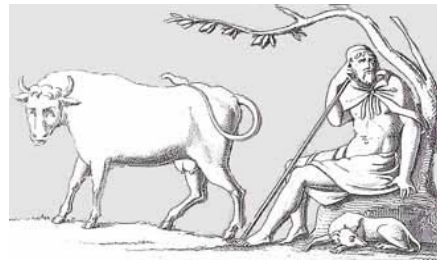
Argos Emulator



Georgios Portokalidis

Asia Slowinska

Herbert Bos

Vrije Universiteit Amsterdam

vrije Universiteit

Why?

- Too many vulnerabilities
- New worm attacks
- Human intervention too slow
- Current solutions are problematic
  - Time consuming
  - Inaccurate

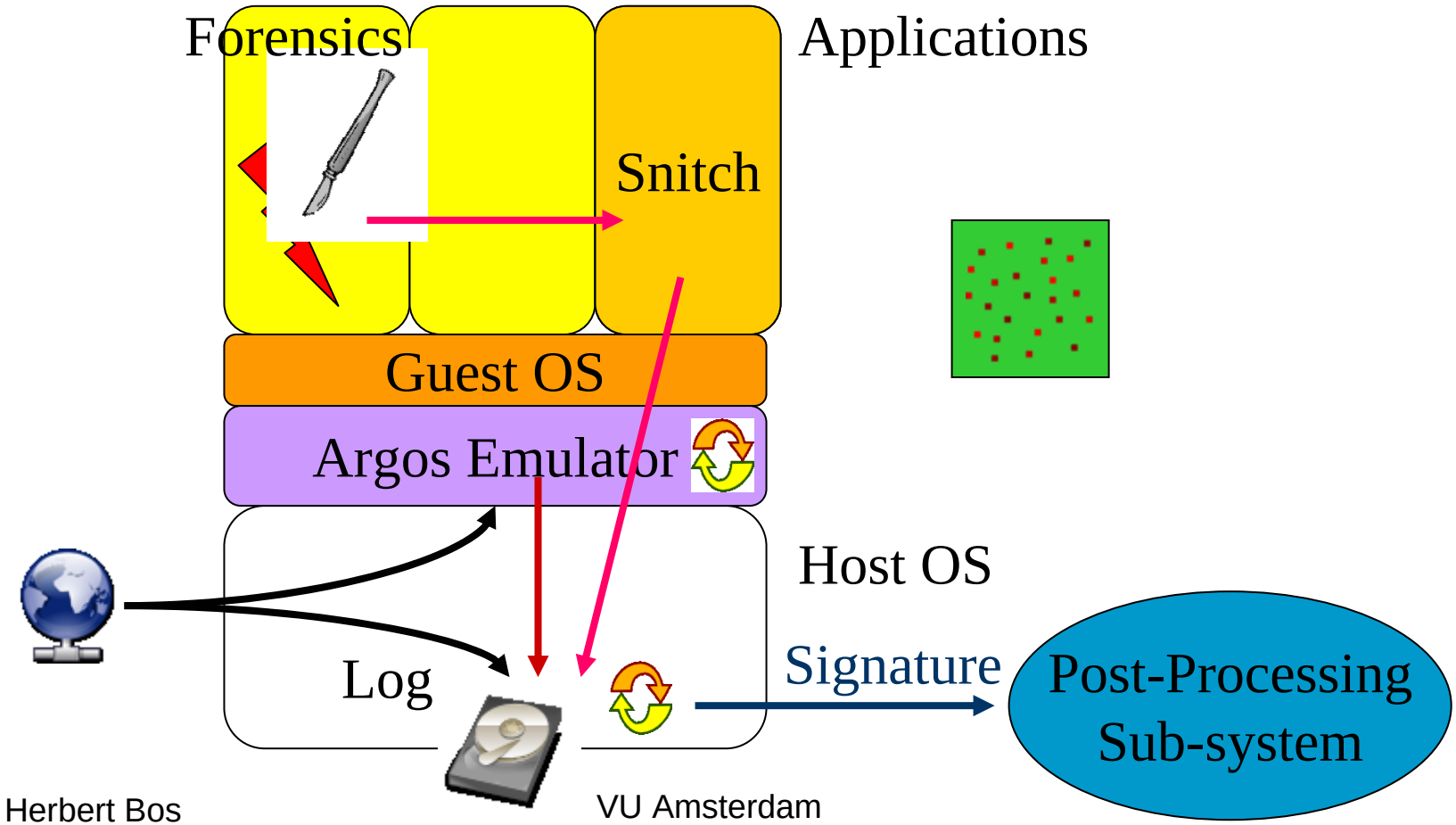
Goals

- Platform for next generation honeypots
- Protect entire OS
- Detect most common attack vectors
- Accuracy

