Policy Enforcement and Compliance Proofs for Xen Virtual Machines

Bernhard Jansen (IBM Zurich Research Laboratory)
HariGovind Ramasamy (IBM T. J. Watson Research Center)
Matthias Schunter (IBM Zurich Research Laboratory)
Broader Problem: Security Management of VMs

- **Complicating Factors**
  - VMs on same physical host may belong to diverse organizations
  - VMs on same physical host may have different security requirements
  - Weaker isolation boundary between VMs than physical machines
  - Security verification & confidentiality may be opposing goals

- **Our focus is on Integrity Management**
  - Protecting VM security policies from modification throughout VM life-cycle
  - Verifying that a VM is compliant with specified security requirements
Contributions

- **Formal model for generalized integrity management of VMs**
  - Extends TPM (Trusted Platform Module) based mechanisms to cover
    - VMs and virtual devices
    - Wider and finer-grained policies
    - any policy that can be expressed as predicate on system log entries

- **PEV (Protection, Enforcement, and Verification) Integrity Architecture**
  - Extensibility: verify compliance even if new virtual devices are added to VMs
  - Flexibility: user can specify which aspects of system’s integrity state are of interest
    - obtains only information corresponding to those aspects for compliance verification
  - Blinding technique for enforcing access restrictions to the system integrity state

- **Xen-based prototype**
Outline

- Formal model for generalized integrity management
- PEV Integrity Architecture & Associated Protocols
- Xen-based prototype
- Use-cases
Trusted Platform Module: Background

- Tamper-evident, smartcard-like hardware chip
- Hardware implementation of multiple roots-of-trust
  - Root-of-trust (assumption): always behaves as expected; trusted by local & remote parties
  - E.g., root-of-trust for storage, for integrity reporting, for integrity measurement
- 160-bit Platform Configuration Registers (PCRs)

Operations
- Extension: Replace content of PCR with hash (old PCR content + input)
- Measurement: Compute the SHA-1 hash of the binary code of a component
- Attestation: Challenge-response style cryptographic protocol
  - Allows a remote party to query the recorded platform measurement values (stored in PCRs)
  - Allows the platform to reliably report the requested values
    - Using the TPM_Quote command to obtain signature on PCR value
- Sealing: Bind data item to PCR values
  - Ensure that a sensitive data item is accessible only under certain platform config.
- Unsealing: Reveals specified data item only if current PCR value(s) = PCR value(s) at sealing time
Formal Integrity Model

- Secure Write-Only Log Repository
  - Log data contains integrity state of the system

- Log contents stored in a tree structure, called log tree
  - One node for each system component
  - Node = (id, type, vector of log values)
  - Extensible

- Secret Repository for conditional release of secrets
  - Store sensitive data item and associated condition, such that the data item is released only if the log data satisfies condition

- Integrity Requirements
  - Expressed using predicates & projections on log data
  - Projection function, p(T)
    - returns a subset of the tree nodes and a subset of the vector elements from the log vector of each node
    - models specific aspects of the system’s integrity state that is of interest to user/verifier
  - Predicate, \( \Pi \)
    - used for specifying integrity conditions
Formal Integrity Model: Generalized Sealing to Protect Integrity

- TPM-Based Solution
  - Sealing: encrypt data item, tie it to values in specified subset of PCRs
  - Unsealing: decryption only if those PCRs have same values as those at sealing time

- Generalization to protect integrity of a data item
  - by making it inaccessible unless specified integrity conditions hold

- Sealing Operation
  - Input: data item, projection $p(\cdot)$, sealing predicate $\Pi$, public part of encryption key
  - Action: Log both $p(\cdot)$ and $\Pi$, then encrypt data item using key

- Unsealing Operation
  - Input: encrypted data item, log tree
  - Output: reveal data item, only if $\Pi(p(T))$ holds
Formal Integrity Model: Generalized Attestation to Verify Integrity

- **TPM-based solution**
  - query and obtain values in a specified subset of PCRs

- **Generalization**
  - Verifier specifies which aspects of system’s integrity state are of interest
    - Via attestation predicate $\Pi$, and projection function $p()$
    - Input: challenge $c$, $\Pi$, $p()$, secret key $K$
    - Output: $\text{sign}_K[\Pi(p(T)), c]$

- **Specialization**
  - to obtain TPM-based binary attestation
    - predicate is simply the identity function
    - projection function specifies which subset of PCRs to use
  - property-based attestation
    - predicate extracts high-level properties from the result of $p(T)$
PEV Integrity Architecture: Key Elements

- **Central Integrity Manager**
  - Implements one master plug-in module for each 
    \( p \ (T) \)
  - Invokes appropriate component plug-in modules and aggregates their output

- **Component Integrity Managers**
  - One per each system component that is of interest from an integrity perspective
  - Manages part of the log tree corresponding to the component
  - E.g., Storage Integrity Manager is responsible for storage sub-tree of the log tree
  - Contains component plug-in modules that implement log projection functions
  - Mapping between plug-in identifier and plug-in functionality is made public

