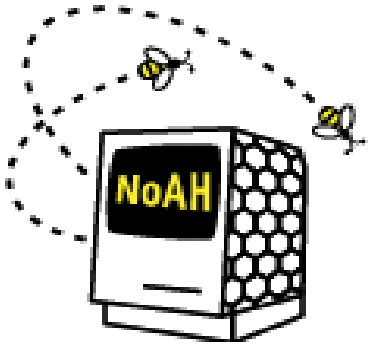


Argos: The High-Interaction back-end Honeypot of NoAH

Asia Slowinska
Georgios Portokalidis
Herbert Bos
Vrije Universiteit Amsterdam

Attacks addressed in the talk

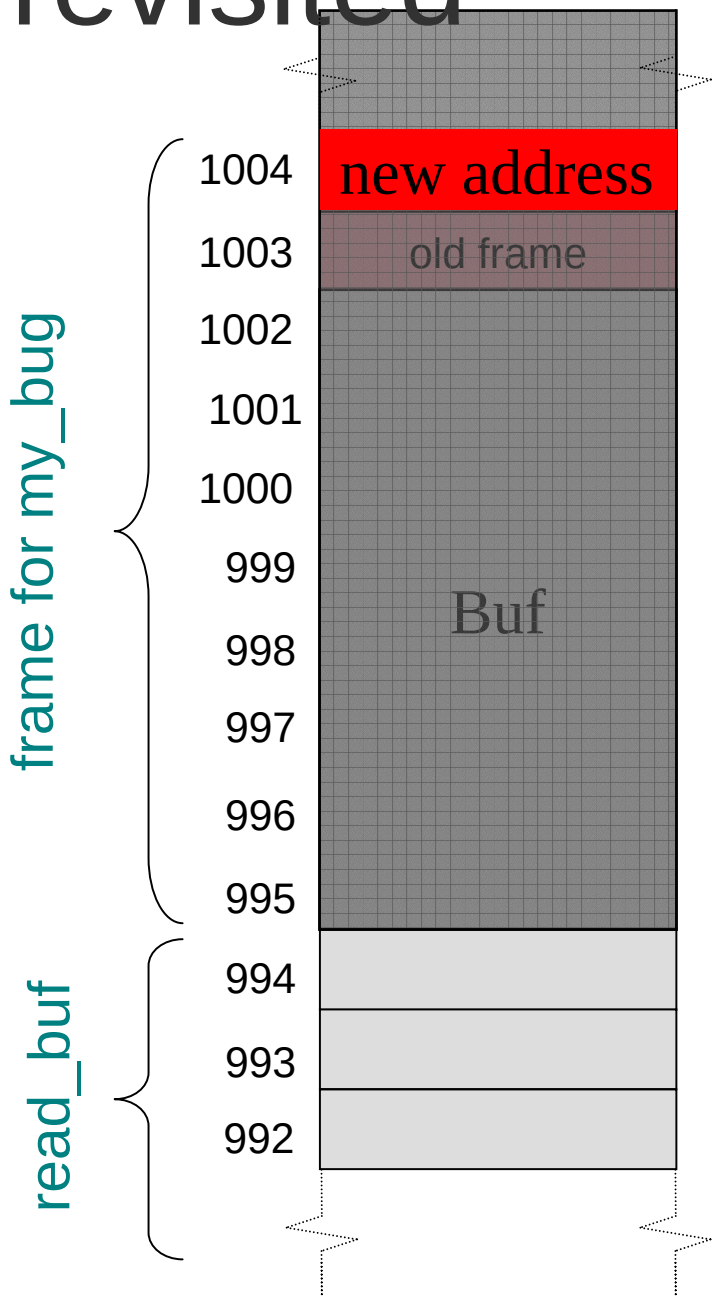
- ✓ self-propagating attacks that divert the OS flow to execute their own instructions,
  - ✓ stack smashing attacks,
  - ✓ heap corruption attacks,
  - ✓ format string attacks

- ✘ javascript-, SQL-injections
- ✘ social engineering techniques
  - ✘ phishing,
  - ✘ malware downloaded by user, e,g., a cool screensaver

Stack smashing attack revisited

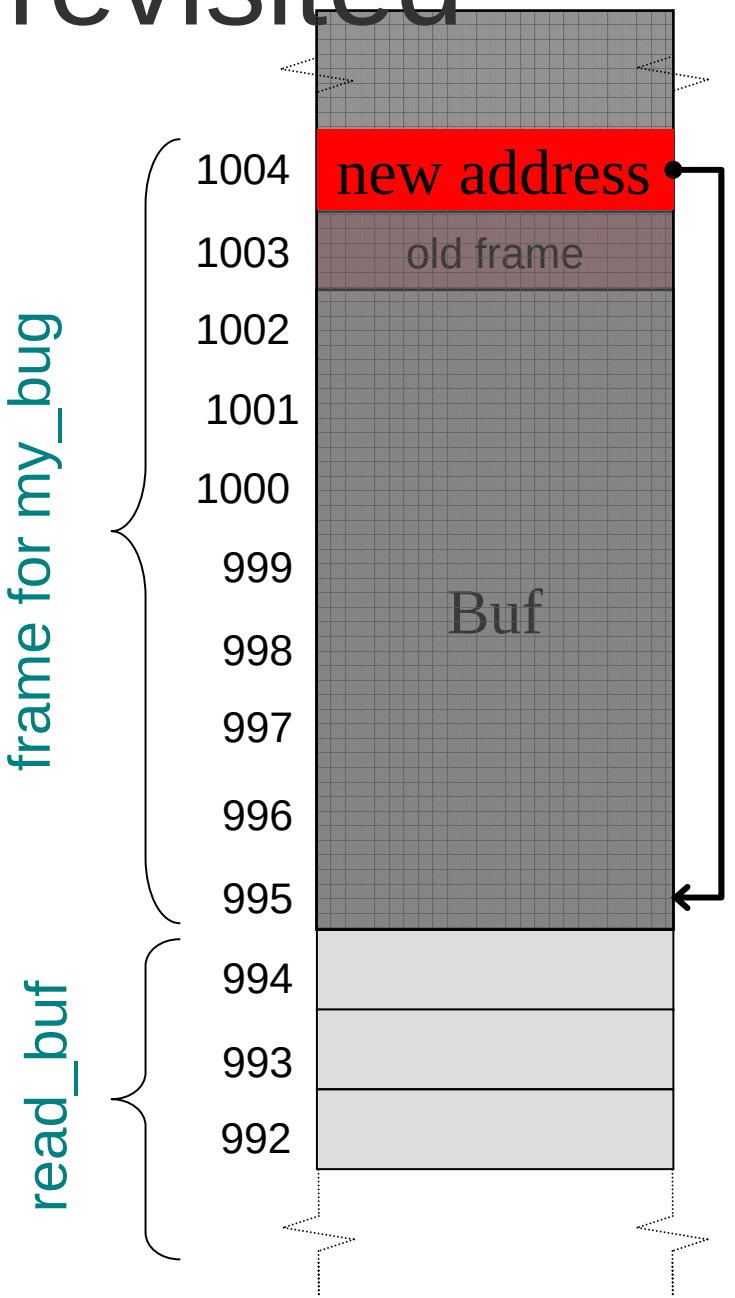
```
read_buf (buf){
   file = open (“my_file”);
   read (file, buf, 64);
   close (file);
   return;
}

my_bug (){
   char buf [10];
   read_buf (buf);
   print (buf);
   return;
}
```

