Strengthening Self-Checksumming via Self-Modifying Code

Jonathon T. Giffin, Mihai Christodorescu, Louis Kruger

Computer Sciences Department
University of Wisconsin

{giffin,mihai,lpkruger}@cs.wisc.edu
Problem 1

Detect malicious modifications to code
Problem 1

Detect malicious modifications to code
Solution: Self-Checksumming

Program contains code to checksum parts of its own code.
Solution: Self-Checksumming

- **Network of guards**  
  - Many overlapping checksumming components  
  
- **Integrity Verification Kernels**  
  - Multithreaded, self-modifying checksumming components  

- **Testers and correctors**  
  
[Chang & Atallah 2001]

[Aucsmith 1996]

[Horne et al. 2001]
Problem 2

Is the checksummed & validated code actually the code executed?
Normal Memory Accesses

CPU → RAM

- Write
- Read
- Fetch

RAM:
- Code
- Data
Page-Replication Attack

[Wurster et al. 2005]
Page-Replication Attack

x86: Set DS & CS registers to different memory segments

CPU

[D-RAM]

Data

Code

[I-RAM]

Code

Read

Write

Fetch

[Wurster et al. 2005]
Page-Replication Attack

SPARC: Fill D-TLB and I-TLB entries at same virtual address with different physical addresses

[D-RAM]

Data

Code

[CPU]

Write

Read

I-RAM

Code

[Fetch]

[Wurster et al. 2005]
Page-Replication Attack

[D-RAM]

Genuine

[Code]

[Code]

[Altered]

[I-RAM]

[Fetch]

[Read]

[Write]

[CPU]

[Data]

[Wurster et al. 2005]
Solution

Observation:
Writes to code affect program differently when a page-replication attack is underway

Use **self-modifying code** to detect page-replication attack
Solution

1. Overwrite instruction $I_1$ at address $\nu$ with new instruction $I_2$ that alters control-flow
2. Read back the value at $\nu$
3. Execute the instruction at $\nu$
No attack: Code alteration visible to both data read & instruction fetch
No attack: Code alteration visible to both data read & instruction fetch
Page-replication attack: Read / fetch mismatch

Attack Detection

CPU

Write
Read

Fetch

D-RAM

Data

Genuine

I-RAM

I₁

Altered

I₁

I₁
Attack Detection

Page-replication attack: Read / fetch mismatch

CPU

D-RAM
Data
\( I_2 \)

I-RAM
\( I_1 \)

Fetch
Write
Read

Genuine

Altered
Solution

1. Overwrite instruction $I_1$ at address $v$ with new instruction $I_2$ that alters control-flow
2. Read back the value at $v$
3. Execute the instruction at $v$

<table>
<thead>
<tr>
<th>Control-flow path followed</th>
<th>Value read</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>Attack</td>
</tr>
<tr>
<td>$I_2$</td>
<td>Attack</td>
</tr>
<tr>
<td>$I_1$</td>
<td>Attack</td>
</tr>
<tr>
<td>$I_2$</td>
<td>No Attack</td>
</tr>
</tbody>
</table>
Self-Checksumming and Reality

Jonathon T. Giffin, Mihai Christodorescu, Louis Kruger

Computer Sciences Department
University of Wisconsin

{giffin,mihai,lpkruger}@cs.wisc.edu
Taking Stock

... So self-checksumming works again, right?

No.

Self-checksumming will always fail in current, realistic threat models.
Problem 3

Attackers first remove checksum code, then maliciously modify program.
Solution: Redefine the Threat

The attacker cannot identify all relevant checksum code within the protected program.

“cannot identify” ➔ “cannot reverse engineer”

➔ Obfuscate
Solution: Redefine the Threat

- **Network of guards**  
  - Many overlapping checksumming components  
  [Chang & Atallah 2001]

- **Integrity Verification Kernels**  
  - Multithreaded, self-modifying checksumming components  
  [Aucsmith 1996]

- **Testers and correctors**  
  [Horne et al. 2001]
Solution: Redefine the Threat

The attacker cannot identify all relevant checksum code within the protected program.

The attacker can reverse engineer & modify any non-checksumming code...

...but the attacker cannot reverse engineer & remove the checksum computation code.
Realistic Threats

The attacker can understand and arbitrarily alter any code in the program.

[Madou et al. DRM 2005]
Root Problem

No trust base.

- Malicious Operating System
- Malicious CPU
- Process
Root Problem

No trust base.

Self-checksumming will inherently and always fail in such an environment.
Root Problem

No trust base.

“Software alone never gets you assurance.”

“Need independent processor & address space.”

-- Brian Snow, 9:29 AM today
Solution

Trusted computing; remote verification

Process
Malicious Operating System
TPM + CPU

Trusted Verifier
Trusted Operating System
Signed Checksum

Trusted CPU
Solution

Trusted computing; remote verification

Trusted hardware alone is insufficient:

Malicious OS or malicious process can alter or remove local verification routines
Solution

Trusted computing; remote verification

Remote verification alone is insufficient:

Malicious OS can again mount page-replication attacks
Conclusions

• Strengthening self-checksumming via self-modifying code
  – Detects page-replication attack

• Fundamental attacks against self-checksumming remain valid

• Trusted hardware + remote verification needed for secure checksum validation
Questions?

Contact the authors:

Jonathon T. Giffin       giffin@cs.wisc.edu
Mihai Christodorescu    mihai@cs.wisc.edu
Louis Kruger            lpkruger@cs.wisc.edu

Computer Sciences Department
University of Wisconsin