SLIME: AUTOMATED ANTI-SANDBOXING DISARMAMENT SYSTEM
Yosuke Chubachi and Kenji Aiko
FFRI, Inc.
About us

Yosuke Chubachi
He is a security engineer at FFRI, Inc. since last spring. He studied at the graduate school of information system engineering, University of Tsukuba. He is a Security Camp lecturer and a member of executive committee of SECCON since 2012.

Kenji Aiko
He is a programmer at FFRI, Inc., and is a one of the developers of "FFR yarai" which is a targeted attack protection software. He is a Security Camp lecturer and a member of executive committee of SECCON since 2012.

March 27, 2015
Contents

- Background and Motivation
- State of the Art of Anti-sandboxing
- SLIME Design and Implementation
- Disarming Real Malware
- Experiments
- Conclusion
Background and Motivation
Background

- Malware explosion — 120,000,000 over in 2014
- Antivirus is dead...?

AV Test: Statistics – New Malware - (Nov. 05 2014 viewed)

March 27, 2015
We need dynamic and automated malware analysis

• “Scalability” is most important factor in information explosion era
  — Cloud
  — Bigdata
  — IoT

• Malware analysis also needs “scalable” methodology
“Use the sandbox, Luke”

- Security engineer and researcher use sandbox environment for malware analyzing
- Automated dynamic analysis technology also based on VM/application sandbox
Malware strike back

• Sophisticated malware arms many anti-analyze techniques
  • Naturally using targeted attacks, cyber espionage, banking malware

• Researchers called those malware “evasive malware”
Related work

- BareCloud [Dhilung K et al., USENIX SEC’14]
  — “5,835 evasive malware out of 110,005 recent samples”

- Prevalent Characteristics in Modern Malware
  [Gabriel et al., BH USA ‘14]
  — “80% malware detect vmware using backdoor port”

What do you think?

March 27, 2015
Motivation

• Investigating into a condition used by sandbox evasion automatically for select right sandbox using investigated conditions
Challenges

• Incorporable and standalone
  — Because we are developing anti virus application
State of the Art of Anti-sandboxing
State-of-the-art anti-sandboxing

- CyberGate (RAT)
- Chthonic (Online Banking Malware)
CyberGate

- Popular RAT tools
- CyberGate can generates remote access server for targeting host
- Anti-sandbox option enabled
CyberGate

Create server

Cancel the execution of the server in the following cases

- Anti Sandboxie
- Anti Virtual PC
- Anti VMWare
- Anti VirtualBox
- Anti ThreatExpert
- Anti Anubis
- Anti CWSandbox
- Anti JoeBox
- Anti Norman Sandbox
- Anti Softice

- Anti Debugger
- Other

March 27, 2015
Anti-sandboxing are generated by CyberGate.
Chthonic

- Banking trojan subspecies of ZeuS Family
- Chthonic downloader injects malicious code into msiexec.exe
- Also downloader changes its behavior if runs on sandbox or virtual machines

See also:
Chthonic

Calling many vm/sandbox detection

March 27, 2015
Chthonic anti-sandboxing

- Finding vm/sandbox artifacts
  - `.\HGFS`, `.\VBoxGuest`, `.\vmci` and `.\Wine`
  - sbie.dll
- Similar “Citadel”
  - Citadel also finding vm/sandbox artifacts
Type of anti-sandbox

- Anti-sandbox maneuver
  - Environment awareness
    - Using result of vm/sandbox detection
    - Host fingerprinting
  - (Stalling code)
  - (User/Network interaction checks)
Environment awareness

- Checking host environments
- If malware runs *decoy routine or exit itself* then it detects analyzer’s sign
  — Malicious behavior never executed

Initialization (unpack) → Sandbox (incl. VM) Detection → Malicious routine

If running on an analyzing environment

March 27, 2015
Sandbox (debug/sandbox/vm) detection

- Artifact fingerprinting
- Execution environment fingerprinting
- (Execution timing detection)
Sandbox (debug/sandbox/vm) detection