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
WMF exploit

Argos Overview



Network Data Tracking

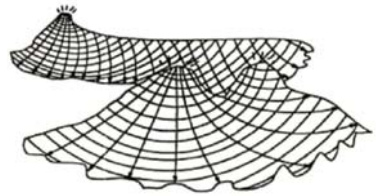
Register = network_read

Reg. A = Reg. A + Reg. B

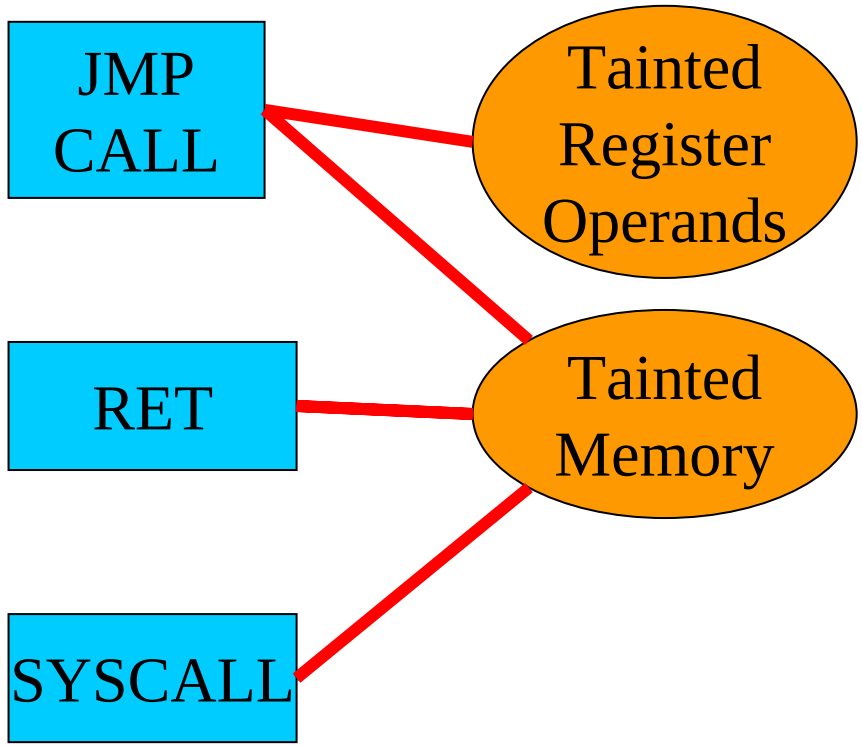
Memory(A) = Reg. A

Reg.B = Reg.A / 156.345

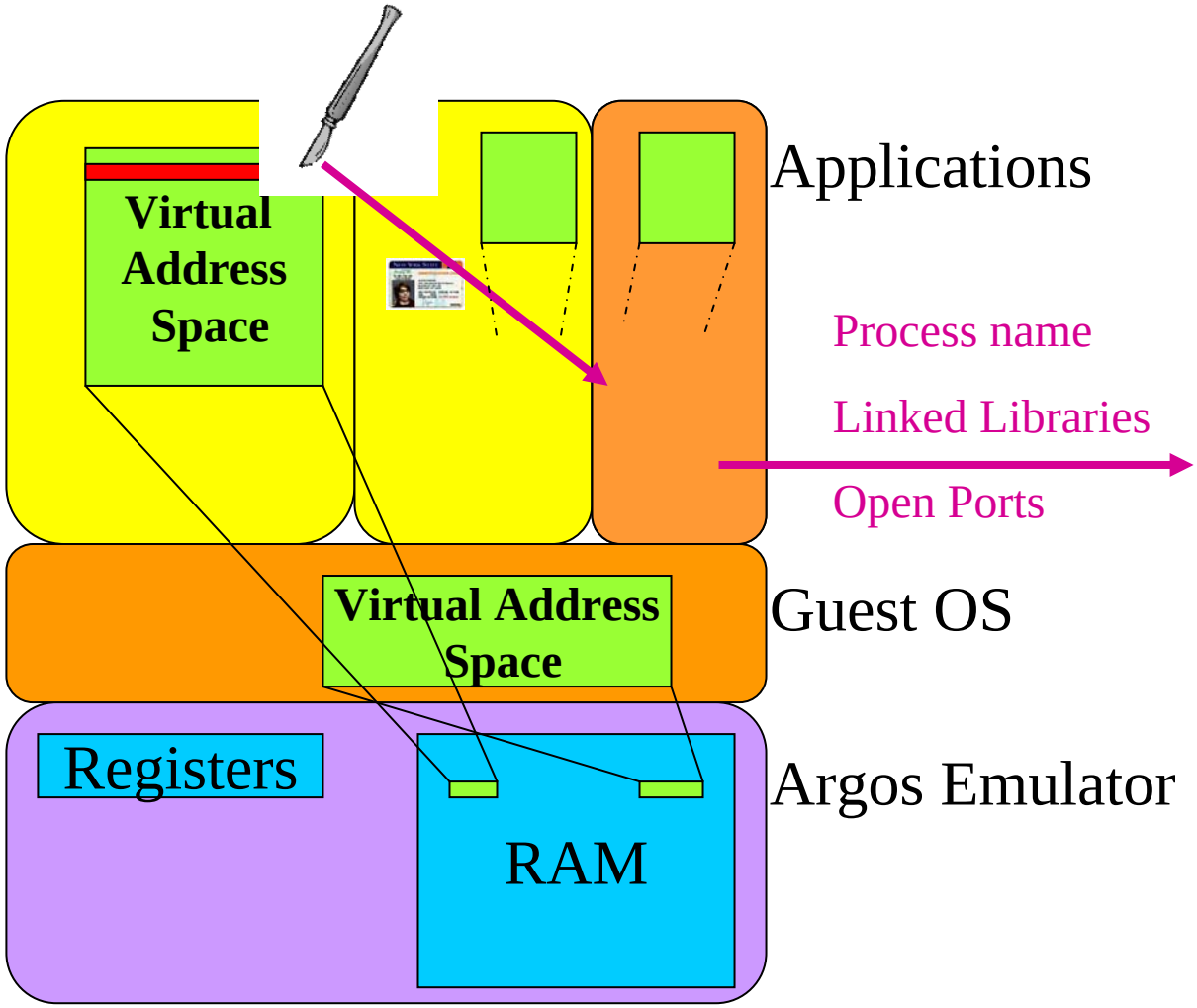
Capturing Attacks



- Diverting control flow
- Executing arbitrary instructions
- Overwriting system call arguments



Forensics

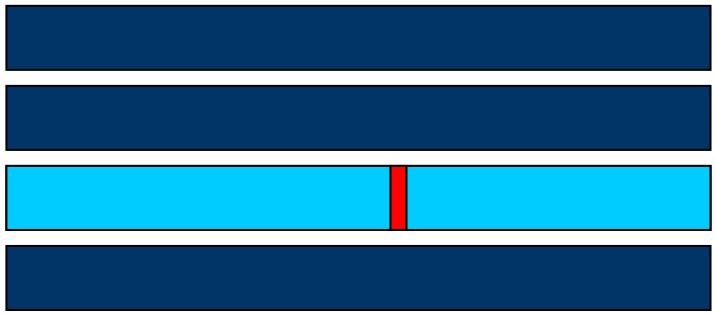

Virtual Address Space
Applications
Process name
Linked Libraries
Open Ports
Virtual Address Space
Guest OS
Registers
RAM
Argos Emulator

