Trustworthy Services Built on Event Based Probing for Layered Defense

Read Sprabery, Zak Estrada, Zbigniew Kalbarczyk, Ravishankar Iyer, Rakesh Bobba*, Roy Campbell
University of Illinois at Urbana-Champaign, *Oregon State University
IEEE International Conference on Cloud Engineering 2017
Virtualization Provided Security
Virtualization Provided Security
Passive Monitoring

LibVMI
Virtual Machine Introspection
Fast, Portable, Simple

Windows Server 2016

KVM

Xen Server™
Passive Monitoring

LibVMI
Virtual Machine Introspection
Fast, Portable, Simple

KVM

Windows Server 2016

Xen Server™
Passive Monitoring
Passive Monitoring for Intrusion Detection
Active Monitoring (Dynamic Probing)
Active Monitoring (Dynamic Probing)
Active Monitoring (Dynamic Probing)
Dynamic Probing for Intrusion Detection
Dynamic Probing for Intrusion Detection

**Existing Techniques:** only monitored some $\Delta t$ after guest boot sequence.
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
Micro-Services and Virtual Appliances
From Unmanageable Rule Sets

• Complex deployments
• Many components per virtual machine
• Too many valid events occurring to know which are security sensitive
To Manageable Rule Sets

- **Simple** deployments
- **Single** service per machine
- **Normal behavior** is easier to classify
Overview

• Cloud Tuned Trustworthy Services
• Contributions
• Attack Model
• Logged Events
• Policy Recording Mechanism
• Intrusion Detection System for Micro-Services
• Evaluation
• Conclusion and Future Work
Cloud Tuned Trustworthy Services

• What does it mean to be cloud “tuned”?  
• There are two key observations  
  • Service Oriented Architectures  
  • Existing Monitoring Frameworks (with extensions)
Trustworthy Services

• Built on a trustworthy logging solution

• **Trustworthy logging** can be the centerpiece of trusted services in the VMM if:
  • The log is not missing data
  • And is difficult to tamper with

• Limitations of event-based probing
  • How can we place trust in a log generated by *dynamically* inserted probes?
  • How can we guarantee **completeness** – guaranteeing every invocation of an event under inspection must appear in the log?

• Which events should be logged to provide the most coverage and least performance impact?
  • `exec`
  • `open`
Monitoring Architectures for Cloud Computing

• Virtual Machine Introspection Techniques
  • Passive Monitoring – monitoring as a side-effect of the VMM/virtualization technology (early approaches).
  • Event based probing – placing probes at specific instructions in the guest
    • Can trigger events on any location
    • HyperProbes [11], DRAKVUF [12], Xenprobes [22], Spider [23]
    • These techniques use instruction replacement, causing a trap to exit to the VMM

• Running beneath existing workloads
• In containers, this would be the host kernel
Dynamic Probing Mechanisms

Existing Techniques: Probes inserted after guest boot sequence.
Contributions

• Algorithm for inserting probes at time of boot before probed instruction is invoked
• Methods to protect against of attacks that attempt to circumvent the logging system
• An IDS built on top of the logging system, tuned to service oriented Virtual Appliances
• Semi-automated policy-recording mechanism
• Evaluation of the performance in terms of impact to applications and in terms of detection capabilities of the IDS
Integrity of Dynamically Inserted Probe

VM1
8533f92c020f68bf

VM2
fbd16a4a0d84e124

Hypervisor:
17643c29e3c4609818f26becf76d29a3
Log Completeness

• Probes must be placed in their respective locations before instructions at those locations have executed even once.

