Data Analytics Layer for High-Interaction Honeypots

Iqra Khan
Agenda

- Motivation
- Virtualization & cloud security
- VMI
- Honeypots
- Malware analysis
- Methodology
- STIX
Motivation

- Cloud computing – today’s most exciting & important technology
- Relocation of systems and services into cloud environments is on the rise
- Users loose direct access / control over their systems
- Memory investigations and forensic processes for attacks/malwares are limited in cloud
Leveraging virtualization in cloud computing

- Deployment of Clouds are all about pooling resources to increase efficiency
- Reduces cost
- Server virtualization, storage virtualization etc.
- High availability
- **Virtualization is an excellent foundation for building clouds**
Cloud Security

- Security of VMs is a hot topic due to their outsourcing in cloud computing
- Large number of VMs
- Network of VMs
- As Scope of virtualization in cloud computing is increased so does the sophistication of attacks on it
Attack Scenario in Cloud
Traditional approaches for VMs security

- In-guest antiviruses or Host based IDSs
  - Provides no isolation
- Network Intrusion detection systems
  - Limited or no context
- Scan VM disk and memory
  - No interposition
Cloud Security

- Move protection out from the VM
  - Hypervisor based isolation
- Full view of the VM state
  - Interpret virtual hardware to see processes, users, connections, files..
- Actively monitor & control
  - Interposition
Virtual Machine Introspection (VMI) is the act of observing the state of VMs from an external entity that can be either another VM or VMM (hypervisor).
VMI (cont.)

- VMI leverages virtualization in three ways:
  - **Isolation**
    - prevents a guest code from reading and writing outside of a VM.
  - **Inspection**
    - VMM can examine the entire state of the guest system (memory, devices, etc.).
  - **Interposition**
    - VMM can interrupt guest code at any time
Why VMI?

- VMI introspection offers greater credibility of malware analysis than traditional antivirus software running on VMs.

- VMI technique inspects and monitors the state of VM in an isolated environment separate from VM.

- This isolation separates the VMI software from tampering by any application or malware inside the monitored VM.
VMI Advantages

- No altering of the target system
- Very hard to detect the monitoring
- Live analysis of memory content
- Detection of advanced memory resident malware
- More reliable data
  - No data corruption through malware
VMI deployment levels

- Process Introspection
- I/O Introspection
- Memory Introspection
Memory Introspection

- Memory introspection deals with live analysis of VM memory.
- Memory contains information like:
  - Running processes
  - Kernel Data Structures
  - Page Tables
  - Registry Entries
Memory Introspection (cont.)

- Majority of malware analysis tools inspect the program behaviour by examining main memory contents of the given program.
- These contents helps in intrusion detection or process analysis of the guest VM.
How can memory of a VM be accessed from outside?

- LibVMI
LibVMI

- LibVMI is an open source library for VMI. It is based on XenAccess library used for VMI.
- XenAccess provides a useful application programming interface (API) for reading to and writing from a virtual machine’s memory.
- Modified to support KVM hypervisor
- That’s why named as LibVMI
- Xen provides built-in functionality to support VMI whereas KVM doesn’t provide any
Features

- Read and write arbitrary data from and to memory
- Access memory using physical addresses, virtual addresses, or kernel symbols
- Parse kernel symbols dynamically from running Windows kernel
- Load Linux kernel symbols from system map file
- Expose useful address translation functions through API functions to resolve kernel symbols to a virtual address or translate a kernel or user virtual address into a physical address
- Pause/unpause the VM through an API function
- Write your introspection code once and have it work across multiple virtualization platforms
Features (cont.)

Using Introspection To View A Kernel Symbol
1. The VMI application requests to view a kernel symbol.
2. LibVMI finds the virtual address for the kernel symbol.
3. Kernel page directory mapped to find correct PT.
4. PT mapped to find correct data page.
5. Data page returned to LibVMI Library.
6. LibVMI returns the data requested by the VMI application (may require mapping multiple pages).
libvmi.conf example

```plaintext
winxpsp2 {
    ostype = "Windows";
    win_tasks    = 0x88;
    win_pdbase   = 0x18;
    win_pid      = 0x84;
    win_kdvcb    = 0x80544ce0;
}

win7sp1x64 {
    ostype = "Windows";
    win_tasks    = 0x188;
    win_pdbase   = 0x28;
    win_pid      = 0x180;
    win_kdvcb    = 0xffffffff800027f10a0;
}
```
Features (cont.)

