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, College Park
“SOLVED” VULNERABILITIES ARE STILL A VERY REAL PROBLEM

Vulnerability Type Change By Year

This visualization is a slightly different view that emphasizes how the assignment of CWEs has changed from year to year.
“SOLVED” VULNERABILITIES ARE STILL A VERY REAL PROBLEM

Vulnerability Type Change By Year

This visualization is a slightly different view that emphasizes how the assignment of CWEs has changed from year to year.
“SOLVED” VULNERABILITIES ARE STILL A VERY REAL PROBLEM

Vulnerability Type Change By Year

This visualization is a slightly different view that emphasizes how the assignment of CWEs has changed from year to year.
“SOLVED” VULNERABILITIES ARE STILL A VERY REAL PROBLEM

Vulnerability Type Change By Year

This visualization is a slightly different view that emphasizes how the assignment of CWEs has changed from year to year.
“SOLVED” VULNERABILITIES ARE STILL A VERY REAL PROBLEM

Vulnerability Type Change By Year

This visualization is a slightly different view that emphasizes how the assignment of CWEs has changed from year to year.
Knock, knock: Digital key flaw unlocks door control systems
“rolling its own crypto rather than relying on tried and tested solutions…. The devices were sending hardcoded encryption keys over the network, and were using a fixed initialization vector… Moreover, the devices didn’t include any message signing”
“rolling its own crypto rather than relying on tried and tested solutions…. The devices were sending hardcoded encryption keys over the network, and were using a fixed initialization vector… Moreover, the devices didn’t include any message signing”
“rolling its own crypto rather than relying on tried and tested solutions….The devices were sending hardcoded encryption keys over the network, and were using a fixed initialization vector…. Moreover, the devices didn’t include any message signing”
“rolling its own crypto rather than relying on tried and tested solutions…. The devices were sending hardcoded encryption keys over the network, and were using a fixed initialization vector… Moreover, the devices didn’t include any message signing”
Knock, knock: Digital key flaw unlocks door control systems

“rolling its own crypto rather than relying on tried and tested solutions….The devices were sending hardcoded encryption keys over the network, and were using a fixed initialization vector… Moreover, the devices didn’t include any message signing”
SBI data leaked: Millions of customers' information exposed from server

Shubham Sharma
NewsBytes 30 January 2019

“SBI's Mumbai-based data center had a server without password protection”
Knock, knock: Digital key flaw unlocks door control systems

05 SEP 2018

SBI data leaked: Millions of customers' information exposed from server

Shubham Sharma
NewsBytes 30 January 2019

Digital sign systems allowed hacker access through default passwords

Updated: A swathe of severe vulnerabilities was found in Tightrope Media Systems' digital signage system.

By Charlie Osborne for Zero Day | February 4, 2019 -- 14:12 GMT (06:12 PST) | Topic: Security
Why do developers continue to make stupid and lazy mistakes?
Why do developers continue to make stupid and lazy mistakes?
Why do developers continue to make stupid and lazy mistakes?

How can we make secure programming easier?
POSSIBLE SOLUTIONS
POSSIBLE SOLUTIONS

- More/Better Education
POSSIBLE SOLUTIONS

- More/Better Education
- Better APIs
POSSIBLE SOLUTIONS

- More/Better Education
- Better APIs
- Better documentation
POSSIBLE SOLUTIONS

- More/Better Education
- Better APIs
- Better documentation
- Automation
POSSIBLE SOLUTIONS

- More/Better Education
- Better APIs
- Better documentation
- Automation
- Etc
POSSIBLE SOLUTIONS

- More/Better Education
- Better APIs
- Better documentation
- Automation
- Etc

How can we improve the effectiveness of these solutions?
IN ORDER TO IMPROVE THESE SOLUTIONS, WE NEED TO UNDERSTAND THE **TYPES, CAUSES, AND PERVERSIVENESS** OF VULNERABILITIES.
HOW CAN WE MEASURE THIS?

