A Virtualized Linux Integrity Subsystem for Trusted Cloud Computing

Stefan Berger
Joint work with: Kenneth Goldman, Dimitrios Pendarakis, David Safford, Mimi Zohar

IBM T.J. Watson Research Center
Outline

• Motivation
  – Trusted Computing and Cloud Computing
  – Cloud Computing Security Concerns

• Technologies
  – Key Contributions
  – VM and Virtual TPM Lifecycle
  – IMA, IMA-Appraisal & EVM
  – Cloud Use Cases

• Virtualization Architecture and Components
  – QEMU and SeaBIOS extensions
  – Architecture and screenshots
  – Libvirt management stack

• Conclusion
Motivation

• Last few years
  – Trusted Computing support available nearly everywhere
  – Virtualization became mainstream
  – Rise of cloud computing

• Cloud Computing
  – Your computations on someone else’s infrastructure
  – Your data located on someone else’s storage
  – Concerns:
    • Trusted provider?
    • Infrastructure correctly configured and trustworthy?
    • Trusted Images?
    • Data theft by third parties?
    • Virtualization software and cloud vulnerabilities?
Cloud Computing
Fundamental Challenges from Security Point of View

• What is unique about cloud computing security?
  – Loss of physical ownership – “technological, cultural and psychological issue”
    • Redefines boundaries of IT infrastructure, redefines “insider attacks”
  – Scale; multiple tenants, misconfiguration

• Need for strong, provable, customer/workload isolation

• Dependency on type of user and sensitivity of applications/data
  – Customers: enterprise, government/defense, academic, consumers
Outline

• Motivation
  – Trusted Computing meets Cloud Computing
  – Cloud Computing Security Concerns

• Technologies
  – Key Contributions
  – VM and Virtual TPM Lifecycle
  – IMA, IMA-Appraisal & EVM
  – Cloud Use Cases

• Virtualization Architecture and Components
  – QEMU and SeaBIOS extensions
  – Architecture and Screenshots
  – Libvirt management stack

• Conclusion
Key Contributions of Our Work

• First instance of practical demonstration of virtualized trusted computing stack
• Application of this stack in an open source hypervisor (KVM/QEMU)
VM and Virtual TPM Lifecycle

VM Image Authoring → Image repository → vTPM Authoring → VM Image Deployment → VM Runtime

VM migration with vTPM

libvirt

Hypervisor

PM

Mgmt

VM

vTPM

Hypervisor

PM

Mgmt

VM

vTPM
Virtual TPM Lifecycle - Functions

vTPM Authoring

• Performed before VM’s first start
  • Create vTPM
    • Create EK
    • Initialize NVRAM Areas
    • Add Certificates

VM Deployment And Start

• Command line support for QEMU for adding a vTPM
  • Optional AES encryption key argument for vTPM’s persistent state
  • libvirt XML extensions

VM Runtime

• VM + vTPM suspend/resume and snapshotting
  • Freeze state of VM and resume at later time
  • Save vTPM state into QCoW2 (encrypted) storage
  • VM + vTPM Migration
Linux Integrity: IMA and EVM

