ENPM 809W
Introduction to Secure Software Engineering

Gananand Kini

Lecture 11

Error Handling and Logging related issues - Attacks

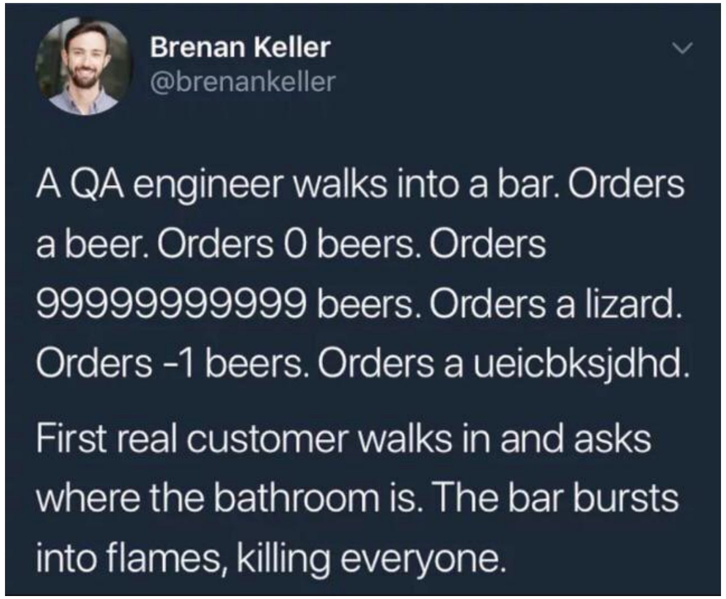
Outline

- **Error Handling Core Concepts**
- **Error Handling weaknesses**
- **Logging and Audit Core Concepts**
- **Logging and Audit weaknesses**

Error Handling Core Concepts

Error Handling Core Concepts

- **While your software system may be designed to run predictably, real world and users will throw curve balls.**
- **Your software system should be designed to handle such unexpected issues and handle any errors that crop up.**
- **Errors occur when the software system has to handle unexpected situations. For example:**
  - Invalid input
  - Invalid code flow paths
  - Invalid state in the software system
  - Not enough resources to continue operation
  - Etc.



Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

Sources:
1. https://9gag.com/gag/aEgnYz9

Error Handling Core Concepts contd…

- **Typically:**
  - Should design system to handle such error conditions
  - Should gracefully recover or exit the system from such conditions
- **Real World:**
  - Very tight deadlines to hit the target release date, so ignore errors
  - Not enough manpower to write code for error handling, the features make up the entire budget!
  - Insert a real world ~~problem~~ excuse here…

Errors at many different levels

- **Typically, developers handle errors at the application level with try … catch … finally blocks.**
- **However, there are many levels to the software system:**
  - In web applications, you can have page level errors.
  - You can have site-level errors (where a site hosts a web application and can involve resource pools etc.)
  - You can have Operating System level errors (for example, running out of resources like memory or disk etc.)
  - You can have Network level errors (unable to establish a connection etc.)
  - You can even have logical errors in the application where the state of the software system itself is inconsistent.