Signature Generation

Multiple Signature Generators

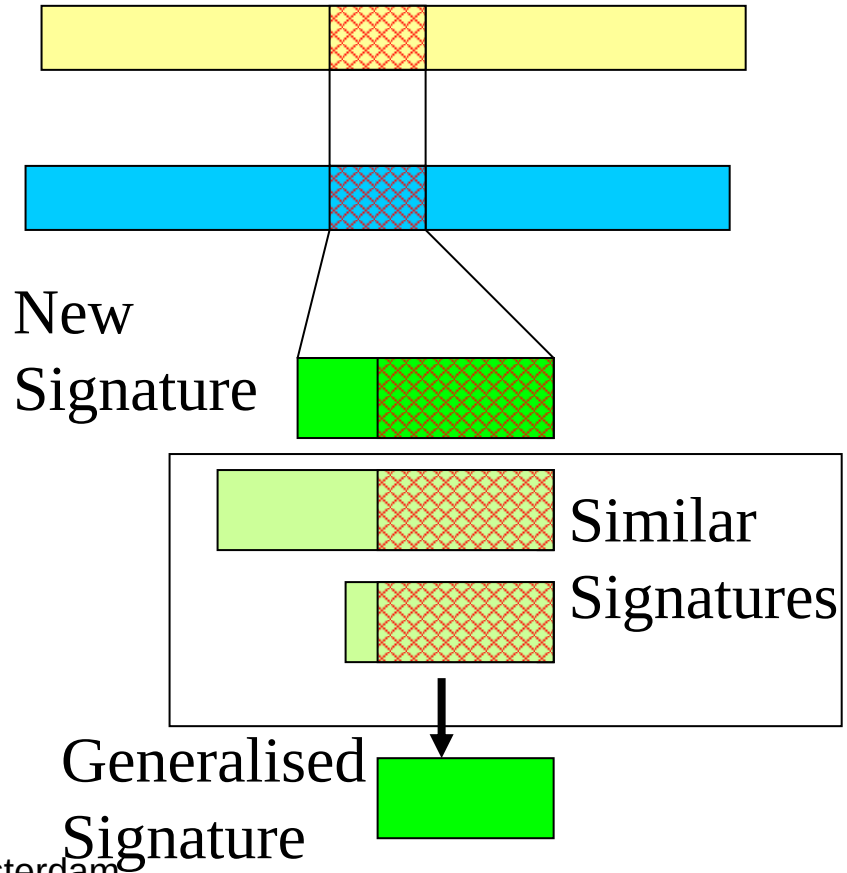
Signature Generation I

Logged Network Flows

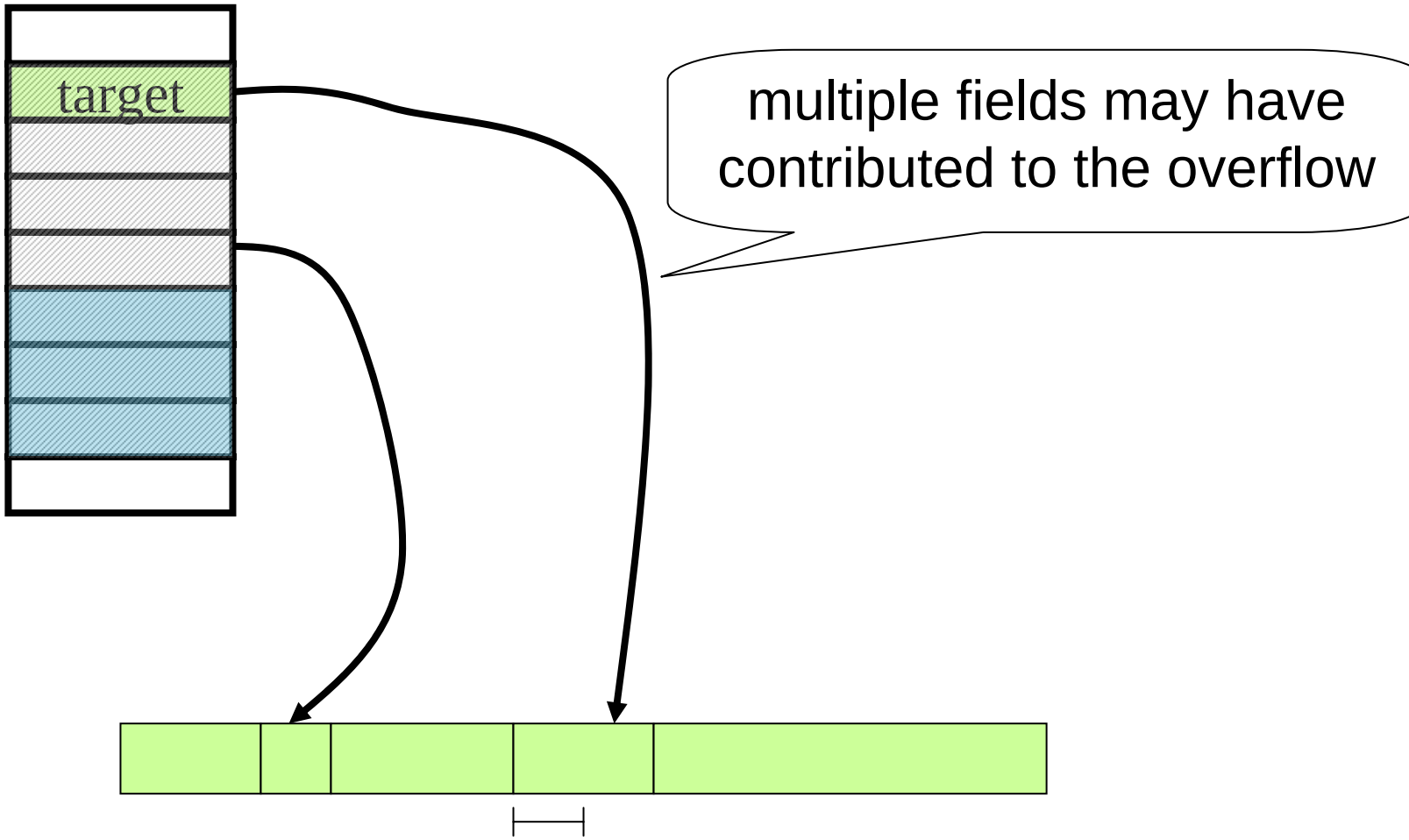


Critical Exploit Bytes
(e.g. value loaded on EIP)

