Hails: Protecting Data Privacy in Untrusted Web Applications

COS 316: Principles of Computer System Design

Amit Levy & Wyatt Lloyd

Hails context

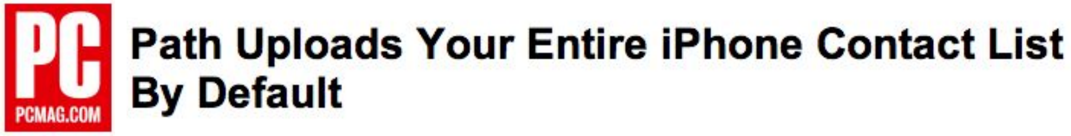
- Daniel B. Giffin, *Amit Levy*, Deian Stefan, David Terei, John Mitchell, David Mazières, & Alejandro Russo
- Developed 2010-~2015
  - First git commit from current version December 20th 2011
- Hails/Gitstar (‘12) -> Gitstar Inc. (‘14) (later renamed Intrinsic) -> VMWare Intrinsic
- Key ideas (aka how it makes MAC practical):
  - Complexity: Leverage Haskell language to build IFC as a library → easier to iterate
  - Performance: Leverage “purity” in Haskell to minimize security checks
  - Simplicity: Natural to *extract* policy from data, should be natural to use end-to-end policies on data

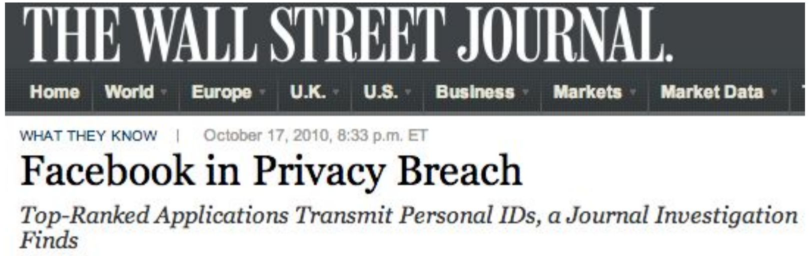
github:develop

Web platforms are **great**!
They allow third-party developers to build apps that use our personal data.

Google Developers

foursquare™ DEVELOPERS


Path Uploads Your Entire iPhone Contact List By Default


THE WALL STREET JOURNAL.

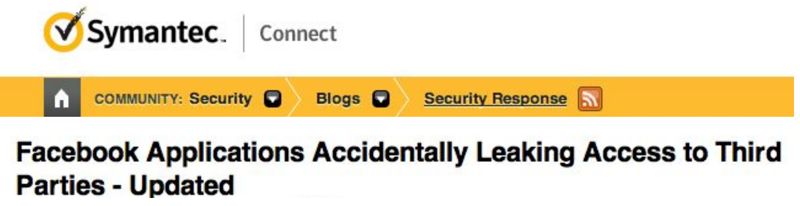
Home | World | Europe | U.K. | U.S. | Business | Markets | Market Data

WHAT THEY KNOW | October 17, 2010, 8:33 p.m. ET

Facebook in Privacy Breach

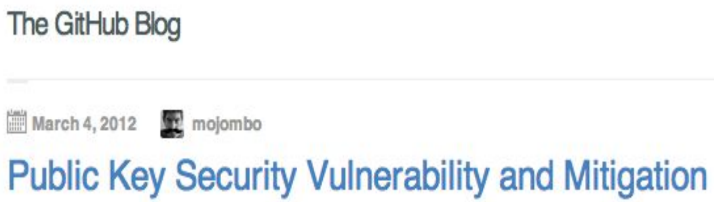
Top-Ranked Applications Transmit Personal IDs, a Journal Investigation Finds

Web platforms are **scary**!
They allow third-party developers to build apps that use our personal data.


Symantec. | Connect

COMMUNITY: Security | Blogs | Security Response

Facebook Applications Accidentally Leaking Access to Third Parties - Updated


The GitHub Blog

March 4, 2012 mojombo

Public Key Security Vulnerability and Mitigation

Trust Concerns

- Don’t know the developers
  - Cannot determine trustworthiness of apps
- They may be malicious or security-unaware
- Building secure web apps is hard
  - Even well-meaning authors cannot be trusted

