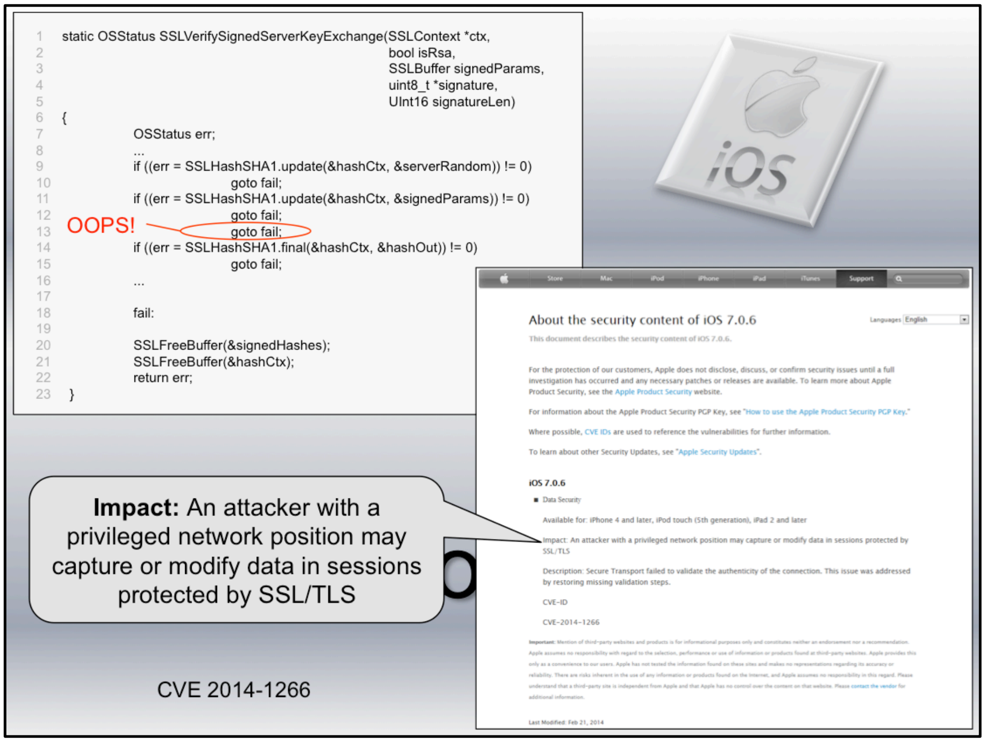


```
static OSStatus SSLVerifySignedServerKeyExchange(SSLContext *ctx,
                                                 bool isRsa,
                                                 SSLBuffer signedParams,
                                                 uint8_t *signature,
                                                 UInt16 signatureLen)
{
        OSStatus err;
        ...
        if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                goto fail;
        if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                goto fail;
                goto fail;
        if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                goto fail;
        ...

        fail:

        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
 }
```

Impact: An attacker with a privileged network position may capture or modify data in sessions protected by SSL/TLS

CVE 2014-1266

About the security content of iOS 7.0.6

This document describes the security content of iOS 7.0.6.

For the protection of our customers, Apple does not disclose, discuss, or confirm security issues until a full investigation has occurred and any necessary patches or releases are available. To learn more about Apple Product Security, see the Apple Product Security website.

For information about the Apple Product Security PGP Key, see "How to use the Apple Product Security PGP Key."

Where possible, CVE IDs are used to reference the vulnerabilities for further information.

To learn about other Security Updates, see "Apple Security Updates".

iOS 7.0.6

- Data Security

Available for: iPhone 4 and later, iPod touch (5th generation), iPad 2 and later

Impact: An attacker with a privileged network position may capture or modify data in sessions protected by SSL/TLS

Description: Secure Transport failed to validate the authenticity of the connection. This issue was addressed by restoring missing validation steps.

CVE-ID

CVE-2014-1266

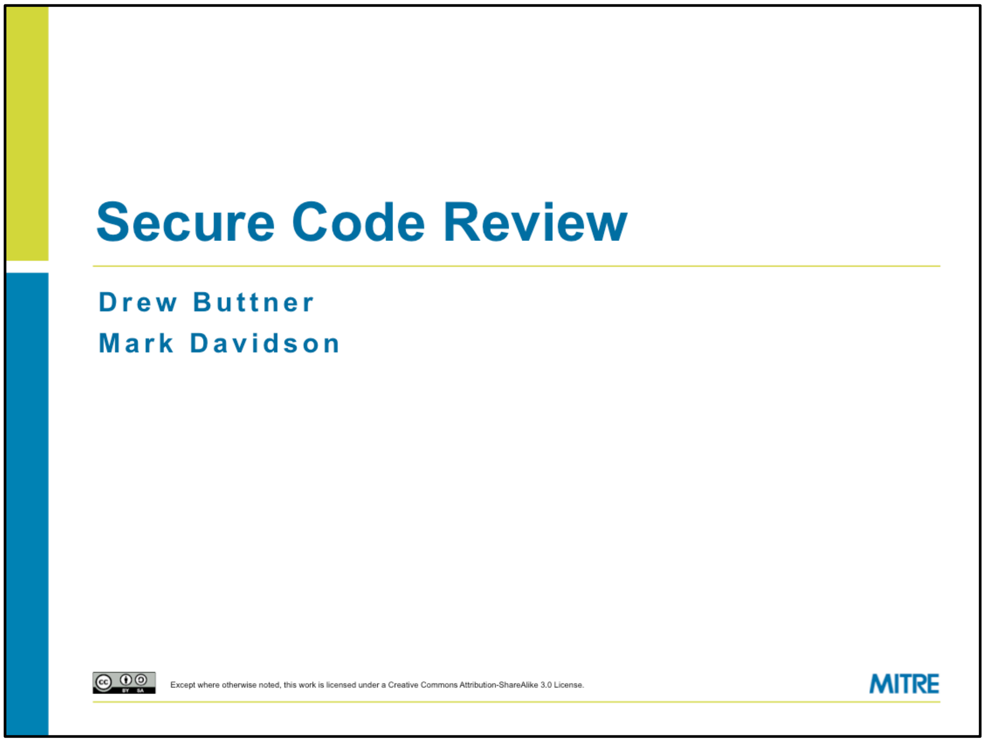
Important: Mention of third-party websites and products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Apple assumes no responsibility with regard to the selection, performance or use of information or products found at third-party websites. Apple provides this only as a convenience to our users. Apple has not tested the information found on these sites and makes no representations regarding its accuracy or reliability. There are risks inherent in the use of any information or products found on the Internet, and Apple assumes no responsibility in this regard. Please understand that a third-party site is independent from Apple and that Apple has no control over the content on that website. Please contact the vendor for additional information.

Last Modified: Feb 21, 2014

Before getting started with the course, we want to talk about a interesting story that came out of Apple in February related to code in iOS 7. The issue became public while we were working on this course and our immediate thought was how a code review could have played a role in helping identify this issue before it found its way into production code.

When a browser on your iPhone makes an SSL/TLS request to a website, the website presents a cryptographic "certificate" chain identifying itself and the authority which issued the certificate. Your device already has a list of issuing authorities which are trusted, and it will check the name of the site and the certificate it presents with that authority. If an invalid or fake certificate is provided, (e.g., one that has the wrong name for the site, or which hasn't been issued by the authority, or which is out of date) then the browser won't trust it and you will get a warning saying that there's something wrong and that you shouldn't proceed or your data could be at risk. To fully trust a site, it is important to verify the authenticity of the certificate.

Part of the validation code during a SSL/TLS key exchange in iOS is shown here on the slide. This code goes through a number of checks against the certificate that was provided. If any of them fail, then it jumps down to the end and returns the failed result. On Feb 21 Apple released a security alert and provide an update to iOS. The alert didn't give many details but it quickly caught the interest of the security community.



Secure Code Review

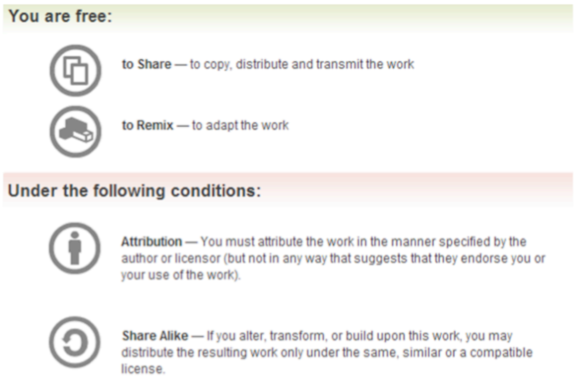
Drew Buttner

Mark Davidson

MITRE

All materials is licensed under a Creative Commons “Share Alike” license.

http://creativecommons.org/licenses/by-sa/3.0/



You are free:

to Share — to copy, distribute and transmit the work

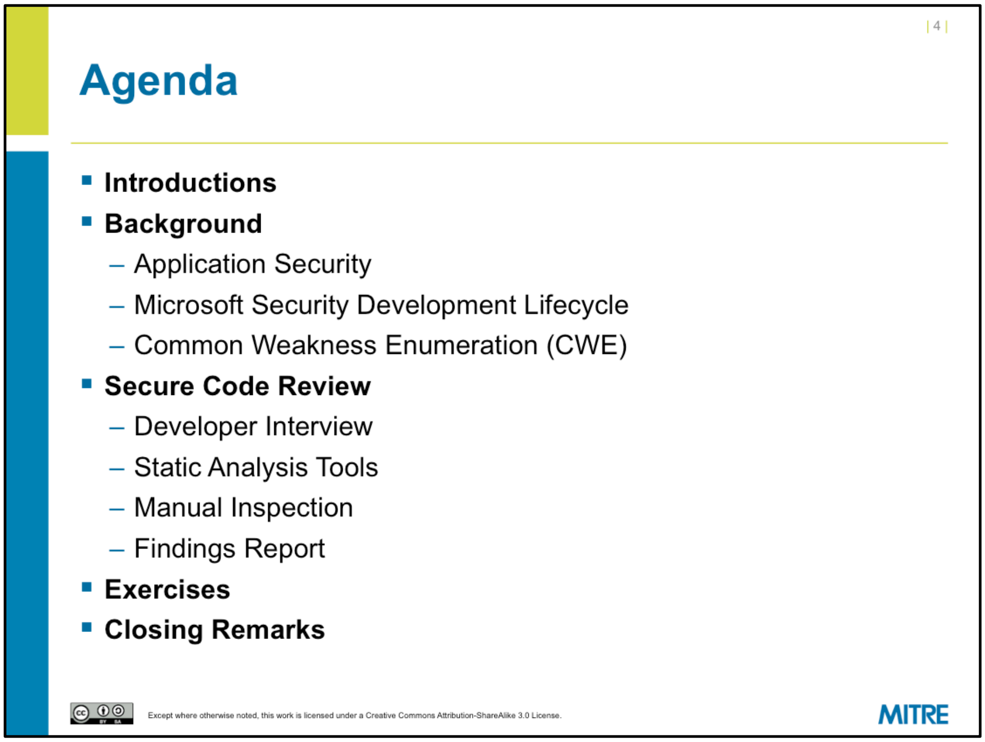
to Remix — to adapt the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

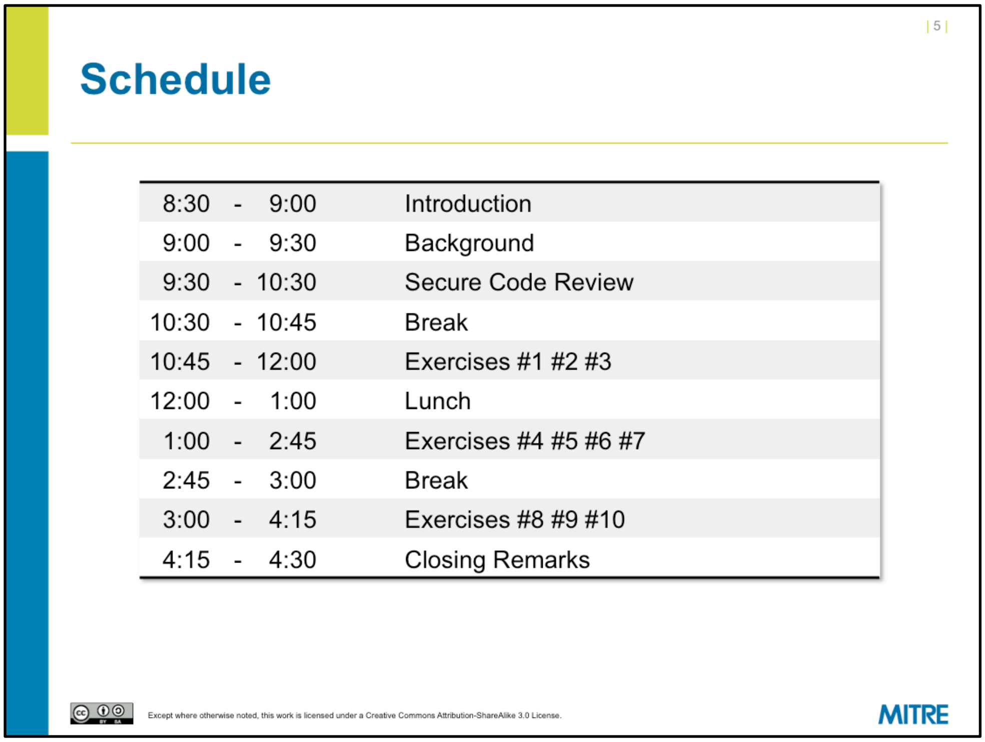
Attribution condition: You must indicate that derivative work "Is derived from Andrew Buttner and Mark Davidson’s 'Secure Code Review’ class, available at http://OpenSecurityTraining.info/SecureCodeReview.html”



Agenda

- **Introductions**
- **Background**
  - Application Security
  - Microsoft Security Development Lifecycle
  - Common Weakness Enumeration (CWE)
- **Secure Code Review**
  - Developer Interview
  - Static Analysis Tools
  - Manual Inspection
  - Findings Report
- **Exercises**
- **Closing Remarks**

MITRE



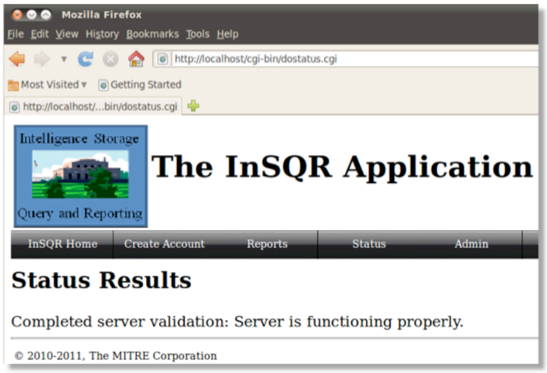
Schedule

8:30 - 9:00	Introduction
9:00 - 9:30	Background
9:30 - 10:30	Secure Code Review
10:30 - 10:45	Break
10:45 - 12:00	Exercises #1 #2 #3
12:00 - 1:00	Lunch
1:00 - 2:45	Exercises #4 #5 #6 #7
2:45 - 3:00	Break
3:00 - 4:15	Exercises #8 #9 #10
4:15 - 4:30	Closing Remarks

MITRE

Exercises

Bulk of the class will be hands-on as together we will perform a full review of The InSQR Application.



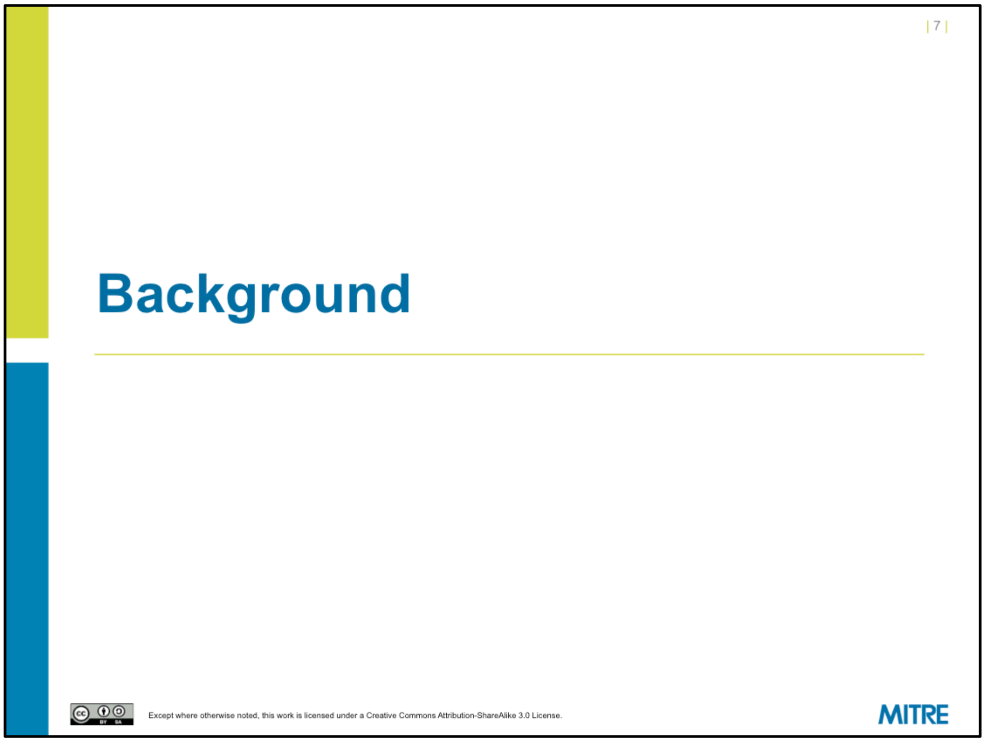
The InSQR Application

Status Results

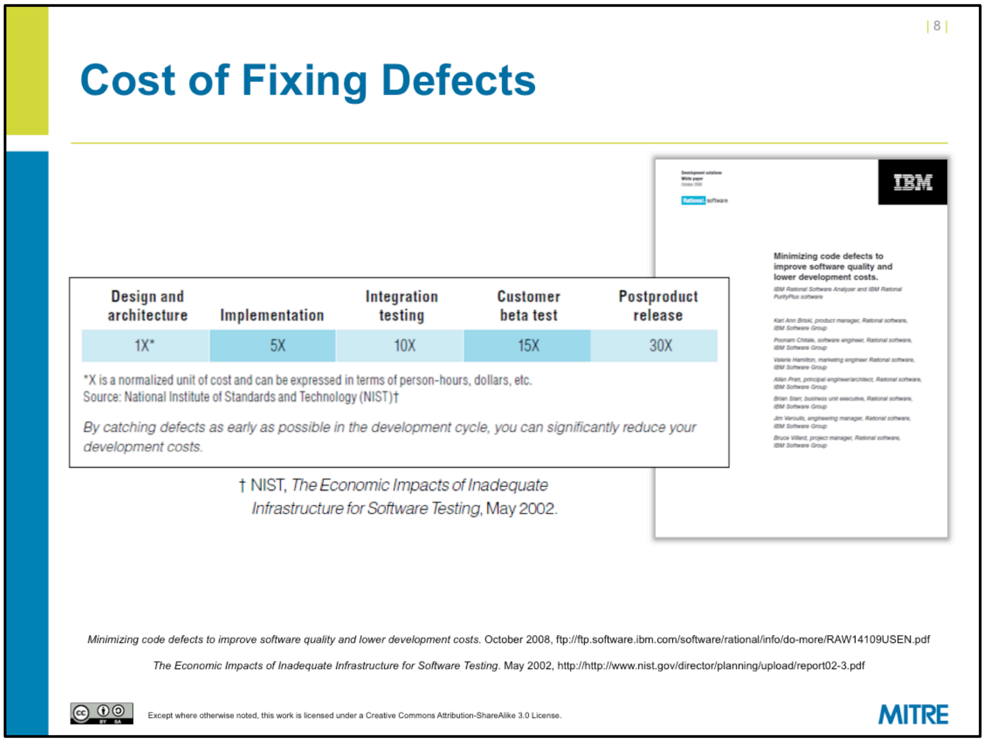
Completed server validation: Server is functioning properly.

© 2010-2011, The MITRE Corporation

MITRE



Background



| 8 |

Cost of Fixing Defects

Design and architecture	Implementation	Integration testing	Customer beta test	Postproduct release
1X*	5X	10X	15X	30X

*X is a normalized unit of cost and can be expressed in terms of person-hours, dollars, etc.
Source: National Institute of Standards and Technology (NIST)†

By catching defects as early as possible in the development cycle, you can significantly reduce your development costs.

† NIST, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.

Development solutions
White paper
October 2008

Rational. software

IBM

Minimizing code defects to improve software quality and lower development costs.

IBM Rational Software Analyzer and IBM Rational PurifyPlus software

Kari Ann Briski, product manager, Rational software, IBM Software Group

Poonam Chitale, software engineer, Rational software, IBM Software Group

Valerie Hamilton, marketing engineer Rational software, IBM Software Group

Allan Pratt, principal engineer/architect, Rational software, IBM Software Group

Brian Starr, business unit executive, Rational software, IBM Software Group

Jim Veroulis, engineering manager, Rational software, IBM Software Group

Bruce Villard, project manager, Rational software, IBM Software Group

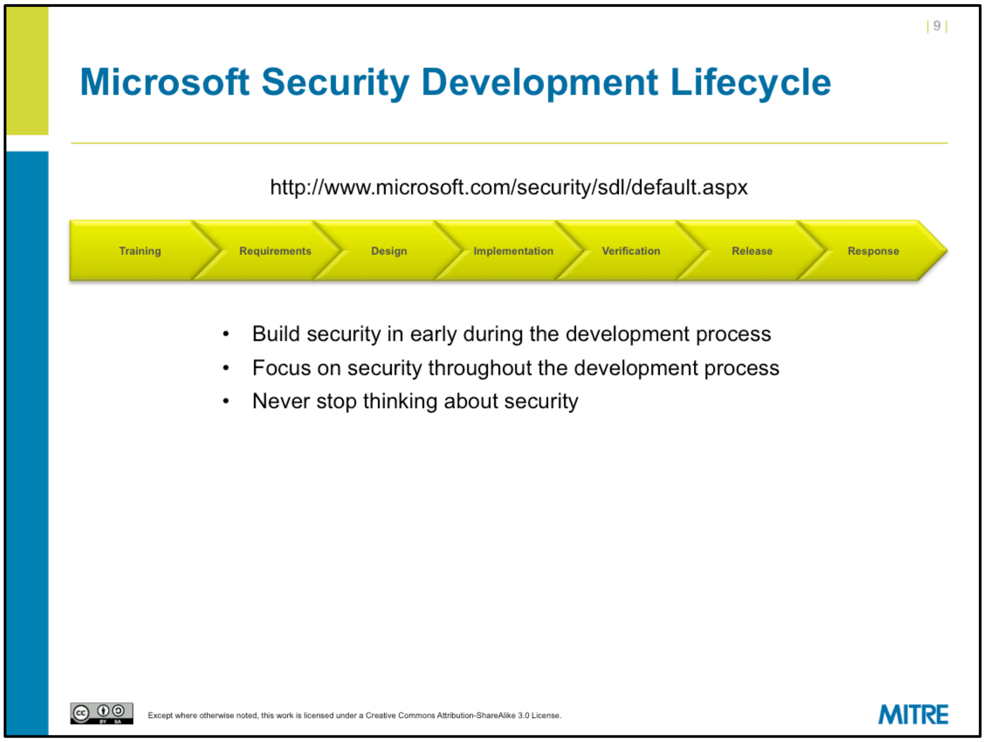
Minimizing code defects to improve software quality and lower development costs. October 2008, ftp://ftp.software.ibm.com/software/rational/info/do-more/RAW14109USEN.pdf

The Economic Impacts of Inadequate Infrastructure for Software Testing. May 2002, http://http://www.nist.gov/director/planning/upload/report02-3.pdf

MITRE

One compelling reason for spending the time and effort to find defects earlier in the development lifecycle is that the cost of fixing a defect increases significantly as the development lifecycle progresses. Research, like the reports from IBM and NIST, have shown over and over that the cost of fixing a defect rises the later in the lifecycle it is found. The NIST report is diagramed in the slide and shows the cost of fixing a defect at each stage of the development lifecycle. The cost is expressed as “X”, a normalized unit of cost that can be expressed in terms of person-hours, dollars, etc. The post product release cost of fixing a defect is shown to be thirty times more than the cost of fixing a defect in the design and architecture phase. This makes sense when you think about the additional activities that have to be performed on code that has been released vs. code that is being designed. Code that has been released has the added cost of integration with other products and services, multiple deployments that must be updated, and the code must repeat the entire release process from start to finish. The return on investment for fixing a defect at the beginning of the software development lifecycle instead of after the code has been released is 30x.

In addition to the cost savings, fixing a defect early can provide other benefits. Fixing defects early improves the security and functionality of the code base, keeping your company’s name out of the news and your customers happy. All too often a simple and unnoticed error – a single equals instead of a double equals in an IF statement, duplicative GOTO statements, or statements outside of the intended scope - can have catastrophic results. C-level executives resign, the public perception of a product shifts



| 9 |

Microsoft Security Development Lifecycle

http://www.microsoft.com/security/sdl/default.aspx

Training
Requirements
Design
Implementation
Verification
Release
Response

- Build security in early during the development process
- Focus on security throughout the development process
- Never stop thinking about security

MITRE

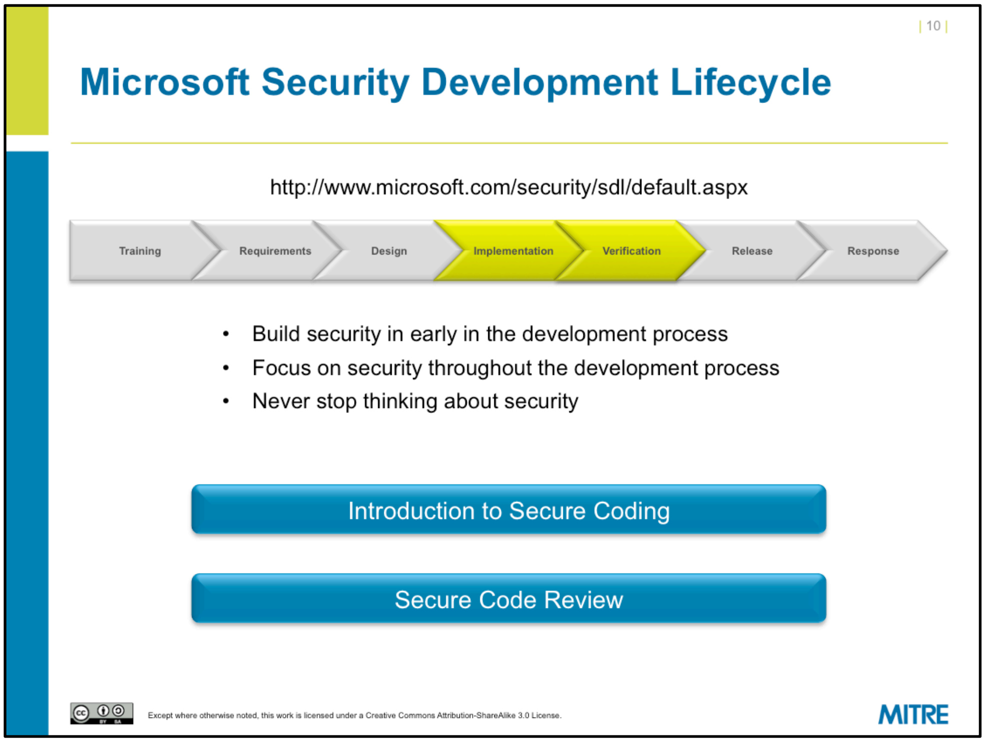
Microsoft has a published, well known high level security development lifecycle, which we use to model our own thinking about secure coding. The Microsoft Security Development Lifecycle defines 17 practices spread across the 7 phases that, when followed, improve the security of software. The Microsoft SDLC includes privacy components as well.

- In the Training phase, foundational concepts like secure design, threat modeling, secure coding, security testing, and best practices surrounding privacy are taught to developers. You are participating in the Training phase right now.

- In the Requirements phase, security and privacy requirements are established to help identify key milestones and minimize disruptions to plans and schedules, minimum acceptable levels of security and privacy are defined, and security and privacy risk assessments are performed to help a team identify which parts of a project will require threat modeling and security design reviews.

- In the Design phase, design requirements are established to help minimize schedule disruptions, the attack surface is analyzed and possibly reduced, and threat modeling is performed in order to help a team more effectively identify security vulnerabilities.

- In the Implementation phase, approved tools and associated security checks help a



| 10 |

Microsoft Security Development Lifecycle

http://www.microsoft.com/security/sdl/default.aspx

Training → Requirements → Design → Implementation → Verification → Release → Response

- Build security in early in the development process
- Focus on security throughout the development process
- Never stop thinking about security

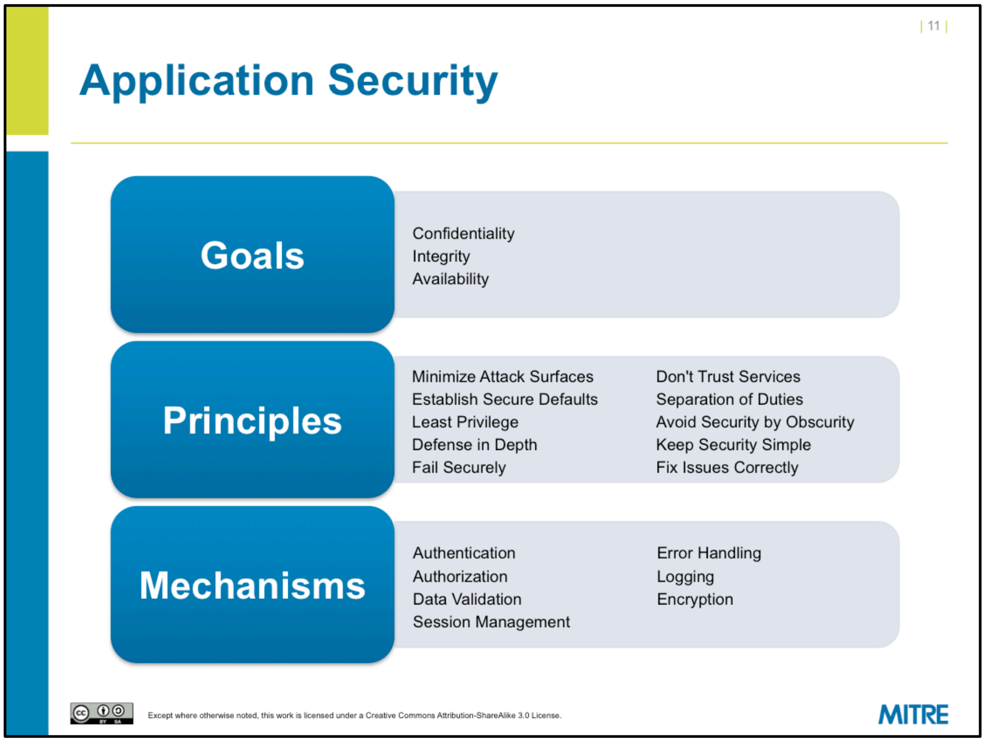
Introduction to Secure Coding

Secure Code Review

MITRE

The Intro to Secure Coding class focuses on the Implementation phase where the coding actually happens. That class teaches developers how to maintain a security mindset while writing software.

This course is a follow on and focuses on the Verification phase. We'll talk primarily about peer reviews and how they can be used to identify potential weaknesses in software. The lessons learned during this course can be applied to the Implementation phase when you write your own software.



| 11 |

Application Security

Goals

Confidentiality
Integrity
Availability

Principles

Minimize Attack Surfaces	Don't Trust Services
Establish Secure Defaults	Separation of Duties
Least Privilege	Avoid Security by Obscurity
Defense in Depth	Keep Security Simple
Fail Securely	Fix Issues Correctly

Mechanisms

Authentication	Error Handling
Authorization	Logging
Data Validation	Encryption
Session Management	

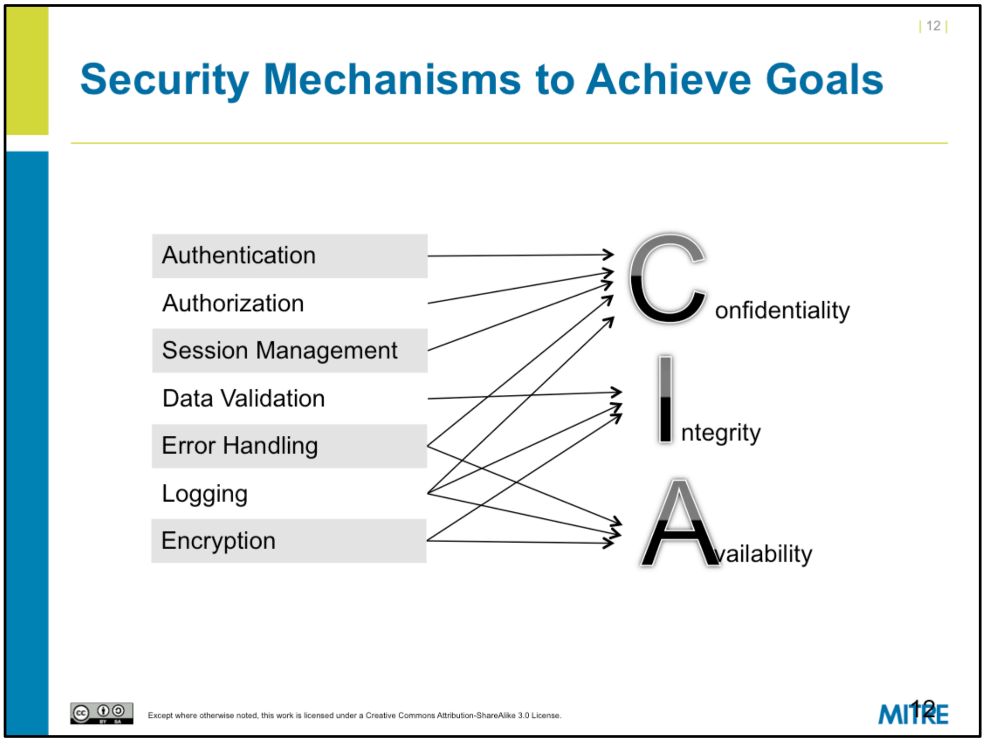
MITRE

This is a review slide from the Intro to Secure Coding class. In the Introduction to Secure Coding class we talked about the thee high level goals of application security:

- Confidentiality is ensuring that the application only grants data access to the users that are authorized to see it.
- Integrity is ensuring that data has not been modified during storage or communication.
- Availability is ensuring that the application is available to perform its function when needed.

These three high level goals help improve application security. You might notice that at this point I've talked about how you can take actions that *help improve* security, but I haven't told you what you can do to guarantee security. That is because there is nothing absolutely guarantees security. You can, however, maximize security. In order to meet the goals of Confidentially, Integrity, and Availability, there are 10 principles that you can follow.

- Minimize Attack Surfaces – Reduce the number of ways users and/or third party services interact with your application. Removing duplicate functionality, removing unnecessary form fields, and removing unnecessary functionality altogether all help minimize the attack surface.
- Establish secure defaults – Never assume that the user/installer of an application



Security Mechanisms to Achieve Goals

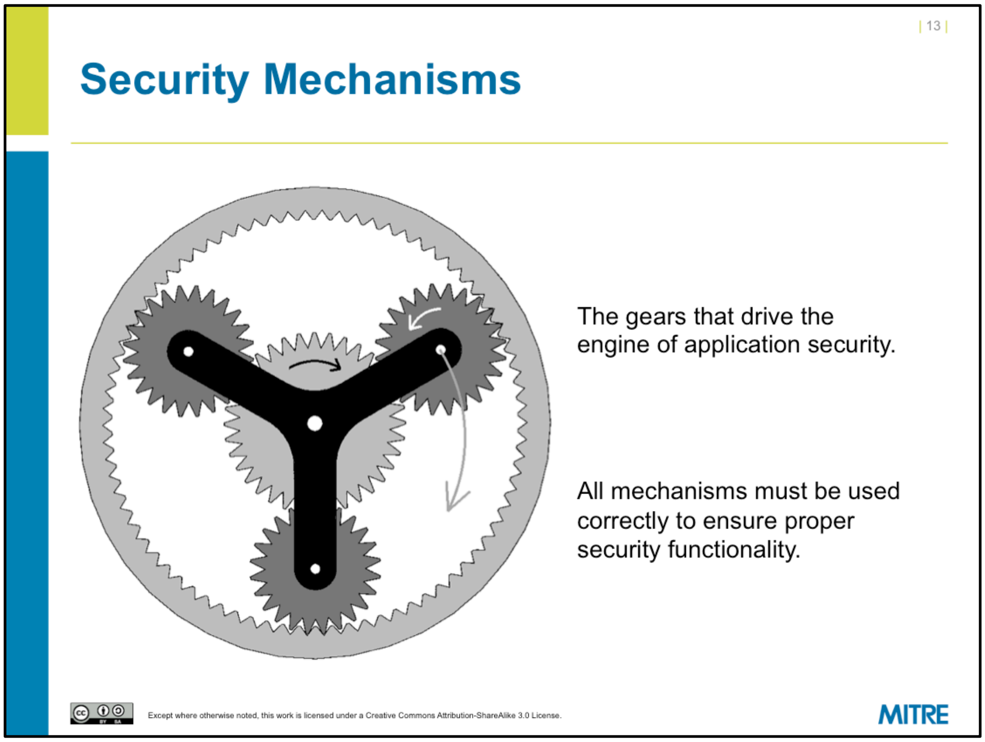
- Authentication
- Authorization
- Session Management
- Data Validation
- Error Handling
- Logging
- Encryption

Confidentiality

Integrity

Availability

Reminder on how mechanisms help achieve goals. We have arrows there, but really all mechanisms apply to all goals.



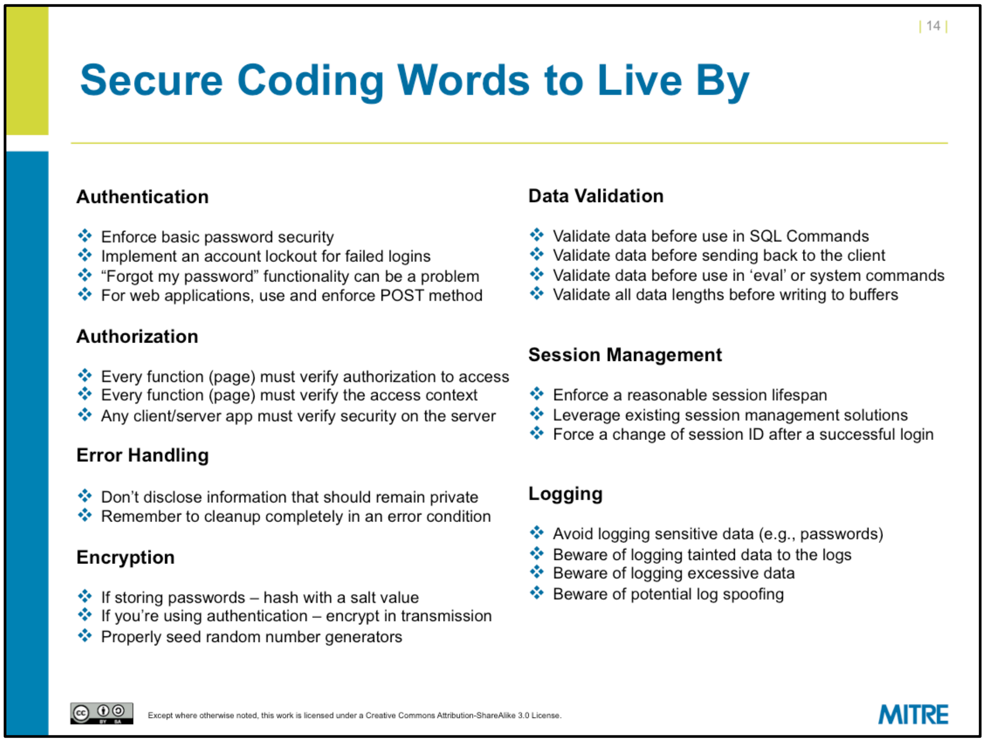
| 13 |

Security Mechanisms

The gears that drive the engine of application security.

All mechanisms must be used correctly to ensure proper security functionality.

In the end, secure coding really comes down to the different mechanisms that are available to ensure adherence to the previously mentioned application security principles. Our job as code reviewers is to assess the implementation of these mechanisms in the product. Make sure that implementation was done properly, etc.