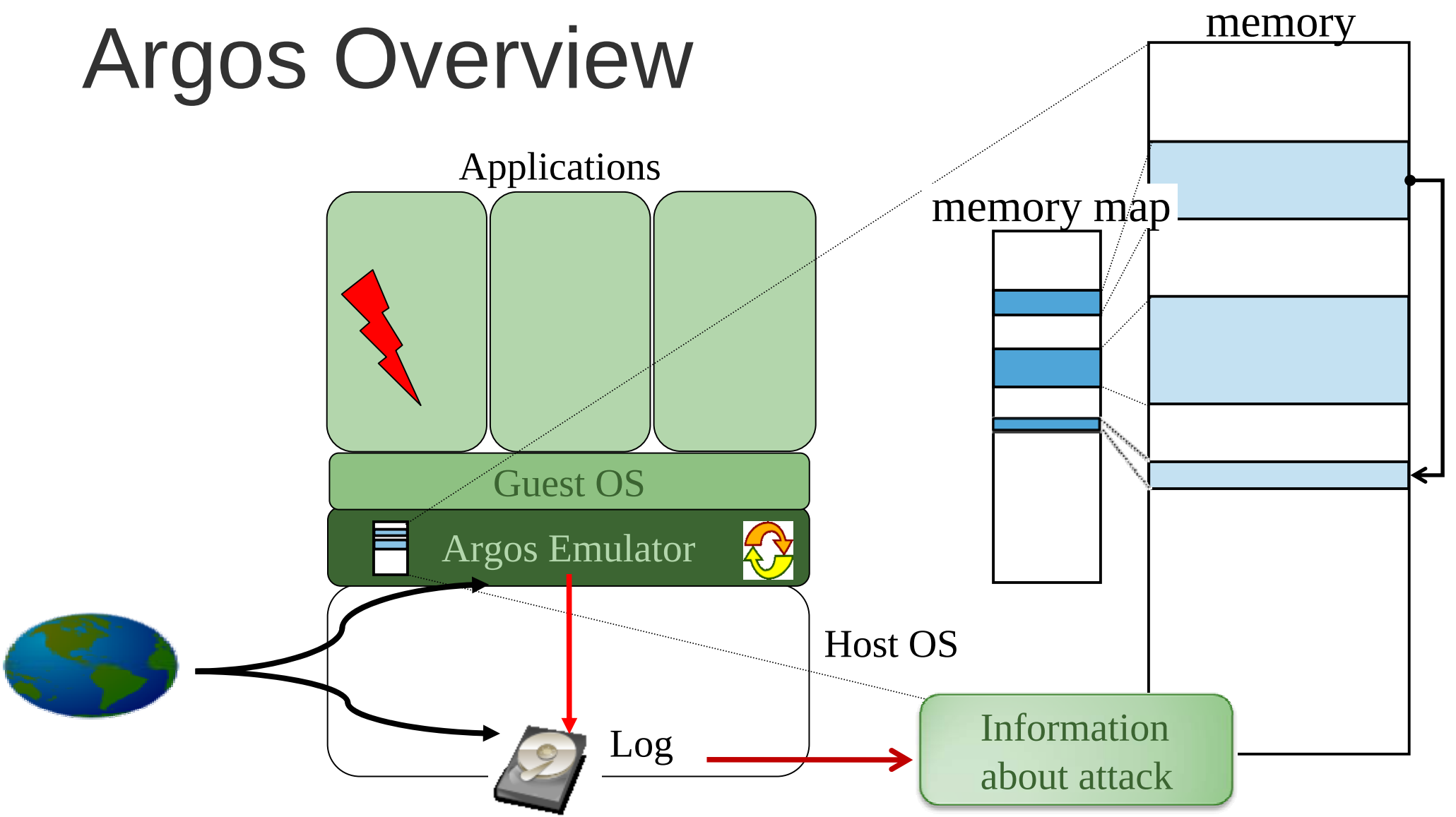


Stack smashing attack revisited

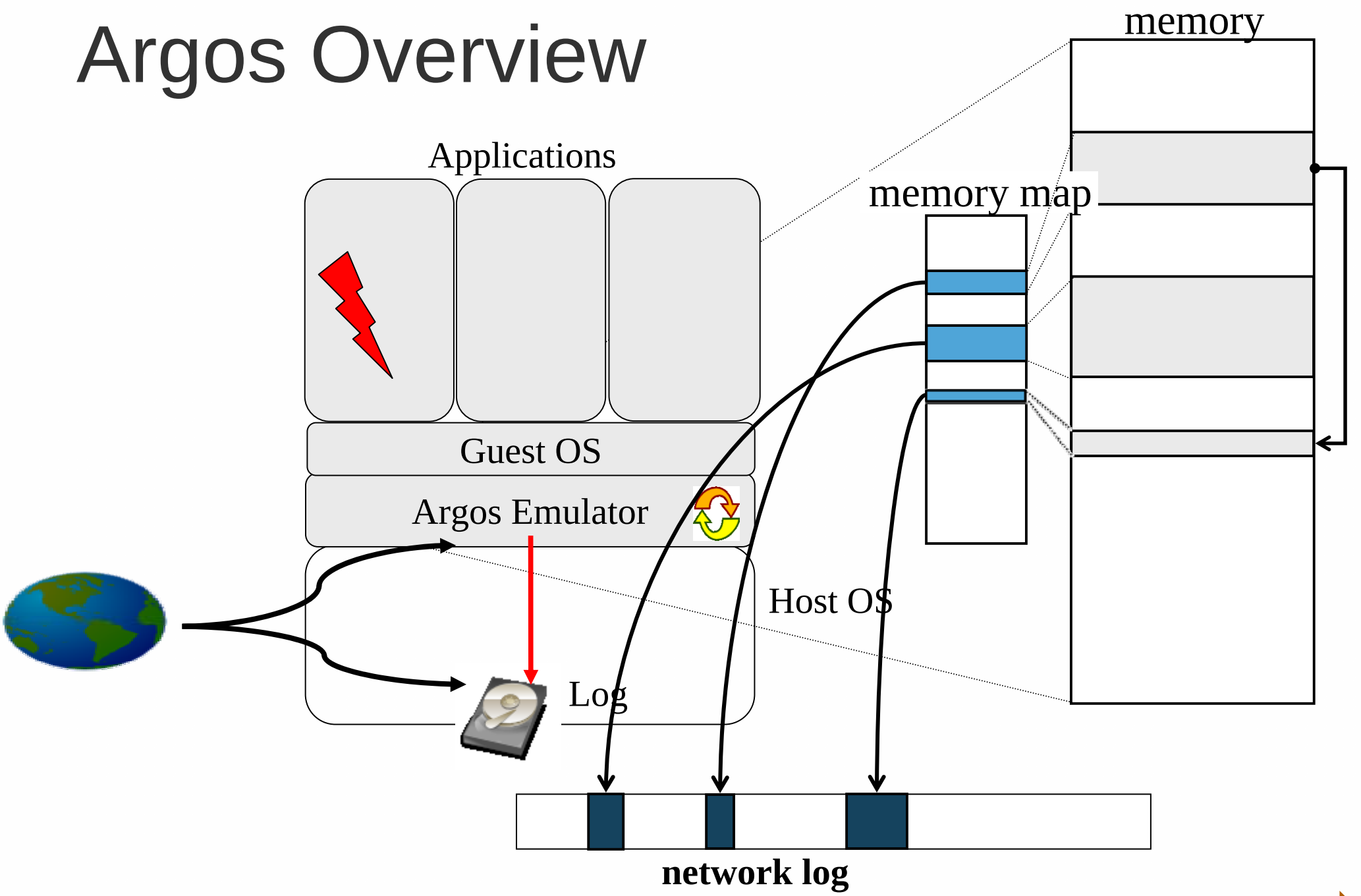
- all we need to do...
  - is load our program in `Buf`
  - overwrite the return address at `1004` with the begin address of `Buf (995)`
  - and away we go….



