SAND2020-6433PE

# CHIRP

## Cloud Hypervisor Forensics and Incident Response Platform

Vince Urias

SANDIA NATIONAL LABORATORIES

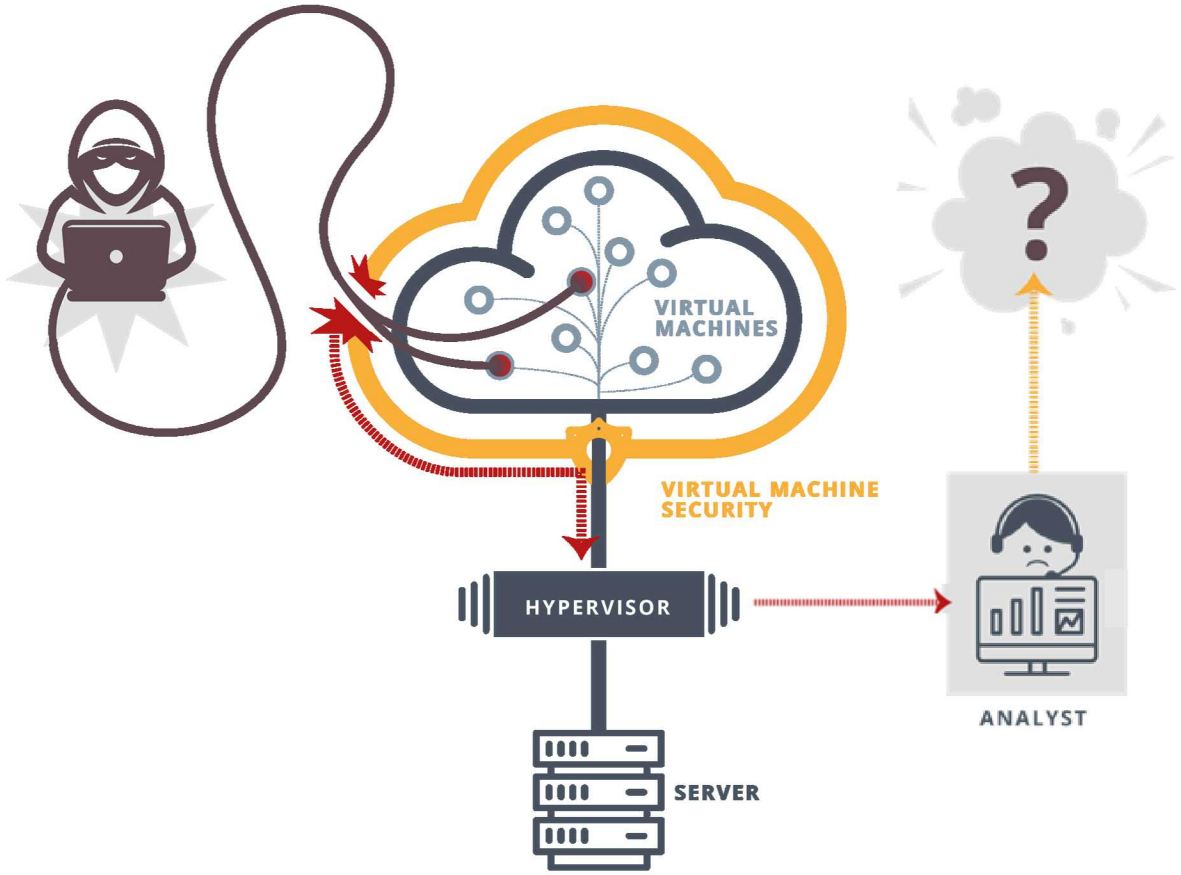
DOE PACT VIRTUAL SHOWCASE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

# Business Problem



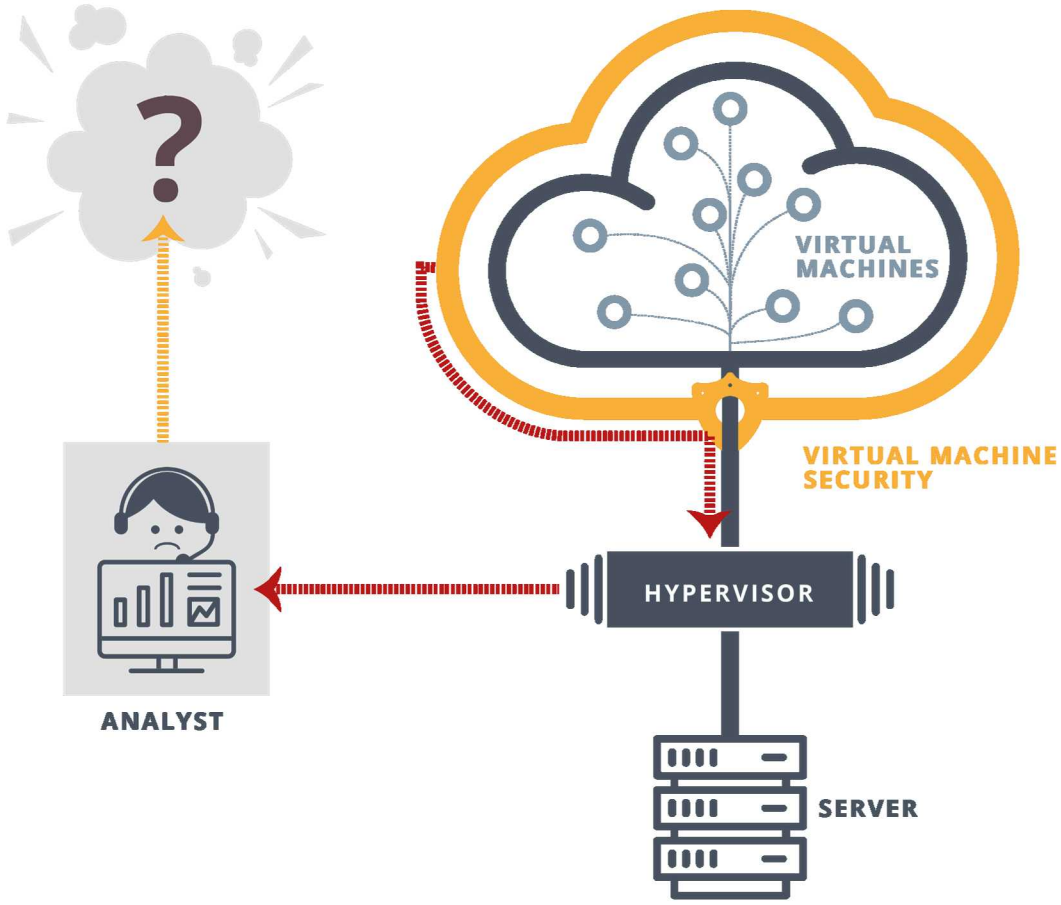
When cyber incidents occur in the cloud, the SOC Analyst has no visibility beyond the hypervisor.



