Designing and Implementing Malicious Hardware

Samuel T. King, Joseph Tucek, Anthony Cozzie, Chris Grier, Weihang Jiang, and Yuanyuan Zhou

Presented by Lauren Biernacki and Shuang Qiu
Background

SoC Supplier

Fabrication

Packaging

Testing

Assembly

Integrated Circuit (IC) Supply Chain
Previous Work

- IBM has developed a trojan circuit that can leak encryption keys using 406 additional gates.

- These trojan circuits are hardcoded modifications to orchestrate simple, specialized attacks on the underlying hardware.

- No research has been conducted on hardware modifications that can support multiple types of software based attacks.

**Novel idea:** Design and implement general purpose hardware to support the design of software based attacks.
Motivation & Goals

**Memory Access Mechanism:** allows us to bypass the memory management unit
- Privilege Escalation Attack

**Shadow Mode:** allows us to execute invisible malicious firmware
- Login Backdoor
- Stealing Passwords
Motivation & Goals

- **Visibility**: Whether or not evidence of the attack appears on the data or address bus

- **Flexibility**: The hardware design can support various software payloads

- **Timing Perturbations**: Reduce the performance impact the modification has on the processor
Hardware Design: Memory Access Mechanism

Microprocessor

- CPU
- MMU
- MA Snoop
- TLB

D-Cache
I-Cache

Main Memory

Address Bus
Data Bus
Hardware Design: Memory Access Mechanism

Microprocessor
- CPU
- MMU
- MA Snoop
- TLB
- Protection Checking Disabled

Main Memory

D-Cache
I-Cache

Address Bus
Data Bus
Magic Bytes
Hardware Design: Memory Access Mechanism

- **Visibility**: It is visible! Memory accessed when protection checking is disabled still appears on the bus.

- **Flexibility**: Gives us the ability to alter any memory, including that belonging to the operating system.

- **Timing Perturbations**: These modifications do not influence performance.

*Requires attacker to already have software running on the system in order to trigger byte sequence.*
Hardware Design: Shadow Mode

- Addresses visibility issue by reserving instruction and data cache lines specifically for the malicious process

- Uses software to initiate the attack, supported by hardware alterations
  - Bootstrap code is used to initialize the attack
  - Monitors for a predefined trigger, which initiates malicious firmware

- Details of the bootstrap attack code depend on assumptions about the machine
Hardware Design: Shadow Mode
Hardware Design: Shadow Mode

Microprocessor

- CPU
- Debugging Logic
- TLB
- MMU
- Data Bus
- Address Bus

Memory

- D-Cache
- I-Cache
- Main Memory
- UDP Header
- Firmware
- Magic Bytes
- Boots..
Hardware Design: Shadow Mode

Microprocessor

CPU

MMU

TLB

D-Cache

I-Cache

Main Memory

Address Bus

Data Bus
Hardware Design: Shadow Mode

Microprocessor

- CPU
- MMU
- TLB
- D-Cache
- I-Cache
- Debugging Logic

Main Memory

Address Bus
Data Bus

Boots...
Firmw...

Hardware Design: Shadow Mode
Hardware Design: Shadow Mode

- **Visibility**: As long as malicious firmware does not access main memory, the attack not visible outside the processor

- **Flexibility**: Supports “nearly arbitrary” attacks

- **Timing Perturbations**: Partitioning the cache does have performance ramifications that depend on how long Shadow Mode runs
Attack: Privilege Escalation

- Memory access mechanism
  - Trojaned hardware turns off memory protection.

Malicious Program
Trojaned Hardware

Effective user ID: \textit{euid1}
Attack: Privilege Escalation

- Memory access mechanism
  - The program changes its effective user ID to root.
  - The program now runs with full system privileges.

