int i = 0;

  auth_user(&user, c);
  while (!end_of_message(c))
    message[i++] = get_next_char(c);

  seteuid(user.user_id);
  ExecuteRequest(message);
}
```



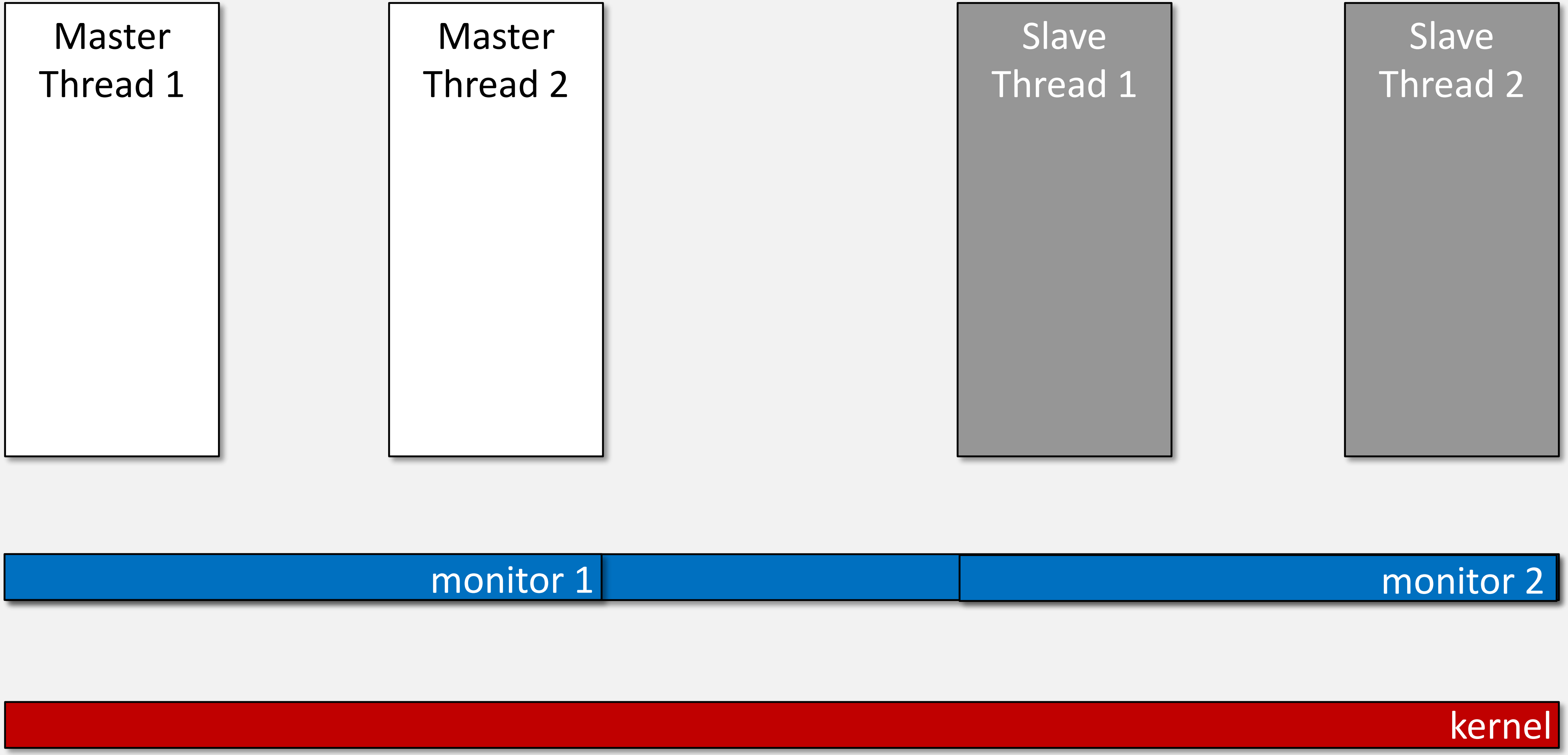
Non-Control Data Attacks

BONUS: Information Leakage Protection

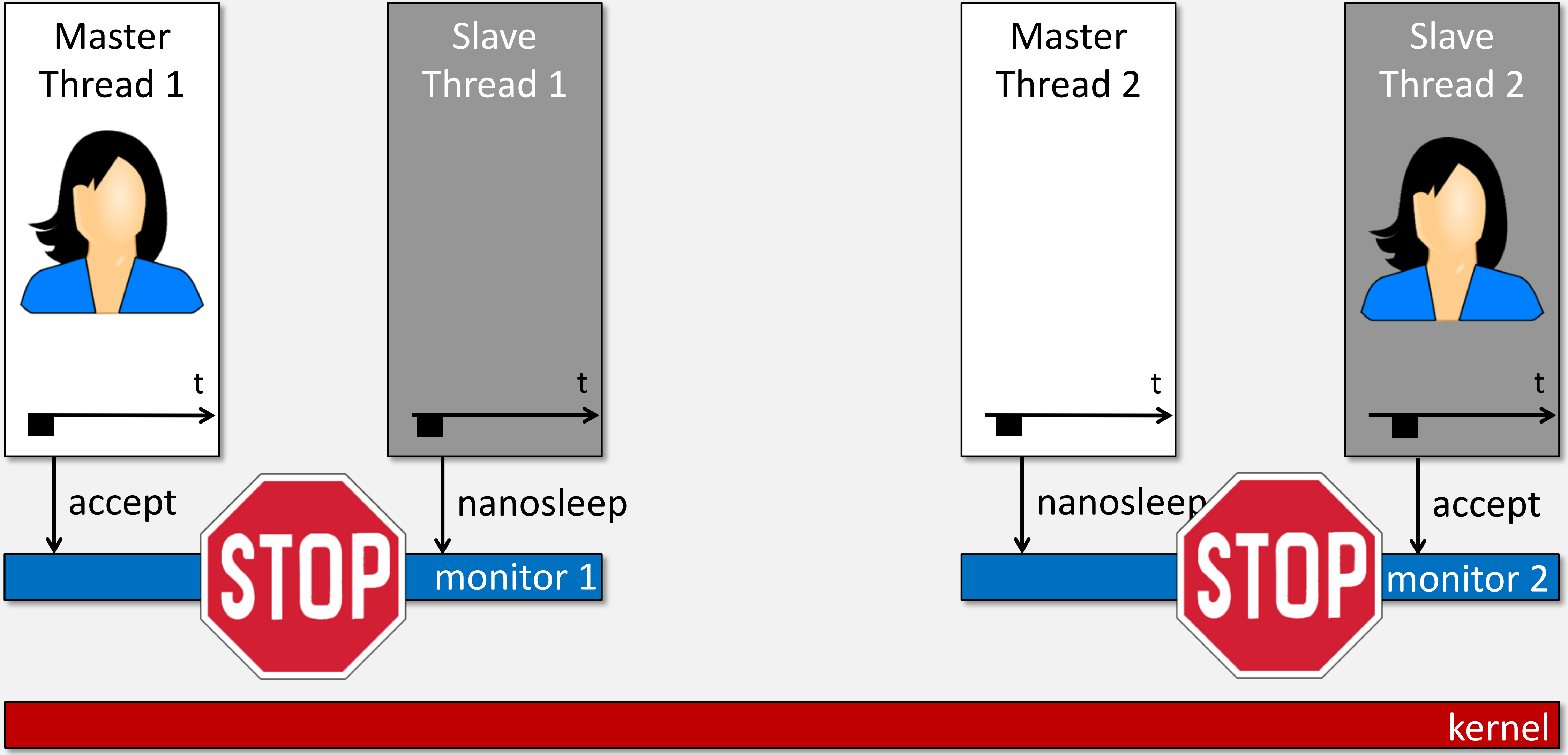