• System calls being probed are loaded at a predictable location in guest physical address space
  • Determines the page number containing the target instruction
  • Induce EPT violations on the page of interest to determine when to load probe
Extended Page Tables

• 2-Level pages tables from Intel
• Allows for direct translation from guest physical address to host physical address in hardware
• Per page RWX-permissions can be configured within the VMM
• Hardware accelerated virtualization technology
  • AMD’s equivalent is Nested Paging
Induced EPT Sequence for Probe Insertion

Initial write violation during guest boot

Ensure eXecute bit is disabled, enable writes

Execute violation on any instruction on page

Load probes on page, disable writes, enable execute

initial write violation during guest boot:
- RWX 100

Ensure eXecute bit is disabled, enable writes:
- RWX 110
  - Guest loads kernel page

Execute violation on any instruction on page:
- RWX 110

Load probes on page, disable writes, enable execute:
- RWX 101
System Call Stack Trace

Hardware

IDTR/MSRs

Hardware Invocation

SYSTEM_CALL

Software

sys_call_table

SYS_EXECVE
Attack Model

1. IDTR/MSRs
2. Risk of Timing Attack
3. SYSTEM_CALL
4. sys_call_table
5. SYS_EXECVE
6. NEFARIOUS CODE BLOCK

**SYSTEM_CALL**:

```
enable_interrupts()
...
{12 Instructions}
call *sys_call_table(,%rax,8)
```

Event Logged

Interrupts Disabled
Logged Events & Probe Placement

• Increase the cost of compromising log completeness:
  • Use existing trusted boot techniques for both the hypervisor [13] and guests [14], [15].
  • Write protect the probed location.
  • Monitor specific hardware registers.

• Log minimal number of locations to provide meaningful information for higher level services
  • To reduce TCB and performance impacts
Logged Events & Probe Placement

- lidt
- wrmsr

- Writes to probes

KVM

EXEC

OPEN
Overview of Trustworthy Services Architecture

Hypervisor Probes
- SysExecProbe
- SysOpenProbe

Output Log

Event Parsing

Log Buffer

{B1,A1,…,E1} \rightarrow \text{Event } \varepsilon_1

{Bn,An,…,En} \rightarrow \text{Event } \varepsilon_n

Policies

- exec: \{filename: `/sbin/dhclient-script`\}
- open: \{read, filename: `/sbin/resolvconf`\}

Policy Recorder

Event \varepsilon_1 \rightarrow \text{Policy } P_1

Event \varepsilon_2 \rightarrow \text{Policy } P_2

Alert System

Policy Reader

Event Monitor & Policy Alerts

38
Example Policy: `$(which) command`

```json
{
  "policies": [
    {
      "exec": {
        "type": "whitelist",
        "filename": "/usr/bin/which"
      }
    },
    {
      "open": {
        "type": "whitelist",
        "access type": "read",
        "filename": "/lib/x86_64-linux-gnu/libc.so.6"
      }
    },
    {
      "open": {
        "type": "whitelist",
        "access type": "read",
        "filename": "/etc/ld.so.cache"
      }
    },
    {
      "open": {
        "type": "whitelist",
        "access type": "read",
        "filename": "/usr/bin/which"
      }
    }
  ]
}
```
Policy Stacking

- Wordpress policy
- MySQL policy
- DHCP policy

40
Evaluation

• Performance
  • Cloud workloads:
    • OpenSSL – request encryption
    • Apache – web serving
    • Redis – key/value store

• Attacks
  • Can the system detect attacks on scale out cloud applications?
Evaluation – Performance (Apache & OpenSSL)

<table>
<thead>
<tr>
<th></th>
<th>Apache</th>
<th>OpenSSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific Handler</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Generic Handler</td>
<td>60%</td>
<td></td>
</tr>
</tbody>
</table>

Performance Slowdown with Probes
Evaluation – Performance (Redis)

- A – no probes
- B – specific handlers
- C – general handler
Evaluation – Attack Scenario

• Wordpress vulnerability
  • AJAX Load More Plugin
    • Allowed a user to inject arbitrary PHP code
    • User passwords for WordPress accounts are easy to guess
    • Full control over the www-data user (web server user)
  • Linux Mint – ISO’s compromised through WordPress vulnerability
    • Attackers used a similar vulnerability to get shell as the www-data user
    • URL’s for ISO’s were changed to a malicious image
Evaluation – Under Attack

```
```
```
```
Evaluation – Under Attack

