Hails: Protecting Data Privacy in Untrusted Web Applications

Daniel B. Giffin, Amit Levy, Deian Stefan, David Terei, John Mitchell, David Mazieres and Alejandro Russo

I. Motivation

- Web platforms are the norm
  - Third-party developers extend websites using apps
  - Apps provide features that use personal user data
- Many trust concerns with third-party developers
  - Cannot determine trustworthiness of apps
  - Developers may be malicious or ignorant
- How do platforms enforce data privacy in apps?
  - User can decide to give app access to data
  - Cannot control how app uses data
- Building secure web apps is hard
  - Typically intertwine policy throughout code
  - Cannot trust well-meaning developers

II. Goal: A web platform framework

- Web platform framework that is deployable today
  - Cannot assume users will install new hardware, OS, etc.
- Usable by average web developers
  - Many similar systems (e.g., HiStar, Aeolus) require developers to be security experts ➔ non-goal
  - Policies should be easy to specify and inspect
  - Apps should be structured in a consistent manner
- Must enforce security policy across untrusted apps
  - Specify policy in single location, enforce end-to-end

III. Changing the hosting model

- Current model
  - App developers host their own apps
  - Difficult for platform to enforce policy, hence usually just specify a terms of service
- New Hails model
  - Platform provider hosts apps
  - Platform can enforce policy using information flow control

IV. Core ideas

- Information flow control
  - Security policy associated with data
  - Policy specifies where information can flow
  - Policy follows data through system ➔ enforced by runtime
- New paradigm: Model-Policy-View-Controller

Model-Policy (MP)

Specify data interface and policy on where data can flow

- Users trust MPs that handle their data, but do not trust VCs
- Developers build apps (VCs) without thinking about security

Naturally specify policy in terms of data itself

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Policy: only user can modify profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>Jen</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>jenaol.com</td>
<td></td>
</tr>
<tr>
<td>friends</td>
<td>[Alice, Bob]</td>
<td></td>
</tr>
</tbody>
</table>

View-Controller (VC)

Implement UI and handle and respond to HTTP requests

- Leverage the fact that data models already encode policy such as ownership, relationships between users, etc.

V. Case study: gitstar.com

- Code hosting website based on git
- GitStar provides MPs the specific projects and users
- Third-party authors implement code viewer, wiki, etc.

VI. Evaluation

- Developer feedback on usability of Hails
  - MPVC simplifies reasoning about security
  - Common security bugs (e.g., mass assignment) ➔ futile
  - Lacks scaffolding tools like Rails, Django, etc.

- Faster than Ruby frameworks, comparable with Apache

![Java Jetty](image)

- Normalized Requests/Second
  - Pong: 47.6K R/s
  - Table: 479 R/s
  - DB Read: 1.1K R/s
  - DB Write: 1.4K R/s

- Hails, Sinatra, Apache PHP