Argos Overview



Argos Overview



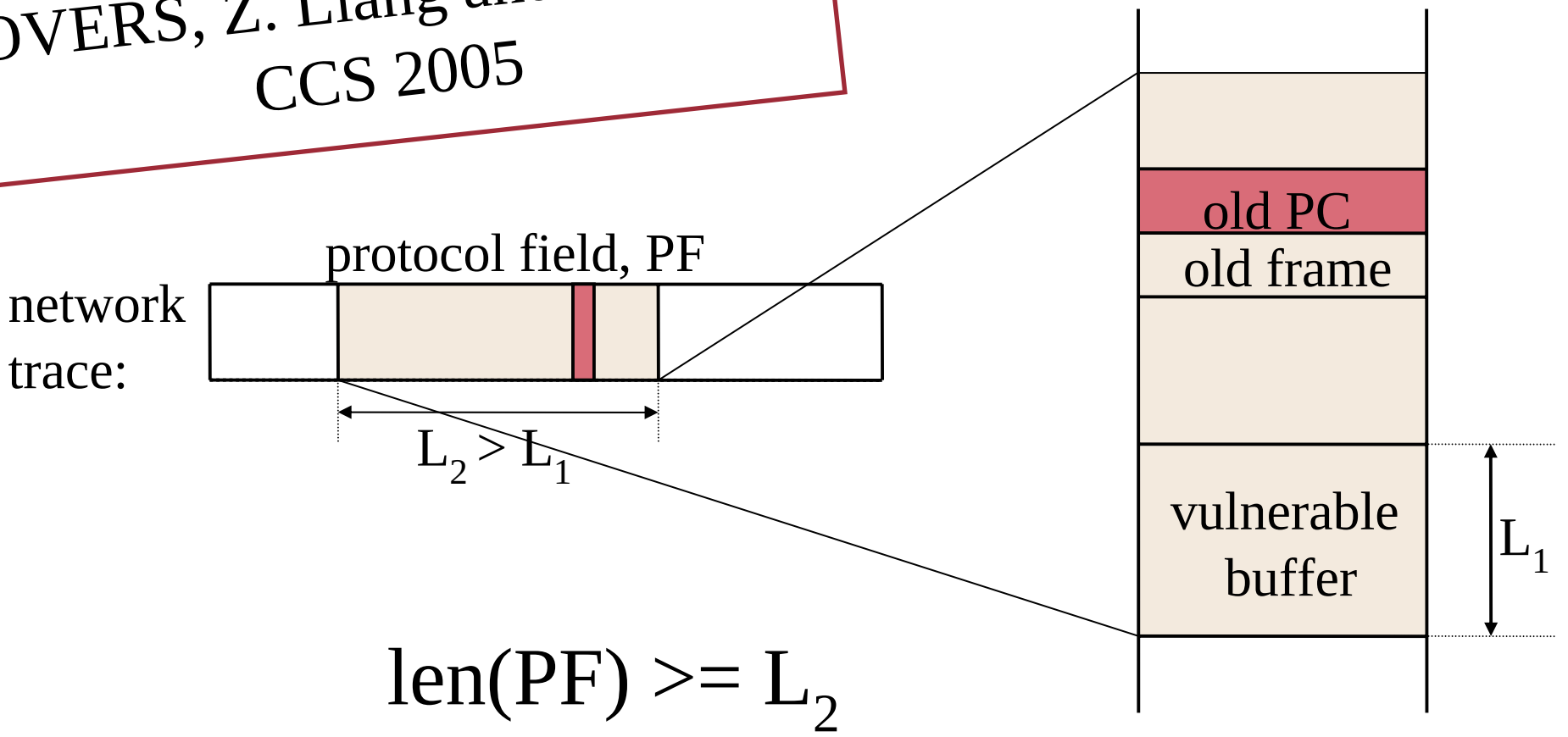
It works!

- Apache chunked encoding overflow
- IIS ISAPI .printer host header overflow
- WebDav ntdll.dll overflow
- FrontPage Server Extensions Debug Overflow
- War-FTP overflow
- ASN.1 Library Bitstring Heap Overflow
- Windows Message Queueing Remote Overflow
- RPC DCOM Interface overflow
- LSASS Overflow
- Windows PnP Service Remote Overflow
- nbSMTP remote format string exploit
- NetApi exploit
- WMF exploit