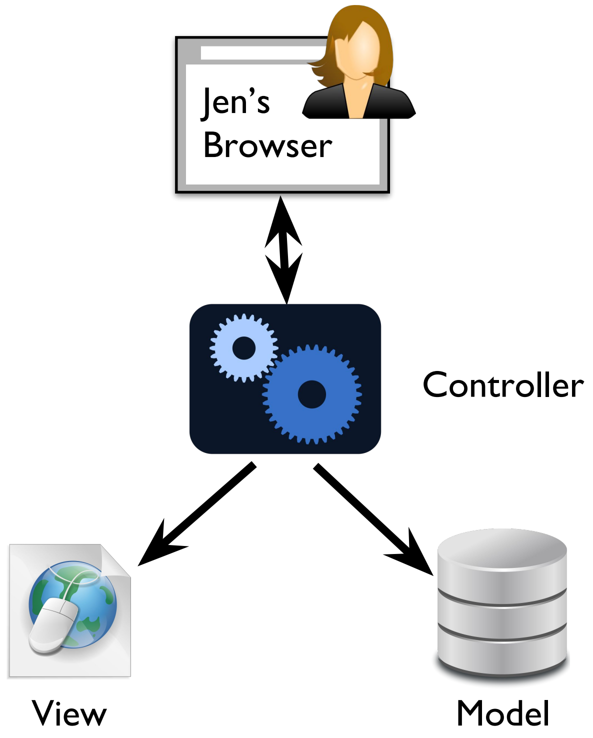
Typical App Design

Use the MVC paradigm

Model: interface to data

View: renders pages

Controller: handles and responds to HTTP requests



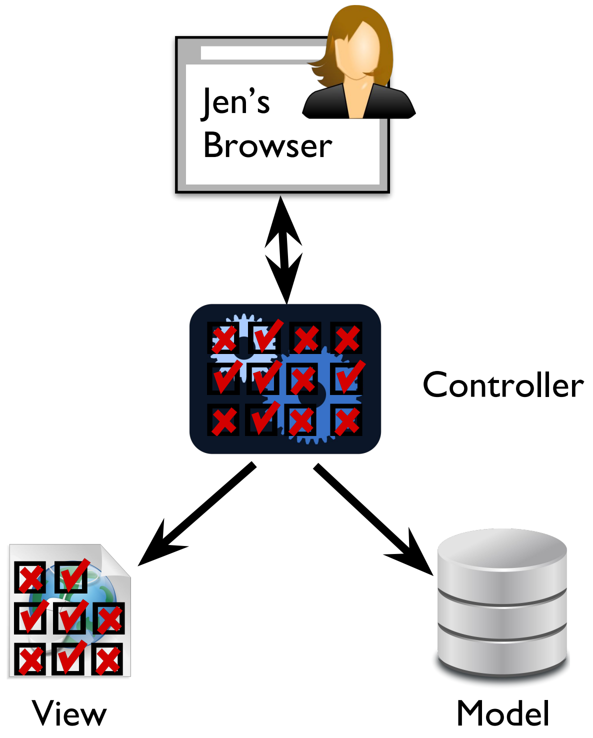
Typical App Design

How is security policy specified and enforced?

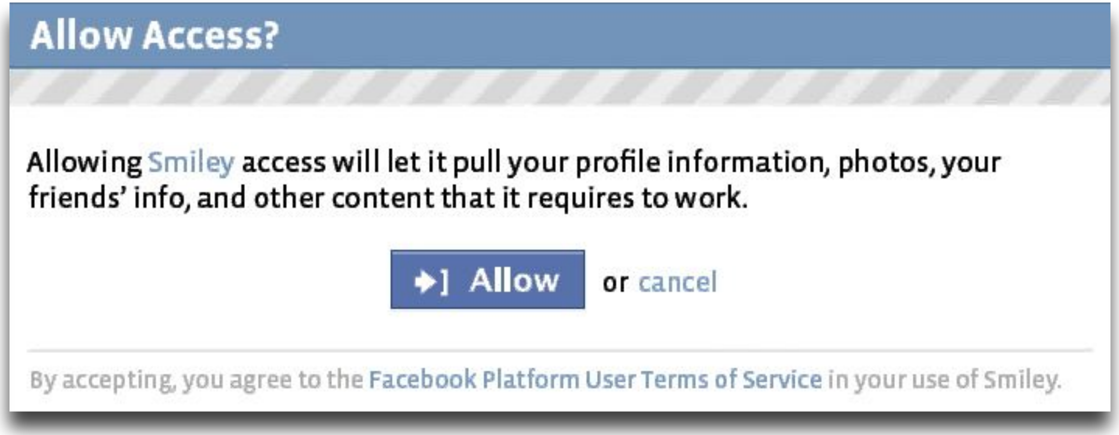
- E.g., only Jen's friends may see her email address

Intertwined throughout code

- Error prone and not scalable



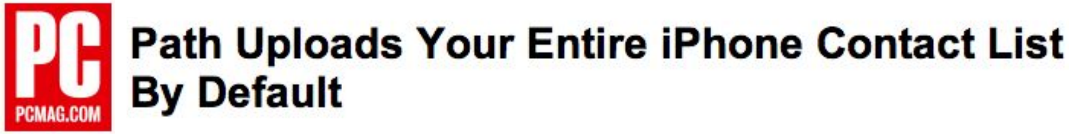
Platform “solutions”


Allow Access?

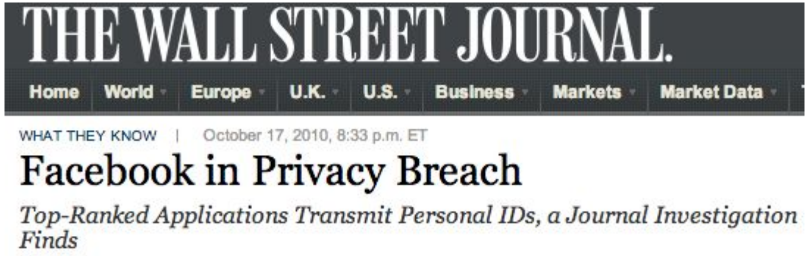
Allowing Smiley access will let it pull your profile information, photos, your friends' info, and other content that it requires to work.

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

Users can decide to give an app access to data, but can’t control how the app uses your data.



Path Uploads Your Entire iPhone Contact List By Default

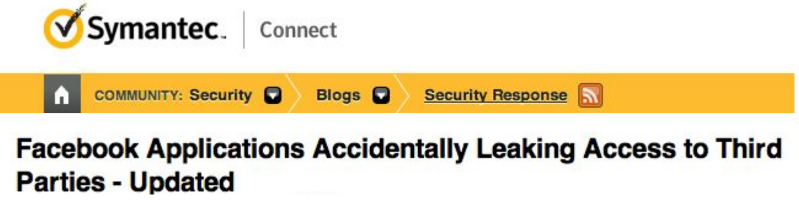


WHAT THEY KNOW | October 17, 2010, 8:33 p.m. ET

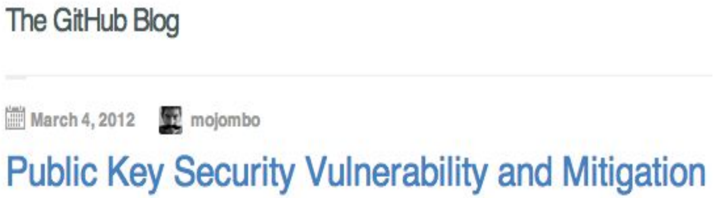
Facebook in Privacy Breach

Top-Ranked Applications Transmit Personal IDs, a Journal Investigation Finds

Is there any hope for privacy on platforms?