# Technical Challenge

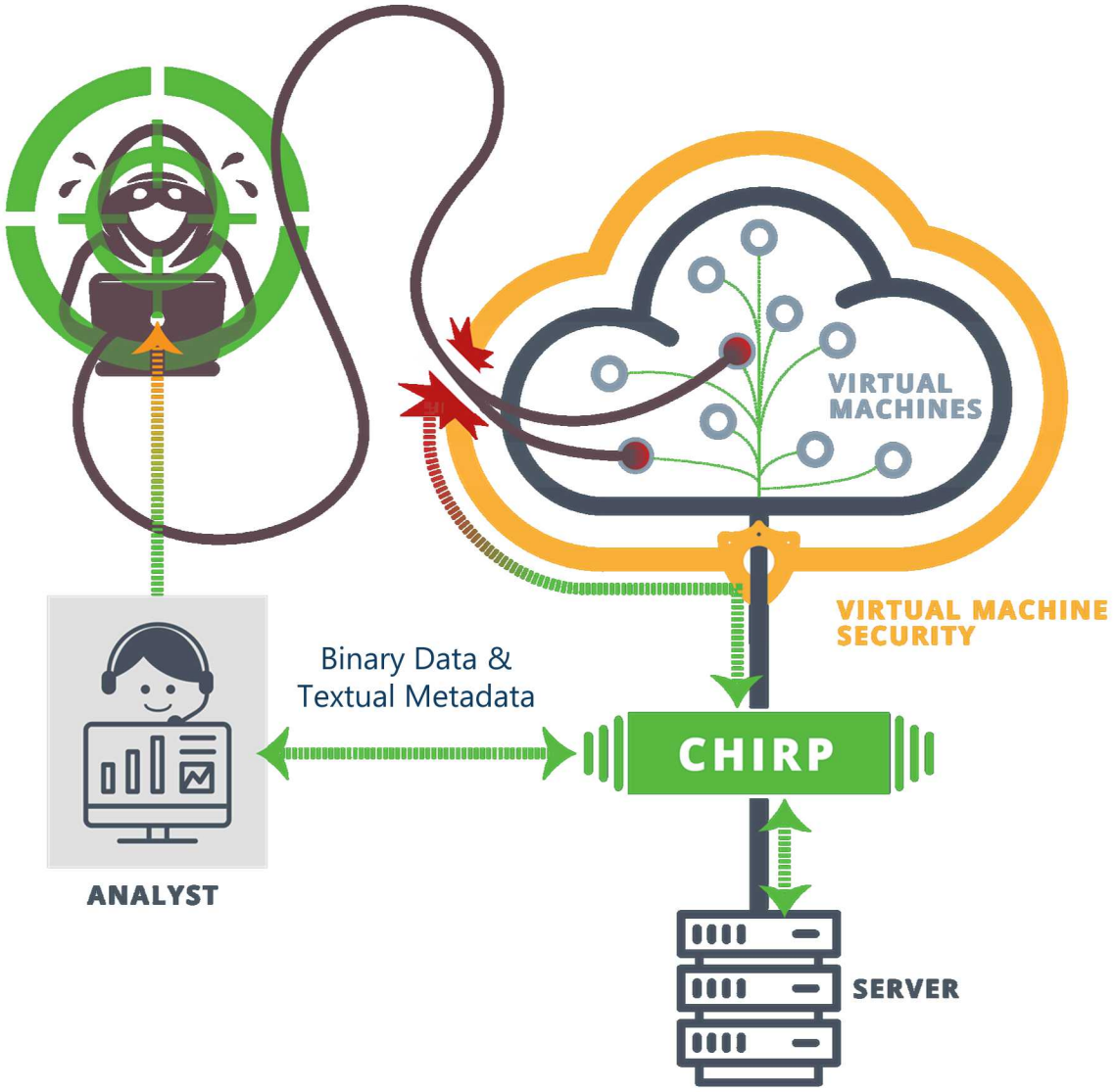
CHIRP

- Ephemeral nature of cloud environment
- Hypervisor instrumentation with minimal overhead
- Hypervisor agnostic
- OS agnostic
- Common data model
- Broad data set collection



# Solution Architecture

CHIRP




## Virtual Machine Introspection (VMI)

- Sandia has developed a custom VMI implementation to address the issue of overhead.
- Platform agnostic
- Text and binary data collection
- Allows correlation with other sources

# Development History & Results

CHIRP

Principal Investigator
Vince Urias

- 20 Years of Cyber experience at Sandia Labs
- Numerous national awards

Developmental History

**2016** Initial Development starts

**2017** First disclosed (Dec)

**2019** R&D 100 Winner (May)

Funding History
~$1M LDRD

Lead Sandia National Laboratories

Partner splunk>

Research Team
William Stout
Caleb Loverro

Technology Readiness Level (TRL):
TRL 6 as of June 2020

IP Protection

- Patent Application # 16/051,005 filed July 31, 2018
- Copyright approved for commercial licensing – May 2018

Market Validation
2 government deployments

# Technical Requirements & Benefits

CHIRP

## TECHNICAL REQUIREMENTS

- Access to the Hypervisor
- Designed for IaaS from the start
- Off prem and on prem cloud-ready

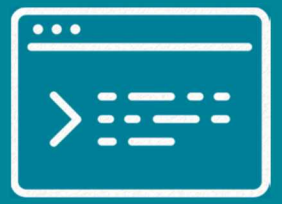
## BENEFITS

- Lightweight
- Real-time dynamic response
- Configurable logging → incident response or forensics

