Motivating Example

- Conscientious web server admin / dev
- Wants to protect most critical data
  - SSL private key, password file, ACL, …
- Evaluates low-cost options
- Her best efforts rest on a house of cards…

Challenge: Reducing the Trusted Computing Base

- Today’s OSes have too much power
- Total access to application data
- App may require little OS support
  - Self-contained computation ‘S’
- Trusted computing base for S includes majority of:
  - OS, drivers, and privileged applications!!!

What is S?

- Self-contained code in an application
- Data secrecy and integrity requirements
- General-purpose computing
- Some examples
  - Manages a private key for web server or CA
  - Manages Access Control List (ACL)
  - Is a compute client in distributed setting
  - Is similar to a Flicker session [McPaPeReIs2008]

Outline

- Motivation (done)
- High-Level Overview
  - Detailed Description
  - Prototype: Apache + SSL
  - Limitations
  - Summary & Conclusions

Meet TrustVisor

- Tiny hypervisor for isolation of code S
  - No scheduling or Inter-Process Communication
- Efficient transitions between OS and S
  - External verification of Output = S(Input)
  - Protected storage for S

Untrusted

Trusted

Attestable

OS

TrustVisor

HW

V
External Verification: Attestation

Verifier: What code are you running?
Target

\[ \text{Sign(S, Inputs, } K_{\text{TPM}}, \text{ } K^{-1}) \]

- Trust in attestation rooted in hardware TPM (Trusted Platform Module)
- SSL-enabled web server scenario:
  - Client can evaluate server before sending data
  - Enables more meaningful SSL server validation

Protected Storage

- Initially, S is “red” (untrusted)
- App can register S \( \rightarrow \) “blue” (attestable)
- TV enables “blue” code to protect data…

Verifier
Target
Protected Storage

- Access-controlled by identity of S (hash)
- Enabled by TPM-like Sealed Storage
- “Micro-TPM” in software

Alternative Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Metric</th>
<th>TCB Size (LoC)</th>
<th>Protection granularity</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic kernel</td>
<td>millions</td>
<td>–</td>
<td>–</td>
<td>best</td>
</tr>
<tr>
<td>Virtualization</td>
<td>millions</td>
<td>VM</td>
<td>consistent</td>
<td>good</td>
</tr>
<tr>
<td>Virtual TPM (vTPM)</td>
<td>millions</td>
<td>process</td>
<td>consistent code</td>
<td>good</td>
</tr>
<tr>
<td>Overshadow etc.</td>
<td>–100K</td>
<td>process</td>
<td>moderate</td>
<td></td>
</tr>
<tr>
<td>Security / µ kernel</td>
<td>&lt;1K</td>
<td>fine</td>
<td>process</td>
<td>moderate</td>
</tr>
<tr>
<td>Flicker</td>
<td>&lt;10K</td>
<td>fine</td>
<td>fine</td>
<td>poor</td>
</tr>
<tr>
<td>TrustVisor</td>
<td>&lt;10K</td>
<td>fine</td>
<td>fine</td>
<td>good</td>
</tr>
</tbody>
</table>

TrustVisor runtime TCB in lines of code:
- ~6500 C/ASM + ~2800 Headers
- Hypervisor + crypto

Outline

- Motivation (done)
- High-Level Overview (done)
- Detailed Description
- Prototype: Apache + SSL
- Limitations
- Summary & Conclusions

TrustVisor \( \leftrightarrow \) OS Architecture

TrustVisor:
- Virtualizes RAM, CPU
- Restricts DMA
- Restricts TPM to Locality 1

TrustVisor \( \leftrightarrow \) S Architecture

TrustVisor API
- Registration
- Invocation
- Micro-TPM

DMA Devices (Network, Disk, USB, etc.)
TPM
CPU, RAM
Chipset
Identifying S to TrustVisor

- Applications identify S via registration – Page-level protection granularity
- Applications make “normal” function calls – TrustVisor detects switch to S via traps
- S runs with no access to legacy OS – One set of Inputs and Outputs per invocation

Sensitive Code Timeline

- Multiple invocations during a single registration cycle
- S’s Runtime State Protected

Micro-TPM Design

- Small subset of hardware TPM operations for:
  - Protected Storage + External Verification
- TPMs are optimized for cost, not speed
- TrustVisor implements critical-path TPM operations in software on main CPU
  - Extend, Seal, Unseal, Quote, GetRand
  - Reduces latency by orders of magnitude
- Trust in Micro-TPM still rooted in hardware TPM
  - Infrequent TPM operations do not require virtualization

Outline

- Motivation (done)
- High-Level Overview (done)
- Detailed Description (done)
- Prototype: Apache + SSL
- Limitations
- Summary & Conclusions

Example App: Apache + SSL

- Goal: Protect long-term private key $K_{SSL}^{-1}$
  - Cert revocation is abysmal in practice
- Desired properties
  - Malware, malicious admin unable to learn $K_{SSL}^{-1}$
  - Externally verifiable configuration
- Two sensitive code modules (S)
  - S1: Generate and seal the long-term key (rare)
  - S2: Unseal and use the key during SSL session establishment (frequent)

Apache + SSL Performance

- ‘ab’ with 10,000 txns / trial, avg 10 trials
- Memory, Context Switching
- Normalized to Vanilla Linux (higher is better)
- 200 Concurrent Transactions
- Vanilla Linux, TrustVisor only, TrustVisor + S
Outline

- Motivation (done)
- High-Level Overview (done)
- Detailed Description (done)
- Prototype: Apache + SSL (done)
- Limitations
- Summary & Conclusions

Limitations

- Design-level
  - Does not currently provide trusted path to user
  - Requires application awareness
- Prototype-level
  - No SMP support (currently single CPU)
  - Only protects $K_{SSL^{-1}}$
  - Executable code for S must be proactively paged into memory before registration
  - AMD-only

Summary & Conclusions

- Tiny hypervisor to support isolation
- Externally verifiable via attestation
- Frequent TPM operations in software
- Compelling performance argument
- Requires no OS changes
- Conclusions
  - Interesting point in the design space
  - Foundation for future trustworthy systems

Q & A

- Thank you!
- jonmccune@cmu.edu
- http://www.ece.cmu.edu/~jmmccune