Facebook Applications Accidentally Leaking Access to Third Parties - Updated



The GitHub Blog

March 4, 2012 mojombo

Public Key Security Vulnerability and Mitigation

Change the hosting model

- Current model
  - App developers host their own apps
  - Platform enforces security: terms of service
- New model
  - Platform provider hosts apps
  - Platform enforces security mandatorily: information flow control

Hails: A web platform framework

- Security policy is explicit and first-class
    - Specified as single concise module
- Users still trust core platform components
- Apps are untrusted
    - *Language-level* information flow control guarantees apps always obey policy

Hails vs Previous Systems

Aeolus, HiStar, Nexus, Jif, Ur/Web, ...

- No guide for structuring applications
- Policies are hard to write
- Not appropriate for dynamic systems, e.g., web
- Modify entire application stack

Goals

- Deplayble
- Usable by Web developers
- Suitable for building extensible Web *platforms*
    - Enforcing policy across untrusted apps

Adding Policy to MVC

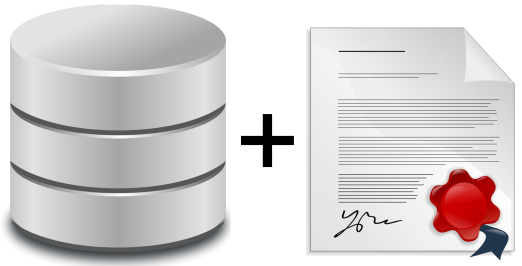
New programming paradigm: Model-***Policy***-View-Controller

- Policy specified alongside data model
  - Models are *partially* trusted to define the policy related to model data
- No policy code in View or Controller
  - Vast majority of bug-prone code
  - All* the code that third-party apps use to handle sensitive data

*Except the front-end code in the browser which, today, is *much* of the app's code

Two categories of code

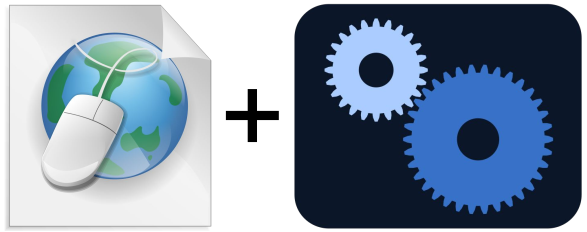
Models-Policies (MPs)



Specify data model and policy on data

Users *trust* MPs they use to handle data

Views-Controllers (VCs)