# Technology Roadmap

CHIRP

Management Dashboard



Expanded Analytics

Improve Deployment

Rigorous Documentation

Private sector pilot for market validation

Bringing CHIRP to Market

# CHIRP

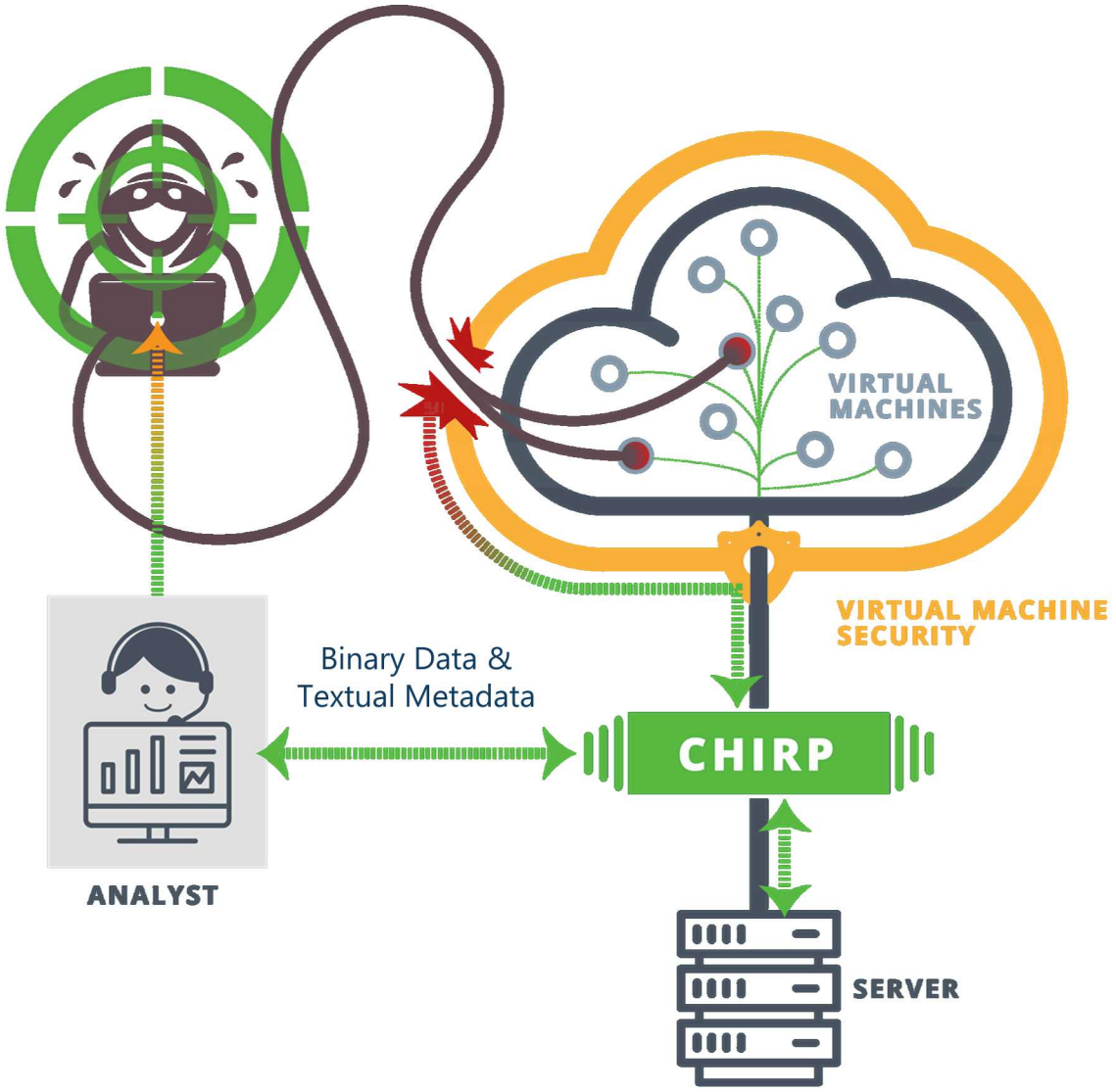
## Cloud Hypervisor Forensics and Incident Response Platform