- Linux Integrity Components and Concepts
  - IMA
    - **Collect** – measure a file before it is accessed (read and hash).
    - **Store** – add the measurement to a kernel resident list and, if a hardware Trusted Platform Module (TPM) is present, extend the IMA PCR
    - **Attest** – if present, use the TPM to sign the IMA PCR value, to allow a remote validation of the measurement list.
vBIOS and IMA Measurements

```bash
$ su -c 'head -10 /sys/kernel/security/tpm0/ascii_bios_measurements'

<table>
<thead>
<tr>
<th>PCR</th>
<th>template-hash</th>
<th>filedata-hash</th>
<th>filename-hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bd19c5f7e66fe93038474bfcc75a17687320c71c5</td>
<td>[SMBIOS]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>89f75dfe4b7e3c37c49132ca2163e2368fa92195</td>
<td>[Option ROM]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9dbd87163112e5670378abe4510491259a61f411</td>
<td>[Start Option ROM Scan]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3821b4b258ba34f18fc3ee16b4f6fblec9178569</td>
<td>[Option ROM]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5ce3dbb0ba4e912eab297118b9e4e961e87df0fd</td>
<td>[Option ROM]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>cle25c3f6b0dc78d57296aa2870ca6f782ccf80f</td>
<td>[Calling INT 19h]</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>d9be6524a5f5047db5866813acf3277892a7a30a</td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>d9be6524a5f5047db5866813acf3277892a7a30a</td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>d9be6524a5f5047db5866813acf3277892a7a30a</td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>d9be6524a5f5047db5866813acf3277892a7a30a</td>
<td>[]</td>
<td></td>
</tr>
</tbody>
</table>

$ su -c 'head -5 /sys/kernel/security/ima/ascii_runtime_measurements'

<table>
<thead>
<tr>
<th>PCR</th>
<th>template-hash</th>
<th>fileddata-hash</th>
<th>filename-hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7971593a7ad22a7cce5b234e4bc5d71b04696af4</td>
<td>ima b5a166c10d153b7cc3e5b4f1eab1f71672b7c524</td>
<td>boot_aggregate</td>
</tr>
<tr>
<td>10</td>
<td>2c7020ad8cab6b7419e4973171c704b6bf52f77</td>
<td>ima e09e048c48301268ff38645f4c006137e42951d0</td>
<td>/init</td>
</tr>
<tr>
<td>10</td>
<td>ef7a0aff83dd46603ebd13d1d789445365adb3b</td>
<td>ima 0f8b3432535d5eab912ad3ba744507e35e3617c1</td>
<td>/init</td>
</tr>
<tr>
<td>10</td>
<td>247da6fc82b346803660382d197c019243e59f</td>
<td>ima 747acb096b906392a62734916e0bb39cef540931</td>
<td>ld-2.9.so</td>
</tr>
<tr>
<td>10</td>
<td>341de30a46fa55976b26e55e0e19ad22b5712dc8</td>
<td>ima 326045fc3d74d8c8b23ac8ec0a4d03fdac9618a</td>
<td>ld.so.cache</td>
</tr>
</tbody>
</table>
```
Linux Integrity: IMA and EVM

- Linux Integrity Components and Concepts
  - IMA Appraisal Extension
    - **Appraise**
      - validation of a measurement against a “good” value
      - value stored in a security extended attribute of the file (Security.ima)
      - refuse to read / execute file if value is not “good”
      - possible to update modified files on file close
      - requires filesystem to be ‘labeled’
EVM: Cryptographic binding of security attributes

- **Goal** – cryptographic protection of security extended attributes
  - security.selinux
  - security.SMACK64
  - security.capability
  - security ima

- Symmetric key HMAC is used to bind attributes and other file metadata
  - Inode number
  - Owner, Group
  - Mode

- Protects all of this metadata against off-line attack
  - The signing key is released by the TPM only if trusted kernel booted
  - Even someone with physical access cannot get key and forge HMAC
Trusted/Encrypted Keys - Concept

• Existing TPM-based RSA keys:
  – Created inside TPM with hardware random number generator
  – Private key is never visible in plaintext outside TPM
  – Usage can depend on state of PCRs

• Existing TPM-based Symmetric keys:
  – Created in user space – potentially visible in plaintext to adversary
  – Submitted to TPM through kernel for sealing/unsealing

• Solution: Trusted and Encrypted key types
  • **Trusted key type**: unsealed by TPM at boot time, only if boot PCR's match
    • User space sees only encrypted blob
  • **Encrypted key type**: kernel managed, encrypted by master Trusted key
    • User space sees only encrypted blob
Cloud Use Cases

• Integrity Measurement Architecture (IMA):
  – Attestation of hosts by cloud provider: Provider to check for trusted executables
  – Attestation of VMs by cloud users: Check trustworthiness of VM image

• Extended Verification Module (EVM) + IMA-Appraisal:
  – VM Image: Have my data been tampered with?

