SAND2016-12078C

Exceptional service in the national interest

Sandia National Laboratories

Hypervisor Assisted Forensics and Incident Response in the Cloud

William M.S. Stout

Vincent E. Urias, Caleb Loverro, John W. Young

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Overview

- Introduction
- Background
- Challenges with Cloud Forensics and IR
- Methodology and Approach
- Experiment Results
- Conclusions

INTRODUCTION

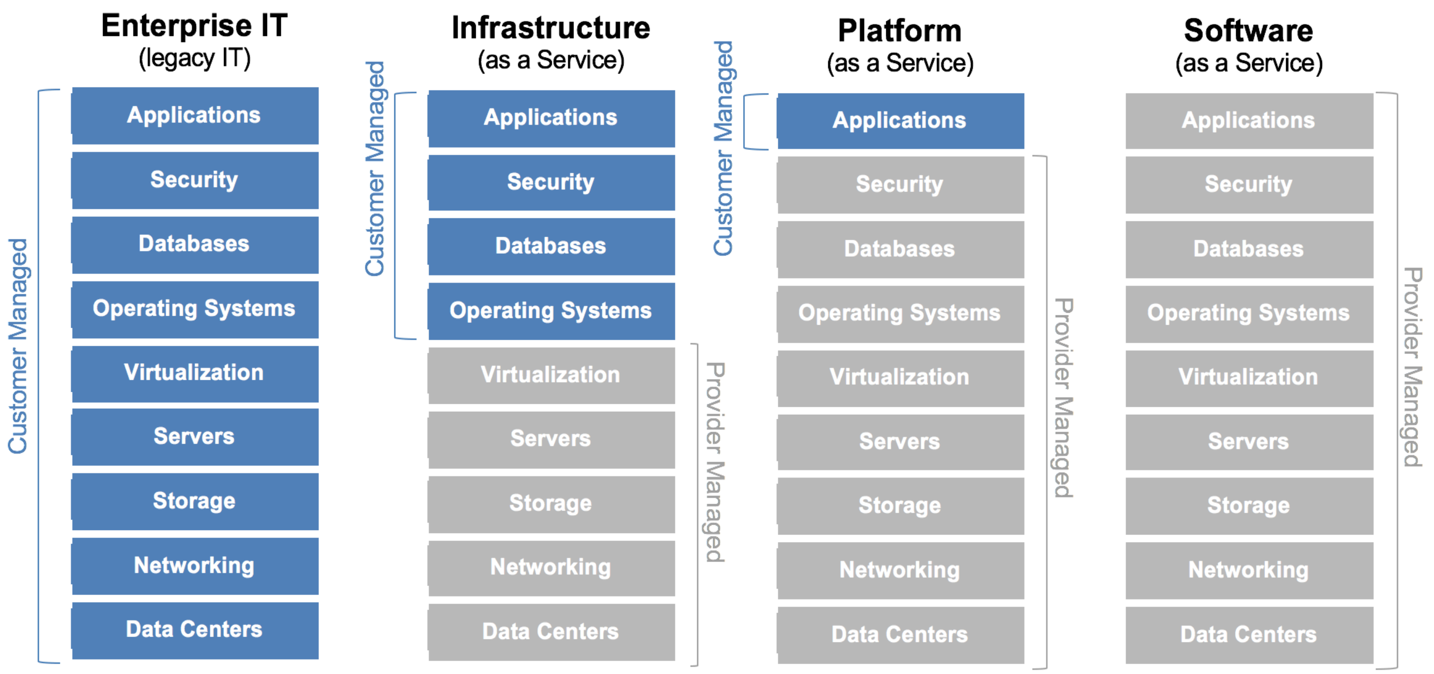
Introduction

- Cloud is becoming pervasive
  - Security/Privacy still not understood
- Traditional Forensics and IR must change to fit the paradigm
  - Difficulties arise with:
    - Ephemerality
    - Attribution
    - Geo-political

BACKGROUND

Background

- The Cloud Computing service models:
  - Software, Platform, Infrastructure



Background

- Digital Forensics foundations:
  1. Identification of an incident from its source(s) and determine its type.
  2. Acquisition of evidence from various sources.
  3. Preservation of the state of evidential data.
  4. Analysis of evidential data, reconstructing fragments and drawing conclusions.
  5. Reporting of results and conclusions about the evidence
- IR Lifecycle foundations:
  - IR: preparation, detection/analysis, containment, eradication and recovery.
  - Incident management includes responding to an incident (cyber), vulnerability and artifact handling, and other related services