Secure Coding Words to Live By

Authentication

- Enforce basic password security
- Implement an account lockout for failed logins
- "Forgot my password" functionality can be a problem
- For web applications, use and enforce POST method

Authorization

- Every function (page) must verify authorization to access
- Every function (page) must verify the access context
- Any client/server app must verify security on the server

Error Handling

- Don't disclose information that should remain private
- Remember to cleanup completely in an error condition

Encryption

- If storing passwords – hash with a salt value
- If you're using authentication – encrypt in transmission
- Properly seed random number generators

Data Validation

- Validate data before use in SQL Commands
- Validate data before sending back to the client
- Validate data before use in 'eval' or system commands
- Validate all data lengths before writing to buffers

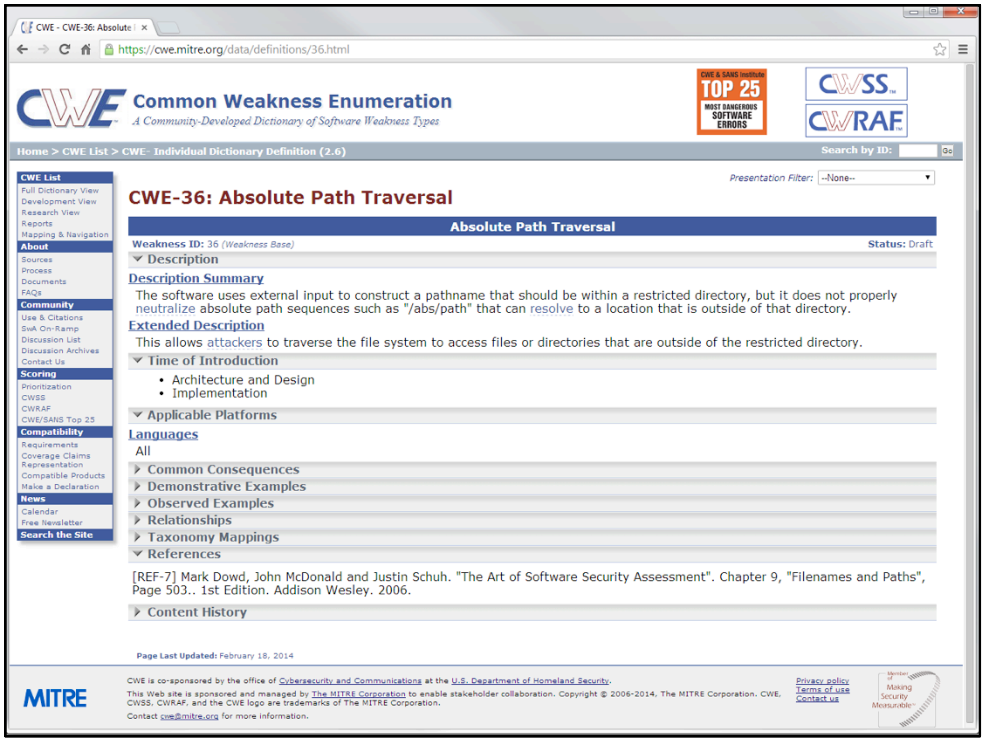
Session Management

- Enforce a reasonable session lifespan
- Leverage existing session management solutions
- Force a change of session ID after a successful login

Logging

- Avoid logging sensitive data (e.g., passwords)
- Beware of logging tainted data to the logs
- Beware of logging excessive data
- Beware of potential log spoofing

Reminder of the words to live by from the previous class. It's a good idea, even for the secure code review team, to go back and look at these. Focus on what the developer is trying to do.



CWE-36: Absolute Path Traversal

Absolute Path Traversal

Weakness ID: 36 (Weakness Base) Status: Draft

Description

Description Summary

The software uses external input to construct a pathname that should be within a restricted directory, but it does not properly neutralize absolute path sequences such as "/abs/path" that can resolve to a location that is outside of that directory.

Extended Description

This allows attackers to traverse the file system to access files or directories that are outside of the restricted directory.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

All

Common Consequences

Demonstrative Examples

Observed Examples

Relationships

Taxonomy Mappings

References

[REF-7] Mark Dowd, John McDonald and Justin Schuh. "The Art of Software Security Assessment". Chapter 9, "Filenames and Paths", Page 503.. 1st Edition. Addison Wesley. 2006.

Content History

Page Last Updated: February 18, 2014

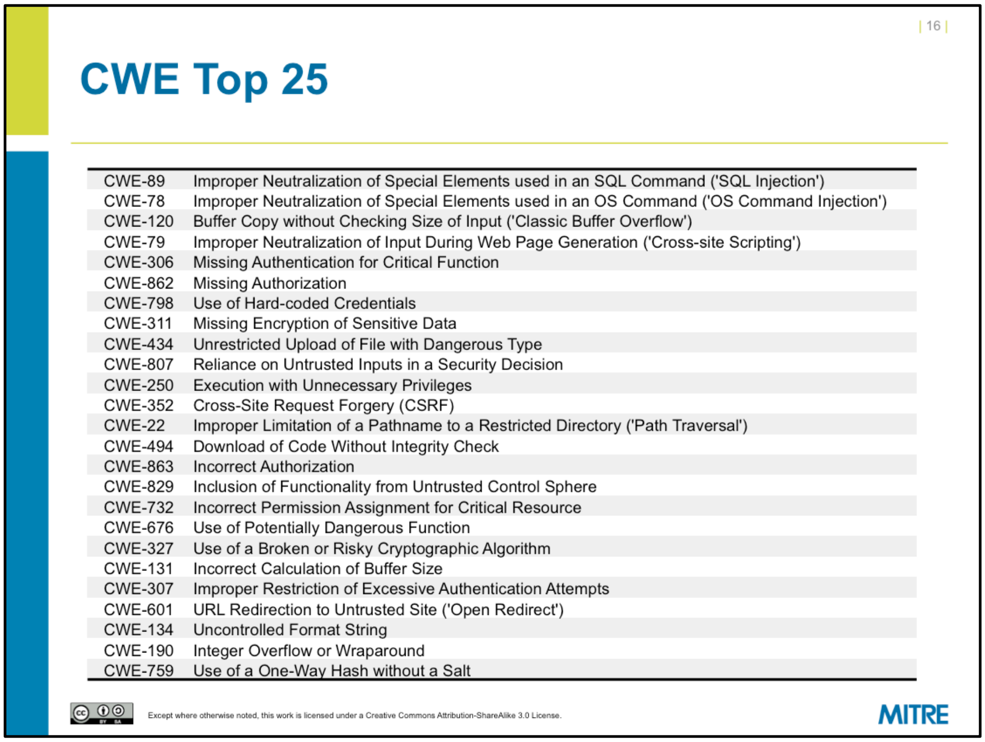
CWE is co-sponsored by the office of Cybersecurity and Communications at the U.S. Department of Homeland Security.

This Web site is sponsored and managed by The MITRE Corporation to enable stakeholder collaboration. Copyright © 2006-2014, The MITRE Corporation. CWE, CWSS, CWRAF, and the CWE logo are trademarks of The MITRE Corporation.

Contact cwe@mitre.org for more information.

One project that everyone should be aware of, and a project we will mention a lot throughout this course, is the Common Weakness Enumeration (CWE). This is a MITRE-run initiative to enumerate and provide standard identifiers for the different coding-level security-related mistakes that developers often make. This standard identifier enable security personnel to share information about weaknesses and for tools to report findings in a way that review teams can easily grasp. There's a lot of good description information on weaknesses, which is a benefit to both the reviewer and the developer. The reviewer doesn't have to spend time duplicating a description that has been used many times over, and the developer doesn't have to rely on the communication skills of the developer. Many static analysis tools use CWEs to report the weaknesses they find.

It's important to note that a CWE describes a weakness, but not a vulnerability. In order for a weakness to become a vulnerability, it has to be exploitable. For example, I reviewed one application where the developer had made a mistake by allowing SQL Injection. I knew the developer of the application and got permission to attempt to exploit the weakness in a non-production system. It turns out that the weakness was not a vulnerability because the database was fully public, meaning I couldn't get access to any information I didn't already have, and the permissions were set to read only and therefore my "drop tables" command didn't have any effect. While, in this case, the weakness turned out to not be a vulnerability, it was still a weakness that the developer fixed.



| 16 |

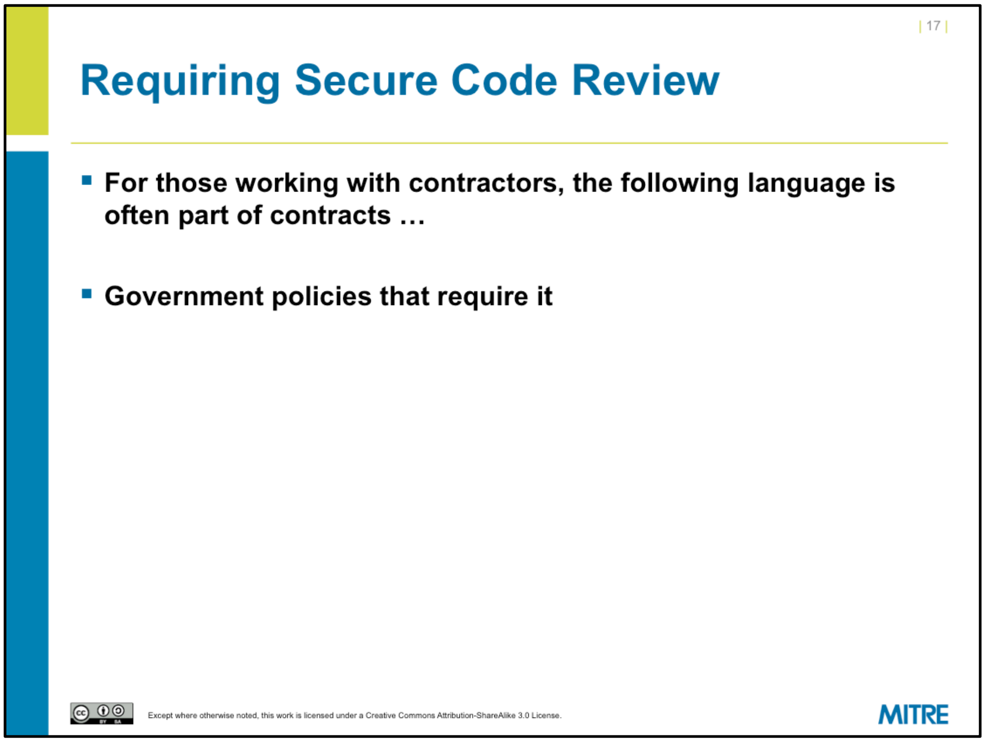
CWE Top 25

CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-306	Missing Authentication for Critical Function
CWE-862	Missing Authorization
CWE-798	Use of Hard-coded Credentials
CWE-311	Missing Encryption of Sensitive Data
CWE-434	Unrestricted Upload of File with Dangerous Type
CWE-807	Reliance on Untrusted Inputs in a Security Decision
CWE-250	Execution with Unnecessary Privileges
CWE-352	Cross-Site Request Forgery (CSRF)
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-494	Download of Code Without Integrity Check
CWE-863	Incorrect Authorization
CWE-829	Inclusion of Functionality from Untrusted Control Sphere
CWE-732	Incorrect Permission Assignment for Critical Resource
CWE-676	Use of Potentially Dangerous Function
CWE-327	Use of a Broken or Risky Cryptographic Algorithm
CWE-131	Incorrect Calculation of Buffer Size
CWE-307	Improper Restriction of Excessive Authentication Attempts
CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
CWE-134	Uncontrolled Format String
CWE-190	Integer Overflow or Wraparound
CWE-759	Use of a One-Way Hash without a Salt

MITRE

The CWE team also compiles a Top 25 list each year that helps identify the 25 most dangerous and prevalent software errors that we see today. This list is a great way to keep the most common issues in the forefront of a developer's mind and help focus effort to make sure that these errors are not introduced. This is a good list to be familiar with, in part because this list is largely composed of weaknesses that should never ever happen anymore. SQL injection is the top weakness found in software today. Prepared Statements have been around as long as I can remember – the only reason SQL injection exists is because developers don't take care to use them.

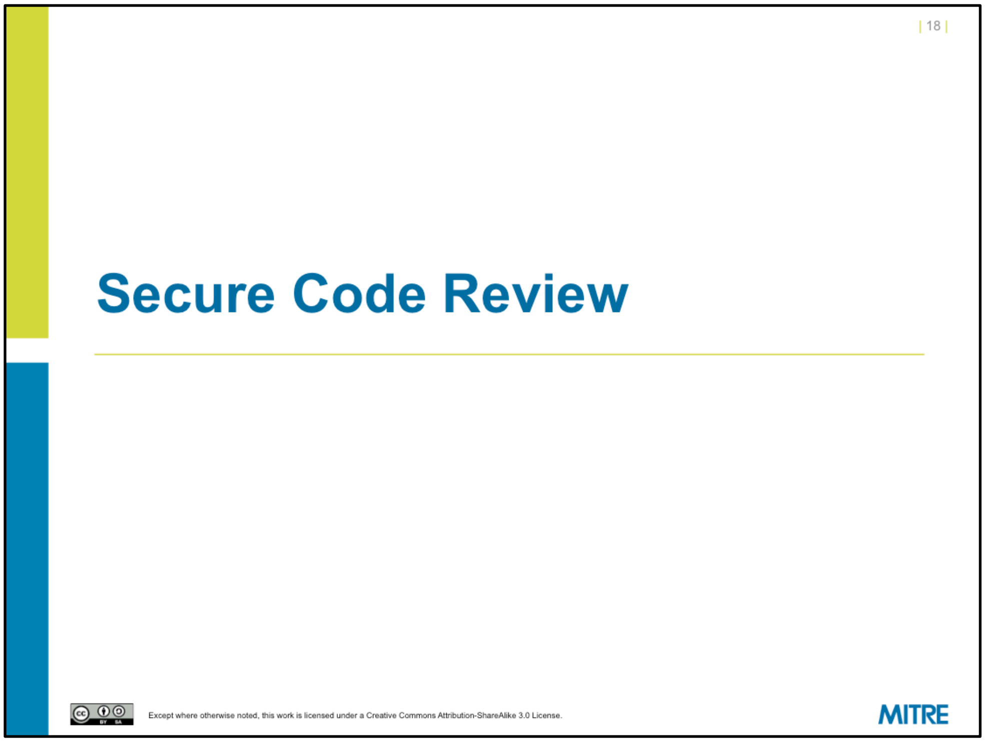
We've done 25 reviews over the last 2 years, and all of them have these kinds of errors. We have yet to review a single application that doesn't have a single finding, and very few that don't have at least one top 25. Many of these are well known things that shouldn't exist anymore.



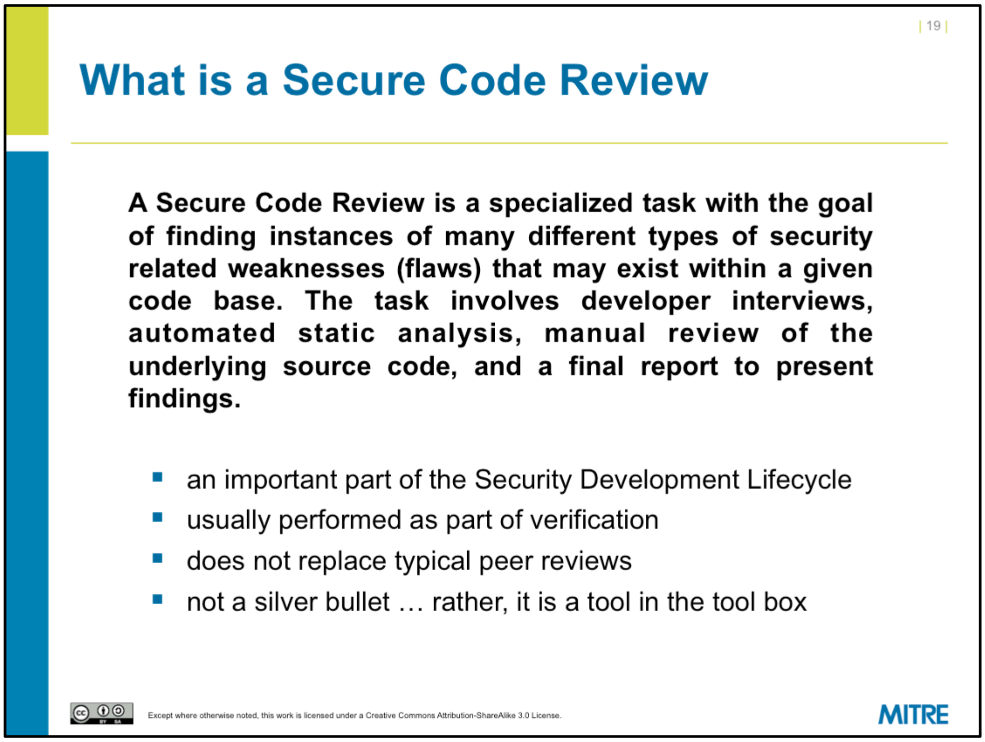
Requiring Secure Code Review

- For those working with contractors, the following language is often part of contracts …
- Government policies that require it

MITRE



Secure Code Review



| 19 |

What is a Secure Code Review

A Secure Code Review is a specialized task with the goal of finding instances of many different types of security related weaknesses (flaws) that may exist within a given code base. The task involves developer interviews, automated static analysis, manual review of the underlying source code, and a final report to present findings.

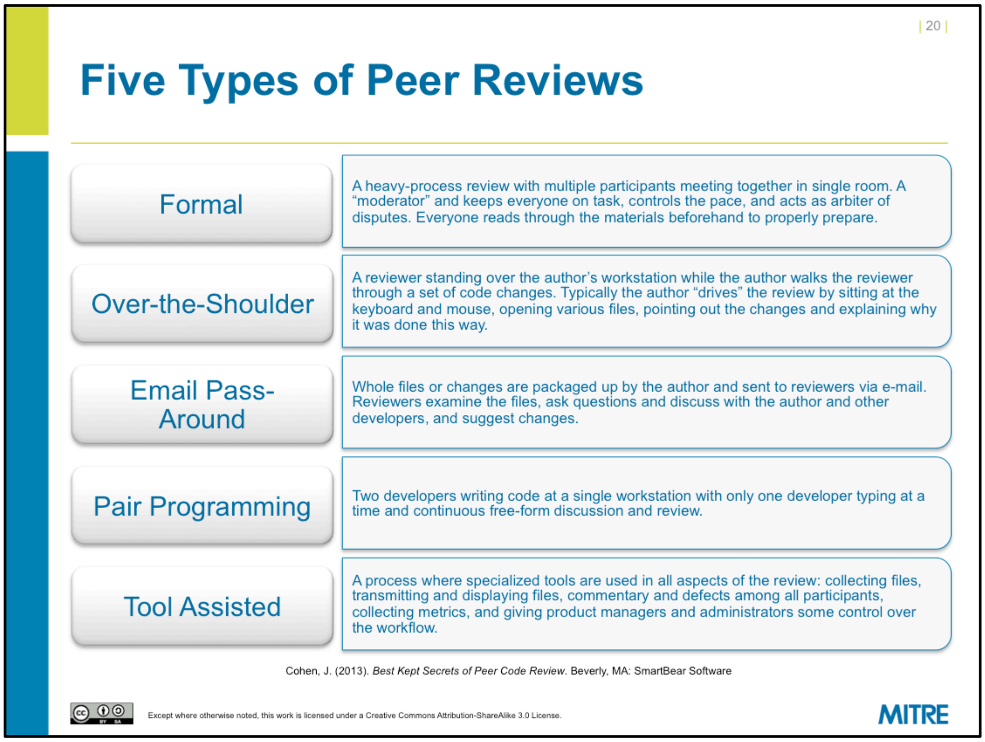
- an important part of the Security Development Lifecycle
- usually performed as part of verification
- does not replace typical peer reviews
- not a silver bullet … rather, it is a tool in the tool box

MITRE

A secure code review is a task at the end of the SDLC that attempts to find any security related flaws that came about during the coding process. This could be due to mistakes by the developers, or maybe a change from the intended design due to coding challenges.

It involves an interview, static analysis, manual analysis, and a final report.

A secure code review does not replace the developer peer reviews that should be taking place during the coding phase. Both activities offer something different and each should be leveraged accordingly. We will talk more about this in the coming slides.



Five Types of Peer Reviews

Formal

A heavy-process review with multiple participants meeting together in single room. A "moderator" and keeps everyone on task, controls the pace, and acts as arbiter of disputes. Everyone reads through the materials beforehand to properly prepare.

Over-the-Shoulder

A reviewer standing over the author's workstation while the author walks the reviewer through a set of code changes. Typically the author "drives" the review by sitting at the keyboard and mouse, opening various files, pointing out the changes and explaining why it was done this way.

Email Pass-Around

Whole files or changes are packaged up by the author and sent to reviewers via e-mail. Reviewers examine the files, ask questions and discuss with the author and other developers, and suggest changes.

Pair Programming

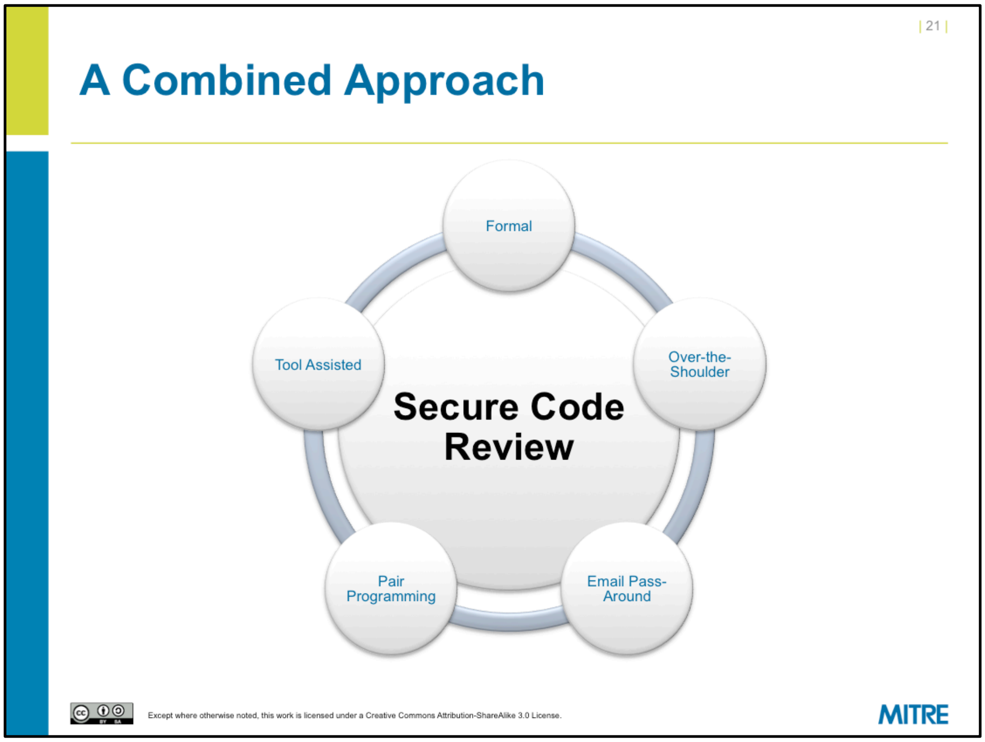
Two developers writing code at a single workstation with only one developer typing at a time and continuous free-form discussion and review.

Tool Assisted

A process where specialized tools are used in all aspects of the review: collecting files, transmitting and displaying files, commentary and defects among all participants, collecting metrics, and giving product managers and administrators some control over the workflow.

Cohen, J. (2013). *Best Kept Secrets of Peer Code Review*. Beverly, MA: SmartBear Software

Although the goals of a secure code review often differ from those of your typical peer review, there is a lot to be gained by looking at the well established methodologies for peer reviews. Peer reviews have evolved into 5 distinct types, each for a target audience and goal. Formal, over-the-shoulder, email pass-around, pair programming, and tool assisted all bring something different to the table.



A Combined Approach

Secure Code Review

Formal

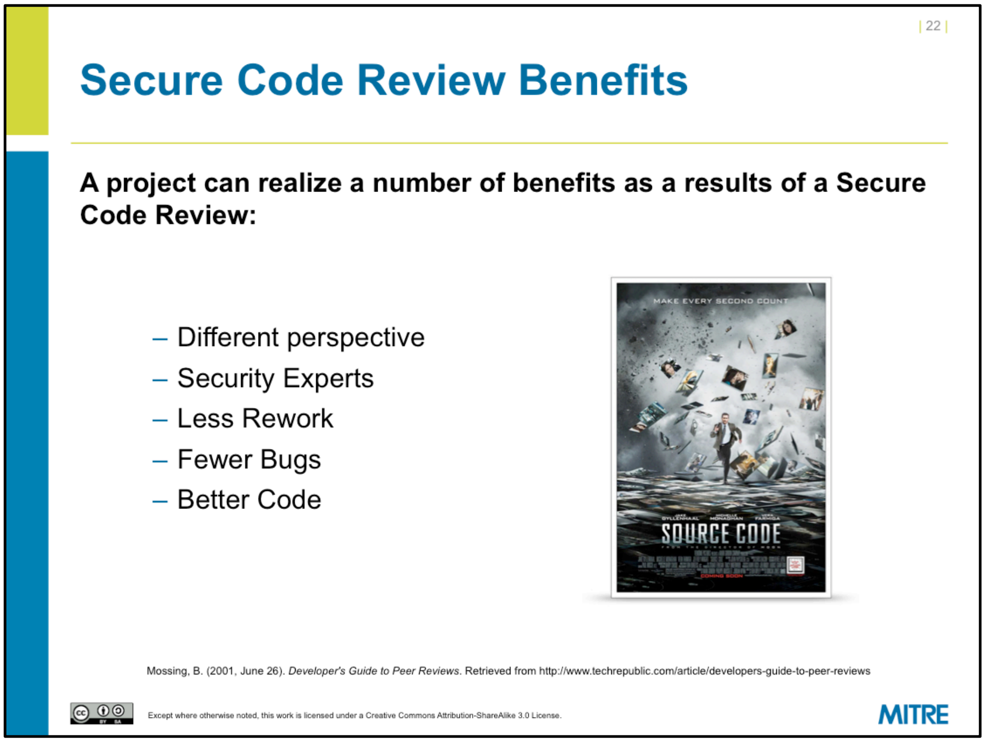
Over-the-Shoulder

Email Pass-Around

Pair Programming

Tool Assisted

A Secure Code Review looks to leverage elements from each of the different types of peer reviews.



| 22 |

Secure Code Review Benefits

A project can realize a number of benefits as a results of a Secure Code Review:

- Different perspective
- Security Experts
- Less Rework
- Fewer Bugs
- Better Code

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

What benefits can you expect from a Secure Code Review? A few of the benefits include:

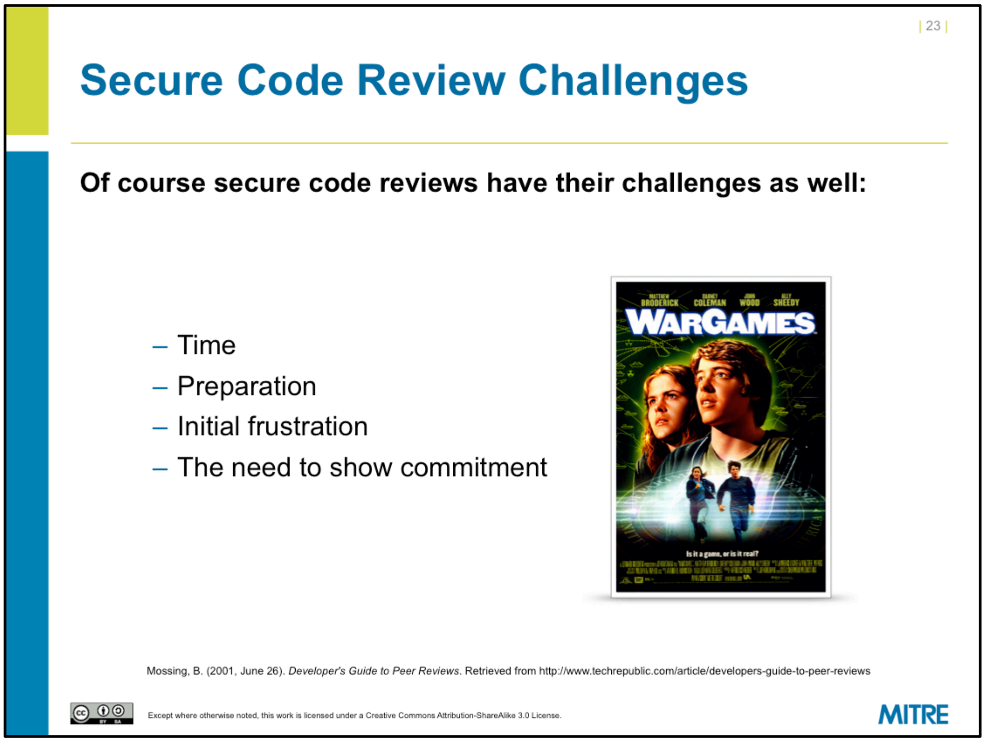
- **A different perspective.** "Another set of eyes" adds objectivity. Similar to the reason for separating your coding and testing teams, secure code reviews provide the distance needed to recognize problems.

- **Security Experts.** The developers have enough on their plates trying to stay current with the latest frameworks and coding practices. They also are often tasked to get thing finished quickly to meet an unrealistic deadline. A Secure Code Review allows a team that focuses on secure coding to look at the code.

- **Less rework.** Do it right the first time. Changes cost more later in the life cycle. The secure code review process catches many errors *before* they go to production.

- **Fewer bugs.** It's better to discover your own problems than to have someone (like a user) point them out to you.

- **Better Code.** At the end of the process, the application being developed is better, and that is the ultimate goal of everyone involved in its development.



| 23 |

Secure Code Review Challenges

Of course secure code reviews have their challenges as well:

– Time
– Preparation
– Initial frustration
– The need to show commitment

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

MITRE

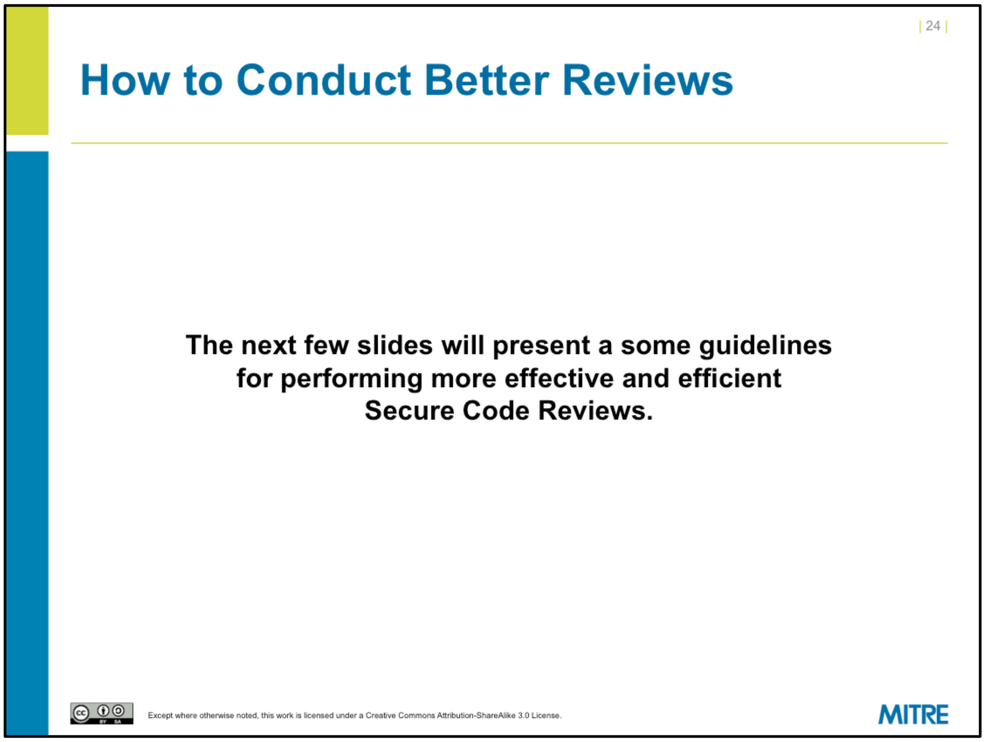
It can't all be roses. A few of the issues you will need to balance when you implement secure code reviews include:

- **Time.** Some secure code reviews take a long time. But like other types of meetings, focusing on the topic, being familiar with the process, and establishing solid ground rules can help keep the time reasonable. Secure Code Reviews invest your time; bugs waste it.

- **Preparation.** Reading unfamiliar code and correlating that code to unfamiliar documentation inevitably means questions for the programmers which takes them away from coding. However, it's a necessary evil. Over time, however, proper preparation for the meeting should take less time, as reviewers learn what to look for and become familiar with the process.

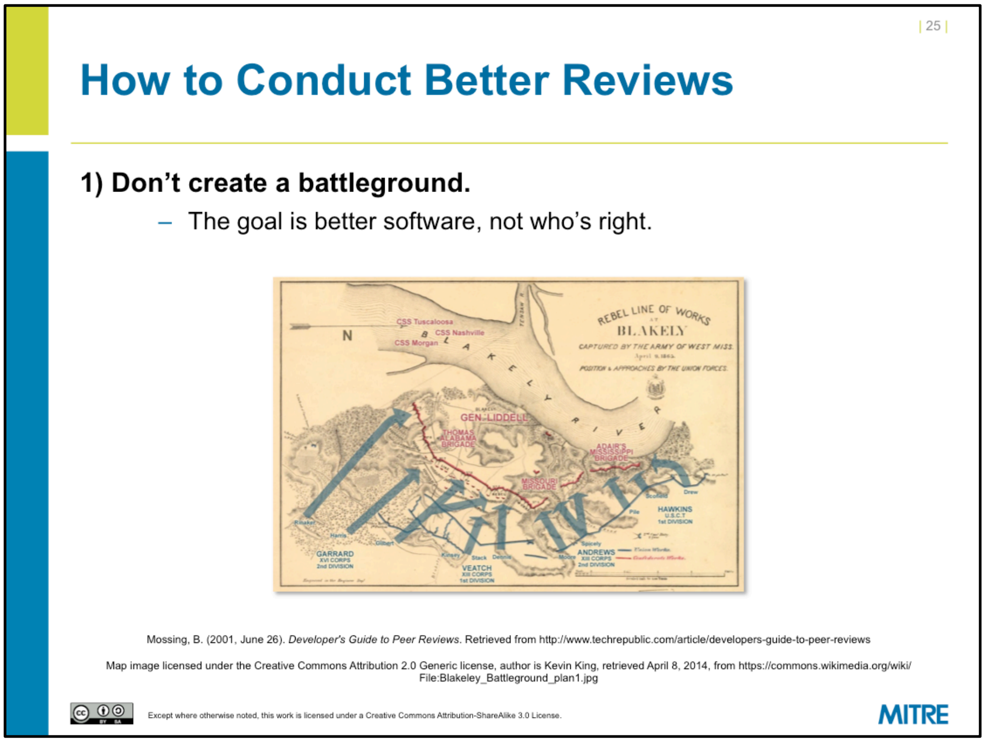
- **Initial frustration.** If team members are not familiar with secure code reviews, the experience can be frustrating for all participants. Teams need to devise a process for secure code reviews, implement it, and modify it only when the situation dictates. In time, members will grow accustomed to the process.

- **The need to show commitment.** The benefit of a secure code review is sometimes hard to see. If it is not done correctly, or if the code was in good



How to Conduct Better Reviews

The next few slides will present a some guidelines for performing more effective and efficient Secure Code Reviews.



How to Conduct Better Reviews

1) Don’t create a battleground.

- The goal is better software, not who’s right.

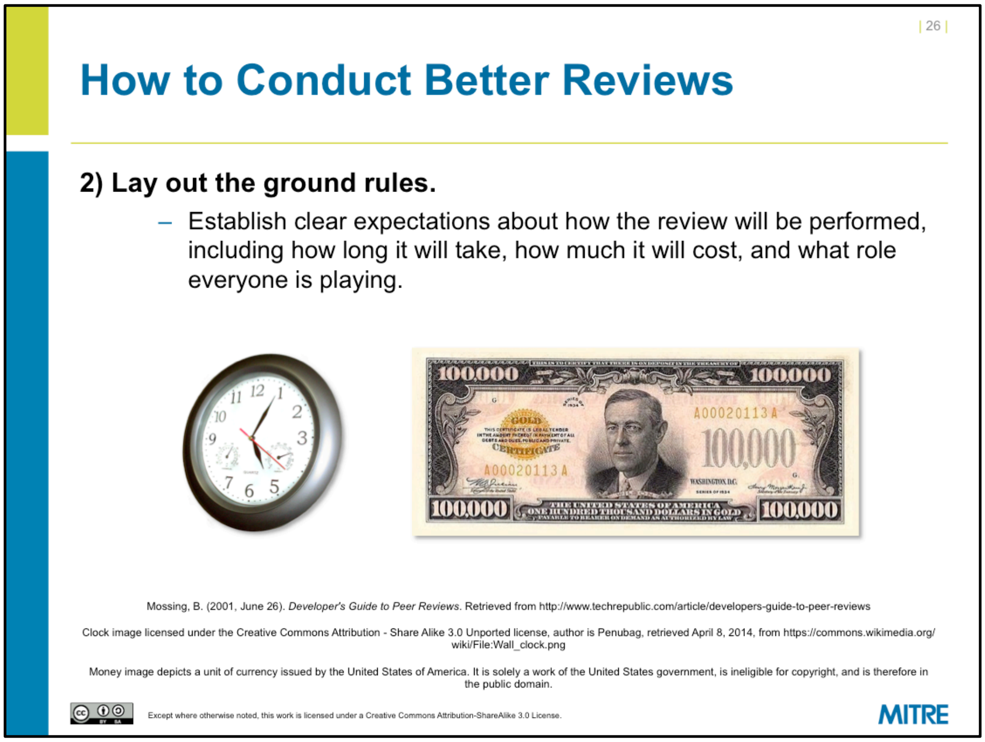
REBEL LINE OF WORKS
AT
BLAKELY
CAPTURED BY THE ARMY OF WEST MISS.
April 9, 1865.
POSITION & APPROACHES BY THE UNION FORCES.
N
CSS Tuscaloosa
CSS Nashville
CSS Morgan
BLAKELY RIVER
TENSAW R.
GEN. LIDDELL
THOMAS ALABAMA BRIGADE
ADAIR'S MISSISSIPPI BRIGADE
MISSOURI BRIGADE
Rinaker
Harris
Gilbert
GARRARD
XVI CORPS
2nd DIVISION
Kinsey
Stack
Dennis
VEATCH
XIII CORPS
1st DIVISION
Moore
Spicely
ANDREWS
XIII CORPS
2nd DIVISION
Pile
Scofield
Drew
HAWKINS
U.S.C.T
1st DIVISION
Union Works.
Confederate Works.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Map image licensed under the Creative Commons Attribution 2.0 Generic license, author is Kevin King, retrieved April 8, 2014, from https://commons.wikimedia.org/wiki/File:Blakeley_Battleground_plan1.jpg

MITRE

The review team is there to support the developers, not to prove that they are smarter than the developers. The developer are there to help the review team, not to show how the review team doesn't know what they are talking about.



| 26 |

How to Conduct Better Reviews

2) Lay out the ground rules.

- Establish clear expectations about how the review will be performed, including how long it will take, how much it will cost, and what role everyone is playing.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Clock image licensed under the Creative Commons Attribution - Share Alike 3.0 Unported license, author is Penubag, retrieved April 8, 2014, from https://commons.wikimedia.org/wiki/File:Wall_clock.png

Money image depicts a unit of currency issued by the United States of America. It is solely a work of the United States government, is ineligible for copyright, and is therefore in the public domain.

MITRE

The $100,000 Series 1934 Gold Certificate feature a portrait of Woodrow Wilson. These notes were printed from December 18, 1934, through January 9, 1935, and were issued by the Treasurer of the United States to Federal Reserve Banks only against an equal amount of gold bullion held by the Treasury Department. The notes were used only for official transactions between Federal Reserve Banks and were not circulated among the general public.[6]Photographic records show that at least seven 1934 $100,000 Gold Certificates are still in existence (#s A00000001A, A00020102A, A00020106A, A00020108A, A00020109A, A00020110A, A00020113A)

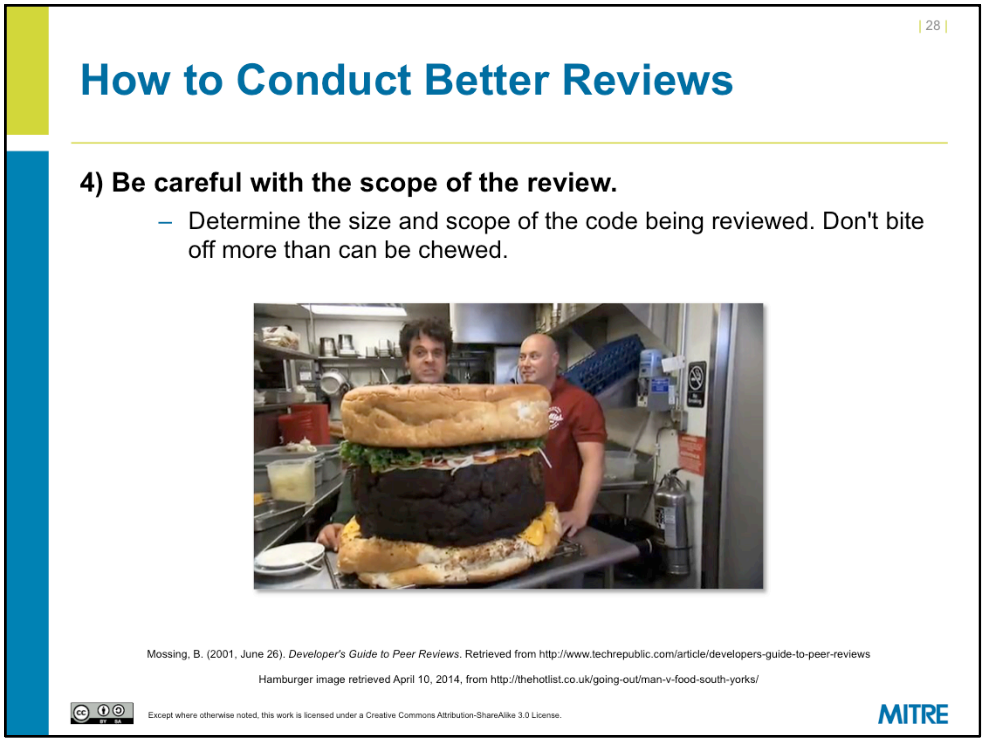
How to Conduct Better Reviews

3) Maintain professionalism.