• vTPM + Trusted/Encrypted keys
  – Filesystem encryption

• PTS: vTPM + IMA
  – Enforce trusted network connectivity
Outline

• Motivation
  – Trusted Computing meets Cloud Computing
  – Cloud Computing Security Concerns

• Technologies
  – Key Contributions
  – VM and Virtual TPM Lifecycle
  – IMA, IMA-Appraisal & EVM
  – Cloud Use Cases

• Virtualization Architecture and Components
  – QEMU and SeaBIOS extensions
  – Libvirt management stack

• Conclusion
x86 QEMU/KVM Components

• QEMU vTPM
  – IBM’s Open-source vTPM packaged as shared library: libtpms
• vTIS
  – TIS Hardware Interface Emulation
  – Interfaces with libtpms
• SeaBIOS Extensions
  – vTPM vTIS driver
  – vTPM initialization
  – User Menu
  – Interrupt Handler : API calls
  – ACPI support : ACPI tables, S3 resume
  – SRTM support
x86 QEMU/KVM + vTPM Architecture
x86 QEMU/KVM + vTPM Architecture
x86 QEMU/KVM + vTPM Architecture

Linux Host

QEMU
vHardware
vTIS
vTPM (libtpms)
vCPU
vVGA
vPIC
vPIT
vRTC
vAPIC
vHPET

Software

TPM TIS
driver

EVM
IMA
TSS

KVM

BIOS
TCG-BIOS extensions

Hardware

TIS
TPM

HW TPM
x86 QEMU/KVM + vTPM Architecture

QEMU

vHardware

EVM
IMA TSS
TPM TIS
driver

vTIS
vTPM (libtpms)

vCPU vVGA vPIC vPIT vRTC vAPIC vHPET ...

vBIOS TCG-BIOS extensions

Linux Host

EVM
IMA TSS
TPM TIS
driver

KVM

BIOS TCG-BIOS extensions

Hardware

TIS

TPM

HW TPM
Nested Virtualization: x86 QEMU/KVM + vTPM Architecture

**QEMU**
- **vHardware**
  - vTIS
  - vTPM
- EVM
  - IMA
  - TSS
  - TPM TIS driver

**QEMU**
- **vHardware**
  - vTIS
  - vTPM
- EVM
  - IMA
  - TSS
  - TPM TIS driver

**Software**
- vBIOS (TCG-BIOS extensions)
- TSS

**Linux Host**
- BIOS (TCG-BIOS extensions)
- KVM
- Hardware
  - TIS
  - TPM

**Guest OS**
- vBIOS (TCG-BIOS extensions)
- EVM
  - IMA
  - TSS
- vCPU
  - vVGA
  - vPIC
  - vRTC
  - vAPIC
  - vTIS
  - vTPM
  - vRTC
  - vAPIC
  - vTIS
  - vTPM
  - vHPET
  ...
TCG support in SeaBIOS

