Hardware Support for Malware Defense and End-to-End Trust

IBM Research
Dimitrios Pendarakis
Research Staff Member and Manager, Secure Systems Group
dimitris@us.ibm.com

Date: September 18, 2013
Team Profile

• Team make-up
  – Sameh Assad
  – Rick Boivie
  – Eknath Ekanadham
  – Kenneth Goldman
  – Eric Hall
  – Guerney Hunt
  – Mohit Kapur
  – Dimitrios Pendarakis, PI
  – David Safford

• Group has a long history of research leadership and transition into products, standards and open source in areas:
  – Operating systems, networking systems, NSFNET
  – Network security protocols, network scalability
  – Secure co-processors like the IBM 4758, 4764, …
  – Trusted Computing and Linux Security
  – Secure Processors
Customer Need

Despite increased investment in security, cybersecurity attacks are increasing.

Need to protect all computing infrastructure: servers, mobile platforms, embedded and cyber-physical systems.

Source: IBM X-Force® Research and Development
Technical Approach: Problem

- **Software verification hard, especially for large code bases**
  - Trusted Computing Base (TCB) in modern systems is large and increasing
    - Firmware, OS, hypervisor, JVM, …
  - Increasingly networked devices (e.g., control systems) multiply risks

- **Objective: hardware trust anchors that allow TCB size reduction**
  - **End-to-End applicability:** from low-end embedded to cloud servers
  - **Trust:** Protect secrets, confidentiality & integrity of code and sensitive data
  - Introduce techniques to **containerize sensitive applications/software**
    - Protect “applications” from each other, including OS/Hypervisor, firmware, privileged software, etc.
  - Perform continuous monitoring and analysis for anomaly detection

- **Technology deployment considerations**
  - Minimize required enhancements in software stacks to utilize new capabilities
    - Allow existing applications to run unmodified
  - Cost effective introduction of hardware and firmware changes
Application to Server Environment: Secure Blue++ w. Access Control Monitor

• Observation: large number of attacks involve unauthorized access of an applications’ memory and/or files
  – While application is “at rest”, in the file system and throughout execution
  – Hence, it is important to control access to memory and (non-volatile) storage resources: through both access control and cryptographic means

• We are developing a subsystem to enforce isolation and controlled sharing across the end-to-end lifecycle
  – Secure Blue++ w. Access Control Monitor (SB++/ACM)
  – Applicable at different granularities: processes, VMs, …

• Objective: subsystem contains just enough controls to
  – Provide “isolation” of data for secure processes
  – Provide “sharing” of data when permitted by the owner and policy
  – Require no new address translations or changes to resource scheduling
  – Without trusting other subsystems (OS or Hypervisor)
  – Maintain backwards compatibility with “non-secure” processes
Security Domains are disjoint collections of processes; in particular all OS processes could be in one domain.

Must enforce data Isolation and Sharing across domains ("data" connotes both instruction and data pages)

- Private data in one domain cannot be accessed by another domain
- Designating and sharing of specific data across domains is permitted by explicit sharing-control primitives

OS continues to do resource scheduling (CPU and memory) for all processes in all domains, and OS can employ conventional techniques of address mapping to accomplish isolation and sharing of data/inst among processes within a domain.
### Functionality requirements

- Measure firmware (prevent supply chain attacks...)
- Lock firmware (protect from online modification)
- Safely update firmware (physical presence)
- Authenticate firmware ("secure boot")

### Vendor constraints

- Zero additional cost
- Zero additional switches/buttons

<table>
<thead>
<tr>
<th>Device</th>
<th>Measure BIOS</th>
<th>Protect BIOS</th>
<th>Signed/Local Updates</th>
<th>Secure Boot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pogoplug</td>
<td>Yes - SATA</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D-Link DIR-505</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TP-Link MR3020</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Linksys WRT54G</td>
<td>Yes - JTAG</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
# BIOS Integrity with Zero Cost Modifications

<table>
<thead>
<tr>
<th>Device</th>
<th>Measure BIOS</th>
<th>Protect BIOS (lock firmware)</th>
<th>Signed/Local Updates (physical presence)</th>
<th>Secure Boot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pogoplug</td>
<td>Yes - SATA</td>
<td>Yes - HPM</td>
<td>Yes – power latch</td>
<td>Yes – RSA signature</td>
</tr>
<tr>
<td>D-Link DIR-505</td>
<td>Yes - buspirate</td>
<td>Yes - HPM</td>
<td>Yes – switch or button</td>
<td>Yes – RSA signature</td>
</tr>
<tr>
<td>TP-Link MR3020</td>
<td>Yes - buspirate</td>
<td>Yes - HPM</td>
<td>Yes - button</td>
<td>Yes – RSA signature</td>
</tr>
<tr>
<td>Linksys WRT54G</td>
<td>Yes - JTAG</td>
<td>Yes - HPM</td>
<td>Yes - button</td>
<td>Yes – RSA signature</td>
</tr>
</tbody>
</table>

* HPM: Hardware Protection Mode locking
Benefits

• Protect sensitive applications and data from “other” software, including OS, hypervisor & malware w. root privileges
• Allow applications to be deployed w. pre-configured secrets
• Can help secure cloud computing environments
• Backward compatibility w. existing application software
• Fairly limited impact on hardware/firmware and required software stack updates
• Integration of new containerization technologies w. Trusted Computing Architecture
• Very low-cost platform modifications can protect the integrity of both embedded Linux devices and sensors/actuators
  – “Trust Dust” demonstration for sensors and actuators shown separately
• **Current Hardware – based approaches**
  – Dedicated hardware crypto cards are expensive and harder to use with general applications/VMs – typically used for “crown jewels”
  – Trusted computing verifies provenance of system software, but does not protect from vulnerable and malicious software

• **Software – only approaches**
  – Cannot provide the same level of trust as hardware-based security
    • Rootkits and stealth malware can evade & subvert security software
    • Supply chain concerns make trusting platform and firmware hard
  – Malware signatures (blacklists) do not scale well, cannot keep up with attackers
    • Number of types of malware proliferating
    • Self modifying malware code may make it extremely hard to look for “signatures”
• **Accomplishments so far**
  – Embedded security prototypes and technical papers
    • "*Embedded Linux Integrity*", David Safford, to be presented at Linux Security Summit (LSS), September 19-20 2003, New Orleans, USA
    • “Trust Dust” prototype demonstrated in the PI meeting
  – Proposed architecture for enhanced server security that combines SecureBlue++ with Access Control Monitor (ACM) and trusted computing
    • Corresponds to deliverable for end-to-end architecture at 12 months from project start
  – Simulation environment for testing proposed processor changes in progress
Next Steps

- Complete architecture for enhanced server security
  - Including initial evaluation of both hardware complexity and software impact
- Experimentation w. enhanced server architecture performance and effectiveness using simulator environment
- Development of a secure mobile environment prototype
- Technology Transition Activities?
  - Exploring commercialization of technologies within IBM and/or partners
  - Advance relevant standards
    - In particular in the embedded security space
Contact Information

Principal Investigator and Programmatic Technical Point of Contact:
Dimitrios Pendarakis, PhD
Research Staff Member and Manager, Secure Systems Group
914-945-1286, dimitris@us.ibm.com

Administrative Contact:
Tyrone Worsley, Jr.
914-945-1363, tjworsley@us.ibm.com