DOE PACT VIRTUAL SHOWCASE

DEMONSTRATION

# Solution Architecture



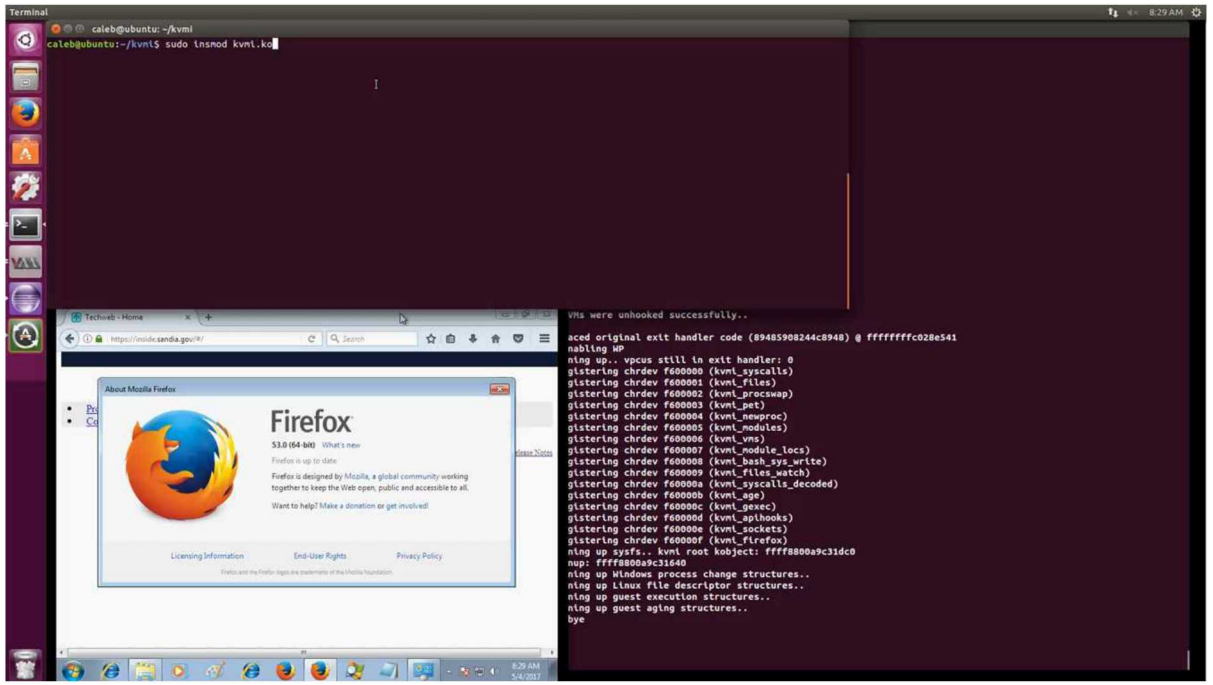


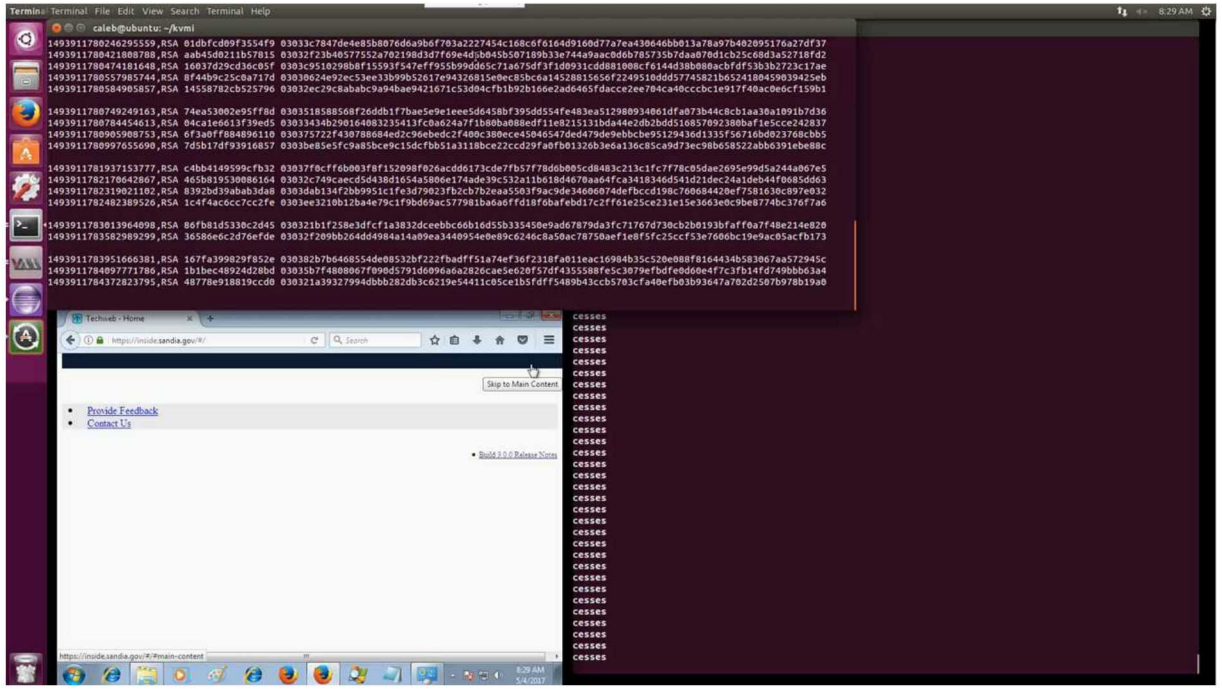

## Vignette Demonstrations

- Module Dumping
- Key Extraction
- Shell
- Carving
- System Call Decoding
- Reverse Engineering Pipeline
- Big Data Extraction

# Key Extraction

CHIRP
DEMONSTRATION



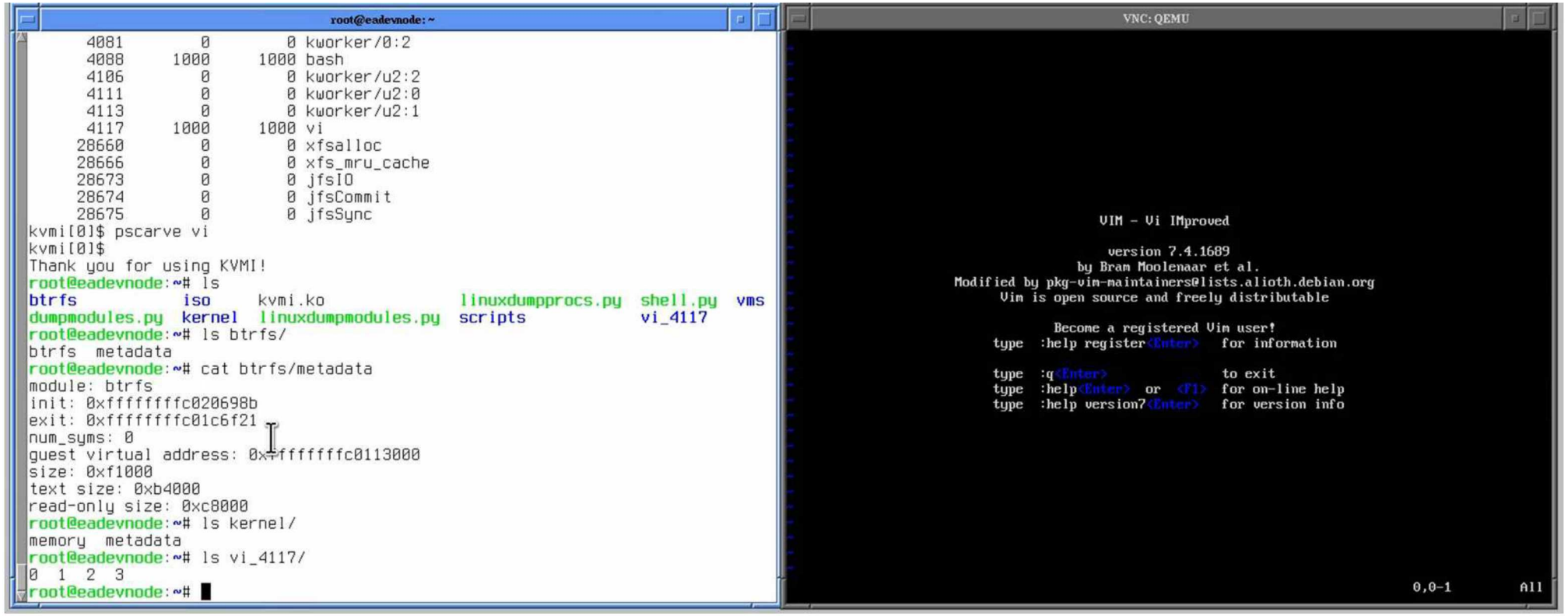




Keys used for encryption are extracted from processes (e.g., Firefox, Windows LSASS, etc.)
Here, Firefox browses to sandia.gov, RSA keys for the session are extracted.

Able to see within the session… eliminating man-in-the-middle attacks.