Signatures

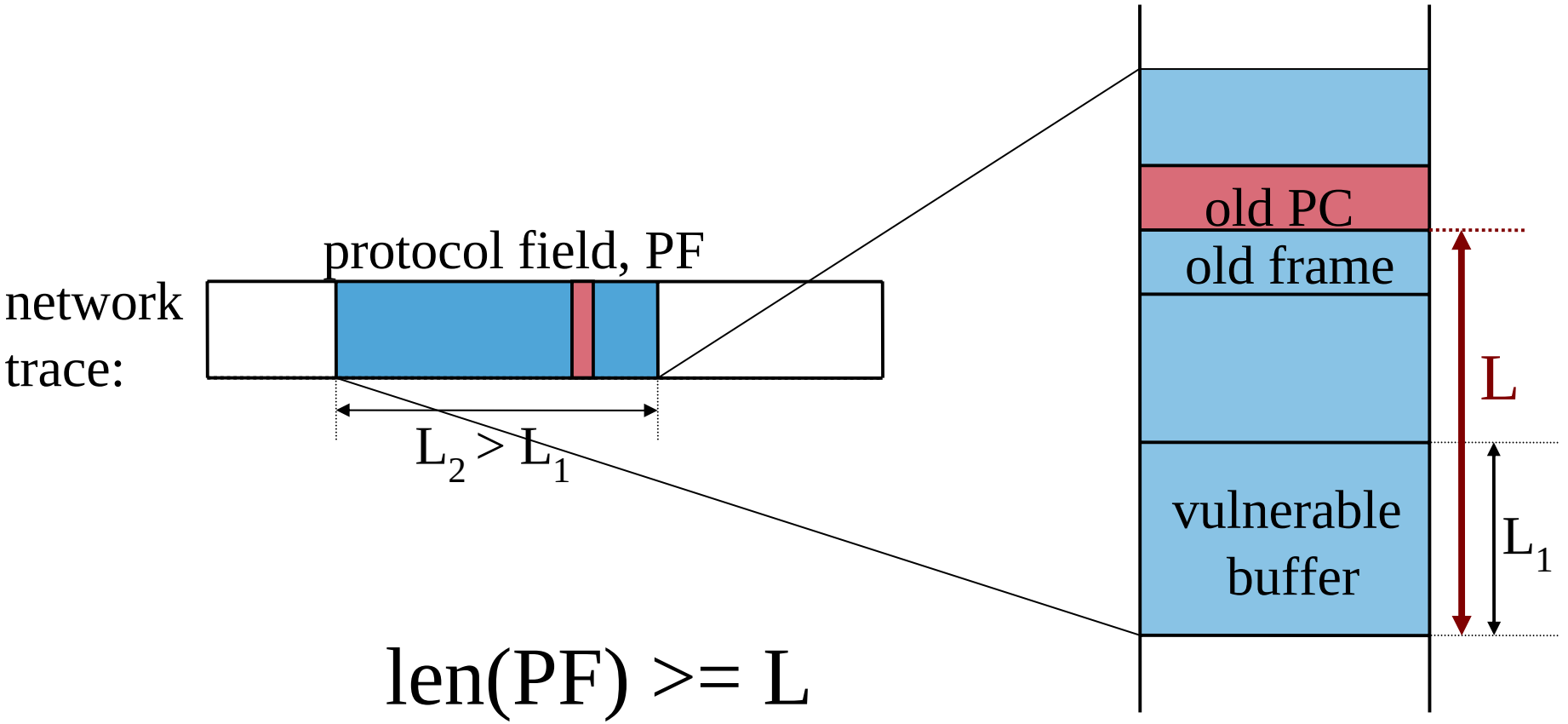
Vulnerability signatures

COVERS, Z. Liang and R. Sekar, CCS 2005



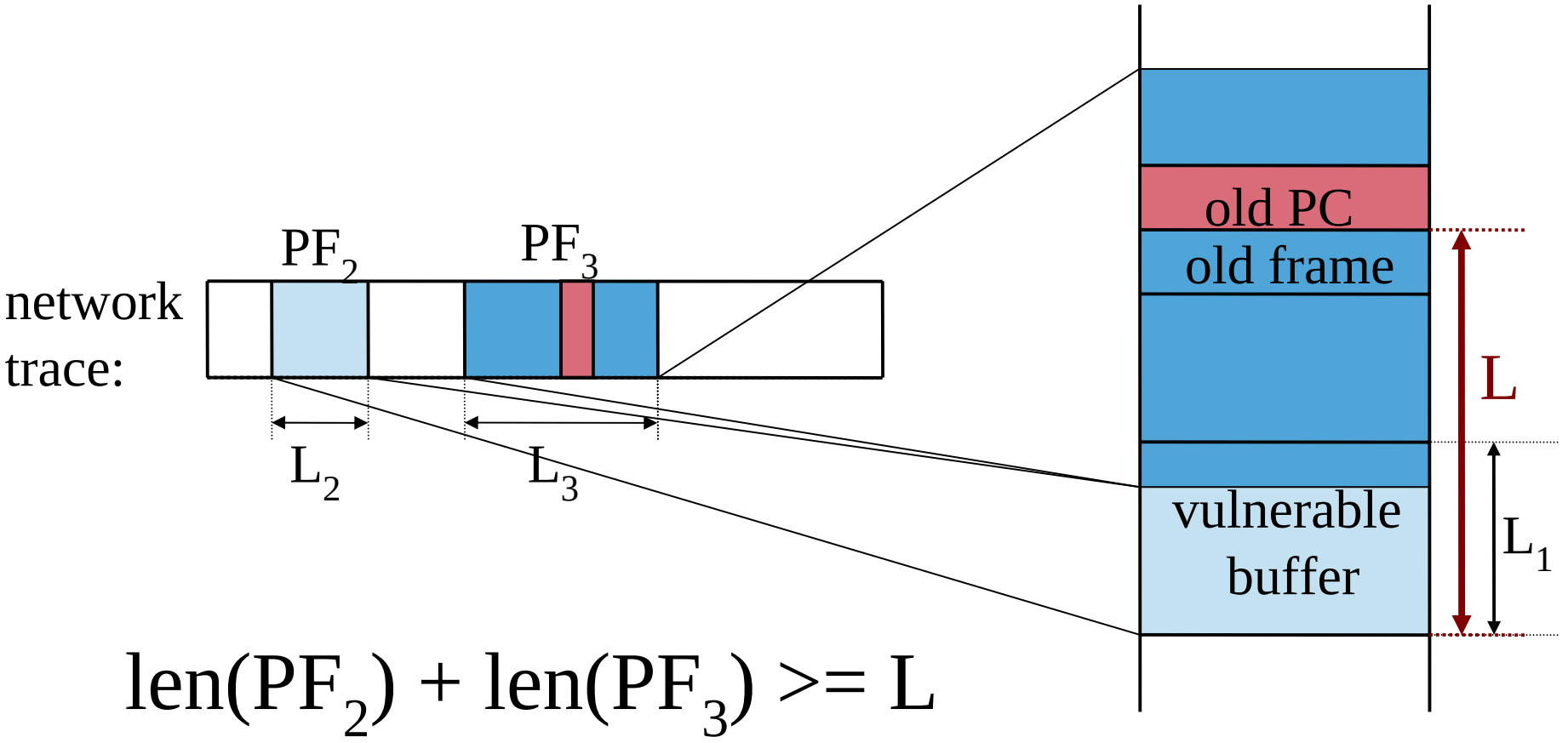
$$\text{len(PF)} >= L_2$$

Prospector: Vulnerability signatures



$$\text{len(PF)} >= L$$

Prospector: Vulnerability signatures

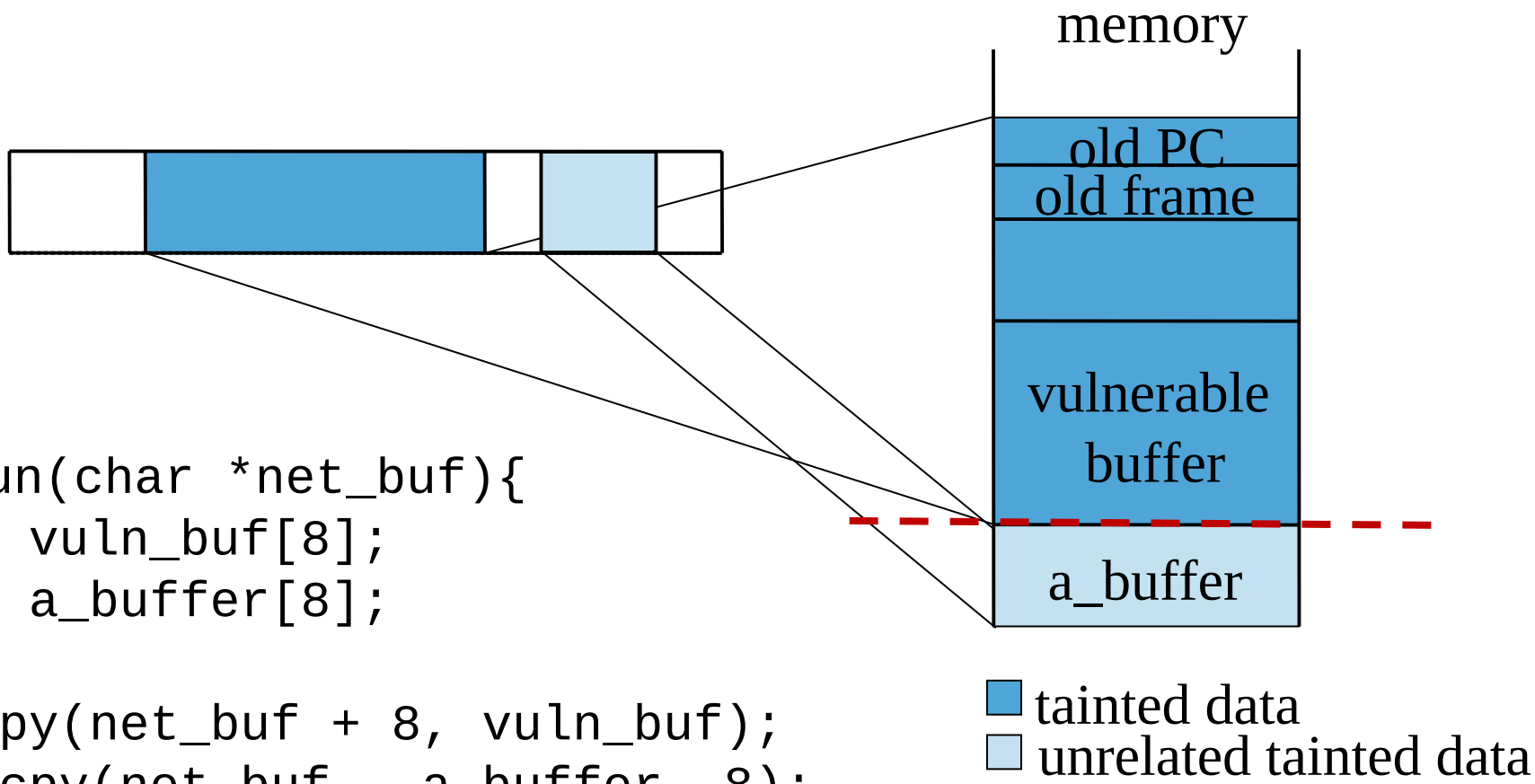


$$len(PF_2) + len(PF_3) >= L$$

approach: which bytes contributed to the attack?

surprisingly hard!

