The DragonBeam Framework: Hardware-Protected Security Modules for In-Place Intrusion Detection

Man-Ki Yoon, Mihai Christodorescu, Lui Sha, Sibin Mohan

University of Illinois at Urbana-Champaign
Qualcomm Research Silicon Valley

June 6, 2016
Security Monitoring

In-place Monitoring

External Monitoring

The DragonBeam Framework
Security Monitoring

In-place Monitoring

External Monitoring

Unsafety of the monitor
Security Monitoring

In-place Monitoring

External Monitoring

Unsafety of the monitor

Semantic gap

The DragonBeam Framework
DragonBeam Framework

Untrusted Layer

Secure Layer

Command/Response
DragonBeam Framework

Monitored Core

- Application
- SKM
- OS

Secure Core

- SecMan
- Secure Memory

Untrusted Layer

Secure Layer

Command/Response
DragonBeam Framework

- **Untrusted Layer**
  - OS
  - Application

- **Monitored Core**
  - SKM

- **Secure Layer**
  - Secure Core
  - SecMan

- **Secure Kernel Module**
  - Performs security monitoring operations
  - Expands the observability
  - Protected by Secure Core
The DragonBeam Framework

**Untrusted Layer**
- OS
- SKM
- Application

**Secure Layer**
- Secure Memory
- SecMan
- Command/Response

**Secure Kernel Module Manager**
- Commands SKM to perform security operations
- Analyzes monitored information
- Guarantees the integrity and the liveness of SKM
**DragonBeam Framework**

- **Secure Memory**
  - Secure communication channel between SKM and SecMan
  - Only accessible by SKM or Secure Core
  - Also hosts SecMan code/data

---

The DragonBeam Framework
Example Use Case

SecMan

SKM

Time

SKM

SecMan

Secure Data Memory

The DragonBeam Framework
Example Use Case

SecMan

Sends Command

SKM

SKM

Secure Data Memory

SecMan

check_syscall_table() {
  send_cmd(CMD_SYSCALL_TABLE);
  settimer(TIMEOUT);
}
Example Use Case

SKM

SecMan

Sends Command

SKM

SecMan

check_syscall_table() {
    send_cmd(CMD_SYSCALL_TABLE);
    settimer(TIMEOUT);
}

Secure Data Memory
Example Use Case

SecMan Sends Command

SKM

SKM

skm_ISR() {
  save sp;
  move sp to secure stack;
  switch (*CMD) {
    ...
    case CMD_SYSCALL_TABLE:
      send_syscall_table();
      break;
    ...
  }
  restore sp;
}

Secure Data Memory

SecMan

check_syscall_table() {
  send_cmd(CMD_SYSCALL_TABLE);
  settimer(TIMEOUT);
}
Example Use Case

SecMan

Sends Command

SKM

Collects Information

Time

SKM

skm_ISR() {
    save sp;
    move sp to secure stack;
    switch (*CMD) {
        ...
        case CMD_SYSCALL_TABLE:
            send_syscall_table();
            break;
        ...
    }
    restore sp;
}

send_syscall_table() {
    get cur_syscall_table;
    for each entry i
        write cur_syscall_table[i];
    response_ready();
}

Secure Data Memory

SecMan

check_syscall_table() {
    send_cmd(CMD_SYSCALL_TABLE);
    settimer(TIMEOUT);
}
**Example Use Case**

**SKM**

1. SKM sends command `send_cmd(CMD_SYSCALL_TABLE);`
2. SKM collects information
3. SKM calls `check_syscall_table()`

**SecMan**

1. SecMan sends command `send_cmd(CMD_SYSCALL_TABLE);`
2. SecMan collects information
3. SecMan calls `check_syscall_table()`

**SKM**

```c
check_syscall_table() {
    send_cmd(CMD_SYSCALL_TABLE);
    settimer(TIMEOUT);
}
```

**SKM**

```c
skm_ISR() {
    save sp;
    move sp to secure stack;
    switch (*CMD) {
        ...
        case CMD_SYSCALL_TABLE:
            send_syscall_table();
            break;
        ...}
    restore sp;
}
```

**SecMan**

```c
send_syscall_table() {
    get cur_syscall_table;
    for each entry i
        write cur_syscall_table[i];
    response_ready();
}
```
Example Use Case

