AN SELINKS BASED SECURE ELECTRONIC VOTING SYSTEM

David An, Brian Corcoran, Eric Hardisty and Ben Langmead
Project Goals

- Examine whether or not SELinks would be a good platform to base an electronic voting system on.
  - What kinds of security guarantees can we provide using labels?
  - How well does SELinks mesh with the needs of an electronic voting system?
- Improve SELinks through use.
  - What features would be helpful?
- Create an electronic voting system which provides more security guarantees than the systems in use, that could reasonably be expanded into a usable system.
Voting System Design

- Inspired by two previous projects/papers:
  - Civitas: A Secure Voting System
    - Written in JIF (JFlow) and encodes voting properties using information-flow
    - Useful for what it reveals about the design of the system and its definitions of security properties
  - Designing Modular Vote-Tallying Software
    - Useful for its glossary of contest types (“tallying strategies”) and its object model
Voting System Design

- Simplifying assumptions:
  - Someone else handles cryptography
  - Ignore scalability/availability
  - Only support Plurality voting
  - Voting is in-person

<table>
<thead>
<tr>
<th>Component</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabulation teller</td>
<td>5,740</td>
</tr>
<tr>
<td>Common</td>
<td>3,173</td>
</tr>
<tr>
<td>Registration teller</td>
<td>1,290</td>
</tr>
<tr>
<td>Supervisor</td>
<td>1,138</td>
</tr>
<tr>
<td>Log service (bulletin board and ballot box)</td>
<td>911</td>
</tr>
<tr>
<td>Voter client</td>
<td>826</td>
</tr>
<tr>
<td>Registrar</td>
<td>308</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>13,386</strong></td>
</tr>
</tbody>
</table>
Security Properties

- The final tally is verifiably correct
- Voter verifiability
- Universal verifiability
- Receipt-freeness
- Roles are respected
  - Supervisor, Registrar, Voter, Tabulation Teller
SEVOTE ORGANIZATION

Ballot Definition

Voter Interface

Vote Tallying

Database
SYSTEM DEMO
SEVote employs SELinks labels to:

- **Authenticate agents**
  - A caller able to supply the appropriate dependent pair \((I’m-a-voter, ()\{I’m-a-voter\})\) is authentic

- **Vouch for intermediate results**
  - E.g., a Contest labeled with \{Supervisor\} has been OK’ed by the Supervisor

- **Ensure mediation**
  - E.g., cast Ballots are labeled \{Registrar\} to prevent the tabulation teller from interacting with them except in certain prescribed ways
**Oracle Label Security**

- Multilevel Security (MLS)
- Allows row-based security policies

<table>
<thead>
<tr>
<th>ID</th>
<th>DES</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABC</td>
<td>HIGHLY SENSITIVE</td>
</tr>
<tr>
<td>2</td>
<td>DEF</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>3</td>
<td>GHI</td>
<td>CONFIDENTIAL</td>
</tr>
</tbody>
</table>

User 1 (CONFIDENTIAL)

SELECT QUERY

<table>
<thead>
<tr>
<th>ID</th>
<th>DES</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>DEF</td>
<td>PUBLIC</td>
</tr>
<tr>
<td>3</td>
<td>GHI</td>
<td>CONFIDENTIAL</td>
</tr>
</tbody>
</table>
FORMALIZING OLS

- Policy function for read access

\[
\text{ReadAccess}(L_d,L_u) = \\
L_d[v] \leq L_u[v] \land \\
\forall c (c \in L_d[C] \rightarrow c \in L_u[C]) \land \\
(L_d[G] = \emptyset \lor \exists g (g \in L_u[G] \land \exists g' (g' \in L_d[G] \land g' < g)))
\]

- Where \( L_d \) is a data label and \( L_u \) is a user label
  - \( v \) is a level, \( C \) is a set of compartments, \( G \) is a set of groups
XTOLS ARCHITECTURE

Oracle DBMS

XTOLS APIs

Oracle Label Security (OLS)

Labeled Data

Unlabeled Data

Label Components

Web Server

Labels and Label Components

Query on Labeled Data

Query on Unlabeled Data

Read/Write
XTOLS Architecture (cont’d)

- Table Checker
  - Interacts with Oracle database for validity of table definitions

- Label Components and Data Labels
  - They must be defined by an OLS admin prior to usage; No arbitrary label can be used to label data
  - Cache label components and data labels for efficiency

- XTOLS APIs
  - Reflects Oracle Label Security policies
  - Provides a reasonable programming interface
SELinks Improvements

- Tutorial
- Preprocessor
  - Exception syntax
  - Oracle syntax extensions (David)
- Modules
  - Tracks errors to original file/line
- Standard Library
  - Maybe, Set, Map
- Database
  - Oracle support
  - Arbitrary SQL commands
**Future Work**

- **Voter Interface**
  - What more can we do with labels?
  - Need a system for write-in candidates.
  - Support additional voting types.

- **Tallying**

- **SELinks**

- **Database Modifications**
Questions?
Backup Slides
MODULAR VOTE-TALLYING SOFTWARE

Figure 4. Object Model for Modular Vote Tallying Software
Figure 1: Civitas architecture
**FORMALIZING OLS**

- Label is a 3-tuple $\langle v, C, G \rangle$ where
  - $v$ is a (sensitive) level
  - $C$ is a set of compartments
  - $G$ is a set of groups

- **Examples:**
  - Levels: Highly Sensitive, Sensitive, Confidential, Public
  - Compartments: Financial, Chemical, Operational
  - Groups: Eastern Region, ER New York, Western Region