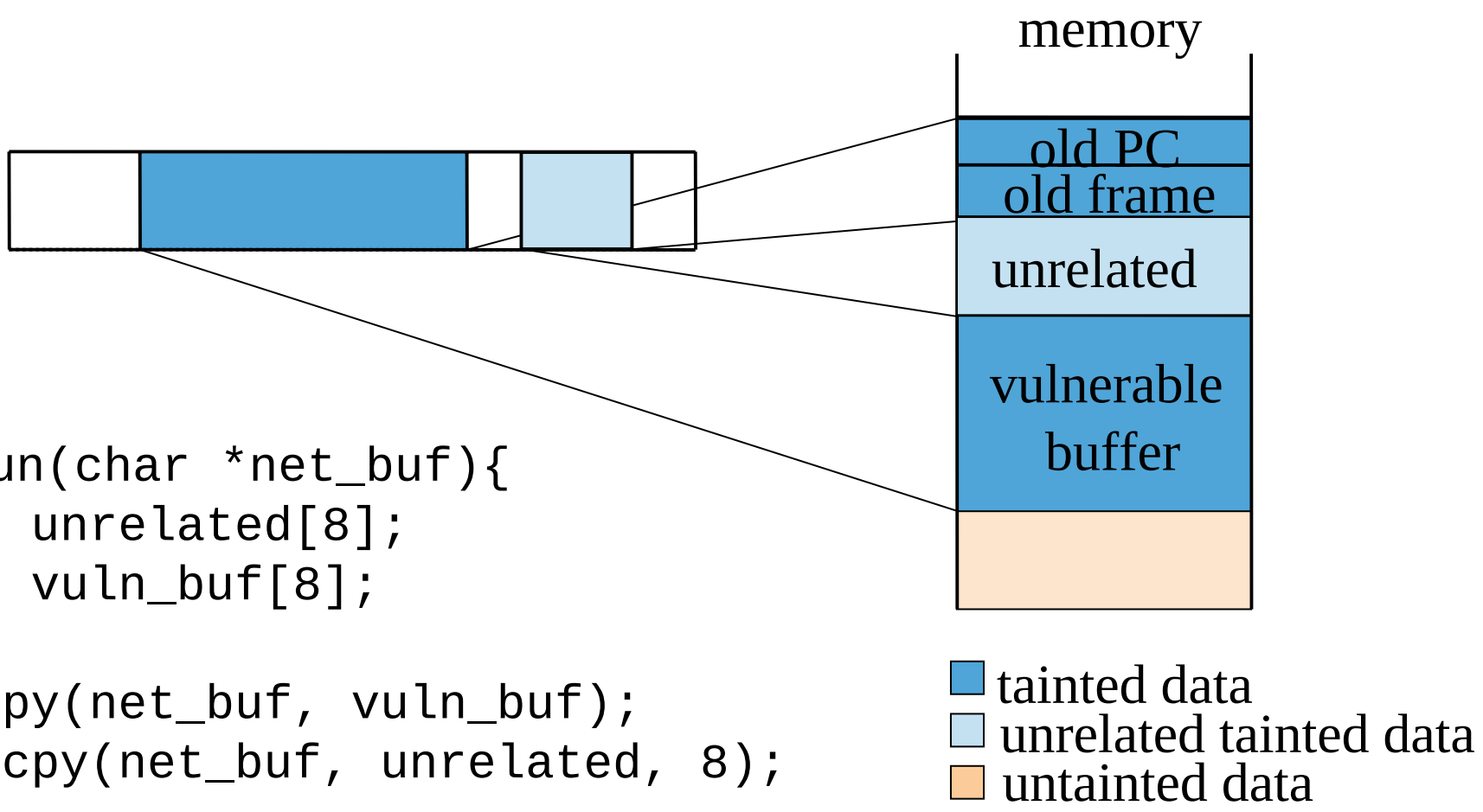
Nasty scenario: unrelated taints




```
void a_fun(char *net_buf){
     char vuln_buf[8];
     char a_buffer[8];

     strcpy(net_buf + 8, vuln_buf);
     strncpy(net_buf,  a_buffer, 8);
}
```

Nasty scenario: gap in the tainted region




```
void a_fun(char *net_buf){
     char unrelated[8];
     char vuln_buf[8];

     strcpy(net_buf, vuln_buf);
     strncpy(net_buf, unrelated, 8);
}
```

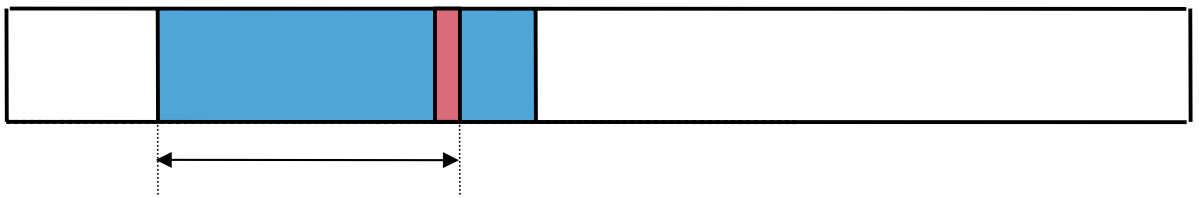
Prospector: Vulnerability s

Published on ACSAC 2007

- Built on top of Argos,
  - added extra instrumentation to the processor emulator tailored to support attack analysis
- Capable of pinpointing bytes that contributed to a buffer overflow attack,
- Information obtained can be used to generate a vulnerability signature that caters to polymorphic attacks

Signatures so far

- Snort-like
  - e.g., `TCP`, `dport=`80, content = "`0EA765F3.*DEADBEEF`"
- generated by Prospector
  - e.g., `HTTP/TCP`, Length(`URL`) < 256



