UNCOVERING VULNERABILITIES IN SECURE CODING GUIDELINES

Deepsec 2018

Fernando Arnaboldi

AGENDA

- Who, What, How, & Why
- Secure Coding Guidelines
  - NIST
  - OWASP
  - CWE
  - CERT
- Conclusions

WHO, WHAT, HOW, & WHY

/huː,hʊ, wɒt, haʊ, ənd wʌɪ/

(n) who cares about this, what was done, how it was performed, and why it makes sense to analyze it.

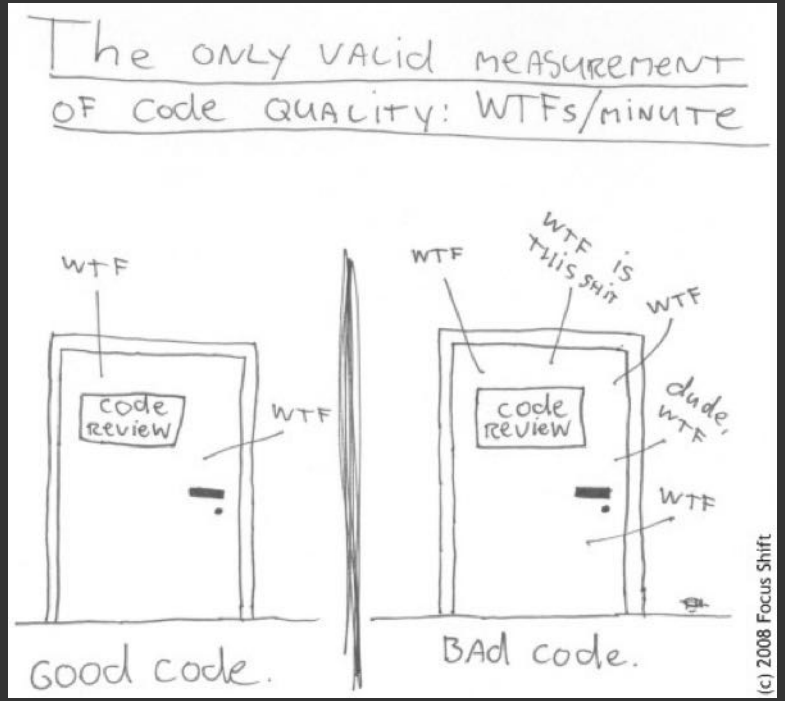
WHO CARES

- Security Consultants
- Software Architects

WHAT WAS ANALIZED

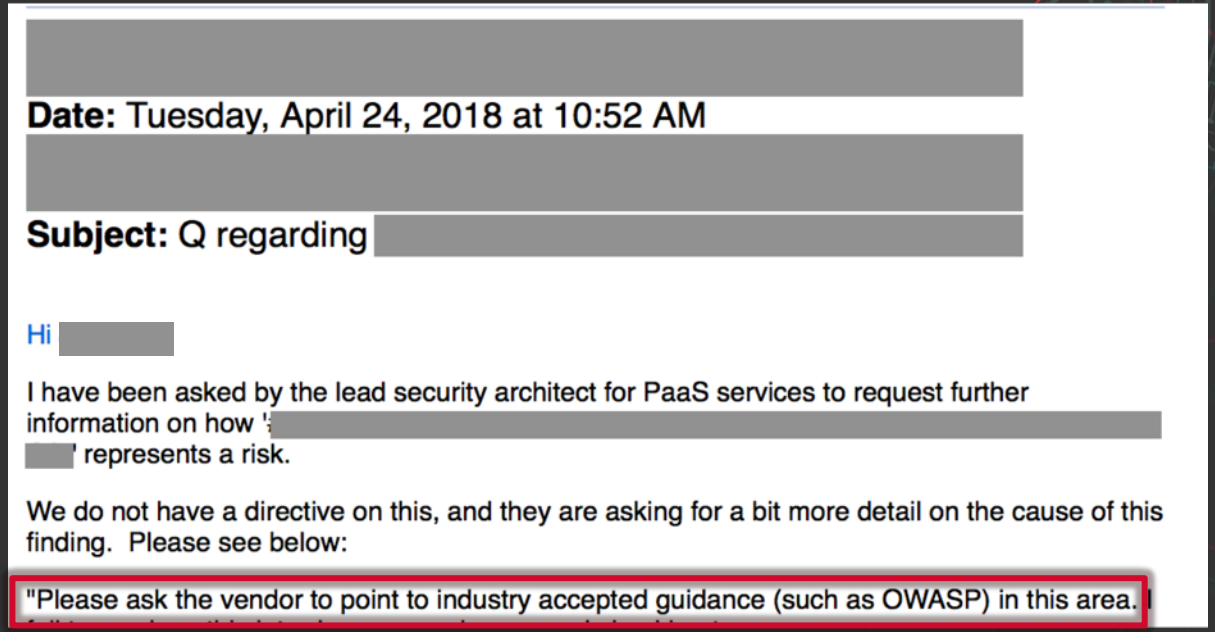
- Secure coding guidelines
- Secure pieces of code
- Coverage