- SeaBIOS with TPM support menu
Linux sees ACPI tables created by SeaBIOS

TPM device
SSDT
TCPA for Logging
TPM device entry in Linux procfs

```
[root@TPM-VM-FC14 ~]# ls
active caps enabled id options pcrs pubek subsystem timeouts
cancel driver firmware node misc owned power resources temp deactivated uevent

[root@TPM-VM-FC14 ~]# cat pcrs

PCR-00: 3A 3F 76 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-01: A7 DD 3C A5 60 7F D1 65 AD B4 F2 FC 74 28 C1 B7 80 C9 1A 1C
PCR-02: 30 6A F3 83 9B 8C 73 96 56 87 9C 11 B8 B8 AB 2C 19 44 E3 6D
PCR-03: 3A 3F 76 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-04: E2 E8 E4 D1 67 1A B1 B9 1E 7B 8B 83 35 98 59 78 96 C9 75 18
PCR-05: 6A 56 64 DC 1E 3A 17 D3 E1 A1 6D AC 8D 49 F0 47 26 38 B6 8B
PCR-06: 70 FA FA 72 54 71 FE 9A 87 16 C6 64 94 3F 36 3D 89 A7 82 A3
PCR-07: 3A 3F 78 0F 11 A4 B4 99 69 FC AA 80 CD 6E 39 57 C3 3B 22 75
PCR-08: D2 29 50 A1 01 8A 4A 8F 29 24 A4 64 95 E1 80 49 99 19 74 BD
PCR-09: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-10: BF D4 91 C4 D7 52 3E LC DE DA CD 55 56 0A 3F 85 9E 0A 59 C7
PCR-11: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-12: 45 40 0E 00 30 00 03 03 2E 21 7A E5 20 80 3C 56 AF FB E6 91
PCR-13: FC CC CF 7E F7 7A DB 88 48 55 64 45 DE 6A F8 5A AA 7C 82 81
PCR-14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-15: 9E 79 36 9E E9 73 DD BB EC C0 E4 A3 38 BC 61 B5 AD 98 8A EB
PCR-16: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PCR-17: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-18: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-19: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-20: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-21: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-22: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
PCR-23: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[root@TPM-VM-FC14 ~]#
```
Linux sees CRTM/Grub Logs

SeaBIOS

Trusted Grub
TrouSerS (TSS) running in the VM

IBM Research

TPM
Security Considerations of the vTPM

• Why not use hardware TPM from VM?
  – Pro: TPM provides protection for (private) keys
    • Key usage may depend on PCRs’ state
      → PCRs must be inside TPM
  – Con: Sharing of hardware TPM between VMs creates problems
    • Sharing of PCRs problematic
      → Each VM needs its own isolated TPM

• Who can compromise guest VMs’ vTPM data?
  – Typical powerful ‘root’ account
    • Use gdb to inspect QEMU and/or vTPM

• Long-term solution: SR-IOV (multi-instance) TPM to provide better protection
TPM Support in Libvirt Management Stack

• Libvirt extended for defining VMs with attached vTPM
  – Automatically creates vTPM’s persistent state file

• Domain XML describing the VM extended to add vTPM to VM:
  – Most simple XML

    <tpm type='libtpms'/>

  – Custom file for TPM persistent state

    <tpm type='libtpms'>
      <storage file='tmp/tpmstate.bin'/>
    </tpm>
TPM Support in Libvirt Management Stack

- vTPM state stored in encrypted form

```xml
<tpm type='libtpms'>
  <storage>
    <encryption format='qcow'>
      <secret type='passphrase' uuid='13ea49f7-2553-7308-5168-e337ade36232'/>
    </encryption>
  </storage>
</tpm>
```

- Initialize vTPM upon first startup

```xml
<tpm type='libtpms'>
  <initstate script='/etc/qemu/tpm_initstate.sh'>
    <parameter value='--take-ownership'/>
  </initstate>
</tpm>
```

Script generates initial TPM state
VirtManager TPM Support

Add TPM device to VM
Outline

• Motivation
  – Trusted Computing meets Cloud Computing
  – Cloud Computing Security Concerns

• Technologies
  – Key Contributions
  – VM and Virtual TPM Lifecycle
  – IMA, IMA-Appraisal & EVM
  – Cloud Use Cases

• Open Source Components + Architecture
  – QEMU and SeaBIOS extensions
  – Libvirt management stack + VirtManager

• Conclusion
Conclusion

• Trusted computing based security architecture to address aspects of cloud security
  – Linux open source components
    • Attestation: IMA
    • Data Integrity: IMA-Appraisal
    • File Metadata Integrity: EVM
    • Data confidentiality: TPM-based keys for encrypted filesystem
  – Virtualization open source components
    • Trusted Computing for QEMU/KVM
    • Management stack extensions for TPM support
Thank you!

Questions?

Stefan Berger, stefanb@us.ibm.com