Detecting the One Percent: Advanced Targeted Malware Detection

Tomer Teller
Check Point Software Technologies
Antivirus 20^{th}+ Anniversary
The Halting Problem

\[ h(i, x) = \begin{cases} 
1 & \text{if program } i \text{ halts on input } x, \\
0 & \text{otherwise.} 
\end{cases} \]
The Malware Problem

\[ m(i) = \begin{cases} 
1 & \text{if program } i \text{ is malicious} \\
0 & \text{otherwise.} 
\end{cases} \]
The Constraints

**TIME**

Cannot analyze program forever
- Slow down loops
- Sleep
- Time-consuming operations (Encryption/Packing)

**SPACE**

Cannot maintain unlimited states
- “Run out the clock”
  - OpenProcess → VirtualAllocEx →
  - WriteProcessMemory → LOOP → .. →
  - CreateRemoteThread
Exploiting the Constraints

Advanced malware exploits these constraints

Thwart static analysis --> SPACE

Thwart dynamic analysis --> TIME + SPACE
More Depressing News

- Elevation of privilege to kernel mode
  - Bypassing security products

- Stolen certificate authorities
  - Breaking the trust

- Automatic static analysis is hard!
  - Packing / obfuscation / encryption

- Manual static analysis
  - Unpacking / time consuming / not scalable

- Dynamic analysis
  - The malware problem!
Relax!
### Current Detection Methods (partial)

#### Pattern Based Analysis
- MD5 / SHA1 / SHA256
- Fuzzy hashing
- Pattern-based
- PCRE/ Regex
- Proprietary language
- Malware classifiers (J48, J48 Graft, PART)

#### Static Analysis
- Anti-VM
- Anti-debugging
- Anti-disassembly
- Obfuscation
- Reverse engineering

**Rodrigo Rubira Branco  BH12**

#### Dynamic Analysis
- API call trace analysis
- Network activities
- Registry modifications
- Process creation/injections
- File activities

*What you see is what you get!*

#### Hybrid Approach
- Semantic-aware detectors
  - Extract dynamic trace
  - Transform into IR
  - Compare to pre-defined templates
- Memory dump analysis (packers)
# Bypassing Detection Methods

<table>
<thead>
<tr>
<th>Pattern Based</th>
<th>Static Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Build variants (e.g. Zeus)</td>
<td>• Packing</td>
</tr>
<tr>
<td>• Append garbage</td>
<td>• Obfuscation</td>
</tr>
<tr>
<td>• Encoding</td>
<td>• Encryption</td>
</tr>
<tr>
<td>• “Stay compliant”</td>
<td>• Anti-reversing techniques</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic Analysis</th>
<th>Hybrid Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Detect analysis*</td>
<td>• Avoid using the same executable template</td>
</tr>
<tr>
<td>• Detect emulation*</td>
<td>• Metasploit AV-evasion</td>
</tr>
<tr>
<td>• Detect security product*</td>
<td>• Reuse “trusted templates”</td>
</tr>
<tr>
<td>• Beat the clock (AV sandbox)</td>
<td>• PowerShell</td>
</tr>
<tr>
<td>• “Split the maliciousness”</td>
<td>• In-memory exploits</td>
</tr>
</tbody>
</table>

*Could be detected during static analysis
MYTH #1
Malware executes immediately
No!
MYTH #2
Malware is usually small
# Malware Detection Based on File Size

<table>
<thead>
<tr>
<th></th>
<th>File Size limit</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>no limit</td>
</tr>
<tr>
<td>exploit</td>
<td>99.83%</td>
<td>99.95%</td>
<td>99.97%</td>
<td>99.97%</td>
<td>99.98%</td>
<td>99.98%</td>
<td>99.99%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>im-worm</td>
<td>98.83%</td>
<td>99.71%</td>
<td>99.90%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>mass-mailer</td>
<td>99.62%</td>
<td>99.87%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>mobile</td>
<td>99.44%</td>
<td>99.78%</td>
<td>99.88%</td>
<td>99.90%</td>
<td>99.93%</td>
<td>99.95%</td>
<td>99.97%</td>
<td>99.98%</td>
<td>99.99%</td>
<td>99.99%</td>
<td>100.00%</td>
</tr>
<tr>
<td>macro virus</td>
<td>99.63%</td>
<td>99.82%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>phish</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>scripts</td>
<td>98.25%</td>
<td>99.64%</td>
<td>99.88%</td>
<td>99.92%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>spyware</td>
<td>95.08%</td>
<td>97.97%</td>
<td>98.88%</td>
<td>99.47%</td>
<td>99.76%</td>
<td>99.83%</td>
<td>99.89%</td>
<td>99.91%</td>
<td>99.94%</td>
<td>99.95%</td>
<td>100.00%</td>
</tr>
<tr>
<td>trojan</td>
<td>97.02%</td>
<td>99.24%</td>
<td>99.62%</td>
<td>99.80%</td>
<td>99.88%</td>
<td>99.93%</td>
<td>99.95%</td>
<td>99.97%</td>
<td>99.98%</td>
<td>99.98%</td>
<td>100.00%</td>
</tr>
<tr>
<td>virus</td>
<td>98.27%</td>
<td>99.37%</td>
<td>99.63%</td>
<td>99.80%</td>
<td>99.89%</td>
<td>99.92%</td>
<td>99.95%</td>
<td>99.97%</td>
<td>99.98%</td>
<td>99.99%</td>
<td>100.00%</td>
</tr>
<tr>
<td>worm</td>
<td>99.02%</td>
<td>99.65%</td>
<td>99.74%</td>
<td>99.86%</td>
<td>99.89%</td>
<td>99.92%</td>
<td>99.94%</td>
<td>99.94%</td>
<td>99.95%</td>
<td>99.96%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