HOW WAS ANALYZED



… & WHY

- People use them



Date: Tuesday, April 24, 2018 at 10:52 AM

Subject: Q regarding

Hi

I have been asked by the lead security architect for PaaS services to request further information on how ' ' represents a risk.

We do not have a directive on this, and they are asking for a bit more detail on the cause of this finding. Please see below:

"Please ask the vendor to point to industry accepted guidance (such as OWASP) in this area.

WHAT'S THE GOAL

- Create awareness what may happen with guidelines:
  - They may have mistakes
  - They may have backdoors
  - However, they are important

```
+ if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
+         retval = -EINVAL;
```

SECURE CODING GUIDELINES: NIST

NIST Software Assurance Metrics

/nist/

(n) a non-regulatory agency of the United States Department of Commerce dedicated to improving software assurance. Software assurance indicates that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its lifecycle, and that the software functions in the intended manner.

THE PHP INCLUDE PROBLEM

- This test case shows how to solve the “PHP Include” problem
- **Fail**: Using the .inc file extension will expose the code when referencing files directly

```
19  This is my main template.<br />
20  <?php
21
22      function fix_phpInclude($name)
23      {
24          /*
25              One of the most secure way is to set what you want to dive into
26              in your file/template/bdd...
27          */
28          switch($name) {
29              case 'links.inc':
30              case 'index.inc':
31                  return $name;
32              default:
33                  return 'error.inc';
34          }
35          return 'error.inc';
36      }
37
38      $page = "index.inc";
39      if (isset($_GET['page']) && strlen($_GET['page']) > 0)
40          $page = htmlentities($_GET['page']);
41
42      include (fix_phpInclude($page));
43  ?>
```

Source: https://samate.nist.gov/SARD/view_testcase.php?tID=1768

SECURE CODING GUIDELINES: OWASP

OWASP

/owɒsp/ 

(n) the Open Web Application Security Project is a worldwide, not-for-profit charitable organization focused on improving the security of software.

USERNAME GUIDANCE

- Make sure your usernames/user-ids are case insensitive. Many sites use email addresses for usernames and email addresses are already case insensitive. Regardless, it would be very strange for user 'smith' and user 'Smith' to be different users. Could result in serious confusion.

Source: OWASP Source Code Review 2.0 Alpha – Page 59:
https://www.owasp.org/images/7/78/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf

BRUTE-FORCING USERNAMES

- What if your users are aware?
- Assume someone is brute-forcing 10.000 user accounts with three different passwords
- If user identifications are case insensitive
  - 10.000*3 = **30.000 requests**
- Case sensitive usernames with six letters require 2^6 combinations per username
  - i.e. master, Master, mAster
  - 10.000 * 2^6 * 3 = 10.000 * 64 * 3 = **1.920.000 requests**

EXCEPTIONS IN TRY/CATCH BLOCKS

2008

Example:

Java Try-Catch:

```
public class DoStuff {
    public static void Main() {
        try {
            StreamReader sr = File.OpenText("stuff.txt");
            Console.WriteLine("Reading line {0}", sr.ReadLine());
        }
        catch(MyClassExtendedFromException e) {
            Console.WriteLine("An error occurred. Please leave to room");
                logerror("Error: ", e);
        }
    }
}
```