Host

VM related Artifacts

Artifact Fingerprinting

Sandbox specific Artifacts

Environment aware Malware

Execution Environment Fingerprinting

Execution Timing Detection

VMM?
Artifact fingerprinting

- Sandbox/VM environment specific files
- Sandbox/VM environment specific registry keys
- Sandbox/VM environment specific devices and its attributes
  - ex). QEMU HDD vendor name
- Sandbox/VM Specific I/O port
  - VMWare backdoor port is most famous artifact in malware
- Sandbox/VM related processes
  - Like vmware, virtualbox etc.
Execution environment fingerprinting

- Using virtual machine implementation specific platform value and reaction
  - CPUID instruction result
  - Redpill
    - Using LDT/GDT and IDT incongruousness
- Interesting research here: Cardinal Pill Testing
Execution timing detection

- Using clock count differential
  - Traditional anti-debug technique

Comparing TSC differentials

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>400022A2</td>
<td>60</td>
<td>PUSHAD</td>
</tr>
<tr>
<td>400022A3</td>
<td>0F31</td>
<td>RDTSC</td>
</tr>
<tr>
<td>400022A5</td>
<td>31C9</td>
<td>XOR ECX,ECX</td>
</tr>
<tr>
<td>400022A7</td>
<td>01C1</td>
<td>ADD ECX,EAX</td>
</tr>
<tr>
<td>400022A9</td>
<td>0F31</td>
<td>RDTSC</td>
</tr>
<tr>
<td>400022AB</td>
<td>29C8</td>
<td>SUB EAX,ECX</td>
</tr>
<tr>
<td>400022AD</td>
<td>3D FF0F0000</td>
<td>CMP EAX,0FFF</td>
</tr>
<tr>
<td>400022B2</td>
<td>61</td>
<td>POPAD</td>
</tr>
<tr>
<td>400022B3</td>
<td>0F83 11010000</td>
<td>JNB 400023CA</td>
</tr>
</tbody>
</table>

March 27, 2015
SLIME key technologies

- Malware palpation
- Code execution integrity (CEI)
- Retroactive condition analysis
Concept: malware palpation

1. Our sandbox runs malware again and again
   — Changing “virtual” artifacts exposure each execution for execution branch detection

2. Retroactive condition analysis
   — Specifying “branch condition” on unnatural process termination
Malware palpation

- SLIME Sandbox fakes different sandbox-related artifacts each malware execution
  — Detecting execution difference using code execution integrity (CEI)

SLIME (first execution)
- Malware
  - CreateFile(...)
  - Not found
  - API emulator

SLIME (vmware faking)
- Malware
  - CreateFile(...)
  - vmmouse
  - .sys
  - faking
  - API emulator

March 27, 2015
Code Execution Integrity (CEI)

- CEI shows uniqueness of instruction execution history
  — Inspired by TPM trust chaining
- “measurement” per instruction

\[ \text{Digest}[i] = \text{SHA1}( \text{fetched CPU instruction} + \text{Digest}[i-1]) \]

```
mov $0x616b6157, %eax
push %ebx
push %eax
mov $4, %edx
mov $1, %ebx
```

```
d[0] = \text{SHA1}(0xb857616b61)
d[1] = \text{SHA1}(d[0] + 0x53)
d[2] = \text{SHA1}(d[1] + 0x50)
...```

March 27, 2015
Execution branch detection

- Using execution step count and code execution integrity (CEI) value

```
CEI[0]_1
CEI[1]_1
CEI[2]_1
CEI[3]_1
CEI[4]_1

CEI[0]_2
CEI[1]_2
CEI[2]_2
CEI[3]_2
CEI[4]_2
CEI[5]_2

March 27, 2015

in     eax, dx
cmp    ebx, 0x564D5868h
jne    NOTVMX
jmp    ISVMX

NOTVMX: mov rc, 0
jmp    done

ISVMX:   mov rc, eax
jmp    done
```
Retroactive condition analysis

- Sandbox retroactive from termination to terminated reason API and arguments when suspicious termination
  — Only a few steps executions
  — To terminate before network activities