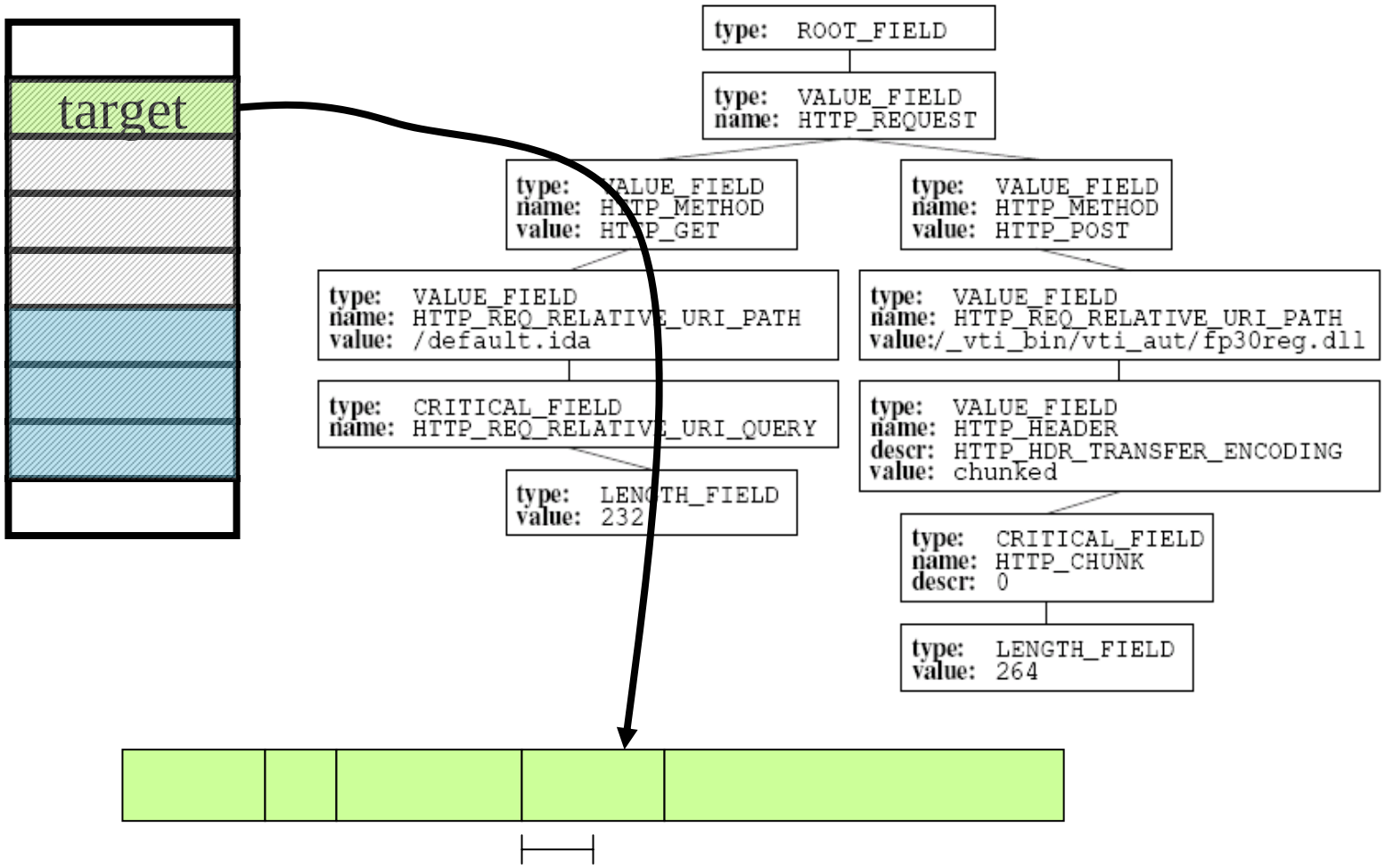
Argos Memory Log



Signature Generation II



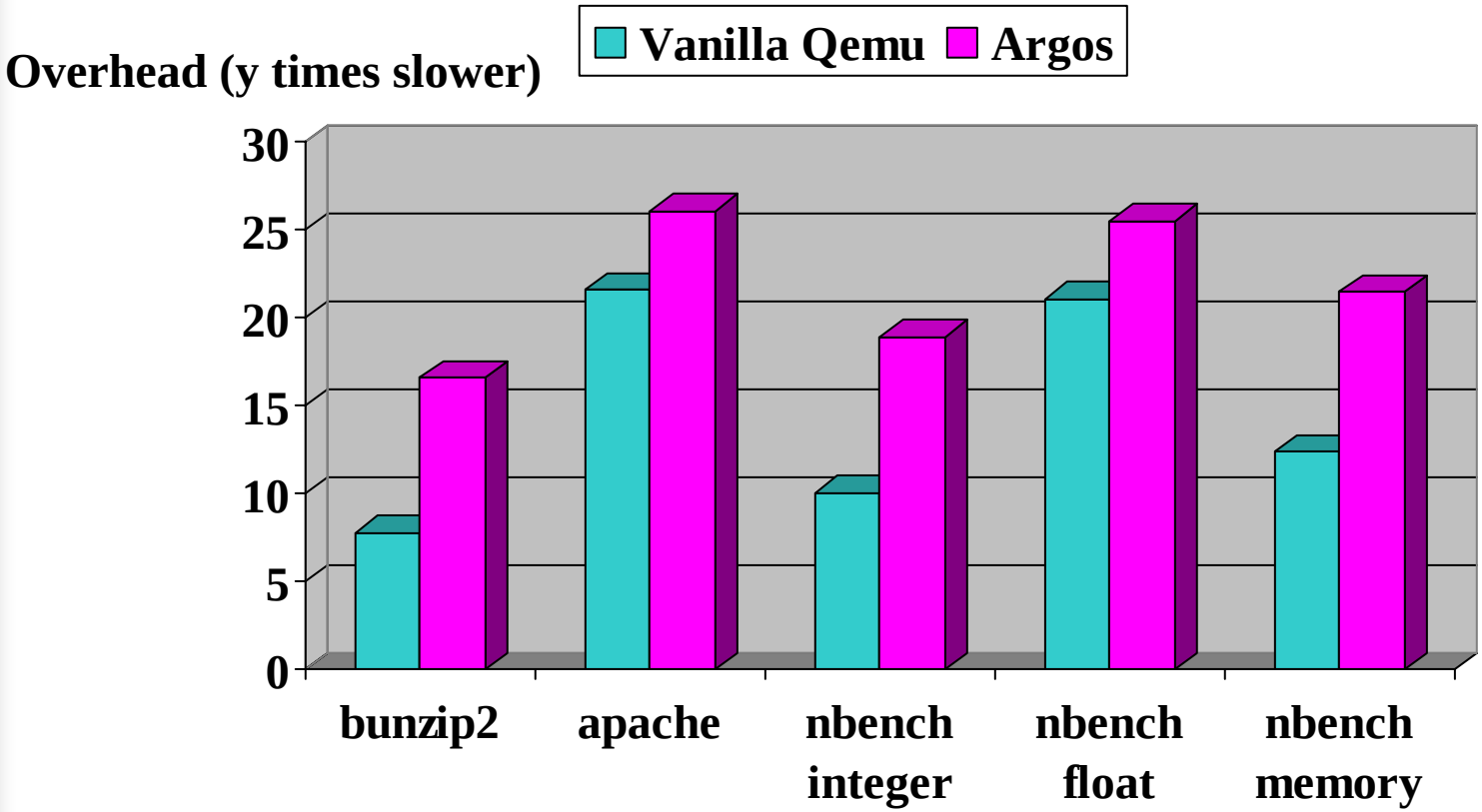
Signature Generation II



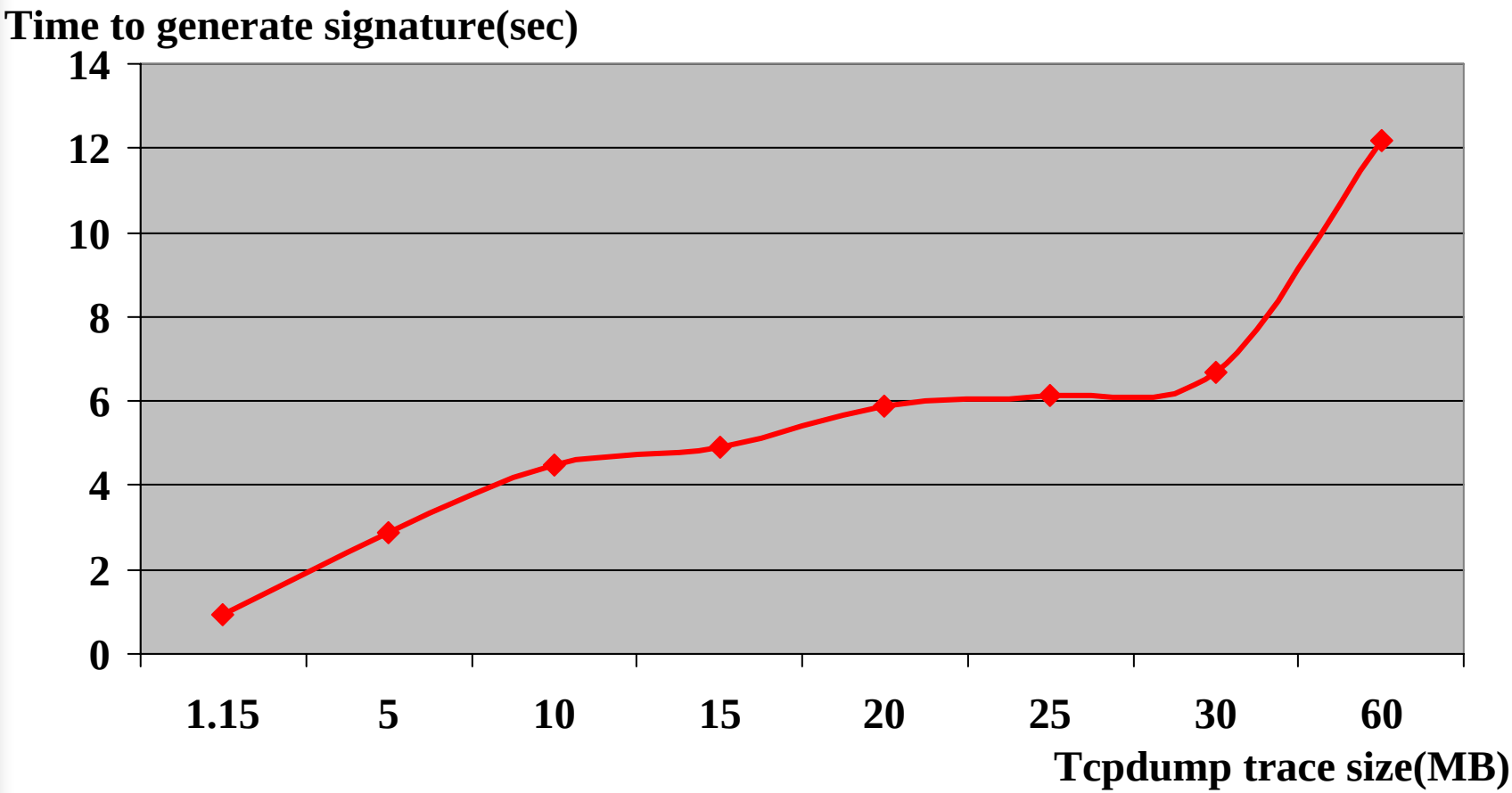
Signature Generator II

- handles polymorphic attacks
- easy to check
- great when considering false positives