Runtime analysis capabilities augment Volatility’s rich memory analysis.

Volatility (memory analysis framework)

pyvmi address space plugin

pyvmi (Python language wrapper for LibVMI)

LibVMI (C language API)

patch KVM Xen Other VMM Memory Snapshot
Virtual Honeypots

- Virtual honeypots exist as a virtual resource instead of dedicated physical system with the purpose of attracting and logging cyber-attacks in real time
  - Often emulate or are exposed to live security vulnerabilities in order to capture and monitor both malware and cyber-attackers
  - Can be used to monitor various protocols, applications, or operating system attacks
  - Malware execution behaviors can be logged and can be used in malware research
Virtual honeypots (cont.)

- Detection & Response not prevention
  - Collects evidence information and detects attack patterns
  - Defenders can respond to these evidences by building better defenses and countermeasures against future security threats
Honeypots Categorization

- Interaction
  - Low-Interaction Honeypots
  - High-Interaction Honeypots

- Deployment
  - Research Honeypots
  - Production Honeypots
Related work

- CloudVMI - VMI offered as a service in public clouds
- VMI-Honeymon - high-interaction honeypot monitor which uses virtual machine memory introspection on Xen
- Livewire
- Collapsar
- VMScope
Methodology

- VMI capability is combined with malware analysis and virtual honeypots to achieve the objective

- Extracted IOCs are then converted in STIX programming language
Architecture Design

- KVM hypervisor
- Server Virtualization
- Host-only networking
- LibVMI and Volatility
- Virtual Honeypots
Architecture Design (cont.)
LibVMI KVM support

- For KVM there are two approaches to access VM memory

1. GDB (GNU Debugger)
2. A patch created for KVM that enabled memory access through a UNIX domain socket
KVM (kernel-based VM) hypervisor

- Hypervisor of choice for open source clouds
- Low cost
- High scalability
- Ease of deployment
- Openstack
- IBM SmartCloud Enterprise
- Intel IT
Deployed Honeypots

- Kfsensor
- Valhala
- HoneyBOT
Individual events

Statistical analysis reports
Attacked services
alerts

Traffic initiated by particular attacker machine
Volatility

- Volatility is an open source memory forensic tool helping incident response and memory forensics.
Used Volatility plug-ins for memory introspection

- pslist
- pstree
- connections
- connscan
- malfind
- handles
- dlllist
- svscan
- getsids
- strings etc.
IOCs to look for?

- Suspicious processes are spawned out of right path?
- Suspicious process is running under its legitimate parent process, or some other process spawned it?
- At what time process started and exited?
- What privileges process under consideration has? Whether this process should have these privileges?
Another important point is process name. See is it matching to some legitimate Windows process and malware attacker change it a bit to match a legitimate Windows process to avoid detection.

See for the associated process objects like threads, mutexes, DLL, process to file mappings, memory Sections, associated sockets and ports open by that process.

Connections initiated by the process and the connection initiated it
Performed Analysis stages

Malware analysis stages

- Manual code reversing
- Static properties Analysis
- Fully Automated Analysis
- Interactive Properties Analysis
Studied attacks

- Reflective Injection
- Trojans
- Attacks on specific vulnerable ports used by most attackers
Flow chart