Implement UI and other functionality

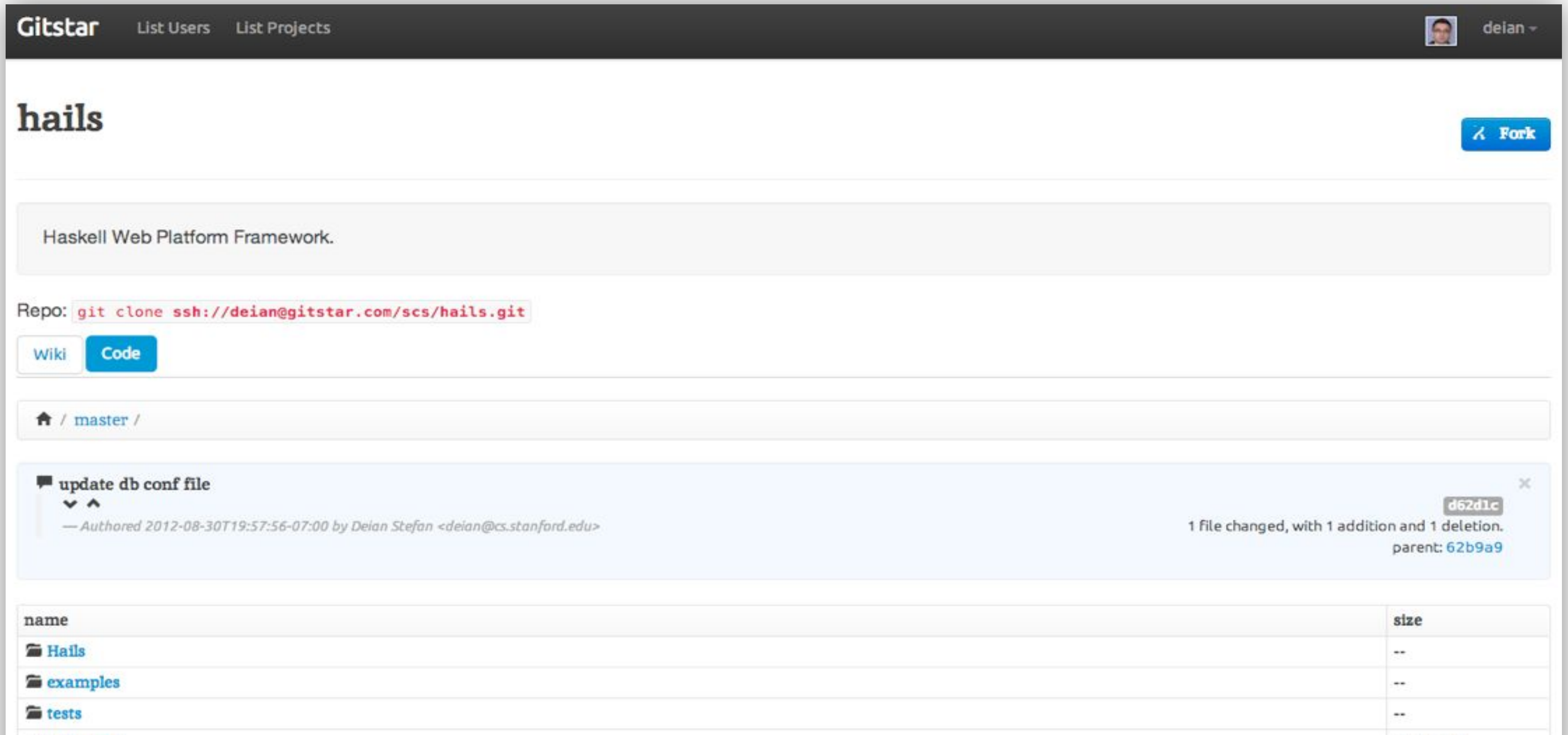
Users need not trust VCs with data

Policy enforced globally

Information flow control

- Policy specifies where data can flow
    - **Wrong**: app cannot read Jen’s email address because it may leak it to Eve
    - **Right**: app can read Jen’s email address, but only reveal it to Jen, Alice or Bob
- Policy follows data through system
- Runtime enforces policy end-to-end
    - E.g., when making an HTTP request

Case study: Gitstar



hails

Haskell Web Platform Framework.

Repo: `git clone ssh://deian@gitstar.com/scs/hails.git`

/ master /

update db conf file

— Authored 2012-08-30T19:57:56-07:00 by Deian Stefan <deian@cs.stanford.edu>

d62d1c

1 file changed, with 1 addition and 1 deletion.

parent: 62b9a9

name	size
Hails	--
examples	--
tests	--

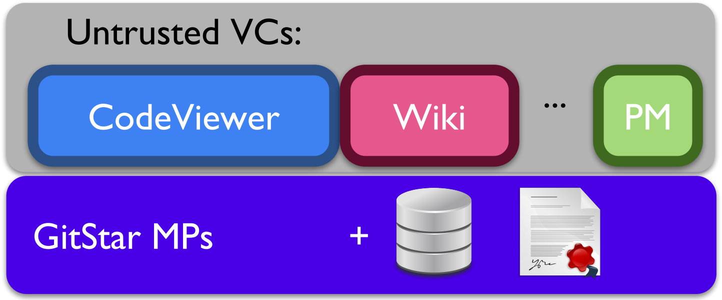
Case study: Gitstar

GitStar provides

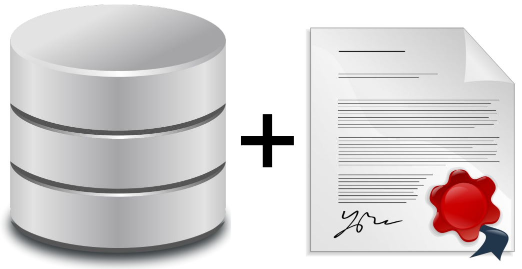
- MPs that specify projects and users
- VC for managing projects and users

Third-party authors provide

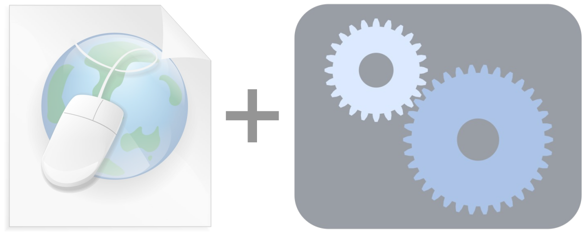
- Code viewer
- Wiki
- Follower app
- etc.



Models-Policies (MPs)



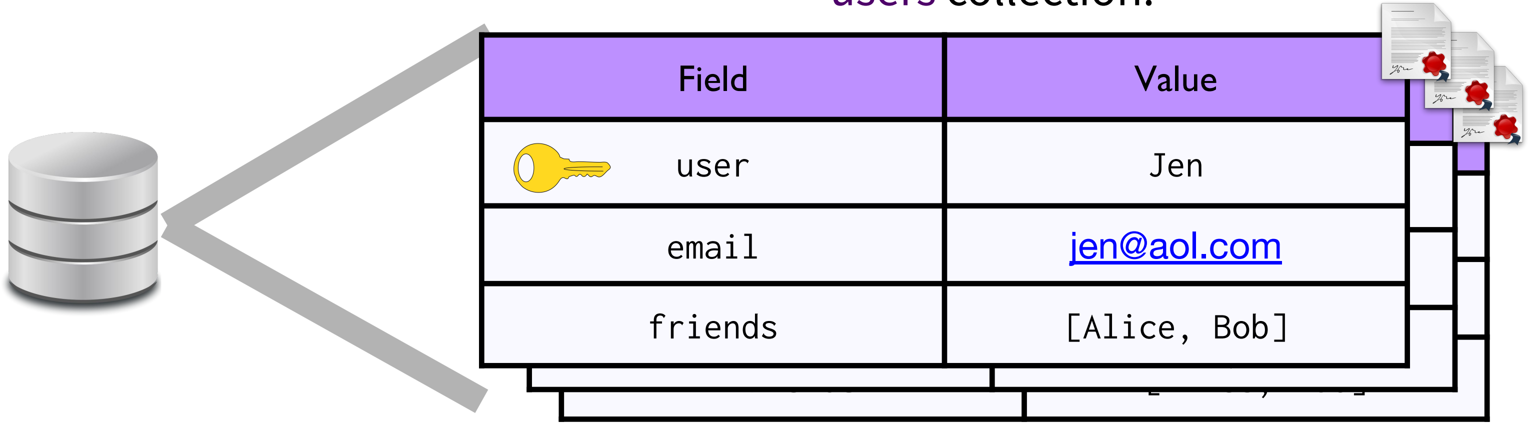
Views-Controllers (VCs)