When things get real …

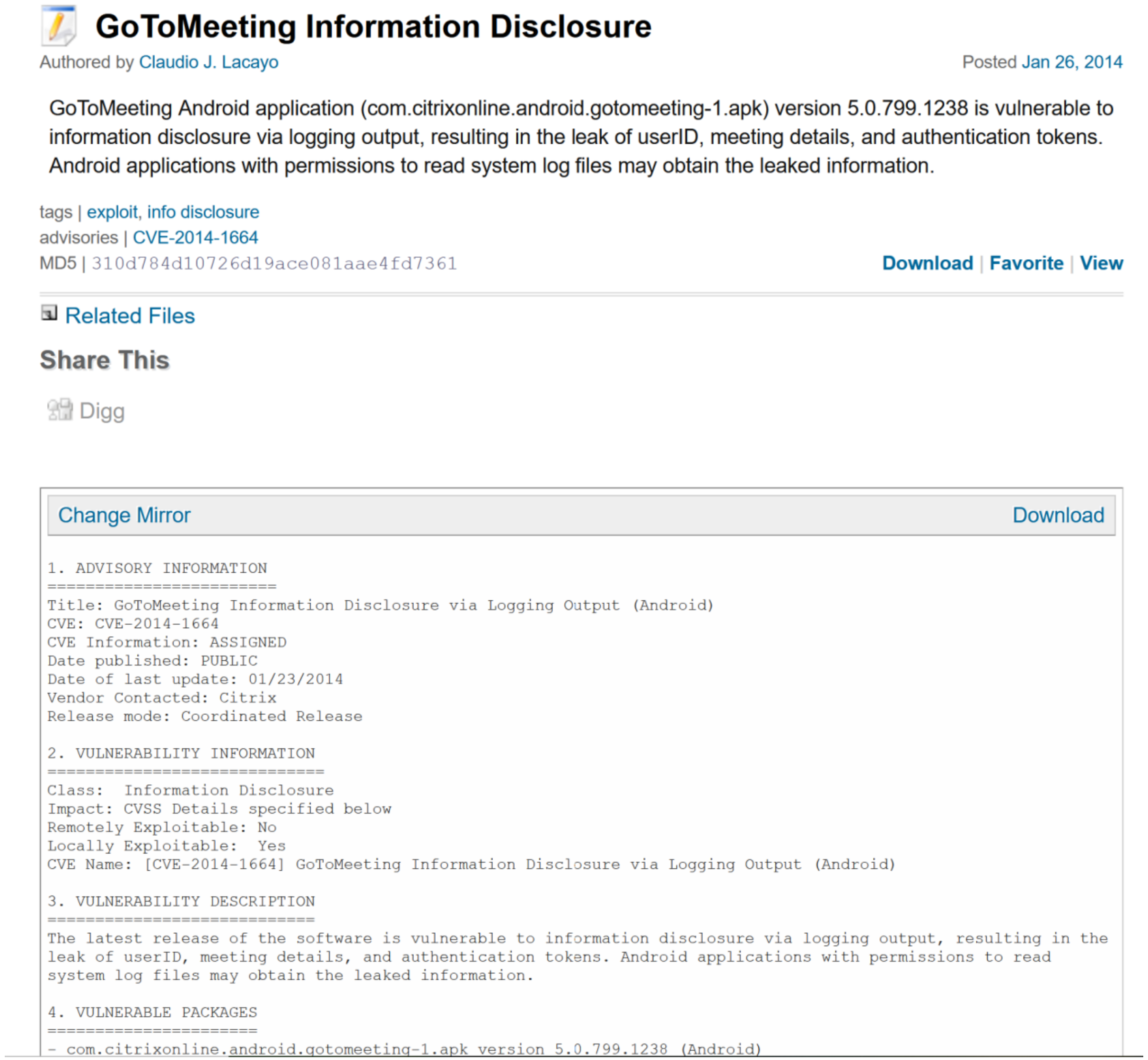
- **Developers do use debug logging to log errors etc.**
  - This is fine!

- **However, what happens when the product is released with such code?**

Sources:
1. KC Green. Gunshow comic. http://gunshowcomic.com/648. Retrieved July 10, 2021.

GoToMeeting Information Disclosure Vulnerability via Logging

- GoToMeeting Android application leaked sensitive information and authentication tokens to LogCat (the logging system used in Android).



GoToMeeting Information Disclosure

Authored by Claudio J. Lacayo Posted Jan 26, 2014

GoToMeeting Android application (com.citrixonline.android.gotomeeting-1.apk) version 5.0.799.1238 is vulnerable to information disclosure via logging output, resulting in the leak of userID, meeting details, and authentication tokens. Android applications with permissions to read system log files may obtain the leaked information.

