Live System Call Trace Reconstruction on Linux

By:
Thanh Nguyen (Nvidia), Meni Orenbach (Nvidia), and Ahmad Atamli (Nvidia)

From the proceedings of
The Digital Forensic Research Conference
DFRWS USA 2022
July 11-14, 2022

LIVE SYSTEM CALL TRACE RECONSTRUCTION ON LINUX

Thanh Nguyen, Meni Orenbach, Ahmad Atamli

WHO ARE WE?

Introduction



System Security Research Group

Working on zero trust security solutions
App Shield
DPU hardware security

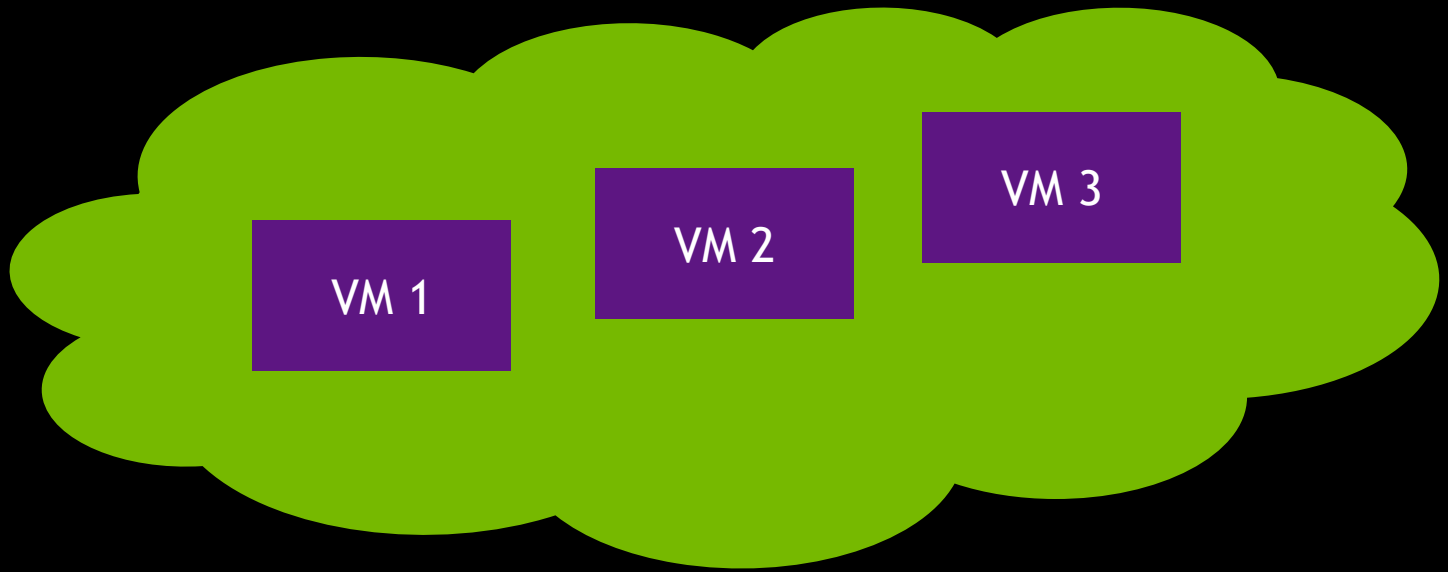
AGENDA

Live System Call Trace Reconstruction on Linux

- Background & motivation
- System call tracing
- Results
- Conclusion

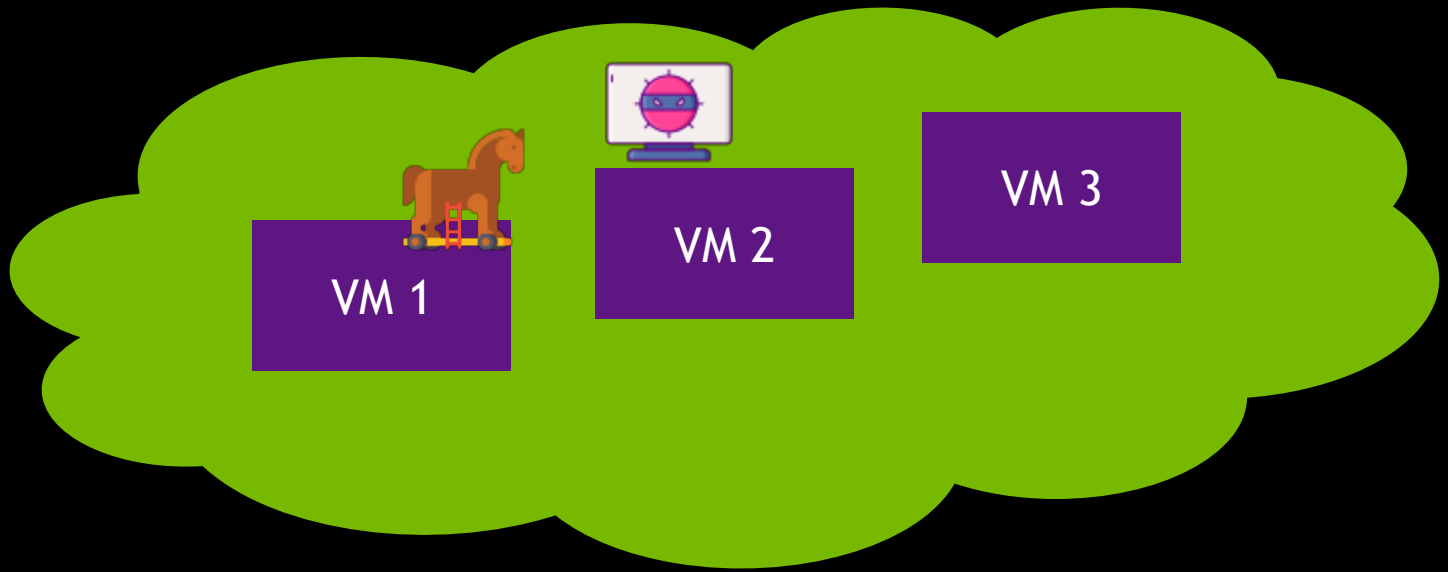
CLOUD COMPUTING

A promising direction



CLOUD COMPUTING

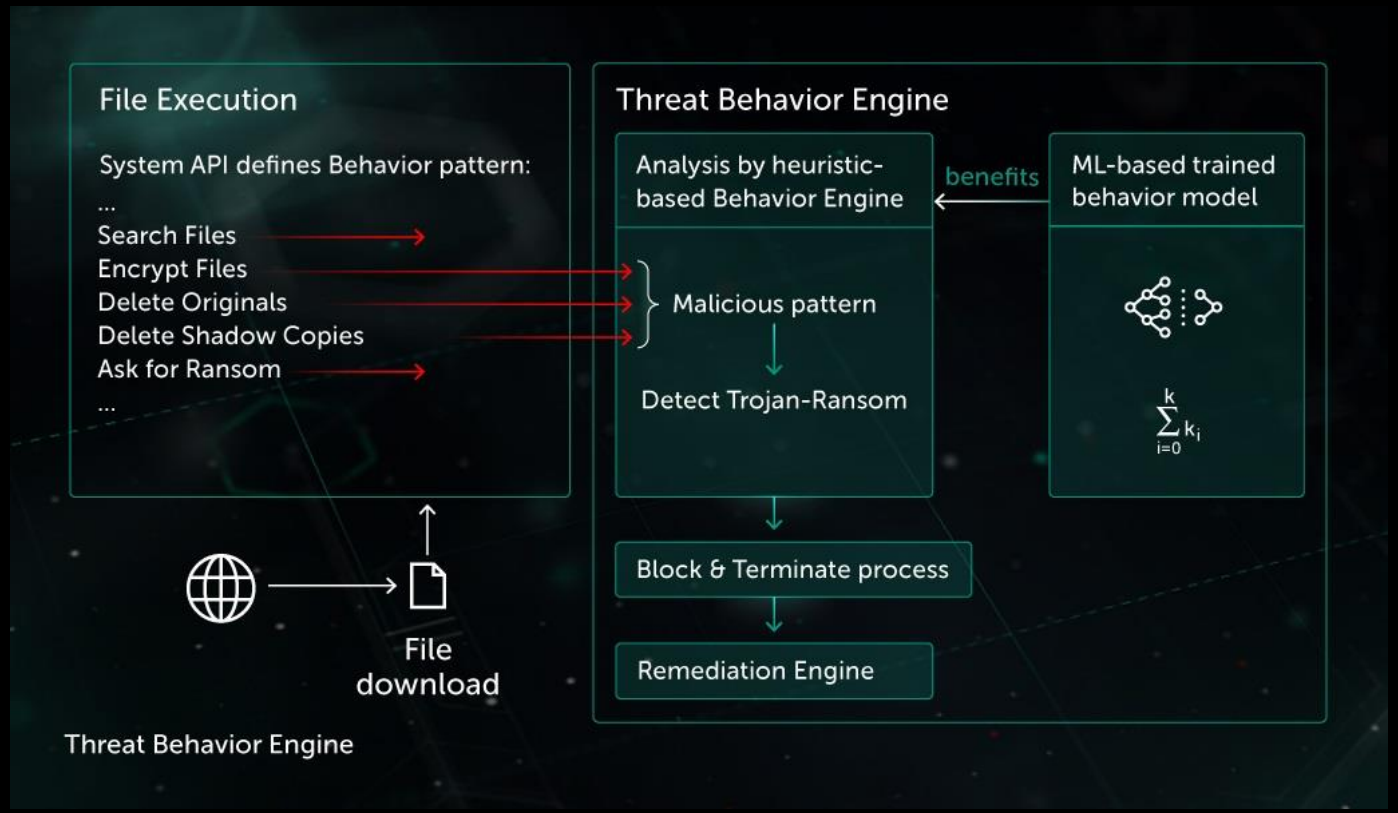
The sad reality



Source: https://icons8.com

MALWARE DETECTION

Behavior analysis



Source: https://www.kaspersky.com/enterprise-security/wiki-section/products/behavior-based-protection

MALWARE DETECTION

Behavior analysis

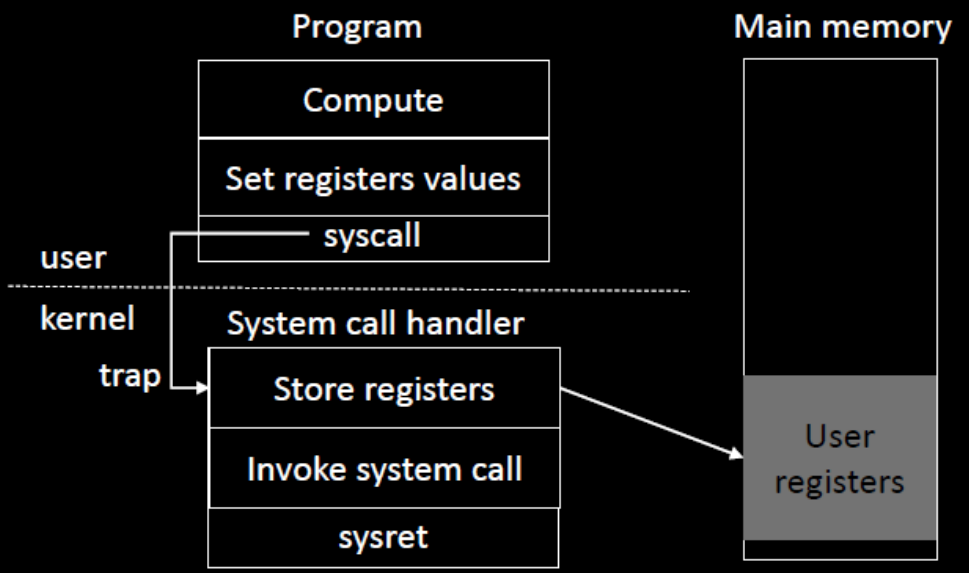
- Promising direction that enables
  - Fileless malware
  - Ransomware
  - Zero-day malware

Our focus: system call-based behavior

SYSTEM CALL INVOCATION

Background

- Applications run in unprivileged mode
- System calls rely on trap handler
- Transfer control to the kernel in privileged mode
- Via special instructions, e.g., syscall in x86
- System call ABI relies on CPU registers



What is the state-of-the-art tools to extract system call traces?

SYSTEM CALL EXTRACTION

strace

- Linux based tracing utility for system calls and signals
- Invoked with the to-be-traced program
- Part of the guest VM

```
[00007f6142f41c38] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|
O_CLOEXEC) = 3
[00007f6142f41cf8] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360\2
15\2\0\0\0\0\0"..., 832) = 832
[00007f6142f41d2e] pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0
\0\0\0\0\0\0"..., 784, 64) = 784
[00007f6142f41d2e] pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\
3\0\0\0\0\0\0\0", 32, 848) = 32
[00007f6142f41d2e] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0~\303\347M\250B\312
<j\233\242\v!0<\341"..., 68, 880) = 68
[00007f6142f41a09] fstat(3, {st_mode=S_IFREG|0755, st_size=1995896, ...}) = 0
```