Model-Policy (MP)

Data model: document-oriented

- Collection: set of documents
- Document: set of field-value pairs


users collection:

Field	Value
user	Jen
email	jen@aol.com
friends	[Alice, Bob]

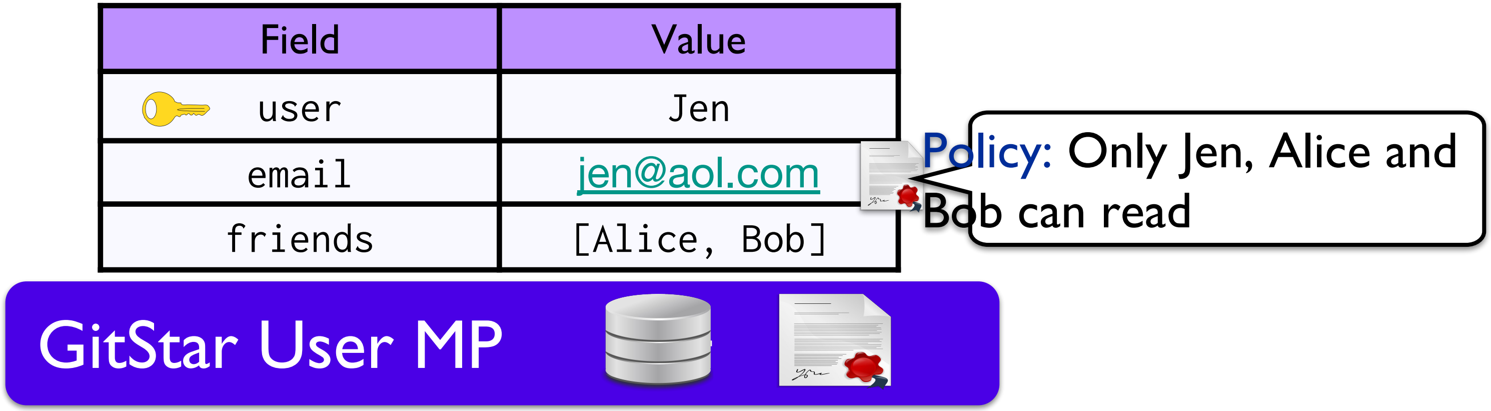
Model-Policy (MP)

- Policy specifies restrictions on:
  - Collections, documents, fields
  - E.g., only Jen may modify her profile
  - E.g., only Jen and her friends may read her email address
- Policy composes
  - E.g., to read document you must be able to read the collection

Example: Enforcing policy

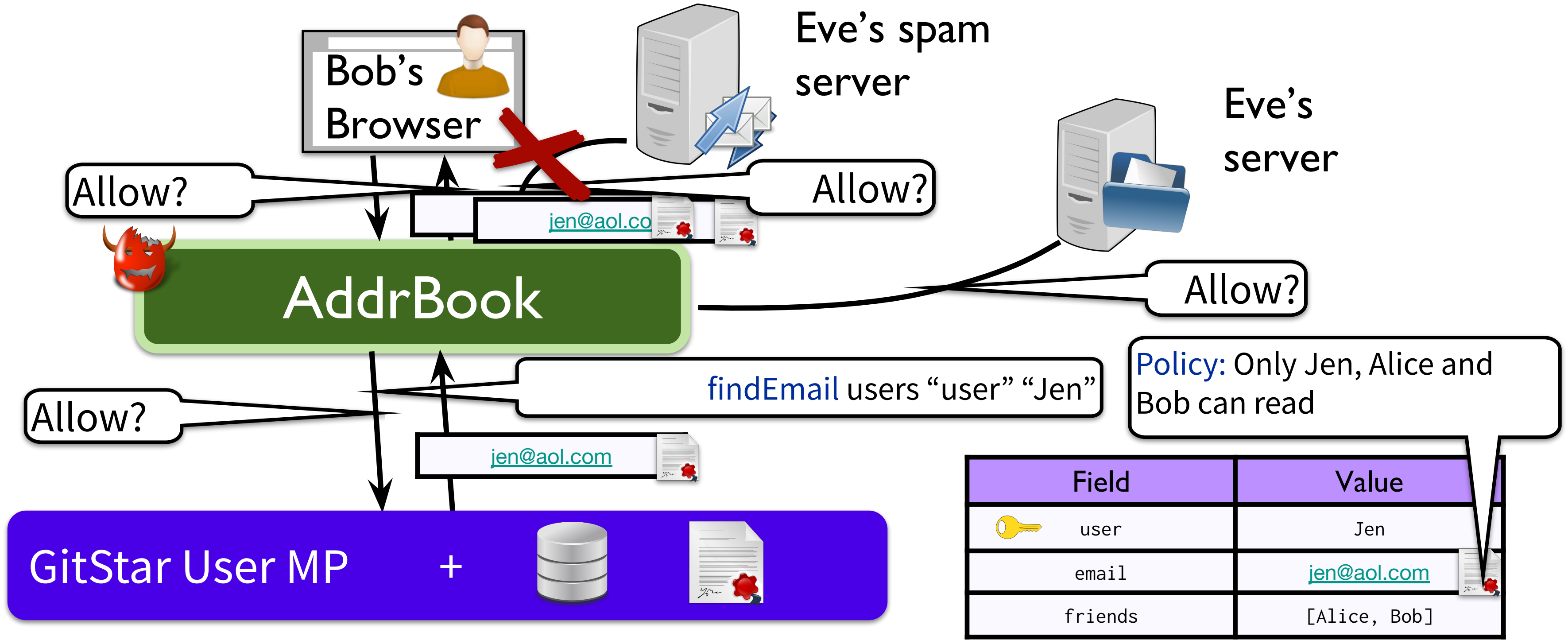
- MP:

Field	Value
user	Jen
email	jen@aol.com
friends	[Alice, Bob]



- Eve's untrusted address book VC:

Example: Enforcing policy

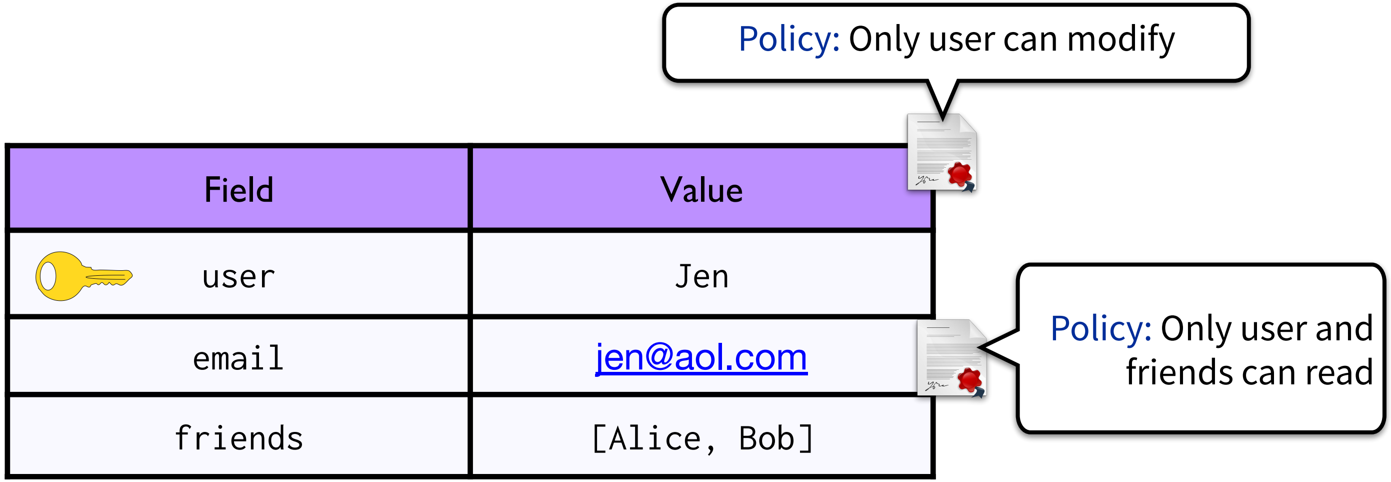


Field	Value
user	Jen
email	jen@aol.com
friends	[Alice, Bob]

Policy specified in terms of data

Web app data models already encode policy

- Ownership
- Relationships between users
- …


Policy: Only user can modify
Policy: Only user and friends can read

Field	Value
🔑 user	Jen
email	jen@aol.com
friends	[Alice, Bob]

Example: Policy specification

```
collection “users” $ do
   access $ do
     readers ==> anybody
     writers ==> anybody
   field “user” key
   document $ λdoc -> do
     readers ==> anybody
     writers ==> (“user” `from` doc)
   field “email” $ labeled $ λdoc -> do
     readers ==> (“user” `from` doc)
             ∨ fromList (“friends” `from` doc)
     writers ==> anybody
```

Collection is public

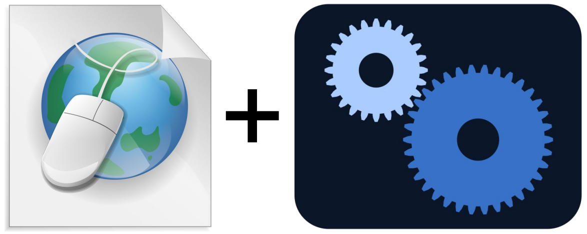
Index documents by user names

Only Jen can modify document fields

Only Jen, Alice and Bob can read Jen’s email address

Models-Policies (MPs)

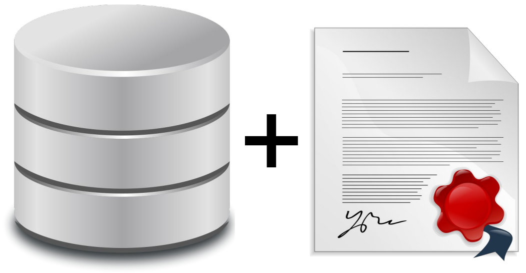
Views-Controllers (VCs)



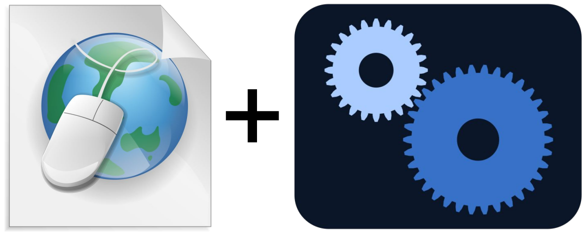
View-Controller (VC)

