On the Creation of a Secure Key Enclave via the Use of Memory Isolation in Systems Management Mode

Dr James Sutherland
Dr Natalie Coull
Dr Ian Ferguson
ian.ferguson@abertay.ac.uk
Intro

• What have we done?
  – Built a secure key-store using only commodity hardware and the existing facilities of the X86 architecture.
  – Evaluated it's functionality, security and performance.

• Talk outline
  – Problem
  – SMM
  – Experimental evaluation
Problem:

- Keeping crypto-keys safe whilst they are in RAM being used
Use case: verification

1. Document arrives...
2. Decrypt...
3. User?
4. Disk?
5. RAM?
6. ...but where from?
7. RAM
...encrypted with my public key
...with my private key

Abertay University
Paged Virtual Memory System

- Pages are 'randomly' intermingled.
- **Should** be protected by the virtual memory system.
  - A process **should** not be able to access a page it doesn't own.....

  ...but.....

- RowHammer (for example)
Motivation

- RowHammer etc.
  - Unexpected interaction between physically proximate memory components – allowed access to 'local' page
  - Privilege escalation due to sensitive system (virtual) memory pages being intermingled with low-privilege pages.
  - Virtual Machines/hypervisors

- Encryption keys stored in RAM….vulnerable
Existing Approaches....

- .....to securing key enclaves.
- Protecting memory
- RAM encryption
- Address Space Layout Randomisation
- Swap encryption
- Process separation
- Process isolation
- VM isolation
- TPM
- SGX
SMM - SMRAM

- A block of DRAM that can only be addressed by the processor (no DMA from other bus devices)...

- ... when the processor is in Systems Management Mode.
Processor

Memory Control Hub

RAM

Legacy Video RAM

SMRAM

data bus

address bus

SMIACT2

D_OPEN

0x0000

0xA0000

0xC0000

0x000000

0xC0000
Systems Management Mode

Locked SMRAM area – only addressable once in SMM

Entry call = switch to SMM via syscall

Exit SMM to userland operation
Proposed Solution

• Overall operation
  – Key negotiation
  – Transition to SMM
Proposed Solution

- 4K RAM page
- mailslot
- signature
- verification
- entry call = switch to SMM via
- exit SMM to userland operation

locked SMRAM area – only addressable once in SMM

enclave

private keys etc.

SMM code
Generalisable authentication

- Technique can protect keys and code for a variety of authentication/crypto purposes in the enclave
Specific example - Webserver

- To prove the SMM enclave approach works, we built a secure webserver that can prove its identify by signing responses with keys/code stored in the enclave.
  - Does it work?
  - Is it secure?
  - Is it fast enough?
# Evaluation – Four Experiments

<table>
<thead>
<tr>
<th>Num</th>
<th>Experiment</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use with range of browsers</td>
<td>Verifying basic webserver functionality</td>
</tr>
<tr>
<td>2</td>
<td>Qualys - SSL Labs</td>
<td>Verifying webserver SSL protocol compliance</td>
</tr>
<tr>
<td>3</td>
<td>Micro-benchmarking</td>
<td>Measuring the ‘real-time’ overhead imposed by entering and exiting SMM</td>
</tr>
<tr>
<td>4a</td>
<td>Comparison of webserver performance with crypto operation performed with 3 different levels of protection</td>
<td>Measuring the rate that pages could be served with crypto-keys handled in-process, i.e., with no protection</td>
</tr>
<tr>
<td>4b</td>
<td></td>
<td>Measuring the rate that pages could be served with crypto-keys handled in a separate process, i.e., with process-separation protection</td>
</tr>
<tr>
<td>4c</td>
<td></td>
<td>Measuring the rate that pages could be served with crypto-keys handled in SMM</td>
</tr>
</tbody>
</table>
Evaluation Process - Functionality

- Tested with a range of browsers/web-clients
  - No problems
Evaluation – Security

SSL Report: home.deadnode.org

Summary

Overall Rating

Certificate
Protocol Support
Key Exchange
Cipher Strength

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server's certificate is not trusted, see below for details.