SYSTEM CALL EXTRACTION

strace

strace is not a silver bullet

observer effect

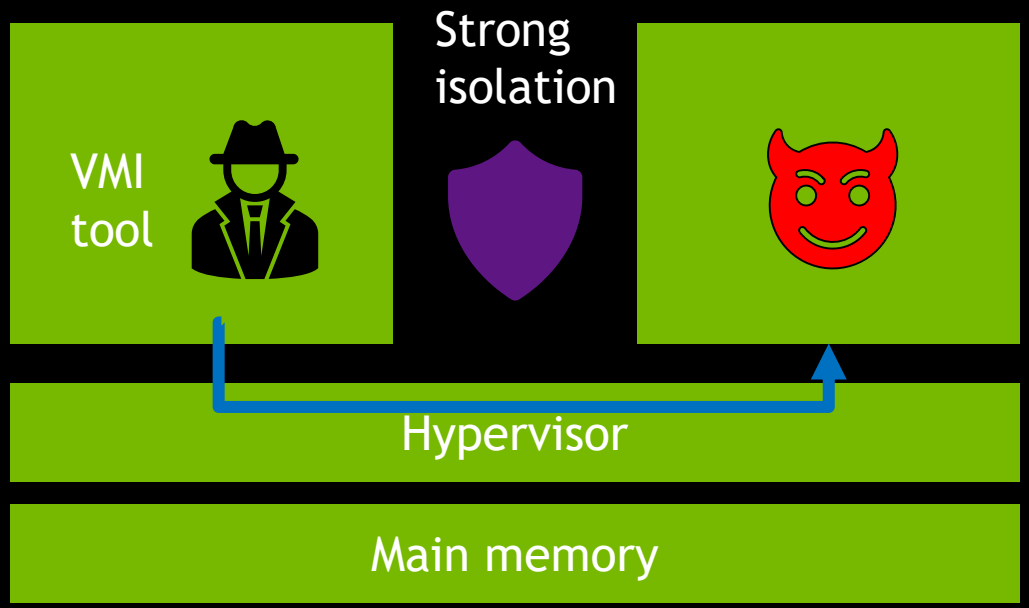
Malware became sophisticated.
Identify tracing tools & disable them

Better defense: Hide the tracing tool with virtual machine introspection!

VIRTUAL MACHINE INTROSPECTION

Background

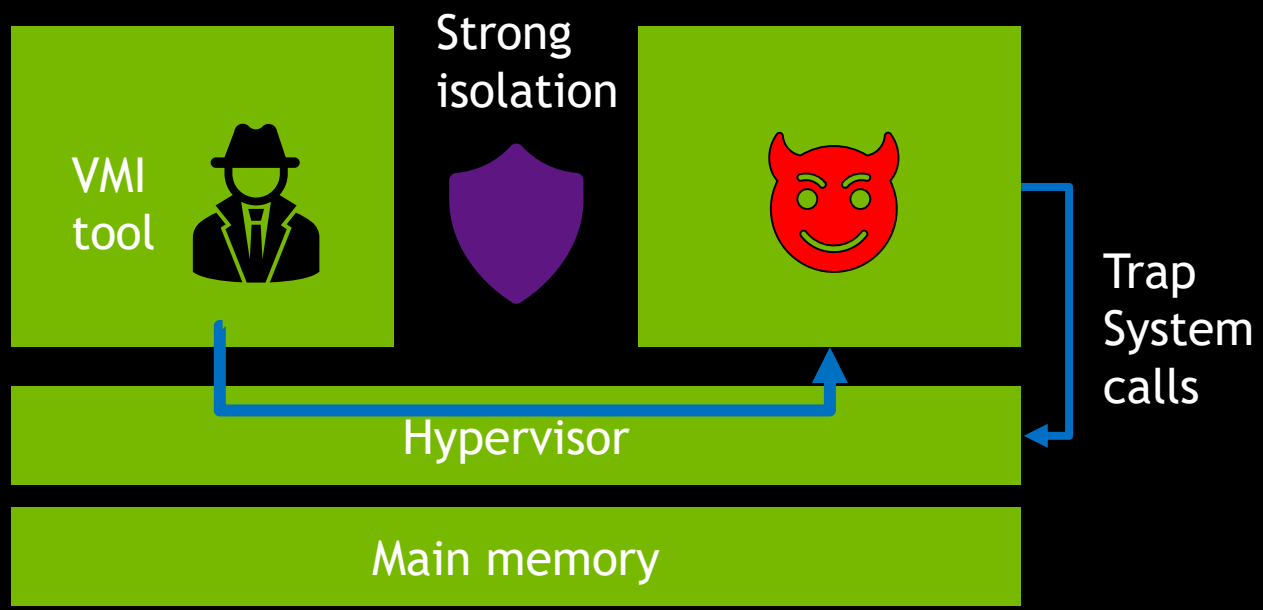
- VMI is isolated from the guest
- VMI enables memory acquisition and inferring high-level semantic information from it



SYSTEM CALL EXTRACTION

Virtual machine introspection (VMI)

- VMI tool configures hypervisor to trap system call for introspected guests
- VMI tool trace each executed system call



SYSTEM CALL EXTRACTION

Virtual machine introspection (VMI)

- Traps are expensive
  - VM exits, high performance penalty
- Hinder adoption – slows down applications being traced
- Timing observer effect – sophisticated malware can detect being slowed down

This is an intrusive method!

SYSTEM CALL EXTRACTION

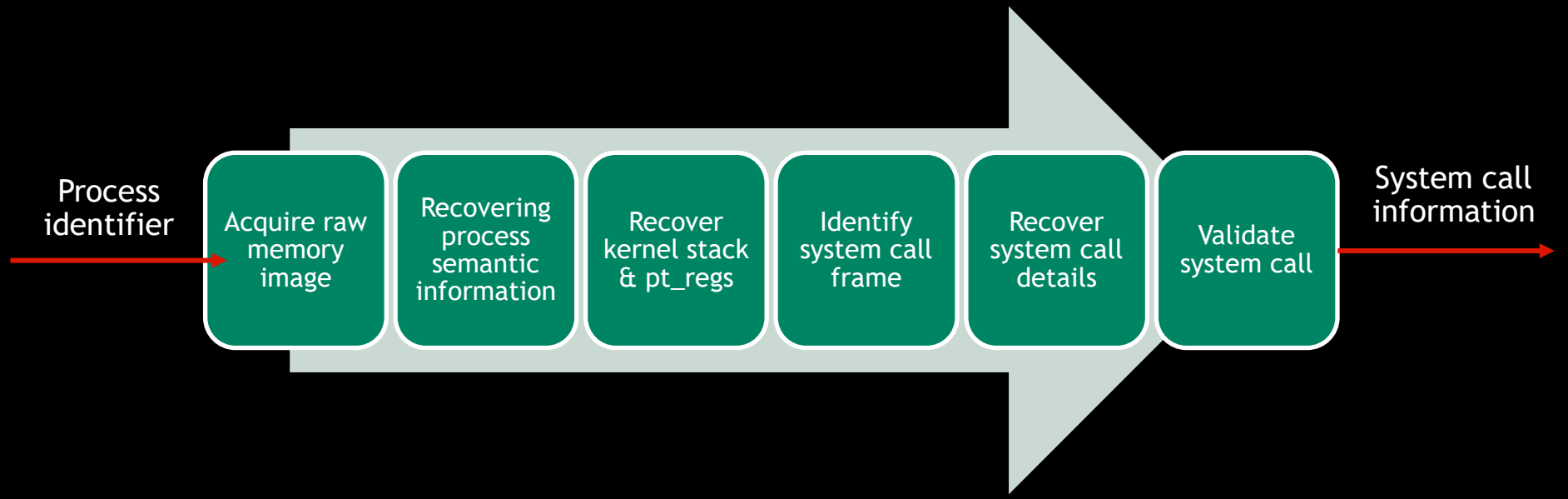
Non-intrusive system call tracing

How?