- Field studies
- Field surveys
- Lab studies
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data
- Pros:
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data
- Pros:
  - We can see what happens in the real world
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data

- Pros:
  - We can see what happens in the real world

- Cons:
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data

- Pros:
  - We can see what happens in the real world

- Cons:
  - Hard to get access to
FIELD STUDIES

- Immerse ourselves in the “real world” to observe and collect data

- Pros:
  - We can see what happens in the real world

- Cons:
  - Hard to get access to
  - Hard to generalize site specific data
FIELD SURVEYS

- CVEs, GitHub, etc
FIELD SURVEYS

- CVEs, GitHub, etc

- Pros:
FIELD SURVEYS

- CVEs, GitHub, etc
- Pros:
  - Large datasets publicly available
FIELD SURVEYS

- CVEs, GitHub, etc

- Pros:
  - Large datasets publicly available
  - Data is already categorized
FIELD SURVEYS

- CVEs, GitHub, etc

- Pros:
  - Large datasets publicly available
  - Data is already categorized

- Cons:
FIELD SURVEYS

- CVEs, GitHub, etc

- Pros:
  - Large datasets publicly available
  - Data is already categorized

- Cons:
  - Hard to understand why
FIELD SURVEYS

- CVEs, GitHub, etc

- Pros:
  - Large datasets publicly available
  - Data is already categorized

- Cons:
  - Hard to understand why
  - Hard to compare possibly unrelated data
LAB STUDIES

- Have people participate in a controlled experiment
LAB STUDIES

- Have people participate in a controlled experiment

- Pros:
LAB STUDIES

- Have people participate in a controlled experiment

- Pros:
  - A lot of control over conditions
LAB STUDIES

- Have people participate in a controlled experiment

- Pros:
  - A lot of control over conditions

- Cons:
LAB STUDIES

- Have people participate in a controlled experiment

- Pros:
  - A lot of control over conditions

- Cons:
  - Ecological validity
LAB STUDIES

- Have people participate in a controlled experiment

- Pros:
  - A lot of control over conditions

- Cons:
  - Ecological validity
  - Potentially simple problems
BUILD IT, BREAK IT, FIX IT

- Secure programming contest
BUILD IT, BREAK IT, FIX IT

- Secure programming contest
- Build-It Phase
BUILD IT, BREAK IT, FIX IT

- Secure programming contest
- Build-It Phase
  - 2 weeks
BUILD IT, BREAK IT, FIX IT

- Secure programming contest

- Build-It Phase
  - 2 weeks
  - Develop to spec with open choices
BUILD IT, BREAK IT, FIX IT

- Secure programming contest
- Build-It Phase
  - 2 weeks
  - Develop to spec with open choices
  - Incentivized:

Ruef et al., CCS 2016
BUILD IT, BREAK IT, FIX IT

- Secure programming contest
- Build-It Phase
  - 2 weeks
  - Develop to spec with open choices
  - Incentivized:
    - Make it performant
BUILD IT, BREAK IT, FIX IT

- Secure programming contest

- Build-It Phase
  - 2 weeks
  - Develop to spec with open choices

- Incentivized:
  - Make it performant
  - Make it secure
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
  - Attack breadth of submissions
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
  - Attack breadth of submissions
  - Find unique vulnerabilities
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
  - Attack breadth of submissions
  - Find unique vulnerabilities
  - Prioritize security bugs over correctness
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
  - Attack breadth of submissions
  - Find unique vulnerabilities
  - Prioritize security bugs over correctness

- Fix-It Phase
BUILD IT, BREAK IT, FIX IT

- Break-It Phase
  - Get other teams’ source code
  - Attack breadth of submissions
  - Find unique vulnerabilities
  - Prioritize security bugs over correctness

- Fix-It Phase
  - Make fixes and get points back
**SECURE LOG PROBLEM**

<table>
<thead>
<tr>
<th>log:</th>
<th>Event Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>User</td>
</tr>
</tbody>
</table>

17
## SECURE LOG PROBLEM

```
./logappend -T 0800 -K XDFLKJSLJDLJFLKJLSDF -E Bob -A -R Gallery log
```
SECURE LOG PROBLEM

log:

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
</tbody>
</table>

./logappend -T 0800 -K XDFLKJSDLJFLKJLSDF -E Bob -A -R Gallery log
SECURE LOG PROBLEM

**./logappend -T 0800 -K XDFLKJSLJDLJFLKJLSDF -E Bob -A -R Gallery log**

**./logappend -T 0801 -K XDFLKJSLJDLJFLKJLSDF -E Alice -A -R Office log**

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
</tbody>
</table>
SECURE LOG PROBLEM

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
<tr>
<td>8:01 AM</td>
<td>Alice</td>
<td>Enter</td>
<td>Office</td>
</tr>
</tbody>
</table>
SECURE LOG PROBLEM

log:

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
<tr>
<td>8:01 AM</td>
<td>Alice</td>
<td>Enter</td>
<td>Office</td>
</tr>
</tbody>
</table>

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

./logread -K XDFLKJSLJDLJFLKJLSDF -R -E Alice log
SECURE LOG PROBLEM

Event Log

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
<tr>
<td>8:01 AM</td>
<td>Alice</td>
<td>Enter</td>
<td>Office</td>
</tr>
</tbody>
</table>

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

./logread -K XDFLKJSLJDLJFLKJLSDF -R -E Alice log Office
SECURE LOG PROBLEM

log:

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
<tr>
<td>8:01 AM</td>
<td>Alice</td>
<td>Enter</td>
<td>Office</td>
</tr>
</tbody>
</table>

./logappend -T 0800 -K XDFLKJSLJDLJFLKJLSDF -E Bob -A -R Gallery log

./logappend -T 0801 -K XDFLKJSLJDLJFLKJLSDF -E Alice -A -R Office log

./logread -K XDFLKJSLJDLJFLKJLSDF -R -E Alice log Office
SECURE LOG PROBLEM

log:

<table>
<thead>
<tr>
<th>Time</th>
<th>User</th>
<th>Action</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Bob</td>
<td>Enter</td>
<td>Gallery</td>
</tr>
<tr>
<td>8:01 AM</td>
<td>Alice</td>
<td>Enter</td>
<td>Office</td>
</tr>
</tbody>
</table>

./logappend -T 0800 -K XDFLKJSDLJFLKJLSDF -E Bob -A -R Gallery log

./logappend -T 0801 -K XDFLKJSDLJFLKJLSDF -E Alice -A -R Office log

./logread -K XDFLKJSDLJFLKJLSDF -R -E Alice log Office
SECURE COMMUNICATIONS PROBLEM
SECURE COMMUNICATIONS PROBLEM

./bank -s auth
SECURE COMMUNICATIONS PROBLEM

auth: XDFLJSLJDLJFLKJLSDF

./bank -s auth
SECURE COMMUNICATIONS

PROBLEM

/auth: XDFLKJSLJDLJFLKJLSDF
SECURE COMMUNICATIONS

PROBLEM

./bank -s auth

./atm -s auth -c card -a bob -n 1000

auth: XDFLKJSLJDLJFLKJLSDF
SECURE COMMUNICATIONS PROBLEM

.b/bank -s auth

.b/atm -s auth -c card -a bob -n 1000

bob balance: 1000

auth: XDFLKJSLJDLJFLKJLSDF

card: DFLLKSDF
SECURE COMMUNICATIONS PROBLEM

./bank -s auth

bob balance: 1050

./atm -s auth -c card -a bob -n 1000

./atm -s auth -c card -a bob -d 50

auth: XDFLKJSLJDLJFLKJLSDF
card: DFLLKSDSF
SECURE COMMUNICATIONS PROBLEM

/auth: XDFLKJSLJDLJFLKJLSDF  card: DFLLKSDF
SECURE COMMUNICATIONS PROBLEM

auth: XDFLKJSLJDJFLKJLSDF  card: DFLLKSDSF

./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
SECURE COMMUNICATIONS PROBLEM