2016

Figure A10.16

Java try-catch block

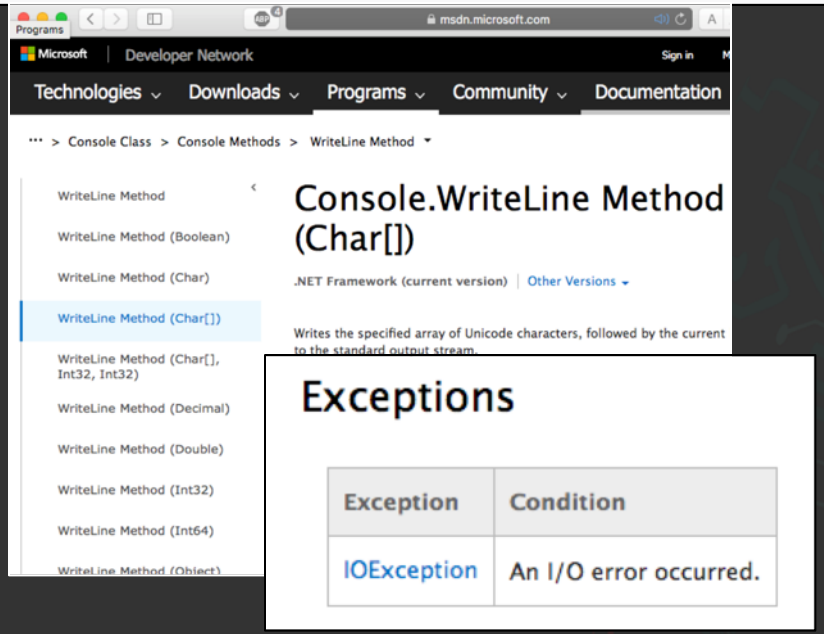
```
try {
    StreamReader sr = File.OpenText("stuff.txt");
    Console.WriteLine("Reading line {0}", sr.ReadLine());
}
catch(MyClassExtendedFromException e) {
    Console.WriteLine("An error occurred. Please leave to room");
        logerror("Error:", e);
}
}
}
```

FYI, this is in fact .NET code, not Java

EXCEPTIONS IN TRY/CATCH BLOCKS (CONT'D)

- **Fail:** This function may throw an IOException within a catch block

```
Console.WriteLine("An error occurred. Please leave to room");
```



CRYPTOGRAPHY

Best Practices

Industry leading Cryptographer's are advising that MD5 and SHA-1 should not be used for any applications. The United State FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION (FIPS) specifies seven cryptographic hash algorithms — SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 are approved for federal use. The code reviewer should consider this standard because the FIPS is also widely adopted by the information technology industry.

What to Review

When reviewing code modules which perform authentication functions, some common issues to look out for include:

- Ensure the login page is only available over TLS. Some sites leave the login page has HTTP, but make the form

Source: OWASP Source Code Review 2.0 Alpha – Page 122 & 59:
https://www.owasp.org/images/7/78/OWASP_AlphaRelease_CodeReviewGuide2.0.pdf

SECURE CODING GUIDELINES: CWE

CWE

/cwe/

(n) the Common Weakness Enumeration is an organization sponsored by United States Computer Emergency Readiness Team (US-CERT). It defines a set of software weaknesses, which enumerate design and architectural vulnerabilities as well as low-level coding and design errors.

CWE-456: MISSING INITIALIZATION OF A VARIABLE

- **Fail #1:** Username disclosure (timing)
- **Fail #2:** Error may not be logged
- **Fail #3:** Logging injection

Example Language: **Java**

```
public class BankManager {
  // user allowed to perform bank manager tasks
  private User user = null;
  private boolean isUserAuthentic = false;
  // constructor for BankManager class

  public BankManager(String username) {
    user = getUserFromUserDatabase(username);

  }
  // retrieve user from database of users

  public User getUserFromUserDatabase(String username) {...}
  // authenticate user

  public boolean authenticateUser(String username, String password) {
    if (user == null) {
      System.out.println("Cannot find user " + username);

    }
    else {
      if (password.equals(user.getPassword())) {
        isUserAuthentic = true;

      }

    }
    return isUserAuthentic;

  }
```

Source: https://cwe.mitre.org/data/definitions/456.html

CWE-749: EXPOSED DANGEROUS METHOD OR FUNCTION

- **Fail:** SQL Injection

Example Language: **Java**

```
private void removeDatabase(String databaseName) {
  try {
    Statement stmt = conn.createStatement();
    stmt.execute("DROP DATABASE " + databaseName);

  } catch (SQLException ex) {...}
  }
```

Source: https://cwe.mitre.org/data/definitions/749.html

SECURE CODING GUIDELINES: CERT

CERT

/səːt/ 

(n) a non-profit United States federally funded research and development center.

