Grey: Making Logic-Based Access Control Practical and Usable

Lujo Bauer
Device-enabled Authorization

- Smartphones on a trajectory to “win” in the market
  - Stand to inherit mobile phone market that shipped over 1.1 billion units in 2008 [IDC]—or more than one phone per six people in the world

- Unique combination of capabilities
  - Computation, communication, user interface

- Goal: to use smartphones to intelligently control environment
  - Loan you my car without giving you my phone
  - Send money from my phone to my friend’s phone
  - Give my secretary temporary access to my email without revealing information (e.g., password) that could be used at a later time
  - Use my phone to open my hotel room door, without ever stopping by the front desk

... and do it all from a distance
Grey Deployment

- Universal, flexible, end-user-driven access-control system for physical and virtual resources
- Deployed in Carnegie Mellon’s Collaborative Innovation Center
  - Approximately 35 Grey-capable doors and 30+ users at the moment
  - Grey-compatible Windows XP, Vista and Linux login modules
- Plus a deployment in progress at UNC Chapel Hill
Grey: An Example Scenario

- Lujo’s students are allowed in 2121
- Faculty are allowed in 2121
- At CMU, Lujo’s secretary speaks on behalf of Lujo

...
Grey: An Example Scenario

1. Hi, Please open 2121
2. Prove Lujo says open 2121
3. Prove Scott says open 2121
   → Lujo says open 2121
4. Proof of Scott says open 2121
   → Lujo says open 2121
5. Proof of Lujo says open 2121
Grey: An Example Scenario

1. Hi, Please open 2121
2. Prove Lujo says open 2121

Lujo

Scott

Lujo’s Office, 2121

- High assurance
- Rich audit logs
- Flexibility in policies

4. Proof of Scott says open 2121 → Lujo says open 2121

Digitally signed credentials...

...assembled in a formal, mechanically verifiable proof
Some Research Challenges

- Logics for access control
  [ESORICS 2006, NDSS 2007, SACMAT 2009]

- Distributed theorem proving

- Helping users configure access-control policies
  [CHI 2008a, SACMAT 2008, CMU TR 2009]

- Improving usability / evaluating usefulness in practice
  [SOUPS 2007, CHI 2008b]

- Technology and approach applicable in other settings
  - Web applications, databases, within OS, in IT environments, ...
Can We Make Configuration Easier?

- Setting up policies takes effort
- Incorrectly set up policies can wrongly allow or deny access

- How to help users easily set up *correct* policies?
Can We Make Configuration Easier?

[Bauer, Garriss, Reiter SACMAT ’08]

- Setting up policies takes effort
- Incorrectly set up policies can wrongly allow or deny access

- How to help users easily set up correct policies?

- Mechanism involves two steps:
  - Identifying intended policy and misconfigurations in the implemented policy
  - Resolving misconfigurations by augmenting the implemented policy

- “Misconfiguration” refers to authority that is intended to exist but has not been given
Observation: access-control policy exhibits patterns

- Inconsistencies in these patterns can indicate misconfigurations
- These patterns are observable from access-control logs
- Need centralized collection of logs to analyze

Use Association Rule Mining [Agrawal and Srikant '94]

- Input: series of records characterized by a fixed number of attributes
  - E.g., record is a shopping cart, attributes describe contents
- Output: rules (or statistical patterns)
  - People who buy peanut butter and jelly usually buy bread

Use rules to identify anomalies

- Alice bought peanut butter and jelly – did she forget bread?
## Data Representation

<table>
<thead>
<tr>
<th>Record1</th>
<th>AttA</th>
<th>AttB</th>
<th>AttC</th>
<th>AttD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>-</td>
</tr>
</tbody>
</table>
## Policy Misconfigurations

### Constructing Rules

<table>
<thead>
<tr>
<th></th>
<th>AttA</th>
<th>AttB</th>
<th>AttC</th>
<th>AttD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record1</strong></td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td><strong>Record2</strong></td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Record3</strong></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td><strong>Record4</strong></td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- **Rule:** $A \Rightarrow B$
  - Confidence: 0.5
- **Rule:** $A \Rightarrow C$
  - Confidence: 0.75
### Policy Misconfigurations

#### Identifying Misconfigurations

<table>
<thead>
<tr>
<th></th>
<th>ResA</th>
<th>ResB</th>
<th>ResC</th>
<th>ResD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>Bob</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Charlie</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>David</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