at any rate: must look inside the traffic

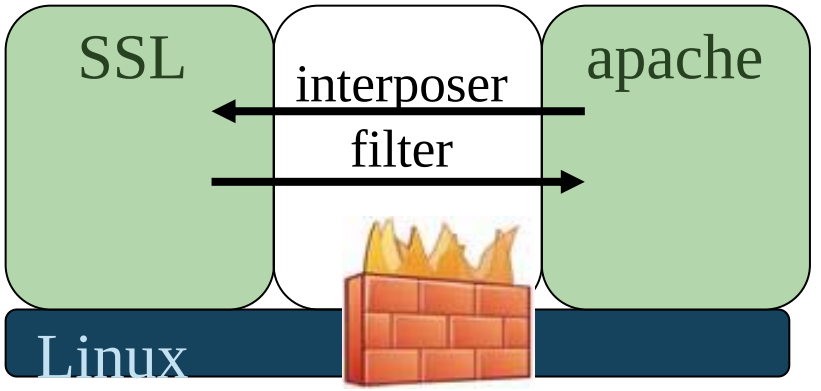
Encryption

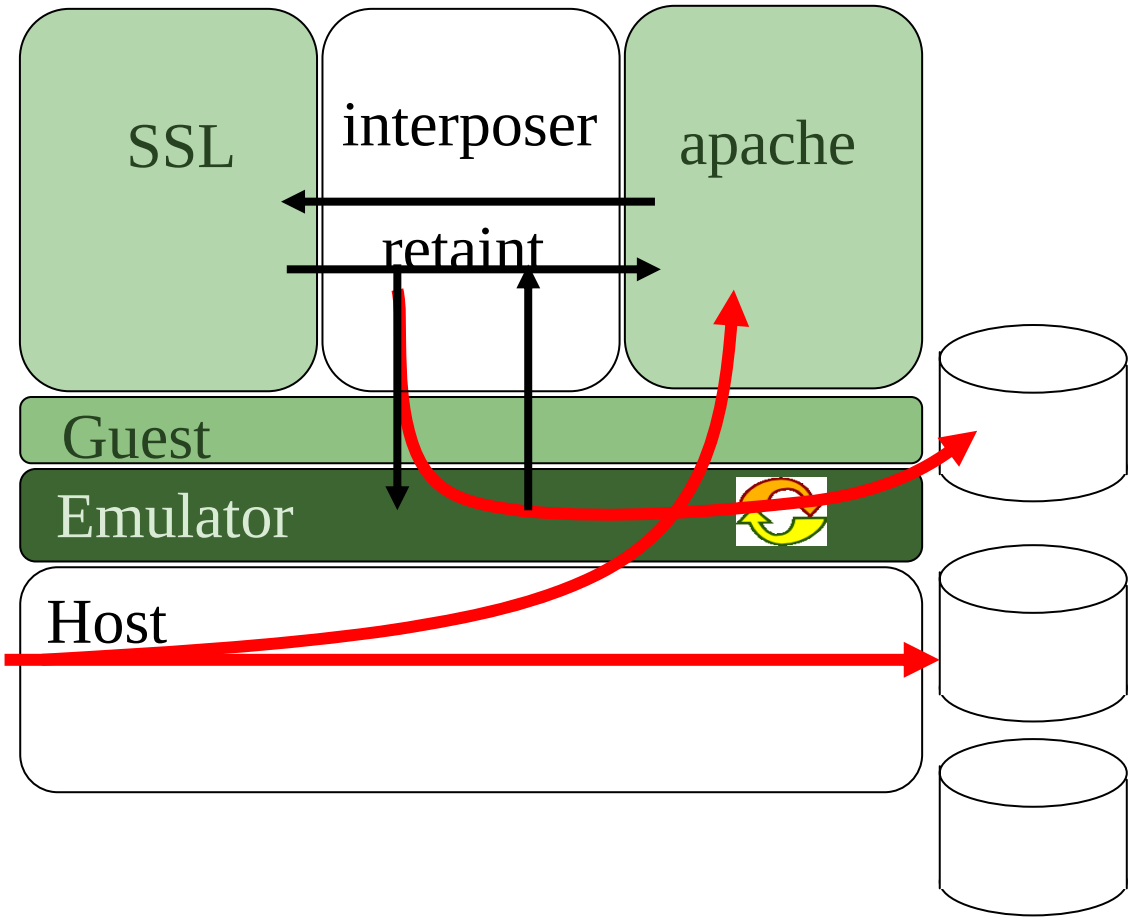
- useful for
  - protecting data,
  - privacy
- “on the wire all bytes are meaningless”
- Considered harmful to
  - attacks detection
  - fingerprinting
- techniques that rely on recognising properties in a network stream are doomed!
  - Snort, Bro, Covers, etc.

Encryption

- we want to…
  - look inside the cryptographically sealed channels
  - apply signatures there

Hassle





decrypted data

network trace

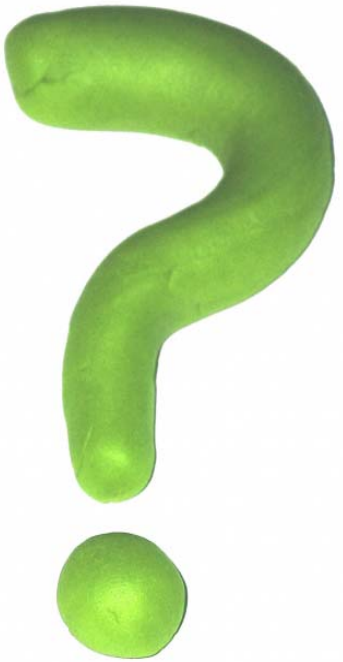
dump of tainted memory

signature generator

Conclusions

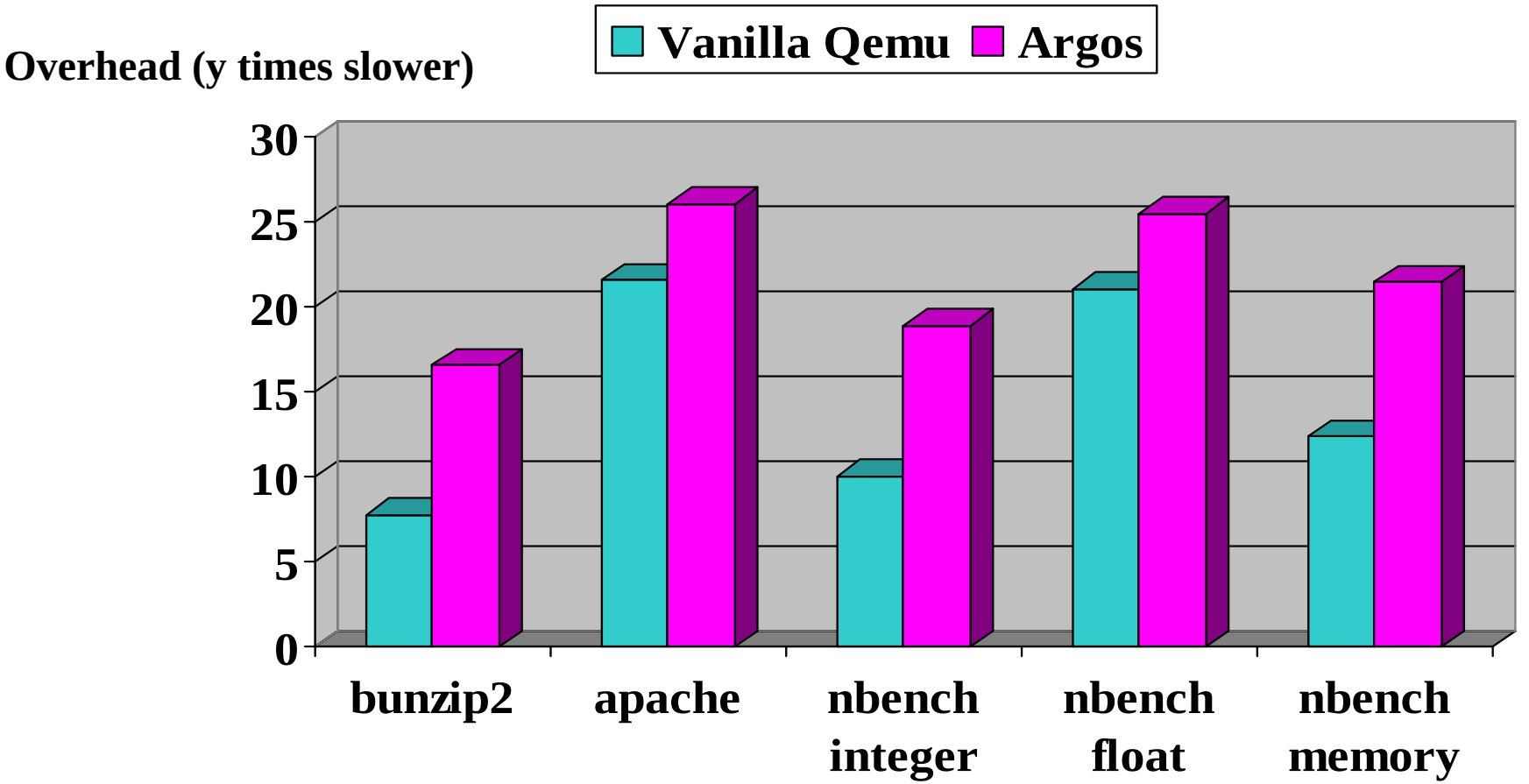
- Argos (EUROSYS 2006)
  - a processor emulator employing dynamic taint analysis to detect when application's control flow is altered by data coming from the network
- Prospector (ACSAC 2007)
  - extends Argos with some instrumentation tailored o support buffer overflow attack analysis
- Hassle (EC2ND 2007)
  - encrypted communication makes it impossible to check for known worms
  - interpose the channel between SSL library and an application , and apply/generate the signatures there

Refer to our papers
Argos - EUROSYS 2006
Prospector - ACSAC 2007
Hassle - EC2ND 2007