- Don’t take the criticism personally and offer only technical advice that will improve the code. Respect others’ opinions, comments, and suggestions.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Tie image licensed under the Creative Commons Attribution - Share Alike 3.0 Unported license, original uploader was Plasmafire, retrieved April 10, 2014, from https://commons.wikimedia.org/wiki/File:Suit_tie.JPG



How to Conduct Better Reviews

4) Be careful with the scope of the review.

- Determine the size and scope of the code being reviewed. Don't bite off more than can be chewed.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Hamburger image retrieved April 10, 2014, from http://thehotlist.co.uk/going-out/man-v-food-south-yorks/

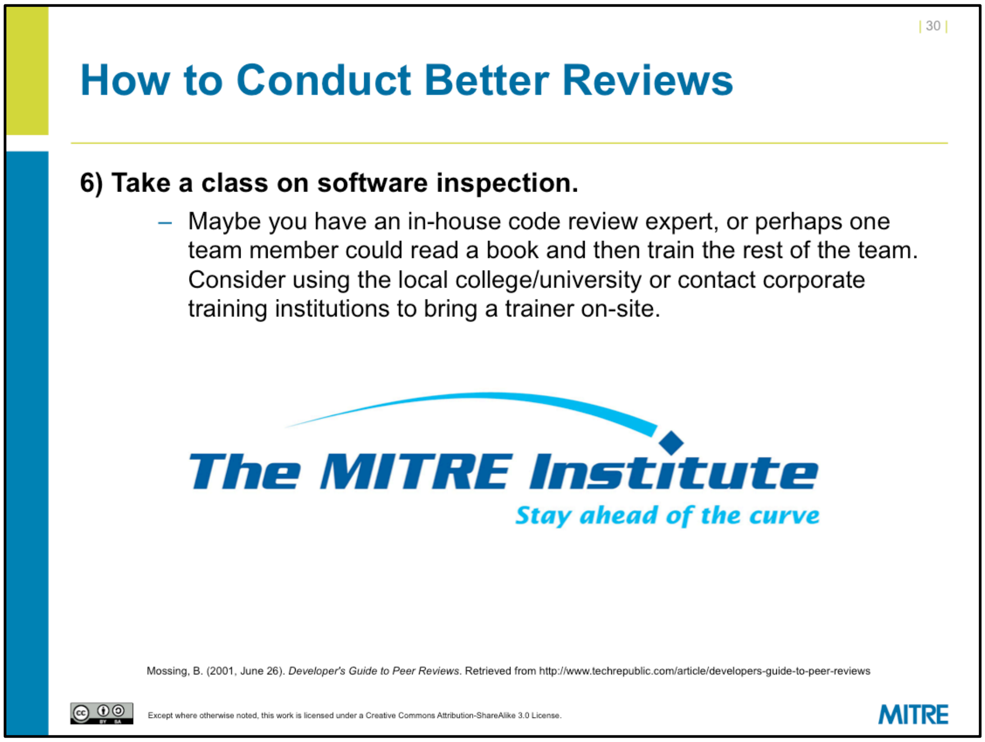
How to Conduct Better Reviews

5) Document what happens.

- Write everything down, especially decisions and action items.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Notes image licensed under the Creative Commons Attribution - Share Alike 3.0 Unported license, author is Tony Webster, retrieved April 11, 2014, from https://commons.wikimedia.org/wiki/File:CityCamp_Idea_Board_with_sticky_notes_4297872645_o.JPG



How to Conduct Better Reviews

6) Take a class on software inspection.

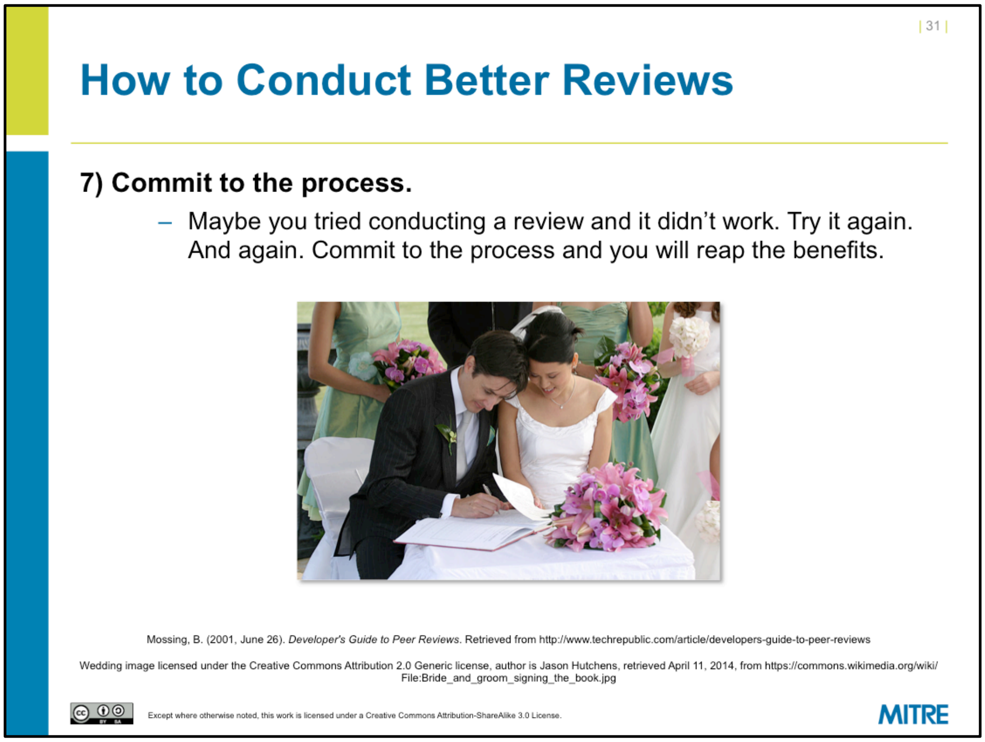
- Maybe you have an in-house code review expert, or perhaps one team member could read a book and then train the rest of the team. Consider using the local college/university or contact corporate training institutions to bring a trainer on-site.

The MITRE Institute

Stay ahead of the curve

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

MITRE



How to Conduct Better Reviews

7) Commit to the process.

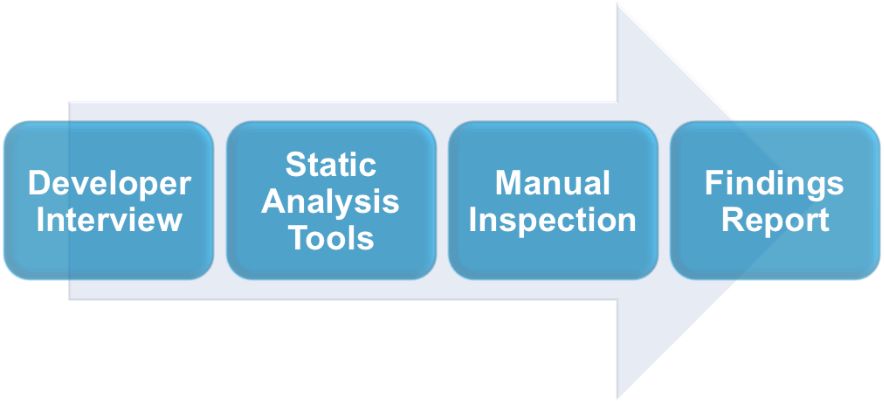
- Maybe you tried conducting a review and it didn’t work. Try it again. And again. Commit to the process and you will reap the benefits.

Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Wedding image licensed under the Creative Commons Attribution 2.0 Generic license, author is Jason Hutchens, retrieved April 11, 2014, from https://commons.wikimedia.org/wiki/File:Bride_and_groom_signing_the_book.jpg

MITRE

Secure Code Review Process

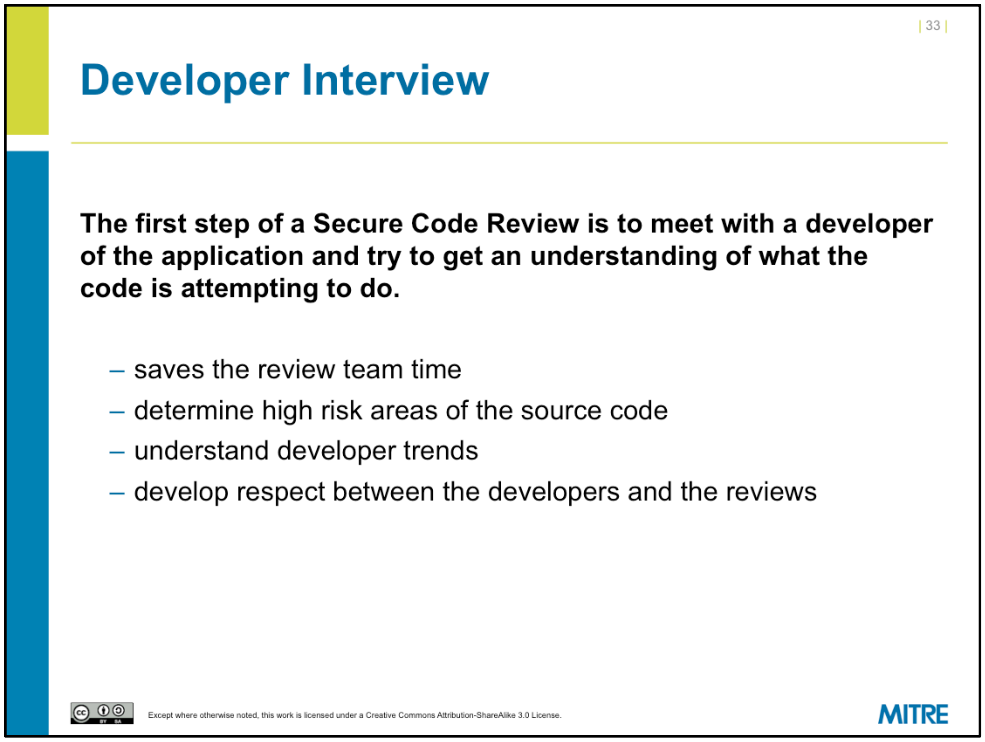


Developer Interview

Static Analysis Tools

Manual Inspection

Findings Report



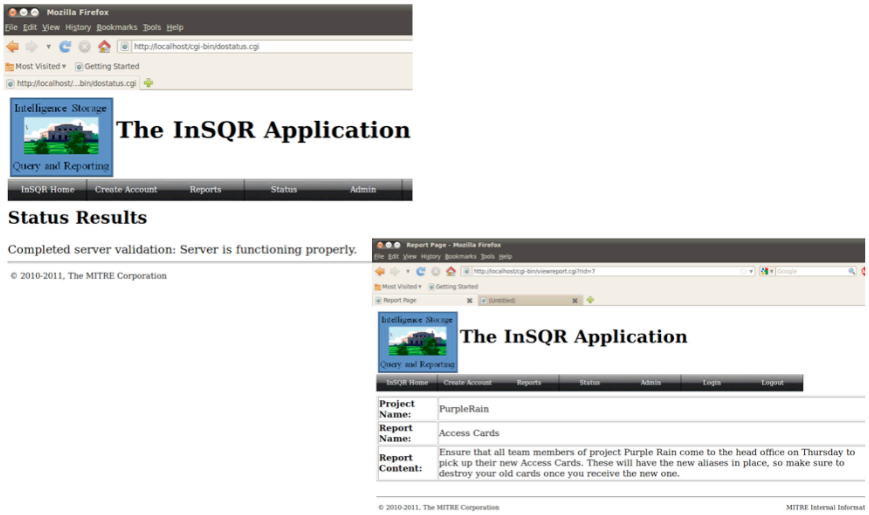
Developer Interview

The first step of a Secure Code Review is to meet with a developer of the application and try to get an understanding of what the code is attempting to do.

- saves the review team time
- determine high risk areas of the source code
- understand developer trends
- develop respect between the developers and the reviews

MITRE

Developer Interview - Role Play



The InSQR Application

Status Results

Completed server validation: Server is functioning properly.

© 2010-2011, The MITRE Corporation

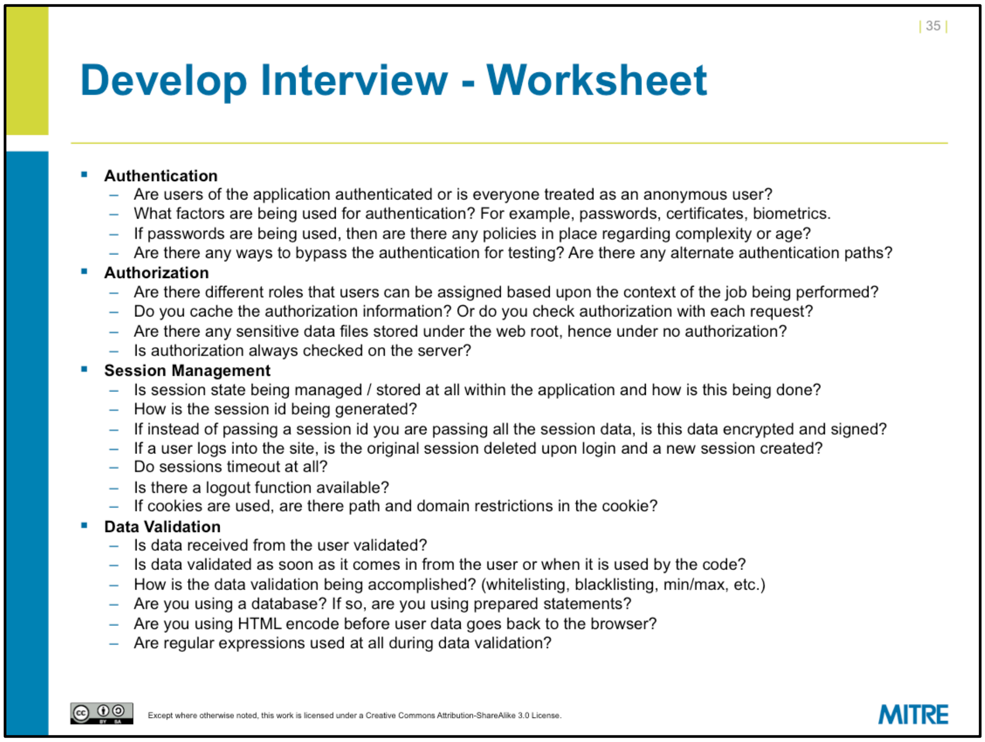
The InSQR Application

Project Name:	PurpleRain
Report Name:	Access Cards
Report Content:	Ensure that all team members of project Purple Rain come to the head office on Thursday to pick up their new Access Cards. These will have the new aliases in place, so make sure to destroy your old cards once you receive the new one.

© 2010-2011, The MITRE Corporation MITRE Internal Informat

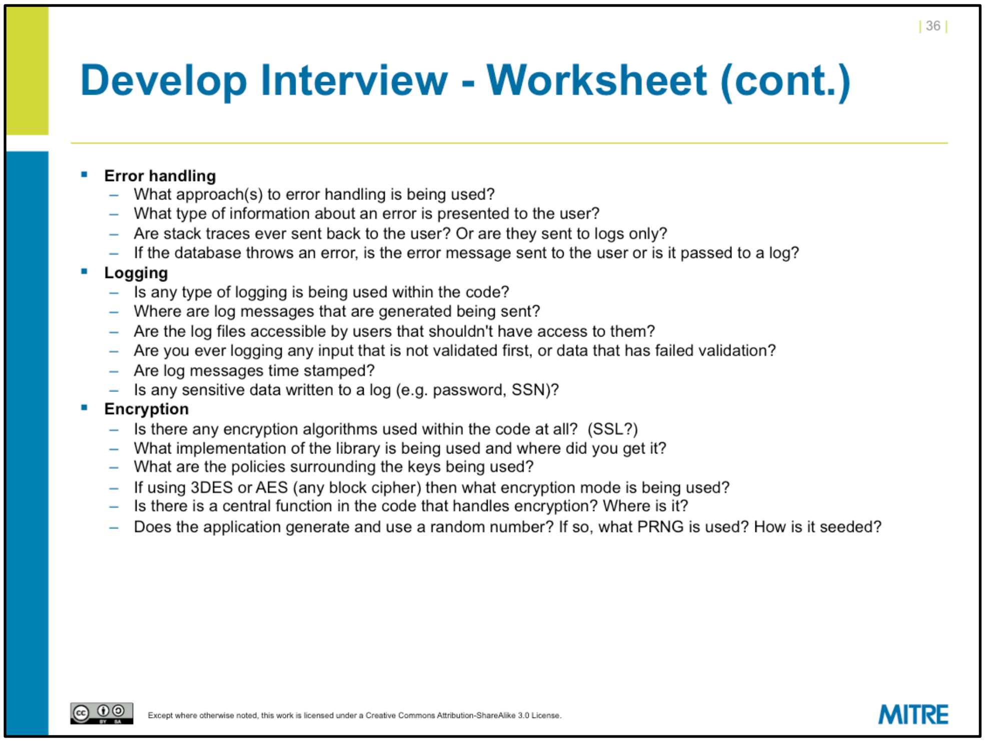
MITRE

Mark and Drew give an example interview



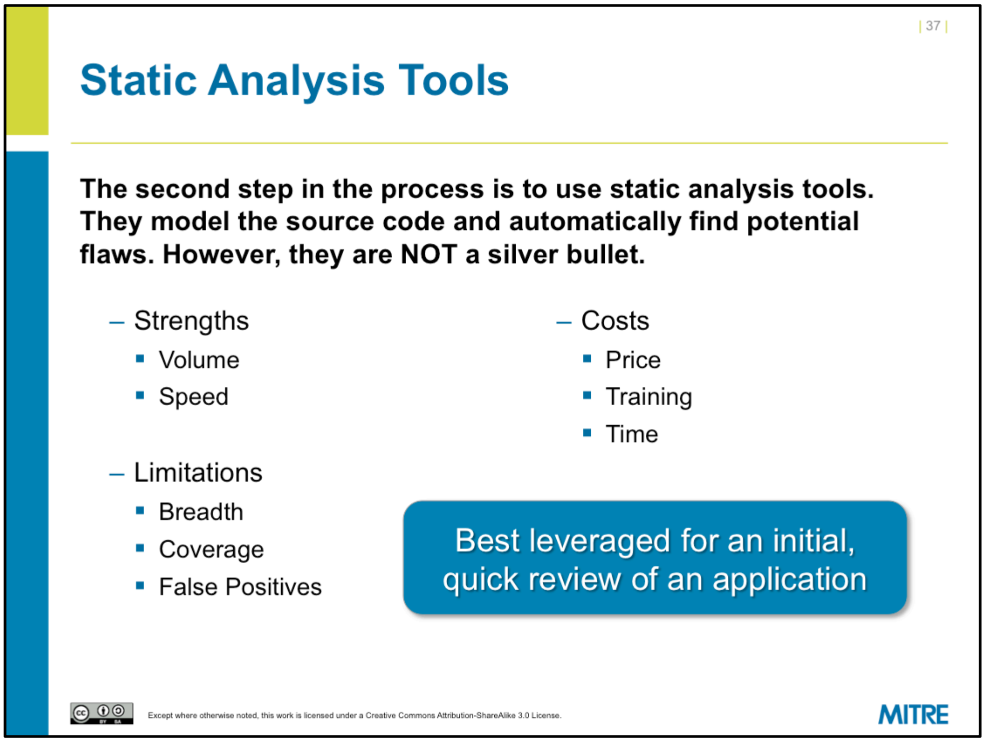
Develop Interview - Worksheet

- **Authentication**
  - Are users of the application authenticated or is everyone treated as an anonymous user?
  - What factors are being used for authentication? For example, passwords, certificates, biometrics.
  - If passwords are being used, then are there any policies in place regarding complexity or age?
  - Are there any ways to bypass the authentication for testing? Are there any alternate authentication paths?
- **Authorization**
  - Are there different roles that users can be assigned based upon the context of the job being performed?
  - Do you cache the authorization information? Or do you check authorization with each request?
  - Are there any sensitive data files stored under the web root, hence under no authorization?
  - Is authorization always checked on the server?
- **Session Management**
  - Is session state being managed / stored at all within the application and how is this being done?
  - How is the session id being generated?
  - If instead of passing a session id you are passing all the session data, is this data encrypted and signed?
  - If a user logs into the site, is the original session deleted upon login and a new session created?
  - Do sessions timeout at all?
  - Is there a logout function available?
  - If cookies are used, are there path and domain restrictions in the cookie?
- **Data Validation**
  - Is data received from the user validated?
  - Is data validated as soon as it comes in from the user or when it is used by the code?
  - How is the data validation being accomplished? (whitelisting, blacklisting, min/max, etc.)
  - Are you using a database? If so, are you using prepared statements?
  - Are you using HTML encode before user data goes back to the browser?
  - Are regular expressions used at all during data validation?



Develop Interview - Worksheet (cont.)

- **Error handling**
  - What approach(s) to error handling is being used?
  - What type of information about an error is presented to the user?
  - Are stack traces ever sent back to the user? Or are they sent to logs only?
  - If the database throws an error, is the error message sent to the user or is it passed to a log?
- **Logging**
  - Is any type of logging is being used within the code?
  - Where are log messages that are generated being sent?
  - Are the log files accessible by users that shouldn't have access to them?
  - Are you ever logging any input that is not validated first, or data that has failed validation?
  - Are log messages time stamped?
  - Is any sensitive data written to a log (e.g. password, SSN)?
- **Encryption**
  - Is there any encryption algorithms used within the code at all? (SSL?)
  - What implementation of the library is being used and where did you get it?
  - What are the policies surrounding the keys being used?
  - If using 3DES or AES (any block cipher) then what encryption mode is being used?
  - Is there is a central function in the code that handles encryption? Where is it?
  - Does the application generate and use a random number? If so, what PRNG is used? How is it seeded?



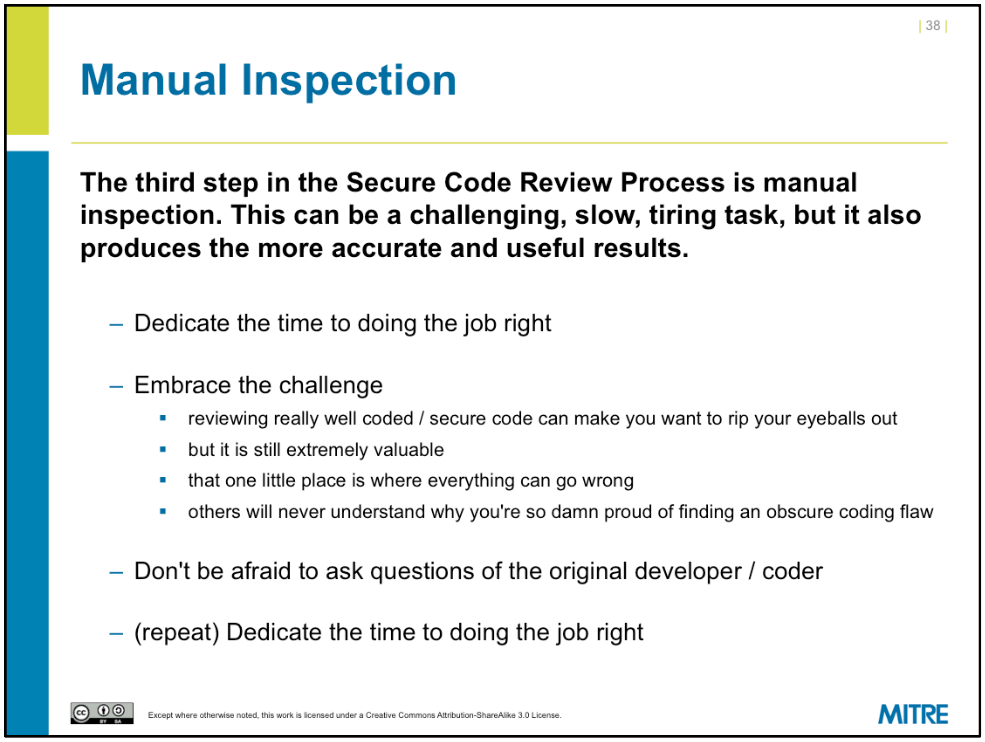
Static Analysis Tools

The second step in the process is to use static analysis tools. They model the source code and automatically find potential flaws. However, they are NOT a silver bullet.

- Strengths
  - Volume
  - Speed
- Limitations
  - Breadth
  - Coverage
  - False Positives
- Costs
  - Price
  - Training
  - Time

Best leveraged for an initial, quick review of an application

MITRE

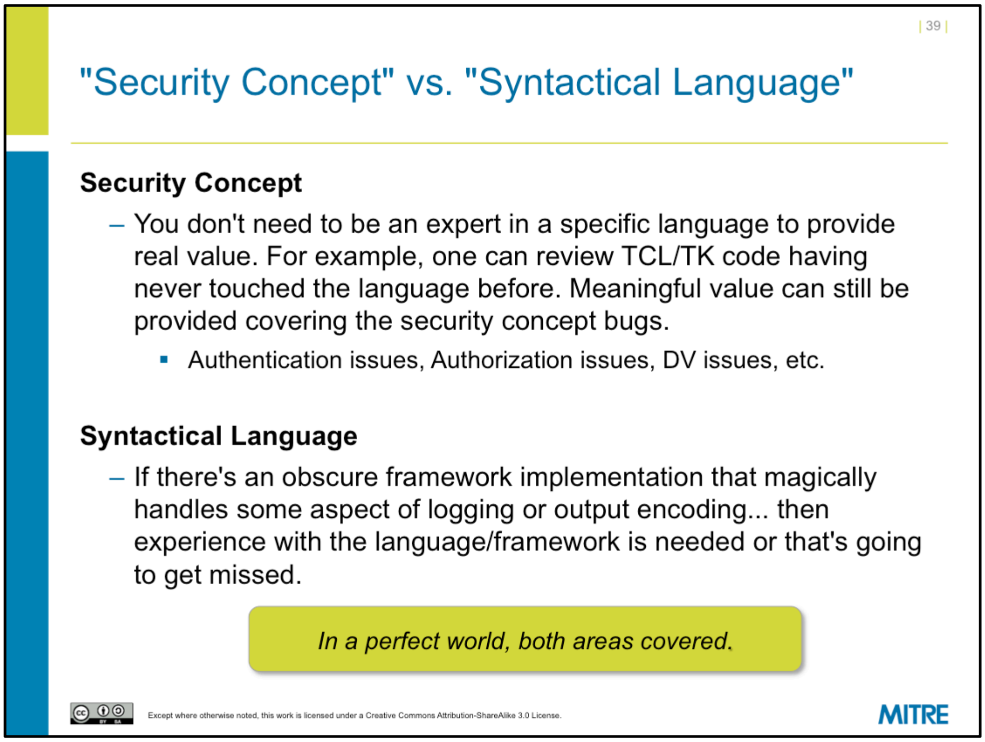


Manual Inspection

The third step in the Secure Code Review Process is manual inspection. This can be a challenging, slow, tiring task, but it also produces the more accurate and useful results.

- Dedicate the time to doing the job right
- Embrace the challenge
  - reviewing really well coded / secure code can make you want to rip your eyeballs out
  - but it is still extremely valuable
  - that one little place is where everything can go wrong
  - others will never understand why you're so damn proud of finding an obscure coding flaw
- Don't be afraid to ask questions of the original developer / coder
- (repeat) Dedicate the time to doing the job right

If it's going to take you a few hours digging through a manual to figure out if a Framework does something... that's likely a bad choice. Go ask someone who already knows the language/framework/etc. well enough to be coding in it. They may not know the answer, and then you have to look it up. But often, they can save you a ton of time and point you right at the info you need or tell you what the system does or does not do.



| 39 |

"Security Concept" vs. "Syntactical Language"

Security Concept

- You don't need to be an expert in a specific language to provide real value. For example, one can review TCL/TK code having never touched the language before. Meaningful value can still be provided covering the security concept bugs.
  - Authentication issues, Authorization issues, DV issues, etc.

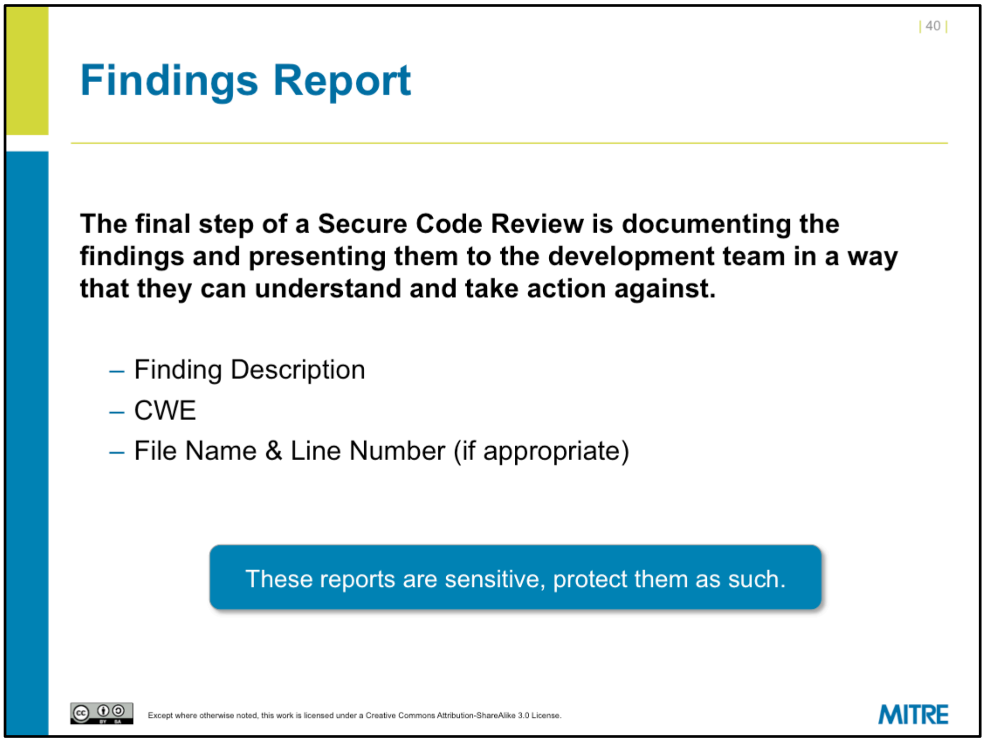
Syntactical Language

- If there's an obscure framework implementation that magically handles some aspect of logging or output encoding... then experience with the language/framework is needed or that's going to get missed.

In a perfect world, both areas covered.

MITRE

One of the things that will become apparent very quickly doing a manual review is the difference between knowledge of security concepts and the syntactical language.



| 40 |

Findings Report

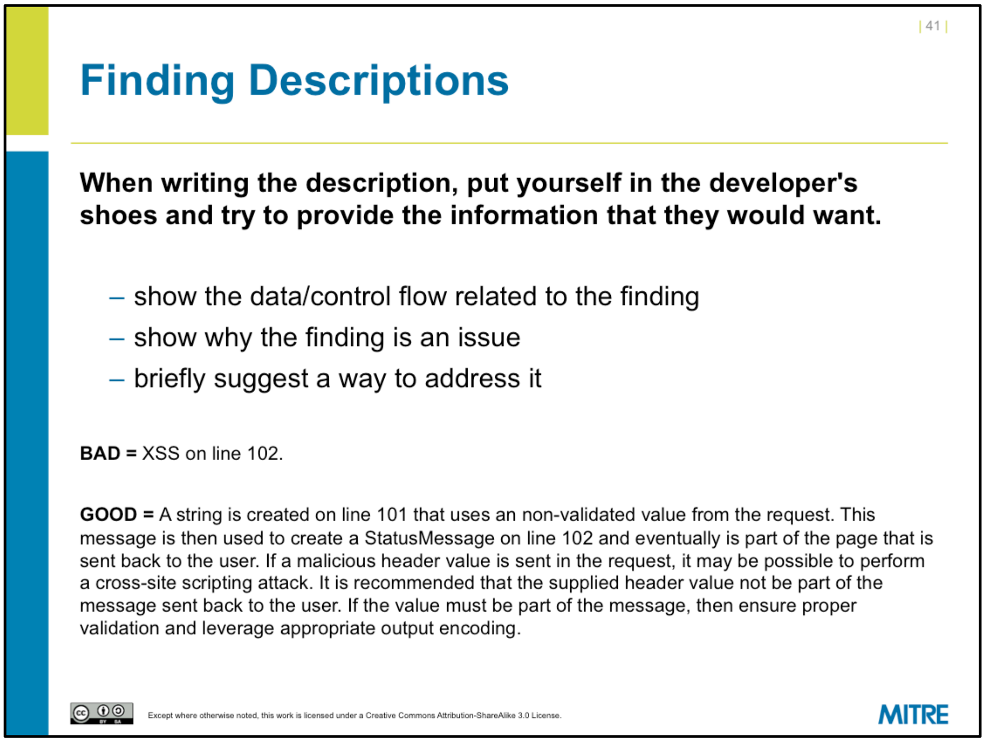
The final step of a Secure Code Review is documenting the findings and presenting them to the development team in a way that they can understand and take action against.

- Finding Description
- CWE
- File Name & Line Number (if appropriate)

These reports are sensitive, protect them as such.

MITRE

a reminder, that especially if they're doing a review on code that may already be in production, that finding a security flaw is sensitive info. Yes, you have to document and share that info with those who need-to-know to get it fixed. But that information should be protected, provided only to those who have a need to have the details, findings documents should likely be encrypted as they're sent around, etc.



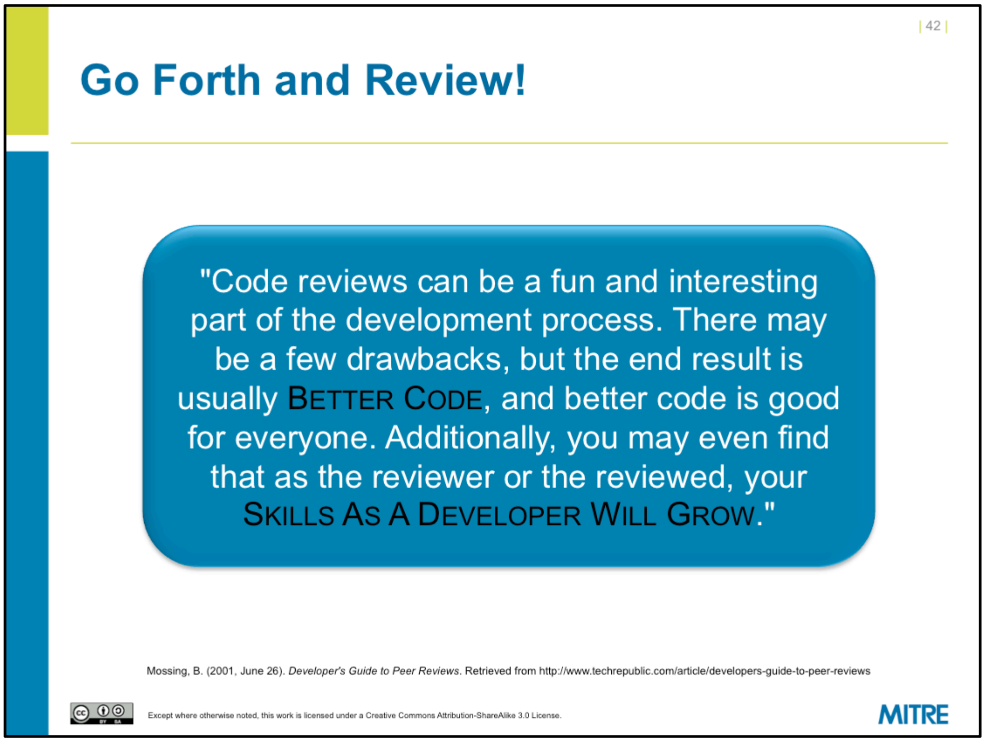
Finding Descriptions

When writing the description, put yourself in the developer's shoes and try to provide the information that they would want.

- show the data/control flow related to the finding
- show why the finding is an issue
- briefly suggest a way to address it

BAD = XSS on line 102.

GOOD = A string is created on line 101 that uses an non-validated value from the request. This message is then used to create a StatusMessage on line 102 and eventually is part of the page that is sent back to the user. If a malicious header value is sent in the request, it may be possible to perform a cross-site scripting attack. It is recommended that the supplied header value not be part of the message sent back to the user. If the value must be part of the message, then ensure proper validation and leverage appropriate output encoding.



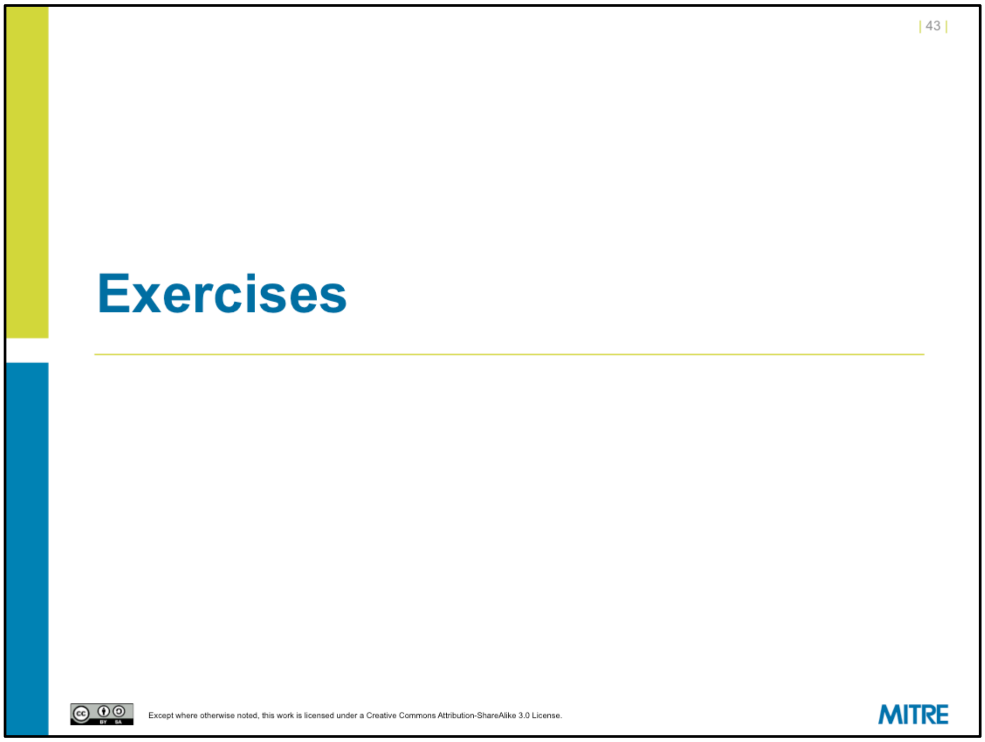
Go Forth and Review!

"Code reviews can be a fun and interesting part of the development process. There may be a few drawbacks, but the end result is usually Better Code, and better code is good for everyone. Additionally, you may even find that as the reviewer or the reviewed, your Skills As A Developer Will Grow."

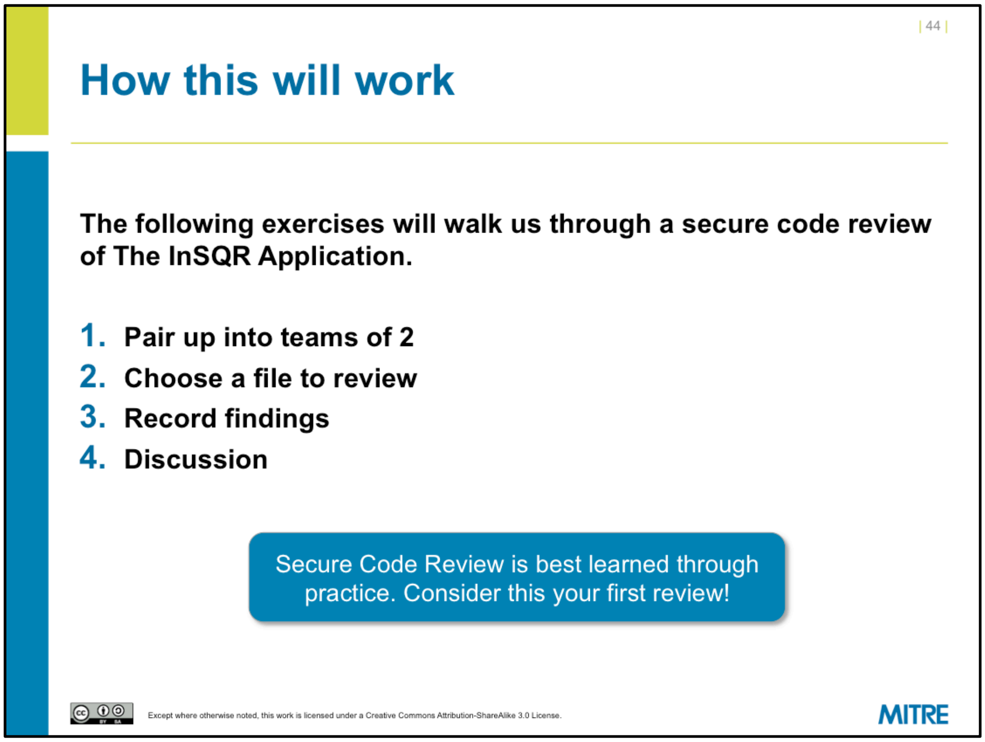
Mossing, B. (2001, June 26). *Developer's Guide to Peer Reviews*. Retrieved from http://www.techrepublic.com/article/developers-guide-to-peer-reviews

Go forth and review

Peer reviews can be a fun and interesting part of the development process. There may be a few drawbacks, but the end result is usually better code, and better code is good for everyone. Additionally, you may even find that as the reviewer or the reviewed, your skills as a developer will grow.



Exercises



How this will work

The following exercises will walk us through a secure code review of The InSQR Application.