SYSTEM CALL EXTRACTION

Non-intrusive system call tracing

- Our approach based on memory introspection and memory forensics



- To analyze system calls from main memory, we first provide background information

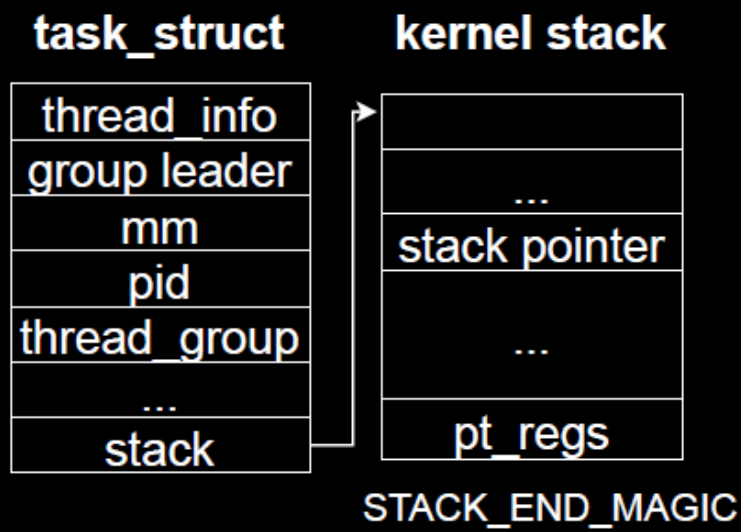
ACQUIRING RAW MEMORY IMAGE

Common Approach

RECOVERING PROCESS INFORMATION

From raw memory image

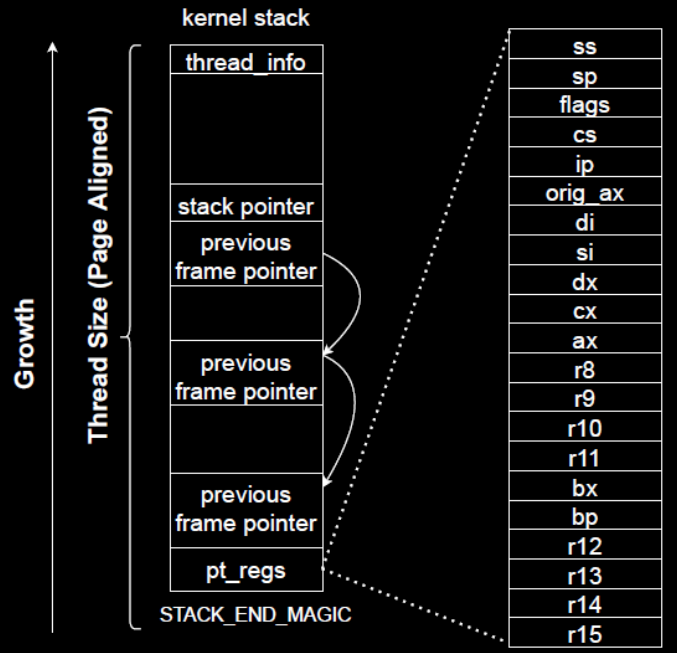
- task_struct structure contains per-process information
  - Obtained from Linux's intermediate symbol table (IST)
- The task_struct contains a pointer to the *kernel stack*
- Kernel stack is used as a traditional stack when invoking functions in the kernel
- System call handler store registers' values at the bottom of the stack in pt_regs
  - pt_regs in older kernels was pointed to by sp0 field which no longer serves this function
  - How to get pt_regs in newer kernels?



KERNEL STACK & REGISTERS

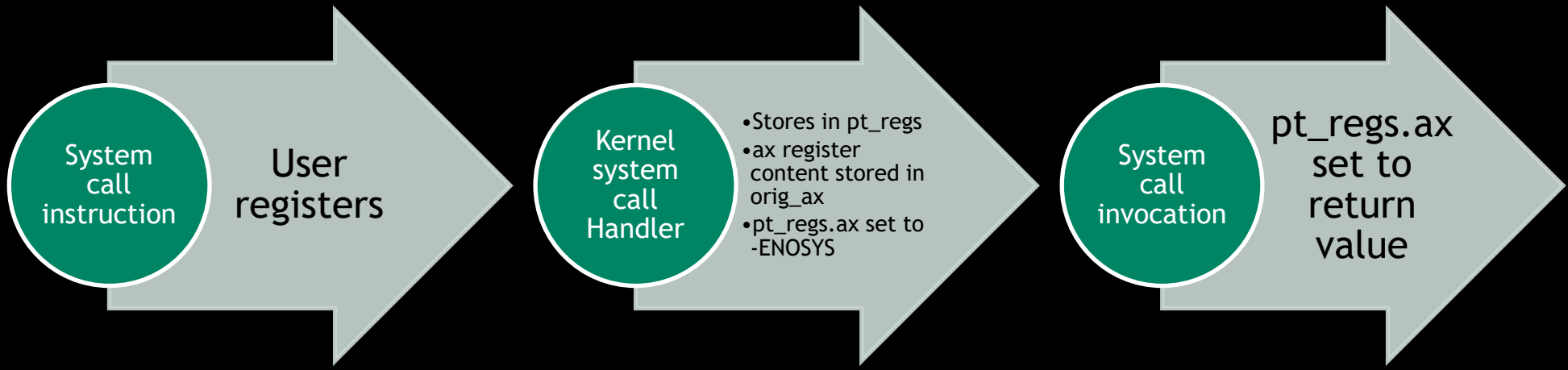
From raw memory image

- The IST does not contain the kernel stack size
  - Required to compute the exact address of pt_regs
  - We propose methods to extract it (see the paper)
- System call identifier stored in the ax register
  - Kernel contains a system call identifier table
  - Extract system call semantic information
- Other registers contain arguments to system calls