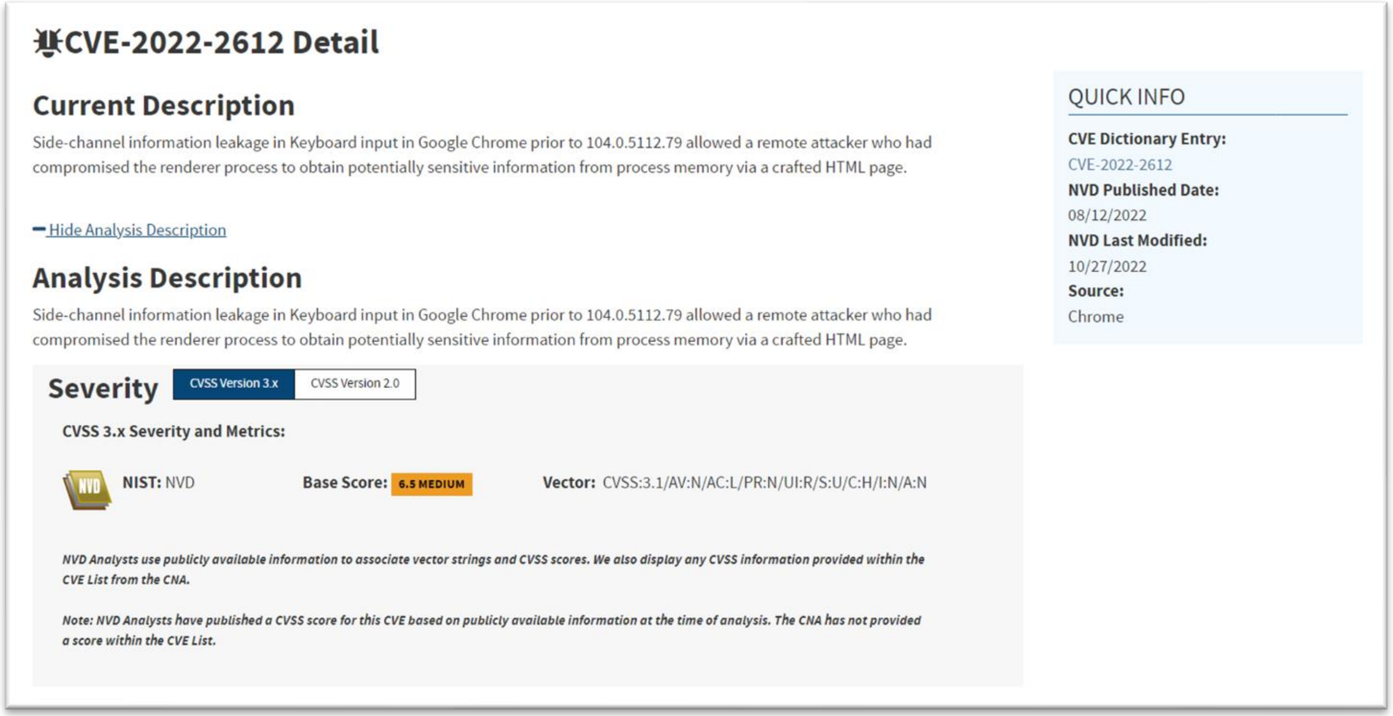
tags | exploit, info disclosure
advisories | CVE-2014-1664
MD5 | 310d784d10726d19ace081aae4fd7361

```
1. ADVISORY INFORMATION
========================
Title: GoToMeeting Information Disclosure via Logging Output (Android)
CVE: CVE-2014-1664
CVE Information: ASSIGNED
Date published: PUBLIC
Date of last update: 01/23/2014
Vendor Contacted: Citrix
Release mode: Coordinated Release

2. VULNERABILITY INFORMATION
=============================
Class:  Information Disclosure
Impact: CVSS Details specified below
Remotely Exploitable: No
Locally Exploitable:  Yes
CVE Name: [CVE-2014-1664] GoToMeeting Information Disclosure via Logging Output (Android)

3. VULNERABILITY DESCRIPTION
============================
The latest release of the software is vulnerable to information disclosure via logging output, resulting in the
leak of userID, meeting details, and authentication tokens. Android applications with permissions to read
system log files may obtain the leaked information.

4. VULNERABLE PACKAGES
======================
- com.citrixonline.android.gotomeeting-1.apk version 5.0.799.1238 (Android)
```

Sources:
1. Lacayo, Claudio J. Packetstorm Security: GoToMeeting Information Disclosure.January 26, 2014. Retrieved July 10, 2021.

Information Leakage in Keyboard input in Google Chrome

- Side-channel information leakage in Keyboard input in Google Chrome prior to 104.0.5112.79 allowed a remote attacker who had compromised the renderer process to obtain potentially sensitive information from process memory via a crafted HTML page.



CVE-2022-2612 Detail

Current Description

Side-channel information leakage in Keyboard input in Google Chrome prior to 104.0.5112.79 allowed a remote attacker who had compromised the renderer process to obtain potentially sensitive information from process memory via a crafted HTML page.

Analysis Description

Side-channel information leakage in Keyboard input in Google Chrome prior to 104.0.5112.79 allowed a remote attacker who had compromised the renderer process to obtain potentially sensitive information from process memory via a crafted HTML page.

Severity

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 6.5 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

QUICK INFO

CVE Dictionary Entry:
CVE-2022-2612
NVD Published Date:
08/12/2022
NVD Last Modified:
10/27/2022
Source:
Chrome

Sources:
1. https://nvd.nist.gov/vuln/detail/CVE-2022-2612.