# Carving

CHIRP
DEMONSTRATION



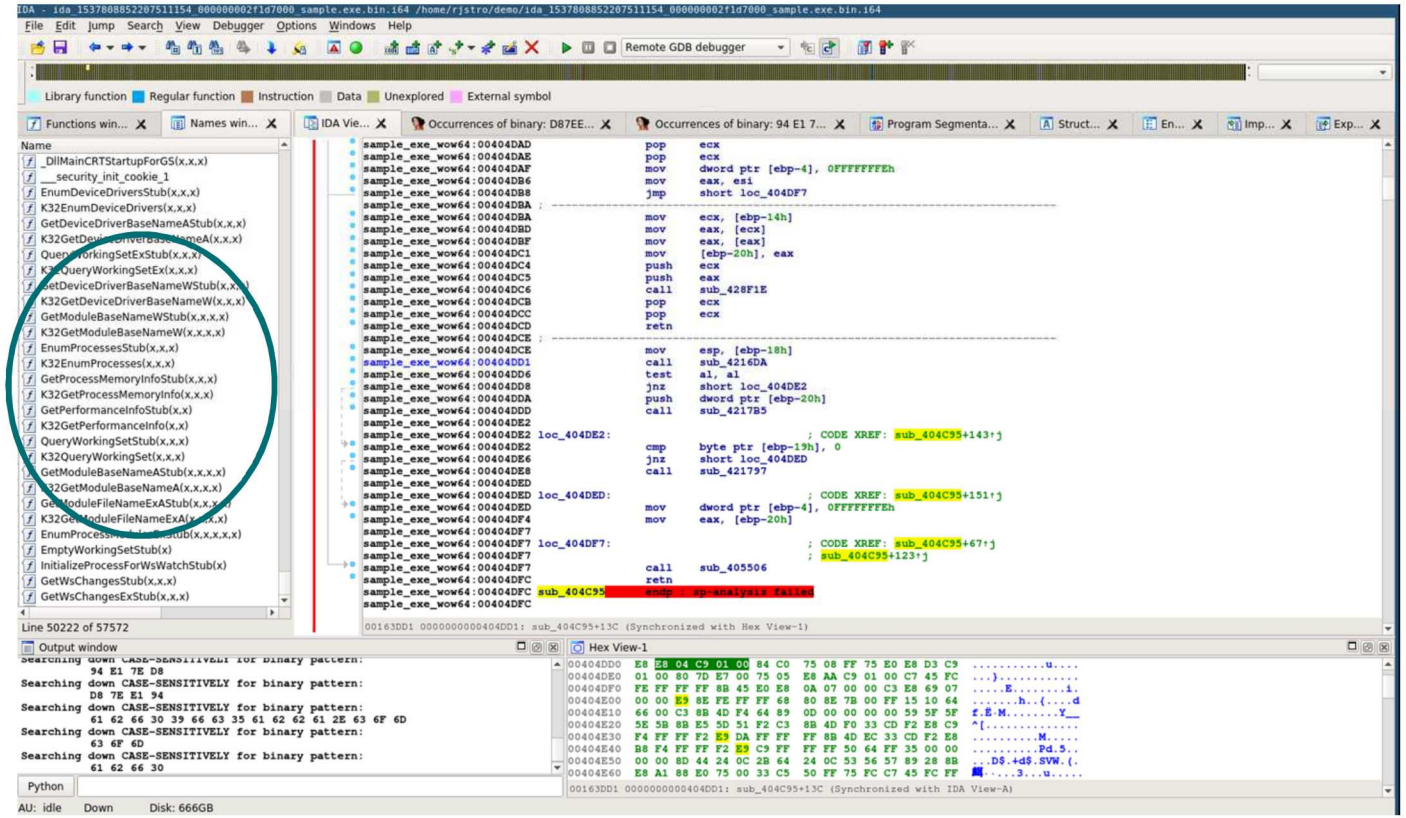

This example shows how the analyst can carve the VI text editor application (process) from the VM

# Reverse Engineering Pipeline



- Track all memory for a process, including all libraries and OS code (not just the binary itself).
- Copy of all these mappings as binary that can be loaded into IDA, with multiple copies so program state can be analyzed at different times during execution.
- An idapython script is used to relocate all those mappings to correct locations, providing the reverse engineer deep insight into a given process.

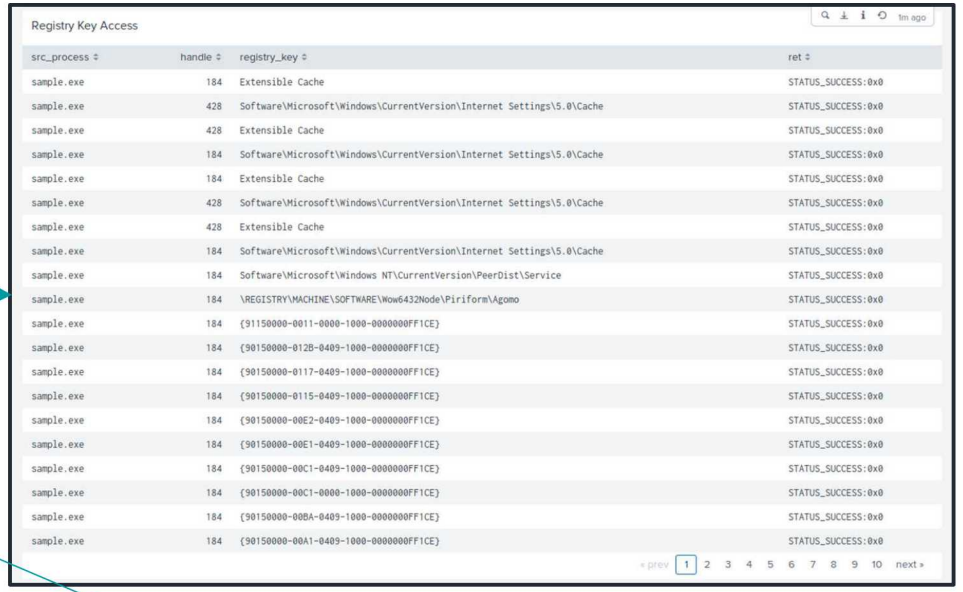
Automatically gather symbols and library information, and correctly relocate (in memory)



# Big Data Extraction

CHIRP
DEMONSTRATION

| Feature | Linux | Windows | Mac |
|---|---|---|---|
| Raw syscalls | Y | Y | Y |
| Decoded syscalls | Y | Y | Y |
| PID/proc name extraction | Y | Y | Y |
| Guest execution | Y | Y | N |
| Kernel carving | Y | Y | N |
| Module carving | Y | Y | N |
| Process carving | Y | Y | N |
| Process tracking (start & exit) | Y | Y | N |
| File extraction | Y | Y | N |
| Biometrics | Y | N | N |
| Socket chardev | N | Y | N |

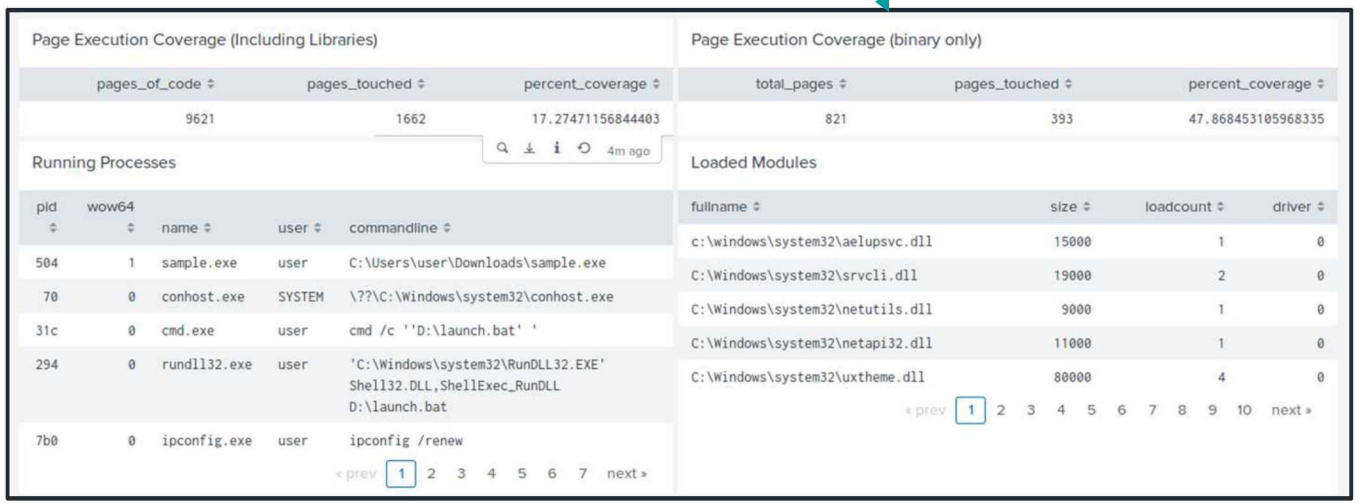


Registry Key Access

1m ago

| src_process | handle | registry_key | ret |
|---|---|---|---|
| sample.exe | 184 | Extensible Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 428 | Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 428 | Extensible Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | Extensible Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 428 | Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 428 | Extensible Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | Software\Microsoft\Windows NT\CurrentVersion\PeerDist\Service | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | \REGISTRY\MACHINE\SOFTWARE\Wow6432Node\Piriform\Agomo | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {91150000-0011-0000-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-012B-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-0117-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-0115-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00E2-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00E1-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00C1-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00C1-0000-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00BA-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |
| sample.exe | 184 | {90150000-00A1-0409-1000-0000000FF1CE} | STATUS_SUCCESS:0x0 |