1. **Pair up into teams of 2**
2. **Choose a file to review**
3. **Record findings**
4. **Discussion**

Secure Code Review is best learned through practice. Consider this your first review!

Story #1 : Know what you are agreeing to

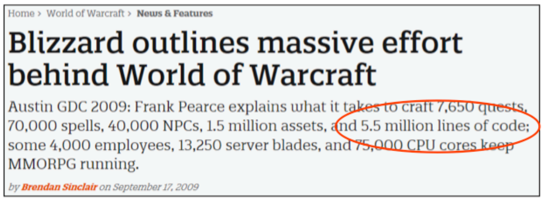
Jack: Can you review some code for us?

Jill: Sure! But I need to balance that with some other work, can I get you the findings by the end of the week?

Jack: No problem with the extra time. Thank you so much for helping me out! We need to get this code checked in.

Jill: Before you go, where can I get a copy of the code?

…



Home › World of Warcraft › **News & Features**

Blizzard outlines massive effort behind World of Warcraft

Austin GDC 2009: Frank Pearce explains what it takes to craft 7,650 quests, 70,000 spells, 40,000 NPCs, 1.5 million assets, and 5.5 million lines of code; some 4,000 employees, 13,250 server blades, and 75,000 CPU cores keep MMORPG running.

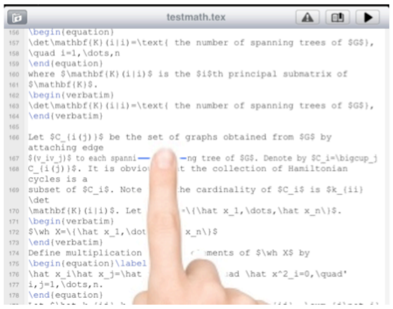
by ***Brendan Sinclair*** *on September 17, 2009*

Sinclair, Brendan. (September 17, 2009) Retrieved March 31, 2014, from http://www.gamespot.com/articles/blizzard-outlines-massive-effort-behind-world-of-warcraft/1100-6228615/

Prioritize and Focus

If you get stuck reviewing more code than you can possibly get through, then prioritize the code and focus on impact areas.

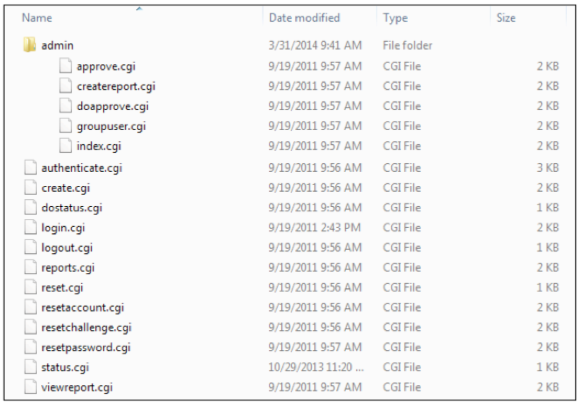
- **Start with files pertaining to high value targets**
  - authentication / authorization (e.g., login page)
  - database handler
  - sensitive data
  - shared libraries
- **Set for "special" strings**
  - password
  - key
  - connection
  - session
  - todo
  - exec / system



```
\begin{equation}
\det\mathbf{K}(i|i)=\text{ the number of spanning trees of $G$},
\quad i=1,\dots,n
\end{equation}
where $\mathbf{K}(i|i)$ is the $i$th principal submatrix of
$\mathbf{K}$.
\begin{verbatim}
\det\mathbf{K}(i|i)=\text{ the number of spanning trees of $G$},
\end{verbatim}

Let $C_{i(j)}$ be the set of graphs obtained from $G$ by
attaching edge
$(v_iv_j)$ to each spanni        ng tree of $G$. Denote by $C_i=\bigcup_j
C_{i(j)}$. It is obvio      t the collection of Hamiltonian
cycles is a
subset of $C_i$. Note        he cardinality of $C_i$ is $k_{ii}
\det
\mathbf{K}(i|i)$. Let        =\{\hat x_1,\dots,\hat x_n\}$.
\begin{verbatim}
$\wh X=\{\hat x_1,\dot          x_n\}$
\end{verbatim}
Define multiplication           ments of $\wh X$ by
\begin{equation}\label
\hat x_i\hat x_j=\hat             ad \hat x^2_i=0,\quad'
i,j=1,\dots,n.
\end{equation}
```

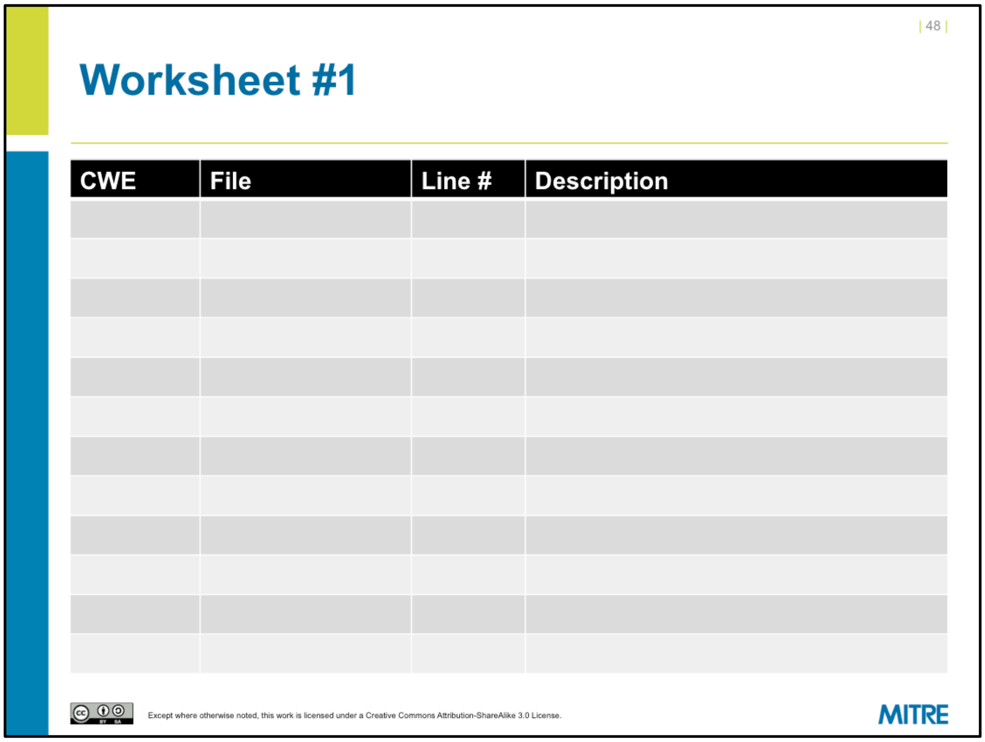
Exercise #1 : Where to Start?

Knowing very little about the codebase, which file(s) would you look at first? What would some other files of interest be?



Name	Date modified	Type	Size
admin	3/31/2014 9:41 AM	File folder	
approve.cgi	9/19/2011 9:57 AM	CGI File	2 KB
createreport.cgi	9/19/2011 9:57 AM	CGI File	2 KB
doapprove.cgi	9/19/2011 9:57 AM	CGI File	2 KB
groupuser.cgi	9/19/2011 9:57 AM	CGI File	2 KB
index.cgi	9/19/2011 9:57 AM	CGI File	2 KB
authenticate.cgi	9/19/2011 9:56 AM	CGI File	3 KB
create.cgi	9/19/2011 9:56 AM	CGI File	2 KB
dostatus.cgi	9/19/2011 9:56 AM	CGI File	1 KB
login.cgi	9/19/2011 2:43 PM	CGI File	2 KB
logout.cgi	9/19/2011 9:56 AM	CGI File	1 KB
reports.cgi	9/19/2011 9:56 AM	CGI File	2 KB
reset.cgi	9/19/2011 9:56 AM	CGI File	1 KB
resetaccount.cgi	9/19/2011 9:56 AM	CGI File	2 KB
resetchallenge.cgi	9/19/2011 9:56 AM	CGI File	2 KB
resetpassword.cgi	9/19/2011 9:57 AM	CGI File	2 KB
status.cgi	10/29/2013 11:20 ...	CGI File	1 KB
viewreport.cgi	9/19/2011 9:57 AM	CGI File	2 KB

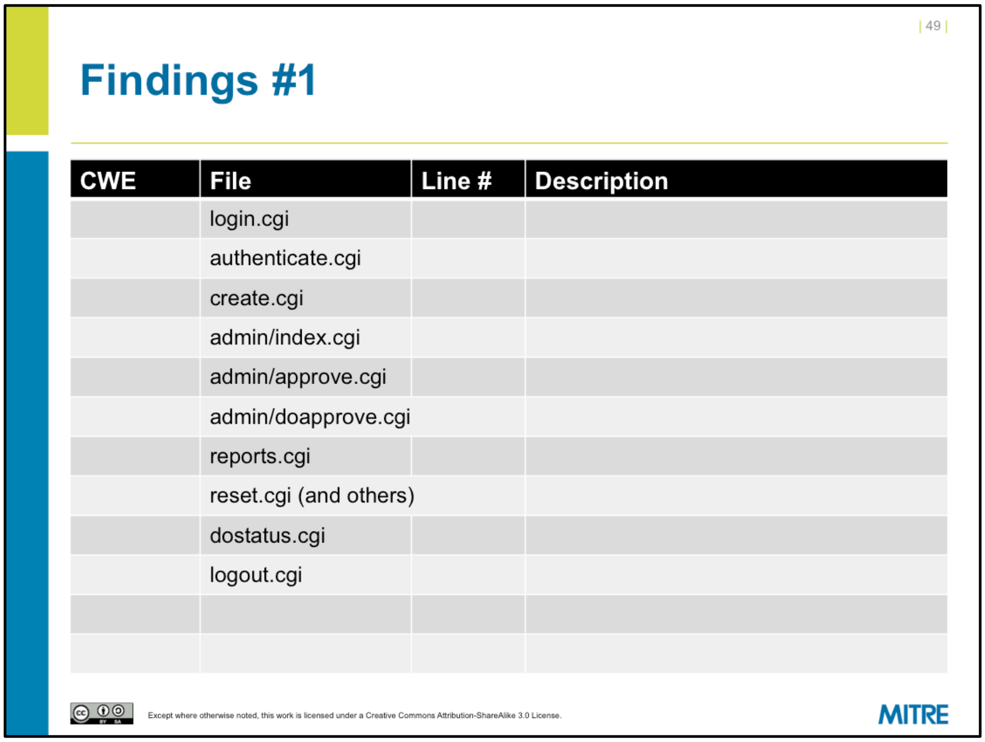
MITRE



Worksheet #1

CWE	File	Line #	Description

MITRE

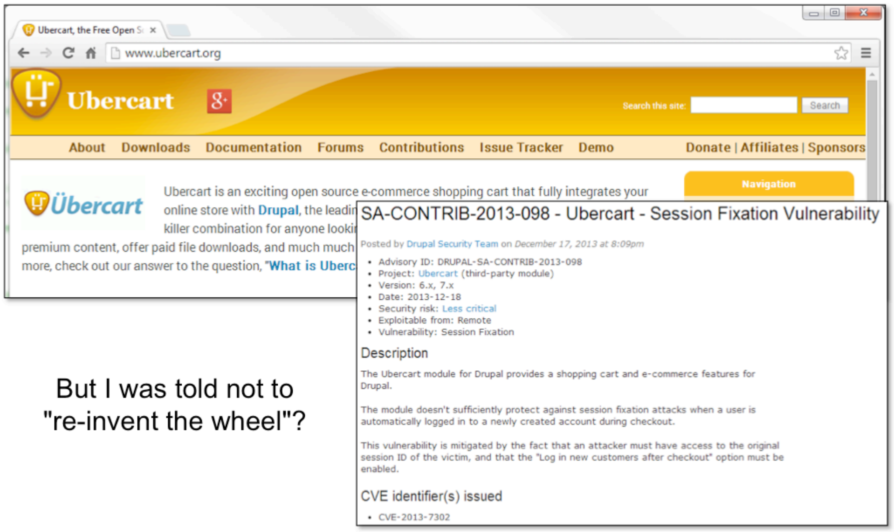


Findings #1

CWE	File	Line #	Description
	login.cgi		
	authenticate.cgi		
	create.cgi		
	admin/index.cgi		
	admin/approve.cgi		
	admin/doapprove.cgi		
	reports.cgi		
	reset.cgi (and others)		
	dostatus.cgi		
	logout.cgi		

MITRE

Story #2 : Ubercart Session Fixation



Ubercart is an exciting open source e-commerce shopping cart that fully integrates your online store with **Drupal**, the leadi killer combination for anyone looki premium content, offer paid file downloads, and much much more, check out our answer to the question, "**What is Uberc**

SA-CONTRIB-2013-098 - Ubercart - Session Fixation Vulnerability

Posted by Drupal Security Team on *December 17, 2013 at 8:09pm*

- Advisory ID: DRUPAL-SA-CONTRIB-2013-098
- Project: Ubercart (third-party module)
- Version: 6.x, 7.x
- Date: 2013-12-18
- Security risk: Less critical
- Exploitable from: Remote
- Vulnerability: Session Fixation

Description

The Ubercart module for Drupal provides a shopping cart and e-commerce features for Drupal.

The module doesn't sufficiently protect against session fixation attacks when a user is automatically logged in to a newly created account during checkout.

This vulnerability is mitigated by the fact that an attacker must have access to the original session ID of the victim, and that the "Log in new customers after checkout" option must be enabled.

CVE identifier(s) issued

- CVE-2013-7302

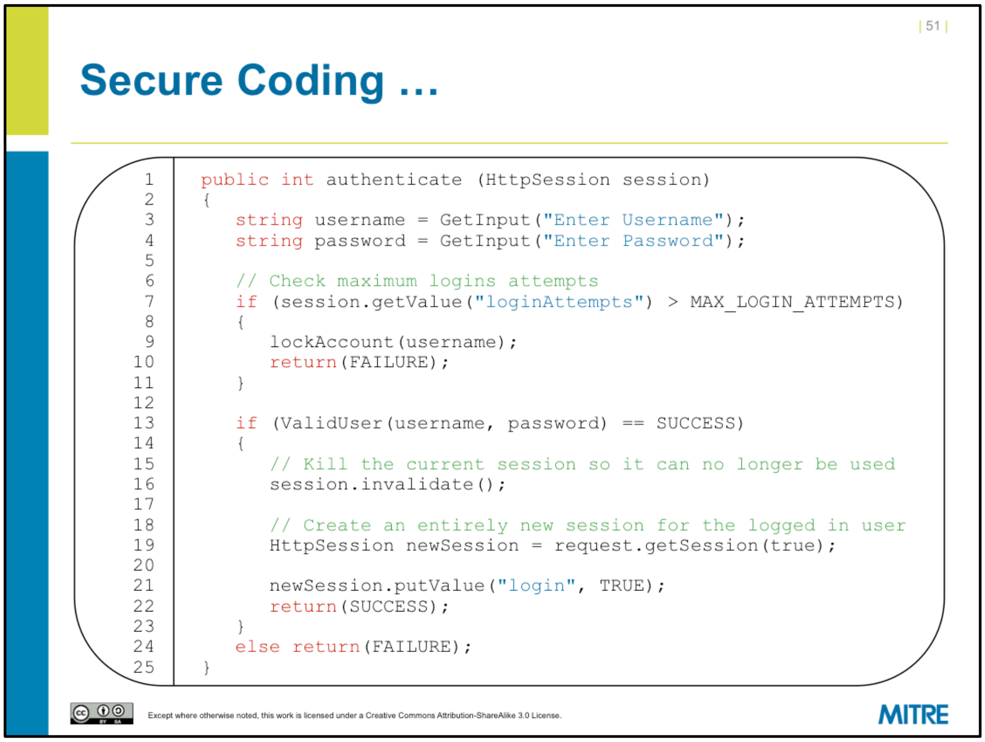
But I was told not to "re-invent the wheel"?

Our second story is related to an open source e-commerce shopping cart. A session fixation vulnerability was identified in December that allowed attackers to gain control of a user's session and access to their payment information. Remember back to the secure coding class about session fixation … one of the biggest issues is when an existing session id is used after a successful login. This allows an adversary to set the id and then trick/wait for the user into logging in. In this case, the "log in new customers after checkout feature" missed this detail. The application set up the new user and logged them in, but never invalidated the existing session.

Now you might be saying to yourself … but we were taught in the secure coding class to use existing solutions and not to re-invent the wheel. Yes, we did say that, and in this case you would still be burned. This just shows why you still need to know about these issues, and if practical you should talk to the developers or test the code that you are brining into your application. Unfortunately, as of today nothing is perfect.

https://drupal.org/node/2158651

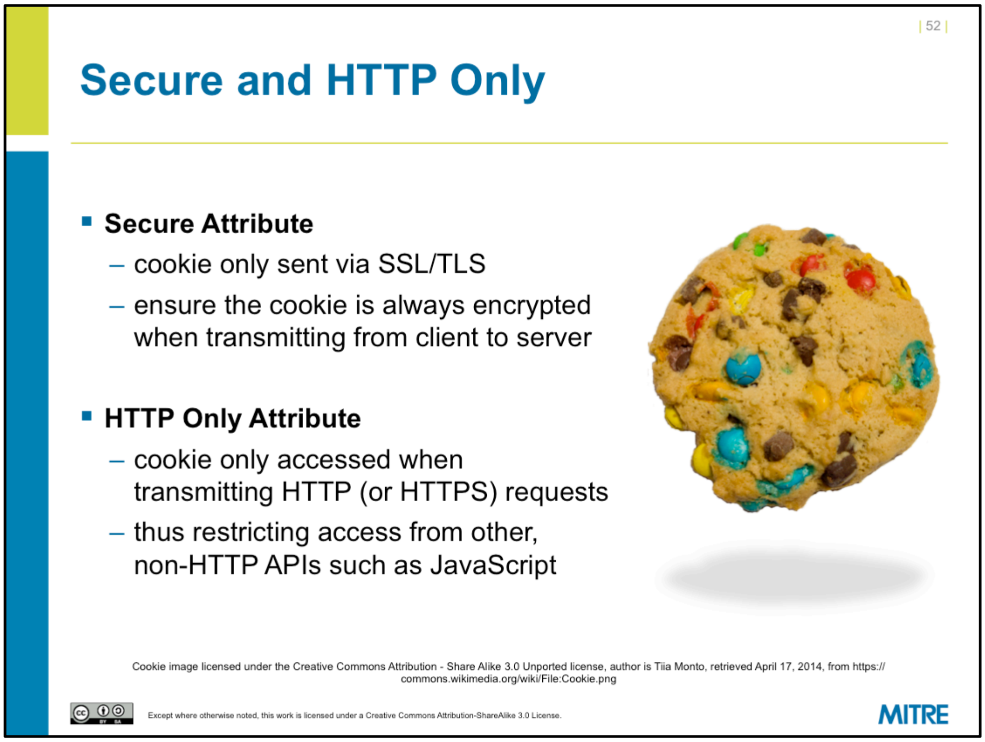
Mark to tie in with story about Wordpress.



Secure Coding …

```
public int authenticate (HttpSession session)
{
   string username = GetInput("Enter Username");
   string password = GetInput("Enter Password");

   // Check maximum logins attempts
   if (session.getValue("loginAttempts") > MAX_LOGIN_ATTEMPTS)
   {
      lockAccount(username);
      return(FAILURE);
   }

   if (ValidUser(username, password) == SUCCESS)
   {
      // Kill the current session so it can no longer be used
      session.invalidate();

      // Create an entirely new session for the logged in user
      HttpSession newSession = request.getSession(true);

      newSession.putValue("login", TRUE);
      return(SUCCESS);
   }
   else return(FAILURE);
}
```

this code is just to refresh about session fixation. This is how it is supposed to work.



Secure and HTTP Only

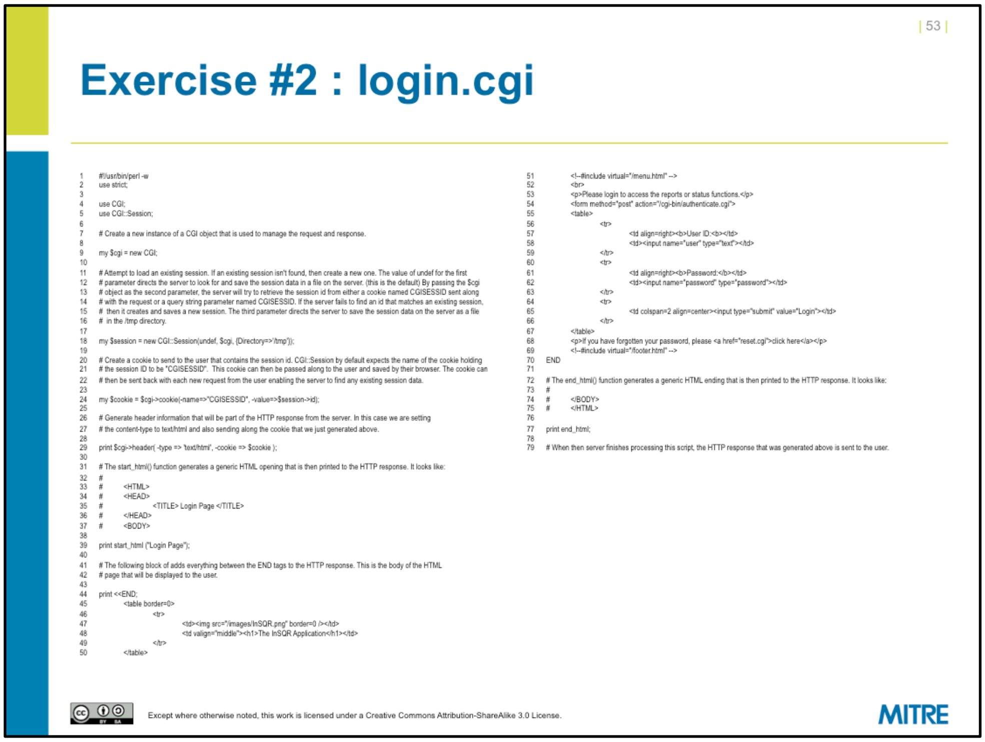
- **Secure Attribute**
  - cookie only sent via SSL/TLS
  - ensure the cookie is always encrypted when transmitting from client to server
- **HTTP Only Attribute**
  - cookie only accessed when transmitting HTTP (or HTTPS) requests
  - thus restricting access from other, non-HTTP APIs such as JavaScript

Cookie image licensed under the Creative Commons Attribution - Share Alike 3.0 Unported license, author is Tiia Monto, retrieved April 17, 2014, from https://commons.wikimedia.org/wiki/File:Cookie.png

MITRE

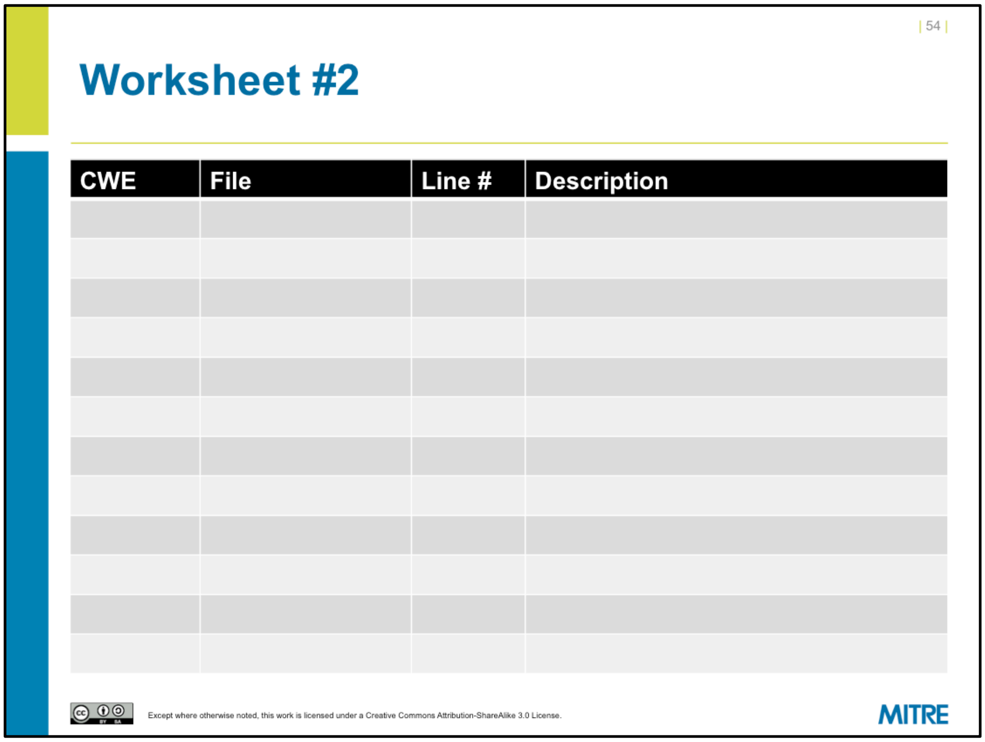
There are two other things related to session ids and more specifically the cookies that are often used to communicate these ids.

http://resources.infosecinstitute.com/securing-cookies-httponly-secure-flags/



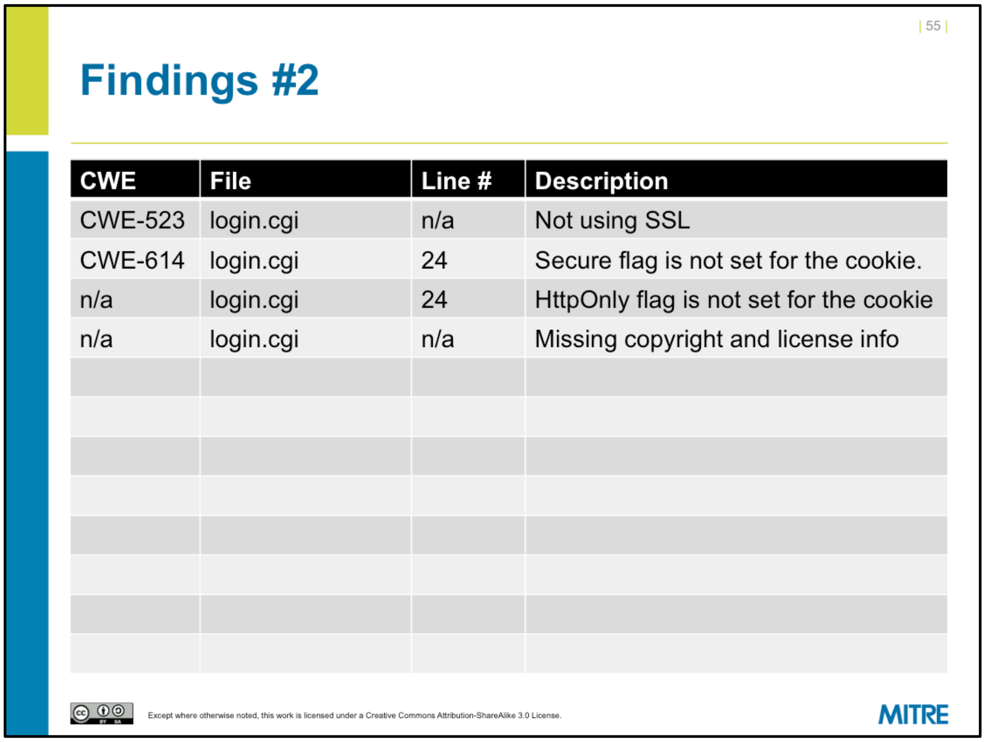
Exercise #2 : login.cgi

```
#!/usr/bin/perl -w
use strict;

use CGI;
use CGI::Session;

# Create a new instance of a CGI object that is used to manage the request and response.

my $cgi = new CGI;

# Attempt to load an existing session. If an existing session isn't found, then create a new one. The value of undef for the first
# parameter directs the server to look for and save the session data in a file on the server. (this is the default) By passing the $cgi
# object as the second parameter, the server will try to retrieve the session id from either a cookie named CGISESSID sent along
# with the request or a query string parameter named CGISESSID. If the server fails to find an id that matches an existing session,
#  then it creates and saves a new session. The third parameter directs the server to save the session data on the server as a file
#  in the /tmp directory.

my $session = new CGI::Session(undef, $cgi, {Directory=>'/tmp'});

# Create a cookie to send to the user that contains the session id. CGI::Session by default expects the name of the cookie holding
# the session ID to be "CGISESSID".  This cookie can then be passed along to the user and saved by their browser. The cookie can
# then be sent back with each new request from the user enabling the server to find any existing session data.

my $cookie = $cgi->cookie(-name=>"CGISESSID", -value=>$session->id);

# Generate header information that will be part of the HTTP response from the server. In this case we are setting
# the content-type to text/html and also sending along the cookie that we just generated above.

print $cgi->header( -type => 'text/html', -cookie => $cookie );

# The start_html() function generates a generic HTML opening that is then printed to the HTTP response. It looks like:
#
#       <HTML>
#       <HEAD>
#               <TITLE> Login Page </TITLE>
#       </HEAD>
#       <BODY>

print start_html ("Login Page");

# The following block of adds everything between the END tags to the HTTP response. This is the body of the HTML
# page that will be displayed to the user.

print <<END;
        <table border=0>
                <tr>
                        <td><img src="/images/InSQR.png" border=0 /></td>
                        <td valign="middle"><h1>The InSQR Application</h1></td>
                </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br>
        <p>Please login to access the reports or status functions.</p>
        <form method="post" action="/cgi-bin/authenticate.cgi">
        <table>
                <tr>
                        <td align=right><b>User ID:<b></td>
                        <td><input name="user" type="text"></td>
                </tr>
                <tr>
                        <td align=right><b>Password:</b></td>
                        <td><input name="password" type="password"></td>
                </tr>
                <tr>
                        <td colspan=2 align=center><input type="submit" value="Login"></td>
                </tr>
        </table>
        <p>If you have forgotten your password, please <a href="reset.cgi">click here</a></p>
        <!--#include virtual="/footer.html" -->
END

# The end_html() function generates a generic HTML ending that is then printed to the HTTP response. It looks like:
#
#       </BODY>
#       </HTML>

print end_html;

# When then server finishes processing this script, the HTTP response that was generated above is sent to the user.
```



Worksheet #2

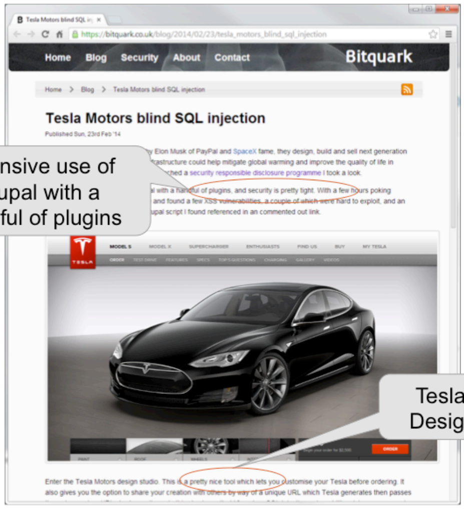
CWE	File	Line #	Description



Findings #2

CWE	File	Line #	Description
CWE-523	login.cgi	n/a	Not using SSL
CWE-614	login.cgi	24	Secure flag is not set for the cookie.
n/a	login.cgi	24	HttpOnly flag is not set for the cookie
n/a	login.cgi	n/a	Missing copyright and license info

Story #3 : Tesla Motors SQL Injection



Tesla Motors blind SQL injection

Published Sun, 23rd Feb '14

…y Elon Musk of PayPal and SpaceX fame, they design, build and sell next generation …frastructure could help mitigate global warming and improve the quality of life in …ched a security responsible disclosure programme I took a look.

…l with a handful of plugins, and security is pretty tight. With a few hours poking … and found a few XSS vulnerabilities, a couple of which were hard to exploit, and an …upal script I found referenced in an commented out link.

Enter the Tesla Motors design studio. This is a pretty nice tool which lets you customise your Tesla before ordering. It also gives you the option to share your creation with others by way of a unique URL which Tesla generates then passes

extensive use of Drupal with a handful of plugins

Tesla Motors Design Studio

"This is a pretty nice tool which lets you customize your Tesla before ordering. It also gives you the option to share your creation with others by way of a unique URL which Tesla generates then passes through a custom URL shortener. It was in this shortener that I found an SQL injection vulnerability, giving me access to Tesla's backend database, including access to all online customer records and admin access to the site."

access to back end database, all online customer records, and admin access to the site

"During testing I noticed that the script behaved a little differently depending on the input and investigated by injecting some quoted strings. After a bit of playing around I had it, a fairly standard blind attack vector:

' + sleep(10) + ' "

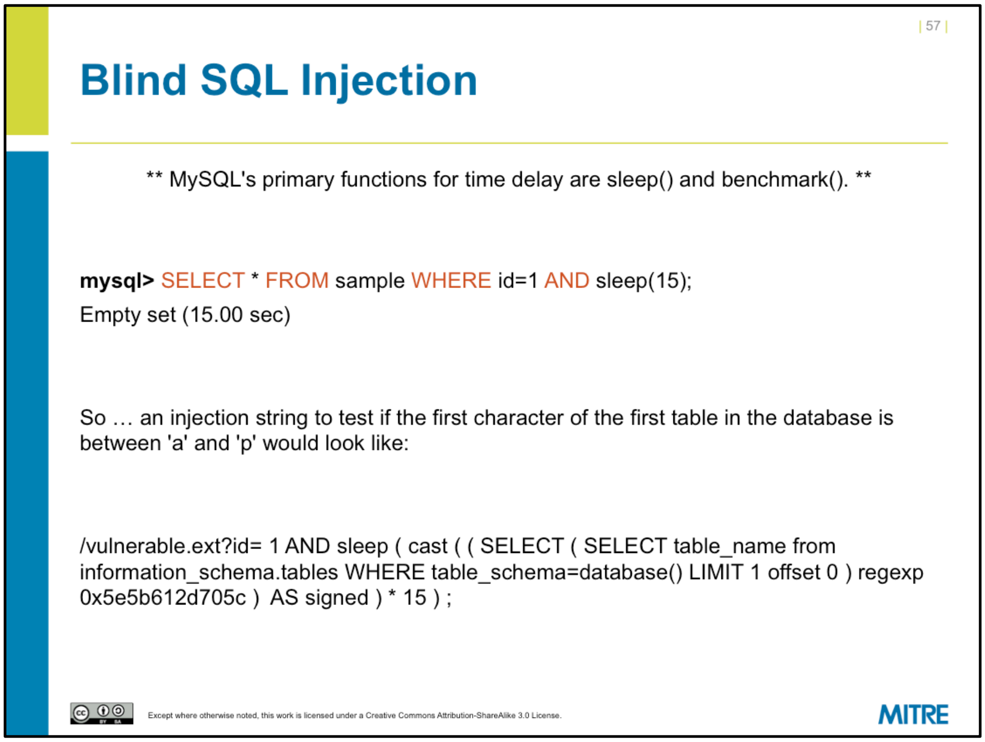
https://bitquark.co.uk/blog/2014/02/23/tesla_motors_blind_sql_injection

MITRE

This is a vulnerability in the Tesla Motors Design Studio.

flaw in the URL shortener routine, not "main" functionality. But still database access, so important.

https://bitquark.co.uk/blog/2014/02/23/tesla_motors_blind_sql_injection



Blind SQL Injection

** MySQL's primary functions for time delay are sleep() and benchmark(). **

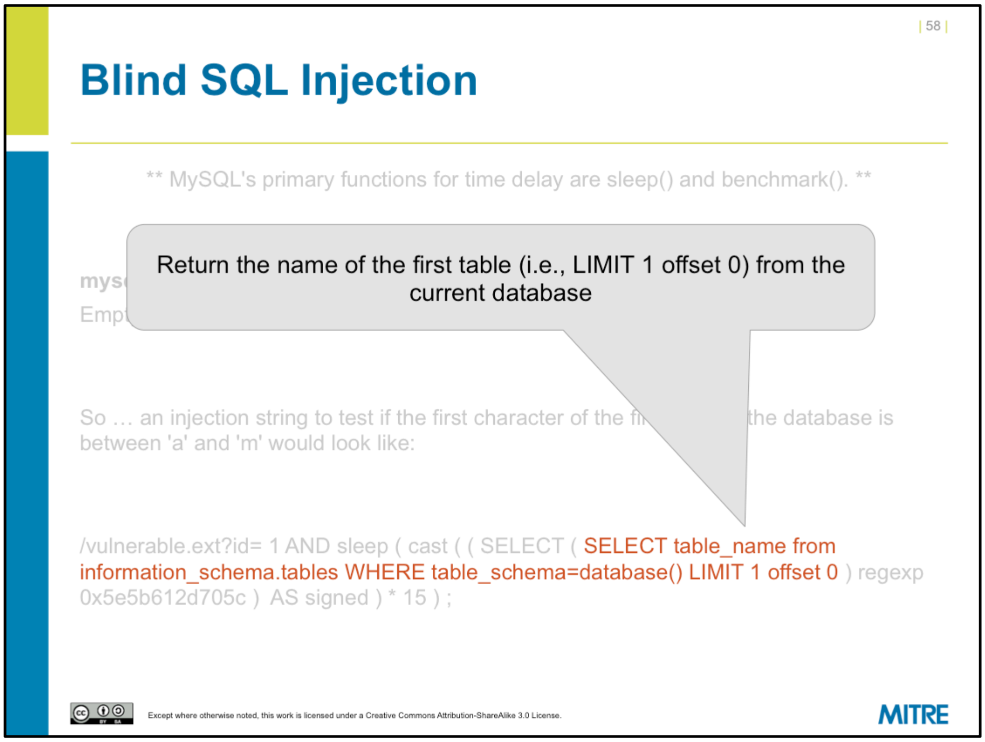
```
mysql> SELECT * FROM sample WHERE id=1 AND sleep(15);
Empty set (15.00 sec)
```

So … an injection string to test if the first character of the first table in the database is between 'a' and 'p' would look like:

```
/vulnerable.ext?id= 1 AND sleep ( cast ( ( SELECT ( SELECT table_name from
information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0 ) regexp
0x5e5b612d705c )  AS signed ) * 15 ) ;
```

MITRE

Sleeps (pauses) for the number of seconds given by the *duration* argument, then returns 0. If SLEEP() is interrupted, it returns 1. The duration may have a fractional part. This function was added in MySQL 5.0.12.



Blind SQL Injection

** MySQL's primary functions for time delay are sleep() and benchmark(). **

Return the name of the first table (i.e., LIMIT 1 offset 0) from the current database

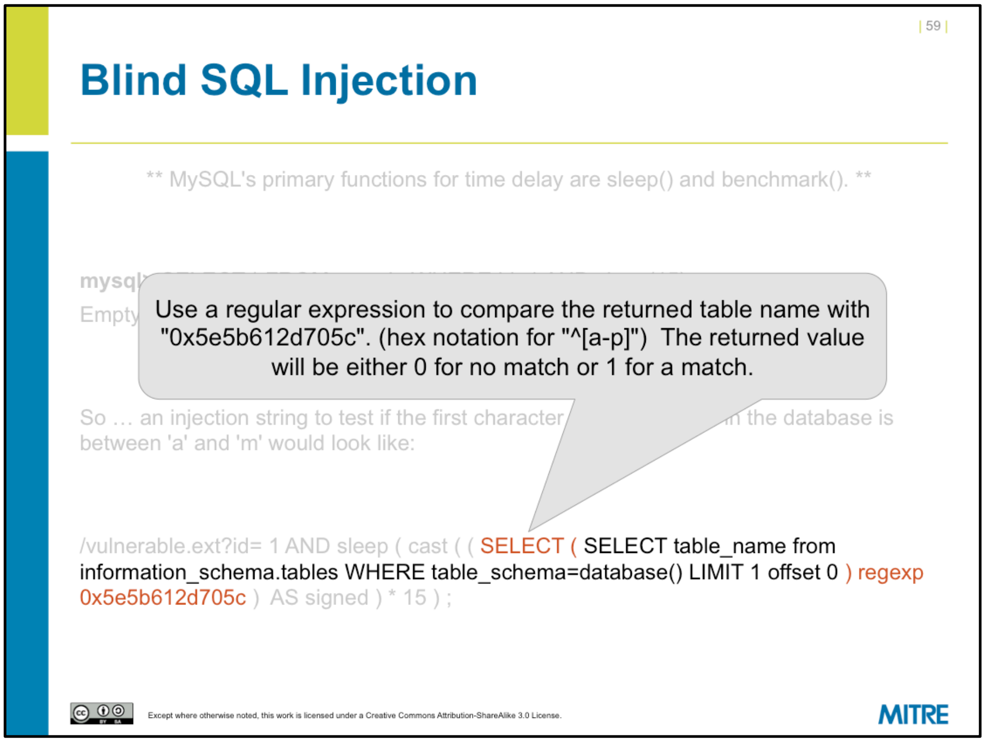
mys

Empt

So … an injection string to test if the first character of the fi the database is between 'a' and 'm' would look like:

```
/vulnerable.ext?id= 1 AND sleep ( cast ( ( SELECT ( SELECT table_name from
information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0 ) regexp
0x5e5b612d705c )  AS signed ) * 15 ) ;
```

MITRE



Blind SQL Injection

** MySQL's primary functions for time delay are sleep() and benchmark(). **

mysql

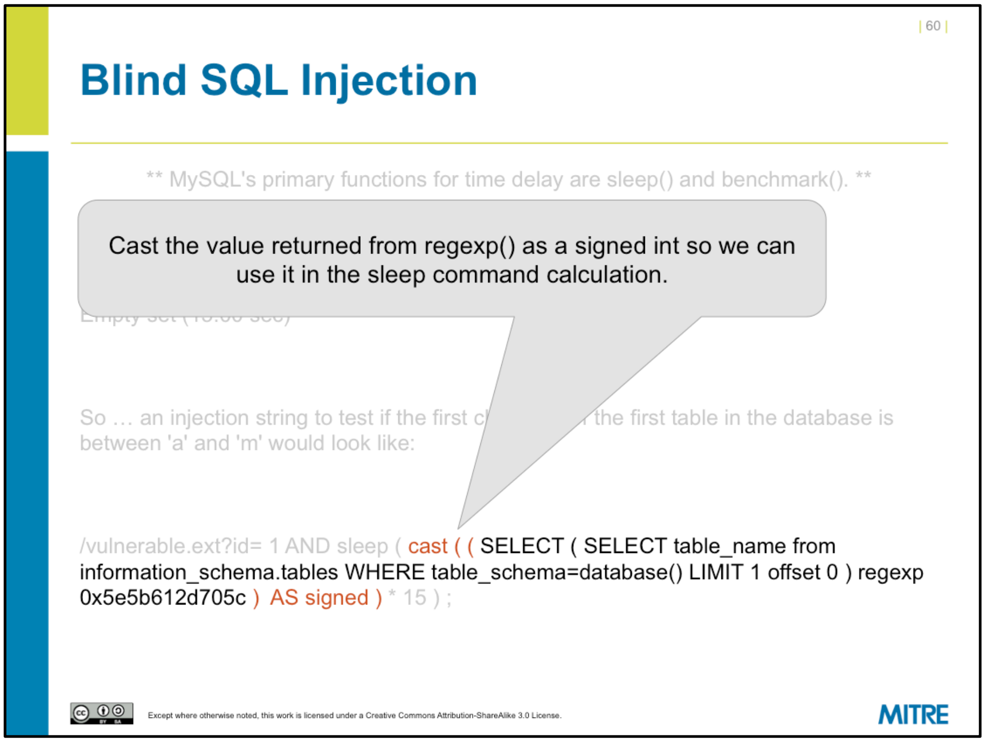
Empty

Use a regular expression to compare the returned table name with "0x5e5b612d705c". (hex notation for "^[a-p]") The returned value will be either 0 for no match or 1 for a match.

