Panoply:
Low-TCB Linux Applications
With SGX Enclaves

Shweta Shinde  Dat Le Tien
Shruti Tople  Prateek Saxena

National University of Singapore
TCB: Hosting a Web Server

Current systems have a large TCB

- Hypervisor: 150 KLOC
- Operating System: 183 KLOC
- Webserver: 10 MLOC

150 KLOC
183 KLOC
10 MLOC
SGX: Hardware-root of Trust

- Web Server Enclave
- Other Applications
- Operating System / VMM
- RAM
- EPC

Ring 3

Ring 0 - 2

- Trusted
- Untrusted
SGX: Hardware-root of Trust

- Hardware
- Hypervisor
- Operating System
- Webserver

Enclave

Confidentiality and Integrity

150 KLOC

183 KLOC

10 MLOC
..but limits the expressiveness of the applications (e.g., no syscalls)
TCB & Expressiveness Trade-off

TCB

~1MLOC

~100KLOC

~KLOC

Expressiveness

TCB & Expressiveness Trade-off

LibraryOS

Haven [OSDI’14]

Graphene-SGX [EuroSys’14]

Containers

Scone [OSDI’16]

Ryoan [OSDI’16]

?.

Syscall

Syscall Threads

Syscall Threads Event Mgmt

Syscall Threads Event Mgmt Fork/ exec
Contributions

• Panoply
  – Expressiveness: All standard POSIX APIs
  – Low TCB: 2 orders of magnitude smaller than LibraryOS
  – Library-enclaves for fine-grained TCB

• Evaluation
  – Absolute 24% and 5-10% compared to LibraryOS
Problem
Challenge I: Expressiveness vs TCB

Webserver
- Syscalls
- Threading
- Event Handling
- Forking

Legacy Application Design

Enclave
- Fork Emulation
- Event Management
- Thread Handling
- Syscall Emulation
- Web Server

TCB
Challenge I: Expressiveness vs. TCB
Challenge II: Multi-Enclave Applications

Single Enclave Application

- zlib
- libevent
- libcrypto
- libssl
- Web Server

Multi-Enclave Application

Web Server

Operating System

libssl
Attacks on Multi-Enclave Applications

```c
session_t session;
certificate_credentials_t xcred;

/* Specify callback function*/
certificate_set_verify_function (...);

/* Initialize TLS session */
init (&session, TLS_CLIENT);
```

[SSL Lib]
Attacks on Multi-Enclave Applications

- Webserver Enclave
- OS
- SSL Library Enclave

- Drop
- Spoof
- Replay
Our Solution: Panoply
Panoply Runtime

Microns keep libc outside the enclave

Micron

Enclave-bound Logic

Panoply Shim Lib  Trusted SGX Lib

Enclave

libc.so  Non-enclave Logic  Untrusted SGX Lib

Non-Enclave

Linux User-level Process
Overview

1. Compiler Instrumentation
   - Add calls to Panoply API
   - Add Flow Checks

2. Creating Enclaves
   - Enclave-bound Code
   - Panoply Shim
   - Intel SGX SDK

Panoply Application
Challenge I: Expressiveness

Delegate rather than emulate

- On-demand threading
- Multi-processing
- Syscalls
- Event Handling

Enclave
## Expressiveness: Panoply APIs

### Core Services

<table>
<thead>
<tr>
<th>Service</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Creation and Control</td>
<td>5</td>
</tr>
<tr>
<td>Signals</td>
<td>6</td>
</tr>
<tr>
<td>Timers</td>
<td>5</td>
</tr>
<tr>
<td>File and Directory Operations</td>
<td>37</td>
</tr>
<tr>
<td>Pipes</td>
<td>4</td>
</tr>
<tr>
<td>C Library (Standard C)</td>
<td>66</td>
</tr>
<tr>
<td>I/O Port Interface and Control</td>
<td>40</td>
</tr>
</tbody>
</table>

### Real-time Extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-Time Signals</td>
<td>4</td>
</tr>
<tr>
<td>Clocks and Timers</td>
<td>1</td>
</tr>
<tr>
<td>Semaphores</td>
<td>2</td>
</tr>
<tr>
<td>Message Passing</td>
<td>7</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>6</td>
</tr>
<tr>
<td>Asynchronous and Synchronous I/O</td>
<td>29</td>
</tr>
<tr>
<td>Memory Locking Interface</td>
<td>6</td>
</tr>
</tbody>
</table>

### Thread Extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread Creation, Control, and Cleanup</td>
<td>17</td>
</tr>
<tr>
<td>Thread Scheduling</td>
<td>4</td>
</tr>
<tr>
<td>Thread Synchronization</td>
<td>10</td>
</tr>
<tr>
<td>Signal Delivery</td>
<td>2</td>
</tr>
<tr>
<td>Signal Handling</td>
<td>3</td>
</tr>
</tbody>
</table>