*Size in MB

## Malware Bypassing Detections

### Stuxnet

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>Dynamic Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x Zero-Days</td>
<td><strong>Obfuscated Entry Point</strong> <em>(Needs a special Loader)</em></td>
</tr>
<tr>
<td><strong>2 x Stolen Certificates</strong> <em>(Break the trust)</em></td>
<td><strong>Multiple Files</strong> <em>(lesser maliciousness entropy)</em></td>
</tr>
<tr>
<td>Unknown DLL loading technique</td>
<td><strong>Execution depends on host</strong></td>
</tr>
</tbody>
</table>

### Flame

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>Dynamic Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 MB of Code!</td>
<td><strong>Does not execute immediately</strong></td>
</tr>
<tr>
<td><strong>Breakthrough in cryptography</strong> <em>(Break the trust)</em></td>
<td><strong>Multiple Files</strong> <em>(lesser maliciousness entropy)</em></td>
</tr>
<tr>
<td><strong>Legitimate Libraries</strong> <em>(LUA)</em></td>
<td><strong>Obfuscated Entry Point</strong> <em>(Needs a special Loader)</em></td>
</tr>
</tbody>
</table>
Malware Thwarting Analysis

Mac OSx

Infected with FlashFake

Exploit Java Vulnerability

Send Hardware UUID (Unique)

C&C

UUID

Encryption Key

Encrypt Malware and Obfuscate

Executable will only run on the original host
Problem

Detection is good but not great
Data-Structure Modifications

**Problem:**
Malware modifies internal data-structures to **avoid** detection

**Solution:**
*Subvert* the malware!
Modify the data-structure before the malware does
Detecting Internal DS Modifications

```
push ebp
mov ebp, esp
push ebx
push esi
```

```
JMP F_clone
```

```
push ebp
mov ebp, esp
```

```
<F Clone>
```

```
<Internal DS>
```

```
<jmp TRAP>
```

Detected
PsSetCreateProcessNotifyRoutine

Linear search for a signature

Internal DS
Example::PsSetCreateProcessNotifyRoutine

Original

Clone

PAGE:005567E ; _stdcall PsSetCreateProcessNotifyRoutine(x, x)
PAGE:005567E public _PsSetCreateProcessNotifyRoutine@@ proc near
PAGE:005567E
PAGE:005567E arg_0 = dword ptr 8
PAGE:005567E arg_4 = byte ptr 0Ch
PAGE:005567E
PAGE:005567E
PAGE:005567E
 PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E

PAGE:005567E ; _stdcall PsSetCreateProcessNotifyRoutineClone(x, x)
PAGE:005567E public _PsSetCreateProcessNotifyRoutineClone@@ proc near
PAGE:005567E
PAGE:005567E arg_0 = dword ptr 8
PAGE:005567E arg_4 = byte ptr 0Ch
PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E
PAGE:005567E

Guard
Technique Usage

- Detect function hooking tempering
  - Hook a function and monitor the hook
  - Protect the monitor routine

- Detection of linear memory scanning
  - Staged attacks
  - Egg hunt

- Detection of internal data-structure manipulation
  - Basic DKOM Detection

- Place calls to Page Guard in strategic places
  - Detect Heap Spraying (“canary” value)
**Problem:**
Malware checks for the existence of a security product process

**Solution:**
Process enumeration using weight based mechanism taint analysis
Detecting “weird” Process Enumerations

- Monitor EPROCESS structure access
- Track **process name** usage (taint analysis)
- Score the process based on “weird” usage
  - HASH
  - Encryption
  - Encoding
  - Etc.
Problem: Malware uses obfuscation to hide malicious code during Drive-by-download attacks

Solution: Hook the browser at strategic places and inspect the de-obfuscated buffers
Obfuscation is a problem!
- Network devices are blind

Possible solution on the network side
- Analyze data entropy to detect possible obfuscation
- Google uses obfuscation -> massive FP

Better solution on the end point
- Hook the browsers (IE/Chrome/Firefox) at strategic places
  - Eval, Document.write, innerhtml, etc'
- Let the browser do the “heavy lifting”
- Communicate the information back to the network devices
Problem:
Malware drop anti-VM technique and focus on anti-Analysis techniques

Solution:
Subvert the analysis machine with a Rootkit before executing the malware
Malware usually cannot detect Rootkit!

Install a rootkit on the analysis machine
- Hide files/processes/drivers
- Hide open ports
- Hide registry values

Malware is not aware that it is being subverted

Results in higher detection rate of advanced malware
Easy-to-use rootkit generator

Choose the process/files/ports/registry values you wish to hide

Generates a customize rootkit

Install rootkit

Benefit!
Future Directions

Detecting internal threats using ML
- Most network behavior analysis tools fail to deliver
  - Bad feature sets that results in massive FP
- Feature set focus on user **behavioral** profile and **not** malware
  - Data entropy / Working hours / Keyboard typing speed
- Based on the protoleak project (RSA 12′)
  - Profile-based decision tree **per node**
- Focus on data exfiltration and behavior deviations

Malware Interaction
- Click/Move Mouse
- Open Applications
How to Apply

► Force malware mistakes, don’t wait for them to strike
► Raise attackers cost by innovating mitigations
► Download & try the tools
► Help fighting the 1% and suggest improvements
Thank You
@djteller