Design and Implementation of a TCG-based Integrity Measurement Architecture

Reiner Sailer, Trent Jaeger, Leendert van Doorn, and Xiaolan Zhang
Secure Systems Department
Overview

- Problem – Runtime integrity guarantees
- Solution – Hierarchical software-stack measurements
  - Load guarantees
  - Property attestation
- Current Implementation
- Future Work
Problem – What is the Integrity of a System?

- Insecure networked world
- SSL and IPSEC provide secure channels
  Answers: With whom am I interacting securely?

Open Problem:
How can you trust this system?
Trusted Computing Group Architecture

Execution Flow

1. CRTM code
2. OS Loader code
3. OS code
4. Application code

Measurement Flow

- Defined by TCG (Platform specific)
- Defined by Grub (IBM Tokyo Research Lab)

TCG-based Integrity Measurement Architecture

Platform Configuration Registers 0-23

0-7 4-7 >= 8
Integrity Measurement Architecture – Solution

Measurement

SHA1(Boot Process)
SHA1(Kernel)
SHA1(Kernel Modules)
SHA1(Program)
SHA1(Libraries)
SHA1(Configurations)
SHA1(Structured data)

TPM-Signed
PCR Integrity Value

System-Representation

ext. Information
(CERT,...)

System Properties

Attested System

Analysis

Known Fingerprints
TPM-Based Integrity Measurement Architecture

- Achievement of our Integrity Measurement Architecture (IMA)
  - Extend TPM-based attestation into the system runtime
  - Attest the Software Stack

- IMA-Guarantees
  - Non-intrusive (not changing system behavior)
  - Load-guarantees for code loaded into the system run-time
  - Detects systems cheating with the measurement list

- Goals
  - Negligible overhead on attested system
  - Usability
Example: Web Server

- **Executables** (Program & Libraries)
  - apachectl, httpd, java, ...
  - mod_ssl.so, mod_auth.so, mod_cgi.so, ...
  - libc-2.3.2.so libjvm.so, libjava.so, ...

- **Configuration Files**
  - httpd.conf, html-pages,
  - httpd-startup, catalina.sh, servlet.jar

- **Unstructured Input**
  - HTTP-Requests
  - Management Data
IMA Implementation – File Measurements

**Measurement** = SHA1(File Contents) at load time

- **Kernel** measures: kernel modules, programs, and shared libraries
- **Applications** measure their own critical input
  
  Examples:
  - Bash Shell measures: scripts before execution
  - Future: Java, Perl, Apache, Jakarta Tomcat …

**Advantage**

- **Unique** Software-Fingerprints (e.g. sendmail-8.12.8-9.90)
  
  → Secure hash represents well known security properties
IMA Implementation – Measurement List Maintenance

**Measurement list aggregation:**

- **Compute** 160bit-SHA1 over the contents of the data (measurement)
- **Adjust** Protected hw Platform Configuration Register (PCR) to maintain measurement list Integrity Value
- **Add** measurement to ordered measurement list
  - Executable content is recorded before it impacts the system
  - That is, before it can corrupt the system

```
PCR0 := 0
System-start
```

```
PCR_k := SHA1( PCR_k || new measurement)
```

```
new measurement
```

```
k
k+1
```

```
Measurement List
PCR Integrity Value
```
IMA Implementation – Measurements by the Kernel

Linux Security Module → Traditional execution path

Execve (*file)

SHA1

Measurement List (Kernel-held)

Memory Map → Schedule

/bin/bash
IMA Implementation – Measurements by Applications

```c
fd = open("http.conf")
measure(fd)
read(fd)
close(fd)
```

ToM-ToU Detection: reader/writer lock

```
SHA1
```

```
/etc/
http.conf
```

```
Integrity Value
```

```
Measurement List (Kernel-held)
```

```
Kernel
```

```
```

Attestation Protocol

1. Create unpredictable Nonce

2. Nonce (160bit)

3. $\text{Sig}(\text{Nonce, PCR})^\text{TPM}$, Measurement List

4. Check TPM Signature & Nonce

5. Validate Measurement List against PCR Integrity Value

6. Validate individual Measurements using Fingerprint DB
### Example: Rootkit compromise analysis

<table>
<thead>
<tr>
<th>Measurement List</th>
<th>Fingerprint DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 : D6DC07881A7EFD58EB8E9184CCA723AF4212D3DB</td>
<td>boot_aggregate</td>
</tr>
<tr>
<td>001 : 84ABD2960414CA4A448E0D2C9364B4E1725BDA4F</td>
<td>init</td>
</tr>
<tr>
<td>002 : 194D956F288B36FB46E46A124E59D466DE7C73B6</td>
<td>ld-2.3.2.so</td>
</tr>
<tr>
<td>003 : 7DF33561E2A467A87CDD4BB8F68880517D3CAECB</td>
<td>libc-2.3.2.so</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>110 : F969BD9D27C2CC16BC668374A9FBA9D35B3E1AA2</td>
<td>syslogd</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

(a) THE GOOD CASE

| 110 : F969BD9D27C2CC16BC668374A9FBA9D35B3E1AA2 | syslogd                 |
| ...              | ...                     |
| 525 : 4CA3918834E48694187F5A4DAB4EECD540AA8EA2 | syslogd-LRK5           |
| ...              |                         |

(b) LRK5-COMPROMISED SYSLOGD
Results

Attested System:

- Implementation: ~ 5000 LOC (LSM kernel module)
- About 400-600 measurements for Fedora C2, Apache, Jakarta Tomcat, etc.

Measurement Overhead

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Application</th>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean Hit</td>
<td>~ 0.1 μs</td>
<td>~ 5 μs</td>
</tr>
<tr>
<td>New (TPM) Measurement</td>
<td>~ 5 ms + SHA1 (~80MB/s)</td>
<td>~ 5ms + SHA1 (~80MB/s)</td>
</tr>
</tbody>
</table>

Attestation service:

- Known Fingerprint DB ~ 20 000 Fingerprints (RedHat 9.0, Fedora, ES3)
- Attestation: 1-2 second “latency” (unoptimized demonstration)
Ongoing & Future Work

Open-Source Integrity Measurement Architecture

- LSM kernel module

Isolation of unknown or distrusted measurements

- Measure Information flow between executables

Predict future system states

- Measure SELinux policy and enforcement
Thank You!

Further Information

http://www.research.ibm.com/secure_systems_department/projects/tcglinux