Policy by Example

- Adwait Nadkarni
Advised by: William Enck
Motivation

• **Our prior work:** Data secrecy for application-specific data. *(Aquifer [CCS’13], Weir [USENIX Security’16])*
  – Example: Backing up receipts to the cloud.

• **What we already know:**
  • How to design a policy that may be appropriate for applications.
  • How to enforce those policies in a system (specifically a *modern* OS such as Android).

**Question:** *How to get the security policy from the user?*
Applications often acquire security context from the user: 1) what data to protect, & 2) how to protect it.

Insight:
• How do we know what to protect?
  – Application-defined tags
  – Environment variables (i.e., location, time).

Research Goal: Provide a framework for the user to harness this context to specify security policies.
Policy by Example

• General Approach:
  1. Get *examples* from the user (i.e., user-specific).
  2. Train on the examples.
  3. Predict policy decisions at runtime.

• *example* = [context] : [policy decision]

• For a *derived data object* that contains data 1) created at the hospital *and* 2) from a document *and* 3) from a receipt,
  -> DENY cloud export

**Fig: A context-policy “example”**

```plaintext
Context tags

<table>
<thead>
<tr>
<th>Hospital</th>
<th>Document</th>
<th>Receipts</th>
</tr>
</thead>
</table>

: DENY

Context

Policy
```
Hypothesis

It is feasible to predict policy decisions based on a set of context-policy examples provided by the user.

Success Criteria

We plan to perform a study with Expert users, with the following success criteria:

1. The approach is accurate in predicting policies.
2. Discover challenges for extending approach to regular users:
   1. **User Error:** We know the rate, and identify the causes.
   2. **Incorrect predictions:** Identify the cause of the fault (e.g., the algorithm, the initial training dataset, or the expressiveness of contexts).
Data Collection & Experiments

• Data collection:
  – Collecting the training set.
  – Collecting weights.

• Experiments:
  – Resolving user-errors and reasoning about change.
  – Testing.
Collecting the training set

1. Getting the context-policy examples (expert users) –
   – Users are given a set of predefined tags to combine.
   – Users can define their own tags.
   – Users provide example contexts, and the respective policy (or policies).

<table>
<thead>
<tr>
<th>Context</th>
<th>Work Cloud</th>
<th>Personal Cloud</th>
<th>Work Email App</th>
<th>Personal Email app</th>
<th>Social App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traveling</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Photos+Work</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Traveling+WhiteboardSnapshot</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• Some stats from the initial study:
  – 8 participants, 30 examples per person on average
  – 44 unique context tags.
Collecting Weights

• Not all context tags are the same: Some may be more important.
  – What important means depends on the kind of policy.
  – E.g., for secrecy, important may mean sensitive.
• 44 unique context tags! → Adding weights is non-trivial

Our approach:
– Make semantic groups of contexts: (9 groups)
– Generate weights from a partial order on the groups.

Tasks for expert users:
1. Verify context groups.
2. Order context groups.

#. Work: Name=Work
  Work=Work, WorkTravel, ManagersHome, FromWorkEmail

#2. Personal: Name=Personal
  Personal=Home, PersonalTravel, FromPersonalEmail, Afterhours, Weekends, Night,
  EarlyMorning, Traveling, School, GirlfriendsHome

#3. Documents: Name=Documents

#4. Recordings: Name=Recordings
  Recordings=Audio, Video, Torrent

#5. Notes: Name=Notes
  Notes=Postit, Note

#6. Scheduling:
  Todo=Memo, Calendar

#7. Medical:
  Medical=MedicalPurpose

# ASSUMPTION: If i<=j (i.e., i; j), then
# The first line after comments contains:
  Finance;Medical;Work;Personal;Documents;History;Notes;Recordings;Todo
  Medical;Finance
  Work;Medical
  Medical;Personal
  Work;Personal
  Notes;Documents
  Work;History
  Documents;History

Task 1: Verify Groups
Task 2: Modify Group orders
Resolving user-errors, reasoning about change

• **Potential error:** Contexts of two examples are similar, but have different policies.

• Our approach:
  
  1. **Manual Review:** Let the user review their training set, and change any policy decision if required.
     
     1. For each change, the user indicates the cause, i.e., “error”, “change of mind”, “don’t understand what this means”, and also provides justification.

  2. Our algorithm then looks for potential errors, and suggests policy-decision changes to the user.
     
     1. E.g., context={Home + Photos}, policy(Workcloud)=deny? (y/n)
Testing

- **Testing set:** 50-100 random context examples.
  - The examples must be *relevant*, i.e., must be close enough to at least one sample in the training set.
- Ask the user to label as many samples as possible in a fixed time period (we are considering 20 minutes).
- Predict policies for the testing set, and compute accuracy.
Thank you!

Questions?