![Diagram showing memory access mechanism with Trojaned Hardware, Malicious Program, and Kernel memory highlighted with effective user ID 'euid1' changing to 'root'.]
**Attack: Login Backdoor**

- **Shadow mode mechanism - Transient**
  - Attacker sends unsolicited UDP packet
  - Monitor notices the magic byte sequence
  - Target OS inspecting UDP packet triggers trojaned hardware
Attack: Login Backdoor

- **Shadow mode mechanism** - Transient
  - Firmware is copied to reserved cache area and activated
  - Attacker logs in as root.
  - Shadow firmware uninstalls automatically.

- Monitor login application
- Uname: root, pwd: letmein
- Make pwd checking return True
Attack: Stealing Passwords

- **Shadow mode mechanism** - Persistent
  - Keep interposing on the *write* and *read* library call to steal password

1. Interpose on *write* call, searching for “Password:” to identify process receiving passwords
2. Record potential passwords on the following *read* call
Attack: Stealing Passwords

- **Shadow mode mechanism** - Persistent
  - Use two techniques to leak password out

![Diagram showing processor with D-Cache, I-Cache, Boots, and Malicious Service]

- Directly use system network call
- Overwrite existing network packet

Password: 12345
Evaluation

- **Circuit-level perturbations**
  - Implemented on FPGA development board with Leon3 processor
  - Modify the processor at the VHDL level
  - Memory access
    - Modify data caches & MMU
    - Memory permission checks are ignored for malicious software
  - Shadow mode
    - Modify instruction and data caches
    - Add new watchpoints and make minor changes to the existing watchpoints

<table>
<thead>
<tr>
<th>Processor</th>
<th>Logic gates</th>
<th>Logic gates increment w.r.t. Baseline CPU</th>
<th>Lines of VHDL codes</th>
<th>VHDL code increment w.r.t. Baseline CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline CPU</td>
<td>1,787,958</td>
<td>--</td>
<td>11,195</td>
<td>--</td>
</tr>
<tr>
<td>CPU + memory access</td>
<td>1,788,917</td>
<td>959 (0.05%)</td>
<td>11,263</td>
<td>68</td>
</tr>
<tr>
<td>CPU + shadow mode</td>
<td>1,789,299</td>
<td>1341 (0.08%)</td>
<td>11,312</td>
<td>117</td>
</tr>
</tbody>
</table>

Table is from the paper “Designing and implementing malicious hardware”
Evaluation

● **Timing perturbations**
  ○ Various benchmarks
    ■ Four CPU bound benchmarks: bzip2, gcc, parser, and twolf
    ■ One I/O bound benchmark: wget
  ○ Four experimental cases (Login backdoor attack)
    ■ Baseline: Unmodified hardware and without attacking
    ■ Known Root: Unmodified hardware. Log in with root password and steal the /etc/shadow file.
    ■ Transient: Hardware with shadow mode support. “Hit-and-run” style attack.
    ■ Persistent: Hardware with shadow mode support. Continuously active login backdoor.
Defense Strategies

- Detecting via analog perturbations
  - Power analysis
    - Countermeasure: constant power draw circuits

- Detecting via digital perturbations
  - IC testing with various inputs and outputs
    - Countermeasure: wait for a specific sequence as a trigger
  - Reverse engineering
    - Time-consuming, expensive, destructive
  - Fault-tolerance techniques
    - Hardware redundancy (3m+1 ICs are needed to cope with m malicious ICs) [1]

- Each single approach is completely ineffective.
- Malicious hardware defense is a potential research direction.

Conclusions

- This paper has laid the groundwork for constructing malicious processors.
- Proposed Two mechanisms: memory access and shadow mode.
- Implemented 3 attacks: privilege escalation, login backdoor, stealing password.
- Few hardware modification with less possibility of detection.
- Malicious hardware defense is a potential research direction.
Discussion Points

- Are these attacks truly “invisible”?
- Should these malicious processors be used over standard Trojan circuits for an attack?
- It seems that all the possible defenses are not feasible. Are there any other potential defense strategies?
- Is this threat realistic?