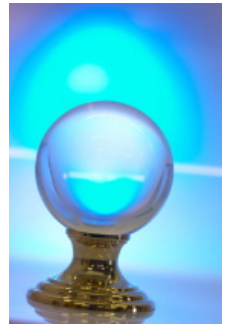
Emulator Performance



Signature Generation Performance



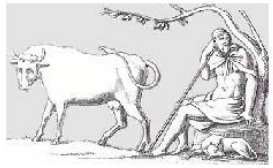
Future Work



- Replaying attacks
- Integration with nepenthes honeypot
- ✓ Increase data tracking precision
- ✓ Protocol aware signature generation
- Generate self certifying alerts

On The Web

http://www.few.vu.nl/argos



Argos: an Emulator

for Capturing

Zero-Day Attacks

vrije Universiteit

> Menu

- Home
- News
- Downloads
- Documentation
- Partners

> Propaganda

Petition for a Software Patent

Argos is a *full* and *secure* system emulator designed for use in Honeypots. It is based on QEMU, an open source processor emulator that uses dynamic translation to achieve a fairly good emulation speed.

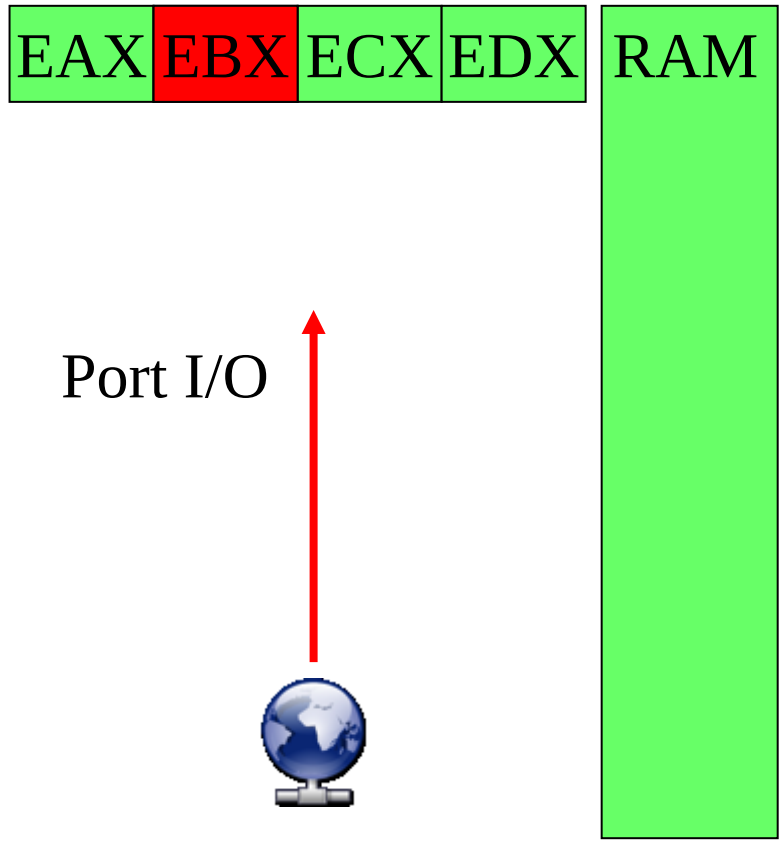
We have extended QEMU to enable it to detect remote attempts to compromise the emulated guest operating system. Using dynamic taint analysis Argos tracks network data throughout the processor's execution and detects any attempts to use them in a malicious way. When an attack is detected the memory footprint of the attack is logged and the emulators exits.

Argos is the first step to create a framework that will use *next generation honeypots* to automatically identify and produce remedies for zero-day worms, and other similar attacks. *Next generation honeypots* should not require that the honeypot's IP address remains un-advertised. On the contrary, it should attempt to publicise its services and even actively generate traffic. In former honeypots this was often impossible, because malevolent and benevolent traffic could not be distinguished. Since Argos is explicitly signaling each possibly successful exploit attempt, we are now able to differentiate malicious attacks and innocuous traffic.

Argos has moved to the local gForge. You can pickup the code and documentation there.

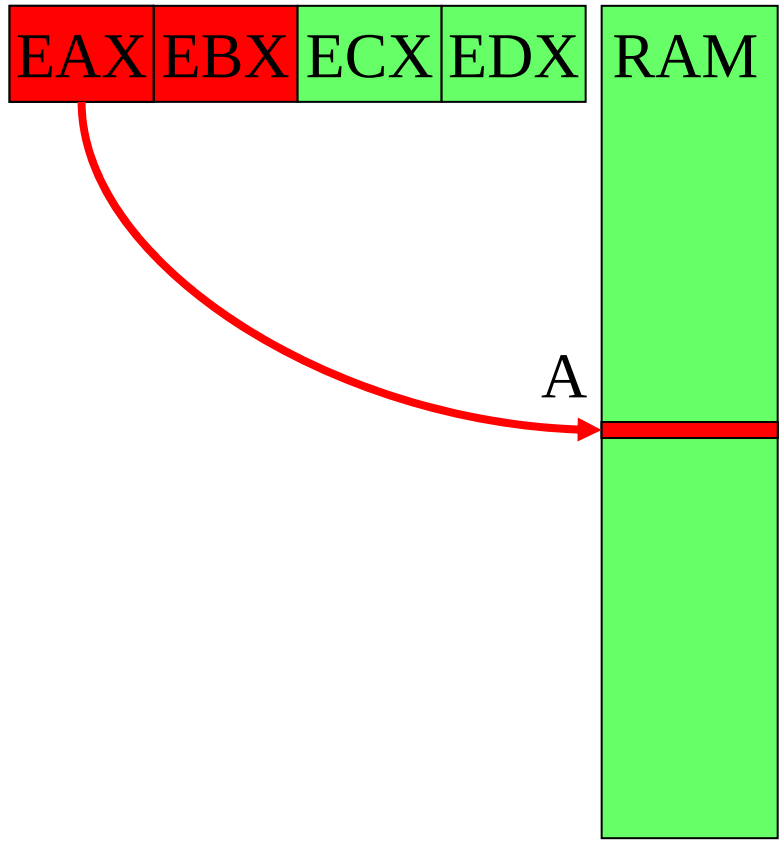
Network Data Tracking

- Tag network data as “tainted”



