INTERACTIVE DEVELOPER SUPPORT FOR APPLICATION SECURITY

Helping developers to interactively detect, understand, and mitigate security vulnerabilities in their code.

Heather Richter Lipford
University of North Carolina at Charlotte

Software Security

Being resistant to malicious attacks which exploit security bugs in software.

Acunetix Vulnerability Testing Report 2017

POSTED ON JUNE 6, 2017 BY IAN MUSCAT

With Cross-site Scripting (XSS) vulnerabilities found on **50% of sampled targets**, this year's findings continue to reaffirm the widely held understanding that the web application vector is a major, viable and low-barrier-to-entry vector for attackers.

Hackers still exploiting eBay's stored XSS vulnerabilities in 2017

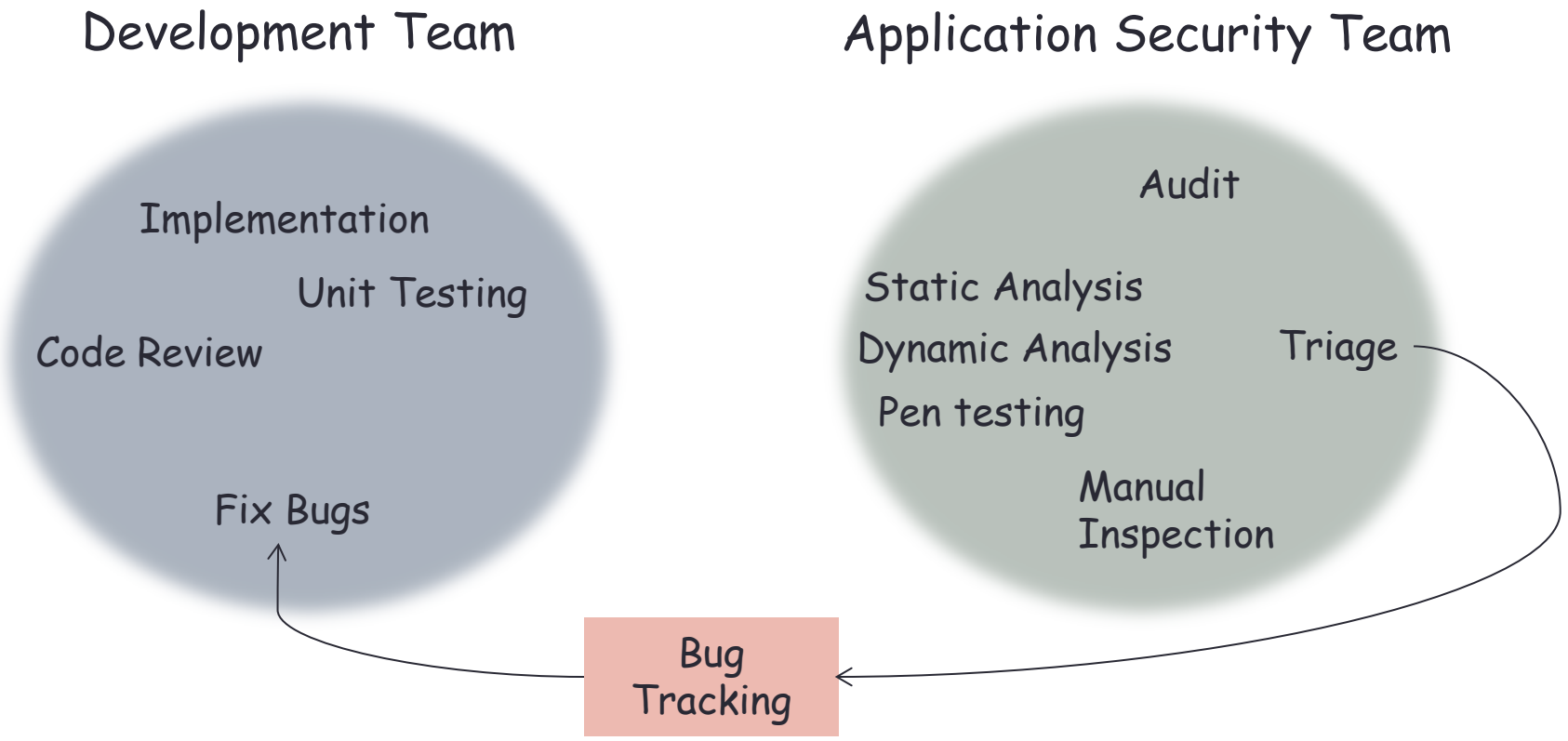
CYBERSECURITY/DATA

5 major cyber security vulnerabilities from the last few days

Data | Charlotte Henry 08:33, November 19 2015

SQL Injection Vulnerability in NextGEN Gallery for WordPress

Application Security Process



Tyler Thomas, Heather Lipford, and Bill Chu. **From the eyes and ears of auditors: An investigation of security during application development.** In preparation.

Out of the security loop

Developers rely on other parties/processes who they believe should handle all security concerns.

Developers believed that security issues do not apply to their particular development context.

A range of factors motivate and constrain attention paid to security concerns.

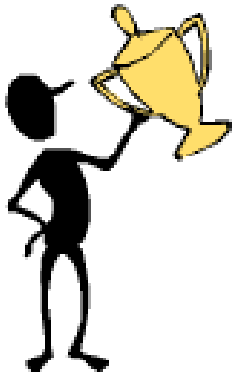
Disconnect between general security knowledge and concrete secure programming practices

Jing Xie, Heather Lipford, and Bill Chu. **Why do programmers make security errors?** In Proc. VL/HCC, 2011.

Communicating with developers

"I think a lot of what I deal with is not necessarily the technology, it's the how you communicate it effectively to developers. My boss likes to joke that we are 60 percent psychologists and 40 percent security professionals."

- **Convince** developers there is a real problem
- **Motivate** stakeholders to fix the vulnerability
- **Explain** how to remediate
- **Train** for the future