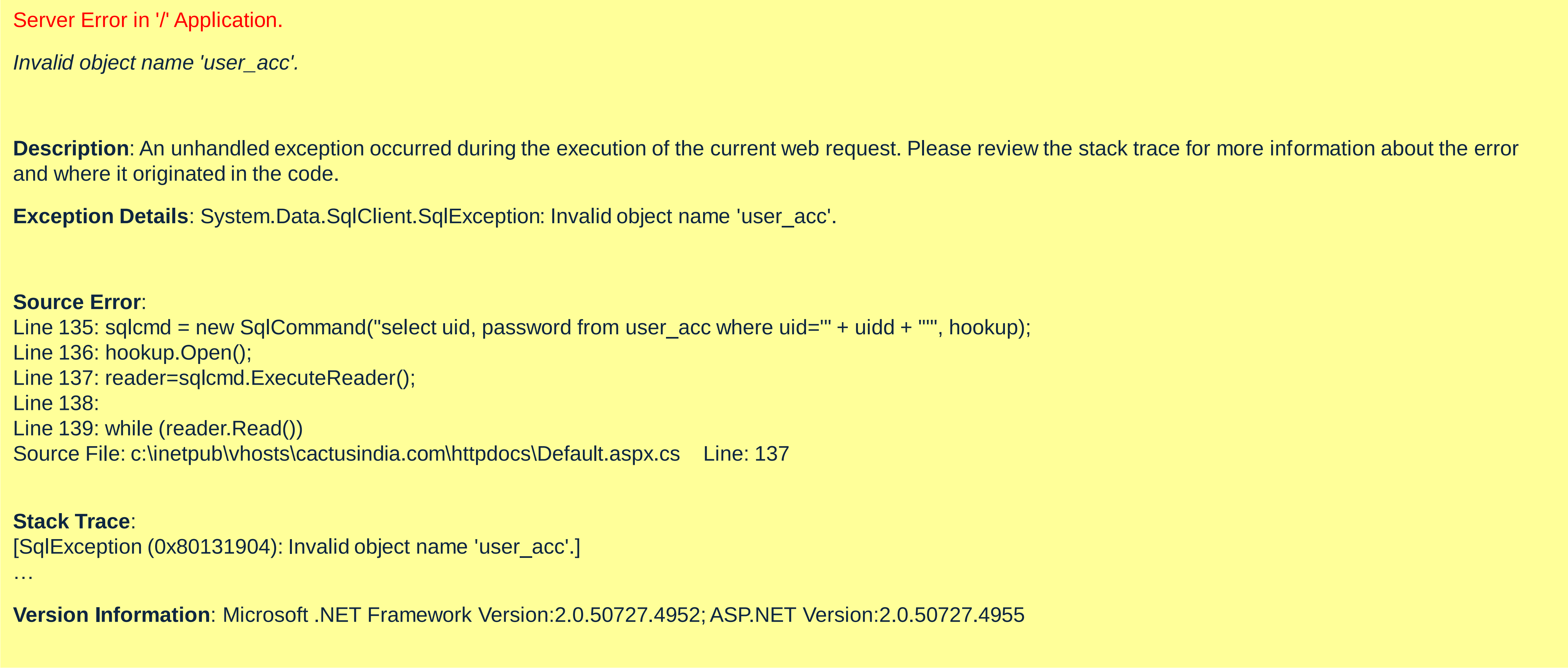
Error Handling Weaknesses

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

- **Description: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.**
- **There are many different kinds of mistakes that introduce information exposures. The severity of the error can range widely, depending on the context in which the product operates, the type of sensitive information that is revealed, and the benefits it may provide to an attacker.**
- **Information might be sensitive to different parties, each of which may have their own expectations for whether the information should be protected.**
- **Information exposures can occur in different ways:**
  - the code explicitly inserts sensitive information into resources or messages that are intentionally made accessible to unauthorized actors, but should not contain the information - i.e., the information should have been "scrubbed" or "sanitized"
  - a different weakness or mistake indirectly inserts the sensitive information into resources, such as a web script error revealing the full system path of the program.
  - the code manages resources that intentionally contain sensitive information, but the resources are unintentionally made accessible to unauthorized actors. In this case, the information exposure is resultant - i.e., a different weakness enabled the access to the information in the first place.

Sources:
1. CWE-200: Exposure of Sensitive Information to an Unauthorized Actor. MITRE CWE. https://cwe.mitre.org/data/definitions/200.html. Retrieved July 20, 2021.

Real World Example – Information Leak (Old school ASP .NET)



Server Error in '/' Application.

Invalid object name 'user_acc'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Invalid object name 'user_acc'.

