Matryoshka: Strengthening Software Protection via Nested Virtual Machines

Sudeep Ghosh (Microsoft Research)
Jason D. Hiser (University of Virginia)
Jack W. Davidson (University of Virginia)
The Problem

Software is now used to perform critical functionality.
- Banks
- Power grids
- Medical software
- Transportation systems
- Internet of Things

- Disassembler
- Debuggers
- Static Analyzers
- Coverage tools
- Simulators

Researchers must continually upgrade and enhance software protection approaches
Threat Model

- Software is created in a secure environment.
- White-box attack model
  - Adversary has access to multiple tools including debuggers, simulators and emulators.
  - They can modify the OS to return inaccurate information.
  - As such, the adversary can inspect, modify or forge any information.
- Given enough time and resources, the adversary can succeed in manually inspecting and modifying programs.
- However, most attacks use algorithmic solutions to disable security features in programs.
Virtualization

System-level VM

Process-level VM
Strata

- Infrastructure designed for building process-level virtualization systems
- Designed with extensibility, portability, and application-independence in mind
- Implement new systems by customizing the VM

- Binary only
  - No source code required
  - Can be applied to any application regardless of source language, compiler used, libraries used, etc.
- Provides common services necessary for software dynamic translation
Strata

Overhead Normalized to Native Run

- DynamoRIO
- PIN
- HDTrans
- Strata

Perlbench  gobmk  gcc  dealII  mcf  Average
Strata Related Work

- **SDT**

- **Obfuscation and anti-tamper**
  - What’s the PointISA? IH 2014.
  - A Secure and Robust Approach to Software Tamper Resistance. IH 2010

- **Security**
Matryoshka: Nested PVMs
Software Protection via Virtualization

Protection Scheme (Guards, encryption) -> Builder (Diablo) -> Protected Binary

Application

PVM_1  PVM_2  PVM_2

PVMs

G1  G2  G3  G4
Nested PVMs

Memory

F$ (SC1)

Strata1

Translated Application Code

Strata2

Translated Application Code

F$ (SC2)

X86 (Hardware)

Disk Image
Use compression ratio as a proxy for diversity and obfuscation.

- Single PVM: 149; N-PVM: 15.63
Evaluation

Cyclomatic Complexity

- Developed by McCabe in 1976 as a measure of software complexity (TSE Vol. 2, No. 4)
- \( M = E - N + 2P \)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CC for PVM</th>
<th>CC for N-PVM</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>176.gcc</td>
<td>1,604</td>
<td>80,109</td>
<td>49X</td>
</tr>
<tr>
<td>181.mcf</td>
<td>351</td>
<td>9828</td>
<td>27X</td>
</tr>
<tr>
<td>256.perlbmk</td>
<td>803</td>
<td>32,903</td>
<td>40X</td>
</tr>
<tr>
<td>179.Art</td>
<td>181</td>
<td>5,130</td>
<td>27X</td>
</tr>
</tbody>
</table>
Run-time Overhead

- With a nesting level of two, the base run-time overhead was 35X.
- The problem is trampoline patching (i.e., self-modifying code), which causes excessive F$ flushes.
Nested PVMs

Memory

F$ (SC1)

Strata!

Translated Application Code

F$ (SC2)

Translated Application Code

Disk Image

X86 (Hardware)
Super Patching

- When \( \text{Strata}_n \) patches a trampoline, the patch information is sent to \( \text{Strata}_{n-1} \).
- When a patched (in \( F^2 \)) target block is translated to \( F^1 \) by \( \text{Strata}_{n-1} \), \( \text{Strata}_{n-1} \) patches its \( F^1 \) (\( F^1_{n-1} \)), thereby avoid the \( F^1 \) flush.
Related Work

- Collberg and Nagra [Pearson 2006] provide an excellent overview of the area.
- Anckaert et al [DRM 2006] showed the promise of virtualization for software protection.
- Anckaert et al [QoP 2007] discuss metrics for software protection.
Summary

- Nested PVMs can significantly increase the complexity of software that is the target of crackers
- More research is needed to determine when and how to apply nested VMs to software to balance run-time performance and the strength of the protection provided