IDENTIFY SYSTEM CALL FRAME

From raw memory image



RECOVER SYSTEM CALL DETAILS

From raw memory image

System Call ID	System Call Name
0	Read
1	Write
2	Open

Bridging the semantic gap

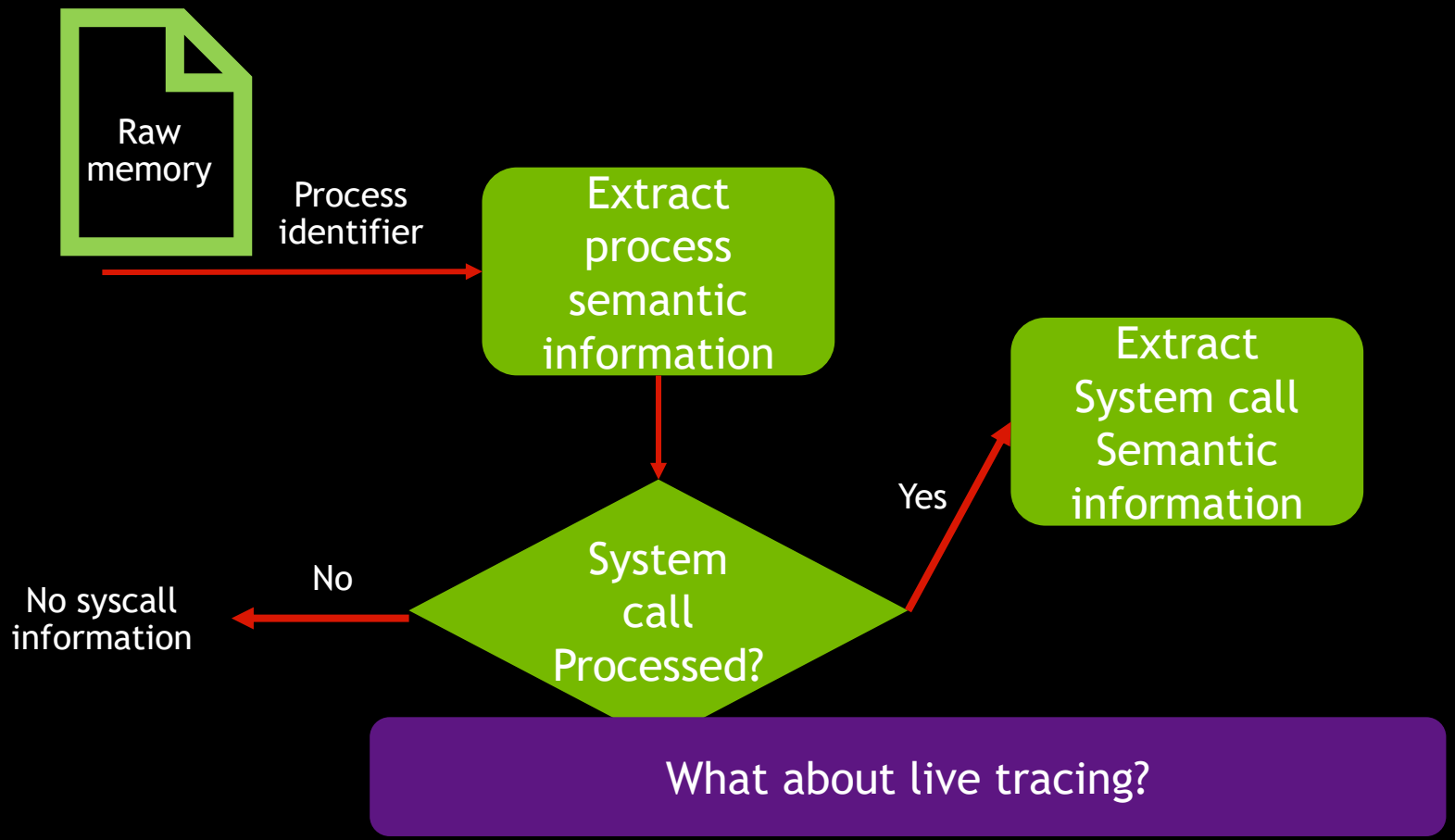
VALIDATION OF SYSTEM CALL

Extracted from raw memory image

- The ax register can store more then just system call number
- Invoking a system call places the next instruction in the pt_regs.ip register as the return address
- Instruction before the return address should contain a system call instruction opcode
- Opcode used to verify if a structure has valid system call before extraction and analysis