So … an injection string to test if the first character n the database is between 'a' and 'm' would look like:

/vulnerable.ext?id= 1 AND sleep (cast ((SELECT (SELECT table_name from information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0) regexp 0x5e5b612d705c) AS signed) * 15) ;

MITRE



Blind SQL Injection

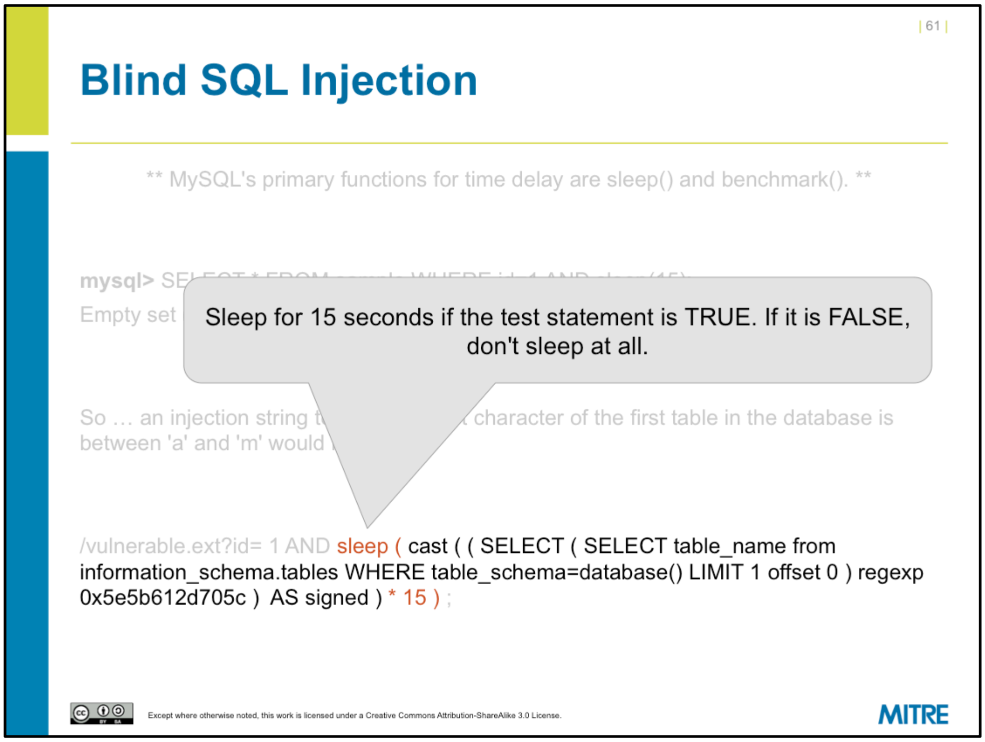
** MySQL's primary functions for time delay are sleep() and benchmark(). **

Cast the value returned from regexp() as a signed int so we can use it in the sleep command calculation.

Empty set (10.00 sec)

So … an injection string to test if the first c the first table in the database is between 'a' and 'm' would look like:

/vulnerable.ext?id= 1 AND sleep (cast ((SELECT (SELECT table_name from information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0) regexp 0x5e5b612d705c) AS signed) * 15) ;



Blind SQL Injection

** MySQL's primary functions for time delay are sleep() and benchmark(). **

mysql> SE

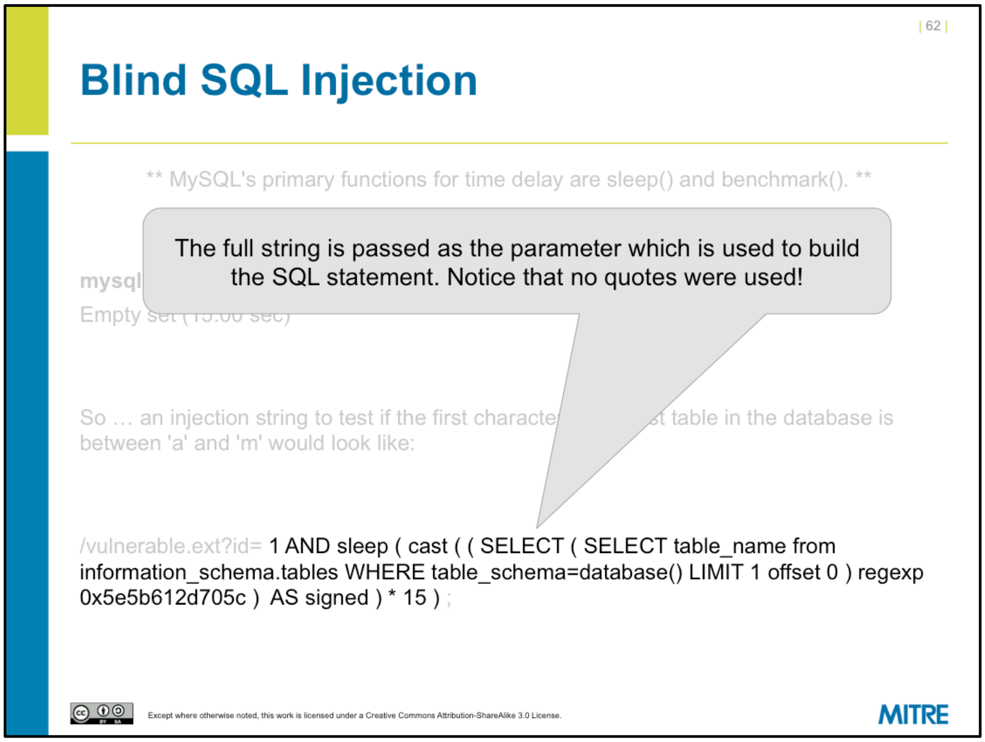
Empty set

Sleep for 15 seconds if the test statement is TRUE. If it is FALSE, don't sleep at all.

So … an injection string t character of the first table in the database is between 'a' and 'm' would

/vulnerable.ext?id= 1 AND sleep (cast ((SELECT (SELECT table_name from information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0) regexp 0x5e5b612d705c) AS signed) * 15) ;

MITRE



Blind SQL Injection

** MySQL's primary functions for time delay are sleep() and benchmark(). **

The full string is passed as the parameter which is used to build the SQL statement. Notice that no quotes were used!

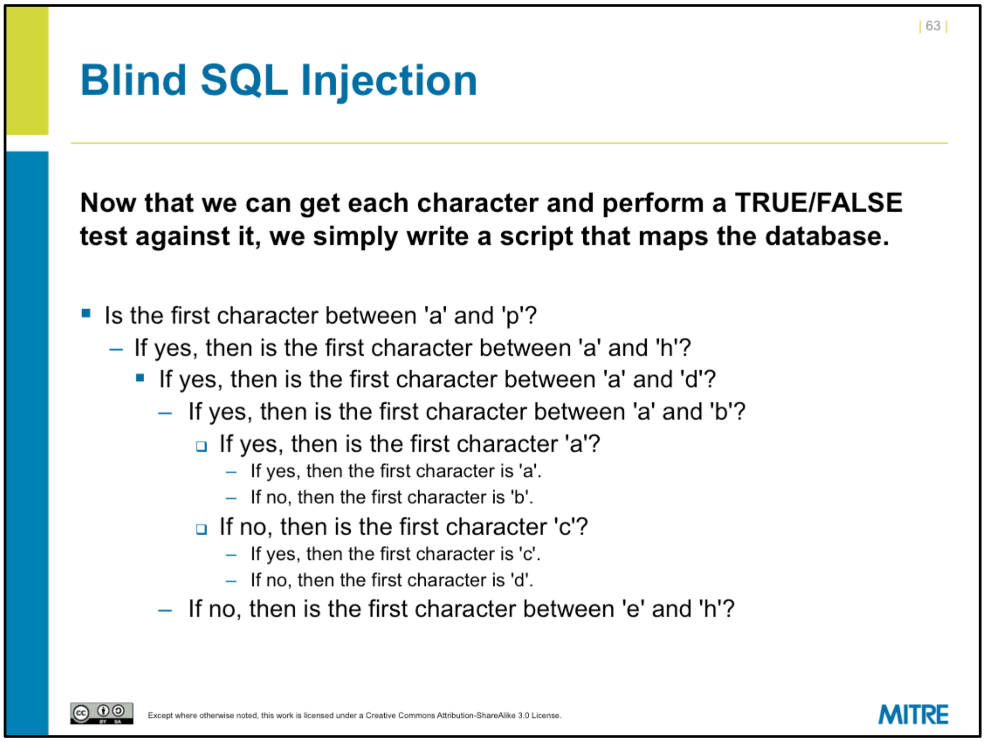
mysql

Empty set (15.00 sec)

So … an injection string to test if the first characte st table in the database is between 'a' and 'm' would look like:

```
/vulnerable.ext?id= 1 AND sleep ( cast ( ( SELECT ( SELECT table_name from
information_schema.tables WHERE table_schema=database() LIMIT 1 offset 0 ) regexp
0x5e5b612d705c )  AS signed ) * 15 ) ;
```

MITRE

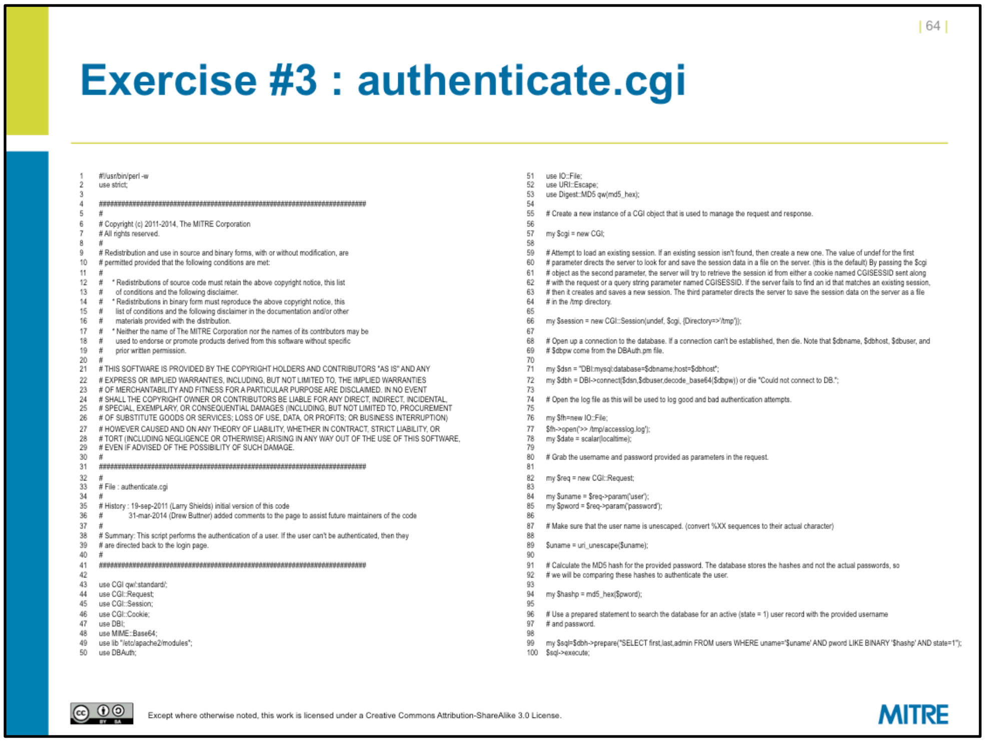


Blind SQL Injection

Now that we can get each character and perform a TRUE/FALSE test against it, we simply write a script that maps the database.

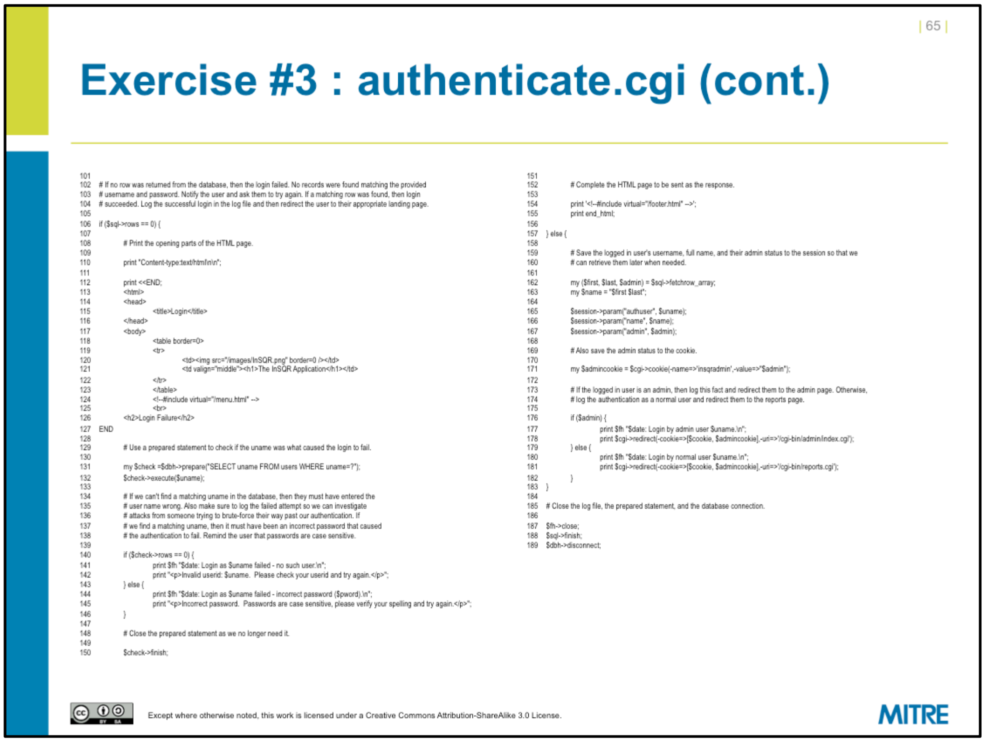
- Is the first character between 'a' and 'p'?
  - If yes, then is the first character between 'a' and 'h'?
    - If yes, then is the first character between 'a' and 'd'?
      - If yes, then is the first character between 'a' and 'b'?
        - ❑ If yes, then is the first character 'a'?
          - If yes, then the first character is 'a'.
          - If no, then the first character is 'b'.
        - ❑ If no, then is the first character 'c'?
          - If yes, then the first character is 'c'.
          - If no, then the first character is 'd'.
      - If no, then is the first character between 'e' and 'h'?

MITRE



Exercise #3 : authenticate.cgi

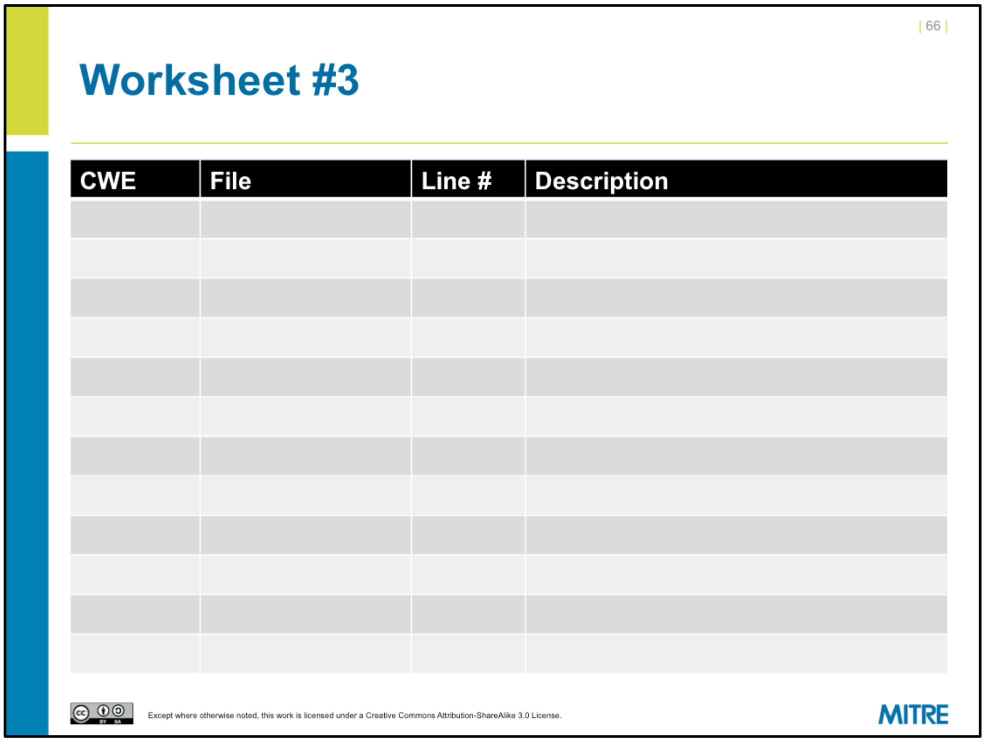
```
#!/usr/bin/perl -w
use strict;

##############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice, this list
#       of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright notice, this
#       list of conditions and the following disclaimer in the documentation and/or other
#       materials provided with the distribution.
#     * Neither the name of The MITRE Corporation nor the names of its contributors may be
#       used to endorse or promote products derived from this software without specific
#       prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
##############################################################################
#
# File : authenticate.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#            31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary: This script performs the authentication of a user. If the user can't be authenticated, then they
# are directed back to the login page.
#
##############################################################################

use CGI qw/:standard/;
use CGI::Request;
use CGI::Session;
use CGI::Cookie;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;
use IO::File;
use URI::Escape;
use Digest::MD5 qw(md5_hex);

# Create a new instance of a CGI object that is used to manage the request and response.

my $cgi = new CGI;

# Attempt to load an existing session. If an existing session isn't found, then create a new one. The value of undef for the first
# parameter directs the server to look for and save the session data in a file on the server. (this is the default) By passing the $cgi
# object as the second parameter, the server will try to retrieve the session id from either a cookie named CGISESSID sent along
# with the request or a query string parameter named CGISESSID. If the server fails to find an id that matches an existing session,
# then it creates and saves a new session. The third parameter directs the server to save the session data on the server as a file
# in the /tmp directory.

my $session = new CGI::Session(undef, $cgi, {Directory=>'/tmp'});

# Open up a connection to the database. If a connection can't be established, then die. Note that $dbname, $dbhost, $dbuser, and
# $dbpw come from the DBAuth.pm file.

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

# Open the log file as this will be used to log good and bad authentication attempts.

my $fh=new IO::File;
$fh->open('>> /tmp/accesslog.log');
my $date = scalar(localtime);

# Grab the username and password provided as parameters in the request.

my $req = new CGI::Request;

my $uname = $req->param('user');
my $pword = $req->param('password');

# Make sure that the user name is unescaped. (convert %XX sequences to their actual character)

$uname = uri_unescape($uname);

# Calculate the MD5 hash for the provided password. The database stores the hashes and not the actual passwords, so
# we will be comparing these hashes to authenticate the user.

my $hashp = md5_hex($pword);

# Use a prepared statement to search the database for an active (state = 1) user record with the provided username
# and password.

my $sql=$dbh->prepare("SELECT first,last,admin FROM users WHERE uname='$uname' AND pword LIKE BINARY '$hashp' AND state=1");
$sql->execute;
```



Exercise #3 : authenticate.cgi (cont.)

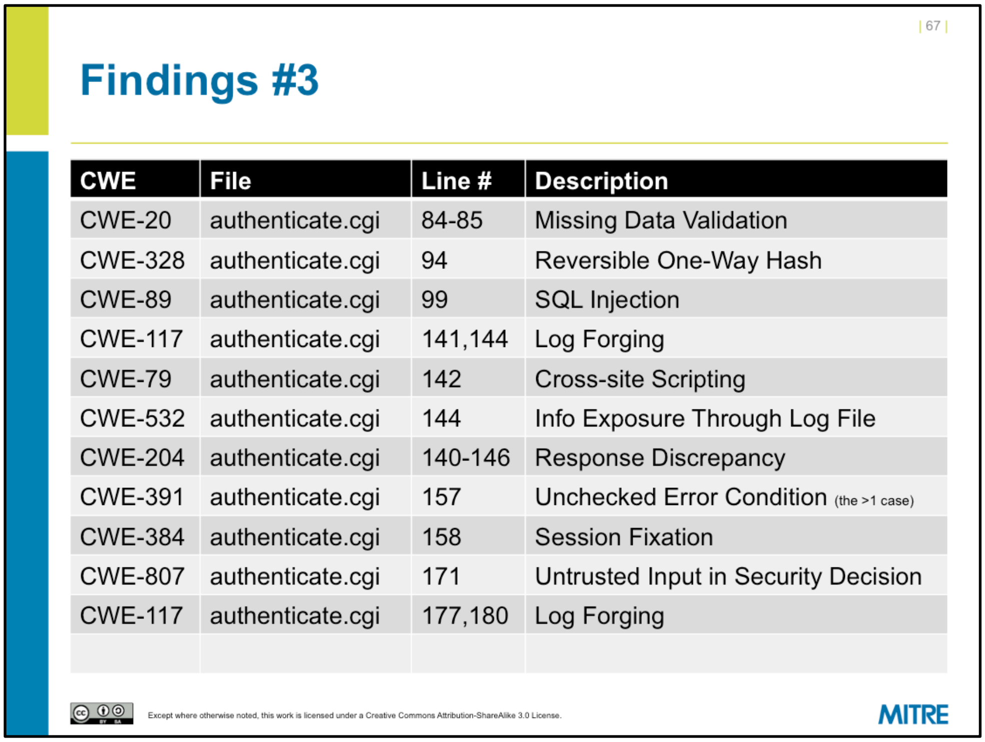
```

# If no row was returned from the database, then the login failed. No records were found matching the provided
# username and password. Notify the user and ask them to try again. If a matching row was found, then login
# succeeded. Log the successful login in the log file and then redirect the user to their appropriate landing page.

if ($sql->rows == 0) {

        # Print the opening parts of the HTML page.

        print "Content-type:text/html\n\n";

        print <<END;
        <html>
        <head>
                <title>Login</title>
        </head>
        <body>
                <table border=0>
                <tr>
                        <td><img src="/images/InSQR.png" border=0 /></td>
                        <td valign="middle"><h1>The InSQR Application</h1></td>
                </tr>
                </table>
                <!--#include virtual="/menu.html" -->
                <br>
        <h2>Login Failure</h2>
END

        # Use a prepared statement to check if the uname was what caused the login to fail.

        my $check =$dbh->prepare("SELECT uname FROM users WHERE uname=?");
        $check->execute($uname);

        # If we can't find a matching uname in the database, then they must have entered the
        # user name wrong. Also make sure to log the failed attempt so we can investigate
        # attacks from someone trying to brute-force their way past our authentication. If
        # we find a matching uname, then it must have been an incorrect password that caused
        # the authentication to fail. Remind the user that passwords are case sensitive.

        if ($check->rows == 0) {
                print $fh "$date: Login as $uname failed - no such user.\n";
                print "<p>Invalid userid: $uname.  Please check your userid and try again.</p>";
        } else {
                print $fh "$date: Login as $uname failed - incorrect password ($pword).\n";
                print "<p>Incorrect password.  Passwords are case sensitive, please verify your spelling and try again.</p>";
        }

        # Close the prepared statement as we no longer need it.

        $check->finish;

        # Complete the HTML page to be sent as the response.

        print '<!--#include virtual="/footer.html" -->';
        print end_html;

} else {

        # Save the logged in user's username, full name, and their admin status to the session so that we
        # can retrieve them later when needed.

        my ($first, $last, $admin) = $sql->fetchrow_array;
        my $name = "$first $last";

        $session->param("authuser", $uname);
        $session->param("name", $name);
        $session->param("admin", $admin);

        # Also save the admin status to the cookie.

        my $admincookie = $cgi->cookie(-name=>'insqradmin',-value=>"$admin");

        # If the logged in user is an admin, then log this fact and redirect them to the admin page. Otherwise,
        # log the authentication as a normal user and redirect them to the reports page.

        if ($admin) {
                print $fh "$date: Login by admin user $uname.\n";
                print $cgi->redirect(-cookie=>[$cookie, $admincookie],-uri=>'/cgi-bin/admin/index.cgi');
        } else {
                print $fh "$date: Login by normal user $uname.\n";
                print $cgi->redirect(-cookie=>[$cookie, $admincookie],-uri=>'/cgi-bin/reports.cgi');
        }
}

# Close the log file, the prepared statement, and the database connection.

$fh->close;
$sql->finish;
$dbh->disconnect;
```

MITRE



Worksheet #3

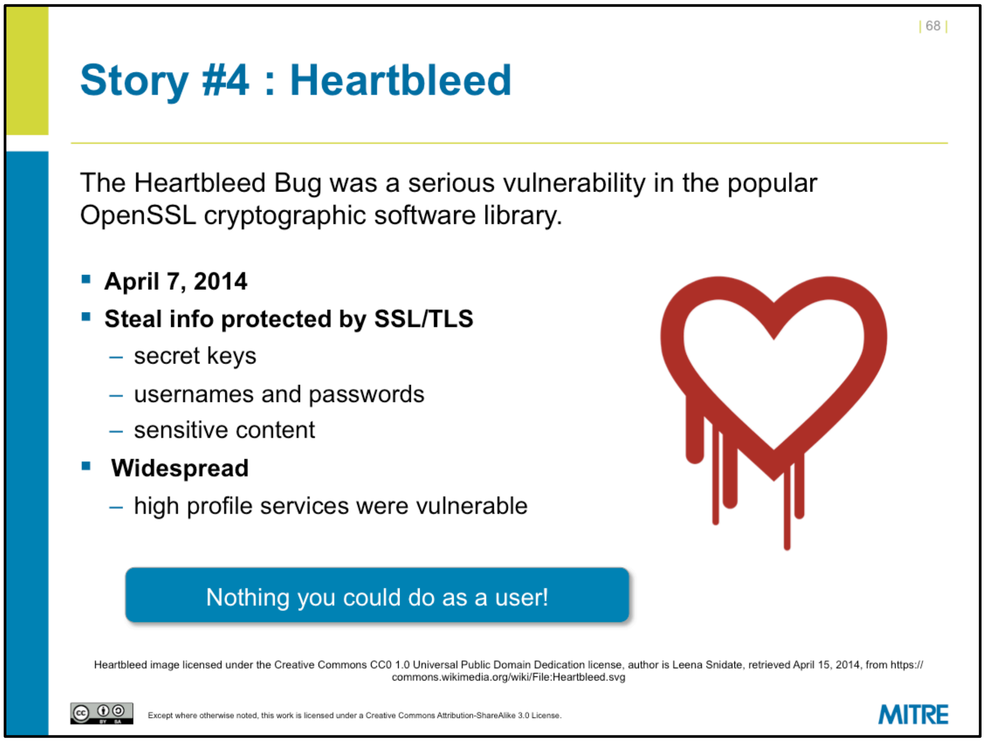
CWE	File	Line #	Description



Findings #3

CWE	File	Line #	Description
CWE-20	authenticate.cgi	84-85	Missing Data Validation
CWE-328	authenticate.cgi	94	Reversible One-Way Hash
CWE-89	authenticate.cgi	99	SQL Injection
CWE-117	authenticate.cgi	141,144	Log Forging
CWE-79	authenticate.cgi	142	Cross-site Scripting
CWE-532	authenticate.cgi	144	Info Exposure Through Log File
CWE-204	authenticate.cgi	140-146	Response Discrepancy
CWE-391	authenticate.cgi	157	Unchecked Error Condition (the >1 case)
CWE-384	authenticate.cgi	158	Session Fixation
CWE-807	authenticate.cgi	171	Untrusted Input in Security Decision
CWE-117	authenticate.cgi	177,180	Log Forging

MITRE



Story #4 : Heartbleed

The Heartbleed Bug was a serious vulnerability in the popular OpenSSL cryptographic software library.

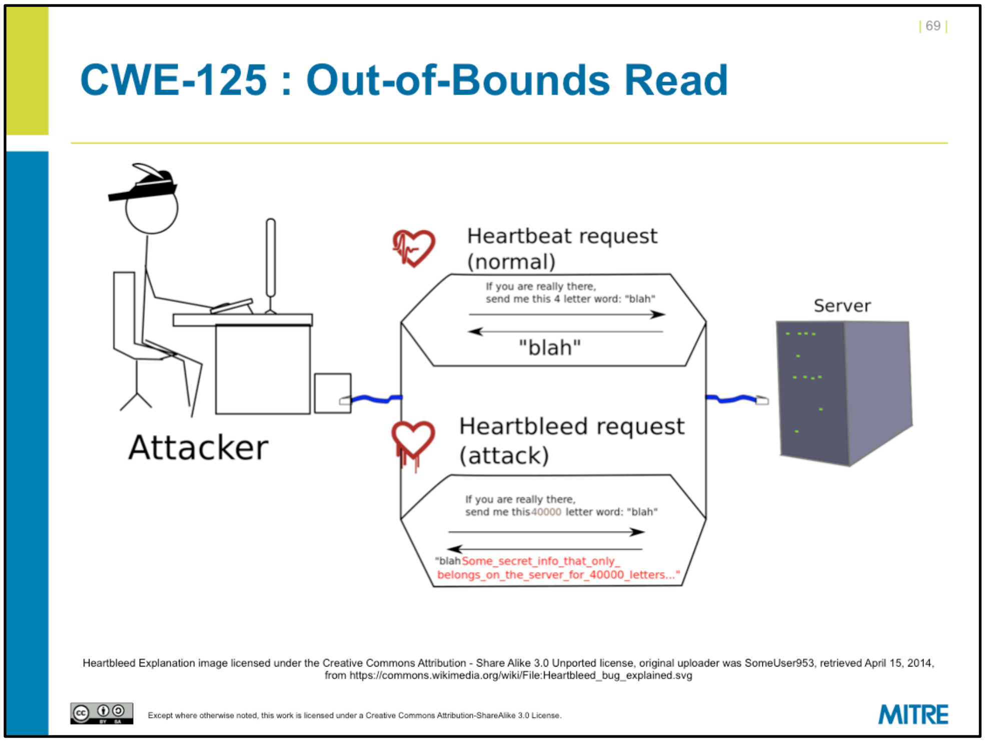
- **April 7, 2014**
- **Steal info protected by SSL/TLS**
  - secret keys
  - usernames and passwords
  - sensitive content
- **Widespread**
  - high profile services were vulnerable

Nothing you could do as a user!

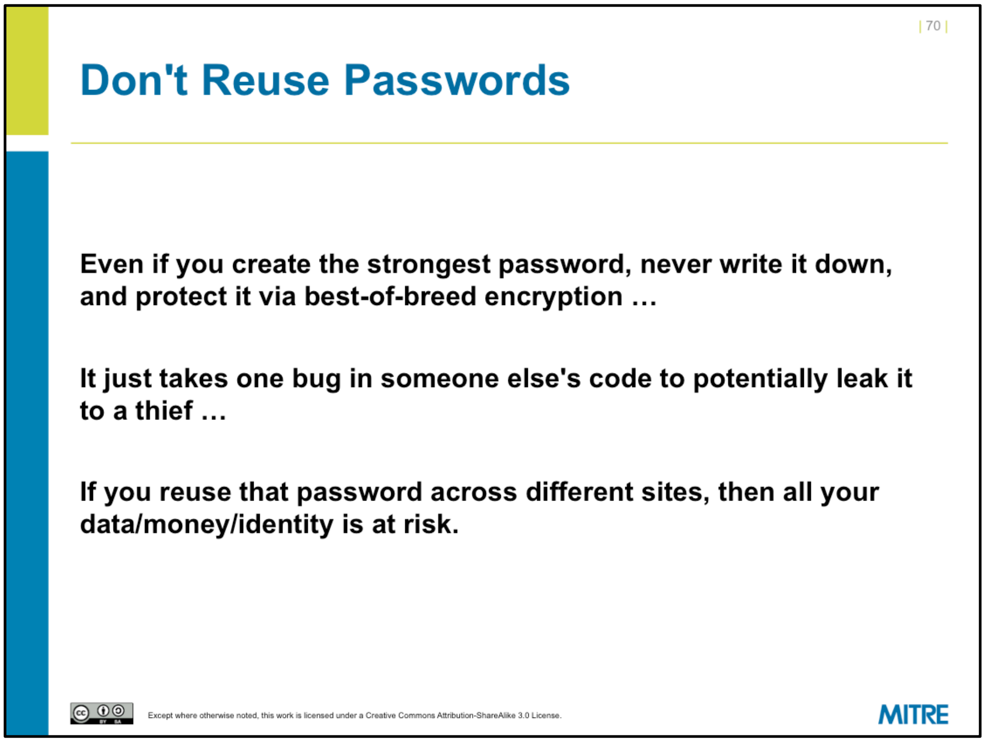
Heartbleed image licensed under the Creative Commons CC0 1.0 Universal Public Domain Dedication license, author is Leena Snidate, retrieved April 15, 2014, from https://commons.wikimedia.org/wiki/File:Heartbleed.svg

The Heartbleed bug occurs because of a chain of two distinct mistakes in the code. The first is an inconsistency in the stated length of the message body, and the body’s actual length. This type of weakness is described in detail by CWE-130 : "Improper Handling of Length Parameter Inconsistency". Following this weakness is an out-of-bounds memory read which is described in CWE-125 : "Out-of-bounds Read".

CWE-125 : Out-of-Bounds Read


Attacker
Heartbeat request (normal)
If you are really there, send me this 4 letter word: "blah"
"blah"
Server
Heartbleed request (attack)
If you are really there, send me this 40000 letter word: "blah"
"blah Some_secret_info_that_only_ belongs_on_the_server_for_40000_letters..."

Heartbleed Explanation image licensed under the Creative Commons Attribution - Share Alike 3.0 Unported license, original uploader was SomeUser953, retrieved April 15, 2014, from https://commons.wikimedia.org/wiki/File:Heartbleed_bug_explained.svg

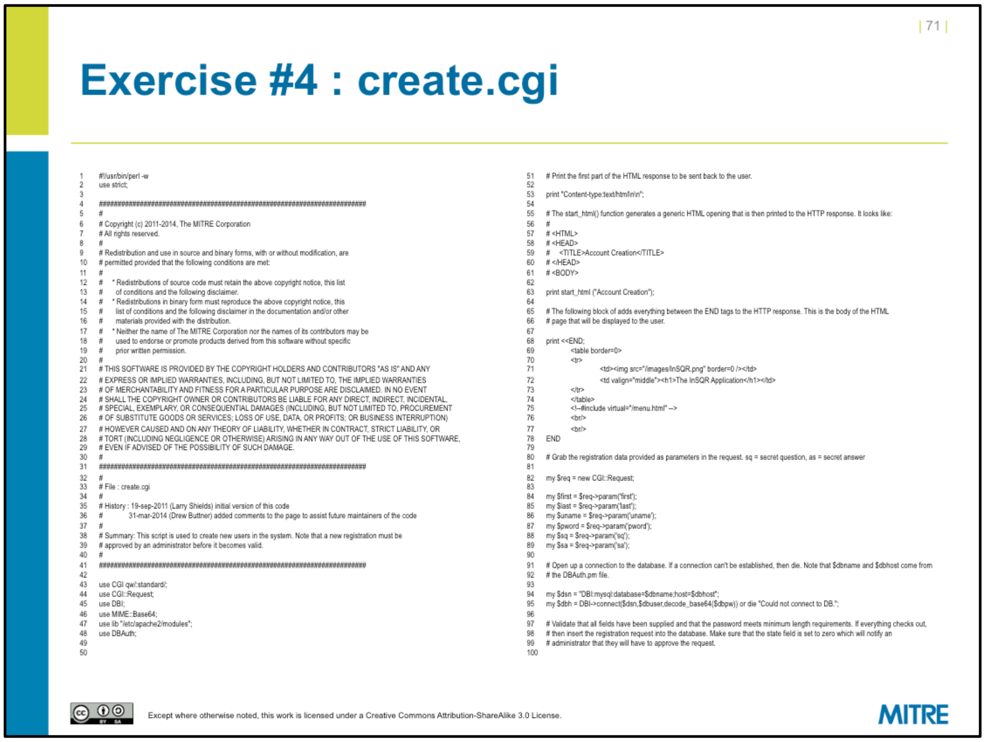


Don't Reuse Passwords

Even if you create the strongest password, never write it down, and protect it via best-of-breed encryption …

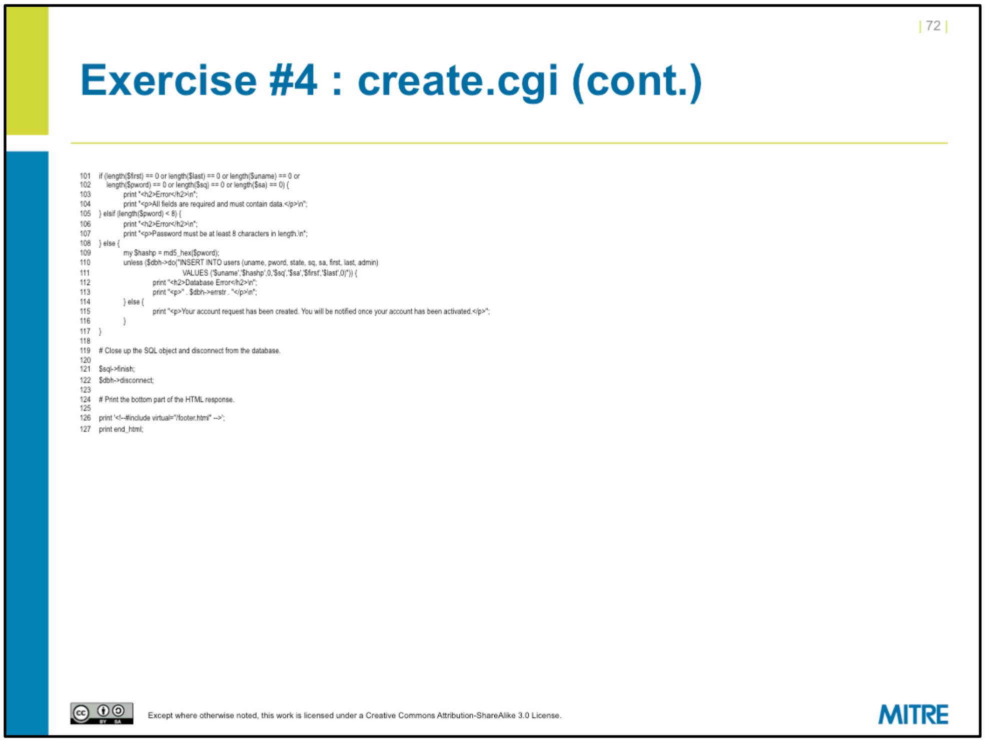
It just takes one bug in someone else's code to potentially leak it to a thief …

If you reuse that password across different sites, then all your data/money/identity is at risk.