ERR01-J. DO NOT ALLOW EXCEPTIONS TO EXPOSE SENSITIVE INFORMATION

- Failure to filter sensitive information when propagating exceptions often results in information leaks that can assist an attacker's efforts to develop further exploits.

- An attacker may craft input arguments to expose internal structures and mechanisms of the application.

Source: https://wiki.sei.cmu.edu/confluence/display/java/ERR01-J.+Do+not+allow+exceptions+to+expose+sensitive+information

ERR01-J. DO NOT ALLOW EXCEPTIONS (CONT'D)

- "*This compliant solution implements the policy that only files that live in c:\homepath may be opened by the user and that the user is not allowed to discover anything about files outside this directory*"

```
class ExceptionExample {
  public static void main(String[] args) {

    File file = null;
    try {
      file = new File(System.getenv("APPDATA") +
             args[0]).getCanonicalFile();
      if (!file.getPath().startsWith("c:\\homepath")) {
        System.out.println("Invalid file");
        return;
      }
    } catch (IOException x) {
      System.out.println("Invalid file");
      return;
    }

    try {
      FileInputStream fis = new FileInputStream(file);
    } catch (FileNotFoundException x) {
      System.out.println("Invalid file");
      return;
    }
  }
}
```

ERR01-J. DO NOT ALLOW EXCEPTIONS (CONT'D)

- Demo: *How to expose information*

ERR01-J. DO NOT ALLOW EXCEPTIONS (CONT'D)

- *"only files that live in c:\homepath may be opened by the user"*

```
file = new File(System.getenv("APPDATA") +
       args[0]).getCanonicalFile();
if (!file.getPath().startsWith("c:\\homepath")) {
  System.out.println("Invalid file");
  return;
```

- **Fail #1:** What about C:\\homepathfail?

ERR01-J. DO NOT ALLOW EXCEPTIONS (CONT'D)

```
file = new File(System.getenv("APPDATA") +
        args[0]).getCanonicalFile();
```

```
$ java ExceptionExample
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at ExceptionExample.main(ExceptionExample.java:11)
```

- **Fail #2:** Does not handle all exceptions
- **Fail #3:** Does not validate method arguments

ERR01-J. DO NOT ALLOW EXCEPTIONS (CONT'D)

```
if (!file.getPath().startsWith("c:\\homepath")) {
  System.out.println("Invalid file");
  return;
}
```

- **Fail #4:** Error may not be logged

FIO00-J. DO NOT OPERATE ON FILES IN SHARED DIRECTORIES

- *"To prevent vulnerabilities, a program must operate only on files in secure directories. […] file links can be swapped out and may not always point to the intended location. As a result, file links in shared directories are untrusted and should not be operated on"*

```
try {
  if (Files.isSymbolicLink(partialPath)) {
    if (!isInSecureDir(Files.readSymbolicLink(partialPath),)) {
                        user, symlinkDepth - 1)
      // Symbolic link, linked-to dir not secure
      return false;
    }
```

Source: https://wiki.sei.cmu.edu/confluence/display/java/FIO00-J.+Do+not+operate+on+files+in+shared+directories

FIO00-J. DO NOT OPERATE ON FILES (CONT'D)

- Demo: *How to leave a secure directory*

Source: https://wiki.sei.cmu.edu/confluence/display/java/FIO00-J.+Do+not+operate+on+files+in+shared+directories

FIO00-J. DO NOT OPERATE ON FILES (CONT'D)

1. Create a symlink and a hard link:

```
$ ln -s /tmp/extract_data symlink
$ ln /tmp/extract_data hardlink
$ ls -l
lrwxr-xr-x  1 fear  staff  17 Nov 27 16:33 symlink -> /tmp/extract_data
-rw-r--r--  2 fear  staff   0 Nov 27 16:32 hardlink
```

2. Check the symlink:

```
$ java isInSecureDir /Users/fear/deepsec/symlink
Accessing: /tmp/extract_data
File not in secure directory
```

3. Check the hard link:

```
$ java isInSecureDir /Users/fear/deepsec/hardlink
Accessing: /Users/test/hardlink
$
```

CONCLUSIONS

/kənˈkluːʒ(ə)ns/

(n) how did this happen and where do we go from here?

CONCLUSIONS

- Lessons learned from the identified issues
- Who should define the guidelines
- How to write secure code