PROTECT ALL
THE PROGRAMS

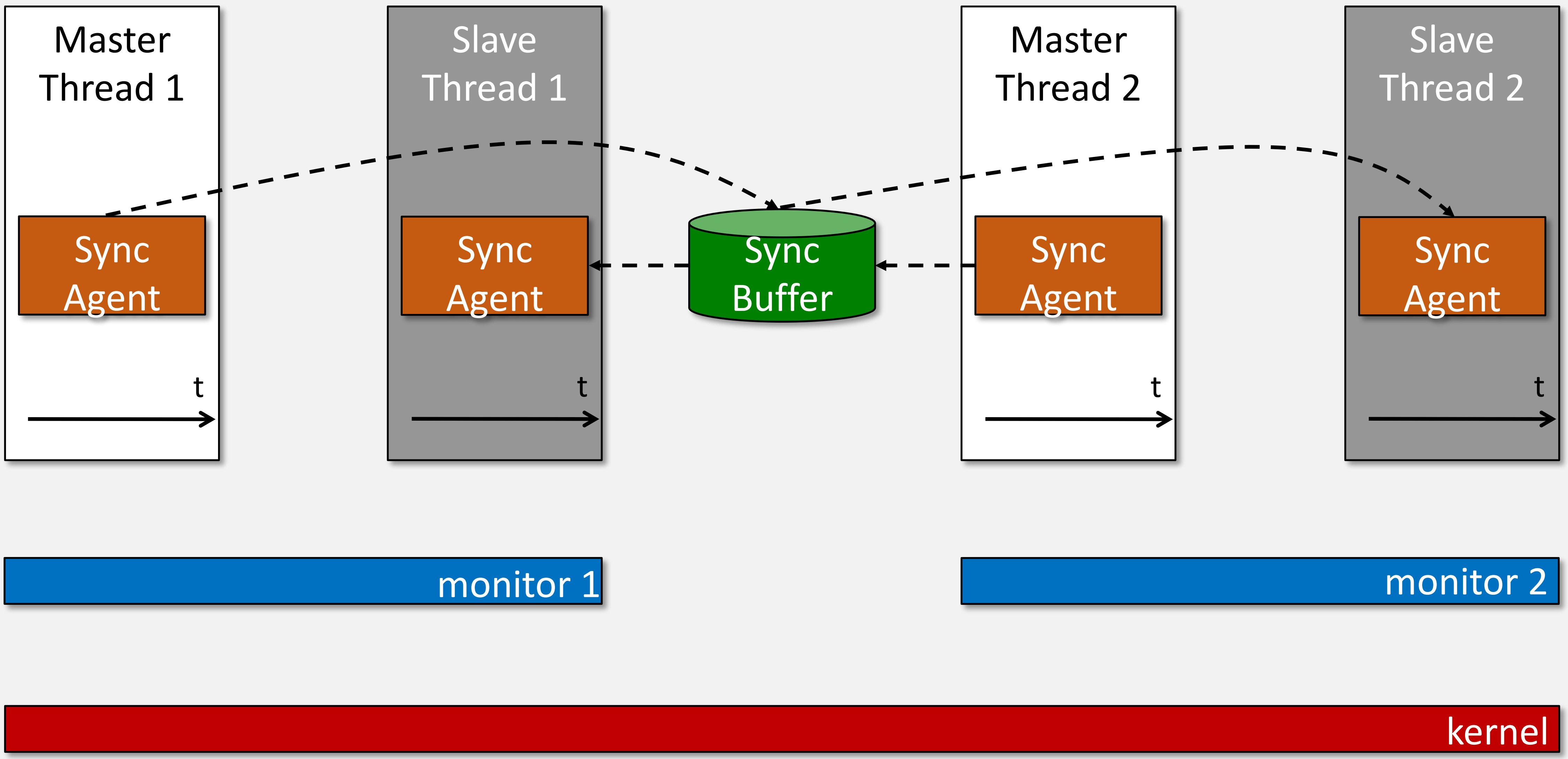
Multithreading



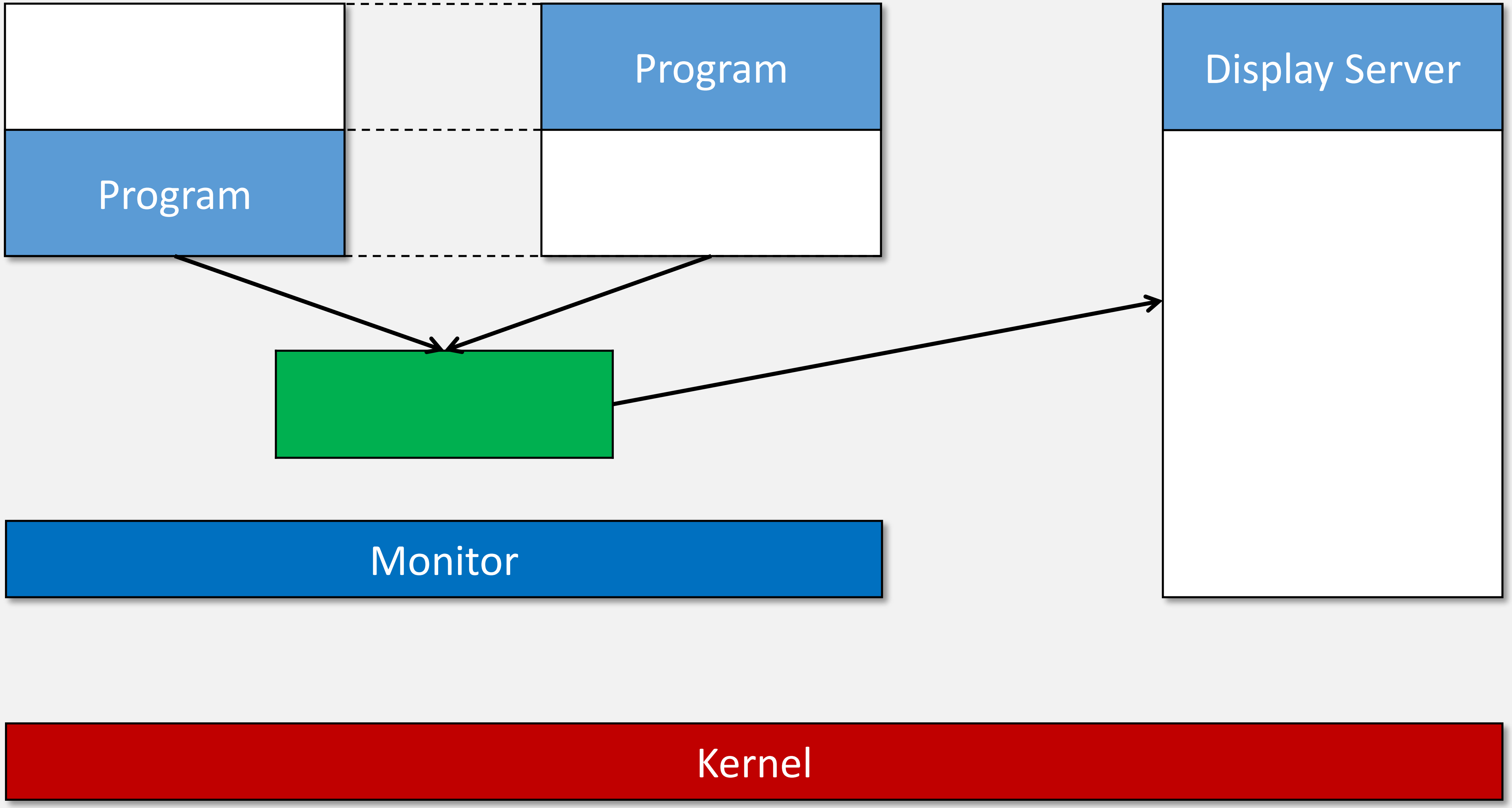
Multithreading



Multithreading



Shared Memory



Other problems

- Data races
- vdso/vsyscall pages
- RDTSC/RDTSCP
- Address Dependence

https://github.com/stijn-volckaert/ReMon