**Rule:** ResA $\rightarrow$ ResC

**Potential Misconfiguration (a.k.a., a prediction)**

- Alice to Charlie
- Bob to Charlie
- David to Charlie

**Users:** Alice, Bob, Charlie, David

**Resources:** ResA, ResB, ResC, ResD
Log of 10,911 accesses drawn from Grey deployment

Spans 16 months

Contains accesses by 25 users to 29 resources

Policy matrix: indicates what accesses should be allowed under implemented and intended policy

- Implemented: accesses that occurred in log
  - assumes no unauthorized access
- Intended: surveyed users
Policy Misconfigurations
Identification Simulation

Chronological Access History

Access Matrix

T  -  T  -
-  -  -  -
T  -  T  -
T  -  -  T

Intended Policy Matrix

T  -  T  -
T  T  -  -
T  T  T  -
T  -  -  T

Prediction correctly identifies a misconfiguration

Mine Rules

(A → C)
Two measures of success:
- Accuracy: what percentage of predictions are “correct”
- Coverage: what percentage of misconfigurations are predicted

We measure accuracy and coverage versus intended policy

Parameter: $minconf$
- Only predict using rules with confidence $> minconf$
Policy Misconfigurations

Prediction Accuracy (Intended Policy)

Rules with higher confidence are more accurate
Policy Misconfigurations

Feedback

- Some mined rules are not a good indicator of policy
- Use feedback to prevent these rules from repeatedly making incorrect predictions
  - Incorrect prediction: decrease score
  - Correct prediction: increase score
- If score of rule falls below a threshold, stop using rule
Policy Misconfigurations

Prediction Accuracy (Intended Policy)

Feedback significantly improves accuracy for low-med confidence rules
Policy Misconfigurations

Accuracy/Coverage Tradeoff (with Feedback)

Good compromise:

~70% accuracy

~55% coverage:

0.3 < minconf < 0.5
Once a misconfiguration is identified, a human must determine if it should be repaired – but which human?

- Single administrator → trivial
- Distributed administration → multiple users can repair
  - Asking all of them is very annoying
Construct a candidate list of users based on previous history

Sort candidate list by the number of times they helped

Strategies we evaluated for constructing candidate list:

- OU: who helped when Other Users accessed this resource
- OR: who helped when this user accessed Other Resources
- OU + OR: Union of OU and OR
- OU + OR + PPA: Union plus Principals who Previously Accessed this resource
Policy Misconfigurations
Resolution Simulation

Logs of previously resolved misconfigurations

Identify misconfigurations

Access Matrix

Construct candidate list

{ Bob, Charlie, ... }

Ask Bob to resolve P

Chronological Access History
Success rate: % of correctly predicted misconfigurations that can be repaired
- Essentially, how often we ask the right person
- Identifying a misconfiguration is only useful if corrected

High-latency accesses: # of misconfigurations repaired at the time of access (fewer = better)
- Each time, one user must be interrupted, and another user must wait

Total user interaction time: rough approximation of the amount of time all users spend interacting with the system
Policy Misconfigurations

Resolution Success Rate

Limit=2 is useful
Limit=none is not useful
Policy Misconfigurations

High-Latency Accesses

44% reduction in high-latency accesses
Policy Misconfigurations

**Total User Interaction Time**

- VERY rough approximation
- Factors in time spent resolving predictions proactively and time saved by reducing high-latency accesses

- Times measured from our deployment, applied to simulated events

- Places where user interaction is required:
  - Time a user waits while misconfiguration repaired
    - Ranged between 25 seconds and 18.6 hours
    - Median = 98 seconds
  - Time user spends repairing misconfiguration
    - Avg = 23 seconds
Policy Misconfigurations

**Total User Interaction Time** (across all users over 16 months)

All time spent resolving misconfigurations proactively is offset by time-of-access savings.
More realistic deployment scenarios
  - Adaptive prediction algorithms
  - Dynamically adapt with environment for best efficiency

More accurate prediction
  - Establishing correlation based on proof structure, credentials, similarity in what has not been accessed
Policy Misconfigurations

Summary

- For reasonable parameters, on our dataset we can simultaneously
  - Reduce high-latency accesses by 44%
  - Reduce total user interaction time
  - Identify ~55% of misconfigurations w.r.t. intended policy
- Works on partial data, performance stable over time
Grey Summary

- A deployed platform for experimenting with distributed access control

- Innovations in:
  - Authorization logics
  - Distributed proving
  - Policy configuration
  - Usability analysis

- Why it matters
  - Access-control technology with *high assurance, auditability, flexibility*
  - Technology and approach applicable in other settings
    - Web applications, databases, within OS, in IT environments, ...
Thanks!

http://www.ece.cmu.edu/~lbauer