« prev 1 2 3 4 5 6 7 8 9 10 next »



Page Execution Coverage (Including Libraries)

| pages_of_code | pages_touched | percent_coverage |
|---|---|---|
| 9621 | 1662 | 17.27471156844403 |

Page Execution Coverage (binary only)

| total_pages | pages_touched | percent_coverage |
|---|---|---|
| 821 | 393 | 47.868453105968335 |

Running Processes

4m ago

| pid | wow64 | name | user | commandline |
|---|---|---|---|---|
| 504 | 1 | sample.exe | user | C:\Users\user\Downloads\sample.exe |
| 70 | 0 | conhost.exe | SYSTEM | \??\C:\Windows\system32\conhost.exe |
| 31c | 0 | cmd.exe | user | cmd /c ''D:\launch.bat' ' |
| 294 | 0 | rundll32.exe | user | 'C:\Windows\system32\RunDLL32.EXE' Shell32.DLL,ShellExec_RunDLL D:\launch.bat |
| 7b0 | 0 | ipconfig.exe | user | ipconfig /renew |

« prev 1 2 3 4 5 6 7 next »

Loaded Modules

| fullname | size | loadcount | driver |
|---|---|---|---|
| c:\windows\system32\aelupsvc.dll | 15000 | 1 | 0 |
| C:\Windows\system32\srvcli.dll | 19000 | 2 | 0 |
| C:\Windows\system32\netutils.dll | 9000 | 1 | 0 |
| C:\Windows\system32\netapi32.dll | 11000 | 1 | 0 |
| C:\Windows\system32\uxtheme.dll | 80000 | 4 | 0 |

« prev 1 2 3 4 5 6 7 8 9 10 next »

NtOpenProcess

| src_process | dst_process | access_mask | ret |
|---|---|---|---|
| sample.exe | conhost.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | cmd.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | GoogleUpdate.e | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | explorer.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | taskeng.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | taskhost.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | svchost.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | svchost.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | svchost.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |
| sample.exe | spoolsv.exe | ACCESS_MASK:1000 | STATUS_SUCCESS:0x0 |

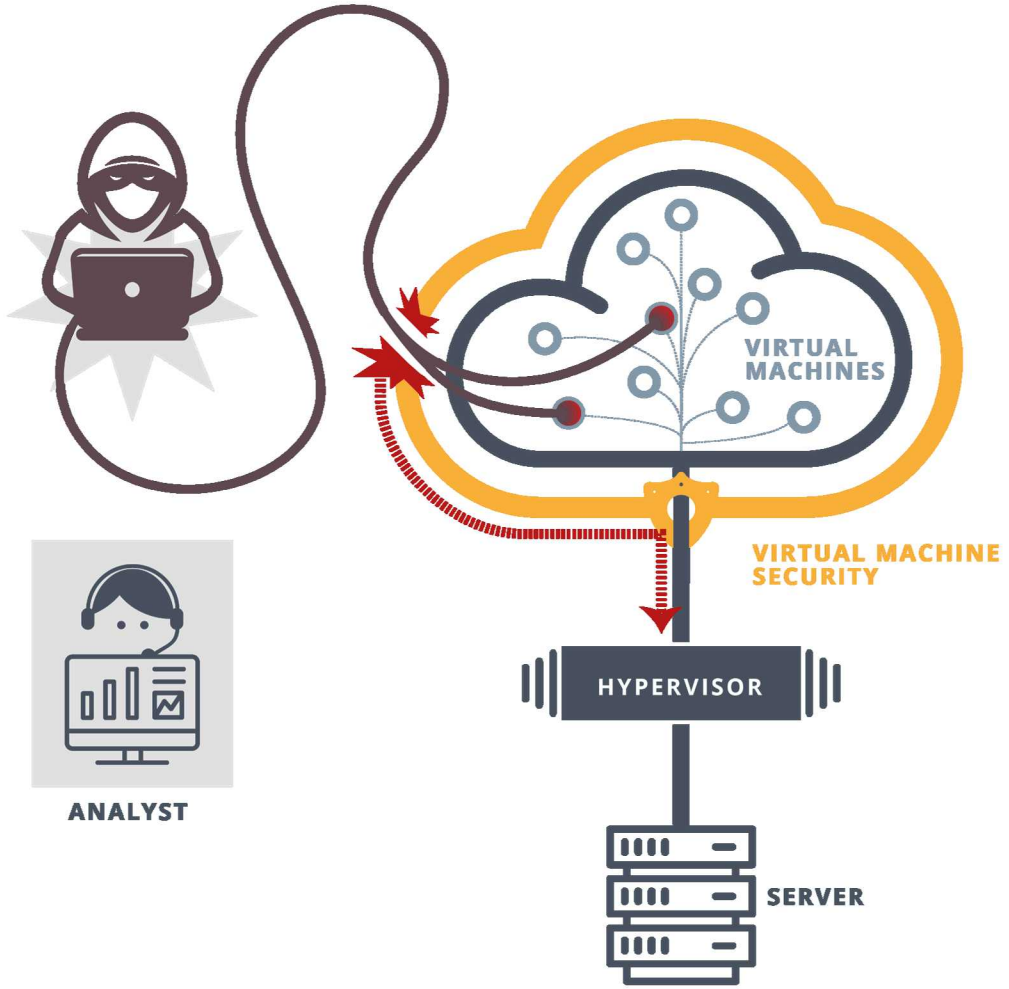
« prev 1 2 3 4 5 6 7 next »


Reserve Slides

# Business Problem

CHIRP

Real-time forensics and incident response for Cloud Service Providers providing deeper analytics of security incidents—leading to better protection and improved visibility.



# Solution Architecture

CHIRP

## Virtual Machine Introspection (VMI)

- Sandia has developed a custom VMI implementation to address the issue of overhead.
- Platform agnostics
- Text and binary data collection
- Allows correlation with other sources

Cloud user accesses the cloud system to instantiate a VM

The CHIRP VMI module is loaded on the Cloud host

The CHIRP intercepts communication between guest and host

Binary data is extracted from the VMI and output to the host

Textual metadata is extracted from the VM and output to the host

Analyst accesses CHIRP to perform duties

# CHIRP Process Flow

## Cloud user accesses the Cloud system to instantiate a VM

- VM specification is sent to a hypervisor on the host.
- Hypervisor allocates resources to the guest VM and starts it.

