vTPM: Virtualizing the Trusted Platform Module

Joshua Schiffman
It could happen to you...

- Computers offer no implicit reason to trust them
  - Key Loggers, root kits, spy ware
- Do you know of anyone who has fallen victim to one of these?
- What can we do?
Trusted Computing

- The Trusted Computing Group suggests we:
  - Deploy a **Trusted Platform Module (TPM)** in all systems
  - And an infrastructure to support their use
    - Shamon?
- TPMs allow a system to:
  - Gather and attest system state
  - Store and generate **cryptographic** data
  - Prove platform **identity**
Virtual Trust?

- Unfortunately, your computer might be an illusion…
- Advances in hardware virtualization
  - Improve system utilization
  - Saves money
  - Gives the illusion of an independent system
- A Virtual Machine Monitor (Hypervisor) could control what data is measured by the TPM
Virtualize the TPM

• Cannot have multiple users per TPM
• Why don’t we just virtualize the TPM?
  ▸ It would violate its security properties
    • Trust rooted in hardware
• VMs support unique lifecycles
  ▸ Suspend and Resume on different platforms
    • How do you trust a transient environment?
Requirements

- Authors identify four requirements for vTPMs:
  - Same usage model and command set
  - Strong association between VM and its vTPM instance
  - Strong association between real TPM and vTPM
  - Easy to distinguish between real and virtual TPM
The vTPM Model

• Similar to Xen, Dom-0:
  ‣ Contains access to the hardware TPM
  ‣ Control of vTPM instances
  ‣ Can spawn vTPMs

• Client-side driver request are routed to the server-side drivers
Secure Coprocessor

- The vTPM design was made to be modular
- PCI-X Cryptographic Coprocessor
  - Accelerated cryptographic engine
  - Tamper-Responsive
    - Not all TPMs have this support
  - Expensive
Xend is your Friend

- Implementation modified Xen to support the vTPM
  - Xen Management tool parses config file
  - Xend determines where the vTPM manager is located
    - Xenstore, frontend, backend, hotplug scripts
  - A table of VM to vTPMs is also maintained
    - Absence of mapping = create new instance
    - Presence of mapping = resume vTPM
Driver Considerations

- Existing TPM protocol assumes
  - Reliable bus
  - Must ensure a response is given to the last message before suspending the OS

- Shared memory (grant tables) is used to communicate between Front and Backends

- Concurrent access
Interesting Issues

- VM migration
- Associating vTPM with underlying system
- Key management
- Trust Management
VM Migration

• Created extensions to TPM 1.2 command set
  ‣ vTPM Management / Migration / Utility commands

• Migration is rather straightforward:
  ‣ Create instance with associated nonce
  ‣ Lock source with nonce and encrypt with key
  • Wrapped with parent TPM instance SRK
  ‣ Serialize and update message digest
  ‣ Migrate data, digest and verify
Associating the Real TPM

- The TPM has **security properties** that makes it different from most hardware
  - Virturalized TPMs cannot be totally transparent
  - Must have knowledge of the underlying system
- Solution: Divide the PCRs into **localities**
  - Lower set used for system
  - Upper set for the vTPM instance
- Issues with nested VMs?
Key Hierarchy

• Root keys stored inside the TPM to prevent leakage
  ‣ Endorsement Key (EK) to identify the platform
    • Manufacture certificates not so common as once prescribed
  ‣ Storage Root Key (SRK) to seal (encrypt) data / keys

• vTPMs are too ephemeral to bind to hardware TPM
  ‣ They are kept independent of the platform
  ‣ Speeds key creation
  ‣ Persistent store emulated with real TPM
Trust Management

- AIKs and SKs follows a chain of keys rooted hardware
- Programs rely on these keys (TSS, User PS)
  - Migrating VMs will change their host TPM EK
  - vTPM EKs are freshly generated for each VM
  - How can we identify a vTPM EK?
- Some VMs may not want to be on certain machines
Suggestions

- Create EK’ for each vTPM
- Create an AIK’ for all vTPMs
- Use a local authority to verify the vTPM
- Use a secure coprocessor
Create an EK

Diagram:
- CA signs AIK
- vTPM creates SRK
- EK is signed by AIK and represents Virtualized Environment
- CA signs AIK
- TPM creates SRK
- EK is signed by AIK and represents MSK
Create AIK

Diagram:
- Create AIK
- vTPM
- SRK
- EK'
- vTPM
- SRK
- EK
- CA
- AIK
- MSK
- Virtualized Environment
Take Away

• Virtualizing a TPM is not as easy as normal hardware
  ‣ Security and trust must be addressed
• It is not clear how to establish trust in a transient TPM identity
  ‣ We can compromise