Exercise #4 : create.cgi

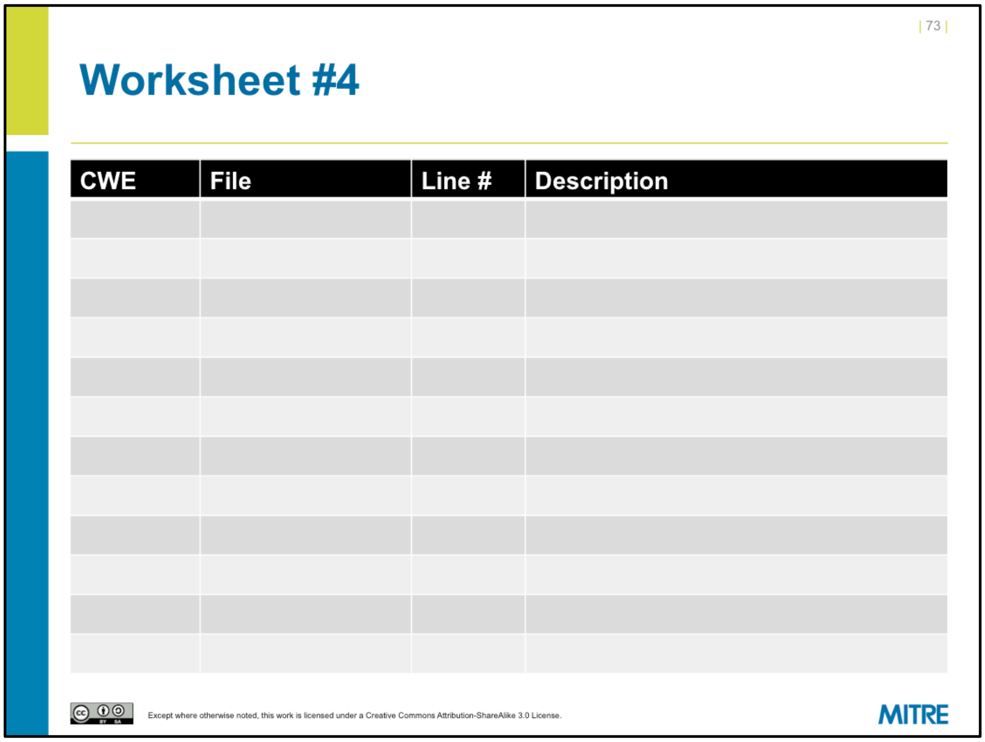
```
#!/usr/bin/perl -w
use strict;

################################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice, this list
#       of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright notice, this
#       list of conditions and the following disclaimer in the documentation and/or other
#       materials provided with the distribution.
#     * Neither the name of The MITRE Corporation nor the names of its contributors may be
#       used to endorse or promote products derived from this software without specific
#       prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
################################################################################
#
# File : create.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#           31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary: This script is used to create new users in the system. Note that a new registration must be
# approved by an administrator before it becomes valid.
#
################################################################################

use CGI qw/:standard/;
use CGI::Request;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;


# Print the first part of the HTML response to be sent back to the user.

print "Content-type:text/html\n\n";

# The start_html() function generates a generic HTML opening that is then printed to the HTTP response. It looks like:
#
# <HTML>
# <HEAD>
#    <TITLE>Account Creation</TITLE>
# </HEAD>
# <BODY>

print start_html ("Account Creation");

# The following block of adds everything between the END tags to the HTTP response. This is the body of the HTML
# page that will be displayed to the user.

print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
        <br/>
END

# Grab the registration data provided as parameters in the request. sq = secret question, as = secret answer

my $req = new CGI::Request;

my $first = $req->param('first');
my $last = $req->param('last');
my $uname = $req->param('uname');
my $pword = $req->param('pword');
my $sq = $req->param('sq');
my $sa = $req->param('sa');

# Open up a connection to the database. If a connection can't be established, then die. Note that $dbname and $dbhost come from
# the DBAuth.pm file.

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

# Validate that all fields have been supplied and that the password meets minimum length requirements. If everything checks out,
# then insert the registration request into the database. Make sure that the state field is set to zero which will notify an
# administrator that they will have to approve the request.

```



Exercise #4 : create.cgi (cont.)

```
if (length($first) == 0 or length($last) == 0 or length($uname) == 0 or
   length($pword) == 0 or length($sq) == 0 or length($sa) == 0) {
          print "<h2>Error</h2>\n";
          print "<p>All fields are required and must contain data.</p>\n";
} elsif (length($pword) < 8) {
          print "<h2>Error</h2>\n";
          print "<p>Password must be at least 8 characters in length.\n";
} else {
          my $hashp = md5_hex($pword);
          unless ($dbh->do("INSERT INTO users (uname, pword, state, sq, sa, first, last, admin)
                              VALUES ('$uname','$hashp',0,'$sq','$sa','$first','$last',0)")) {
                    print "<h2>Database Error</h2>\n";
                    print "<p>" . $dbh->errstr . "</p>\n";
          } else {
                    print "<p>Your account request has been created. You will be notified once your account has been activated.</p>";
          }
}

# Close up the SQL object and disconnect from the database.

$sql->finish;
$dbh->disconnect;

# Print the bottom part of the HTML response.

print '<!--#include virtual="/footer.html" -->';
print end_html;
```

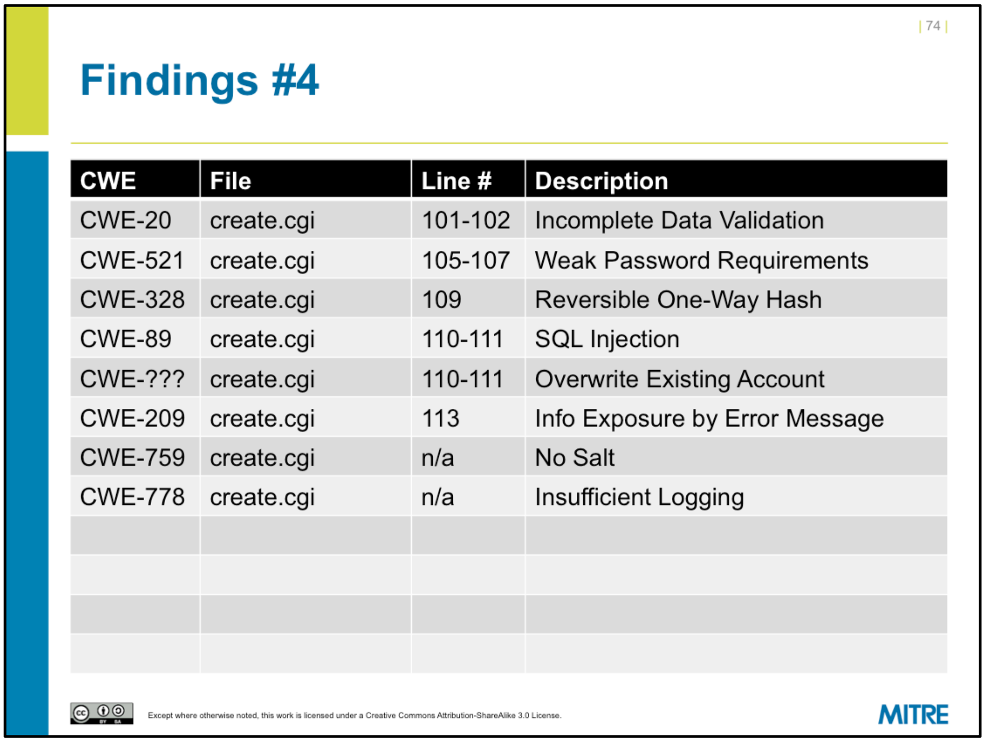
MITRE



Worksheet #4

CWE	File	Line #	Description

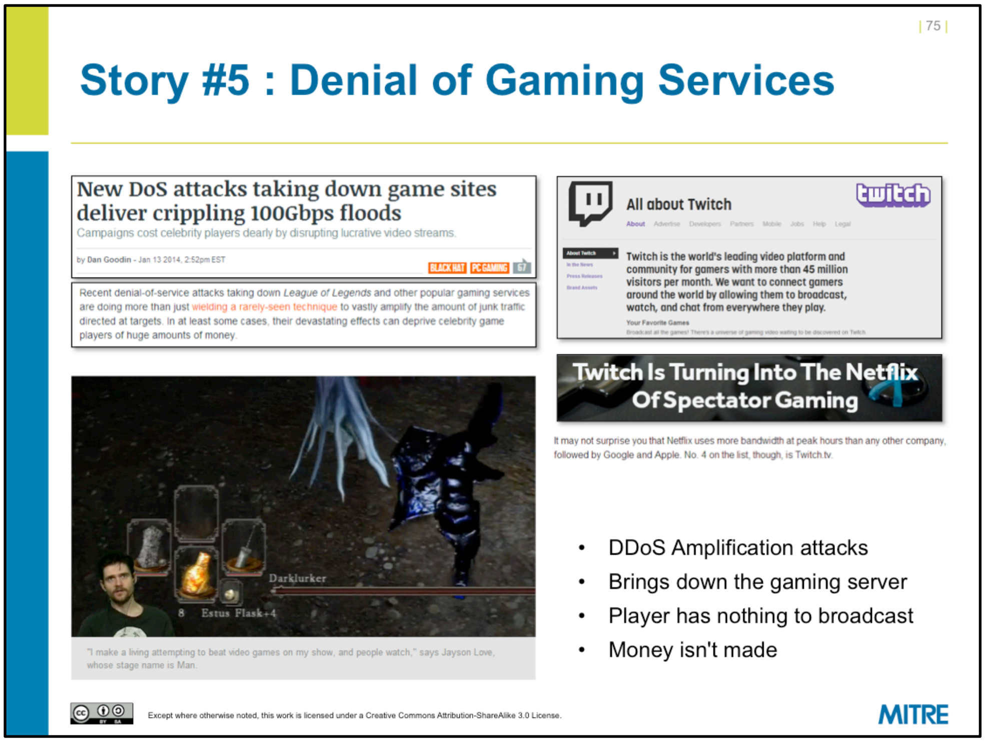
MITRE



Findings #4

CWE	File	Line #	Description
CWE-20	create.cgi	101-102	Incomplete Data Validation
CWE-521	create.cgi	105-107	Weak Password Requirements
CWE-328	create.cgi	109	Reversible One-Way Hash
CWE-89	create.cgi	110-111	SQL Injection
CWE-???	create.cgi	110-111	Overwrite Existing Account
CWE-209	create.cgi	113	Info Exposure by Error Message
CWE-759	create.cgi	n/a	No Salt
CWE-778	create.cgi	n/a	Insufficient Logging

MITRE



Story #5 : Denial of Gaming Services

New DoS attacks taking down game sites deliver crippling 100Gbps floods

Campaigns cost celebrity players dearly by disrupting lucrative video streams.

by Dan Goodin - Jan 13 2014, 2:52pm EST

BLACK HAT PC GAMING 67

Recent denial-of-service attacks taking down *League of Legends* and other popular gaming services are doing more than just wielding a rarely-seen technique to vastly amplify the amount of junk traffic directed at targets. In at least some cases, their devastating effects can deprive celebrity game players of huge amounts of money.

"I make a living attempting to beat video games on my show, and people watch," says Jayson Love, whose stage name is Man.

All about Twitch

Twitch is the world's leading video platform and community for gamers with more than 45 million visitors per month. We want to connect gamers around the world by allowing them to broadcast, watch, and chat from everywhere they play.

Your Favorite Games

Broadcast all the games! There's a universe of gaming video waiting to be discovered on Twitch.

Twitch Is Turning Into The Netflix Of Spectator Gaming

It may not surprise you that Netflix uses more bandwidth at peak hours than any other company, followed by Google and Apple. No. 4 on the list, though, is Twitch.tv.

- DDoS Amplification attacks
- Brings down the gaming server
- Player has nothing to broadcast
- Money isn't made

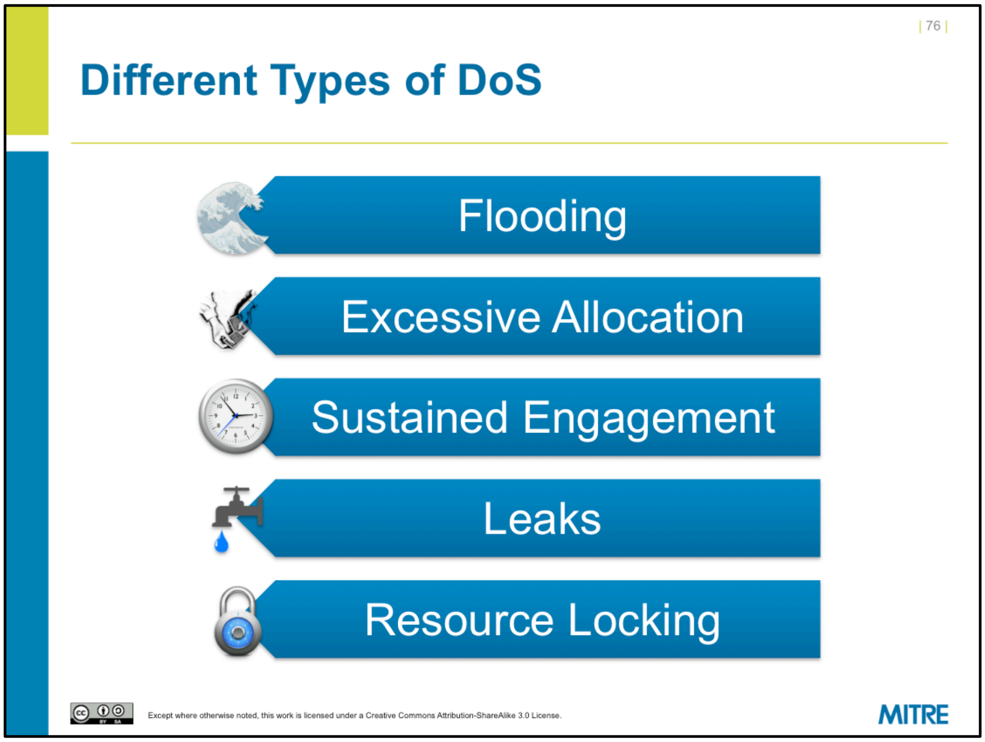
MITRE

http://arstechnica.com/security/2014/01/new-dos-attacks-taking-down-game-sites-deliver-crippling-100-gbps-floods/

http://www.twitch.tv/p/about

http://readwrite.com/2014/04/02/twitch-xbox-one-ps4#awesm=~oBIftGqykwW3wm

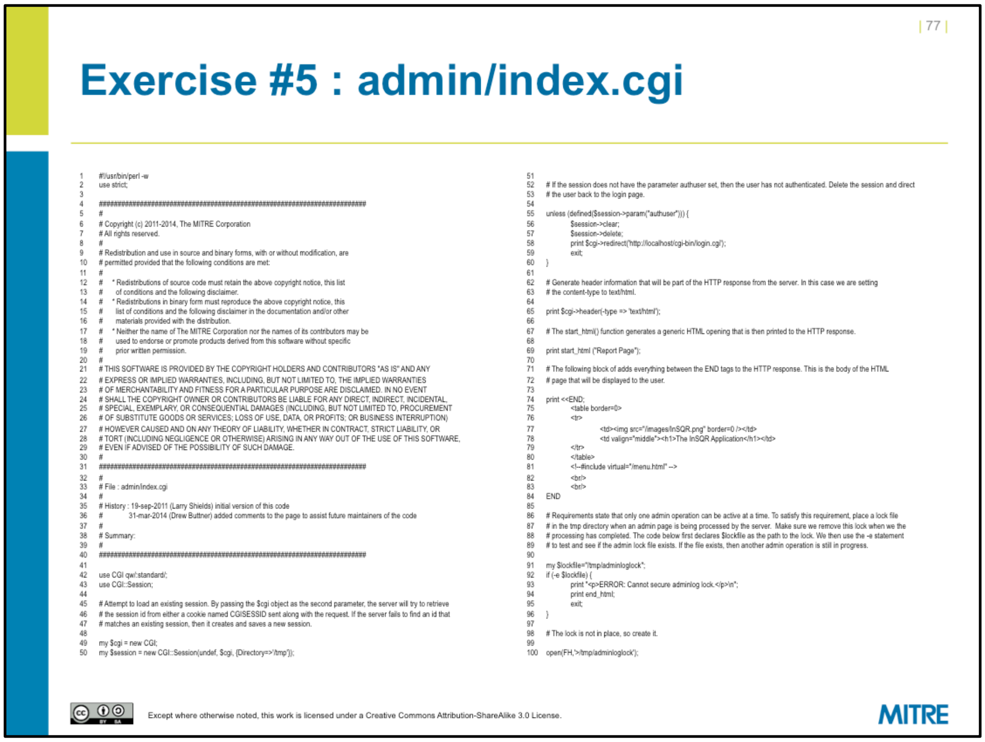
http://www.npr.org/blogs/alltechconsidered/2014/04/04/298775179/twitch-boosts-a-new-pro-category-video-game-player



Different Types of DoS

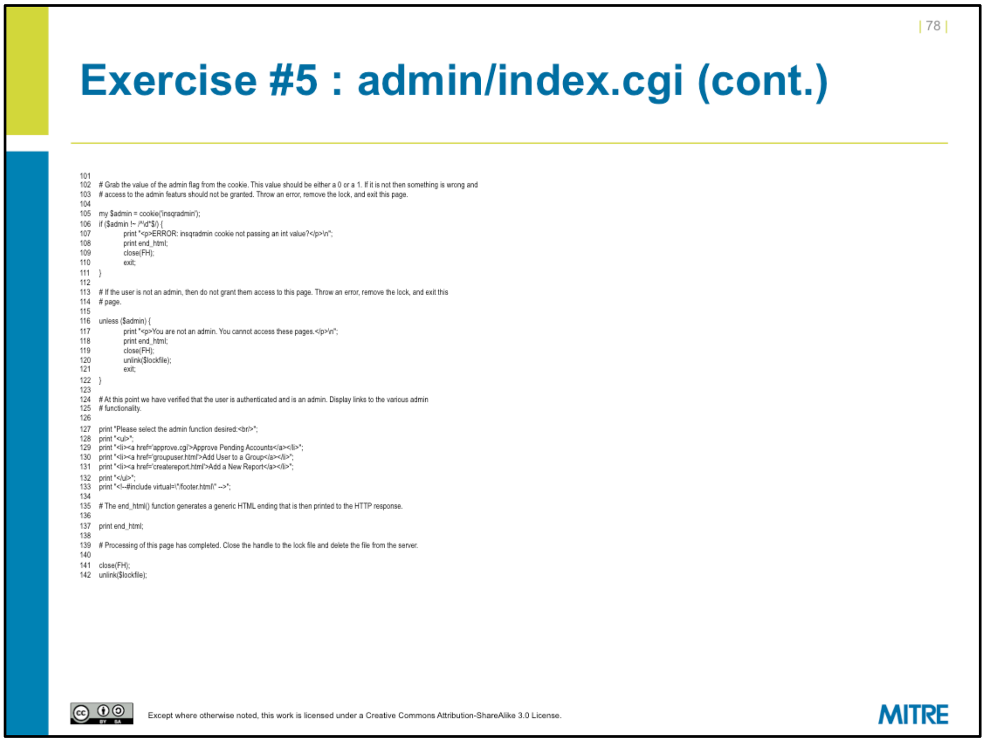
- Flooding
- Excessive Allocation
- Sustained Engagement
- Leaks
- Resource Locking

This slide is used to bridge the conversation from the previous story about DDoS to the code review exercise related to resource locking.



Exercise #5 : admin/index.cgi

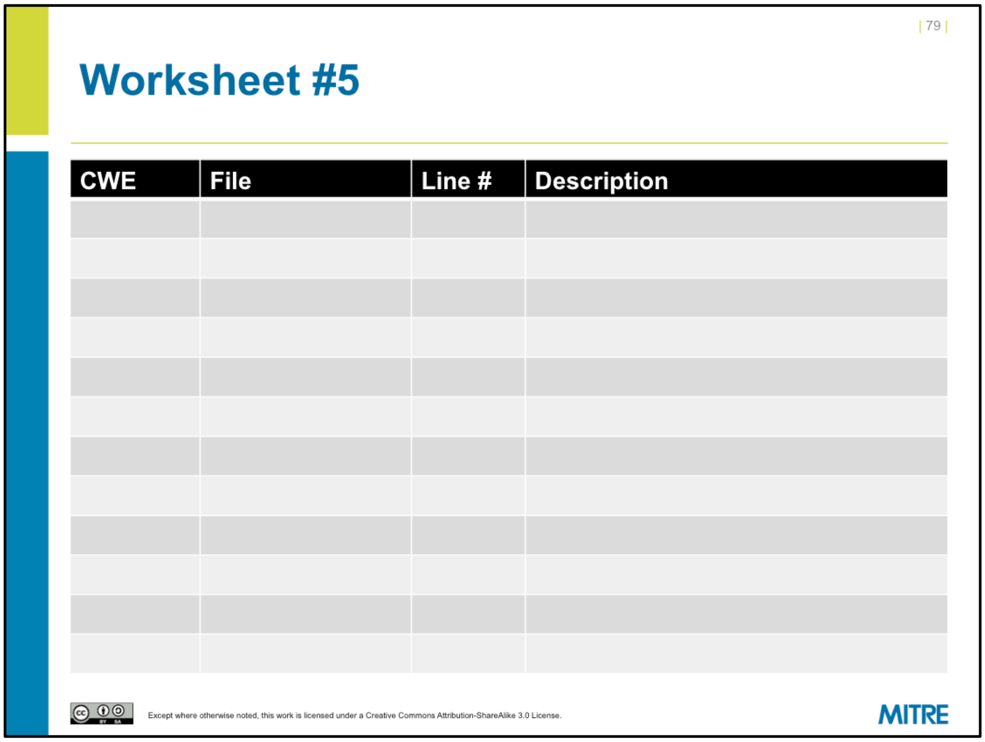
```
#!/usr/bin/perl -w
use strict;

##############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#    * Redistributions of source code must retain the above copyright notice, this list
#      of conditions and the following disclaimer.
#    * Redistributions in binary form must reproduce the above copyright notice, this
#      list of conditions and the following disclaimer in the documentation and/or other
#      materials provided with the distribution.
#    * Neither the name of The MITRE Corporation nor the names of its contributors may be
#      used to endorse or promote products derived from this software without specific
#      prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
##############################################################################
#
# File : admin/index.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#           31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary:
#
##############################################################################

use CGI qw/:standard/;
use CGI::Session;

# Attempt to load an existing session. By passing the $cgi object as the second parameter, the server will try to retrieve
# the session id from either a cookie named CGISESSID sent along with the request. If the server fails to find an id that
# matches an existing session, then it creates and saves a new session.

my $cgi = new CGI;
my $session = new CGI::Session(undef, $cgi, {Directory=>'/tmp'});

# If the session does not have the parameter authuser set, then the user has not authenticated. Delete the session and direct
# the user back to the login page.

unless (defined($session->param("authuser"))) {
        $session->clear;
        $session->delete;
        print $cgi->redirect('http://localhost/cgi-bin/login.cgi');
        exit;
}

# Generate header information that will be part of the HTTP response from the server. In this case we are setting
# the content-type to text/html.

print $cgi->header(-type => 'text/html');

# The start_html() function generates a generic HTML opening that is then printed to the HTTP response.

print start_html ("Report Page");

# The following block of adds everything between the END tags to the HTTP response. This is the body of the HTML
# page that will be displayed to the user.

print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
        <br/>
END

# Requirements state that only one admin operation can be active at a time. To satisfy this requirement, place a lock file
# in the tmp directory when an admin page is being processed by the server.  Make sure we remove this lock when we the
# processing has completed. The code below first declares $lockfile as the path to the lock. We then use the -e statement
# to test and see if the admin lock file exists. If the file exists, then another admin operation is still in progress.

my $lockfile="/tmp/adminloglock";
if (-e $lockfile) {
        print "<p>ERROR: Cannot secure adminlog lock.</p>\n";
        print end_html;
        exit;
}

# The lock is not in place, so create it.

open(FH,'>/tmp/adminloglock');
```



Exercise #5 : admin/index.cgi (cont.)

```

# Grab the value of the admin flag from the cookie. This value should be either a 0 or a 1. If it is not then something is wrong and
# access to the admin featurs should not be granted. Throw an error, remove the lock, and exit this page.

my $admin = cookie('insqradmin');
if ($admin !~ /^\d*$/) {
        print "<p>ERROR: insqradmin cookie not passing an int value?</p>\n";
        print end_html;
        close(FH);
        exit;
}

# If the user is not an admin, then do not grant them access to this page. Throw an error, remove the lock, and exit this
# page.

unless ($admin) {
        print "<p>You are not an admin. You cannot access these pages.</p>\n";
        print end_html;
        close(FH);
        unlink($lockfile);
        exit;
}

# At this point we have verified that the user is authenticated and is an admin. Display links to the various admin
# functionality.

print "Please select the admin function desired:<br/>";
print "<ul>";
print "<li><a href='approve.cgi'>Approve Pending Accounts</a></li>";
print "<li><a href='groupuser.html'>Add User to a Group</a></li>";
print "<li><a href='createreport.html'>Add a New Report</a></li>";
print "</ul>";
print "<!--#include virtual=\"/footer.html\" -->";

# The end_html() function generates a generic HTML ending that is then printed to the HTTP response.

print end_html;

# Processing of this page has completed. Close the handle to the lock file and delete the file from the server.

close(FH);
unlink($lockfile);
```

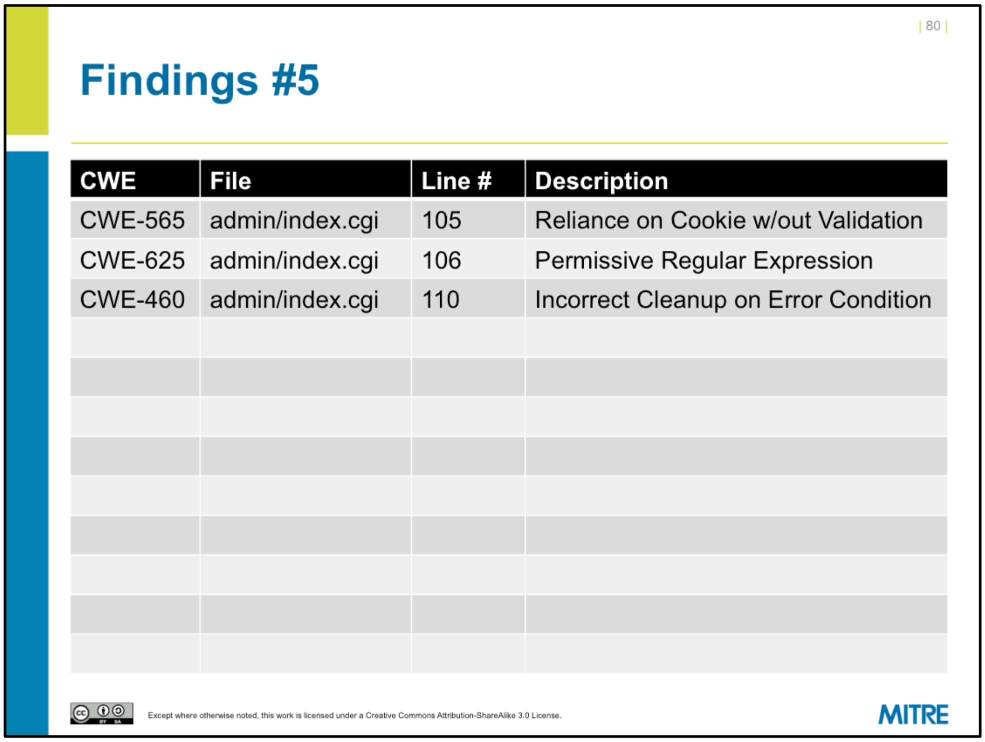
MITRE



Worksheet #5

CWE	File	Line #	Description

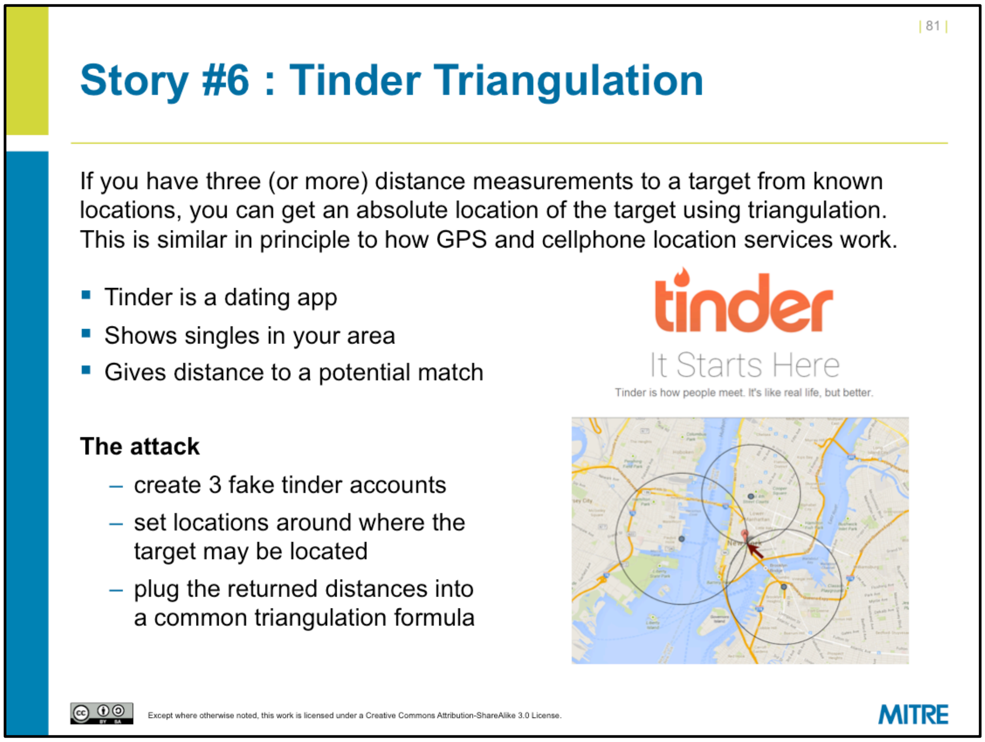
MITRE



Findings #5

CWE	File	Line #	Description
CWE-565	admin/index.cgi	105	Reliance on Cookie w/out Validation
CWE-625	admin/index.cgi	106	Permissive Regular Expression
CWE-460	admin/index.cgi	110	Incorrect Cleanup on Error Condition

MITRE



Story #6 : Tinder Triangulation

If you have three (or more) distance measurements to a target from known locations, you can get an absolute location of the target using triangulation. This is similar in principle to how GPS and cellphone location services work.

- Tinder is a dating app
- Shows singles in your area
- Gives distance to a potential match

tinder

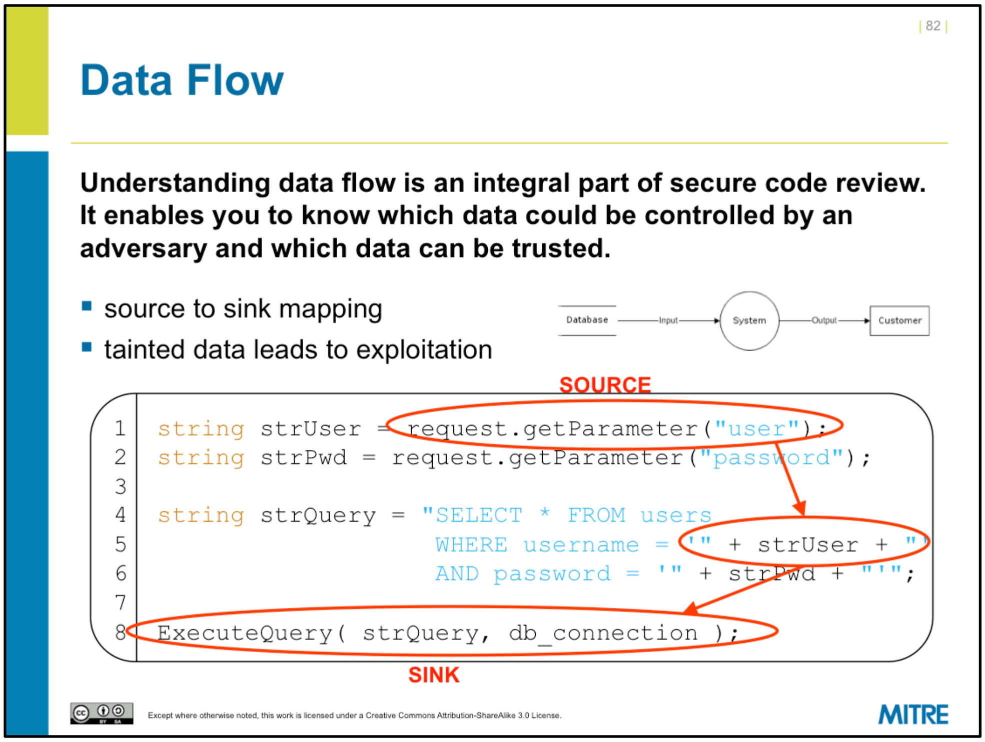
It Starts Here

Tinder is how people meet. It's like real life, but better.

The attack

- create 3 fake tinder accounts
- set locations around where the target may be located
- plug the returned distances into a common triangulation formula

http://blog.includesecurity.com/2014/02/how-i-was-able-to-track-location-of-any.html



Data Flow

Understanding data flow is an integral part of secure code review. It enables you to know which data could be controlled by an adversary and which data can be trusted.

- source to sink mapping
- tainted data leads to exploitation

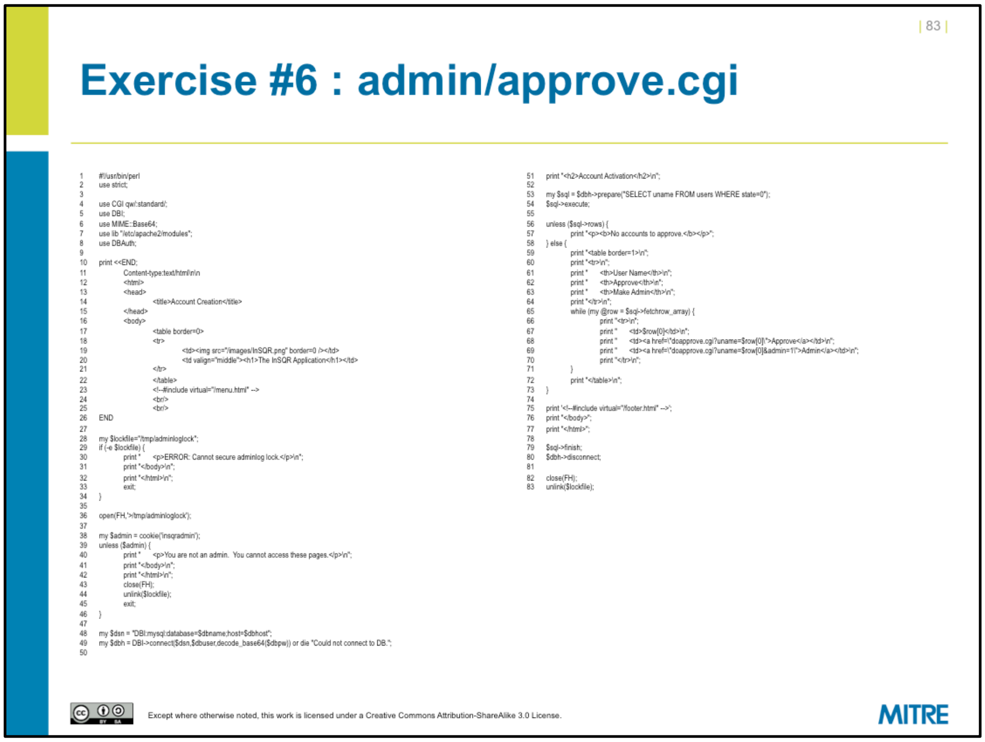
Database
Input
System
Output
Customer

SOURCE

```
string strUser = request.getParameter("user");
string strPwd = request.getParameter("password");

string strQuery = "SELECT * FROM users
                   WHERE username = '" + strUser + "'
                   AND password = '" + strPwd + "'";

ExecuteQuery( strQuery, db_connection );
```

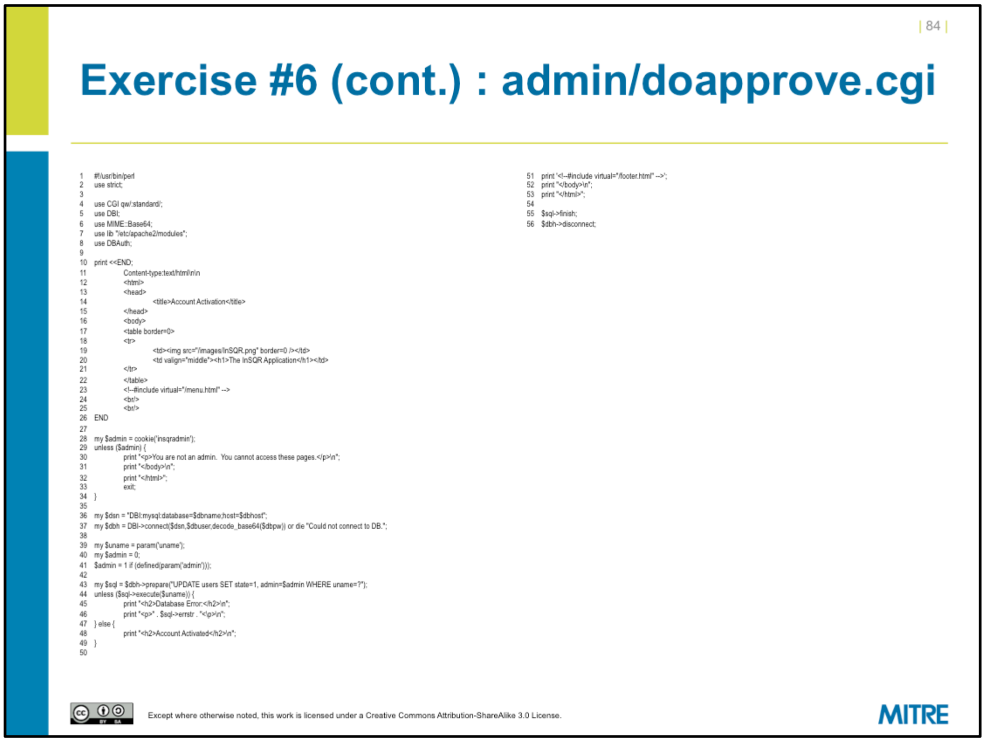
SINK

MITRE



Exercise #6 : admin/approve.cgi

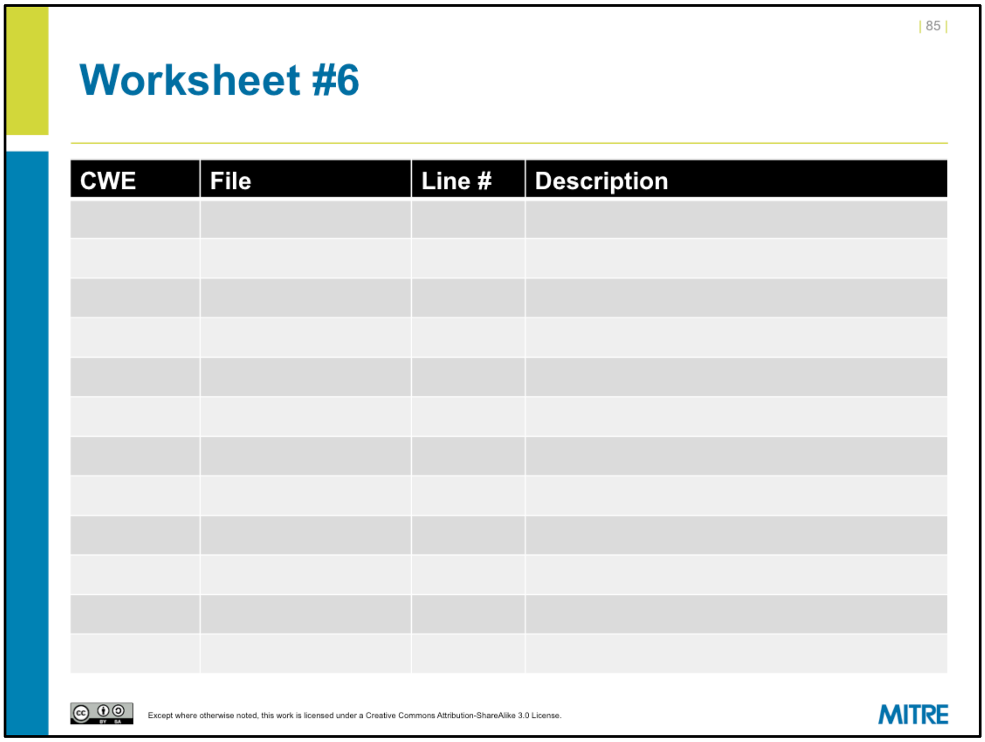
```
#!/usr/bin/perl
use strict;

use CGI qw/:standard/;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;

print <<END;
        Content-type:text/html\n\n
        <html>
        <head>
                <title>Account Creation</title>
        </head>
        <body>
                <table border=0>
                <tr>
                        <td><img src="/images/InSQR.png" border=0 /></td>
                        <td valign="middle"><h1>The InSQR Application</h1></td>
                </tr>
                </table>
                <!--#include virtual="/menu.html" -->
                <br/>
                <br/>
END

my $lockfile="/tmp/adminloglock";
if (-e $lockfile) {
        print "     <p>ERROR: Cannot secure adminlog lock.</p>\n";
        print "</body>\n";
        print "</html>\n";
        exit;
}

open(FH,'>/tmp/adminloglock');

my $admin = cookie('insqradmin');
unless ($admin) {
        print "     <p>You are not an admin.  You cannot access these pages.</p>\n";
        print "</body>\n";
        print "</html>\n";
        close(FH);
        unlink($lockfile);
        exit;
}

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

print "<h2>Account Activation</h2>\n";

my $sql = $dbh->prepare("SELECT uname FROM users WHERE state=0");
$sql->execute;

unless ($sql->rows) {
        print "<p><b>No accounts to approve.</b></p>";
} else {
        print "<table border=1>\n";
        print "<tr>\n";
        print "     <th>User Name</th>\n";
        print "     <th>Approve</th>\n";
        print "     <th>Make Admin</th>\n";
        print "</tr>\n";
        while (my @row = $sql->fetchrow_array) {
                print "<tr>\n";
                print "     <td>$row[0]</td>\n";
                print "     <td><a href=\"doapprove.cgi?uname=$row[0]\">Approve</a></td>\n";
                print "     <td><a href=\"doapprove.cgi?uname=$row[0]&admin=1\">Admin</a></td>\n";
                print "</tr>\n";
        }
        print "</table>\n";
}

print '<!--#include virtual="/footer.html" -->';
print "</body>";
print "</html>";

$sql->finish;
$dbh->disconnect;

close(FH);
unlink($lockfile);
```