## The CHIRP VMI module is loaded on the Cloud host

- CHIRP is injected into kernel-space of the host Cloud server
- CHIRP detects hypervisor and places itself in the same administrative domain

## The CHIRP intercepts communication between guest and host

- CHIRP hooks VM-exit handler, assuming hypervisor functions
- Dynamically detects all VM Operating Systems and state on the host

## Binary data is extracted from the VM and output to the host

- Binary data (memory, files, documents, malware, etc.) stored to the host
- Binary data are analyzed

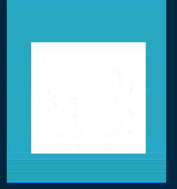
## Textual metadata is extracted from the VM and output to the host

- Textual content (e.g., logs) are ingested by a Security Information and Event Management (SIEM) system
- Textual data is correlated against other data (network, system) in the SIEM

## Analyst accesses CHIRP to perform duties

- Inspects binary data (executes files, reverse engineers malware, etc.)
- Inspects textual data to identify indicators of compromise or attack
- Modifies VM state as needed to aid/prolong an investigation

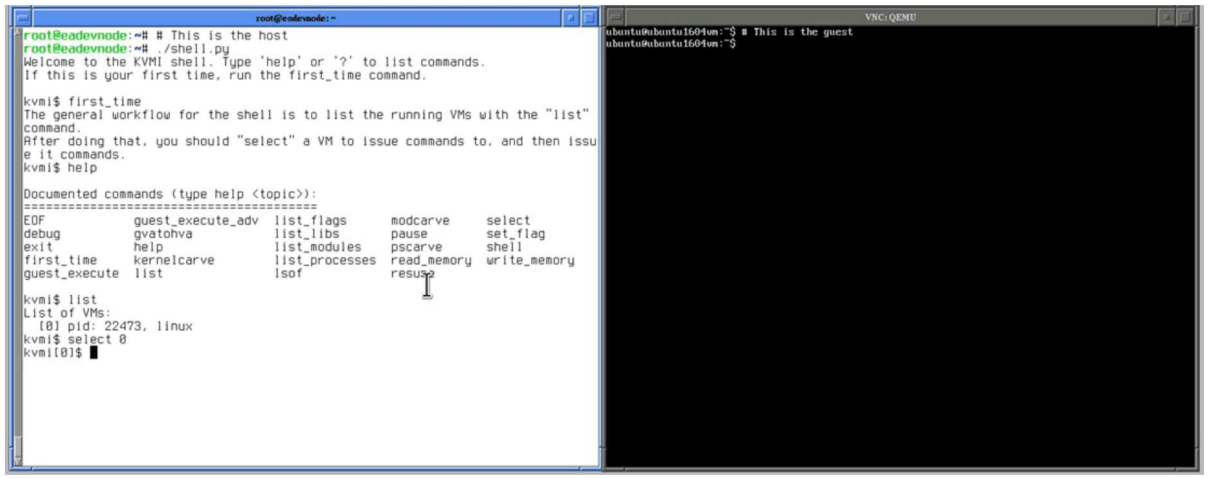
# Demonstration Outline

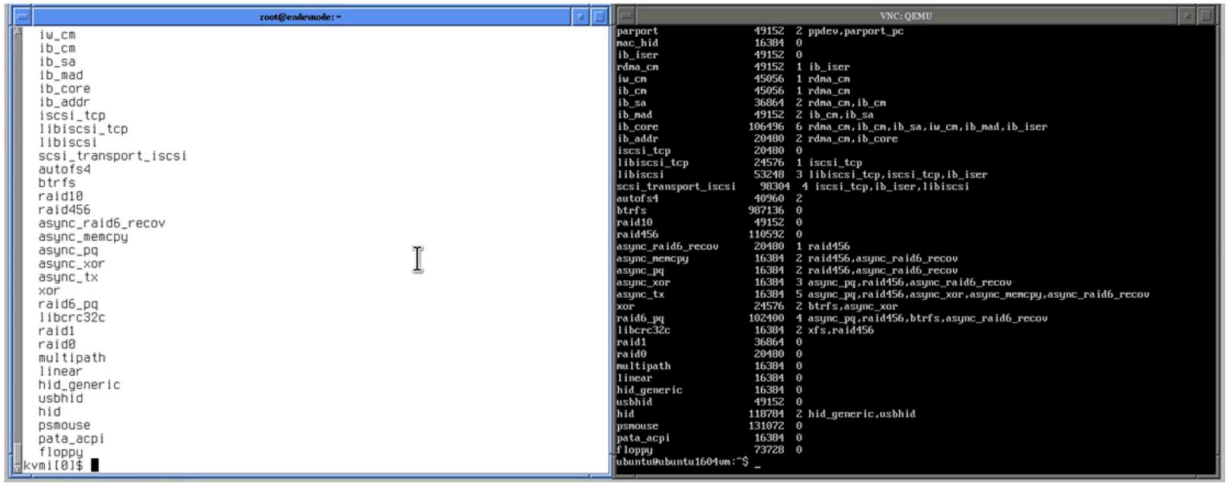


## Vignette Demonstrations

- Module Dumping
- Key Extraction
- Shell
- Carving
- System Call Decoding
- Reverse Engineering Pipeline
- Big Data Extraction