auth: XDFLKJSLJDLJFLKJLSDF

./bank -s auth

bob balance: 450

/ATM

.card: DFLLKSDSF

./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
MULTIUSER DATABASE 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"
***
MULTIUSER DATABASE PROBLEM

```plaintext
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 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
***
RESEARCH QUESTIONS
RESEARCH QUESTIONS

- What types of vulnerabilities do developers introduce?
RESEARCH QUESTIONS

- What types of vulnerabilities do developers introduce?
- How severe are the vulnerabilities? If exploited, what is the effect on the system?
RESEARCH QUESTIONS

- What types of vulnerabilities do developers introduce?
- How severe are the vulnerabilities? If exploited, what is the effect on the system?
- How exploitable are the vulnerabilities? What level of insight is required and how much work is necessary?
RESEARCH QUESTIONS

- What types of vulnerabilities do developers introduce?
- How severe are the vulnerabilities? If exploited, what is the effect on the system?
- How exploitable are the vulnerabilities? What level of insight is required and how much work is necessary?
ANALYSIS APPROACH
ANALYSIS APPROACH

- Examine projects and associated exploits in detail
ANALYSIS APPROACH

- Examine projects and associated exploits in detail
- Iterative open coding
ANALYSIS APPROACH

- Examine projects and associated exploits in detail
- Iterative open coding
  - Two independent researchers with high reliability
ANALYSIS APPROACH

- Examine projects and associated exploits in detail
- Iterative open coding
  - Two independent researchers with high reliability
- 76 projects with 866 submitted exploits
ANALYSIS APPROACH

- Examine projects and associated exploits in detail
- Iterative open coding
  - Two independent researchers with high reliability
- 76 projects with 866 submitted exploits
- Both qualitative and quantitative analysis performed
RESULTS
Vulnerability classes

- No implementation
- Misunderstanding
- Mistake

- Intuitive
- Unintuitive
- Bad Choice
- Conceptual Error
Vulnerability classes

No implementation
Vulnerability classes

- Missed something “Intuitive”

No implementation

Intuitive
Vulnerability classes

No implementation

- Missed something “Intuitive”
- No encryption (log, ATM)

Intuitive
Vulnerability classes

No implementation

- Missed something “Intuitive”
- No encryption (log, ATM)
- No access control (MD)

Intuitive
Vulnerability classes

- Missed something “Intuitive”
- No encryption (log, ATM)
- No access control (MD)
- Missed something “Unintuitive”
- No MAC (log)
Vulnerability classes

- Missed something “Intuitive”
  - No encryption (log, ATM)
  - No access control (MD)
- Missed something “Unintuitive”
  - No MAC (log)
  - Side-channel leakage (ATM, MD)
Vulnerability classes

- Missed something “Intuitive”
  - No encryption (log, ATM)
  - No access control (MD)
- Missed something “Unintuitive”
  - No MAC (log)
  - Side-channel leakage (ATM, MD)
  - No replay prevention (ATM)
Vulnerability classes

Misunderstanding
• Made a “Bad Choice”

Vulnerability classes
Vulnerability classes

- Made a “Bad Choice”
- Weak algorithms (log, ATM)
Vulnerability classes

- Made a “Bad Choice”
  - Weak algorithms (log, ATM)
  - Homemade encryption (log, ATM)
• Made a “Bad Choice”
  • Weak algorithms (log, ATM)
  • Homemade encryption (log, ATM)
  • strcpy (log, ATM, MD)
Vulnerability classes

- Made a “Conceptual Error”
Vulnerability classes

- Made a “Conceptual Error”
- Fixed value (log, ATM, MD)
```python
def fillercrypter(sharedkey, text):
    ...
    encryption_suite = AES.new(sharedkey,
        AES.MODE_CBC, 'This is an IV456')
    ...
```
```python
def fillercrypter(sharedkey, text):
    ...  
    encryption_suite = AES.new(sharedkey,
        AES.MODE_CBC, 'This is an IV456')
    ...
```
```python
1 def fillercrypter(sharedkey, text):
2     ...
3     encryption_suite = AES.new(sharedkey,
4         AES.MODE_CBC, 'This is an IV456')
5     ...
```

From [this](https://example.com) site I have this code snippet:

```python
>>> from Crypto.Cipher import AES
>>> obj = AES.new('This is a key123', AES.MODE_CBC, 'This is an IV456')
>>> message = "The answer is no"
>>> ciphertext = obj.encrypt(message)
>>> list(bytewater(ciphertext))
[214, 131, 141, 100, 33, 86, 84, 146, 170, 96, 65, 5, 224, 155, 139, 241]
```
Vulnerability classes

- Made a “Conceptual Error”
  - Fixed value (log, ATM, MD)
Vulnerability classes

- Made a “Conceptual Error”
  - Fixed value (log, ATM, MD)
  - Lacking sufficient randomness (log, ATM)

Misunderstanding

Bad Choice

Conceptual Error
Vulnerability classes

- Made a “Conceptual Error”
  - Fixed value (log, ATM, MD)
  - Lacking sufficient randomness (log, ATM)
  - Disabling protections in library (log)
self.db = self.sql.connect(filename, timeout=30)
self.db.execute('pragma key="" + token + "";';)
self.db.execute('PRAGMA kdf_iter=
    + str(Utis.KDF_ITER) + ";;");
self.db.execute('PRAGMA cipher_use_MAC = OFF;';)
...
self.db = self.sql.connect(filename, timeout=30)
self.db.execute('pragma key="" + token + "";';)
self.db.execute('PRAGMA kdf_iter=
    + str_Utils.KDF_ITER) + "";';)
self.db.execute('PRAGMA cipher_use_MAC = OFF;';)
...
• Made a “Mistake”
Vulnerability classes

- Made a “Mistake”
- Control flow mistake (ATM, MD)
Vulnerability classes

• Made a “Mistake”
  • Control flow mistake (ATM, MD)
  • Skipped algorithmic step (ATM, MD)
Vulnerability classes

• Made a “Mistake”
  • Control flow mistake (ATM, MD)
  • Skipped algorithmic step (ATM, MD)

```python
def checkReplay(nonce, timestamp):
    # First we check for timestamp delta
    dateTimeStamp = datetime.strptime(timestamp, 
        '%Y-%m-%d %H:%M:%S.%f')
    deltaTime = datetime.utcnow() - dateTimeStamp
    if deltaTime.seconds > MAX_DELAY:
        raise Exception("ERROR: Expired nonce ")
    # The we check if it is in the table
    global bank
    if (nonce in bank.nonceData):
        raise Exception("ERROR: Reinyected package")
```
Vulnerability classes

- Made a “Mistake”
  - Control flow mistake (ATM, MD)
  - Skipped algorithmic step (ATM, MD)

