Privacy Enforcement for Untrusted Cloud Applications

Emil Stefanov  (UC Berkeley)

http://www.emilstefanov.net/

with

Krste Asanovic
UC Berkeley

Jin Chen
Fudan University

David Culler
UC Berkeley

Petros Maniatis
Intel

Prashanth Mohan
UC Berkeley

Charalampos Papamanthou
UC Berkeley

Elaine Shi
University of Maryland

Dawn Song
UC Berkeley

Mohit Tiwari
UC Berkeley

Nguyen Tran
Google

Special thanks to Mohit Tiwari for providing some slides.
The Team
The Age of Big Data

Plentiful, and Private
Cloud

• Pool of shared resources
• Available on demand
• Highly scalable
• Lack of control & visibility
Need Data Protection as a Service

Vulnerable software

(Un) Intentional Misuse

Insider Attacks

RockYou Sued Over Data Breach

By Brennon Slattery, PCWorld  Dec 31, 2009 8:40 AM

FRIDAY, MARCH 12, 2010

Netflix Prize Update

In light of all this, we have decided to not pursue the Netflix Prize sequel that we announced on August 6, 2009.

Google fires employee for snooping on users

The Internet search giant says the software engineer broke its 'strict internal privacy policies.' He allegedly accessed information about four teenagers.

September 16, 2010  |  By Jessica Guynn, Los Angeles Times
Challenge #1

- Untrusted applications have access to user’s data.
Challenge #2

Novice Users
Challenge #3

• Decentralized:
  – Data
  – Authentication
  – Authorization
  – Identity Management
Basic Components

• Application Instances
  – Applications are split into instances.
  – Ex: document, photo, chatroom, GIT repository.

• Data Capsules
  – One data capsule per application instance.
  – For non-volatile data.

• Individual Containers
  – Each instance is isolated: placed inside container.

• Aggregate Containers
  – Have read-only access to multiple data capsules.
Access Control

• **Data Capsules have ACLs**
  – { Alice:rw, Bob:rw, Carol:rw, ... }

• **Initial ACL**
  – Creator can read/write.
  – If Alice creates a data capsule, the ACL is:
    • { alice:rw }

• **User-initiated sharing**
  – Users can change ACLs
  – Standardized across all apps
100 Top Web Applications

Cloud Storage
Personal Documents
Real-time applications
E-commerce
Social applications
Miscellaneous: Browsing, peer-to-peer

Source: CNET Webware 100 Winners  http://www.cnet.com/100/
Compatibility

• Want maximum compatibility.

• Legacy applications
  – Very few changes to port.

• Currently supports Linux applications
  – Including web applications running under Apache

• Techniques are generalizable to other operating systems.
LXC Containers

• Like lightweight Linux VM’s
• Share common Linux kernel
• Different namespaces
  – E.g., PIDs, network interfaces
  – Further isolation via cgroups
• Boots from subdirectory containing a Linux root directory.
• http://lxc.sourceforge.net/
ACL Based Communication

- ACLs transformed into firewall rules
• Each container has its own CoW root file system.
  – Saves lots of space & improves caching.
  – Container is chroot’ed into its CoW directory.
• Fuse based (more stable than kernel based unionfs)
LXC Container Life Cycle

• Too many containers slow down the system (e.g., more than 1,000)
• Need to shut down and recycle containers.
• Need to sync to disk first.
• Applications must specify when they are doing background work.
Viewers

• Problem:
  – Data spread across many containers

• Want to aggregate
  – Search
  – Summary
  – Explore efficiently

• Want to still enforce ACLs

• Our solution:
  – Viewers
Example: Aggregates search results

Allows the user to navigate to a search result

Want:
  - Generic (not application specific)
  - Privacy preserving (ACLs enforced)
  - Query privacy (Alice’s query does not leak to Bob) – not covered
• Untrusted view template specifies layout
• Trusted viewer with client-side isolation does aggregation
  – Cross-origin (cross-container) isolation in browser
  – Cross-origin navigation prohibited.
PPD Viewer Example

Outer view (static HTML)
*Specified in the application manifest*

```html
<html>
...
Search: <input name="Query" ... />
<input type="submit" ... />
...
<p>Showing results matching "PPD_Query":</p>...
<div id="Container1">...
<div id="Container2">...
</div>
</div>
</html>
```

Inner view for container 2 (static HTML)
*Generated by container 2*

```html
<table>
<tr>
<td><a href="#_PPD_Bindweed.jpg">
<img src="Bindweed.jpg" />
</a><br/>
 Bindweed flower</td>
<td><a href="#_PPD_BlueGem.jpg">
<img src="BlueGem.jpg" />
</a><br/>
 Blue gem flower</td>
</tr>
</table>
```

Resulting web page

Search: flower
Showing results matching "flower":

**Hawaii**
- Lily flower
- Buttercup flower
- Oleander flower

**Florida**
- Lantana flower
- Anagallis flower
- Verbascum flower

- Bindweed flower
- Blue gem flower
Untrusted Backend Services

Overview:
• Developers create untrusted backend services.
• Other developers can use them.
• Developers don’t have to trust each other.

Benefit:
• Decrease overhead while maintaining security.
Verifiable Untrusted Backend Services

• Verifiable services:
  – File system
  – Key-value store

• Uses on-the-fly cryptographic integrity checking.

• Untrusted backend service sees plaintext
  – Deduplication, compression

• Reduced TCB
Centralized Authentication

- User authentication is delegated to PPD
- Application just receives auth cookie
  - i.e., authenticated username
Platform Integrity Verification

- Maliciously modified platform will not boot.
- User data can only be decrypted by the verified platform (uses Intel TXT & TPM).
Remote Attestation

Audit

Platform

TXT/TPM

1. Auditor
2. Platform
3. TXT/TPM

- load kernel
- launch Intel TXT
- secure boot
- ok
- verify kernel
- unlock TPM data
- fetch key
- random challenge \( c \)
- \( c' = \text{Sign}_k(c) \)
- verify \( c' \) using platform public key
Evaluation

• Micro benchmarks
  – LXC Startup

• Applications
  – GIT
  – Etherpad
  – Friendshare (custom written)
LXC Startup Performance

- File System
- Linux Namespace
- Network Device

Container Creation Time (ms)
Friendshare Throughput

Small Requests: 10 filenames

Big Requests: 10KB images
Friendshare Latency

CDF

Latency [ms]