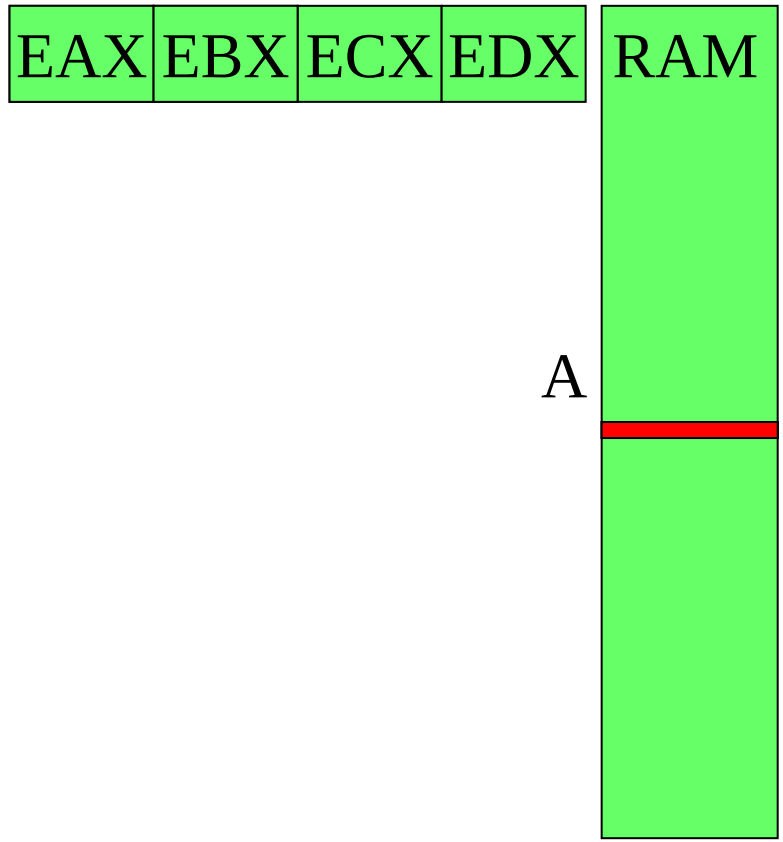
Network Data Tracking

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



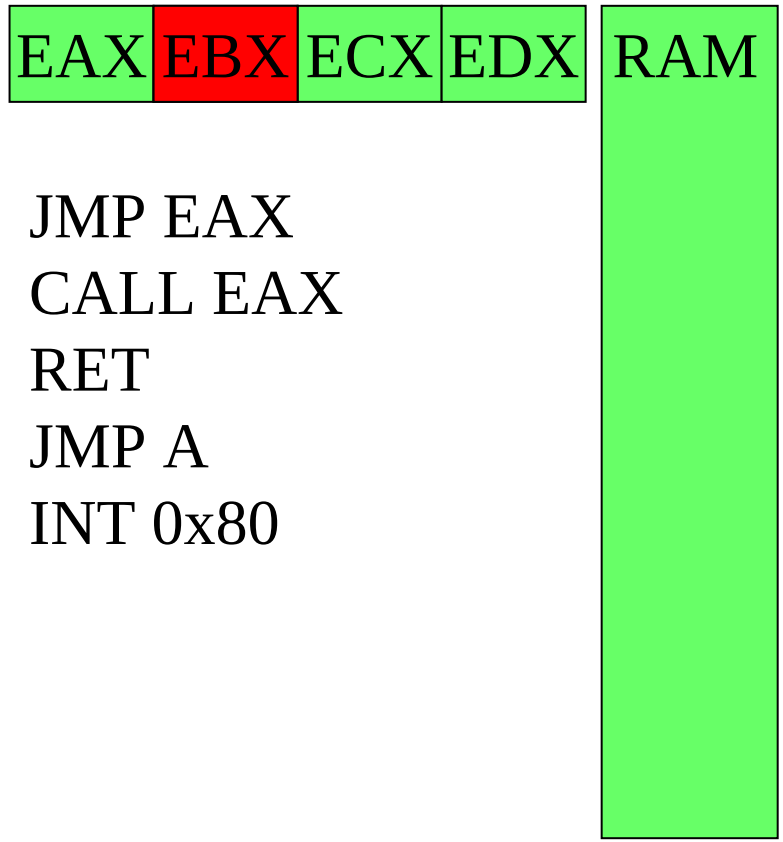
Network Data Tracking

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

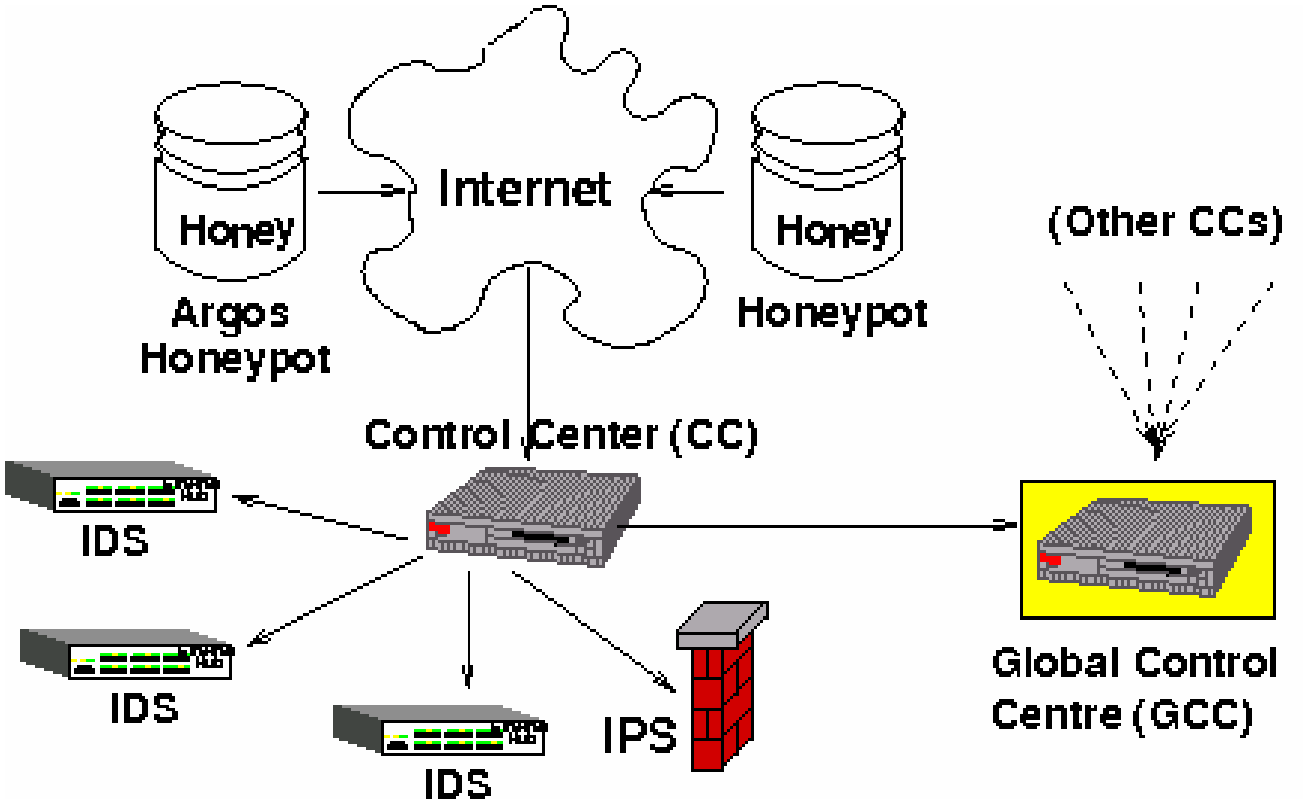


Identifying Attacks

