A Distributed Architecture for Sharing Ecological Data Sets with Access and Usage Control Guarantees

Ph. Bonnet, J. Gonzalez and J. Granados
jgon@itu.dk
Sensor data lifecycle

- Focus on instrumentation and data sharing
  - DB models defined at institution level for archiving/sharing
  - Typically, each data set is a separate entity. Focus is on making data accessible, not on making it easy/efficient to later work with this data.
  - Sharing policies defined but not enforced by the IT infrastructure.
Interact Project – Workflow Scenario

- **Transnational access (INTERACT)**
  - GOAL: People work on several sites and integrate observations across sites.
- **Field stations (Zackenberg, Abisko, ...)**
  - Some sensors and data loggers are deployed for automatic data acquisition. Technicians (A) or researchers (B) go in the field to collect data from data loggers or to manually perform data acquisition.
  - Data is handed to a program manager (monitoring activities). Researchers gather their observations (research activities). This is a form of manual sneakernet.
  - Field stations are not online. They are hard to access. People work on a single site.
- **Monitoring**
  - Program managers collect raw data and lead the cleaning process that might involve colleagues in other institutions.
  - Cleaned up data is then made publically available (as promised to the funding agency). In practice the public data sets from the monitoring programs are very difficult to work with for people who have not collected them (as data sets are just dumped, there is no query capabilities).
- **Research**
  - B collected data in the field, and collaborates with C to build a model, interpret it and write a paper about it.
  - Until the paper is published, B and C jealously keep their data. When the paper is published B and C make the data sets used for the graphs publically available on KBN (as required by Nature).
  - B archives the raw data and shares it with C (password protected ftp server at U.Oulu)
  - R reads a nature paper and gets access to the derived data set.
Data Sharing

• Sensitive data?
  • No, not really. Nothing like a patient record. Data is neither personal, nor classified.
  • However, we need incentives to encourage data sharing.
    Most researchers happily embrace the idea of sharing. It opens up observations to independent scrutiny, fosters new collaborations and encourages further discoveries in old data sets. But in practice those advantages often fail to outweigh researchers' concerns. What will keep work from being scooped, poached or misused? What rights will the scientists have to relinquish? Where will they get the hours and money to find and format everything?


• Our vision
  • A decentralized architecture enables researchers to keep on working as they used to.
  • They will now be able to control and audit who is accessing your data.
Interact Project - Tomorrow

- **Virtual Instrumentation**
  - Data loggers are networked and made available online (via a gateway that controls access to a field station server that collects data from the data loggers).
  - The data streams generated by the sensor are gathered on the data loggers, replicated on the field server and on the gateway.
  - Gateways are publically accessible (via subscriptions or access to snapshot data sets). Access rights should be enforced to differentiate public access to views over the data streams/snapshots and full access (to some data streams) by M on the data she manages.
  - Sneakernet is still required for those data which are manually collected (for backup purposes).

- **Decentralized Architecture to enforce Access and Usage Control**
  - What is usage control?
  - How to enforce it?
Audit: a posteriori control of how rights were used

Enforcement: a priori control of usage rights

Audit: a posteriori control of how rights were used

Requirements UCON\textsubscript{ABC}

UCON needs to be implemented within a security perimeter protected by hardware so that attackers with administrator privileges cannot disable it using software.

From inside the security perimeter it should be possible to “monitor” programs outside the security perimeter.

Communicating with programs in the security perimeter should entail a low overhead.
# Secure Platforms

<table>
<thead>
<tr>
<th></th>
<th>Level of Protection</th>
<th>Communication</th>
<th>Execution Environment</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Only Software</strong></td>
<td>Low</td>
<td>Messages, Sockets, SM</td>
<td>Rich OS</td>
<td>Android, Linux, IOS, Windows</td>
</tr>
<tr>
<td><strong>Software and Hardware</strong></td>
<td>Medium/High</td>
<td>Shared Memory</td>
<td>Rich OS + TEE</td>
<td>ARM TrustZone</td>
</tr>
<tr>
<td><strong>SW and Tamper resistant HW</strong></td>
<td>Very High</td>
<td>Narrow Interfaces</td>
<td>Secure Ad-hoc OS</td>
<td>IBM CryptoCards, TPM, Secure Token</td>
</tr>
</tbody>
</table>

**Normal Execution Environment**

- All Sorts of Applications
- Monitor
- With Low Overhead

**Security Perimeter**

- Monitoring Applications
- Protected by Hardware

---

*IT University of Copenhagen*
Building Blocks

- Enforce usage policies
- Define usage policies (UCON)
- Confidentiality (encryption)
- Integrity
Trusted Integrity Module (TIM)

Background Information

Secondary storage is exposed to attacks and data corruption, both when data is at rest and in use.

Logs are a vital component to verify the integrity of a running system. They can - and will – be tampered with by an attacker (101 system hacking).

Integrity is the base (root of trust) to build usage control. Indeed, usage control policies emerge from it.
Trusted Integrity Module (TIM)

Architecture

Rich Environment

- Platform Independent
  - Rich App
  - GLIBC

- System-Call Interface (SCI)
  - VFS
    - struct inode
      - ino (inode id)
      - user id
      - group id
    - super-block
    - dentry
      - file
      - directory

- Secure Environment Communication Interface
  - TIM
  - TSM
    - Transaction Logs
    - History Logs
    - VFS Metadata
    - Hash (file)
    - Hash (inode)
    - Hash (dentry)

- VFS Trusted Extension

- Shared Memory

Secure Environment

- Crypto Keys

Tamper Resistant Unit
Trusted Integrity Module (TIM)

Contribution

Provide an architecture to guarantee the integrity of system files adding a low overhead to file system primitives

Provide a method to log actions involving system files in trusted storage, preventing attackers to clean after themselves

Usage control policies emerge naturally from TIM, acting as a base for complex UCON-based policies
Next Steps

Implement UCON on top of TIM to add complex usage control policies to secure file operations (i.e., enforcement, embedded behaviour, and audit)

Extend TIM with a machine learning algorithm to learn from past attacks - antifrigility

Exploring how to monitor running processes without introducing a big overhead. Ideas: Use of resources, peripherals, etc.
Conclusion

• **Vision**: Data sharing policies enforced by IT infrastructure
  – Potential benefits for scientific data sharing, as well as *privacy* in general

• Work under way to realize this vision based on software components executed in Trusted Execution Environments (protected by hardware).
A Distributed Architecture for Sharing Ecological Data Sets with Access and Usage Control Guarantees

Ph. Bonnet, J. Gonzalez and J. Granados
jgon@itu.dk

Slides available at http://javigon.com/publications