Security champion

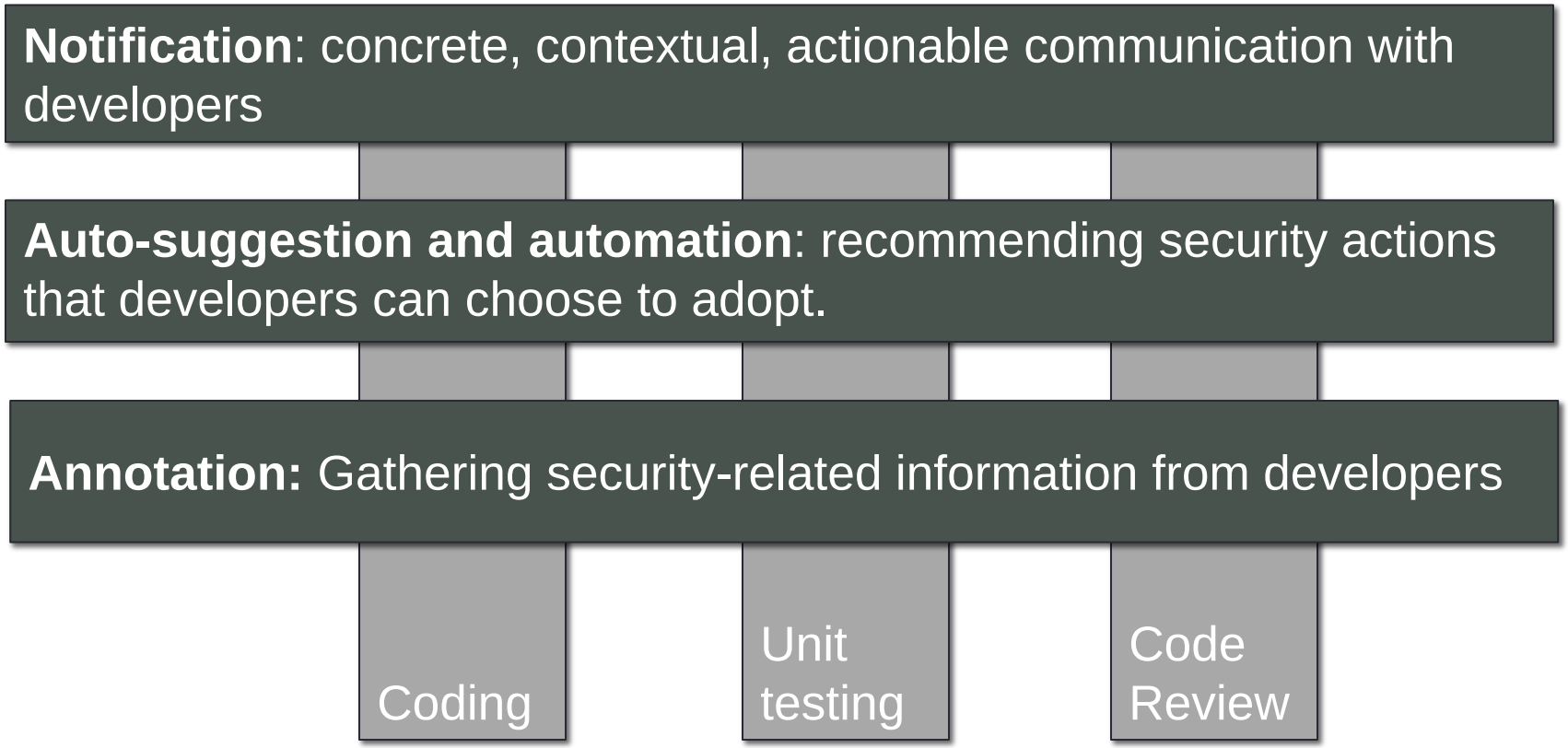
Member of the development team serving as an important liaison and advocate for security

Moving security “to the left”

“I would say that the big challenge there is that security and development have traditionally been disjointed and they have been separate teams. Security is the watchdogs, development does the work and all security has ever done is scan stuff. So write code, ask questions later. And we have to change that.”

- Engage developers with security within their existing tools and processes.
- Reduce the burden of security work by communicating based on their knowledge, and utilizing automated support as much as possible.
- Utilize developer knowledge of the application context to customize security tool support.