### The DragonBeam Framework

**SKM**
- Sends Command
- Collects Information
- Analyzes Data

**SecMan**
- Sends Command
- Analyzes Data

---

**SKM**

```c
void skm_ISR() {
    save sp;
    move sp to secure stack;
    switch (*CMD) {
        ... case CMD_SYSCALL_TABLE: 
            send_syscall_table();
            break;
        ... }
    restore sp;
}
```

**SecMan**

```c
void check_syscall_table() {
    send_cmd(CMD_SYSCALL_TABLE);
    settimer(TIMEOUT);
}
```

```c
void recv_syscall_table() {
    cleartimer(TIMEOUT);
    retrieve current syscall table;
    for each entry i
        if (cur.table[i]!=org.table[i])
            Raise alert!
}
```
Challenges

- SKM identification
- Secure memory access control
- SKM integrity and liveness guarantee

The DragonBeam Framework
SKM Registration

- Requested by SKM, verified by SecMan
  - Calculates a hash of SKM’s code
  - Directly from physical frames

The DragonBeam Framework
SKM Registration

- Requested by SKM, verified by SecMan
  - Calculates a hash of SKM’s code
  - Directly from physical frames

**SKM**

- SKM Loading
- Registration request

**SecMan**

- Find phys. frames of SKM .text

**Diagram**

- Base address
- SKM .text
- Page Size
- Virtual Address Space
- SKM Size
- Physical Frames
- Physical Address Space

The DragonBeam Framework
SKM Registration

- Requested by SKM, verified by SecMan
  - Calculates a hash of SKM’s code
  - Directly from physical frames

**SKM**

SKM Loading
Registration request

**SecMan**

Find phys. frames of SKM .text

Calculate the hash of SKM .text

Match
Not

Begin operations
Halt and alarm

The DragonBeam Framework
Secure Memory Access Control

• Who initiated memory transaction?
  • Use the current program counter and page mapping information
Secure Memory Access Control

• What if attacker modifies SKM’s page mapping?

The DragonBeam Framework
Secure Memory Access Control

- What if attacker modifies SKM’s page mapping?
Secure Memory Access Control

• What if attacker modifies SKM’s page mapping?
  • Solution: Regularly translate virt-to-phys address and verifies SKM .text hash
Heartbeat and Hashing

- **Heartbeat**
  - Checks if SKM is alive
  - Only SKM can respond
Heartbeat and Hashing

- **Heartbeat**
  - Checks if SKM is alive
  - Only SKM can respond

- **SKM .text hashing**
  - Checks if SKM’s code and page mapping have not been altered
Random Check Intervals

- To prevent TOCTTOU (Time Of Check To Time Of Use) attacks
- Attacker cannot guess the pattern of checks
Implementation

- Leon3 processor on Xilinx ZC702 FPGA
  - SPARC V8, soft-core
  - 83.3 MHz
  - 256 MB
Implementation

Leon3 on-chip SRAM (128KB)

- Leon3 processor on Xilinx ZC702 FPGA
  - SPARC V8, soft-core
  - 83.3 MHz
  - 256 MB

Leon3 Core 1 (Monitored Core)
- Instruction Pipeline
- PC
- MMU
- CTP

AHBRAM
- Secure Memory
- Controller
- Access Control

Leon3 Core 2 (Secure Core)
- Instruction Pipeline
- PC
- MMU
- CTP

AHB2AXI Bridge
- SKM
- Linux

AHB2APB Bridge
- Multiprocessor Interrupt Controller

AMBA AHB BUS

The DragonBeam Framework
Implementation

PC and CTP for Secure Memory access control

• Leon3 processor on Xilinx ZC702 FPGA
  • SPARC V8, soft-core
  • 83.3 MHz
  • 256 MB
Implementation

- Leon3 processor on Xilinx ZC702 FPGA
  - SPARC V8, soft-core
  - 83.3 MHz
  - 256 MB

Unmodified Linux 3.8

The DragonBeam Framework
Evaluation – Use Cases

1) System call table integrity check
2) Hidden module detection
Evaluation – Use Cases

1) System call table integrity check
2) Hidden module detection
Evaluation – Use Cases

1) System call table integrity check
2) Hidden module detection
Evaluation – Use Cases

1) System call table integrity check
2) Hidden module detection
Evaluation – Performance Overhead