# Interactive Shell



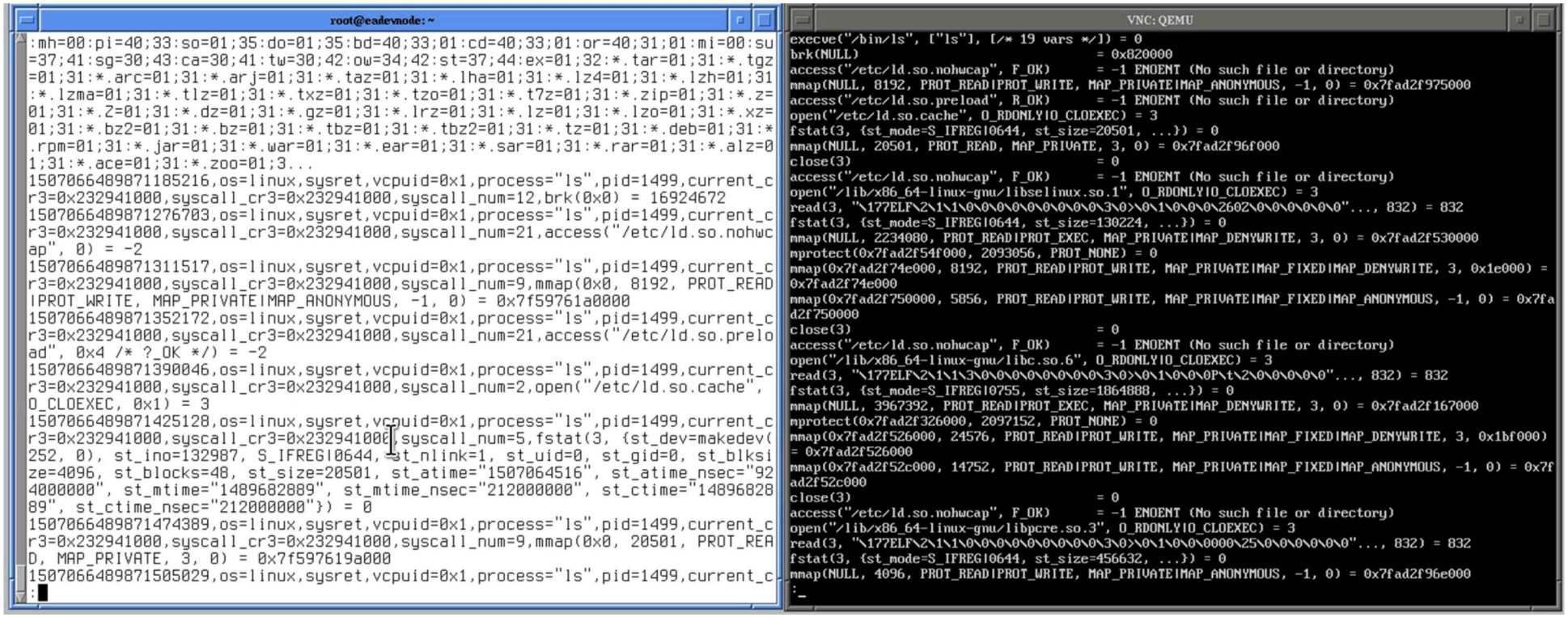




An interactive shell can be used by the analyst to gather information non-obtrusively from the VM in real-time.

- List VMs
- List libraries in use
- List modules executed
- List file descriptors

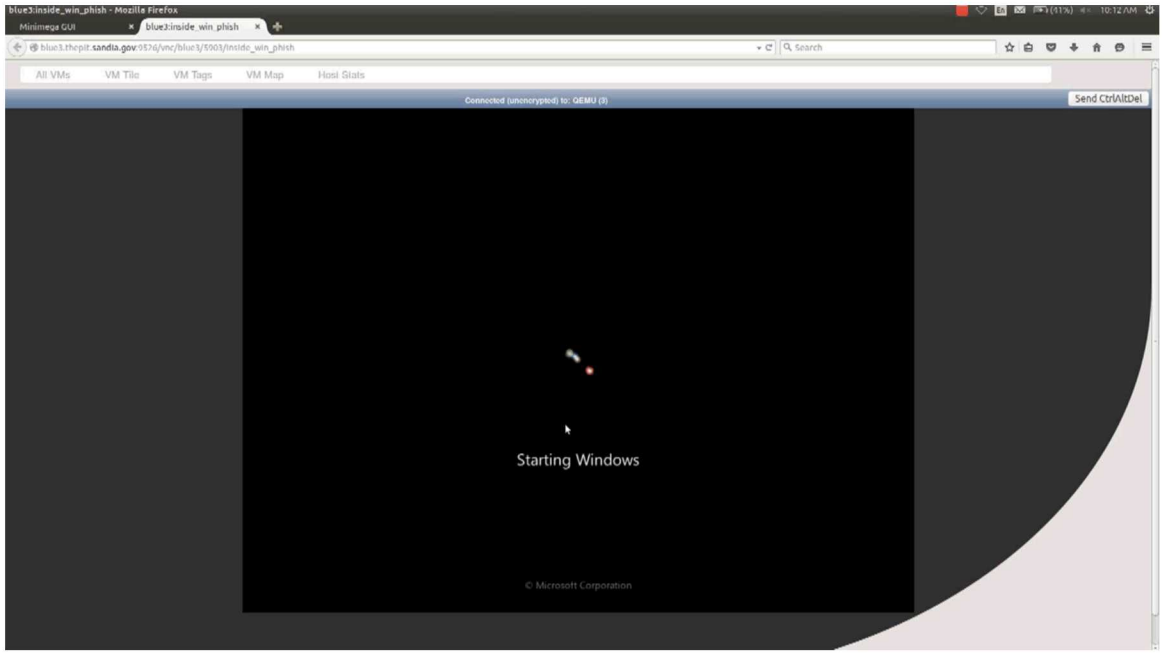
- List running processes
- Write into memory/process
- Start processes on the VM
- Carve objects from memory
  - Files
  - Operating system kernel

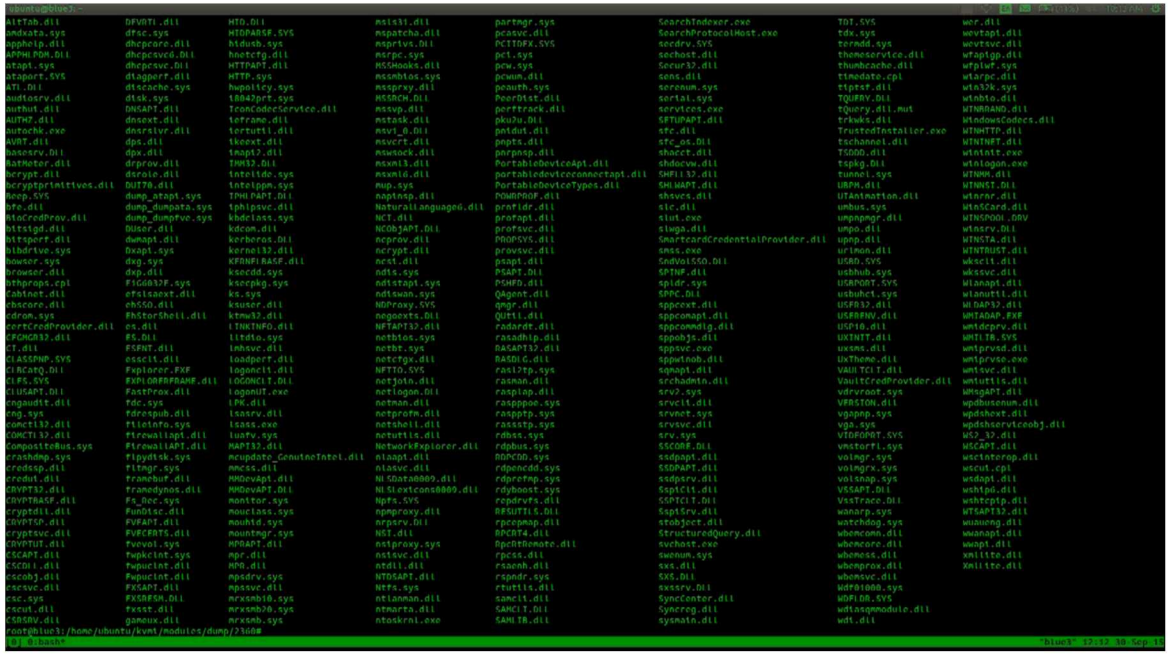
# System Call Decoding




This example shows how an analyst can further watch actions on the VM through system call analysis.
Decoded system calls with parameters are shown in the shell, with an strace of the ‘ls’ program in the right side (VM).

# Module Dumping



Executed binaries are extracted automatically from the guest virtual machine.
Here, on initial boot, 500 modules (dynamic linked libraries (*.dll), executables (.*exe, *.sys)) are extracted.

From boot up onward, CHIRP captures valuable information.