Certificate #1: EC 256 bits (SHA256withECDSA)

Server Key and Certificate #1

Subject: home.deadnode.org

Common names: home.deadnode.org

Alternative names: home.deadnode.org

Serial Number: NA/Unknown
Evaluation – Performance

• Is using SMM practical?
• Does it slow down the system too much to be useful?
  - Micro-benchmarking
    • Real time measurements of the transitions to-from SMM
  - Webserving comparison
    • How fast can we serve pages with different levels of key-isolation?
# Evaluation – Micro-benchmarking

<table>
<thead>
<tr>
<th>Operation</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP SMI</td>
<td>Round trip to/from SMM</td>
</tr>
<tr>
<td>open-close</td>
<td>System call requiring access to kernel memory</td>
</tr>
<tr>
<td>getpid()</td>
<td>Trivial system call to reflect minimal kernel transition cost</td>
</tr>
<tr>
<td>signing</td>
<td>Execute a cryptographic operation - specifically generate a signed certificate</td>
</tr>
<tr>
<td>Model</td>
<td>X200</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>CPU</td>
<td>Core 2 Duo P8400</td>
</tr>
<tr>
<td>Clockspeed</td>
<td>2.26 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GiB</td>
</tr>
<tr>
<td>BIOS</td>
<td>Libreboot</td>
</tr>
</tbody>
</table>
## Micro-benchmarking results

### Table V. Execution time for system calls and SMI invocations

<table>
<thead>
<tr>
<th>Operation</th>
<th>X200</th>
<th>T60</th>
<th>T60 Qemu-KVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units</td>
<td>µs</td>
<td>µs</td>
<td>TSC</td>
</tr>
<tr>
<td>NOP SMI</td>
<td>448</td>
<td>Not available</td>
<td>1310</td>
</tr>
<tr>
<td>getpid</td>
<td>0.4</td>
<td>1.1</td>
<td>620</td>
</tr>
<tr>
<td>open/close</td>
<td>3</td>
<td>7.1</td>
<td>3900</td>
</tr>
<tr>
<td>signing</td>
<td>Not available</td>
<td>878</td>
<td>1.606m</td>
</tr>
</tbody>
</table>
### TABLE VI. **Execution Time (TSC ticks) on Bare Metal**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Minimum</th>
<th>1st Quartile</th>
<th>Median</th>
<th>3rd Quartile</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>getpid</td>
<td>1133</td>
<td>1155</td>
<td>1155</td>
<td>1155</td>
<td>5211503</td>
</tr>
<tr>
<td>open-close</td>
<td>6347</td>
<td>6479</td>
<td>6512</td>
<td>6545</td>
<td>3776872</td>
</tr>
<tr>
<td>signing</td>
<td>1534995</td>
<td>1542285.25</td>
<td>1544378</td>
<td>1547757.75</td>
<td>2924856</td>
</tr>
</tbody>
</table>

### TABLE VII. **Execution Time (TSC ticks) under KVM**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Minimum</th>
<th>1st Quartile</th>
<th>Median</th>
<th>3rd Quartile</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP SMI</td>
<td>2235276</td>
<td>2326436.75</td>
<td>2921712.5</td>
<td>3618389</td>
<td>26339800</td>
</tr>
<tr>
<td>getpid</td>
<td>20229</td>
<td>20295</td>
<td>20317</td>
<td>20361</td>
<td>33031357</td>
</tr>
<tr>
<td>open-close</td>
<td>44902</td>
<td>45397</td>
<td>45496</td>
<td>45595</td>
<td>29565196</td>
</tr>
<tr>
<td>signing</td>
<td>1536480</td>
<td>1543069</td>
<td>1546578</td>
<td>1596921</td>
<td>12533972</td>
</tr>
</tbody>
</table>
Webserving

- Testing speed of page serving with 3 level of key protection:
  - Q0 - None
  - Q1 - Process separation (None SMM)
  - Q2 - Full SMM isolation
- https requests generated via curl
- Page size varied
Performance in each configuration

- Requests per second
- Response size (1k, 10k, 100k, 1M)
- Configurations: T60 0, T60 1, Q 0, Q 1, Q 2
Conclusions

- The SMM technique offers greater key protection than process separation with minimal impact on processing speed.
Future Work

- Intrusion counter-measures
- Operation batching
- Other applications/protocols