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxies</td>
<td>no</td>
<td>A proxy chain of format type</td>
<td></td>
</tr>
<tr>
<td>RHOST</td>
<td>192.168.122.239</td>
<td>yes</td>
<td>The target address</td>
</tr>
<tr>
<td>RPORT</td>
<td>80</td>
<td>yes</td>
<td>The target port</td>
</tr>
<tr>
<td>SSL</td>
<td>false</td>
<td>no</td>
<td>Negotiate SSL/TLS for outgoing data layer</td>
</tr>
<tr>
<td>TARGETURI</td>
<td>/</td>
<td>yes</td>
<td>The base path to the webroot</td>
</tr>
<tr>
<td>VHOST</td>
<td>no</td>
<td>HTTP server virtual host</td>
<td></td>
</tr>
<tr>
<td>WP PASSWORD</td>
<td>netlab1</td>
<td>yes</td>
<td>Valid password for the provider</td>
</tr>
<tr>
<td>WP_USERNAME</td>
<td>ubuntu</td>
<td>yes</td>
<td>A valid username</td>
</tr>
</tbody>
</table>

![Terminal output](image)

```
msf exploit(wp_ajax_load_more_file_upload) > exploit

[*] Started reverse TCP handler on 192.168.122.120:4444
[*] Uploading payload
[*] Calling uploaded file
[*] Sending stage (33864 bytes) to 192.168.122.239
[*] Meterpreter session 1 opened (192.168.122.120:4444 -> 192.168.122.239:45035)

meterpreter > ls
Listing: /var/www/html/wp-content/plugins/ajax-load-more/core/repeater

Mode Size Type Last modified Name
-r--r--r-- 951 fil 2016-05-03 19:32:31 -0500 default.php
meterpreter >
```
Evaluation – Under Attack

```plaintext
msf exploit(wp_ajax_load_more_file_upload) > exploit
[-] Started reverse TCP handler on 192.168.122.120:4444
[*] Uploading payload
[*] Calling uploaded file
[*] Sending stage (33684 bytes) to 192.168.122.239
[*] Meterpreter session 1 opened (192.168.122.120:4444 -> 192.168.122.239)

[!] This exploit may require manual cleanup of 'default.php' on the target.
```

```
Terminal - ubuntu@ubuntu: ~

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxies</td>
<td>no</td>
<td>A proxy chain of format type</td>
<td></td>
</tr>
<tr>
<td>RHOST</td>
<td>192.168.122.239</td>
<td>yes</td>
<td>The target address</td>
</tr>
<tr>
<td>RPORT</td>
<td>80</td>
<td>yes</td>
<td>The target port</td>
</tr>
<tr>
<td>SSL</td>
<td>false</td>
<td>no</td>
<td>Negotiate SSL/TLS for outgoing connections</td>
</tr>
<tr>
<td>TARGETURI</td>
<td>/</td>
<td>yes</td>
<td>The base path to the wp-content</td>
</tr>
<tr>
<td>VHOST</td>
<td></td>
<td>no</td>
<td>HTTP server virtual host</td>
</tr>
<tr>
<td>WP_PASSWORD</td>
<td>netlab1</td>
<td>yes</td>
<td>Valid password for the provided target</td>
</tr>
<tr>
<td>WP_USERNAME</td>
<td>ubuntu</td>
<td>yes</td>
<td>A valid username</td>
</tr>
</tbody>
</table>

Exploit target:

```

```
Evaluation – Under Attack

- Attacker opens a shell
Evaluation – Under Attack

• Attacker opens a shell
Limitations

• Limited detection for intra-process attacks
  • Attack(er) must perform an action outside of defined behavior
    • Access a file
    • Execute a binary

• Attack will go undetected if it remains entirely inside of the compromised binary
  • Inside of the PHP interpreter in the given example.
Conclusion & Future Work

• User configurable tradeoff between resilience and performance
  • Via probe location
• Algorithm for trustworthy insertion of dynamic probes
• Low performance overheads in many cases
• Future work:
  • Runtime integrity checking (detecting intra-process attacks)
  • Improving log resilience at lower performance costs
Questions?

spraber2@Illinois.edu