- SPEC Benchmarks on the monitored core

![Graph showing performance overhead for SPEC benchmarks](image)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Heartbeat (100 ms)</th>
<th>SKM .text hashing (100 ms)</th>
<th>System call table check (100 ms)</th>
<th>Hidden module detection (100 ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>1.00</td>
<td>1.01</td>
<td>1.02</td>
<td>1.03</td>
</tr>
<tr>
<td>hmmer</td>
<td>1.04</td>
<td>1.05</td>
<td>1.06</td>
<td>1.07</td>
</tr>
<tr>
<td>libquantum</td>
<td>1.03</td>
<td>1.04</td>
<td>1.05</td>
<td>1.06</td>
</tr>
<tr>
<td>mcf</td>
<td>1.01</td>
<td>1.02</td>
<td>1.03</td>
<td>1.04</td>
</tr>
<tr>
<td>sjeng</td>
<td>1.00</td>
<td>1.01</td>
<td>1.02</td>
<td>1.03</td>
</tr>
</tbody>
</table>
Evaluation – Performance Overhead

- SPEC Benchmarks on the monitored core

### Average ratio of execution time to the case of No SKM

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Heartbeat (100 ms)</th>
<th>SKM .text hashing (100 ms)</th>
<th>System call table check (100 ms)</th>
<th>Hidden module detection (100 ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>1.01</td>
<td>1.03</td>
<td>1.04</td>
<td>1.05</td>
</tr>
<tr>
<td>hmmer</td>
<td>1.02</td>
<td>1.04</td>
<td>1.05</td>
<td>1.06</td>
</tr>
<tr>
<td>libquantum</td>
<td>1.03</td>
<td>1.05</td>
<td>1.06</td>
<td>1.07</td>
</tr>
<tr>
<td>mcf</td>
<td>1.04</td>
<td>1.06</td>
<td>1.07</td>
<td>1.08</td>
</tr>
<tr>
<td>sjeng</td>
<td>1.05</td>
<td>1.07</td>
<td>1.08</td>
<td>1.09</td>
</tr>
</tbody>
</table>

**In-SKM Procedure**
Evaluation – Performance Overhead

- SPEC Benchmarks on the monitored core

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Average ratio of execution time to the case of No SKM</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>1.00</td>
</tr>
<tr>
<td>hmmer</td>
<td>1.01</td>
</tr>
<tr>
<td>libquantum</td>
<td>1.02</td>
</tr>
<tr>
<td>mcf</td>
<td>1.03</td>
</tr>
<tr>
<td>sjeng</td>
<td>1.04</td>
</tr>
</tbody>
</table>

Extra features:
- Memory traffic interference
Evaluation – Performance Overhead

- SPEC Benchmarks on the monitored core

**Random order**
Random interval: [0,1,…,199,200] ms
Back-to-back with prob. of 5%
Evaluation – Hardware Cost

The DragonBeam Framework
Evaluation – Hardware Cost

<table>
<thead>
<tr>
<th>Resource</th>
<th>Original</th>
<th>W/ DragonBeam</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>10258</td>
<td>10356</td>
<td>0.96%</td>
</tr>
<tr>
<td>Look-up Tables</td>
<td>19482</td>
<td>19511</td>
<td>0.15%</td>
</tr>
</tbody>
</table>
Extension to Multiple Monitored Cores

- Extended to quad-core
- Works only for SMP
  - Single SKM
Extension to Multiple Monitored Cores

- Extended to quad-core
- Works only for SMP
  - Single SKM

Extension for N-1 monitored cores
Extension to Multiple Monitored Cores

- Extended to quad-core
- Works only for SMP
  - Single SKM

<table>
<thead>
<tr>
<th>Resource</th>
<th>Original</th>
<th>W/ DragonBeam</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Core Registers</td>
<td>10258</td>
<td>10356</td>
<td>0.96%</td>
</tr>
<tr>
<td>Quad Core Look-up Tables</td>
<td>19482</td>
<td>19511</td>
<td>0.15%</td>
</tr>
<tr>
<td>Quad Core Registers</td>
<td>18932</td>
<td>19029</td>
<td>0.51%</td>
</tr>
<tr>
<td>Quad Core Look-up Tables</td>
<td>37777</td>
<td>37835</td>
<td>0.15%</td>
</tr>
</tbody>
</table>
Conclusion

• **DragonBeam Framework**
  - HW/SW framework for trusted security monitoring
    - Bootstrap trust into SW layer from trusted HW
    - Multicore-based
  - Expanded observability due to in-place monitoring
    - Secure “Kernel Module”
  - Allows for customized security modules to system developers
Thank you