FORENSICS/IR CHALLENGES

Cloud Forensic Challenges

- Challenges revolve around:
  - Modified threat surface and response responsibility
  - Multitenancy
  - Virtual/temporary infrastructure
    - VM Migration / VM Location
  - Elasticity (data)
  - Forensically “sound” images

Cloud Forensic Challenges

1. **Architecture: diversity, complexity, provenance, multitenancy, data segregation.**
2. **Data collection: data integrity, data recovery, data location, imaging.**
3. **Analysis: correlation, reconstruction, time sync, logs, metadata, timelines.**
4. **Anti-forensics: obfuscation, data hiding, malware.**
5. Incident first responders: trustworthiness of cloud providers, response time, reconstruction.
6. Role management: data owners, identity management, users, access control.
7. Legal: jurisdiction, laws, SLA, contracts, subpoenas, international cooperation, privacy, ethics.
8. Standards: operating procedures, interoperability, testing, validation.
9. Training: forensic investigators, cloud providers, qualification, certification

METHODOLOGY & APPROACH

Methodology and Approach

- Virtualized in the cloud→ hypervisor becomes an increasingly appropriate place to collect: performance data, system state, system landscape, function calls, transaction traces, and other characteristics.
- Propose a method by which an introspection application may be coupled with a hypervisor to “reach into” the VM with minimal intrusiveness to collect data critical to the reconstruction of events, files, and operations.
- Agentless Virtual Machine Introspection (VMI) coupled with network flow/DPI information → KVMi.

KVMi: Existing APIs

- Why not use existing APIs?
  - APIs do exist for hypervisors like KVM, Xen, VMWare..
    - All have varying levels of usefulness
  - One main reason, levels of indirection
- Understandably, hypervisor writers don't want to give you that much access
  - Why? Issues in the VMI tool can crash the whole system

KVMi: Initial Idea

- Write a Linux kernel module that hooks the VM-exit handler of KVM, to gain complete control over guests before KVM even knows they have exited
  - VMs run until something causes them to VM-exit which passes control to the hypervisor and allows it to view and/or modify their state
- At a low level, the most control you can possibly have is
  - Getting hypervisor-level execution during every VM-exit
  - Reading or writing any VM state you would like
  - Modifying the hypervisor configuration/state (to enable or disable hardware virtualization features, or force future exits with various tricks, for example)
- Hooking the exit handler lets you *be* the hypervisor, which gives you all of this

KVMi: Implementation

- Don't need to patch KVM, no dependencies (other than the Linux kernel).. Why not be loadable/unloadable on demand?
- Desired build process
  - Drop code (c, assembly) on a Linux box, run make, then "insmod kvmi.ko"
  - No other setup (VMs detected at runtime, etc)
  - Allows for use in the largest number of scenarios
    - "Live forensics" on already running VMs
    - Unknown number and/or types of VMs
    - No pre-configuration done

KVMi: Implementation Decisions

- OS agnostic vs. OS specific VMI
  - Chose specific, much more interest in Windows, can do a lot more with OS specific VMI
  - Need some Windows knowledge and/or RE work
- Symbols vs. No-Symbols
  - Chose No-Symbols, Why?
    - Fits better with our implementation model, no symbols needed means no prior configuration steps are necessary, downloading and tracking symbols for Windows versions, etc.
    - Harder but don't want to rely on Microsoft not removing certain symbols in the future (they have done it before)
    - If a VM has patched more recently than your last symbol update you may be in trouble