Exercise #6 (cont.) : admin/doapprove.cgi

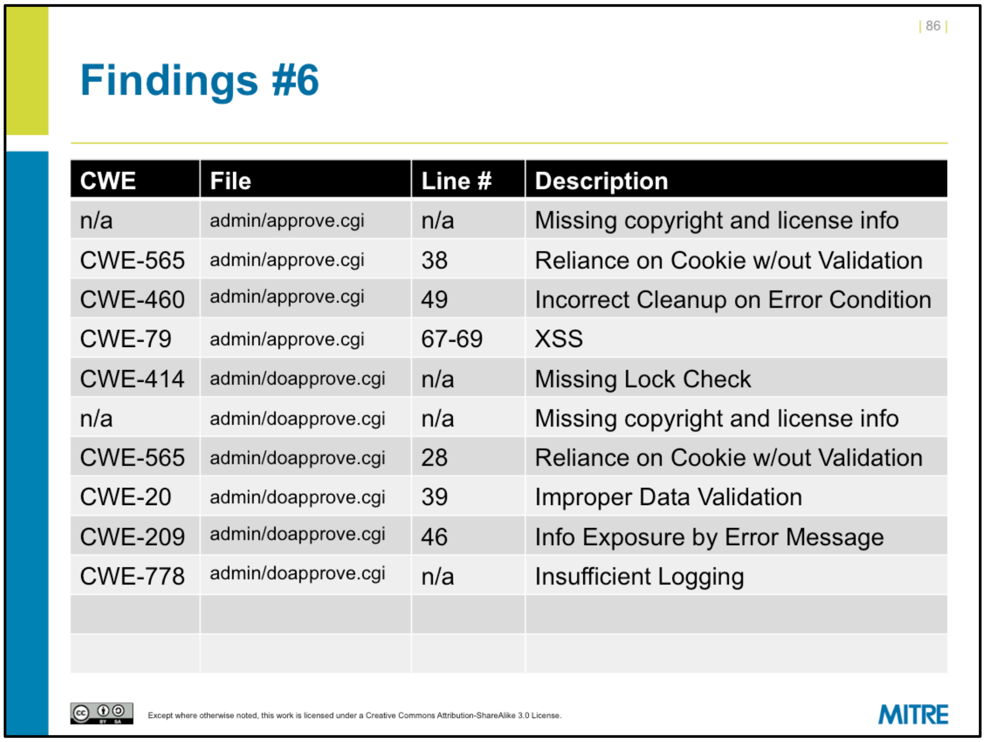
```
#!/usr/bin/perl
use strict;

use CGI qw/:standard/;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;

print <<END;
        Content-type:text/html\n\n
        <html>
        <head>
                <title>Account Activation</title>
        </head>
        <body>
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
        <br/>
END

my $admin = cookie('insqradmin');
unless ($admin) {
        print "<p>You are not an admin.  You cannot access these pages.</p>\n";
        print "</body>\n";
        print "</html>";
        exit;
}

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

my $uname = param('uname');
my $admin = 0;
$admin = 1 if (defined(param('admin')));

my $sql = $dbh->prepare("UPDATE users SET state=1, admin=$admin WHERE uname=?");
unless ($sql->execute($uname)) {
        print "<h2>Database Error:</h2>\n";
        print "<p>" . $sql->errstr . "<\p>\n";
} else {
        print "<h2>Account Activated</h2>\n";
}

print '<!--#include virtual="/footer.html" -->';
print "</body>\n";
print "</html>";

$sql->finish;
$dbh->disconnect;
```

MITRE



Worksheet #6

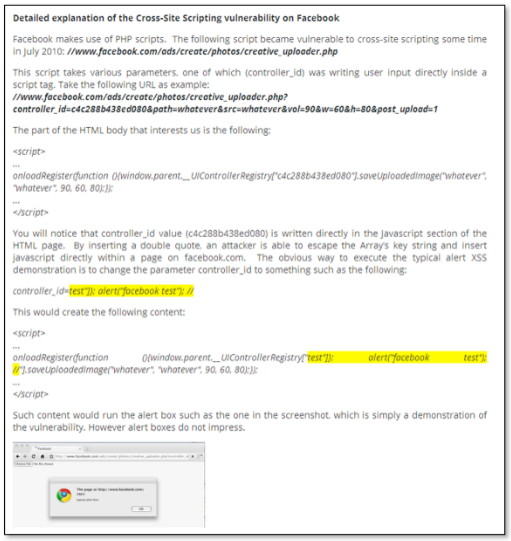
CWE	File	Line #	Description



Findings #6

CWE	File	Line #	Description
n/a	admin/approve.cgi	n/a	Missing copyright and license info
CWE-565	admin/approve.cgi	38	Reliance on Cookie w/out Validation
CWE-460	admin/approve.cgi	49	Incorrect Cleanup on Error Condition
CWE-79	admin/approve.cgi	67-69	XSS
CWE-414	admin/doapprove.cgi	n/a	Missing Lock Check
n/a	admin/doapprove.cgi	n/a	Missing copyright and license info
CWE-565	admin/doapprove.cgi	28	Reliance on Cookie w/out Validation
CWE-20	admin/doapprove.cgi	39	Improper Data Validation
CWE-209	admin/doapprove.cgi	46	Info Exposure by Error Message
CWE-778	admin/doapprove.cgi	n/a	Insufficient Logging

Story #7 : Facebook and XSS



Detailed explanation of the Cross-Site Scripting vulnerability on Facebook

Facebook makes use of PHP scripts. The following script became vulnerable to cross-site scripting some time in July 2010: ***//www.facebook.com/ads/create/photos/creative_uploader.php***

This script takes various parameters, one of which (controller_id) was writing user input directly inside a script tag. Take the following URL as example:
//www.facebook.com/ads/create/photos/creative_uploader.php?controller_id=c4c288b438ed080&path=whatever&src=whatever&vol=90&w=60&h=80&post_upload=1

The part of the HTML body that interests us is the following:

```
<script>
...
onloadRegister(function (){window.parent.__UIControllerRegistry["c4c288b438ed080"].saveUploadedImage("whatever", "whatever", 90, 60, 80);});
...
</script>
```

You will notice that controller_id value (c4c288b438ed080) is written directly in the Javascript section of the HTML page. By inserting a double quote, an attacker is able to escape the Array's key string and insert javascript directly within a page on facebook.com. The obvious way to execute the typical alert XSS demonstration is to change the parameter controller_id to something such as the following:

```
controller_id=test"]); alert("facebook test"); //
```

This would create the following content:

```
<script>
...
onloadRegister(function (){window.parent.__UIControllerRegistry["test"]); alert("facebook test"); //"].saveUploadedImage("whatever", "whatever", 90, 60, 80);});
...
</script>
```

Such content would run the alert box such as the one in the screenshot, which is simply a demonstration of the vulnerability. However alert boxes do not impress.

facebook

- XSS not limited to text fields
- No <script> tag needed
- Note that this code supports ads!

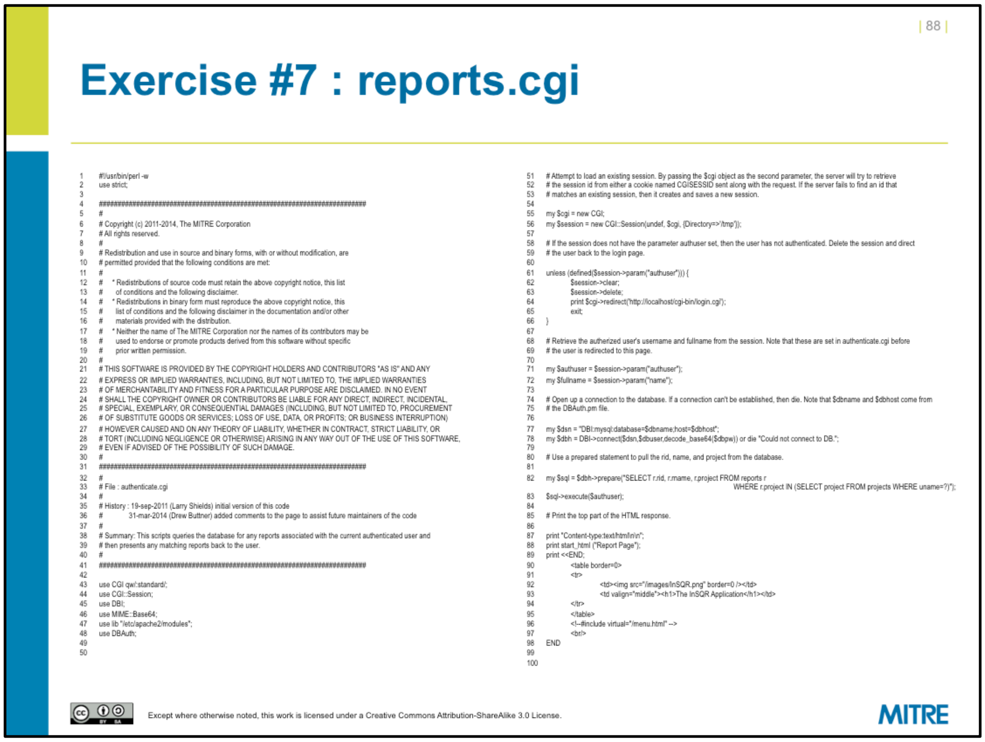
https://www.acunetix.com/websitesecurity/xss-facebook/

MITRE

Code to support ads.

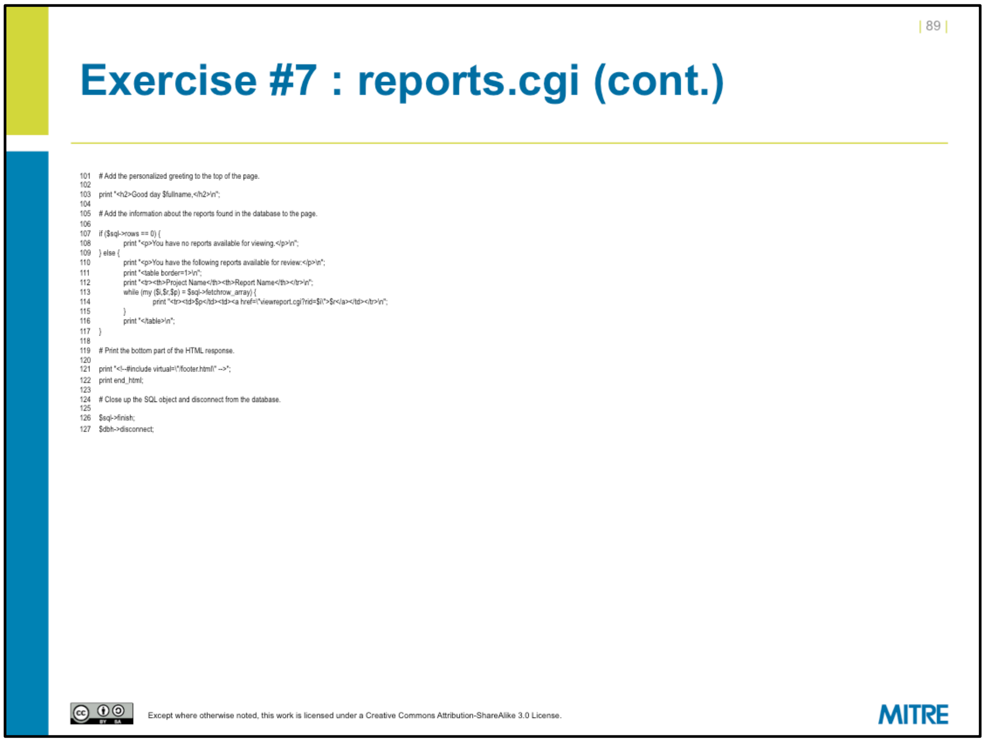
controller id was supplied by the user and was then used directly inside a script tag.

https://www.acunetix.com/websitesecurity/xss-facebook/



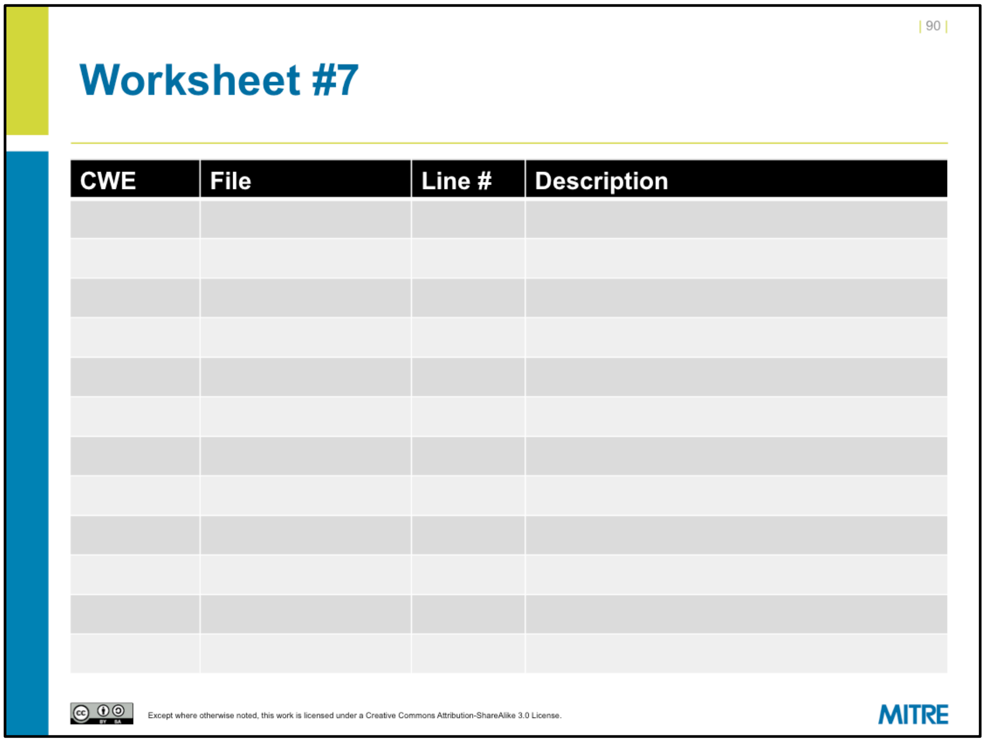
Exercise #7 : reports.cgi

```
#!/usr/bin/perl -w
use strict;

##############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice, this list
#       of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright notice, this
#       list of conditions and the following disclaimer in the documentation and/or other
#       materials provided with the distribution.
#     * Neither the name of The MITRE Corporation nor the names of its contributors may be
#       used to endorse or promote products derived from this software without specific
#       prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
##############################################################################
#
# File : authenticate.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#           31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary: This scripts queries the database for any reports associated with the current authenticated user and
# then presents any matching reports back to the user.
#
##############################################################################

use CGI qw/:standard/;
use CGI::Session;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;


# Attempt to load an existing session. By passing the $cgi object as the second parameter, the server will try to retrieve
# the session id from either a cookie named CGISESSID sent along with the request. If the server fails to find an id that
# matches an existing session, then it creates and saves a new session.

my $cgi = new CGI;
my $session = new CGI::Session(undef, $cgi, {Directory=>'/tmp'});

# If the session does not have the parameter authuser set, then the user has not authenticated. Delete the session and direct
# the user back to the login page.

unless (defined($session->param("authuser"))) {
        $session->clear;
        $session->delete;
        print $cgi->redirect('http://localhost/cgi-bin/login.cgi');
        exit;
}

# Retrieve the autherized user's username and fullname from the session. Note that these are set in authenticate.cgi before
# the user is redirected to this page.

my $authuser = $session->param("authuser");
my $fullname = $session->param("name");

# Open up a connection to the database. If a connection can't be established, then die. Note that $dbname and $dbhost come from
# the DBAuth.pm file.

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

# Use a prepared statement to pull the rid, name, and project from the database.

my $sql = $dbh->prepare("SELECT r.rid, r.rname, r.project FROM reports r
                                                            WHERE r.project IN (SELECT project FROM projects WHERE uname=?)");
$sql->execute($authuser);

# Print the top part of the HTML response.

print "Content-type:text/html\n\n";
print start_html ("Report Page");
print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
END


```



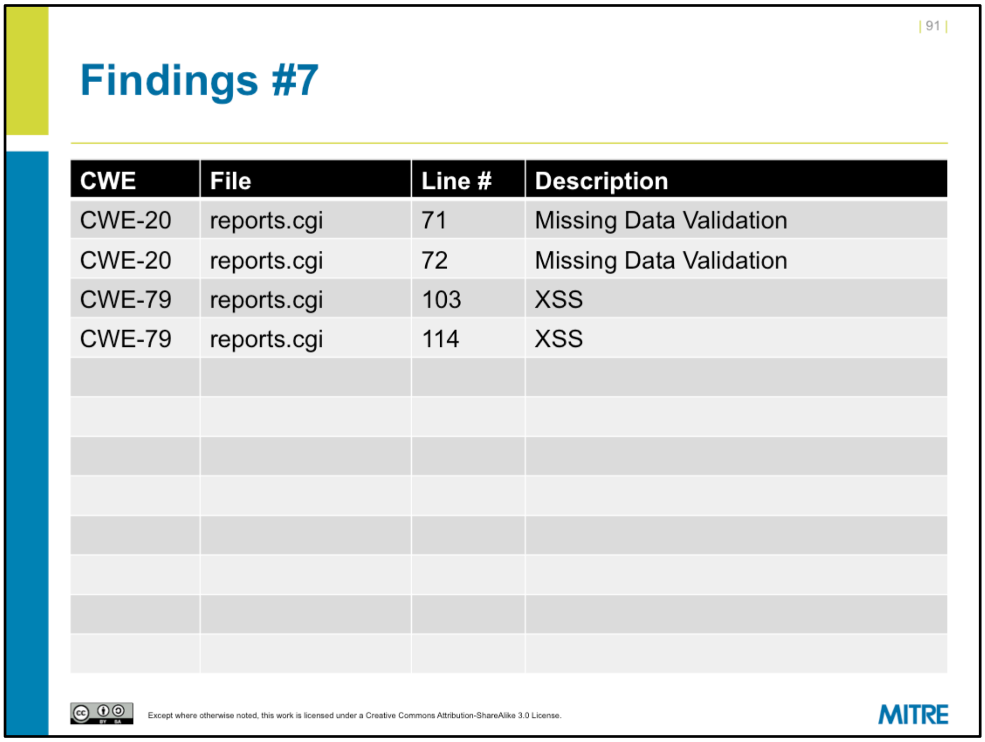
Exercise #7 : reports.cgi (cont.)

```
# Add the personalized greeting to the top of the page.

print "<h2>Good day $fullname,</h2>\n";

# Add the information about the reports found in the database to the page.

if ($sql->rows == 0) {
        print "<p>You have no reports available for viewing.</p>\n";
} else {
        print "<p>You have the following reports available for review:</p>\n";
        print "<table border=1>\n";
        print "<tr><th>Project Name</th><th>Report Name</th></tr>\n";
        while (my ($i,$r,$p) = $sql->fetchrow_array) {
                print "<tr><td>$p</td><td><a href=\"viewreport.cgi?rid=$i\">$r</a></td></tr>\n";
        }
        print "</table>\n";
}

# Print the bottom part of the HTML response.

print "<!--#include virtual=\"/footer.html\" -->";
print end_html;

# Close up the SQL object and disconnect from the database.

$sql->finish;
$dbh->disconnect;
```



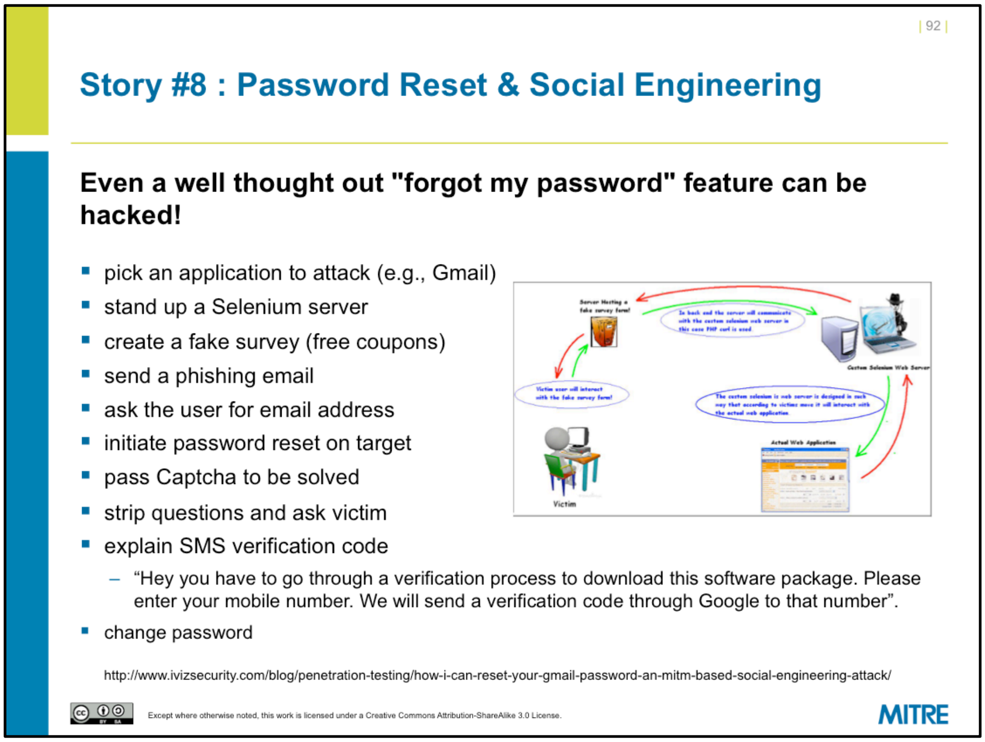
Worksheet #7

CWE	File	Line #	Description



Findings #7

CWE	File	Line #	Description
CWE-20	reports.cgi	71	Missing Data Validation
CWE-20	reports.cgi	72	Missing Data Validation
CWE-79	reports.cgi	103	XSS
CWE-79	reports.cgi	114	XSS



Story #8 : Password Reset & Social Engineering

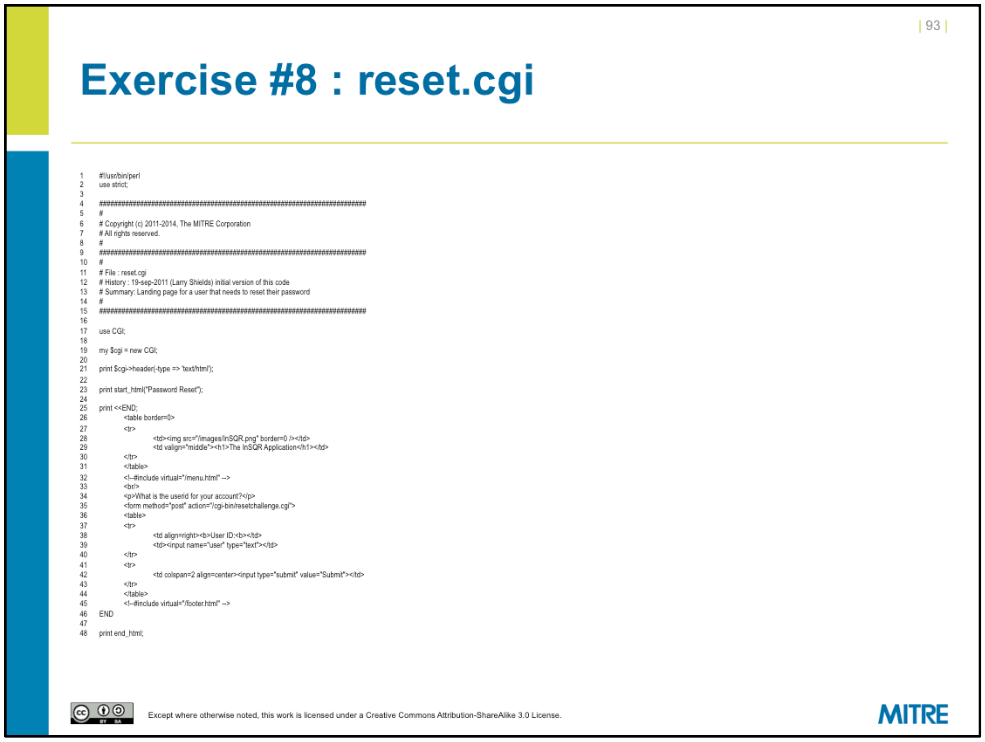
Even a well thought out "forgot my password" feature can be hacked!

- pick an application to attack (e.g., Gmail)
- stand up a Selenium server
- create a fake survey (free coupons)
- send a phishing email
- ask the user for email address
- initiate password reset on target
- pass Captcha to be solved
- strip questions and ask victim
- explain SMS verification code
  - "Hey you have to go through a verification process to download this software package. Please enter your mobile number. We will send a verification code through Google to that number".
- change password

Server Hosting a fake survey form!
In back end the server will communicate with the custom selenium web server in this case PHP curl is used.
Custom Selenium Web Server
Victim user will interact with the fake survey form!
The custom selenium is web server is designed in such way that according to victims move it will interact with the actual web application.
Actual Web Application
Victim

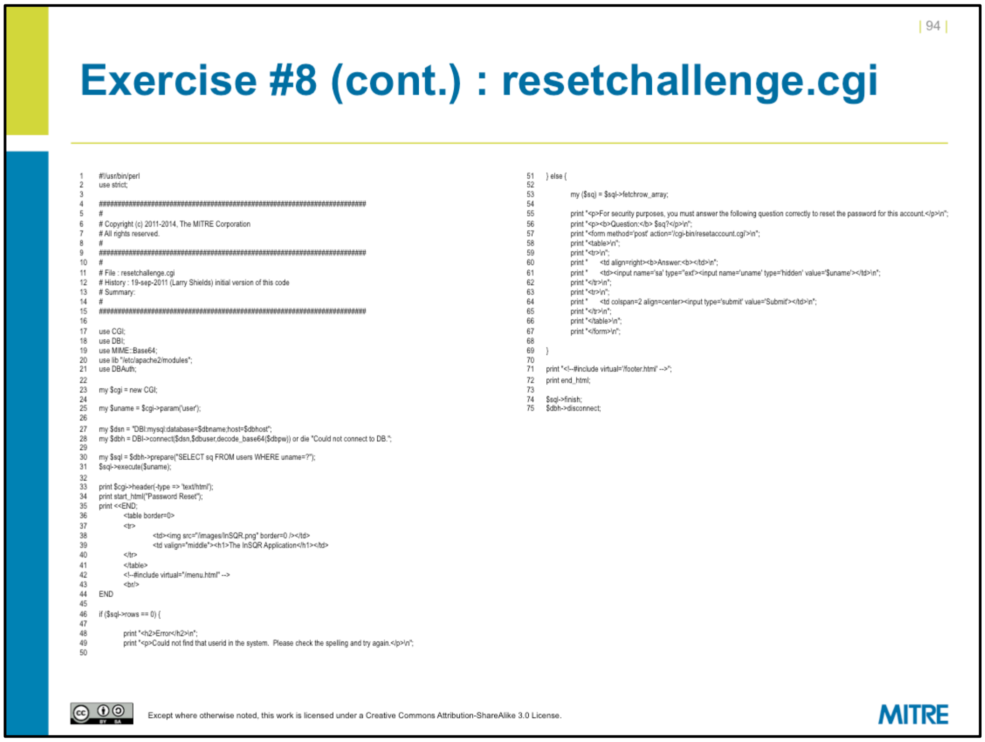
http://www.ivizsecurity.com/blog/penetration-testing/how-i-can-reset-your-gmail-password-an-mitm-based-social-engineering-attack/

Selenium is a software testing framework for web applications. Selenium can automate browser locally or remotely. http://seleniumhq.org/.)



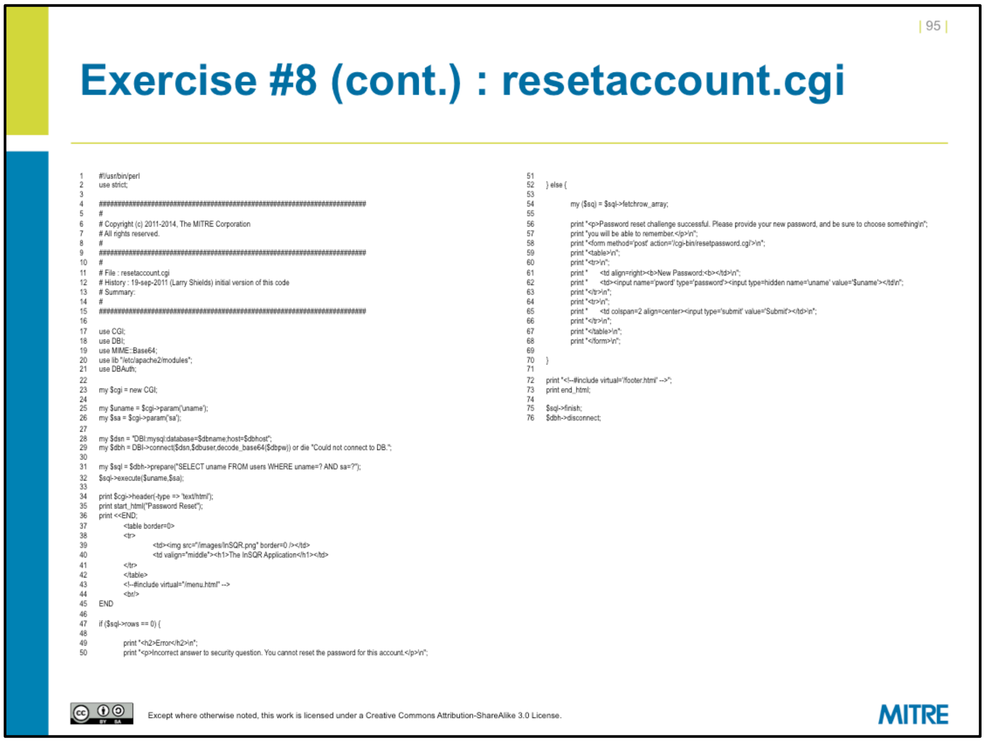
Exercise #8 : reset.cgi

```
#!/usr/bin/perl
use strict;

############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
############################################################################
#
# File : reset.cgi
# History : 19-sep-2011 (Larry Shields) initial version of this code
# Summary: Landing page for a user that needs to reset their password
#
############################################################################

use CGI;

my $cgi = new CGI;

print $cgi->header(-type => 'text/html');

print start_html("Password Reset");

print <<END;
	<table border=0>
	<tr>
		<td><img src="/images/InSQR.png" border=0 /></td>
		<td valign="middle"><h1>The InSQR Application</h1></td>
	</tr>
	</table>
	<!--#include virtual="/menu.html" -->
	<br/>
	<p>What is the userid for your account?</p>
	<form method="post" action="/cgi-bin/resetchallenge.cgi">
	<table>
	<tr>
		<td align=right><b>User ID:<b></td>
		<td><input name="user" type="text"></td>
	</tr>
	<tr>
		<td colspan=2 align=center><input type="submit" value="Submit"></td>
	</tr>
	</table>
	<!--#include virtual="/footer.html" -->
END

print end_html;
```



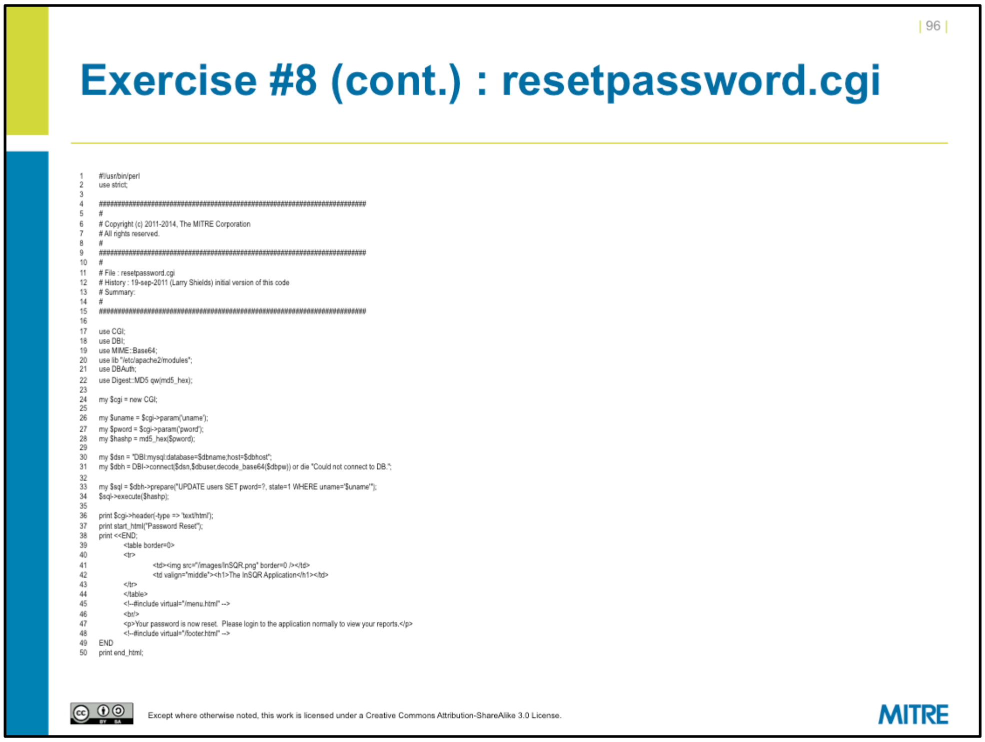
Exercise #8 (cont.) : resetchallenge.cgi

```
#!/usr/bin/perl
use strict;

############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
############################################################################
#
# File : resetchallenge.cgi
# History : 19-sep-2011 (Larry Shields) initial version of this code
# Summary:
#
############################################################################

use CGI;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;

my $cgi = new CGI;

my $uname = $cgi->param('user');

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

my $sql = $dbh->prepare("SELECT sq FROM users WHERE uname=?");
$sql->execute($uname);

print $cgi->header(-type => 'text/html');
print start_html("Password Reset");
print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
END

if ($sql->rows == 0) {

        print "<h2>Error</h2>\n";
        print "<p>Could not find that userid in the system.  Please check the spelling and try again.</p>\n";

} else {

        my ($sq) = $sql->fetchrow_array;

        print "<p>For security purposes, you must answer the following question correctly to reset the password for this account.</p>\n";
        print "<p><b>Question:</b> $sq?</p>\n";
        print "<form method='post' action='/cgi-bin/resetaccount.cgi'>\n";
        print "<table>\n";
        print "<tr>\n";
        print "     <td align=right><b>Answer:<b></td>\n";
        print "     <td><input name='sa' type=''ext'><input name='uname' type='hidden' value='$uname'></td>\n";
        print "</tr>\n";
        print "<tr>\n";
        print "     <td colspan=2 align=center><input type='submit' value='Submit'></td>\n";
        print "</tr>\n";
        print "</table>\n";
        print "</form>\n";

}

print "<!--#include virtual='/footer.html' -->";
print end_html;

$sql->finish;
$dbh->disconnect;
```



Exercise #8 (cont.) : resetaccount.cgi

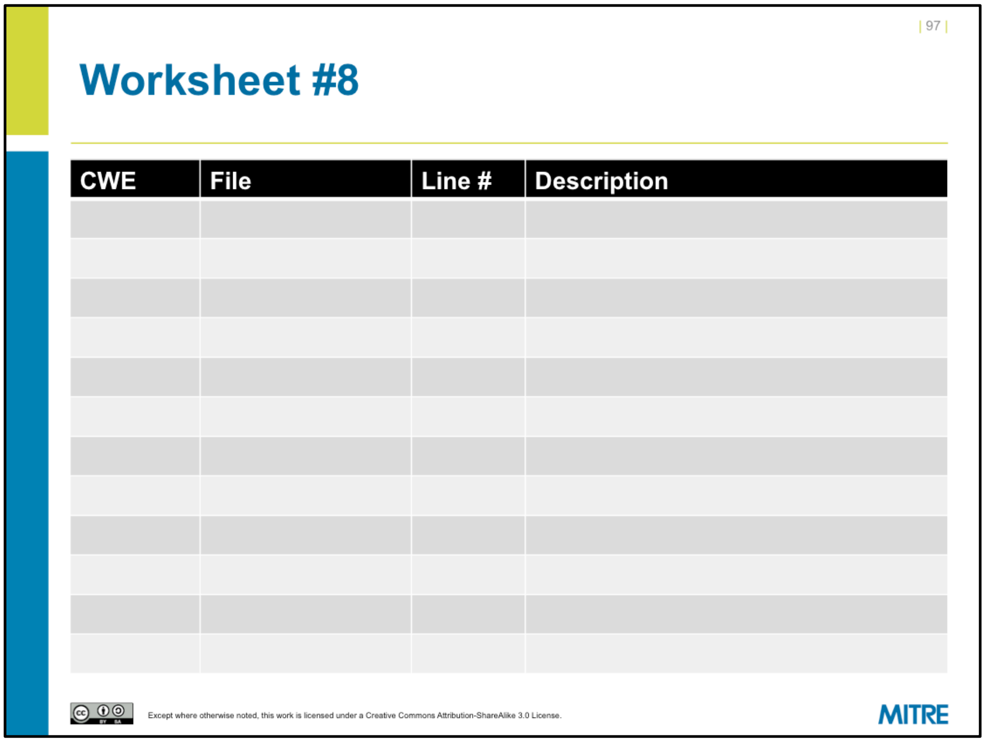
```
#!/usr/bin/perl
use strict;

###########################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
###########################################################################
#
# File : resetaccount.cgi
# History : 19-sep-2011 (Larry Shields) initial version of this code
# Summary:
#
###########################################################################

use CGI;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;

my $cgi = new CGI;

my $uname = $cgi->param('uname');
my $sa = $cgi->param('sa');

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

my $sql = $dbh->prepare("SELECT uname FROM users WHERE uname=? AND sa=?");
$sql->execute($uname,$sa);

print $cgi->header(-type => 'text/html');
print start_html("Password Reset");
print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
END

if ($sql->rows == 0) {

        print "<h2>Error</h2>\n";
        print "<p>Incorrect answer to security question. You cannot reset the password for this account.</p>\n";

} else {

        my ($sq) = $sql->fetchrow_array;

        print "<p>Password reset challenge successful. Please provide your new password, and be sure to choose something\n";
        print "you will be able to remember.</p>\n";
        print "<form method='post' action='/cgi-bin/resetpassword.cgi'>\n";
        print "<table>\n";
        print "<tr>\n";
        print "     <td align=right><b>New Password:<b></td>\n";
        print "     <td><input name='pword' type='password'><input type=hidden name='uname' value='$uname'></td\n";
        print "</tr>\n";
        print "<tr>\n";
        print "     <td colspan=2 align=center><input type='submit' value='Submit'></td>\n";
        print "</tr>\n";
        print "</table>\n";
        print "</form>\n";

}

print "<!--#include virtual='/footer.html' -->";
print end_html;

$sql->finish;
$dbh->disconnect;
```



Exercise #8 (cont.) : resetpassword.cgi

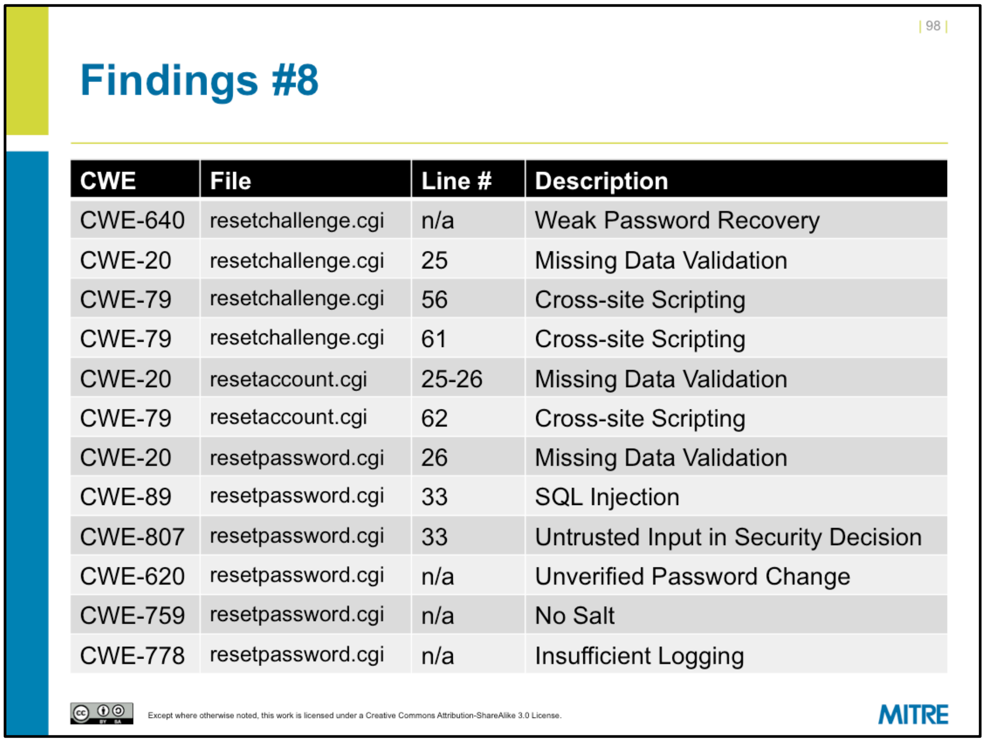
```
#!/usr/bin/perl
use strict;

################################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
################################################################################
#
# File : resetpassword.cgi
# History : 19-sep-2011 (Larry Shields) initial version of this code
# Summary:
#
################################################################################

use CGI;
use DBI;
use MIME::Base64;
use lib "/etc/apache2/modules";
use DBAuth;
use Digest::MD5 qw(md5_hex);

my $cgi = new CGI;

my $uname = $cgi->param('uname');
my $pword = $cgi->param('pword');
my $hashp = md5_hex($pword);

my $dsn = "DBI:mysql:database=$dbname;host=$dbhost";
my $dbh = DBI->connect($dsn,$dbuser,decode_base64($dbpw)) or die "Could not connect to DB.";

my $sql = $dbh->prepare("UPDATE users SET pword=?, state=1 WHERE uname='$uname'");
$sql->execute($hashp);

print $cgi->header(-type => 'text/html');
print start_html("Password Reset");
print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
        <p>Your password is now reset.  Please login to the application normally to view your reports.</p>
        <!--#include virtual="/footer.html" -->
END
print end_html;
```

MITRE



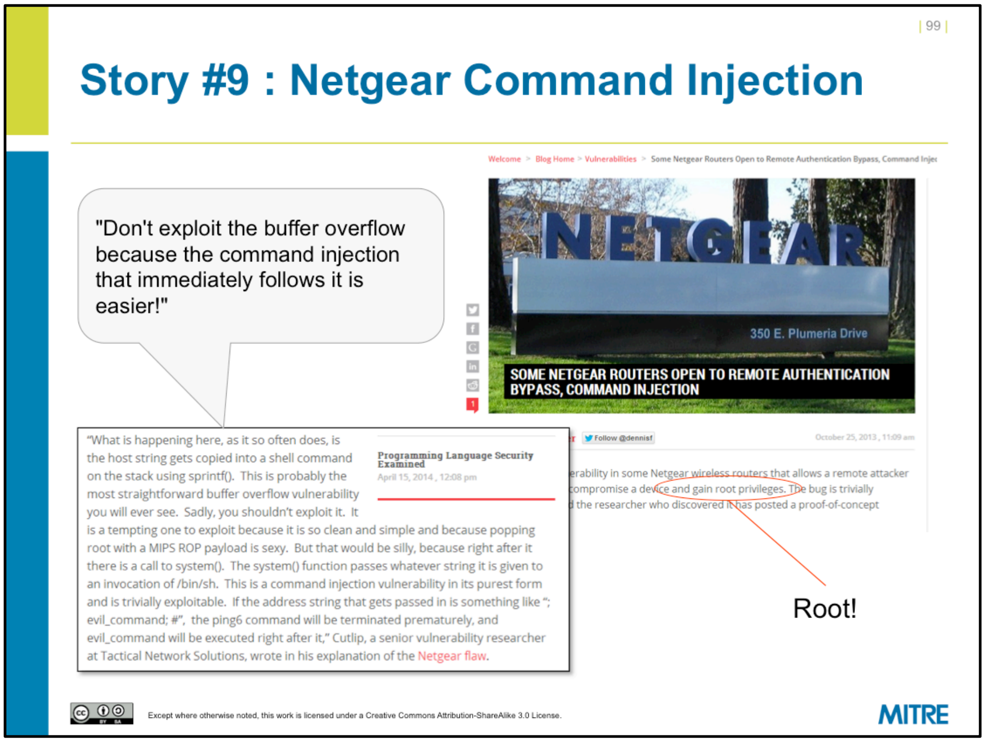
Worksheet #8

CWE	File	Line #	Description



Findings #8

CWE	File	Line #	Description
CWE-640	resetchallenge.cgi	n/a	Weak Password Recovery
CWE-20	resetchallenge.cgi	25	Missing Data Validation
CWE-79	resetchallenge.cgi	56	Cross-site Scripting
CWE-79	resetchallenge.cgi	61	Cross-site Scripting
CWE-20	resetaccount.cgi	25-26	Missing Data Validation
CWE-79	resetaccount.cgi	62	Cross-site Scripting
CWE-20	resetpassword.cgi	26	Missing Data Validation
CWE-89	resetpassword.cgi	33	SQL Injection
CWE-807	resetpassword.cgi	33	Untrusted Input in Security Decision
CWE-620	resetpassword.cgi	n/a	Unverified Password Change
CWE-759	resetpassword.cgi	n/a	No Salt
CWE-778	resetpassword.cgi	n/a	Insufficient Logging



Story #9 : Netgear Command Injection

"Don't exploit the buffer overflow because the command injection that immediately follows it is easier!"