**Flexibility**: plug-in modules to encode arbitrarily complex log projection functions

**Extensibility**: integrity state of newly added VMs/virtual devices can be covered by adding new plug-in modules to TCB
PEV Integrity Architecture: Sealing-Unsealing Protocol

- **Input**
  - Data item to be encrypted
  - Key (public part is used for encryption; private part is revealed only if unsealing is successful)
  - Identifiers of plug-ins (which implement the log projection function)
    - Which aspects of the system's integrity state are of interest to the user and need to be verified
    - Predicate
      - Condition for revealing the private part of sealing key

- **Key sealed against [ state of TCB + hash(plug-in IDs + predicate) ]**
  - State of TCB stored in non-resettable PCRs
  - Hash stored in resettable PCR, which ensures that
    - TCB is aware of conditions to be satisfied before revealing key during unseal operation
    - these conditions are not changed prior to unsealing

- **Unseal operation**
  - Stage 1
    - TPM reveals key to Integrity Manager if state of TCB is unchanged
    - TCB checks whether resettable PCR still contains hash
  - Stage 2
    - Integrity Manager invokes Predicate Evaluator
    - if predicate holds, Integrity Manager reveals key to user
Xen: Background

- Domains = running instances of VMs
- Dom0
  - the first domain created
  - a special domain that controls other domains, called DomUs or user domains
- Device virtualization
  - For a given physical device, the native device driver is part of at most one VM
  - VM with native device driver exports a back-end driver
  - Any VM that wants to share the device exports a front-end (virtual) device driver
  - Front-end driver connected to back-end driver through device channels
Xen-based Prototype

- **Compartment Manager (CM)**
  - VM life cycle management
  - orchestrator of IM and SVDM

- **Integrity Manager (IM)**
  - Storage Integrity Plug-In (SIP)
    - measures disk images, files
  - Attestation & Sealing Module (ASM)
    - performs sealing, attestation protocols
    - invokes TPM operations

- **Secure Virtual Device Manager (SVDM)**
  - Managing virtual devices
  - Consists of two specialized low-level component plug-in modules
    - one for virtual encrypted hard disk
    - one for virtual network interface card

---

**Integrity Model PEV Architecture Use Cases**

<table>
<thead>
<tr>
<th>Model</th>
<th>Xen-based Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projection p()</td>
<td>component measurement plug-in</td>
</tr>
<tr>
<td>Predicate II</td>
<td>XSLT stylesheet</td>
</tr>
</tbody>
</table>

---

VEE 2008, Seattle
3/13/2008
© 2008 IBM Corporation
Use Case 1: Attestation the Status of a VM Disk Image

- **Attestation Descriptor**
  - XML structure which tells which log projection function to choose
  - contains measurement descriptor(s)
  - what is to be measured?
    - which measurement plug-in modules to invoke?
    - e.g., VM disk image
  - where to store measurement results?
    - e.g., PCR 16

- **attestationResponse**
  - TPM_Quote result
  - relevant log files

- Verifier can check result
  - recompute hash over the relevant log files
  - compare result with PCR value in attestationResponse
Use Case 2: Compliance Proof for Correct Network Configuration

Goal: Validate the Config. Of Virtual Networking Sub-system on a given Host in the Customer Network

- Similar sequence of steps as in Use Case 1.
- Instead of invoking the SIP, the IM invokes the network measurement plug-in.
- Plug-in outputs measurement results as XML structure
  - Indicates the flow control policy in place
- Compliance proof = attestation of [ TCB + plug-in output ]
- Verifier uses XSLT stylesheet on XML output of network measurement plug-in to validate that the network policy at the host reflects the desired network topology

Desired Virtual Network Topology

XML Representation of Flow Control Policy

```
<flow-policy>
  <zone id="customer-net">
    <permit id="mgmt-net"/>
    <permit id="dmz"/>
  </zone>
  ...
</flow-policy>
```

XSLT Stylesheet Encoding Conditions to be Checked (used at verifier)

```
<xsl:template match="/flow-policy/zone[@id='customer-net']">
  <xsl:choose>
    <xsl:if test="count(*[@id='dmz'])=1 and count(*[@id='mgmt-net'])=1">
      <true/>
    </xsl:if>
  </xsl:choose>
</xsl:template>
```
Summary

- Introduced a formal model that generalizes TPM-based integrity management
  - Log data stored in extensible tree structure
  - Projection functions allow choosing specific aspects of the system’s integrity state
  - Predicates allow encoding arbitrarily complex integrity conditions

- PEV integrity architecture for policy enforcement and compliance checking
  - Flexibility: plug-in modules to encode arbitrarily complex projection functions
  - Extensibility: integrity state of newly added VMs/virtual devices can be covered by adding new plug-in modules to TCB

- Xen-based prototype
  - demonstrated capabilities using multiple use-cases