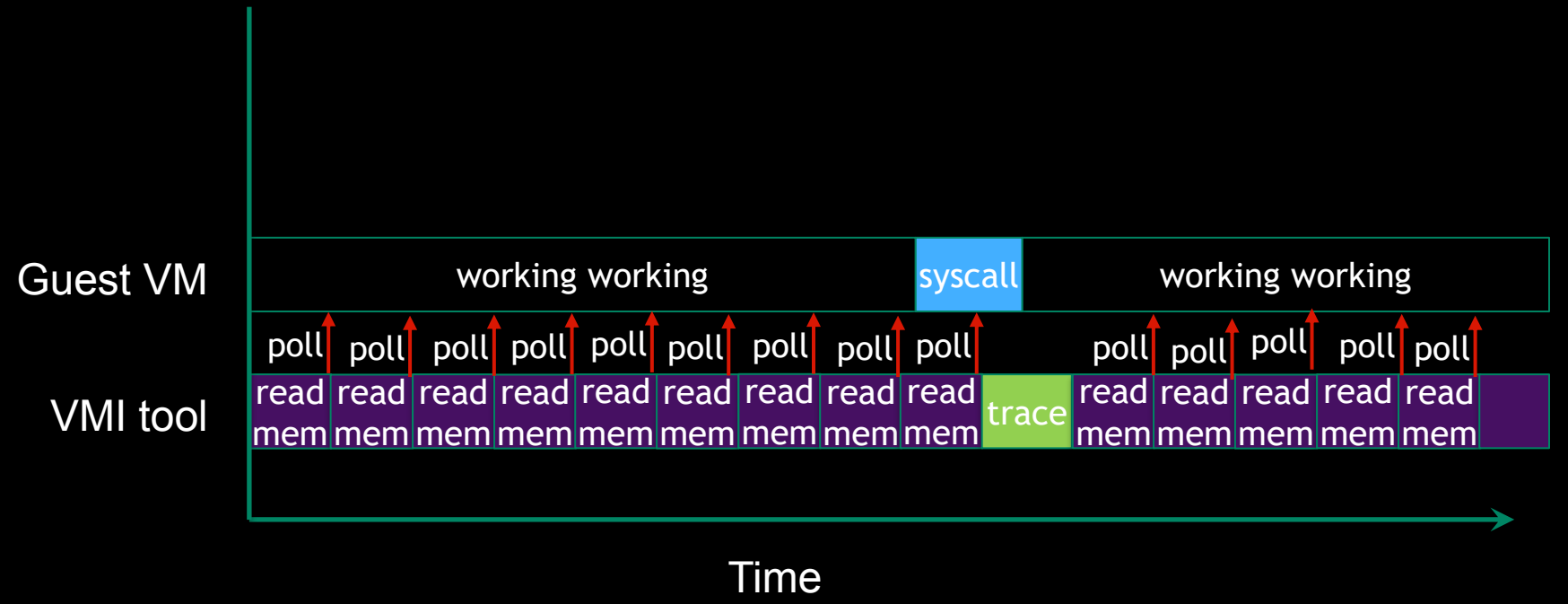
EXTRACTING SYSTEM CALL

From raw memory image



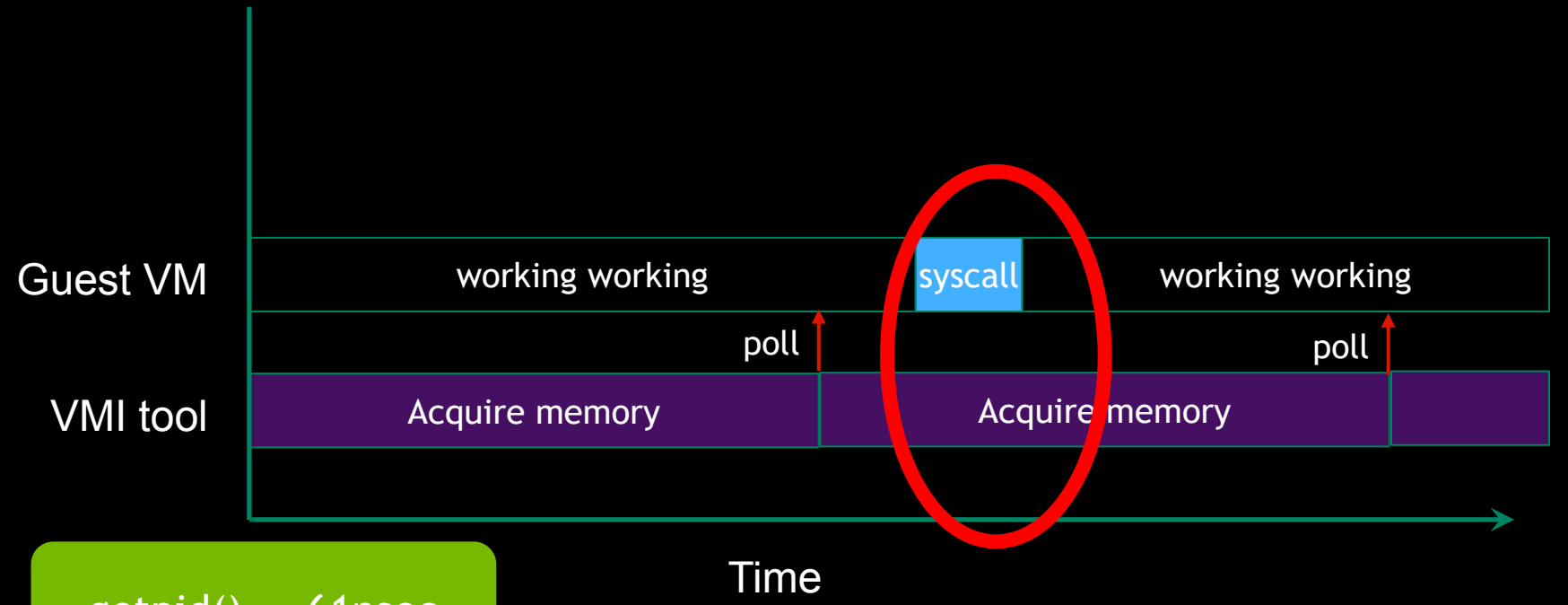
LIVE SYSTEM CALL TRACING

Timing matters



LIVE SYSTEM CALL TRACING

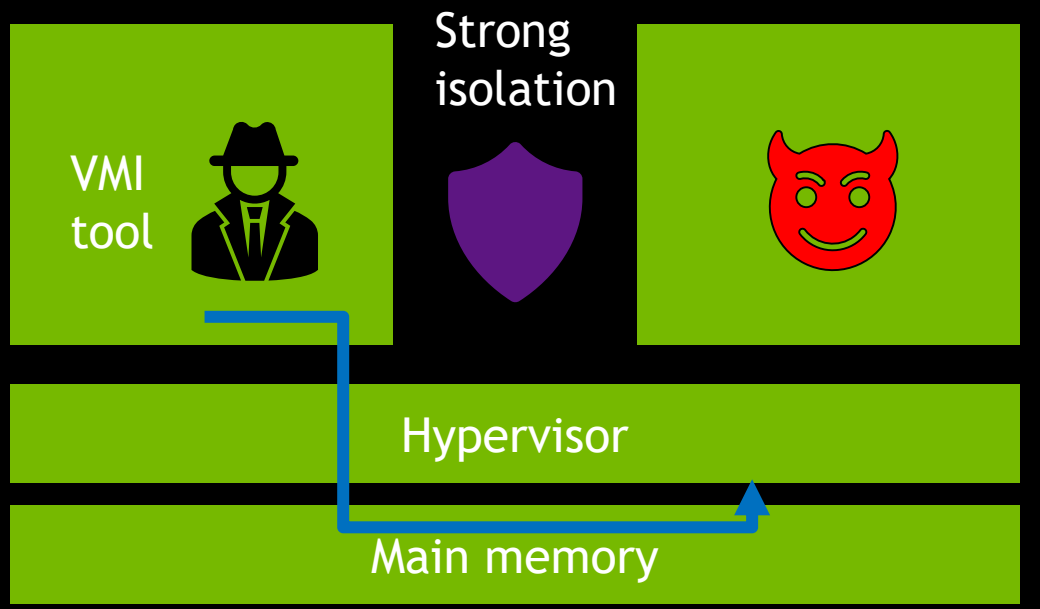
Timing matters



MEMORY ACQUISITION

LibVMI

- Based on UNIX sockets
- Reading memory requires system call invocation from LibVMI to VMI tool
    - Too slow for live system call tracing