- Honeypot alert of remote IP connection
- Introspected with volatility and LibVMI
- Process detected (skccca.exe)
- Infected the system and start listening to remote IP
- Connections & connscan
  - Handles (mutant)
    - privs
    - dlllist
    - apihooks
    - malfind
      - vaddump
      - strings
      - getsids
      - svscan
Memory region starting at address 0x00400000 details contains suspicious tag VadS. Also it contains mapped file skcca.exe.

```
42
```

```
812 svchost.exe 7 SeTcbPrivilege
812 svchost.exe 15 SeCreatePagefilePrivilege
812 svchost.exe 4 SeLockMemoryPrivilege
812 svchost.exe 14 SeIncreaseBasePriorityPrivilege
812 svchost.exe 16 SeCreatePermanentPrivilege
812 svchost.exe 20 SeDebugPrivilege
812 svchost.exe 21 SeAuditPrivilege
812 svchost.exe 23 SeChangeNotifyPrivilege
812 svchost.exe 10 SeLoadDriverPrivilege
812 svchost.exe 13 SeProfileSingleProcessPrivilege
812 svchost.exe 12 SeSystemtimePrivilege
812 svchost.exe 25 SeUndoCKPrivilege
812 svchost.exe 29 SeImpersonatePrivilege
812 svchost.exe 30 SeCreateGlobalPrivilege
```

```
42
```

```
812 svchost.exe 7 SeTcbPrivilege
812 svchost.exe 15 SeCreatePagefilePrivilege
812 svchost.exe 4 SeLockMemoryPrivilege
812 svchost.exe 14 SeIncreaseBasePriorityPrivilege
812 svchost.exe 16 SeCreate PermanentPrivilege
812 svchost.exe 20 SeDebugPrivilege
812 svchost.exe 21 SeAuditPrivilege
812 svchost.exe 23 SeChangeNotifyPrivilege
812 svchost.exe 10 SeLoadDriverPrivilege
812 svchost.exe 13 SeProfileSingleProcessPrivilege
812 svchost.exe 12 SeSystemtimePrivilege
812 svchost.exe 25 SeUndoCKPrivilege
812 svchost.exe 29 SeImpersonatePrivilege
812 svchost.exe 30 SeCreateGlobalPrivilege
```
<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Pid</th>
<th>Handle</th>
<th>Access Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0639a2a8</td>
<td>1276</td>
<td>0x84e0f</td>
<td>Mutant</td>
<td>aspnet_statesusq</td>
</tr>
<tr>
<td>0x064e1c20</td>
<td>1276</td>
<td>0x8e0x8e0</td>
<td>Mutant</td>
<td>ShimCacheMutex</td>
</tr>
</tbody>
</table>

**Registry**

- **Key name**: aspnet_statesusq (S)
- **Subkeys**: (S) Security (V) Enum
- **Values**:
  - REG_DWORD: Type: (S) 16
  - REG_DWORD: Start: (S) 2
  - REG_DWORD: ErrorControl: (S) 0
  - REG_SZ: DisplayName: (S) ASP.NET State Services\vb Transaction Coordinator Service
  - REG_SZ: ObjectName: (S) LocalSystem
  - REG_SZ: Description: (S) Provides support for out-of-process\mx Transaction Coordinator Service.
Execution Time (s) of Volatility plug-ins Executed against live VMs

Plug-ins

- plist
- malfind
- dillist
- aphooks
- hivelist
- strings
- vaddrmap
- svcscan
- handles
- pstree
- privs
- conlections

Time (s)

- 74.528
- 84.723
- 79.377
- 304.895
- 451.381
- 94.750
- 98.803
- 90.366
- 83.447
- 65.733
- 83.781
Structured Threat Information Expression (STIX)

- A programming language for conveying data about cybersecurity threats in a common language that can be easily understood by humans and security technologies.

- A variety of high-level cyber security use cases rely on such information including:
  - Analyzing cyber threats
  - Specifying indicator patterns for cyber threat
  - Managing cyber threat response activities
  - Sharing cyber threat information

- Consistency, efficiency, interoperability, and overall situational awareness.

- CybOX: Cyber Observable eXpression
STIX Architecture

- **Cyber Observables** - what activities we are observing on our networks or systems
- **Indicators** - What threats should I look for on my networks and systems and why?
- **Incidents** - Where has this threat been seen?
- **Adversary Tactics, Techniques, and Procedures** (including attack patterns, malware, exploits, kill chains, tools, infrastructure, victim targeting, etc.) - What does it do?
STIX Architecture (cont.)

- **Exploit Targets** (e.g., vulnerabilities, weaknesses or configurations) - *What weaknesses does this threat exploit?*

- **Courses of Action** (e.g., incident response or vulnerability/weakness remedies or mitigations) - *What can we do about it?*

- **Cyber Attack Campaigns** - *Why does it do this?*

- **Cyber Threat Actors** - *Who is responsible for this threat?*
Converted STIX IOCs
Future Work

- Extract low-level information programmatically through LibVMI
- Using a network of honeypots
References

- https://www.usenix.org/conference/cset12/workshopprogram/presentation/Lengyel
Thanks !