KVMi: Focus on Speed, Efficiency

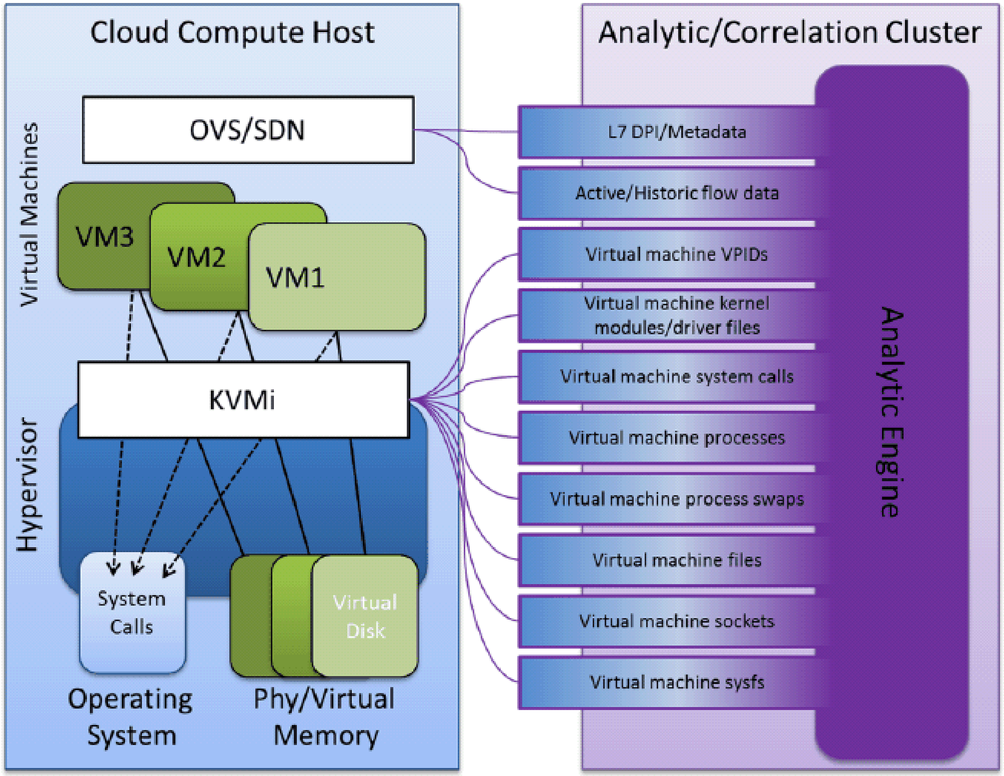
- KVMi provides interfaces for input and output to processes or users running on the host machine
  - Character devices for both input/output
  - SysFs (current work)
- Important details
  - Do as little as possible during VM-exits
  - No ring switching, no blocking
  - Copy or otherwise store the data and offload the I/O work so VMs can resume quickly

KVMi: High-Level Abilities

- Gather process information – user, command line, PE image
- DLLs loaded in the process, (or drivers in the system process)
  - Exported functions from each DLL/driver/EXE (free symbols)
- Hooks (running in-line with modification ability) on arbitrary functions (by address or export name) and syscalls
  - File access reconstruction, socket info, crypto-key dumps, etc.
- Single stepping
  - Several methods, most not possible without the ability to modify hypervisor configuration
- Run arbitrary functions in VMs
  - Save VM state, redirect execution, collect results, restore state
- Age guests
  - Locate system-wide timestamps of interest, change them

Guest and Network Data Fusion

KVMi coupled with Network Level DPI: L7 Classification and metadata extraction for HTTP (request, servers, URIs, MIME types), DNS (hosts, queries, servers), SMTP (mailfrom, header), Kerberos (login, server), LDAP (hostname).



EXPERIMENT RESULTS

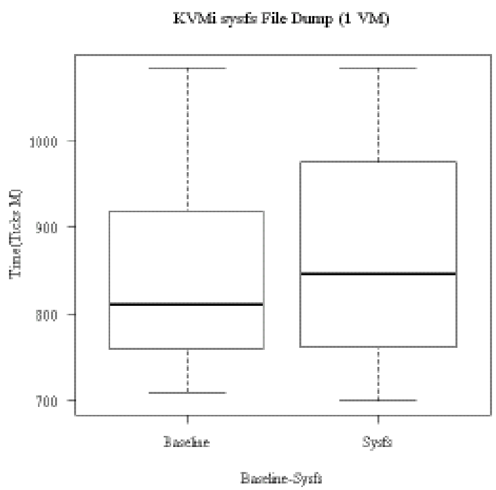
Experiment Set Up

- Experiments based on three use-cases:
  1. VM as a Platform for Attackers
  2. VM as an Exploited Endpoint
  3. Using Cloud as a Relay