Interactions and Touchpoints

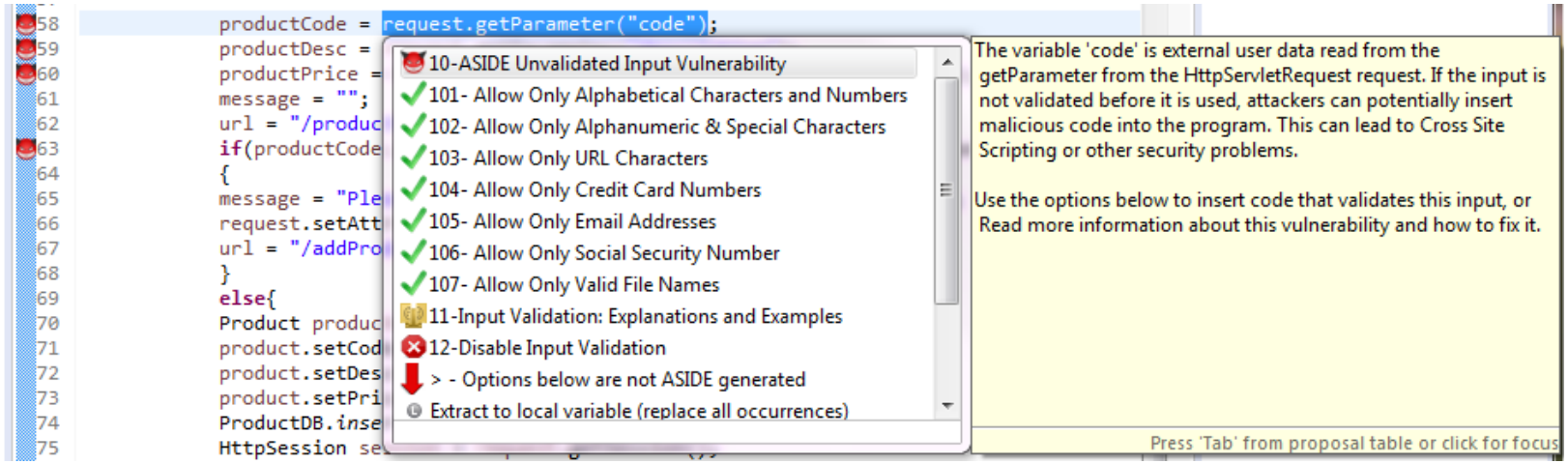


Touchpoint #1: Coding

Interactive Static Analysis

Increase developers' **awareness** of security vulnerabilities, and their secure programming **knowledge** and **behavior**.

