Bridging the Security Gap with Decentralized Information Flow Control


Stanford, †MIT, *Chalmers
Project Goal

Make it possible for programmers who are not security experts to build secure web applications
Architecture

- **Server**: Language framework for building secure apps
  - Hails, LIO

- **Client**: Sandboxing and browser security
  - Starlight, Dune
The Server Side Today: Web Apps

- Most apps structured around MVC (Model-View-Controller)
  - Rails, Django, Struts, .NET, others...
- Useful for compartmentalizing development
Why is the Web so &$@*ing Broken?!
The Server Side Today: Web Apps

• No notion of security policies

• Ad-hoc security checks throughout applications
  
  • Easy to forget a check (e.g. GitHub mass assignment vulnerability)
  
  • Extracting the policy requires looking at the whole application

• Often breaking MVC abstraction
Minding the Gap

Lots of research on secure systems
Jif, HiStar, Asbestos, Nexus, Ur/Web, …

Technologies not adopted

• Modify entire stack
• Not appropriate for dynamic systems like the web
• Policies are hard to write
• No guide for structuring applications
Hails: A web platform framework

- Hails targets web platforms, not just apps
  - All code is untrusted and potentially malicious

Goals

- Suitable for web platforms
- Usable by web developers
  - Easy to write policies
  - Easy to write the rest of the app
- Deployable today
  - Change as little of the stack as possible
The Server Side Today: Web Platforms

Give me Jen’s profile, please

Aalyah
Zen
Lopez
Current Solution

Allow Access?

Allowing Smiley access will let it pull your profile information, photos, your friends’ info, and other content that it requires to work.

[ ] Allow or cancel

By accepting, you agree to the Facebook Platform User Terms of Service in your use of Smiley.
Change the hosting model

- Developers host apps on in their own datacenters
- Platforms enforce security contractually (e.g. terms of service)

Hails: A new approach

- *Platforms* host apps on their own hardware, on top of Hails
- Use information flow control to **ensures** apps obey security policies
Case Study: Gitstar.com
Adding Policy to MVC

- New paradigm: Model-Policy-View-Controller
  - Policy specified independantly
  - No policy in the Model, View or Controller

- Hails has two types of third-party code
  - Model-Policies (MPs)
  - Provide data model and policy
  - View-Controllers (VCs)
  - Web server executables that link to MPs
Trust Model in Hails

- View-Controllers are completely untrusted
  - Includes most of the interesting functionality, like UI
- Model-Policies must only be trusted with the data they define
  - Users have to trust that they set good policies.
- Hails uses information flow control (IFC) to enforce policies on data models, end-to-end
**MPs and VCs in Gitstar**

- The Gitstar platform provides:
  - MPs for projects and users
  - A VC for managing projects and users
    (http://www.gitstar.com)

- Third-party authors:
  - Source code browser
  - Wiki
  - Follower app
  - Their own MPs

- In fact, nothing special about the Gitstar VC
MPs and VCs

A closer look...
Model Policy

A document oriented data-store:

- Documents are stored in collections, stored in databases
- Semi-structured schema with flexible data-types

```
users collection:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>Jen</td>
</tr>
<tr>
<td>email</td>
<td><a href="mailto:jen@aol.com">jen@aol.com</a></td>
</tr>
<tr>
<td>friends</td>
<td>[Alice, Bob]</td>
</tr>
</tbody>
</table>
```
Model Policy

- Web app data *already* encodes policy
- Function from a document to a policy

users collection:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>Jen</td>
</tr>
<tr>
<td>email</td>
<td><a href="mailto:jen@aol.com">jen@aol.com</a></td>
</tr>
<tr>
<td>friends</td>
<td>[Alice, Bob]</td>
</tr>
</tbody>
</table>
collection "users" $ do
  access $ do
    readers ==> anybody
    writers ==> anybody
  field "user" key
document $ \doc -> do
  readers ==> anybody
  writers ==> ("user" „from„ doc)
field "email" $ labeled $ \doc -> do
  readers ==> ("user" „from„ doc)
  \ fromList ("friends" „from„ doc)
  writers ==> anybody
View Controller

- A VC is a web request handler
- Implement UI and external API
  - Source code viewer, RSS feed, Wiki editor,…
- Handle all data persistence through MPs
- Low barrier, since new VCs can reuse existing MPs

Bugs in VCs are manifested as broken features – never as vulnerabilities
Hails: Protecting Data Privacy in Untrusted Applications

Resources
- Brief motivation and architecture overview
- Tutorial (slightly outdated)

Installation
You can compile and install Hails as usual with cabal-dev:
- $ cabal-dev instal-deps
- $ cabal-dev install

Launching an app
If you define your main application (named `server`) in YourAppModule.hs, you can launch it with:
- $ cabal-dev run

Haskell Web Platform Framework.
Goals

- Suitable for web *platforms*
- Usable by web developers
- Deployable today
Evaluation: Usability

√ MPVC simplified reasoning about security √ Hails rendered common security bugs futile
χ Need scaffolding tools
χ Writing policies is hard.
? Better with new policy DSL
Evaluation: Performance

<table>
<thead>
<tr>
<th></th>
<th>Hails</th>
<th>Sinatra</th>
<th>Apache PHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pong</td>
<td>47.6K R/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>479 R/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB Read</td>
<td>1.1K R/s</td>
<td></td>
<td>1.4K R/s</td>
</tr>
<tr>
<td>DB Write</td>
<td>1.1K R/s</td>
<td></td>
<td>1.4K R/s</td>
</tr>
</tbody>
</table>
Limitations / Present & Future Work

- Confined to Haskell
  - Now - cjail
  - Future - Dune

- Covert channels
  - Internal timing closed ([ICFP 2012])
  - External timing - mitigation
  - How much to mitigate?
  - More work to do…
  - Cache-based timing attack
tl;dr

- Current platforms: functionality vs. privacy
- Hails platforms guarantee security end-to-end
  - Host apps on platform
  - Make policy explicit
  - Enforce policy with information flow control

$ cabal install hails