```c
sub esp, 1024
mov ebx, esp
push 400h
push ebx
push 0h

call GetModuleFileNameA
lea eax, MyPath
push eax
push ebx

call lstrcmpA
test eax, eax
push 0h
lea eax, MsgCaption
push eax

jz _ok
lea eax, NGMsgText
push eax
push 0h
call MessageBoxA

invoke ExitProcess, NULL

_ok:
lea eax, OKMsgText
```
Implementation

- We have already CPU Emulator-based sandbox for win32 execution (in-house use)
  —Like IDA Bochs PE operation mode[11]
Execution logging framework

• SLIME logs instruction per execution
  — Tracing specific API call and its arguments for Retroactive condition analysis
    • lstrcmpi, strcmp, GetModuleFileName, ...

• Code execution integrity calculation per execution
  — For execution branch detection
Camouflaging VM/sandbox related artifact existence

- VMWare
  - Camouflaging backdoor port, some registry entry and files

- VirtualBox
  - Some registry entry and files

- QEMU
  - Some registry entry and files

- Sandbox
  - Anubis
  - Sandboxie
  - ThreatExpert
Disarming Real Malware
Disarming demo
Anti-VMWare

SHA256: C1A7E51E5E2F94193D6E17937B28155D0F121207
Detect sandbox evasion

SHA256: 39517A057CC4A1AE34E786873C8010291A33BAB7

March 27, 2015
Experiments
Dataset

- Trying to disarm 89,119 malware
  - Collected in one year (2014/01/01-2014/12/31)
  - Original data amounts: 5,244,297
  - Random sampling
  - Filtered in PE(32bit) and loadable our sandbox
Results

<table>
<thead>
<tr>
<th>Anti-Sandbox Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detecting VMWare</td>
<td>63</td>
</tr>
<tr>
<td>Detecting VirtualBox</td>
<td>70</td>
</tr>
<tr>
<td>Detecting QEMU</td>
<td>84</td>
</tr>
<tr>
<td>Detecting Sandbox (sbie.dll and dbghelp.dll)</td>
<td>11,102</td>
</tr>
<tr>
<td>Evasive Malware</td>
<td>36</td>
</tr>
</tbody>
</table>

* Throughput: 6 malware per minutes
Are Anti-VM Too Few?

- We guess that more Anti-VM malware exists in this dataset CPU
  - Because our CPU emulator coverage is not enough to run malware
- Original sandbox was developed for unpacking
### Offtopic: Artifact finding by Yara

#### Using customized Anti-VM rules @ YaraRules

<table>
<thead>
<tr>
<th>Anti-Sandbox Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Found VMWare Signature</td>
<td>11,029</td>
</tr>
<tr>
<td>Found VirtualBox Signature</td>
<td>530</td>
</tr>
<tr>
<td>Found QEMU Signature</td>
<td>247</td>
</tr>
<tr>
<td>Sandbox detection</td>
<td>235</td>
</tr>
</tbody>
</table>

#### Using SLIME implemented artifact only

<table>
<thead>
<tr>
<th>Anti-Sandbox Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Found VMWare Signature</td>
<td>10,985</td>
</tr>
<tr>
<td>Found VirtualBox Signature</td>
<td>142</td>
</tr>
<tr>
<td>Found QEMU Signature</td>
<td>127</td>
</tr>
<tr>
<td>Sandbox detection</td>
<td>221</td>
</tr>
</tbody>
</table>
Can Virtual Machine Protects You from Malware?

- No
  - The proportion of Anti-VM armed malware is low in the wild
  - Anti-VM activity is one of method of black list avoiding
Can I Ignore Anti-Sandboxing?

• No!
  — Many anti-sandboxing founds before malicious behavior such as suspicious download or code injection
  — If you do not pay attention, you will be miss significant threat
Conclusion

• SLIME can investigate into a condition used by sandbox evasion automatically

• The proportion of anti-VM armed malware is low in the wild

• However, there is no doubt that sophisticated malware often uses anti-sandboxing
Bibliography

• IDA Boch PE operation mode http://www.hex-rays.com/products/idav/support/idadoc/1332.shtml
Fin.