Obfuscation-Resilient Executable Payload Extraction From Packed Malware

Binlin Cheng*, Jiang Ming*, Erika A Leal, Haotian Zhang, Jianming Fu, Guojun Peng, Jean-Yves Marion
This Talk is About

- Restoring Executable Malware From Packed Binary Sample
Challenge: API Obfuscation

Original Code Section
......
call [f1];
......
next inst.;

IAT Section
f1: TargetAPI
......

kernel32.dll
TargetAPI:
......
mov edi,edi
push ebp
mov ebp, esp
......
ret

Standard API Call

Original Code Section
......
call [f1];
......
next inst.;

IAT Section
f1: Trampoline
......

Trampoline
......
jmp TargetAPI;

API Obfuscation

kernel32.dll
TargetAPI:
......
mov edi,edi
push ebp
mov ebp, esp
......
ret
The effect of API obfuscation

- Anti-Static Analysis
- Anti-Dynamic Execution
## In-depth study of API obfuscation schemes

<table>
<thead>
<tr>
<th>Obfuscation Type</th>
<th>Control Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard API Call</td>
<td>Original Code ⇒ TargetAPI</td>
</tr>
<tr>
<td>IAT Redirection</td>
<td>Original Code ⇒ Trampoline ⇒ TargetAPI</td>
</tr>
<tr>
<td>Rewrite API Callsite</td>
<td>Original Code ⇒ Trampoline ⇒ TargetAPI</td>
</tr>
<tr>
<td>Anti-debugging Routine</td>
<td>Original Code ⇒ Trampoline ⇒ Anti-debugging API ⇒ Trampoline ⇒ TargetAPI</td>
</tr>
<tr>
<td>ROP Redirection</td>
<td>Original Code ⇒ Trampoline ⇒ End of TempAPI ⇒ Trampoline ⇒ TargetAPI</td>
</tr>
<tr>
<td>Stolen Code</td>
<td>Original Code ⇒ Trampoline ⇒ TargetAPI+n</td>
</tr>
</tbody>
</table>
Assumptions of API de-obfuscation approaches (1)

• Assumptions 1:
  Target API’s address can be statically identified in the unpacked code.

• Exception case:
  IAT Redirection via SEH:

(b) IAT Redirection via SEH
Assumptions of API de-obfuscation approaches (2)

- **Assumptions 2:**
  When the control flow arrives at a DLL, it necessarily points to the target API’s entry point.

- **Exception cases:**
  - Anti-debugging Routine
  - ROP Redirection
  - Stolen Code
Assumptions of API de-obfuscation approaches (3)

- Assumptions 3:
  API calls are necessarily referred to the IAT.

- Exception case:
- Rewrite API Callsite
Our Approach

- **API-Xray**: A hardware-assisted approach without any assumption.
Hardware-Assisted API Micro Execution (1)

- **Req1**: executing the trampoline code at each API callsite;

- **Solution**: API Micro Execution.

[ICSE’14] Patrice Godefroid. Micro Execution
• **Req2**: capturing the control flow branch in trampoline, so that we can identify the target API.

• **Solution**: Intel BTS

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBR</td>
<td>It records 16 or 32 most recent branch pairs into a register.</td>
</tr>
<tr>
<td>BTS</td>
<td>It records all kinds of branch pairs into a memory buffer</td>
</tr>
<tr>
<td>IPT</td>
<td>It does not record unconditional direct branches</td>
</tr>
</tbody>
</table>
The evaluation of API-obfuscation resistance

Table 5: The comparison of API-obfuscation resistance. “●” means this tool can defeat an API obfuscation type.

<table>
<thead>
<tr>
<th>Obfuscation Type</th>
<th>BinUpack</th>
<th>S&amp;P 15</th>
<th>RePEconstruct</th>
<th>API-Xray</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAT Redirection</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Rewrite API Callsite</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Stolen Code</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>ROP Redirection</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
<tr>
<td>Anti-debugging Routine</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
</tr>
</tbody>
</table>
Table 7: The distribution of API obfuscation types.

<table>
<thead>
<tr>
<th>API Obfuscation Type</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1: IAT Redirection</td>
<td>36.5%</td>
</tr>
<tr>
<td>Type 2: Stolen Code</td>
<td>12.7%</td>
</tr>
<tr>
<td>Type 3: Rewrite API callsite</td>
<td>11.8%</td>
</tr>
<tr>
<td>Type 4: Anti-debugging Routine</td>
<td>7.8%</td>
</tr>
<tr>
<td>Type 5: ROP Redirection</td>
<td>6.9%</td>
</tr>
</tbody>
</table>
Table 8: The case study of an unknown malware sample.

<table>
<thead>
<tr>
<th>Sample</th>
<th>#APIs</th>
<th>#VirusTotal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unpacked Code</td>
<td>API-Xray</td>
</tr>
<tr>
<td>Unknown Trojan(^1)</td>
<td>0</td>
<td>63</td>
</tr>
</tbody>
</table>

\(^1\) MD5: d4f377c849b86d5ca89776bc56eea832.
Possible Attacks

• Attacks to BTS
• Attacks to NX bit.
• Statically-Linked Library
• Stolen Function.
• Argument-Sensitive Trampoline.

Please see our countermeasures in our paper!
Limitations

• Custom DLLs.
  API-Xray cannot restore import tables from custom DLLs, which are absent in our testing environment.

• OEP Obfuscation.
  Some unpacked PE files with complete import tables crashed at run time due to the OEP obfuscation.
Application to Linux Malware

- API-Xray’s technique is applied to Linux malware as well.
- That’s because API-Xray is designed to work on Intel CPU, which is independent of OS.
Q & A

Binlin Cheng (binlincheng@163.com) & Jiang Ming (jiang.ming@uta.edu)

http://ranger.uta.edu/~ming/