Welcome > Blog Home > Vulnerabilities > Some Netgear Routers Open to Remote Authentication Bypass, Command Injec
NETGEAR
350 E. Plumeria Drive
1
Follow @dennisf

SOME NETGEAR ROUTERS OPEN TO REMOTE AUTHENTICATION BYPASS, COMMAND INJECTION

October 25, 2013 , 11:09 am

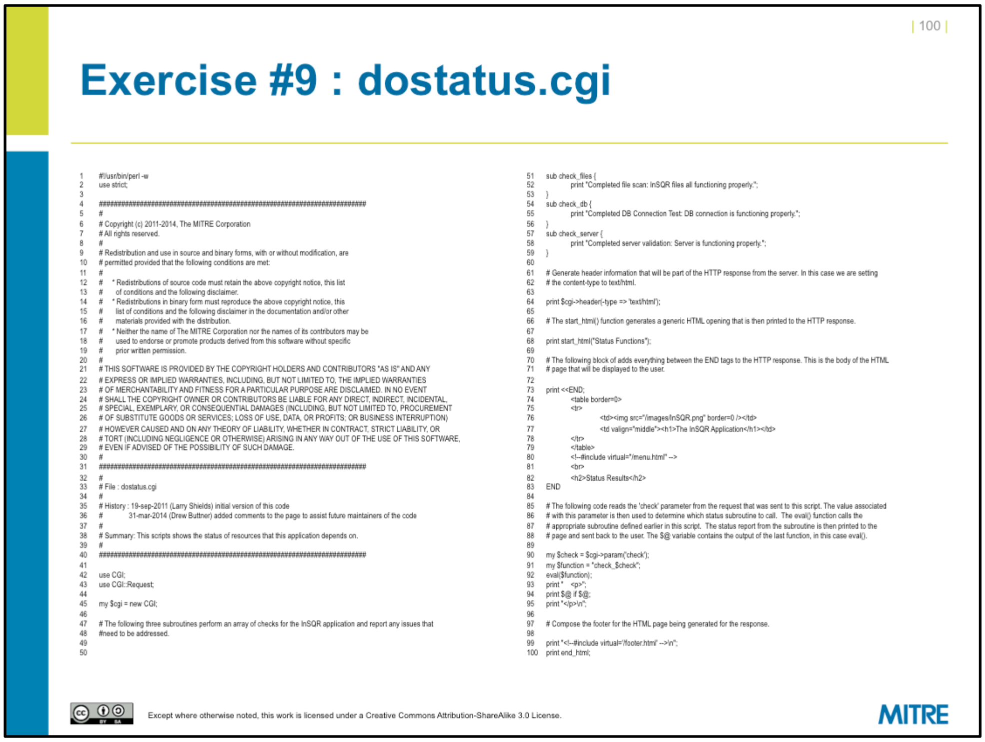
erability in some Netgear wireless routers that allows a remote attacker compromise a device and gain root privileges. The bug is trivially d the researcher who discovered it has posted a proof-of-concept

Root!

"What is happening here, as it so often does, is the host string gets copied into a shell command on the stack using sprintf(). This is probably the most straightforward buffer overflow vulnerability you will ever see. Sadly, you shouldn't exploit it. It is a tempting one to exploit because it is so clean and simple and because popping root with a MIPS ROP payload is sexy. But that would be silly, because right after it there is a call to system(). The system() function passes whatever string it is given to an invocation of /bin/sh. This is a command injection vulnerability in its purest form and is trivially exploitable. If the address string that gets passed in is something like "; evil_command; #", the ping6 command will be terminated prematurely, and evil_command will be executed right after it," Cutlip, a senior vulnerability researcher at Tactical Network Solutions, wrote in his explanation of the Netgear flaw.

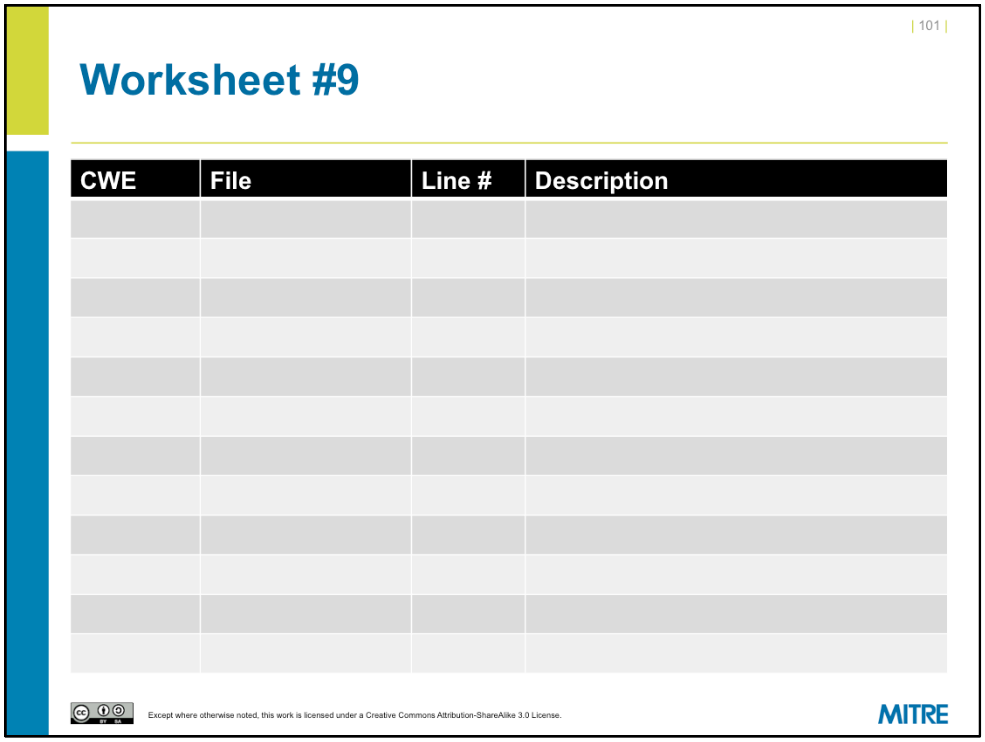
Programming Language Security Examined
April 15, 2014 , 12:08 pm

http://threatpost.com/some-netgear-routers-open-to-remote-authentication-bypass-command-injection/102689



Exercise #9 : dostatus.cgi

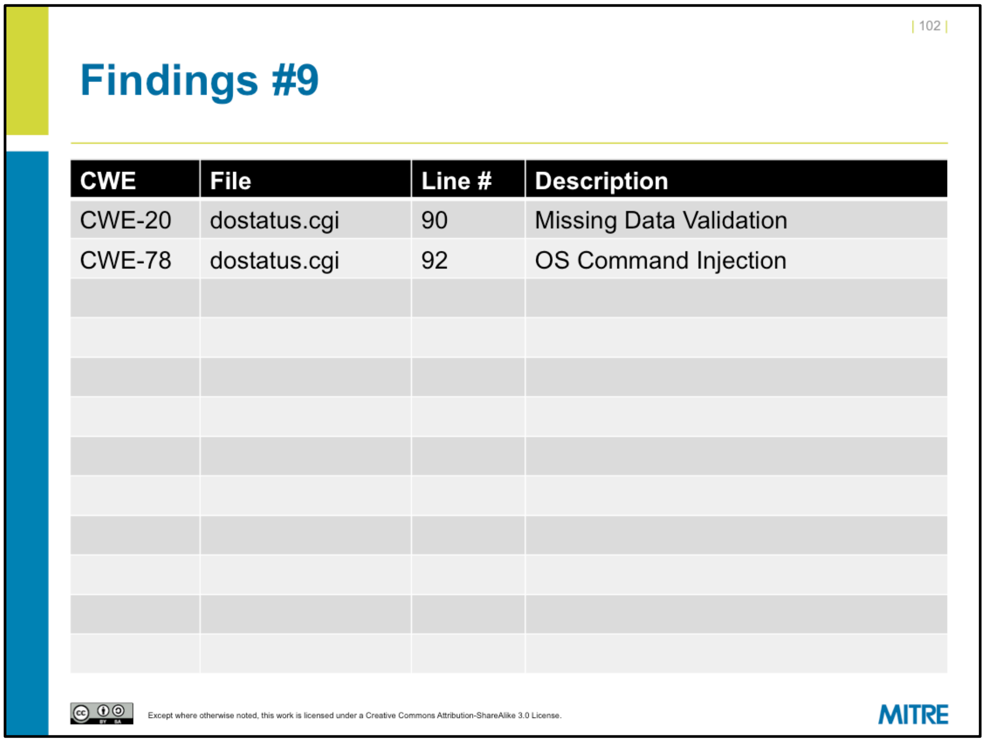
```
#!/usr/bin/perl -w
use strict;

###############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice, this list
#       of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright notice, this
#       list of conditions and the following disclaimer in the documentation and/or other
#       materials provided with the distribution.
#     * Neither the name of The MITRE Corporation nor the names of its contributors may be
#       used to endorse or promote products derived from this software without specific
#       prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
###############################################################################
#
# File : dostatus.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#           31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary: This scripts shows the status of resources that this application depends on.
#
###############################################################################

use CGI;
use CGI::Request;

my $cgi = new CGI;

# The following three subroutines perform an array of checks for the InSQR application and report any issues that
#need to be addressed.


sub check_files {
        print "Completed file scan: InSQR files all functioning properly.";
}
sub check_db {
        print "Completed DB Connection Test: DB connection is functioning properly.";
}
sub check_server {
        print "Completed server validation: Server is functioning properly.";
}

# Generate header information that will be part of the HTTP response from the server. In this case we are setting
# the content-type to text/html.

print $cgi->header(-type => 'text/html');

# The start_html() function generates a generic HTML opening that is then printed to the HTTP response.

print start_html("Status Functions");

# The following block of adds everything between the END tags to the HTTP response. This is the body of the HTML
# page that will be displayed to the user.

print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br>
        <h2>Status Results</h2>
END

# The following code reads the 'check' parameter from the request that was sent to this script. The value associated
# with this parameter is then used to determine which status subroutine to call.  The eval() function calls the
# appropriate subroutine defined earlier in this script.  The status report from the subroutine is then printed to the
# page and sent back to the user. The $@ variable contains the output of the last function, in this case eval().

my $check = $cgi->param('check');
my $function = "check_$check";
eval($function);
print "    <p>";
print $@ if $@;
print "</p>\n";

# Compose the footer for the HTML page being generated for the response.

print "<!--#include virtual='/footer.html' -->\n";
print end_html;
```



Worksheet #9

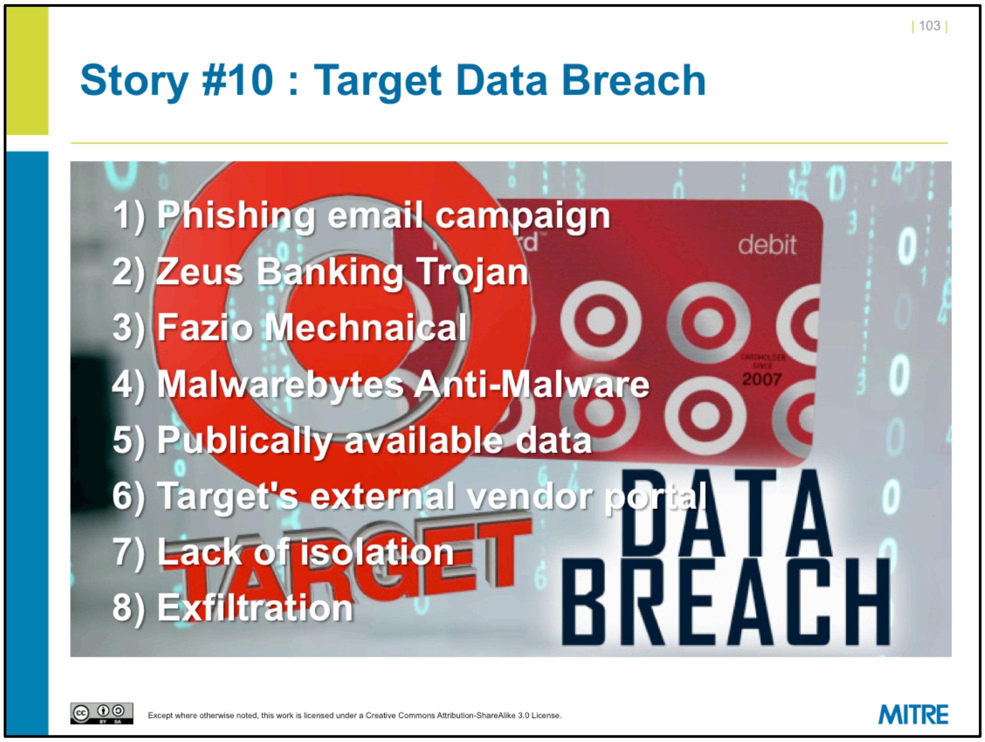
CWE	File	Line #	Description

MITRE



Findings #9

CWE	File	Line #	Description
CWE-20	dostatus.cgi	90	Missing Data Validation
CWE-78	dostatus.cgi	92	OS Command Injection



Story #10 : Target Data Breach

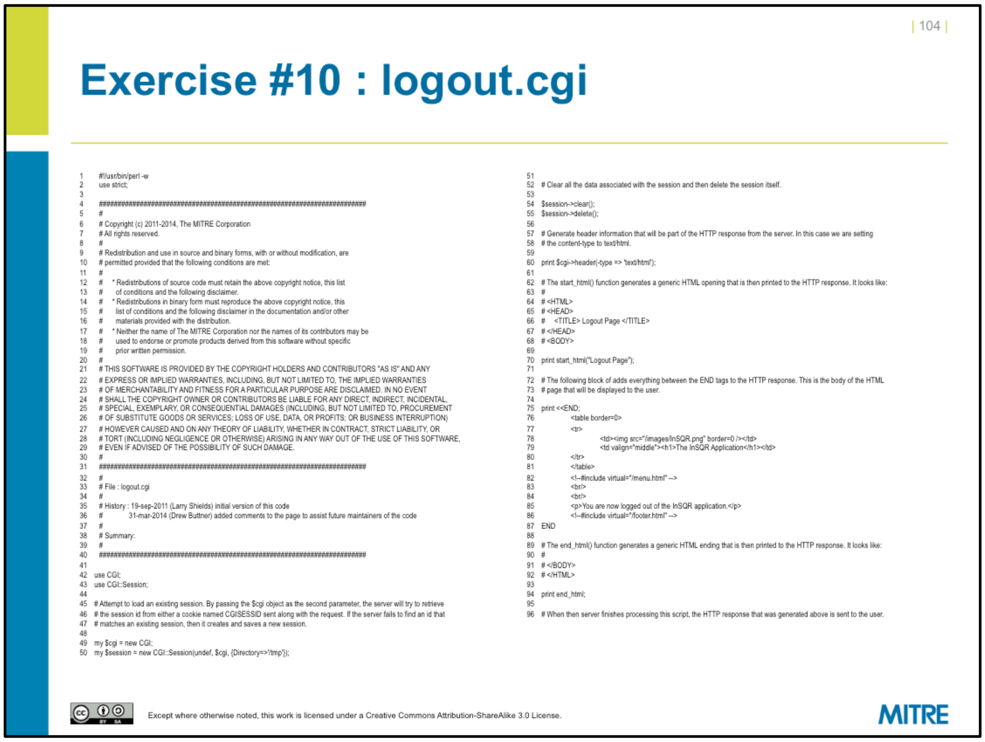
1) Phishing email campaign
2) Zeus Banking Trojan
3) Fazio Mechnaical
4) Malwarebytes Anti-Malware
5) Publically available data
6) Target's external vendor portal
7) Lack of isolation
8) Exfiltration

debit
CARDHOLDER SINCE
2007
TARGET
DATA
BREACH

MITRE

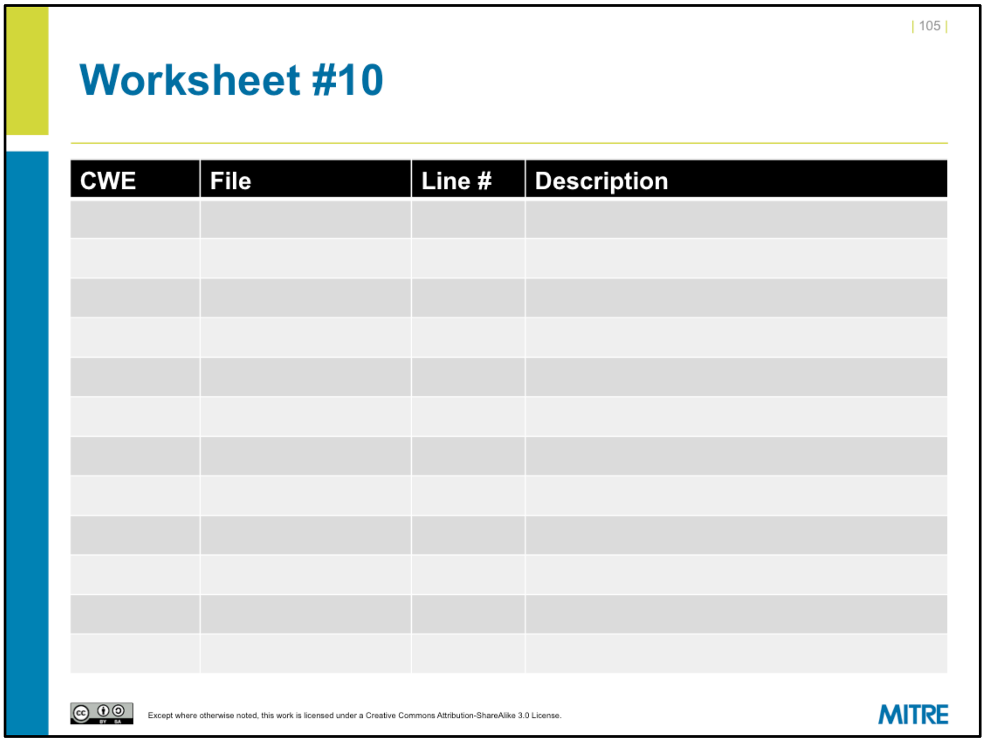
http://krebsonsecurity.com/2014/02/email-attack-on-vendor-set-up-breach-at-target/#more-24313

http://kansasfirstnews.com/2014/04/17/cyber-cops-target-hackers-may-take-years-to-find/



Exercise #10 : logout.cgi

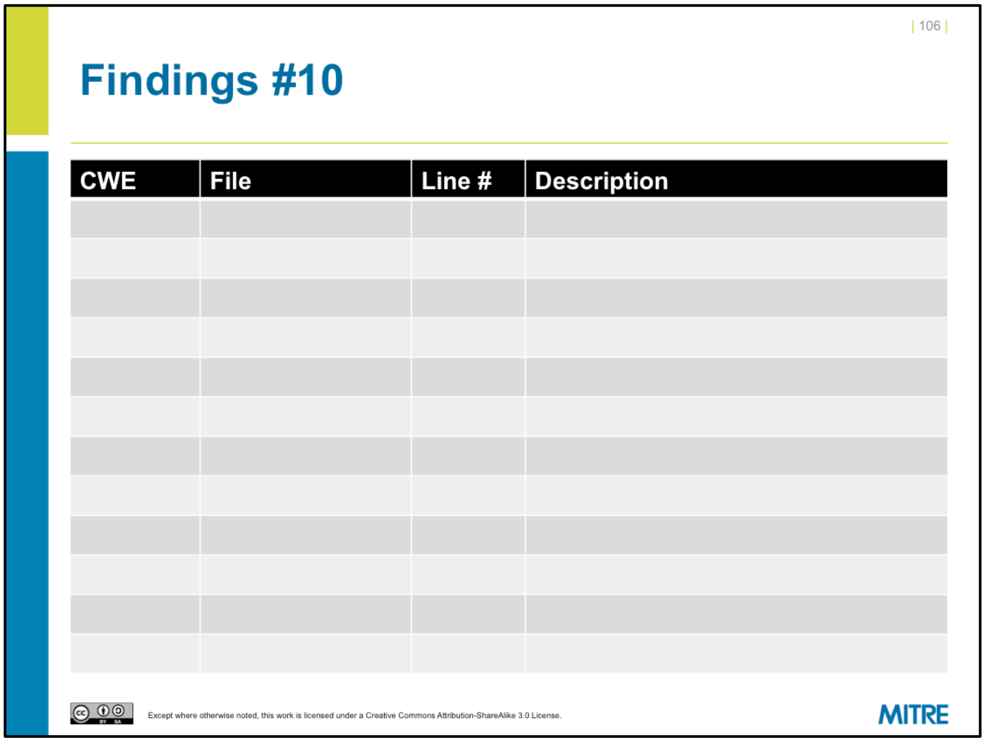
```
#!/usr/bin/perl -w
use strict;

##############################################################################
#
# Copyright (c) 2011-2014, The MITRE Corporation
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without modification, are
# permitted provided that the following conditions are met:
#
#     * Redistributions of source code must retain the above copyright notice, this list
#       of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright notice, this
#       list of conditions and the following disclaimer in the documentation and/or other
#       materials provided with the distribution.
#     * Neither the name of The MITRE Corporation nor the names of its contributors may be
#       used to endorse or promote products derived from this software without specific
#       prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
# OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
# SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
# OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
# TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
# EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
##############################################################################
#
# File : logout.cgi
#
# History : 19-sep-2011 (Larry Shields) initial version of this code
#           31-mar-2014 (Drew Buttner) added comments to the page to assist future maintainers of the code
#
# Summary:
#
##############################################################################

use CGI;
use CGI::Session;

# Attempt to load an existing session. By passing the $cgi object as the second parameter, the server will try to retrieve
# the session id from either a cookie named CGISESSID sent along with the request. If the server fails to find an id that
# matches an existing session, then it creates and saves a new session.

my $cgi = new CGI;
my $session = new CGI::Session(undef, $cgi, {Directory=>'/tmp'});

# Clear all the data associated with the session and then delete the session itself.

$session->clear();
$session->delete();

# Generate header information that will be part of the HTTP response from the server. In this case we are setting
# the content-type to text/html.

print $cgi->header(-type => 'text/html');

# The start_html() function generates a generic HTML opening that is then printed to the HTTP response. It looks like:
#
# <HTML>
# <HEAD>
#    <TITLE> Logout Page </TITLE>
# </HEAD>
# <BODY>

print start_html("Logout Page");

# The following block of adds everything between the END tags to the HTTP response. This is the body of the HTML
# page that will be displayed to the user.

print <<END;
        <table border=0>
        <tr>
                <td><img src="/images/InSQR.png" border=0 /></td>
                <td valign="middle"><h1>The InSQR Application</h1></td>
        </tr>
        </table>
        <!--#include virtual="/menu.html" -->
        <br/>
        <br/>
        <p>You are now logged out of the InSQR application.</p>
        <!--#include virtual="/footer.html" -->
END

# The end_html() function generates a generic HTML ending that is then printed to the HTTP response. It looks like:
#
# </BODY>
# </HTML>

print end_html;

# When then server finishes processing this script, the HTTP response that was generated above is sent to the user.
```



Worksheet #10

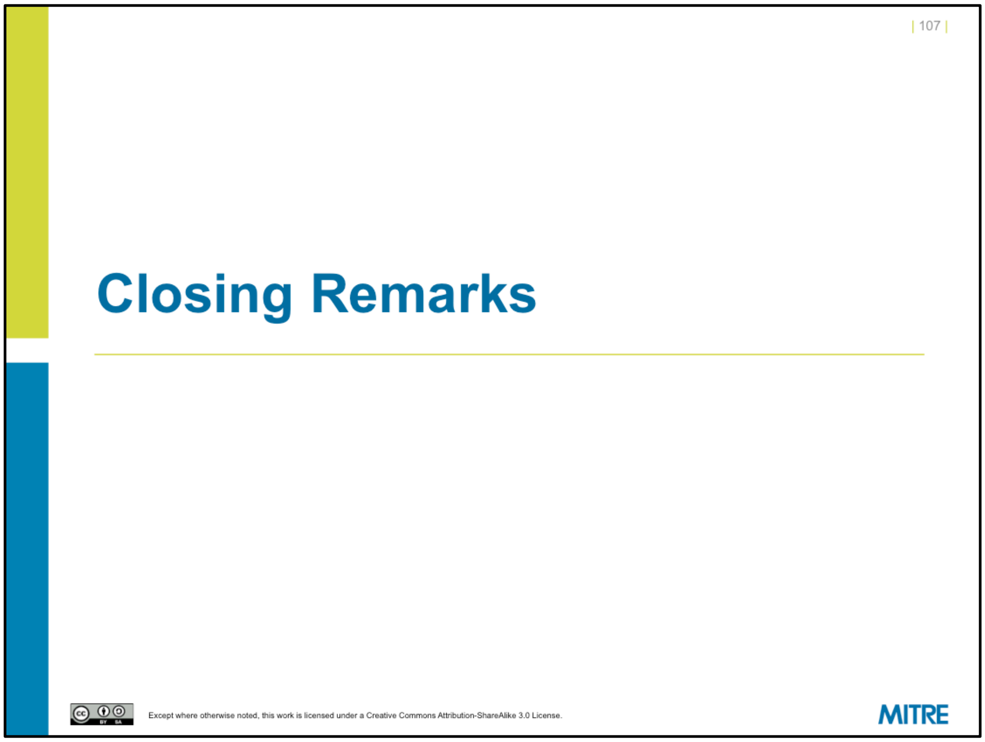
CWE	File	Line #	Description

MITRE

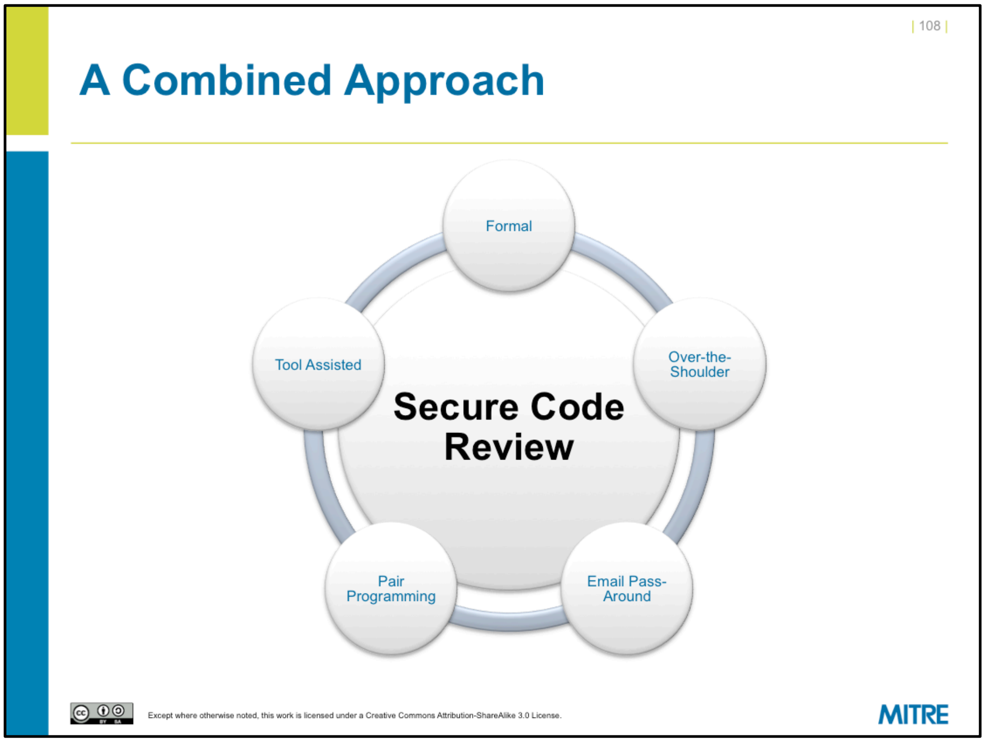


Findings #10

CWE	File	Line #	Description



Closing Remarks



| 108 |

A Combined Approach

Secure Code Review

Formal

Over-the-Shoulder

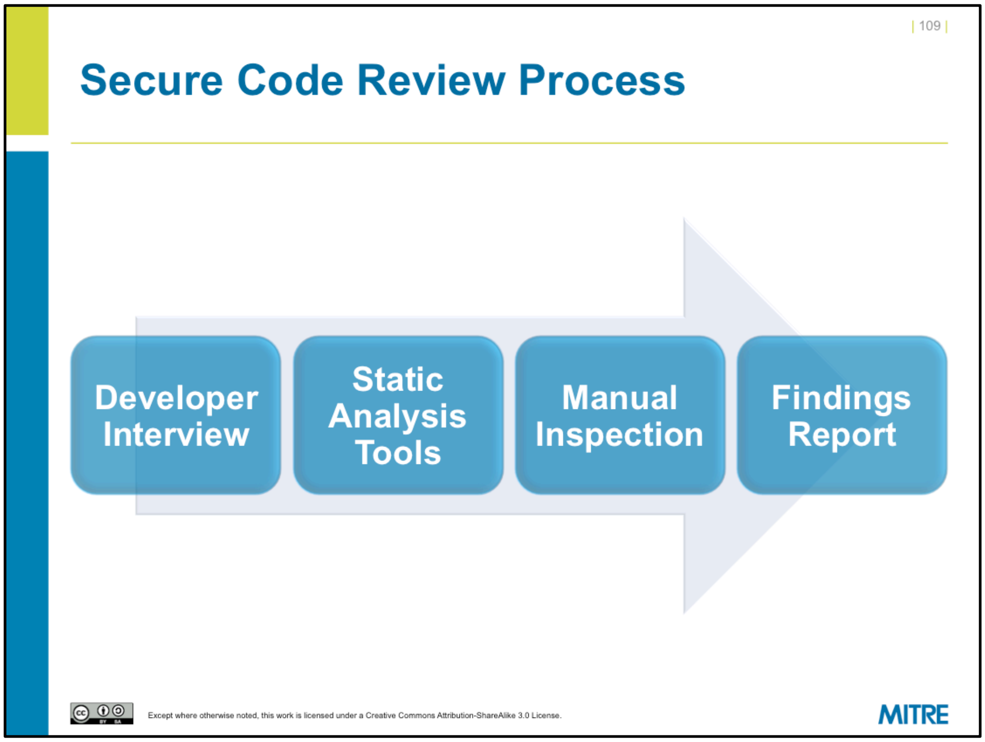
Email Pass-Around

Pair Programming

Tool Assisted

MITRE

A Secure Code Review looks to leverage elements from each of the different types of peer reviews.



Secure Code Review Process

Developer Interview

Static Analysis Tools

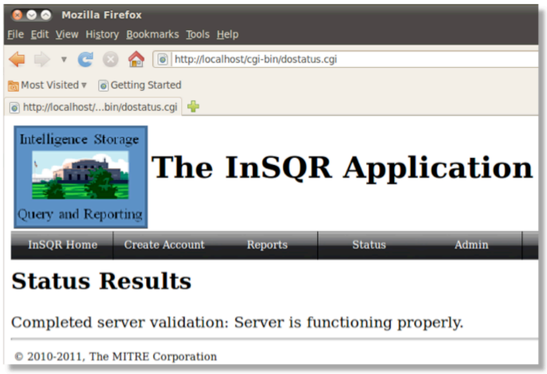
Manual Inspection

Findings Report

MITRE

Exercises

Together we performed a full review of The InSQR Application.

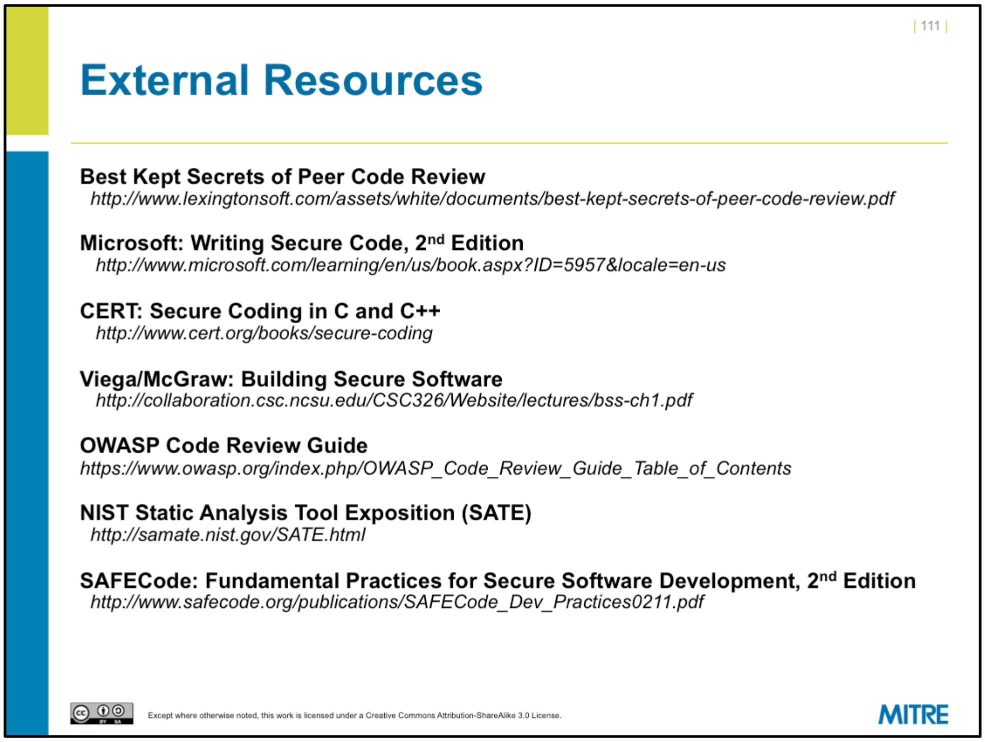


The InSQR Application

Status Results

Completed server validation: Server is functioning properly.

© 2010-2011, The MITRE Corporation



External Resources

Best Kept Secrets of Peer Code Review
http://www.lexingtonsoft.com/assets/white/documents/best-kept-secrets-of-peer-code-review.pdf

Microsoft: Writing Secure Code, 2nd Edition
http://www.microsoft.com/learning/en/us/book.aspx?ID=5957&locale=en-us

CERT: Secure Coding in C and C++
http://www.cert.org/books/secure-coding

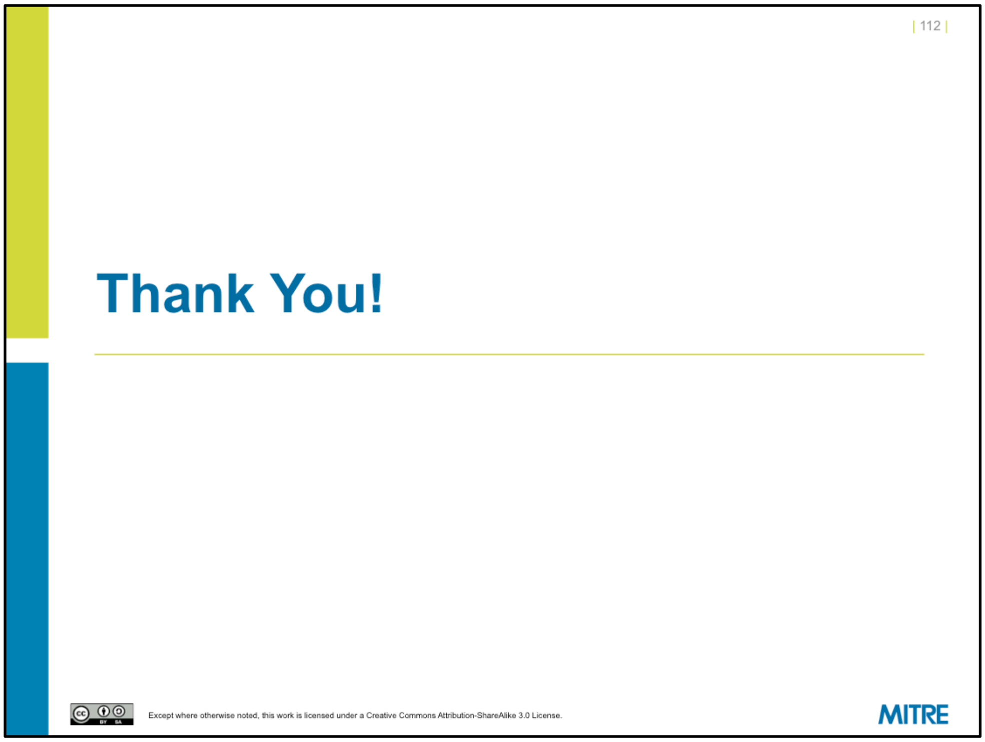
Viega/McGraw: Building Secure Software
http://collaboration.csc.ncsu.edu/CSC326/Website/lectures/bss-ch1.pdf

OWASP Code Review Guide
https://www.owasp.org/index.php/OWASP_Code_Review_Guide_Table_of_Contents

NIST Static Analysis Tool Exposition (SATE)
http://samate.nist.gov/SATE.html

SAFECode: Fundamental Practices for Secure Software Development, 2nd Edition
http://www.safecode.org/publications/SAFECode_Dev_Practices0211.pdf

MITRE



Thank You!