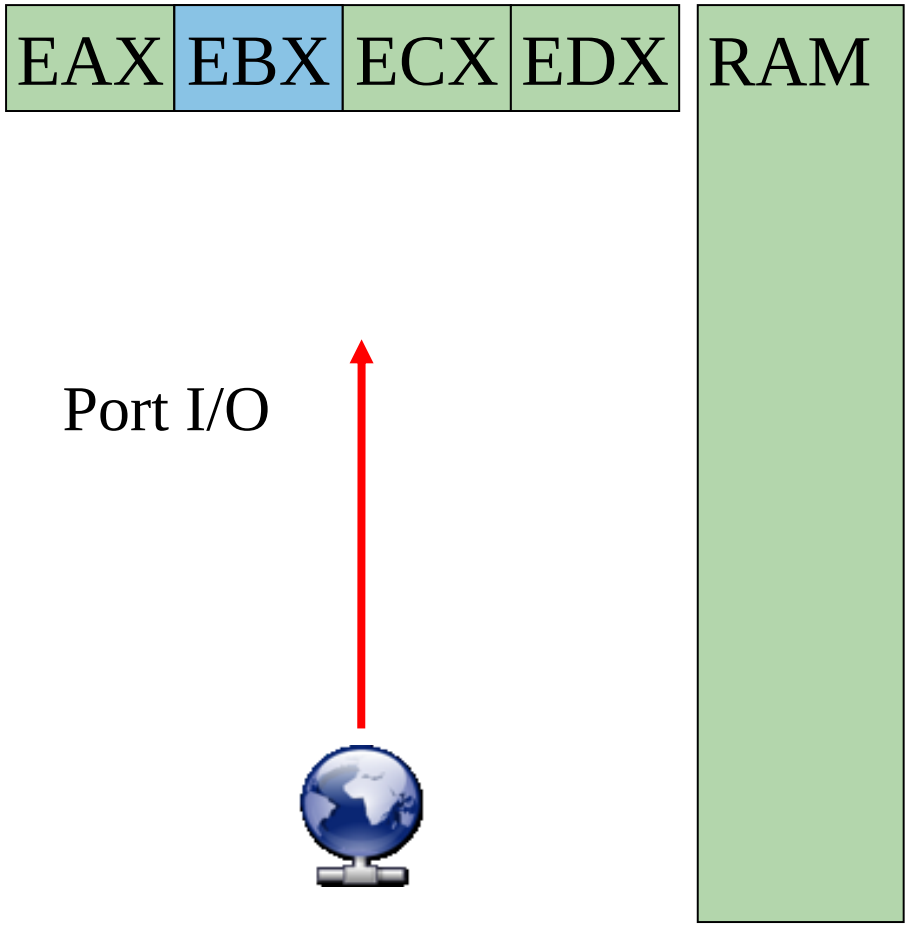
www.few.vu.nl/~asia
www.few.vu.nl/argos/
asia@few.vu.nl

Emulator Performance



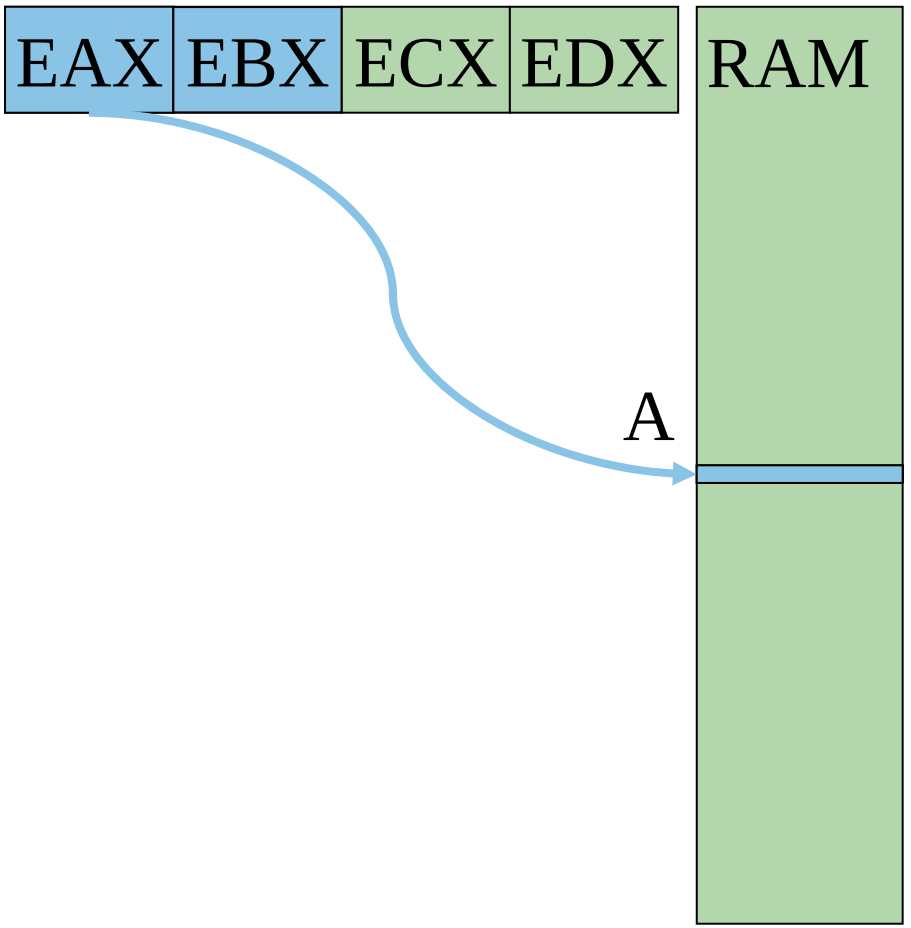
Network Data Tracking

- Tag network data as “tainted”



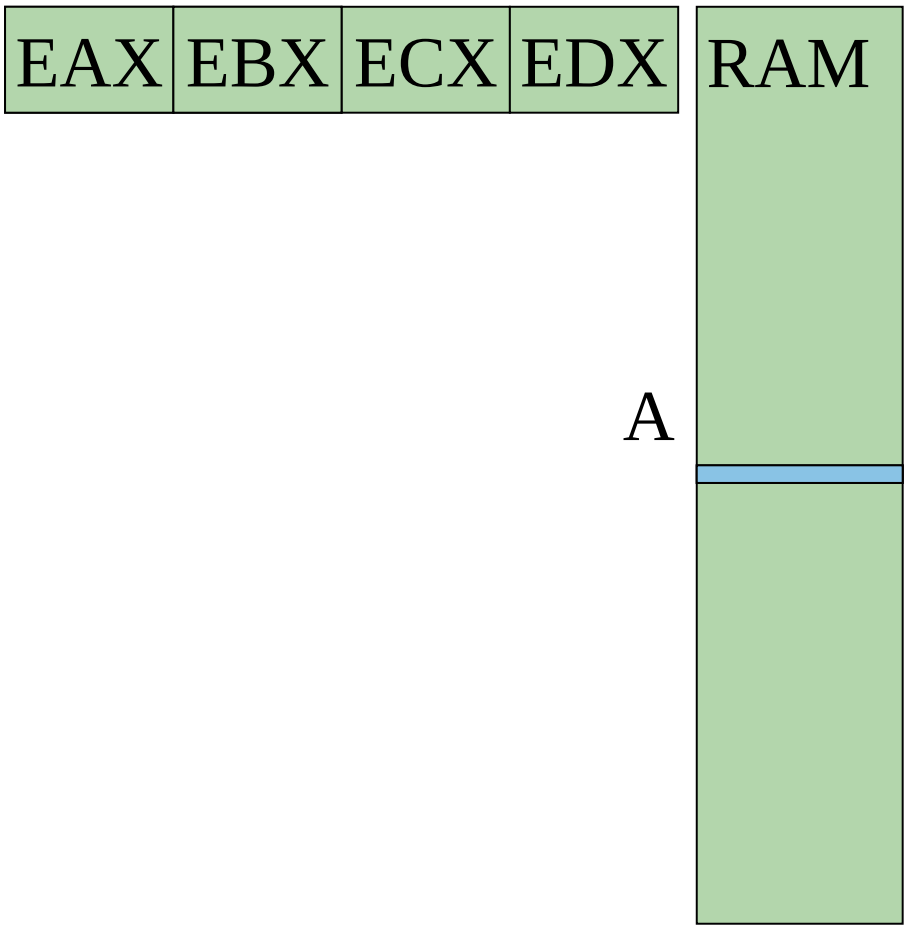
Network Data Tracking

- Tag network data as “tainted”
- Track “tainted” data propagation
  - Arithmetic, logical operations
  - Memory operations