Source Error:

```
Line 135: sqlcmd = new SqlCommand("select uid, password from user_acc where uid='" + uidd + "'", hookup);
Line 136: hookup.Open();
Line 137: reader=sqlcmd.ExecuteReader();
Line 138:
Line 139: while (reader.Read())
Source File: c:\inetpub\vhosts\cactusindia.com\httpdocs\Default.aspx.cs    Line: 137
```

Stack Trace:

```
[SqlException (0x80131904): Invalid object name 'user_acc'.]
…
```

Version Information: Microsoft .NET Framework Version:2.0.50727.4952; ASP.NET Version:2.0.50727.4955

Jating2. (2010, November 15). *Server error in '/' application.* Retrieved February 21, 2011, from http://forums.asp.net/p/1623626/4169865.aspx

Information Leakage

- **Information leakages are an attackers best friend!**
- **Is each of the following an example of information leakage or not?**
  - User account does not have sufficient funds to perform this transaction. Minimum required balance is $5,000.
  - User password must be a minimum of 8 characters.
  - Failed validation – username must not contain the characters < > ‘ “ ( ) ;

Call Only Interfaces Intended for Programmers

- **Usually unwise to invoke a program intended for human interaction (text or GUI)**
  - Programs for humans are intentionally rich & often difficult to completely control
  - May have "escape hatches" to unintended functions
  - Interactive programs often try to intuit the "most likely" meaning
  - May not be what you were expecting.
  - Attacker may find a way to exploit this.
- **Usually there's a different program/API for other programs' use; use that instead**
  - E.G., don't invoke ed/vi/emacs for text processing, use sed/awk/perl
  - In ASP.NET Core Razor pages, using "asp-action" instead of manually writing onClick() javascript that performs an HTTP request to navigate to another page.
- **Similarly, provide an API/programmer interface if sensible**
  - Perhaps have GUI that then invokes API (good approach anyway)

Check All System Call Returns

- **Check every system call that can return an error condition**
  - Nearly all system calls require limited system resources, and users can often affect resources in a variety of ways
  - If the error cannot be handled gracefully, then fail safe

Check information when it returns

- **Reuse input filtering concepts from earlier… values from libraries are yet more input**
  - If number: Is it within some plausible range?
  - If string: Does it match a allowlist filter?
  - If complex (e.g., file/data type):

Is it one of the permitted file/data types?

  - If an image (e.g., tile from geometry server):

Height/width in range? Is it really an image format?

  - If it takes too long to respond, consider alternatives
- **This can be hard to do everywhere, in which case, prioritize where it's riskier**
- **Can be helpful in countering defects in components, even if it is not security issue**

CWE-252: Unchecked Return Value

- **Description: The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.**
- **Two common programmer assumptions are "this function call can never fail" and "it doesn't matter if this function call fails".**
- **If an attacker can force the function to fail or otherwise return a value that is not expected, then the subsequent program logic could lead to a vulnerability, because the software is not in a state that the programmer assumes.**
- **For example, if the program calls a function to drop privileges but does not check the return code to ensure that privileges were successfully dropped, then the program will continue to operate with the higher privileges.**

Sources:

1. CWE-252: Unchecked Return Value. MITRE CWE. https://cwe.mitre.org/data/definitions/252.html. Retrieved July 20, 2021.

Beware of functions that change state of the software system …

- When certain methods are called, developers make assumptions about the state of the system.
- HTTP for example has multiple verbs supported that might change the state on the webserver.
- Developers have to verify the operations are not going to cause side effects …
- This is part of “logical” error cases handling.

CWE-650: Trusting HTTP Permission Methods on the Server Side

- Description: The server contains a protection mechanism that assumes that any URI that is accessed using HTTP GET will not cause a state change to the associated resource.
- This might allow attackers to bypass intended access restrictions and conduct resource modification and deletion attacks, since some applications allow GET to modify state.
- The HTTP GET method and some other methods are designed to retrieve resources and not to alter the state of the application or resources on the server side. Furthermore, the HTTP specification requires that GET requests (and other requests) should not have side effects. Believing that it will be enough to prevent unintended resource alterations, an application may disallow the HTTP requests to perform DELETE, PUT and POST operations on the resource representation.
- However, there is nothing in the HTTP protocol itself that actually prevents the HTTP GET method from performing more than just query of the data. Developers can easily code programs that accept a HTTP GET request that do in fact create, update or delete data on the server.
- For instance, it is a common practice with REST based Web Services to have HTTP GET requests modifying resources on the server side. However, whenever that happens, the access control needs to be properly enforced in the application. No assumptions should be made that only HTTP DELETE, PUT, POST, and other methods have the power to alter the representation of the resource being accessed in the request.

Sources:

1. CWE-650: Trusting HTTP Permission Methods on the Server Side. MITRE CWE. https://cwe.mitre.org/data/definitions/650.html. Retrieved July 20, 2021.

