Portable Data Integrity and Confidentiality using Graduated Access Control

Puneet Mehrotra, Amanda Carbonari, Peter Chen, Ivan Beschastnikh, Andrew Warfield
Network, Systems, and Security Lab
CS@UBC
Motivating Example
What could go wrong?
Dropbox gets compromised

Curious eyes in public spaces

Changed circumstances – lost/stolen device
Bob’s data is mobile, Bob’s data access policy is **NOT**

**Problem 1:** distribution of the data and the data access policy are synonymous and binary

**Problem 2:** data access policy on remote devices (e.g., Alice’s phone) may be inadequate or not enforced
Access Policies Today

• Depend only on the application and device it is on
• Evaluated at time of distribution
• Binary decision at time of sharing (can/cannot)
Enforceable access policies for mobile data
Graduated Access Control on Remote Devices

Is Alice at home?

Mobile
Dynamically Resolvable
Programmable
Backward Compatible
Benefits

Dropbox gets compromised and encrypted

Curious eyes in public spaces

Data remains safe and encrypted

Bob decides who can access his data and where

Changed circumstances - lost/stolen device

Bob in public places who can access his data and where

Benefits
Use cases

Identity Protection
Revocation: remote delete, auto delete, policy change, remote state change

Electronic Health Records
Audit trails, role-based access

Data integrity and Provenance
Detect tampered data, audit trails

Sensitive Documents
Redaction, geo-fencing, time-fencing, role-based access
Our Solution

- Data centric abstraction of graduated access control
- Data and Policy encrypted together in a single mobile unit: **Trusted Capsule**
Our Solution: Trusted Execution Environment

- Examples include Intel Secure Guard Extensions (SGX) and ARM TrustZone
- Available on commodity ARM chipsets
- Hardware partitions CPU and memory into two logical worlds: secure and normal
- Compromising the normal world does not compromise the secure world
- Implement system call interceptor to allow applications to transparently operate on trusted capsules
- Secure world offers TEE for trusted capsule applications to evaluate capsule policy at syscall granularity

Diagram:
- Secure World
  - Trusted Application
  - Secure World OS
- Normal World
  - Application
  - Normal World OS (Linux)
- Isolation
- Secure Monitor

12
Our Solution: Trusted Capsule Server

- Maintain data owner policy uniform across all trusted capsule copies

- Actions:
  - Receive logging information from trusted capsules
  - Initiate policy change (ex: Remote delete)
Our Solution: Policy Engine

- Lua based policy language
- Global variables – trusted server IP and port
- States
  - Normal world OS states ex: process ids
  - Peripheral device information
  - Remote states
- Evaluates policy on op where op is the system call

```lua
1  -- API keywords
2  policy_version = 0
3  remote_server = "10.0.0.2:3490"
4
5  -- log
6  log_open = true
7  log_close = true
8
9  -- return keywords
10 policy_result = POLICY_ALLOW
11 comment = ""
12
13  -- policy-specific keywords
14 replace_var1 = "THIS IS A SECRET"
15
16 function evaluate_policy( op )
17  err = reduct( 12, 20, "replace_var1" )
18  if err := POLICY_NIL then
19    policy_result = err
20  end
21  return
22
23  if op == POLICY_OP_OPEN then
24    elseif op == POLICY_OP_CLOSE then
25    else
26      policy_result = POLICY_ERROR_UNKNOWN_OP
27      comment = "Unknown Operation"
28  end
29
30  end
31```
Implementation

- Prototype on LeMaker HiKey
  - ARM Cortex A53 processors
  - 8 GB eMMC Flash
  - 2 GB RAM
  - TrustZone unlocked

- Linaro OP-TEE OS version 1.0 (Secure World)

- Debian Linux Kernel 3.18.0 (Normal World)

- 128-bit AES and SHA-256 (Trusted Capsules)

Samsung Knox uses ARM TrustZone
Evaluation – Policy Language

<table>
<thead>
<tr>
<th>Policy</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merger Document</td>
<td>24</td>
</tr>
<tr>
<td>Transcript</td>
<td>25</td>
</tr>
<tr>
<td>Royal Photo</td>
<td>30</td>
</tr>
<tr>
<td>EHR</td>
<td>41</td>
</tr>
</tbody>
</table>

- Express all our use case policies with small LOCs
- Complex policies such as redaction can be expressed with few lines of code
- Lua interpreter required <2KB of stack
### Evaluation – Storage Overhead

<table>
<thead>
<tr>
<th></th>
<th>Data (KB)</th>
<th>Capsule (KB)</th>
<th>Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF Doc</td>
<td>137.34 KB</td>
<td>139.38 KB</td>
<td>1.42%</td>
</tr>
<tr>
<td>JPEG Image</td>
<td>204.10 KB</td>
<td>207.00 KB</td>
<td>1.42%</td>
</tr>
<tr>
<td>MP4 Video</td>
<td>4142.40 KB</td>
<td>4175.94 KB</td>
<td>0.80%</td>
</tr>
<tr>
<td>FODT Doc</td>
<td>54.80 KB</td>
<td>56.70 KB</td>
<td>3.47%</td>
</tr>
</tbody>
</table>

- Negligible storage overhead
Conclusion

- Current day policies are application/device-centric, evaluated once, binary and unchangeable

- We introduce graduated access control
  - Data owner can enforce access policies on remote devices
  - Define a continuum of actions rather than a binary can/cannot
  - Decouples access policy from data distribution

**Trusted capsules** based implementation using ARM TrustZone as our TEE
  - Mobile
  - Dynamically Resolvable
  - Programmable
Backup Slides
Graduated Access Control on Remote Devices

- **Mobile**: data access policy moves with the data
- **Dynamically Resolvable**: data access policy re-evaluated at time of access
- **Programmable**: data access policy is nuanced
- **Backward Compatible**: does not require application modification