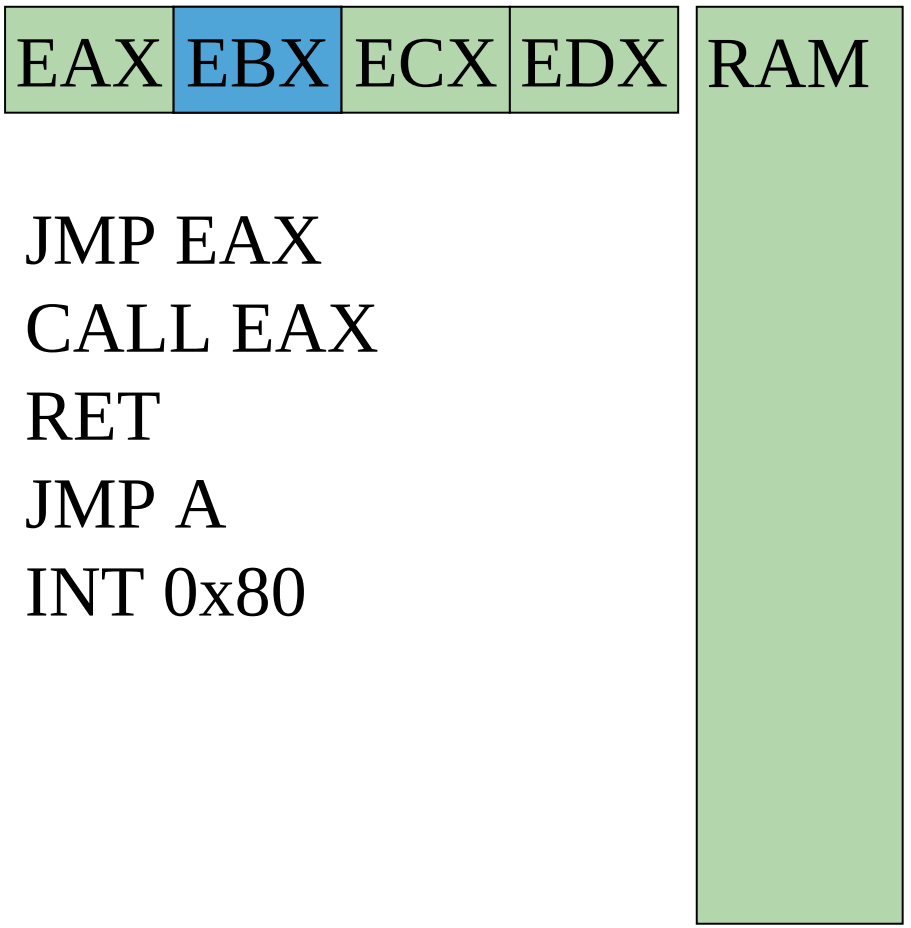
Network Data Tracking

- Tag network data as “tainted”
- Track “tainted” data propagation
  - Arithmetic, logical operations
  - Memory operations
- Sanitise data
  - Floating point, SSE

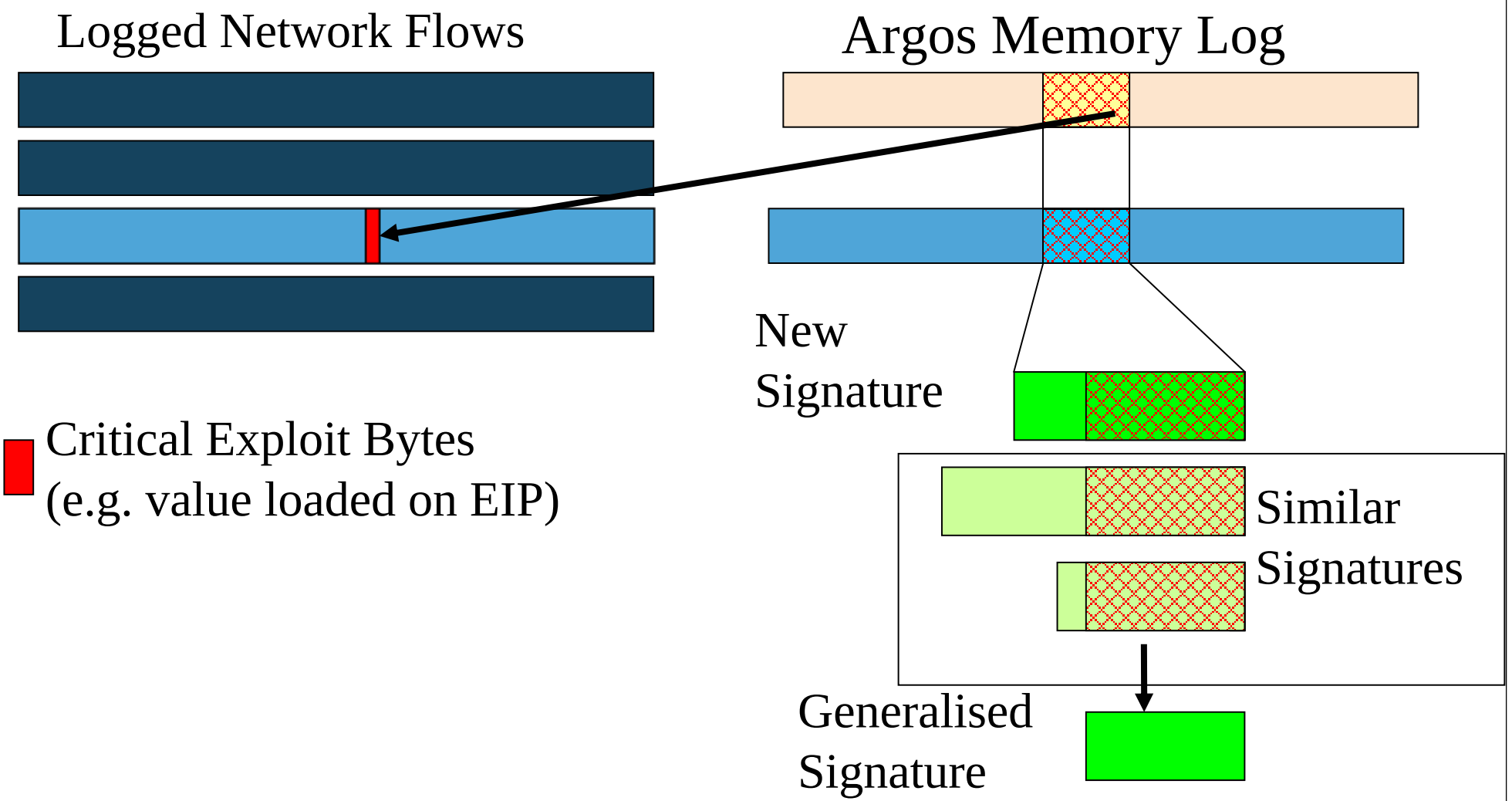


Identifying attacks

- Jumps
- Function calls
- Function returns
- System calls



Signature Generation I



Signature Generation I

- focus on exploits
  - less mutable than payload
  - used by different payloads
- generates Snort-like signatures
- works well with many existing threats
- true polymorphism is harder…