MEMORY ACQUISITION

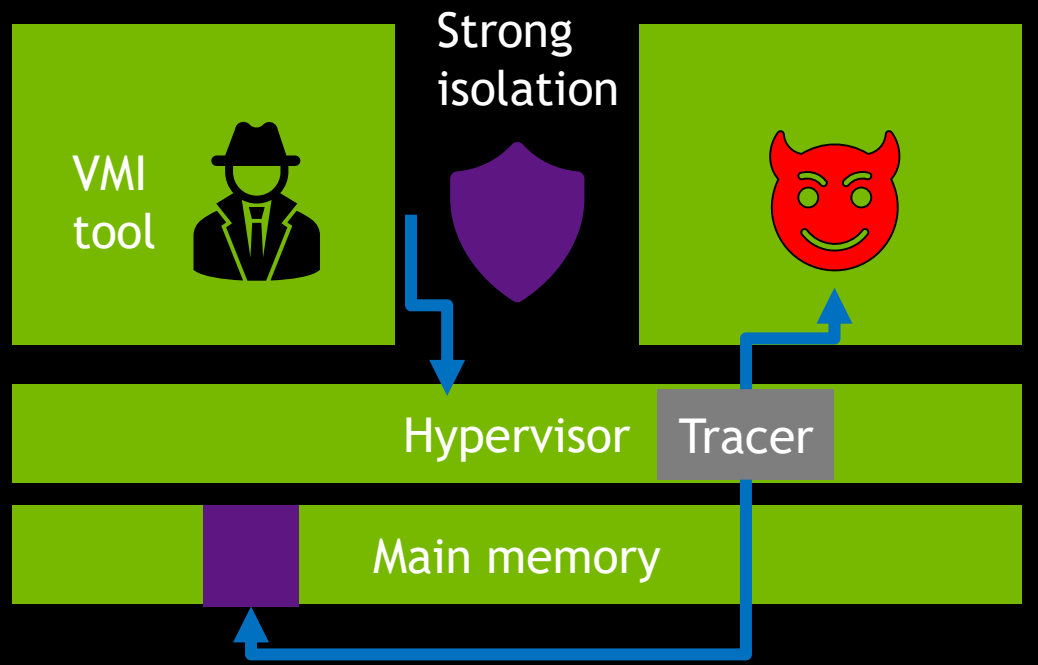
Low latency

- Based on shared memory
- Hypervisor tracer thread

  checks if system call if processed

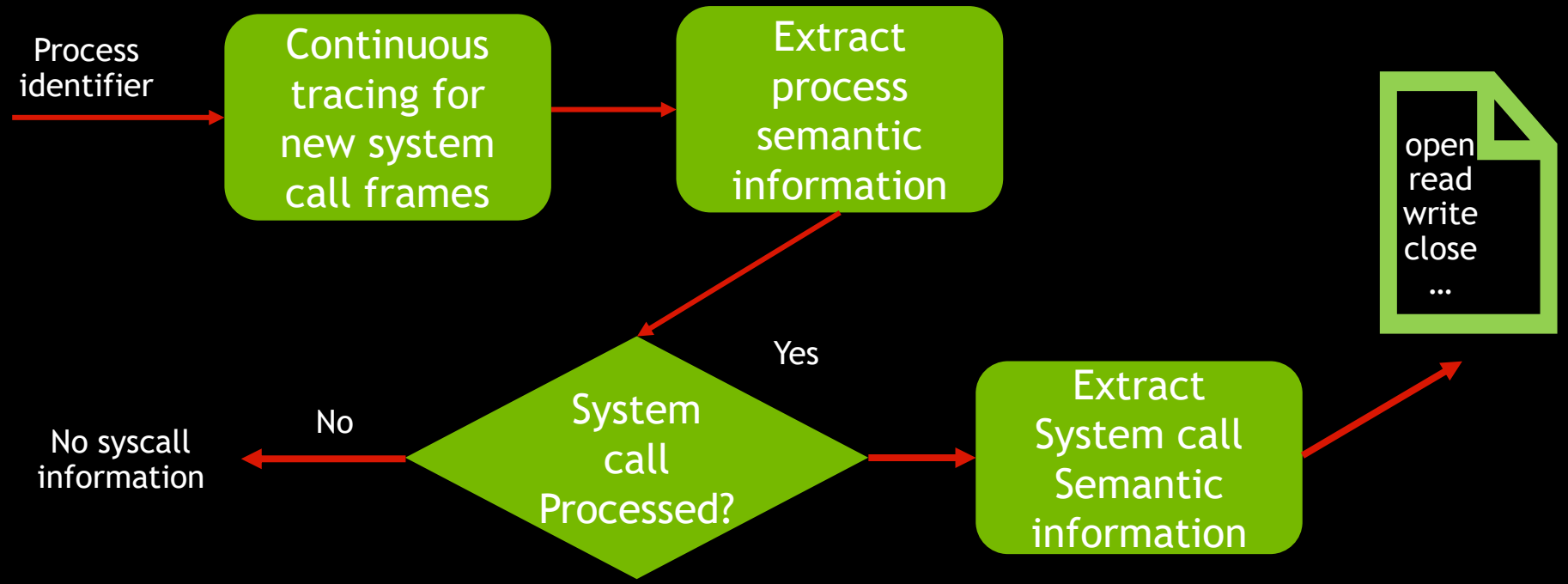
  in traced guest
- Sends information to VMI tool

  over shared buffer



PUTTING IT ALL TOGETHER

Live non-intrusive system call tracing



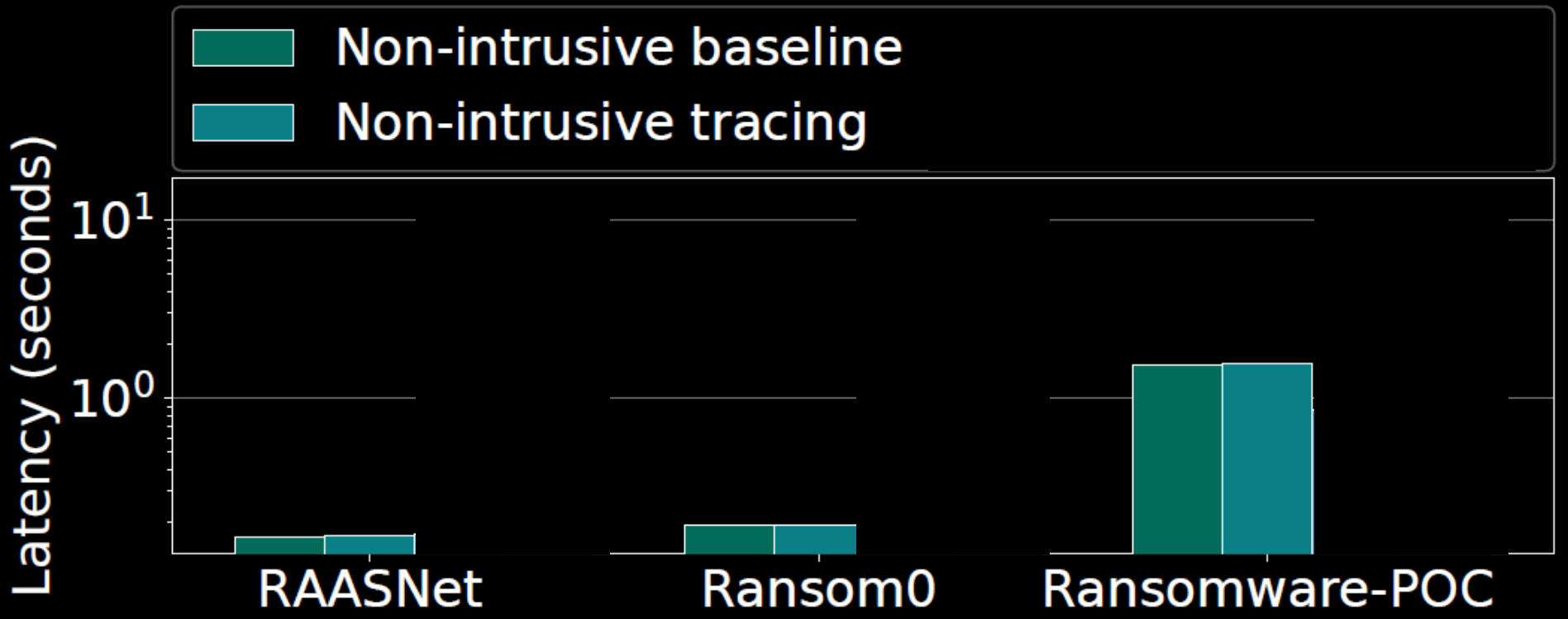
RESULTS

Methodology

- Implemented intrusive and non-intrusive system call tracers in LibVMI
- Traced benign and malicious applications
- Synchronized start between our tracers and the applications
- System call tracing validation by comparing to strace
- Latency measurement performed from the hypervisor to avoid timing skews in guests