- A VC is a request handler
- Provide application functionality
  - E.g., source code browser, blog editor, …
- Invoke MPs to store/fetch user data
- Bugs in VCs are never vulnerabilities
  - Runtime enforces security policy

Models-Policies (MPs)

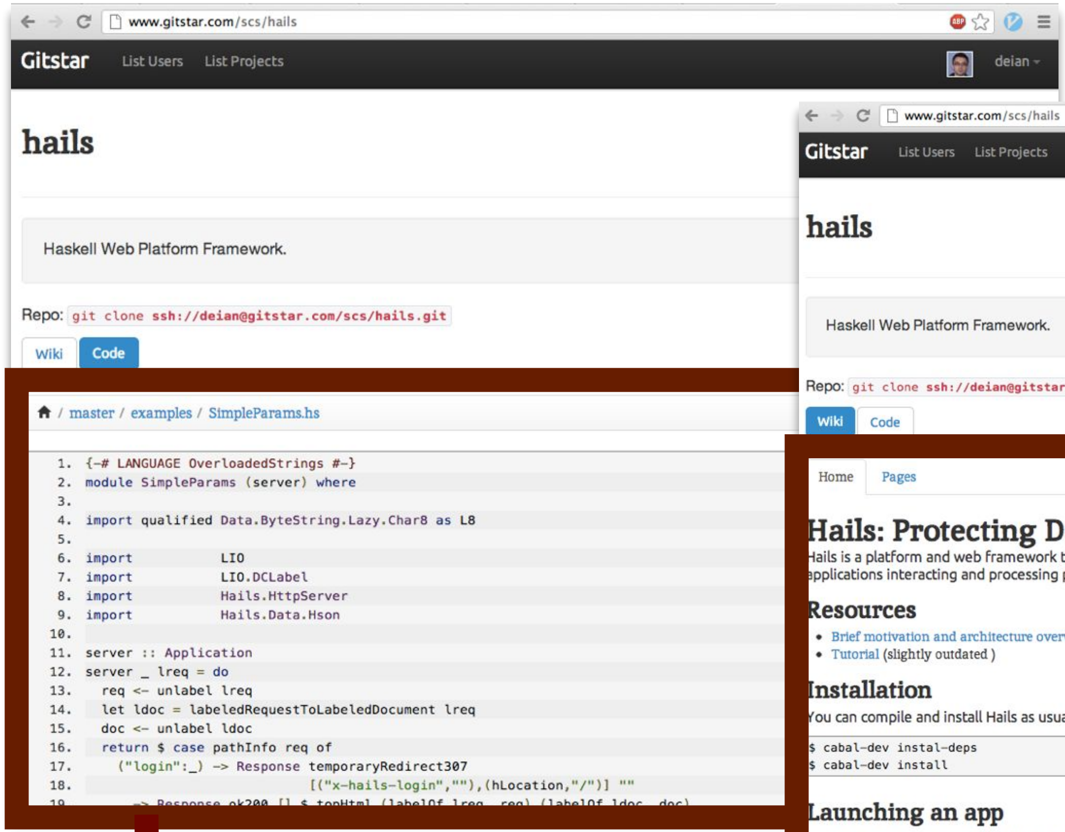


Views-Controllers (VCs)



Implications of MPVC

- Users: choose VCs based on functionality
- Developers: build apps on top of existing user-data
    - Models and policies are reusable



hails

Haskell Web Platform Framework.

Repo: `git clone ssh://deian@gitstar.com/scs/hails.git`

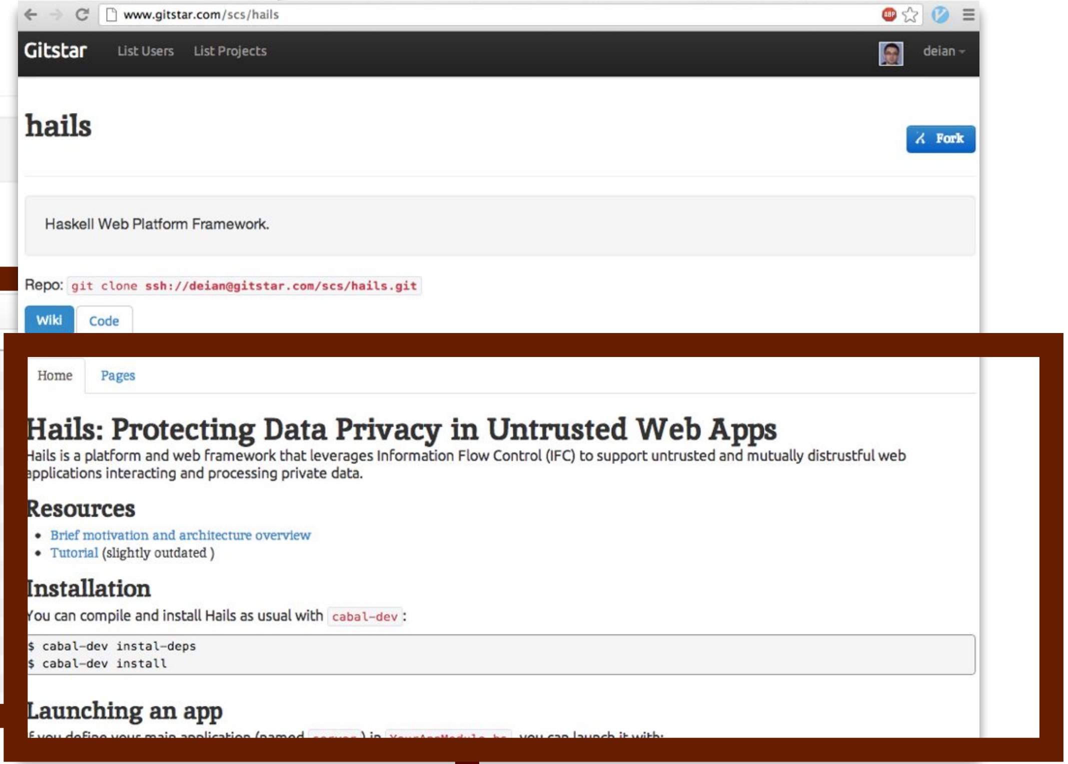
master / examples / SimpleParams.hs

```haskell
{-# LANGUAGE OverloadedStrings #-}
module SimpleParams (server) where

import qualified Data.ByteString.Lazy.Char8 as L8

import           LIO
import           LIO.DCLabel
import           Hails.HttpServer
import           Hails.Data.Hson

server :: Application
server _ lreq = do
  req <- unlabel lreq
  let ldoc = labeledRequestToLabeledDocument lreq
  doc <- unlabel ldoc
  return $ case pathInfo req of
    ("login":_) -> Response temporaryRedirect307
                                   [("x-hails-login",""),(hLocation,"/")] ""
      -> Response ok200 [] $ topHtml (labelOf lreq, req) (labelOf ldoc, doc)
```