Other Error Handling weaknesses

- **Handle errors or exceptions. See CWE-248. Most languages now allow for this. No excuses.**
- **Generically handling exceptions where the expectation is to perform different operations for different kinds of errors. See CWE-396.**
- **Slightly more subtle problem: Improper clean up of resources after an error is thrown can lead to a denial-of-service.**
  - An attacker just has to keep re-creating the error condition until resources that are in use lead to a resource consumption or denial of service.
- **Keep in mind: Client applications using your server application may also inappropriately display response information or data from the error returned by your web service or application.**

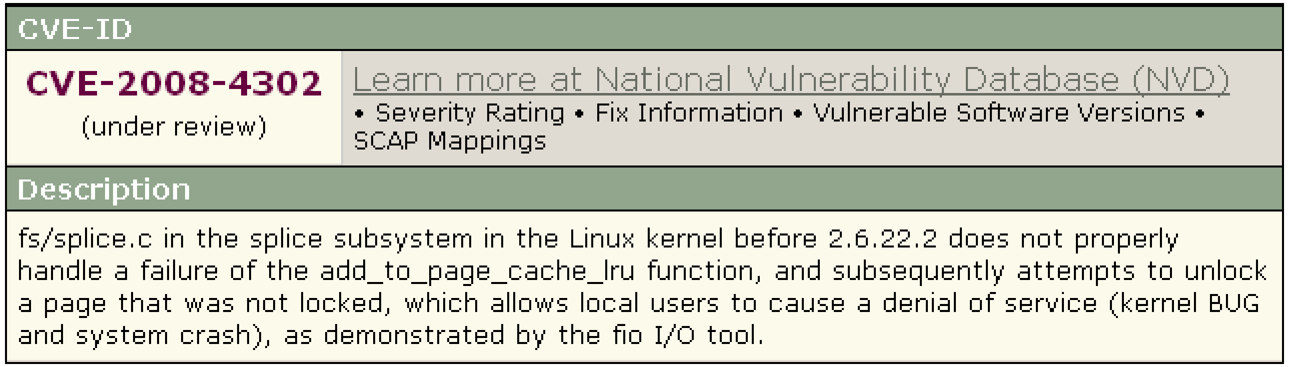
CWE-460: Improper Cleanup on Thrown Exception

- **Description: The product does not clean up its state or incorrectly cleans up its state when an exception is thrown, leading to unexpected state or control flow.**
- **Often, when functions or loops become complicated, some level of resource cleanup is needed throughout execution. Exceptions can disturb the flow of the code and prevent the necessary cleanup from happening.**

Sources:

1. CWE-460: Improper Cleanup on Thrown Exception. MITRE CWE. https://cwe.mitre.org/data/definitions/460.html. Retrieved July 20, 2021.

Real World Example – Improper Cleanup



CVE-ID	
CVE-2008-4302 (under review)	Learn more at National Vulnerability Database (NVD) • Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings

Description

fs/splice.c in the splice subsystem in the Linux kernel before 2.6.22.2 does not properly handle a failure of the add_to_page_cache_lru function, and subsequently attempts to unlock a page that was not locked, which allows local users to cause a denial of service (kernel BUG and system crash), as demonstrated by the fio I/O tool.

If function hits an error, it fails to secure a page lock.

However, the fail code path attempts to call an unlock on a page that was not locked.

Bug in the kernel causes system DoS.

CVE-2008-4302. (n.d.) Retrieved February 21, 2011, from http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4302

CVE-2008-4302

```
boolean DoStuff ()
{
   try
   {
      while (condition == true)
      {
         ThreadLock(TRUE);
         // do some stuff
         // an exception may be thrown
         ThreadLock(FALSE);
      }
   }
   catch (Exception e)
   {
      System.err.println("Something bad happened!");
      return (FAILURE);
   }
   return (SUCCESS);
}
```

If an exception is thrown while the thread is locked, then the function will return without unlocking the thread.

Logging and Audit Core Concepts

Logging and Audit Core Concepts

- **Log: A record of the events occurring within an organization's systems and networks.**
- **Audit Log: A chronological record of information system activities, including records of system accesses and operations performed in a given period.**
- **Software systems typically use log files/output to communicate important information about:**
  - Status of the application
  - Errors in the application
  - Any actions that need to be taken for maintenance and ongoing operations
- **Organizations also use log files/output in some cases:**
  - For auditing (ensuring only sanctioned activities occur by examining the audit logs/trails).
  - For monitoring .
  - For detection.
  - In some cases even for response (Example: enacting emergency response plans).

Sources:

1. NIST Glossary: Log. https://csrc.nist.gov/glossary/term/log. Retrieved July 10, 2021.
2. NIST Glossary: Audit Log. https://csrc.nist.gov/glossary/term/audit_log. Retrieved July 10, 2021.
3. This is Fine meme. Know your meme, Literally Media, 2016, https://knowyourmeme.com/memes/this-is-fine/.