```python
def checkReplay(nonce, timestamp):
    # First we check for timestamp delta
    dateTimeStamp = datetime.strptime(timestamp,
                                          '%Y-%m-%d %H:%M:%S.%f')
    deltaTime = datetime.utcnow() - dateTimeStamp
    if deltaTime.seconds > MAX_DELAY:
        raise Exception("ERROR:Expired nonce ")
    # The we check if it is in the table
    global bank
    if (nonce in bank.nonceData):
        raise Exception("ERROR:Reinjected package")
```
PREVALENCE

Percentage of projects that introduced a mistake, misunderstanding, and no implementation vulnerability grouped by problem:
PREVALENCE

Percentage of projects that introduced a mistake, misunderstanding, and no implementation vulnerability grouped by problem:
PREVALENCE

Percentage of projects that introduced a mistake, misunderstanding, and no implementation vulnerability grouped by problem:

![Bar chart showing prevalence of projects with different types of mistakes and misunderstandings across various vulnerability classes.]

- No Impl.
- Misund.
- Mistake

Vulnerability Class
- ATM
- EHR
- Gallery
- Total

Secure log
Secure communication
Multiuser database
Totals

% of projects: 0, 20, 40, 60, 80

80
60
40
20
0
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unituitive
- Bad choice
- Concept error

<table>
<thead>
<tr>
<th>% of Projects</th>
<th>0</th>
<th>14</th>
<th>28</th>
<th>41</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Impl.</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misund.</td>
<td></td>
<td></td>
<td>28</td>
<td>41</td>
<td>55</td>
</tr>
</tbody>
</table>

36
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unituitive
- Bad choice
- Concept error

No Impl.
Misund.

% of projects

0 14 28 41 55
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unituitive
- Bad choice
- Concept error

<table>
<thead>
<tr>
<th>% of projects</th>
<th>Intuitive</th>
<th>Unituitive</th>
<th>Bad choice</th>
<th>Concept error</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Impl.</td>
<td>0</td>
<td>36</td>
<td>28</td>
<td>55</td>
</tr>
<tr>
<td>Misund.</td>
<td>14</td>
<td></td>
<td>41</td>
<td></td>
</tr>
</tbody>
</table>

PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unituitive
- Bad choice
- Concept error

<table>
<thead>
<tr>
<th>% of Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

No Impl. Misund.
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unituitive
- Bad choice
- Concept error

No Impl.
Misund.

% of projects
0 14 28 41 55

36
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unintuitive
- Bad choice
- Concept error

No Impl. Misund.
PREVALENCE

% of Projects that introduced each subclass

- Intuitive
- Unintuitive
- Bad choice
- Concept error

No Impl. Misund.
TRENDS WITHIN MISTAKES
TRENDS WITHIN MISTAKES

- Complexity breeds mistakes.
TRENDS WITHIN MISTAKES

- Complexity breeds mistakes.
- Most common in the multi-user database problem (most complex) and least common in log problem (least complex)
TRENDS WITHIN MISTAKES

- Complexity breeds mistakes.
- Most common in the multi-user database problem (most complex) and least common in log problem (least complex)
- Writing security checks once reduced mistakes
TRENDS WITHIN MISTAKES

- Complexity breeds mistakes.
  - Most common in the multi-user database problem (most complex) and least common in log problem (least complex)
  - Writing security checks once reduced mistakes

Almost all mistakes were found in the Break-It phase
RECOMMENDATIONS
RECOMMENDATIONS

- Simplify API design
  - Build in security primitives and focus on common use-cases
RECOMMENDATIONS

- Simplify API design
  - Build in security primitives and focus on common use-cases
- Indicate security impact of non-default use in API Documentation
  - Explain the negative effects of turning off certain things
RECOMMENDATIONS

- Simplify API design
  - Build in security primitives and focus on common use-cases

- Indicate security impact of non-default use in API Documentation
  - Explain the negative effects of turning off certain things

- Vulnerability Analysis Tools
  - More emphasis on design-level conceptual issues
SUMMARY
SUMMARY

- Developers struggle with security concepts
SUMMARY

- Developers struggle with security concepts
  - Mostly knew they needed security and picked the right tools
SUMMARY

- Developers struggle with security concepts
  - Mostly knew they needed security and picked the right tools
  - Didn’t know all the security requirements (Unintuitive) or all the implementation details (Conceptual Error)
SUMMARY

- Developers struggle with security concepts
  - Mostly knew they needed security and picked the right tools
  - Didn’t know all the security requirements (Unintuitive) or all the implementation details (Conceptual Error)
- Mistakes happen, but can be reduced through code review and best practices
SUMMARY

- Developers struggle with security concepts
  - Mostly knew they needed security and picked the right tools
  - Didn’t know all the security requirements (Unintuitive) or all the implementation details (Conceptual Error)
- Mistakes happen, but can be reduced through code review and best practices
- Improve API design, documentation, and automation to handle conceptual nuances
Developers struggle with security concepts

- Mostly knew they needed security and picked the right tools
- Didn’t know all the security requirements (Unintuitive) or all the implementation details (Conceptual Error)

Mistakes happen, but can be reduced through code review and best practices

Improve API design, documentation, and automation to handle conceptual nuances