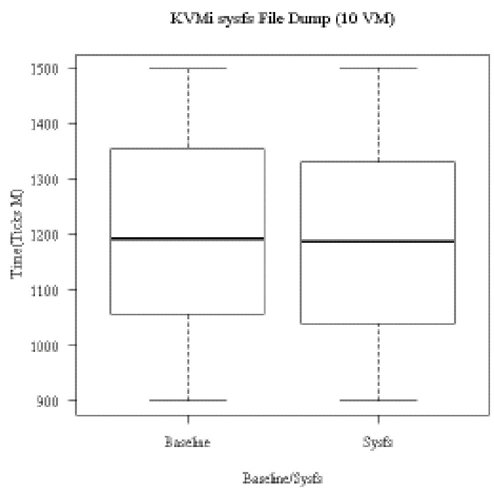
Experiment 1

- VM as a Platform for Attackers
  - Method: Download files to the guest through a web interface, and then saved them to disk. Time ticks were counted during each of the downloads for differences between baseline (that is, without KVMi extracting the file) and with the KVMi sysfs functionality enabled.
  - Results: KVMi kernel module is attached to the KVM hypervisor; its existence is not visible from inside the guest. The only indicator of visibility from inside the guest might be through timing analysis. For the experiment of concurrently downloading a pdf file with 1, 10 and 25 VMs on a host, the time in millions of CPU ticks for the download are show in the box plots below. Each download was run 30 times on each VM instance.

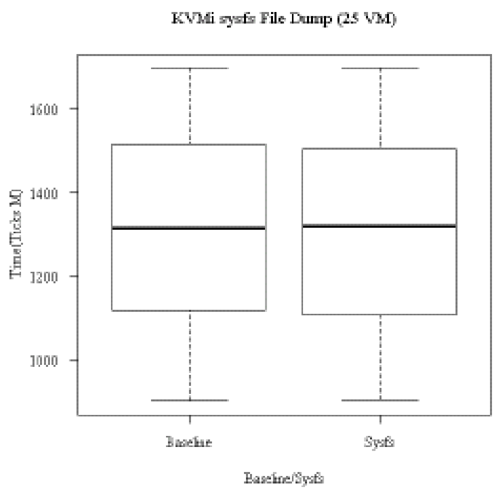
Experiment 1



(a) Case I. 1 VM



(b) Case II. 10 VMs



(c) Case III. 25 VMs

Experiment 2

- VM as an Exploited Endpoint
  - A user on a VM would visit a ``malicious'' website, that would then exploit a browser vulnerability, providing the attacker privileged control of the virtual machine. At this time, the attack would then pivot to other machines in the network, using metasploit to gain passwords.
  - This particular experiment makes use of KVMi to introspect on guest VMs, and network forensic tools (as described above) to correlate guest data to network data. The results of the experiment largely focus on log data to navigate the attack in realtime and identify the actions done on the target VM. By logging the cloud compute host, virtual machine name/ID, and IP addresses, the VM in multitenancy can be quickly identified. KVMi includes data pulled form Windows APIs, with parameters. The collection of guess and network data address the semantic-gap problem of pulling context from the guest to the host.

Experiment 2

- To start the chain of events, an administrator logs into the Vulnerable Workstation and adds an Administrative share using his Domain Administrator credentials. He then visits a phishing website hosted on the attackers machine (“attack.com”).

```
1460340185,dpi_log,path=base.ip.udp.dns,srcip=.66,dstip=.82,udp_port_src=53,udp_port_dst=54799,bytes=396,packets=5,metadata="query:attacker.com,name:attacker.com,addr:.2"
host = blue3 | source = /tmp/flow_logs/dpilog.log | sourcetype = dpi_log
```

- The Attacker compromises the machine using a Silverlight Exploit through a XAP file and runs a bind meterpreter on port 2222

```
1460340189,dpi_log,path=base.ip.tcp.http.silverlight,srcip=.2,dstip=..82,tcp_port_src=8080,tcp_port_dst=49211,bytes=149385,packets=193,metadata="mime-type:application/x-silverlight-2"
host = blue3 | source = /tmp/flow_logs/dpilog.log | sourcetype = dpi_log
```

- The Attacker then starts a new process notepad.exe and migrates to the process so when the user closes iexplore.exe they don't close the meterpreter session.

```
1460340194.32,kvmi_newproc,host_pid=60f0,vpid=11,eproc_hva=7f4306e68b30,peb_hva=7f42ccc48000,peb64=7efdf000,peb32=7efde000,cr3=13fd7000,pid=840,wow64=1,name="notepad.exe"
host = blue3 | source = /tmp/kvmi_logs/kvmi_newproc | sourcetype = kvmi_newproc
```