CWE-532: Insertion of Sensitive Information into Log File

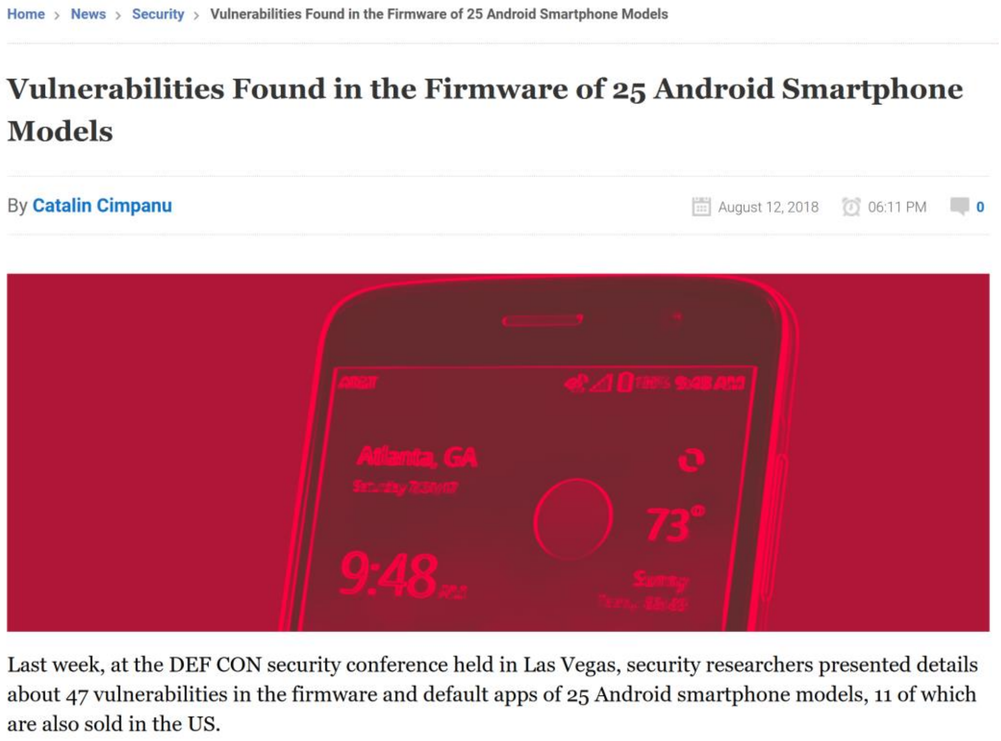
- **Description: Information written to log files can be of a sensitive nature and give valuable guidance to an attacker or expose sensitive user information.**
- **While logging all information may be helpful during development stages, it is important that logging levels be set appropriately before a product ships so that sensitive user data and system information are not accidentally exposed to potential attackers.**
- **Different log files may be produced and stored for:**
  - Server log files (e.g. server.log). This can give information on whatever application left the file. Usually this can give full path names and system information, and sometimes usernames and passwords.
  - log files that are used for debugging

Sources:

1. CWE-532: Insertion of Sensitive Information into Log File. MITRE CWE. https://cwe.mitre.org/data/definitions/532.html. Retrieved July 20, 2021.

Issues with incorrect Logging

- **Android applications notorious for leaking data in log messages viewable using Logcat tool!**



Home › News › Security › Vulnerabilities Found in the Firmware of 25 Android Smartphone Models

Vulnerabilities Found in the Firmware of 25 Android Smartphone Models

By Catalin Cimpanu August 12, 2018 06:11 PM 0

Last week, at the DEF CON security conference held in Las Vegas, security researchers presented details about 47 vulnerabilities in the firmware and default apps of 25 Android smartphone models, 11 of which are also sold in the US.