hails

Haskell Web Platform Framework.

Repo: `git clone ssh://deian@gitstar.com/scs/hails.git`

Hails: Protecting Data Privacy in Untrusted Web Apps

Hails is a platform and web framework that leverages Information Flow Control (IFC) to support untrusted and mutually distrustful web applications interacting and processing private data.

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:

GitStar Project MP +

Implementation

- Hails is a Haskell library
  - Quick turnaround on API design
  - Developers can use existing tools and libraries
- Hails runtime system
  - Provides HTTP server that invokes VC
  - Enforces information flow at the language-level

Evaluation: Usability

✓ MPVC simplifies reasoning about security when building a platform

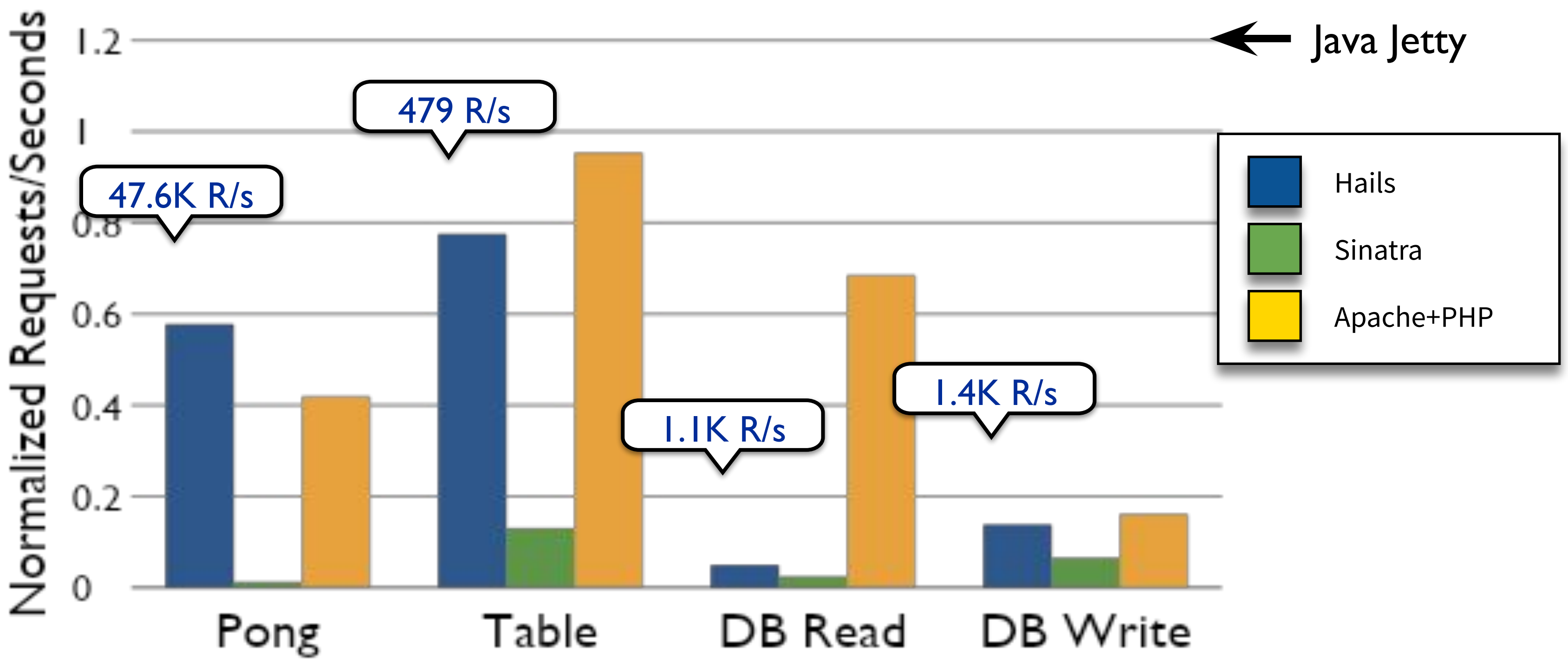
✓ Hails renders common security bugs futile
E.g., mass assignment vulnerability

- Need scaffolding tools

- ~~Writing raw policy is hard~~

    ✓ Writing policy with DSL is simpler

Performance evaluation



Conclusions

Current platforms: functionality vs. privacy

Hails platforms guarantee security across apps

- Hosts apps on platform
- Make policy explicit
- Enforce policy with information flow control