### POSIX APIs

 Supported for Commodity Linux Apps
Expressiveness Example: Fork

LibraryOSes emulate fork semantics

Fork Semantics

- **Parent Process**
  - Page Copy
  - PID Mgmt

- **Child Process**
  - Page Copy
  - PID Mgmt

LibraryOS Fork Implementation

- **Parent Enclave**
  - Page Copy
  - PID Mgmt

- **Child Enclave**
  - Page Copy
  - PID Mgmt

OS
Expressiveness Example: Delegating Fork

- Creating child process and child enclave

- Child enclave has a clean memory state
Expressiveness Example: Achieving Fork Semantics

- Mirroring parent’s memory in child enclave
  - After the fork call, before resuming execution
Expressiveness Example: Achieving Fork Semantics

• Mirroring parent’s memory in child enclave
  – Full replica: default mode in Panoply

• Alternative strategies to full replica
  – Copy on demand: Requires page-fault support from SGX v2

  – Copy on need: Replicate selected addresses which are pre-determined by static analysis
Expressiveness Example: Multi-Threading

Virtual Threads

Micron A

Panoply Shim

TCS 1
TCS 2
TCS 3

Micron A'

Panoply Shim

TCS 1
TCS 2
TCS 3

Shared Variables
Thread Control Manager Micron
Challenge II: Multi-enclave Applications

Webserver Enclave

OS

SSL Service Enclave

Drop

Spoof

Replay
Securing Multi-Enclave Apps

Enclave Identity  Pair-wise Nonce  OS

Enclave 1  Enclave 2

Call Ack

Attack
Spoofing
Replay
Silent Drops

Security Property
Sender / Receiver Authentication
Message Freshness
Reliable Delivery
Evaluation
Benchmarks

• Real-world use-cases for SGX
  – 4 apps: Tor, H2O web server, FreeTDS, OpenSSL

• Operating system stress testing
  – 26 LMBench benchmarks tests
  – 17 metrics for memory, network, signal, syscall APIs
## TCB Evaluation

### Panoply

<table>
<thead>
<tr>
<th>Component</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panoply Library</td>
<td>10425</td>
</tr>
<tr>
<td>API Wrappers</td>
<td>9788</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20213</strong></td>
</tr>
</tbody>
</table>

### Graphene-SGX

<table>
<thead>
<tr>
<th>Component</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glibc</td>
<td>1156740</td>
</tr>
<tr>
<td>libPal-LinuxSGX</td>
<td>16901</td>
</tr>
<tr>
<td>libPal-enclave</td>
<td>33103</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1206744</strong></td>
</tr>
</tbody>
</table>

Panoply reduces TCB by 2 orders of magnitude.
Performance Evaluation

- Create delete takes large fraction of the time
- Overhead increases with number of Out-Calls

Panoply incurs 24% overhead

<table>
<thead>
<tr>
<th>App</th>
<th>Panoply</th>
<th>Empty Enclave</th>
<th>Overhead (% increase)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL</td>
<td>3.16</td>
<td>2.79</td>
<td>13</td>
</tr>
<tr>
<td>H2O</td>
<td>8.79</td>
<td>6.56</td>
<td>34</td>
</tr>
<tr>
<td>FreeTDS</td>
<td>8.74</td>
<td>8.60</td>
<td>1</td>
</tr>
<tr>
<td>Tor</td>
<td>6.72</td>
<td>4.54</td>
<td>48</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>24</strong></td>
<td><strong>24</strong></td>
<td><strong>24</strong></td>
</tr>
</tbody>
</table>
Throughput Evaluation

Size of served static web-page

Throughput in Requests/second

Baseline H2O

Panoply H2O

200 Bytes

1KB

6KB

94933.2

82480.69

48490.84

66042.22

42293.46

12818.84
Throughput Evaluation

Overhead for SGX-apps is proportional to the size of requests
Comparison with Graphene-SGX
Comparison with Graphene-SGX

Panoply performance varies by 5-10% as compared to Graphene-SGX
Conclusion

TCB

- ~1MLOC
- ~100KLOC
- ~KLOC

Expressiveness

- 254 APIs
- Panoply - 20KLOC
- 24% Overhead
Contact

• Shweta Shinde
  shweta24@comp.nus.edu.sg

• Panoply Benchmarks & Case-studies:
  http://shwetasshinde24.github.io/Panoply/

Thank You!
References

• [OSDI’ 14] A. Baumann, M. Peinado, and G. Hunt, Shielding Applications from an Untrusted Cloud with Haven


• [OSDI’ 16] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data

• [EuroSys’ 14] Graphene-SGX Library OS - a library OS for Linux multi-process applications, with Intel SGX support