Experiment 2

- The Attacker then uploads a binary and executes it. The binary is seen here, and also dumped to from the guest for further inspection.

```
1460340204.11,kvmi_apihooks,host_pid=60f0,vpid=11,function="notepad.exe:ntdll.dll:
NtCreateFile(PHANDLE:399e7b8, '\??\C:\Users\win7\Desktop\binary.exe')"
host = blue3 | source = /tmp/kvmi_logs/kvmi_apihooks | sourcetype = kvmi_apihooks
```

- The Attacker then downloads a file located on the compromised machine's desktop to the attacker machine, we see this process started by a walk of the directory tree

```
1460340205.11,kvmi_apihooks,host_pid=60f0,vpid=11,function="explorer.exe:ntdll.dll
:NtCreateFile(PHANDLE:3c4db28, '\??\C:\Users\win7\Desktop')"
host = blue3 | source = /tmp/kvmi_logs/kvmi_apihooks | sourcetype = kvmi_apihooks
```

Experiment 2

- The Attacker runs hashdump and loads mimikatz to collect passwords. As information is transferred back to the attacker machine, high entropy URIs are seen in the DPI log over a meterpreter bound port 2222.

```
1460340316,dpi_log,path=base.ip.tcp.http,srcip=[redacted].82,dstip=[redacted].2,tcp
_port_src=49215,tcp_port_dst=2222,bytes=12739,packets=76,metadata="full-uri:/_1Bi2n
ZvZkH4aflor2L9sQP7EaFbHFOI_4HnTAWJPfKXLtn4c73EVfnRKPRxhcd_2hU2dIsoFt_T7b-tMI-QQ7erS
DA7-OW87dEGBF8HeiqdPARmJ/,server:attacker.com"
host = blue3 | source = /tmp/flow_logs/dpilog.log | sourcetype = dpi_log
```

- Using the credentials and pivot, the attacker uses psexec to login to the domain controller; collects passwords and hashes. Exfiltration communication from the AD to the Attacker over port 3333.

```
1460340265.7,flow_log,event=delete_flow,dpid=9a-7b-ad-72-91-45,vlan=100,srcip=[redacted].66,dstip=[redacted].2,nwproto=6,srcport=3333,dstport=36857,duration=13,pack
et_count=65,byte_count=23010
host = blue3 | source = /tmp/flow_logs/flowlog.log | sourcetype = flow_log
```

Experiment 3

- Using Cloud as a Relay
  - Several connections from the VM are made, combining both normal applications and malicious applications (as denoted by the experimenters).
  - Using the KVMi sockets monitoring feature, the VM making connections and the endpoints (IPs) to which connections are made can be identified. What's novel is the binding of the network connection to the requesting application. The VM (host process id 0xC27) can be seen making connections to IP .33 over port 80, with the process iexplorer.exe (Internet Explorer).

Experiment 3

```
host_pid=c27,vpid=0,pid=1fc,process=svchost.exe,function="BIND:0.0.0.0:0"
host_pid=c27,vpid=0,pid=720,process=iexplore.exe,function="BIND:0.0.0.0:0"
host_pid=c27,vpid=0,pid=720,process=iexplore.exe,function="CONNECT:[redacted].33:80"

host_pid=c27,vpid=0,pid=1fc,process=svchost.exe,function="BIND:0.0.0.0:0"

host_pid=c27,vpid=0,pid=338,process=iexplore.exe,function="BIND:0.0.0.0:0"
host_pid=c27,vpid=0,pid=338,process=iexplore.exe,function="CONNECT:[redacted].33:80"
host_pid=c27,vpid=0,pid=1fc,process=svchost.exe,function="BIND:0.0.0.0:0"
```

CONCLUSIONS

Conclusions

- Several challenges arise when conducting digital forensics and incident response in the Cloud.
- Discussed: challenges, current shortcomings, and proposed a unique approach and tools to meet those challenges.
  - VMI and Network Layer Data Fusion

Future Work

- Extending KVMi for other platforms and various operating systems;
- Furthering KVMi's capability to make on-the-fly modifications to guest execution, such targeted encryption key extraction, or making certain suspicious actions trigger enhanced introspection;
- Further decouple KVMi from KVM, in both its memory accessing ability, and general execution. We are also in discussion with commercial hypervisor companies to extend the KVMi capability to their hypervisors.
- Extending KVMi for general cloud security requirements.

Exceptional service in the national interest

Sandia National Laboratories

Questions/Comments

Hypervisor Assisted Forensics and Incident Response in the Cloud

William M.S. Stout

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP