UNDERSTANDING SECURITY MISTAKES DEVELOPERS MAKE

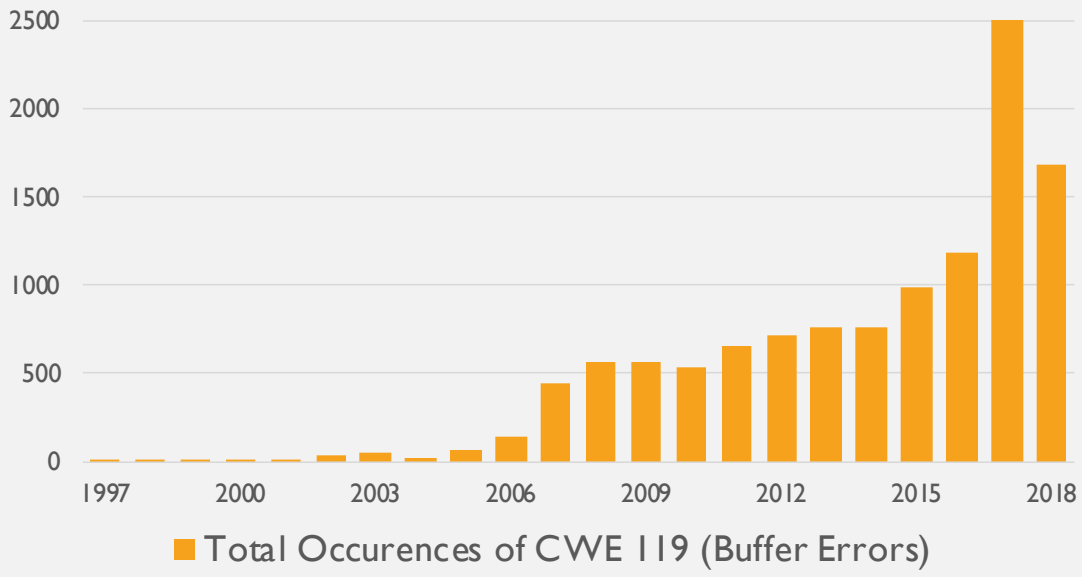
Qualitative Analysis from Build It, Break It, Fix It

Daniel Votipka, Kelsey Fulton, James Parker, Matthew Hou,
Michelle Mazurek, and Mike Hicks
University of Maryland

SP2 MARYLAND CYBERSECURITY CENTER

MANY REAL VULNERABILITIES ARISE FROM "SOLVED" PROBLEMS

- Buffer overflows
- SQL injection
- Bad randomness
- Static keys



https://nvd.nist.gov/vuln/search

From Reaves et al., "Mo(bile) Money, Mo(bile) Problems," USENIX 2015.

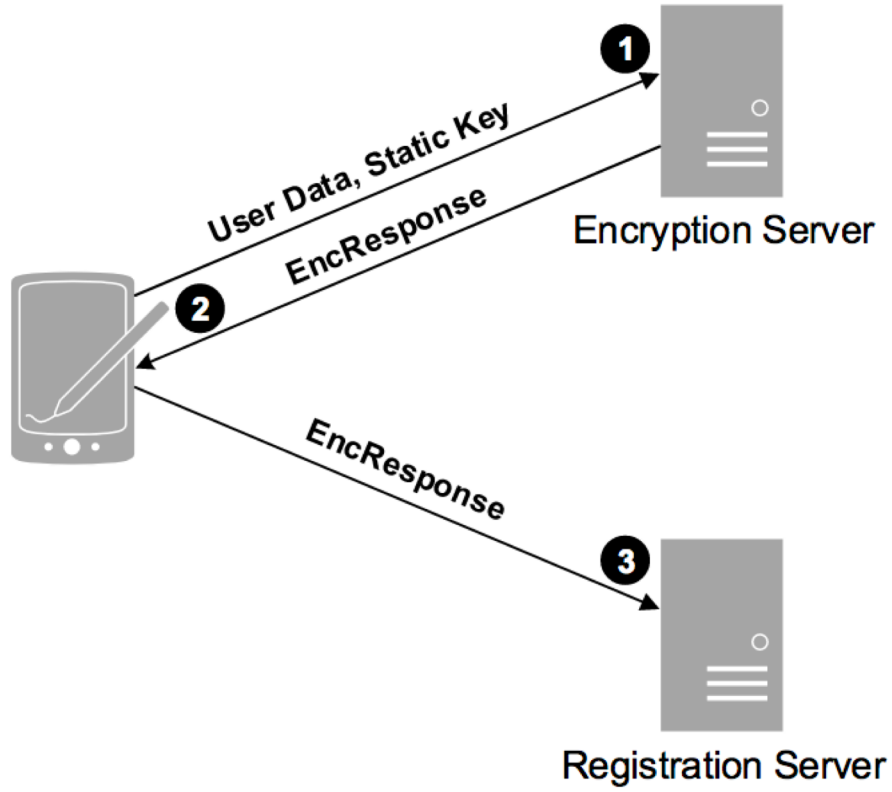


Figure 5: The user registration flow of MoneyOnMobile. All communication is over HTTP.

Author of Linux.Encoder Fails for the Third Time, Ransomware Is Still Decryptable

Lucky Linux server admins are lucky, ransomware is still a dud, fails to properly hide its encryption key

Jan 5, 2016 22:10 GMT · By Catalin Cimpanu · Share:

Security 31

Insane blackhats behind world's most expensive ransomware 'forget' to backup crypto keys

Only Linux victims can decrypt warped $247,000 BlackEnergy module - and then only maybe

Ransomware Developer Asks Security Researcher for Help in Fixing Broken Crypto

By Catalin Cimpanu November 16, 2016 12:55 PM 10

Fabian Wosar, Emsisoft security researcher, is facing a moral dilemma like very few security researchers have faced before.

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING &

WIDE OPEN —

Cryptography failure leads to easy hacking for PlayStation Classic

Plug-and-play hardware lacks even basic functional security for crucial bootrom.

KYLE ORLAND - 12/10/2018, 12:03 PM

" … hackers found that the most sensitive parts of the system are signed and encrypted solely using a key that's embedded on the device itself, rather than with the aid of a private key held exclusively by Sony."

~~Why are **developers** stupid or lazy?~~

How can we make **secure programming** easier?

SOME POSSIBLE SOLUTIONS

- Better languages
- Better APIs
- Better documentation
- More education
- Static, dynamic analysis tools
- Threat modeling / design
- Open source, bug bounties
- Etc.

But how to prioritize, improve effectiveness?

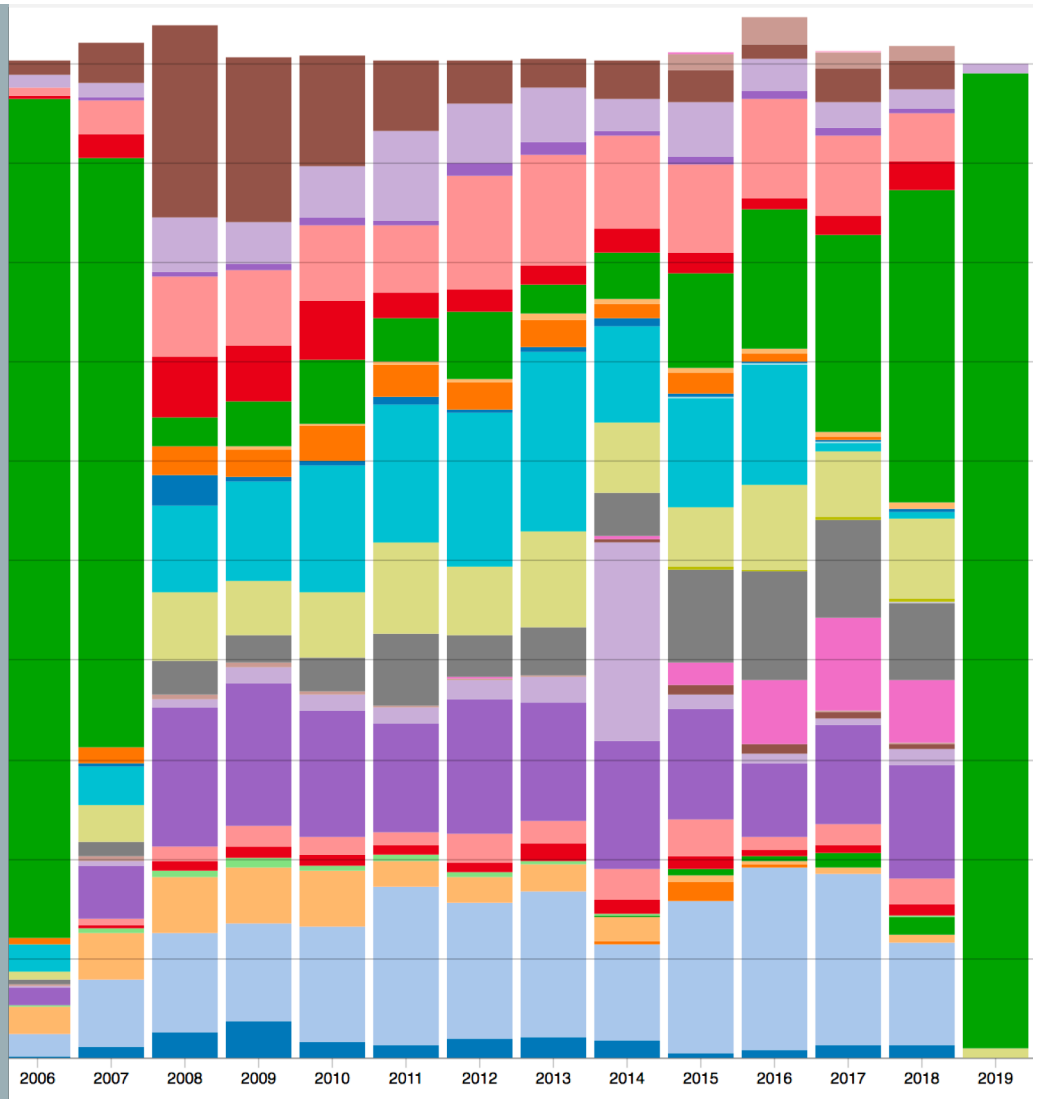
We need to understand causes and prevalence of vulnerabilities.

But measuring this is hard.

1. Field studies

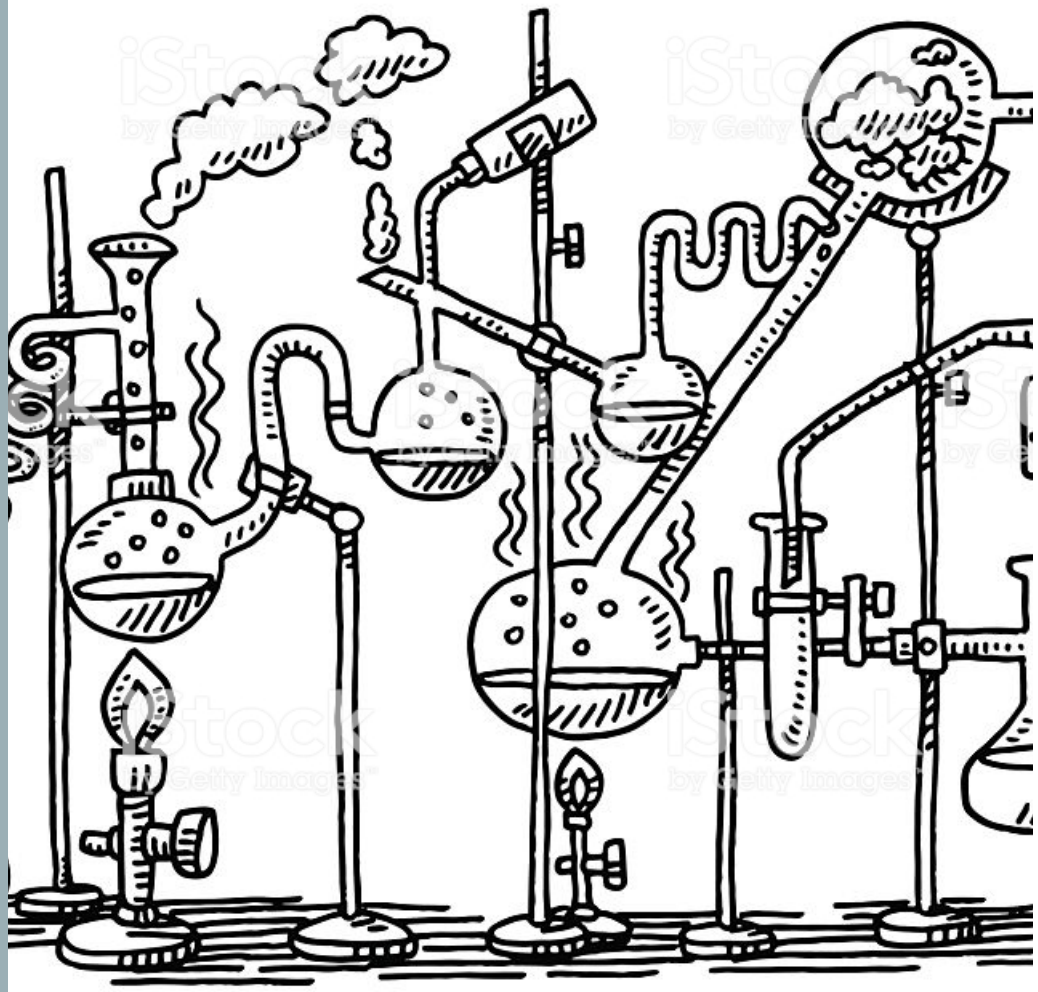
http://s3files.core77.com/blog/images/519972_34481_56003_00AM57OgX.jpg

1. Field studies
2. Field measures (CVEs, etc.)



http://s3files.core77.com/blog/images/519972_34481_56003_00AM57OgX.jpg

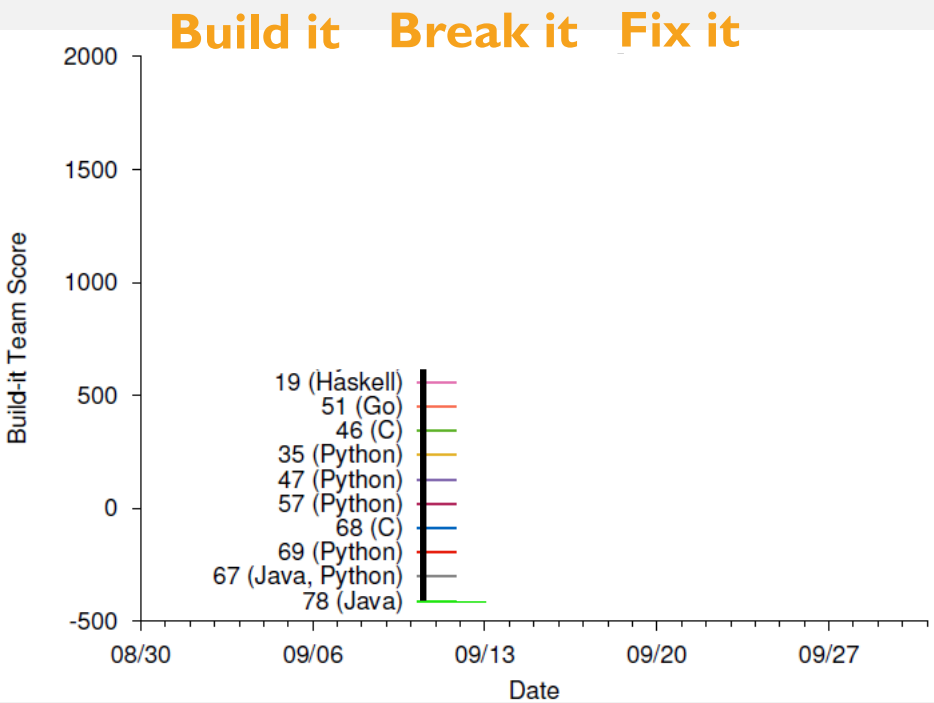
1. Field studies
2. Field measures (CVEs, etc.)
3. Lab studies



http://media.istockphoto.com/vectors/chemistry-experiment-laboratory-drawing-vector-id514323963

BUILD IT, BREAK IT, FIX IT

- Secure development contest
- Build to spec
- Then break other teams
- Incentive design is important!



Ruef et al., CCS 2016

BUILDERS	BREAKERS
Make it performant	Prefer security to correctness
Make it secure	Attack breadth of submissions
	Find unique vulnerabilities

More control than field studies.
More realistic than lab studies.

Result: Rich data about vulnerability introduction.

SECURE LOG PROBLEM

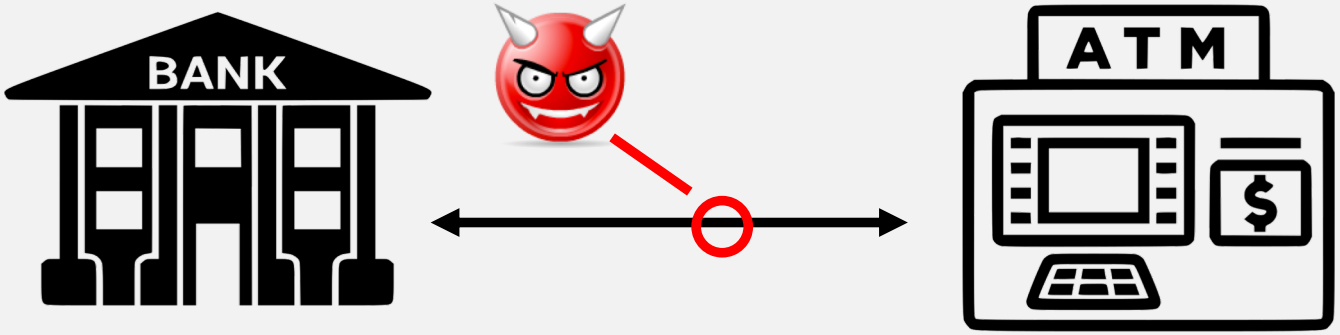
log:

Event Log			
Time	User	Action	Where
8:00 AM	Bob	Enter	Gallery
8:01 AM	Alice	Enter	Office
8:15 AM	Alice	Exit	Office

```
./logappend -T 0800 -K XDFLKJSLJDLJFLKJLSDF -E Bob -A -R Gallery log
./logappend -T 0801 -K XDFLKJSLJDLJFLKJLSDF -E Alice -A -R Office log
./logappend -T 0815 -K XDFLKJSLJDLJFLKJLSDF -E Alice -L -R Office log

./logread -K key -R -E Alice log  Office
```

SECURE COMMUNICATIONS PROBLEM




```
./bank -s auth
```

bob balance: 450

```
./atm -s auth -c card -a bob -n 1000
./atm -s auth -c card -a bob -d 50
./atm -s auth -c card -a bob -w 600
```

auth: XDFLKJSLJDLJFLKJLSDF

card: DFLLKSDF

https://cdn.onlinewebfonts.com/svg/img_449093.png
http://cdn.onlinewebfonts.com/svg/img_456116.png

SECURE DATA SERVER PROBLEM

```
as principal admin password "admin" do
   create principal alice "alices_password"
   set msg = "Hi Alice. Good luck in Build it, Break it, Fix it!"
   set delegation msg admin read -> alice
   return "success"
***

as principal alice password ”alices_password" do
   return msg
***

as principal bob password ”bobs_password" do
   return msg
***
```

https://www.shareicon.net/download/2015/08/14/85119_database_512x512.png

ANALYSIS APPROACH

- Examine each project and each vulnerability in detail
  - Breaker-identified and researcher-identified
- Iterative open and axial coding
  - Two independent coders; high reliability
- 76 projects, more than 800 vulnerabilities
- Qual and quant analysis on resulting categories

VULNERABILITIES

Vuln type
Severity

Chained

Discovery difficulty

Exploit difficulty

PROJECTS

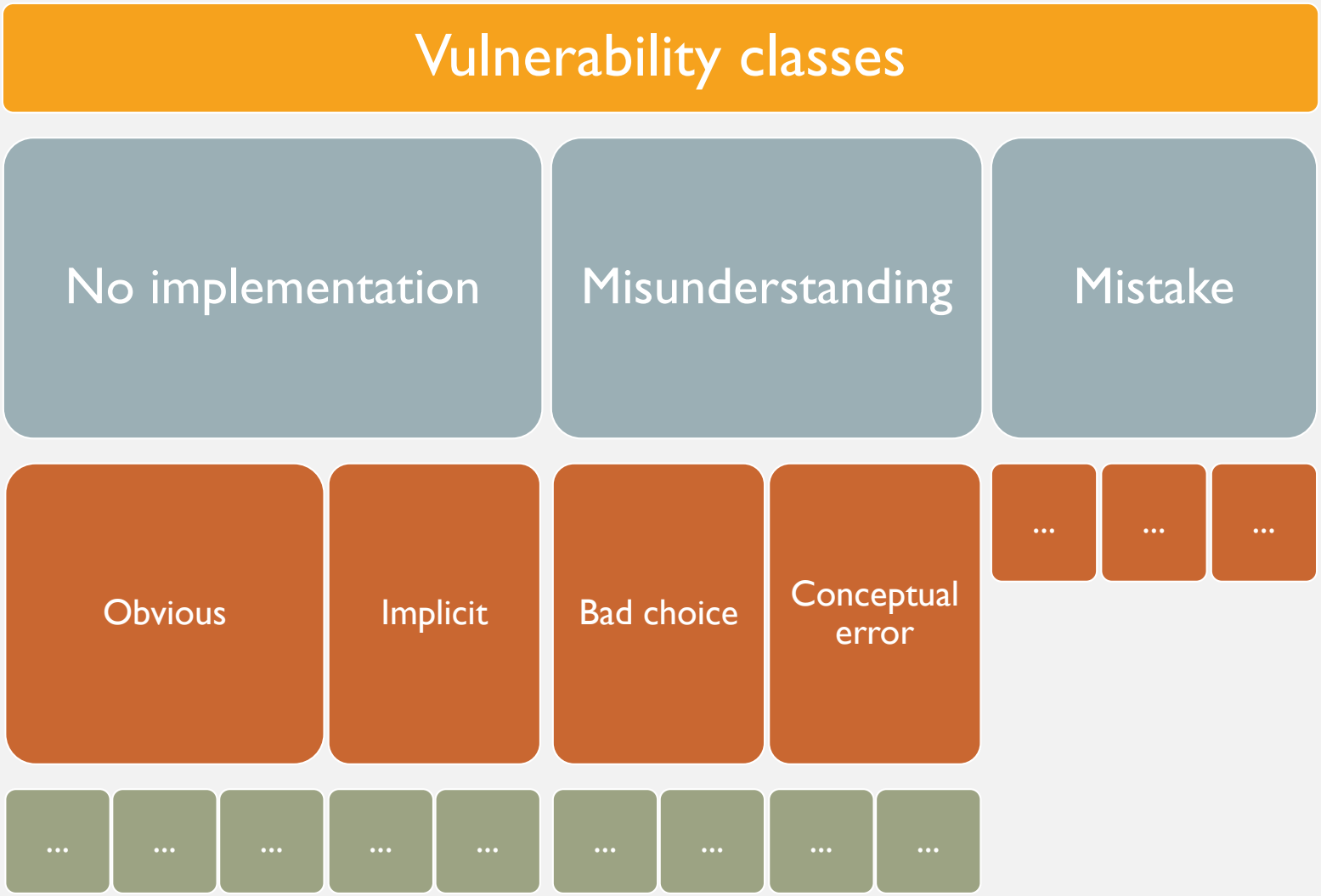
Modularity

Comments

Meaningful var. names

Minimal trust

Economy of mechanism


Vulnerability classes
No implementation
Misunderstanding
Mistake
Obvious
Implicit
Bad choice
Conceptual error
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...

RESULTS

PREVALENCE

Non-attempts >> mistakes

Misunderstandings >> mistakes

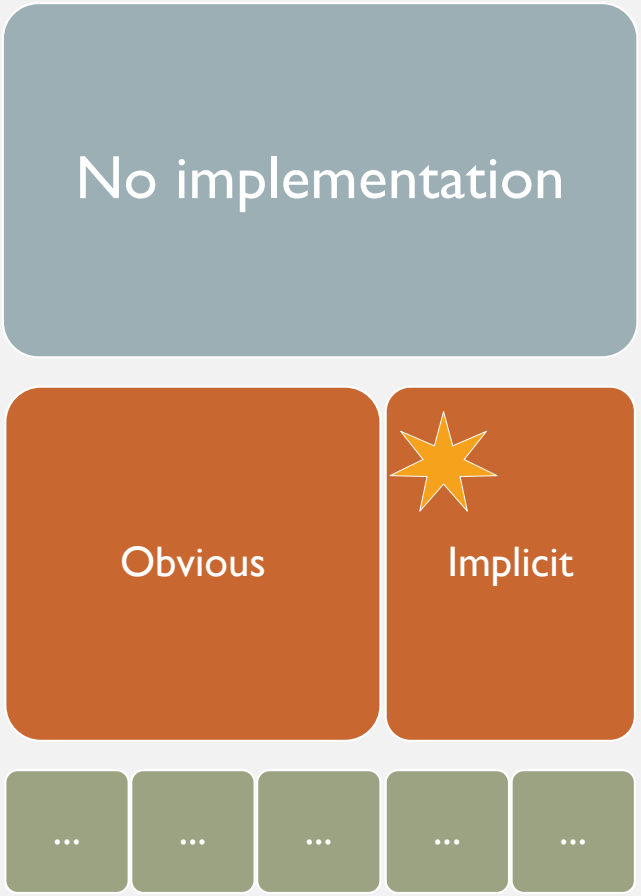
Implicit >> obvious

Concept errors >> bad choices

COMPARING PROBLEMS

- Mistakes most common for secure server, then ATM (problem complexity)
- Implicit issues, concept errors in the ATM problem (lots of unstated requirements, lots of moving parts)
- Bad choices in the secure log problem (why?)

Vulnerability classes



Obvious

- No encryption (log, ATM)
- No access control (server)

Implicit

- Side channels
- No MAC
- No nonce
- No checking delegation chain (server)

Vulnerability classes

- Weak encryption algo (e.g., WEP)
- Unkeyed function
- strcpy

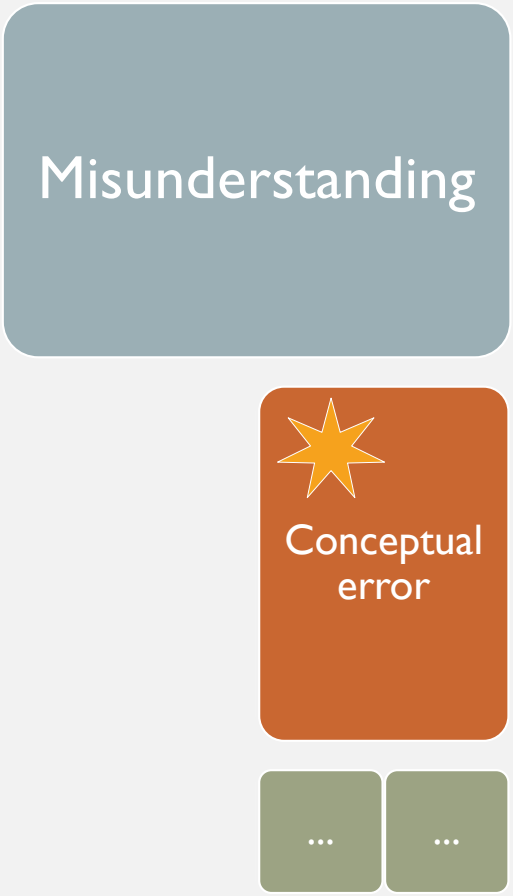
Misunderstanding

Bad choice

... ...

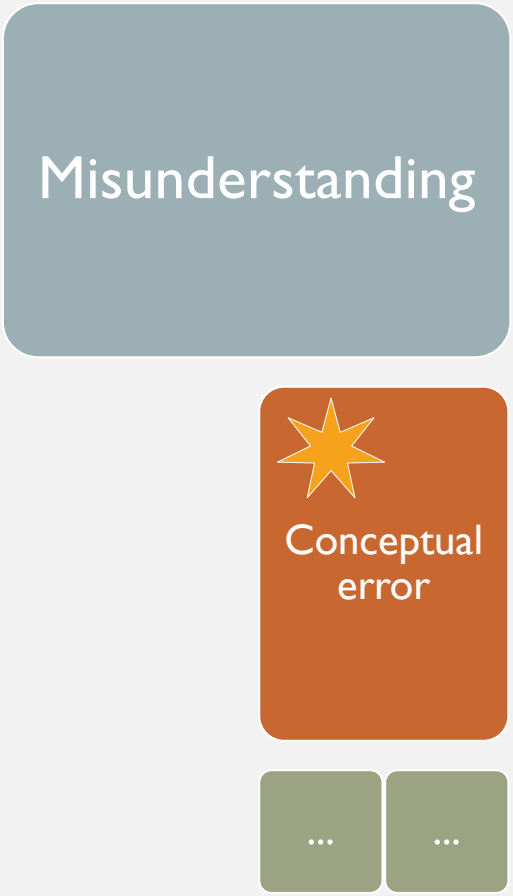
Vulnerability classes

- Subset of necessary
  - MAC only per line
  - MAC of key instead of log data
  - TLS w/o client authentication (ATM)



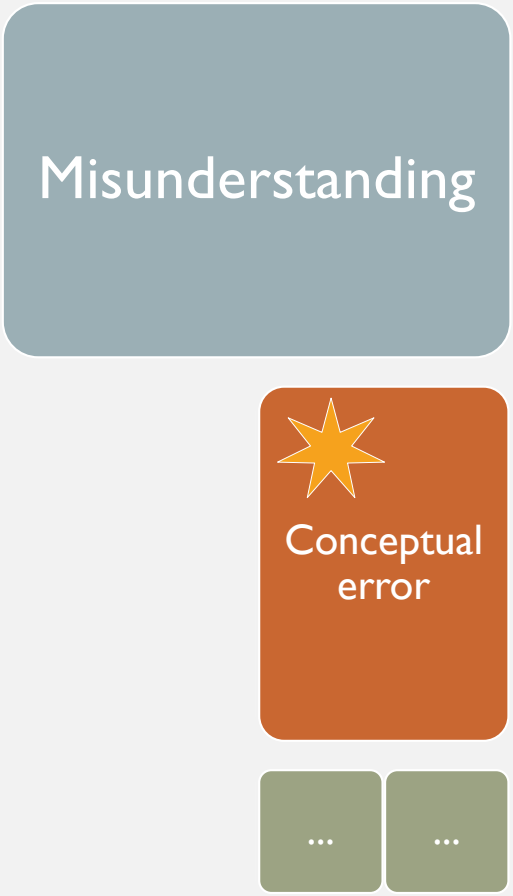
Vulnerability classes

- Misuse of library/API
  - Access control library can't handle loops in delegation list
  - Used SQLCipher but turn off automated MAC



Vulnerability classes

- Fixed instead of random
  - Hardcode key, IV, password



From this site I have this code snippet:

asked 5 years, 2 months ago

```
obj = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
```

```
>>> list(bytearray(ciphertext))
[214, 131, 141, 100, 33, 86, 84, 146, 170, 96, 65, 5, 224, 155, 139, 241]
```

BLOG

Stack Overflow plus bad documentation assumptions … oops.

Vulnerability classes

- Fixed instead of random
  - Hardcode key, IV, password
- Insufficient randomness
  - Nonce overflow
  - IV counts up
  - Nonce timestamp window too large

Misunderstanding

Conceptual error

... ...

Vulnerability classes

- Bad NOT in nested conditionals
- Uncaught exception on replay
- Ignore error from wrong nonce
- Null pointer issues

THINKING ABOUT SOLUTIONS

- Improve abstraction levels in APIs
  - Semantic primitives
- Improve documentation
  - Clarify what you can(not) copy/paste
  - No mysterious error messages
- Tools and automation
  - Wizards, API misuse detection, semantic analysis

MORE ANALYSIS COMING SOON!

- Relating features (modularity, comment quality, language used, etc.) to vulnerability types and quantities
- Factors related to likelihood of vulnerability being found
- Insight into contest incentives/improvements

Understanding developer errors is hard; BIBIFI is one useful design point.

Vulnerabilities arise from nuanced security properties.

Abstractions and documentation matter (and not just in lab studies).

Consider joining our participant panel!

https://ter.ps/SecPros

Michelle Mazurek
University of Maryland

mmazurek@umd.edu

This research supported in part by NSF, NIST, and Google.