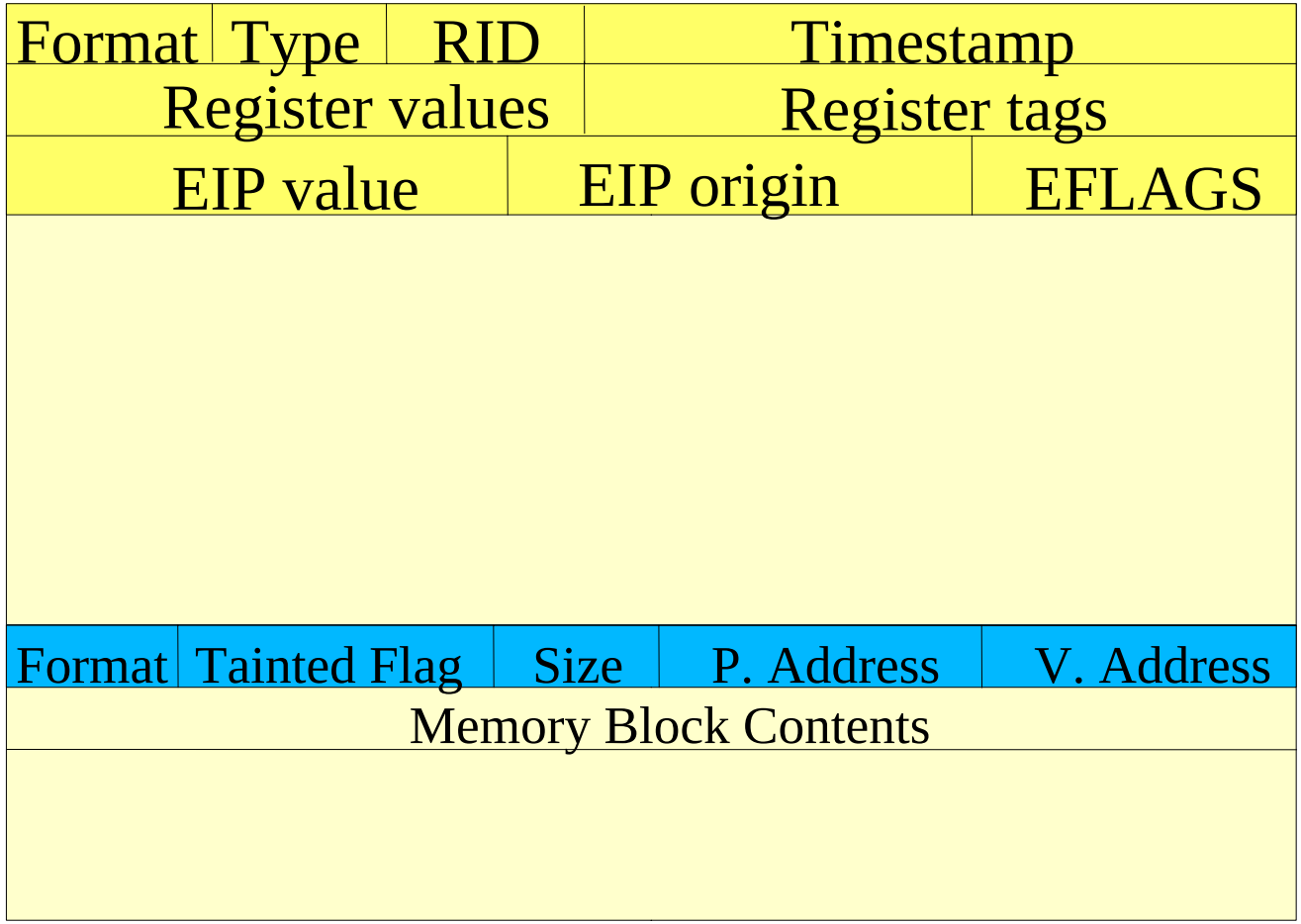
- Jumps
- Function calls
- Function returns
- System calls



SweetBait Design

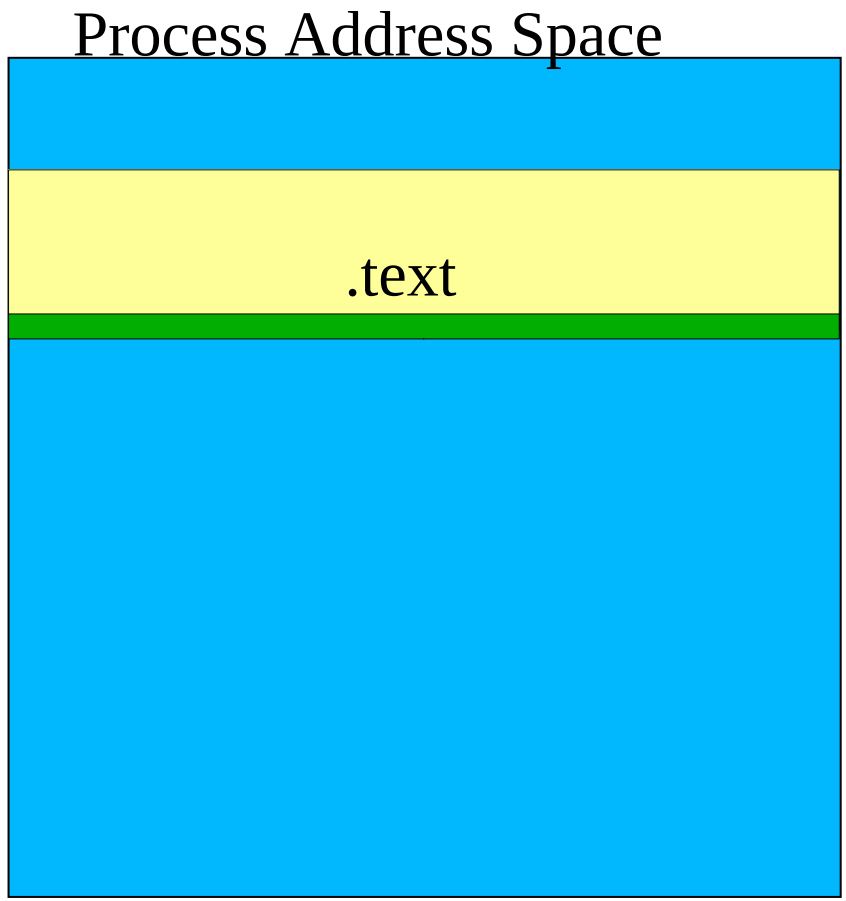


Logs Format


Format
Type
RID
Timestamp
Register values
Register tags
EIP value
EIP origin
EFLAGS
Format
Tainted Flag
Size
P. Address
V. Address
Memory Block Contents