- Vulnerabilities related to lack of input validation, output encoding, or SQL injection
- Access control vulnerabilities

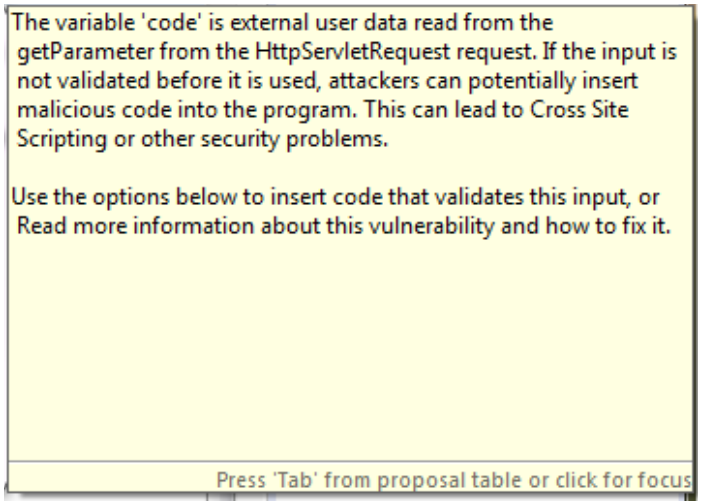


ASIDE: Application Security in the IDE

Interaction: Notification

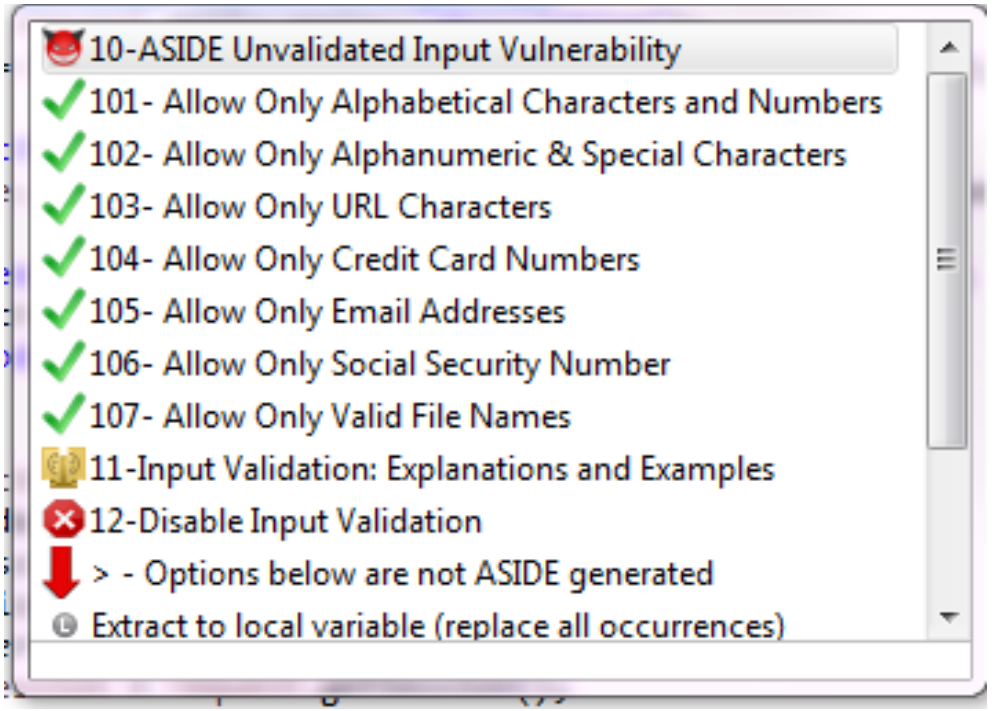
Warnings and messages need to be:

- **Understandable** by developers with range of security expertise
- **Consistent** in their structure and presentation
- **Contextual** descriptions based upon the relevant lines of code
- **Actionable** guidance regarding how to trace and resolve the vulnerability



Interaction: Auto-suggestion

- Automatically insert sanitization code based on the type of input or output chosen
- Uses ESAPI validation methods

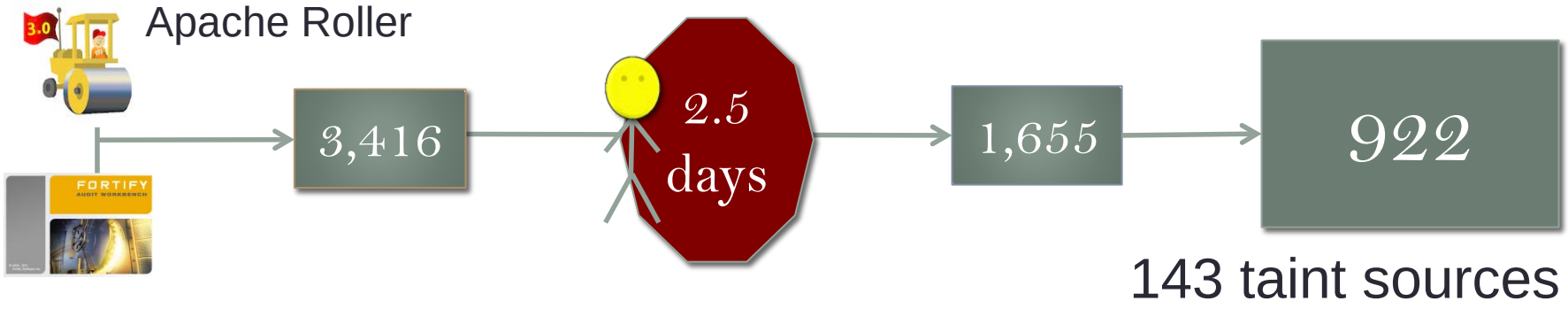


Interaction: Annotation

```
45          String accountName = request.getParameter("AccountName");
46          AccountMapper accounts = getAccounts();
47
48          if (((User) request.getSession().getAttribute("USER")).ownAccount(accountName))
49          {
50              accounts.updateAccount(accountName, 0);
51          }
52          else
53          {
54
55          }
```

```
45          String accountName = request.getParameter("AccountName");
46          AccountMapper accounts = getAccounts();
47
48          if (((User) request.getSession().getAttribute("USER")).ownAccount(accountName))
49          {
50              accounts.updateAccount(accountName, 0);
51          }
52          else
53          {
54
55          }
```

Security Performance – Injection,XSS



- 131/143 (92%) taint sources identified
- Remainder due to JSP/frameworks not yet supported

- 118 additional taint sources identified
- 94 potentially exploitable
- 24 false positives

Jing Xie, Bill Chu, and Heather Lipford. **ASIDE: IDE Support for Web Application Security**. In Proc. ACSAC, 2011.