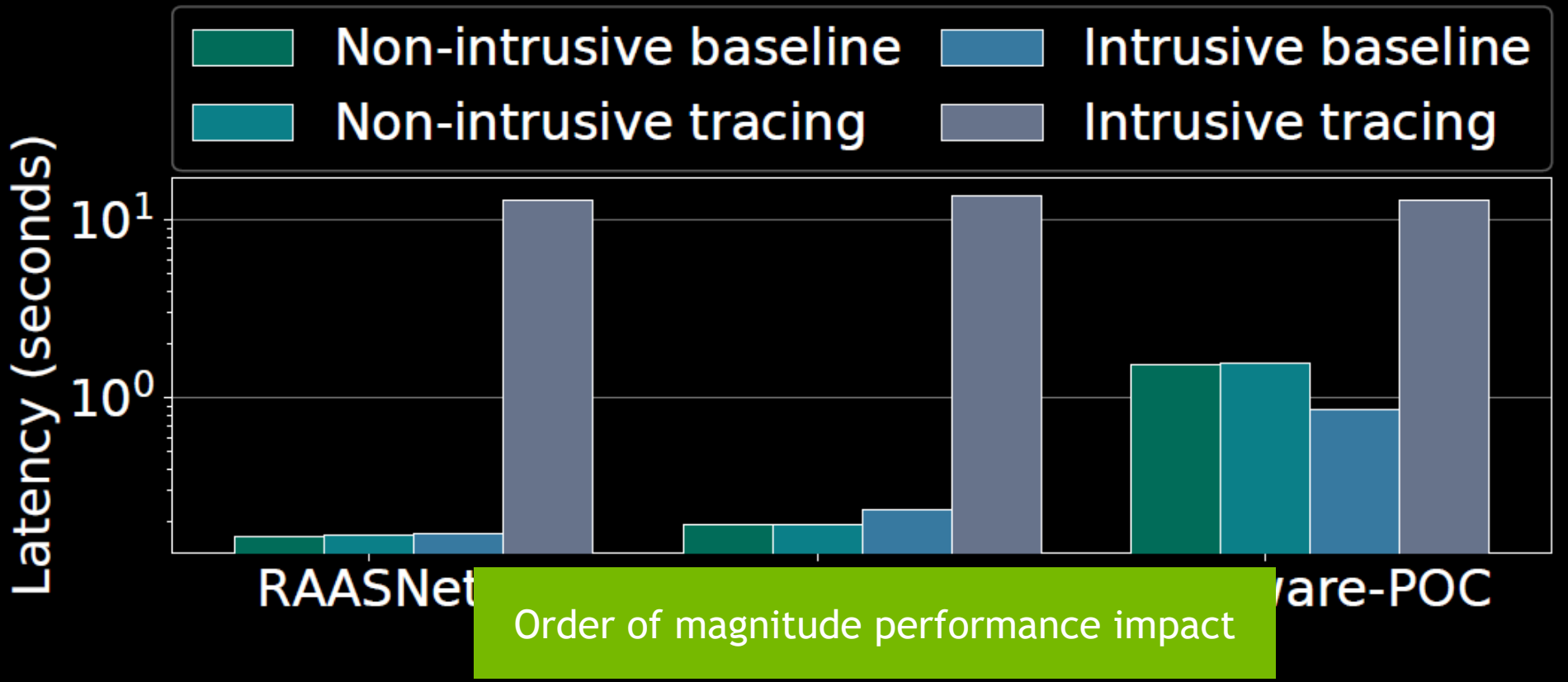
LATENCY

Evaluation



LATENCY

Evaluation



Order of magnitude performance impact

ACCURACY RATE

Evaluation

Benign applications

Program	Number of system calls
lspci	959
netstat	1,398
ps	2,229

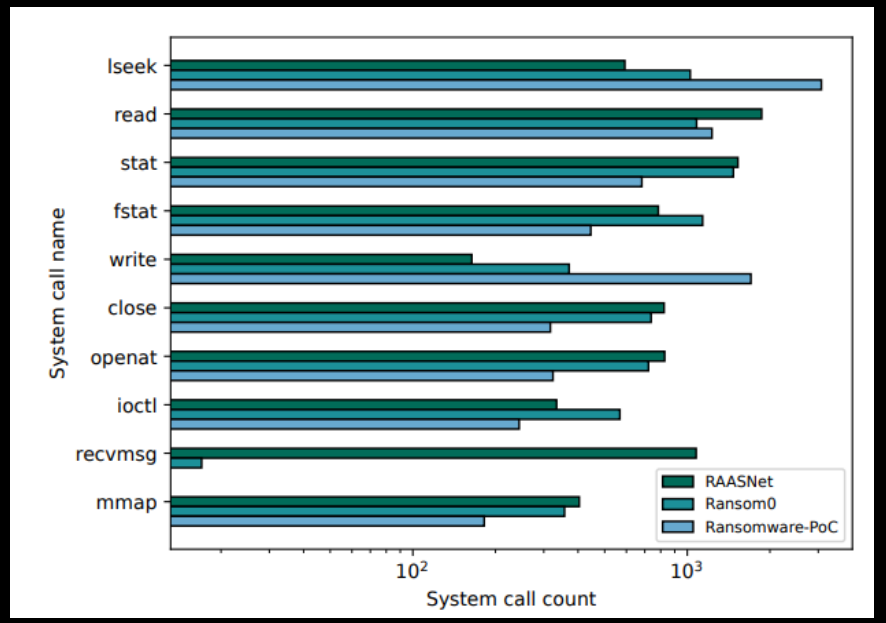
Malicious applications

Program	Number of system calls
RAASNet	9,694
Ransom0	13,811
Ransomware-PoC	11,522

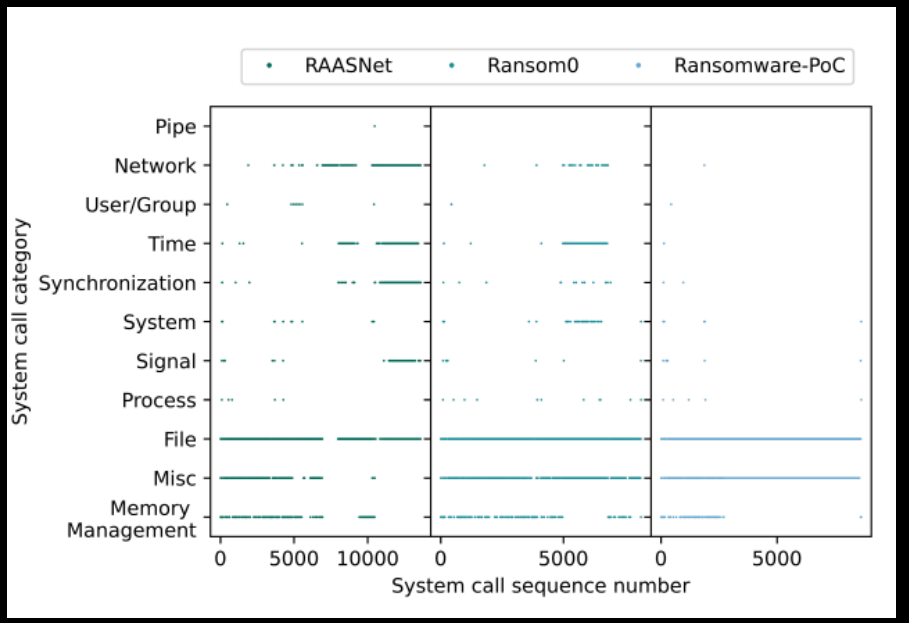
100% accuracy

MALWARE CASE STUDY

Evaluation



Top 10 system calls



Clustered by category

CONCLUSION

- System call tracing facilitates next-generation VM protection
  - Memory forensic methods are instrumental to bridge the semantic gap
  - Methods need to be updated to work with latest kernels
- Malware are becoming ever more sophisticated
- Non-intrusive tools are required to reduce timing observation
- Increase adoption potential due to lower performance impact on traced applications

Thank you!
Any questions?