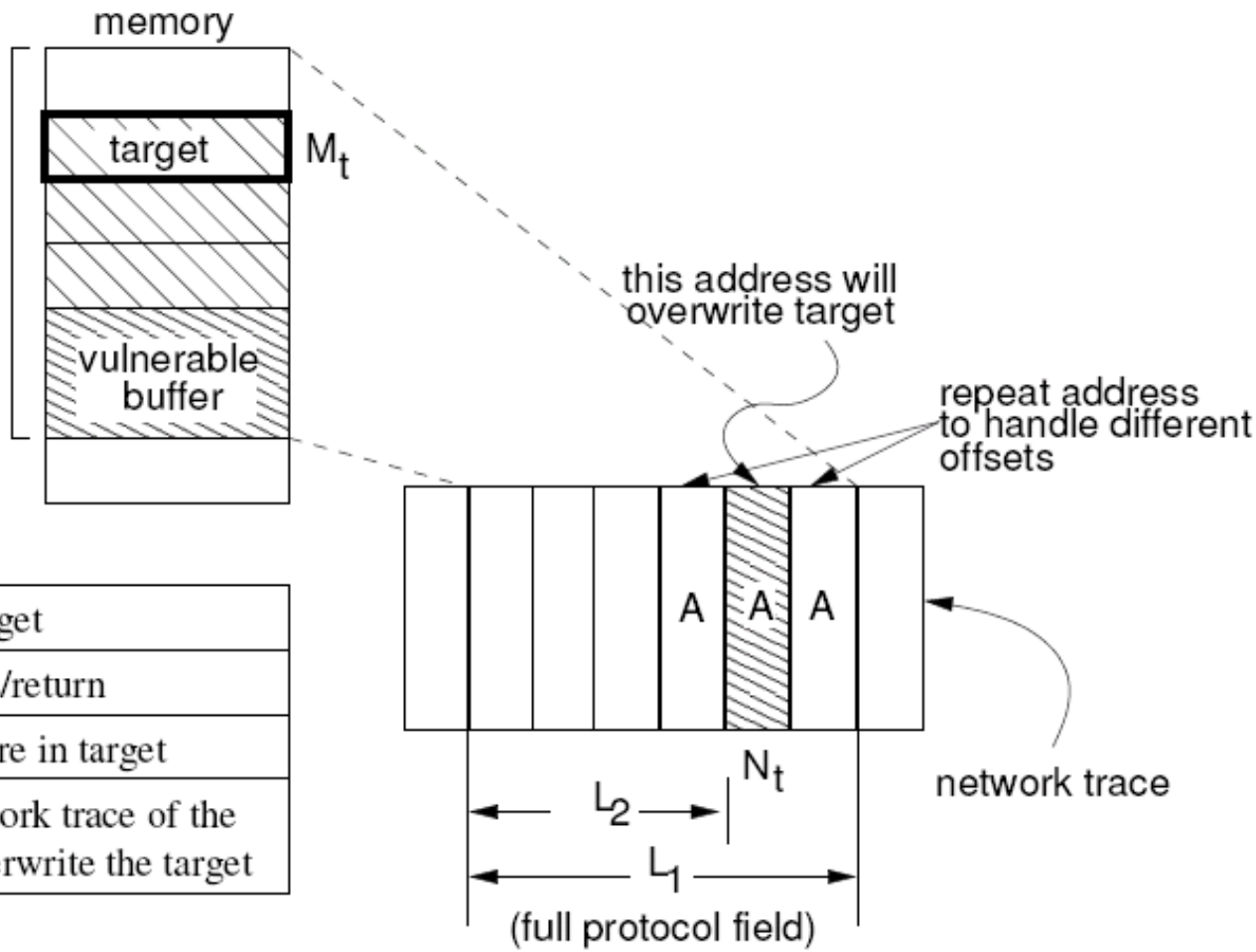
Forensics Shellcode Injection

- Lookup process's read-only pages
- Inject code at last text segment page
- Point EIP to shellcode



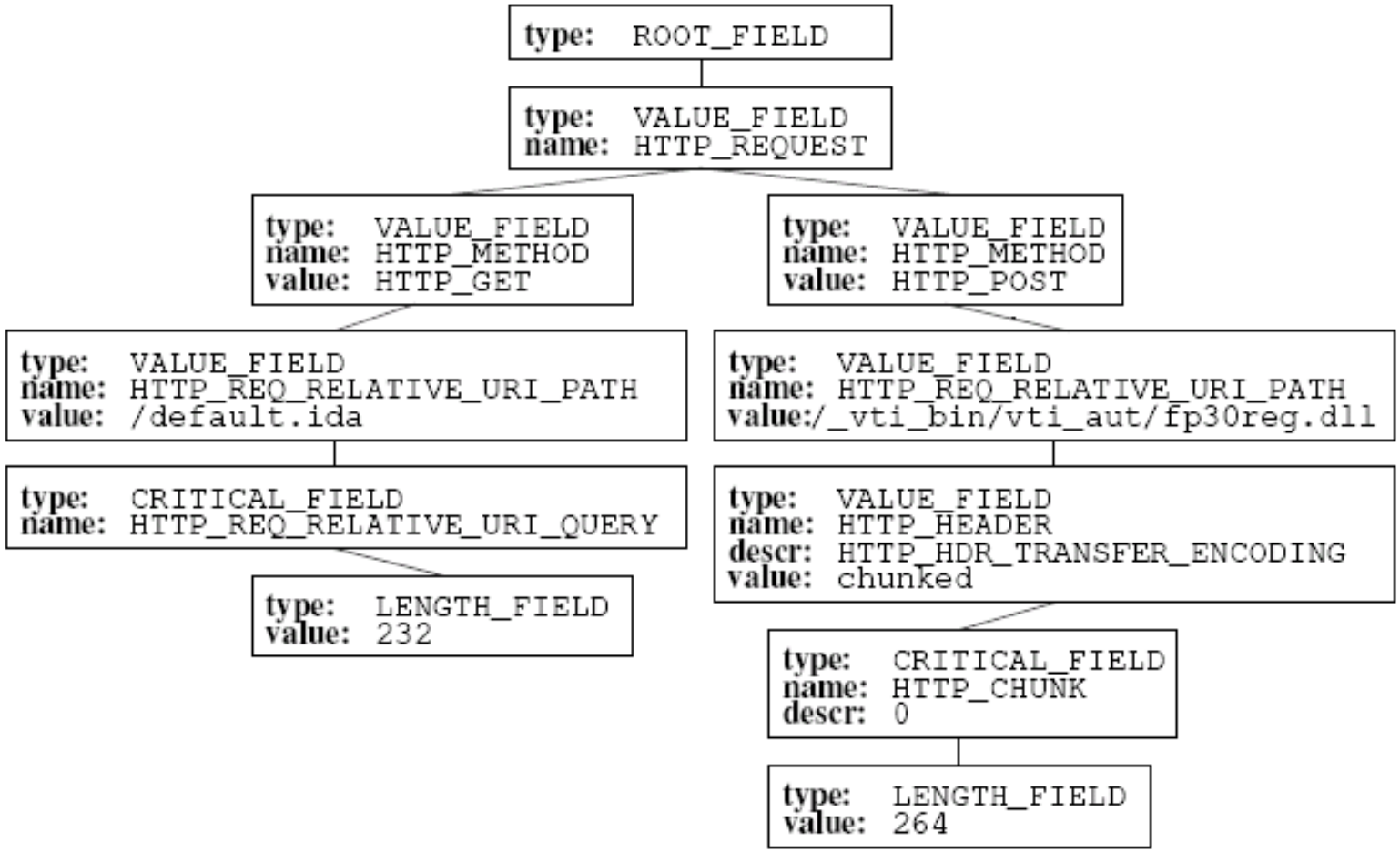
Forensics – The Snitch

- Pid = getpid()
- Rid *[injected by Argos]*
- Connect(localhost)
- Send(pid & rid)
- Listen()
- Accept()
- Read(pid & rid)
- Exec(Netstat or OpenPorts)
- Connect(argos host)
- Send(info)



Legend

M_t	= address of target
target	= target address/return
A	= address to store in target
N_t	= offset in network trace of the bytes that overwrite the target


type: ROOT_FIELD
type: VALUE_FIELD
name: HTTP_REQUEST
type: VALUE_FIELD
name: HTTP_METHOD
value: HTTP_GET
type: VALUE_FIELD
name: HTTP_METHOD
value: HTTP_POST
type: VALUE_FIELD
name: HTTP_REQ_RELATIVE_URI_PATH
value: /default.ida
type: VALUE_FIELD
name: HTTP_REQ_RELATIVE_URI_PATH
value:/_vti_bin/vti_aut/fp30reg.dll
type: CRITICAL_FIELD
name: HTTP_REQ_RELATIVE_URI_QUERY
type: VALUE_FIELD
name: HTTP_HEADER
descr: HTTP_HDR_TRANSFER_ENCODING
value: chunked
type: LENGTH_FIELD
value: 232
type: CRITICAL_FIELD
name: HTTP_CHUNK
descr: 0
type: LENGTH_FIELD
value: 264