Security Performance – Access Control

- Comparison against known access control vulnerabilities in 6 open source projects
  - 26 known and 20 zero-day vulnerabilities detected

Project	Known Vul.	Known Vul. By ASIDE-PHP	0-day Vul. by ASIDE-PHP
Moodle 2.1.0	6[I], 7[L]	6[I]	1[I]
Mybloggie 2.1.3	3[M], 3[UT]	3[M], 3[UT]	15[M]
SCARF 1.0	1[M], 10[UT]	1[M], 10[UT]	
Bilboblog 0.2.1	1[UT]	1[UT]	
Wheatblog 1.1	1[M]	1[M]	
PhpStat 1.5	1[M]	1[M]	4[M]

M: Missing check I: Inconsistent
UT: Untrusted data L: Logic error

Jun Zhu, Bill Chu, Heather Lipford, and Tyler Thomas. **Mitigating Access Control Vulnerabilities through Interactive Static Analysis**. In Proc. SACMAT 2015.

User Behavior

Multiple user studies with advanced students, and two with professionals

The good:

- Raised awareness
- Almost all correct actions
- Liked the quick fixes and help

"Here I would like to see numbers only, but I don't quite see an option for that. I would probably go ahead and activate the letters and numbers quickfix and then modify it so that it's just numbers."

Needs improvement:

- Vulnerability severity and ranking
- Customize fixes

Tyler Thomas et. al **What Questions Remain? An Examination of How Developers Understand an Interactive Static Analysis Tool**. In Workshop on Security Information Workers, SOUPS 2016.

User Behavior

ASIDE is unobtrusive and informative.

Automated code generation valuable.

Successful identification of access control logic, but difficulty tracing an access control vulnerability.

Needs more examples of exploits and severity of risks.

ASIDE increases awareness of security vulnerabilities, could improve the practice of secure programming.

Jing Xie, Heather Lipford, and Bill Chu. **Evaluating Interactive Support for Secure Programming**. CHI, 2012.
Tyler Thomas et al. **A Study of Interactive Code Annotation for Access Control Vulnerabilities.** VL/HCC, 2015.

Touchpoint #2: Unit Testing

Dynamic analysis to detect vulnerabilities not found in static analysis

- Detect Cross-Site Scripting (XSS) vulnerabilities due to improper encoding of untrusted data.
  - **Automated** construction and evaluation of XSS unit tests.
  - **Notification** of exact line number of vulnerable code

Minimize false positives by confirming vulnerabilities via execution.

Minimize false negatives by systematically generating attack strings.

M. Mohammadi et. al **Detecting Cross-Site Scripting Vulnerabilities through Automated Unit Testing**. IEEE QRS. 2017.
M. Mohammadi et. al **Automatic Web Security Unit Testing: XSS Vulnerability Detection**. AST Workshop at ICSE. 2016.

Interaction: Automation

- Unit Test Extraction
  - Control flow analysis to slice the code into execution paths
  - Taint analysis to determine injection points of untrusted data
- Attack Generation
  - Model “context switching” as a context free grammar
  - Generate attack strings using sentences of the grammar
- Attack Evaluation
  - Execute attack strings in JWebUnit
  - Change Web page title with line number being tested

	Vulnerabilities reported	True Positive	False Positive
ZAP	31	9	22
XSS Unit Testing	17	17	0

