

Building secure systems with LIO

Deian Stefan and David Mazières



STANFORD
UNIVERSITY

Building systems is hard.



```
if ((err = SSLHashSHA1.update(&sha1, &rec->data[rec->len])) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&sha1, &rec->data[rec->len])) != 0)
    goto fail;
goto fail;
if ((err = SSLHashSHA1.final(&sha1, &hash)) != 0)
    goto fail;
```

Building secure systems is harder.

The average developer



- Not equipped to write secure code
 - SSL developers (arguably) are
- Most bugs are in application code
 - Recent MIT study¹: 83% of CVE's are in app code!

The average developer



Can we expect them to build secure systems?

➡ **Yes!** Use IFC to minimize damage cause by bugs.

Information flow control (IFC)

- **Goal:** data confidentiality and integrity
- **Idea:** track and control flow of information
 - Associate policy with data
 - Ensure that all code abides by data policies
- ➡ code that doesn't specify policy can be untrusted!

In this talk: LIO & co.

- **IFC system**
 - Dynamic IFC enforcement as a library
- **Policy specification**
 - Simple label model: DCLabels
 - Hails-like automatic labeling for web applications
- **Running Example**
 - Conference review system



DCLabels (demo)

LIO

- **Idea:** mostly-coarse grained IFC
 - Single context label protects all values in scope
 - Labels can be associated with references, files, etc.
 - Use clearance to restrict reads/writes (DAC)
- **Idea:** implement IFC system as a Haskell library
 - Use Haskell's monad support to create sublanguage

**Context label
(demo)**

Labeled objects (demo)

Clearance (demo)

Challenge: policy specification

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

... this is hard!

Web apps: use Hails MPVC model

- Structure web app into:



Model-Policy: specify policy alongside data model



View-Controller: app logic, no policy code

- Leverage authoritative information in data
 - Specify policy as function of the data its protecting
 - Automatically label at DB interface

Data model & policy in chair

- Paper

- **Secrecy:** PC members and authors can read
- **Integrity:** authors can modify paper before deadline

- Review

- **Secrecy:** non-conflicting PC member can read and, if review process is done, so can authors
- **Integrity:** only reviewer can modify

MP with LIO's SimpleDB (demo)

What now?

- **Extend the app to a web app (+VC)**
 - Use lio-simple web framework or Hails
- **In a similar manner we built other web apps**
 - LambdaChair, GitStar-{manager,wiki,viewer}, LearnByHacking, commenting system, user auth
- **Students (at UPenn & Stanford) managed to extend policies and apps in non-trivial ways**

Conclusions

- Building secure systems is hard
- IFC with LIO makes the problem tractable
 - Flexible & permissive enforcement mechanism
 - DCLabels & MPVC simplify policy specification
- Lots of research to be done on both fronts
... though policy specification is still behind!

Thank you!

www.labeled.io

References

1. David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich.
Why does cryptographic software fail? A case study and open problems.
In Proceedings of the 5th Asia-Pacific Workshop on Systems, Beijing, China, June 2014.