Evaluation on iTrust, open source medical records application

Touchpoint #3: Code Review

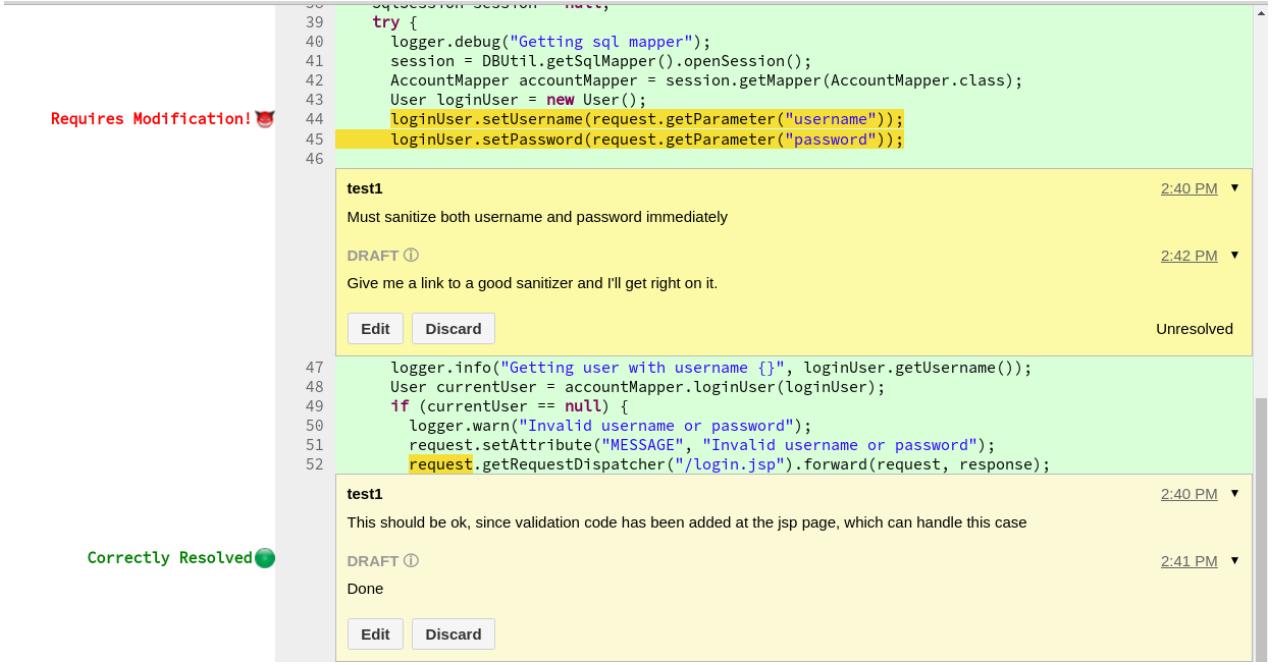
Security-oriented code review with static analysis

- Interaction between developers and application security experts earlier in the process
- Use static analysis (for example, ASIDE) to focus reviews around detected and mitigated vulnerabilities

- How can we support communication between developers and application security experts?
- How effective is code review for improving developer awareness and practice of application security?

Interaction: Annotation

- Extending Gerrit, an open-source code review tool



- Reviewers rate vulnerabilities, leave comments
- Developers ask questions and receive feedback

Interactions and Touchpoints

	Coding	Unit testing	Code review
Notification	Vulnerable practices	Identify vulnerable code based on test results	*Security decisions, e.g. incorrect remediation*
Auto-suggest	Mitigation controls, e.g. input validation code	Security unit tests for XSS vulnerability detection	*Prioritizing issue review*
Annotation	Security decisions, e.g. access control	*Seek developer input to help security unit test generation*	*Comments and rationale of security decisions.*

Research Challenges

- Reducing false positives
  - Understanding what developers interpret as a false positive
  - Focus on techniques and tools with low false positives
  - Utilize contextual information to improve accuracy
- Reducing false negatives
  - Characterize false negatives and remaining risk
- Reflect risk assessments
  - Prioritize tool feedback and functionality based upon risk
- Incentives
  - Understand tool features that discourage interaction
  - Understand costs versus benefits of tools and change in process
  - Organizational structures and processes that motivate security work

Acknowledgements

Supported by NSF #1318854, 1044745, 1523041

Collaborators:

- Bill Chu
- Mahmoud Mohammadi, Madiha Tabassum, Tyler Thomas, Jing Xie, and Jun Zhu
- Emerson Murphy-Hill and Justin Smith

http://aside.uncc.edu