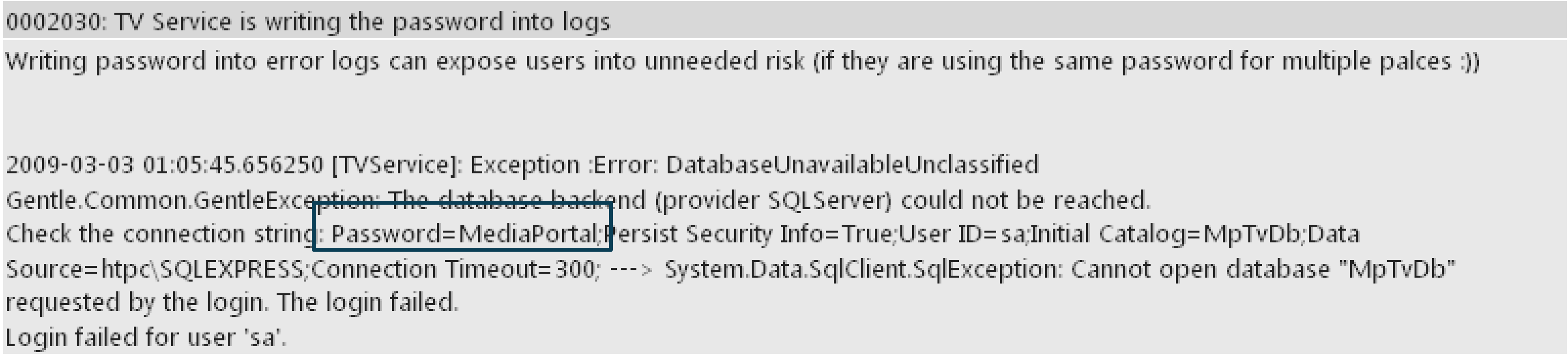
Sources:

1. Catalin Cimpanu. Vulnerabilities found in the firmware of 25 Android smartphone models. https://www.bleepingcomputer.com/news/security/vulnerabilities-found-in-the-firmware-of-25-android-smartphone-models/. August 12, 2018. Retrieved July 10, 2021.

Real World Example – Logging Sensitive Data

MediaPortal

... the open source media center



0002030: TV Service is writing the password into logs

Writing password into error logs can expose users into unneeded risk (if they are using the same password for multiple palces :))

```
2009-03-03 01:05:45.656250 [TVService]: Exception :Error: DatabaseUnavailableUnclassified
Gentle.Common.GentleException: The database backend (provider SQLServer) could not be reached.
Check the connection string: Password=MediaPortal;Persist Security Info=True;User ID=sa;Initial Catalog=MpTvDb;Data
Source=htpc\SQLEXPRESS;Connection Timeout=300; ---> System.Data.SqlClient.SqlException: Cannot open database "MpTvDb"
requested by the login. The login failed.
Login failed for user 'sa'.
```

0002030: TV Service is writing the password into logs. (2009, March, 26). Retrieved February 21, 2011, from http://mantis.team-mediaportal.com/view.php?id=2030

CWE-117: Improper Output Neutralization for Logs

- **Description: The software does not neutralize or incorrectly neutralizes output that is written to logs.**
- **This can allow an attacker to forge log entries or inject malicious content into logs.**
- **Log forging vulnerabilities occur when:**
  1. Data enters an application from an untrusted source.
  2. The data is written to an application or system log file.

Sources:

1. CWE-117: Improper Output Neutralization for Logs. MITRE CWE. https://cwe.mitre.org/data/definitions/117.html. Retrieved July 20, 2021.

CWE-93: Improper Neutralization of CRLF Sequences ('CRLF Injection')

- **Description: The software uses CRLF (carriage return line feeds) as a special element, e.g. to separate lines or records, but it does not neutralize or incorrectly neutralizes CRLF sequences from inputs.**

- **Can end up being used to tamper with logs.**

Sources:

1. CWE-93: Improper Neutralization of CRLF Sequences ('CRLF Injection'). MITRE CWE. https://cwe.mitre.org/data/definitions/93.html. Retrieved July 20, 2021.

Log/debug entries can become security vulnerabilities

- **Data destined for logs may include untrusted user data**
  - Including debugging systems – which *will* be used, since operational systems sometimes have problems
- **Attackers may intentionally create data that will create problems later, e.g.:**
  - Crash/take over logging system
  - Forge log entries
  - Create attack on later retrieval
- **Many store ASCII text**
  - In that case, encode all nonprintable chars (esp. control chars) so they’re something else, e.g., URL-encode or \ddd

Log forging example (1)

```
// Do not do this:
String val = request.getParameter("val");
try {
  int value = Integer.parseInt(val);
}
catch (NumberFormatException) {
  log.info("Failed to parse val = " + val);
}
```

- **If user submit “val” value of “twenty-one”, then this entry is logged:**

```
INFO: Failed to parse val=twenty-one
```

Log forging example (2)

- **But if attacker submits “val” value of:**

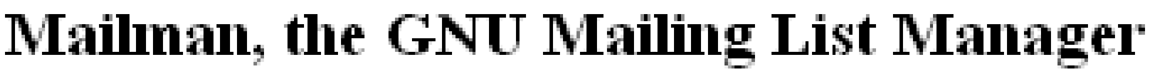
```
twenty-one%0a%0aINFO:+User+logged+out%3dbadguy
```

- **Then the log will falsely record:**

```
INFO: Failed to parse val=twenty-one
INFO: User logged out=badguy
```

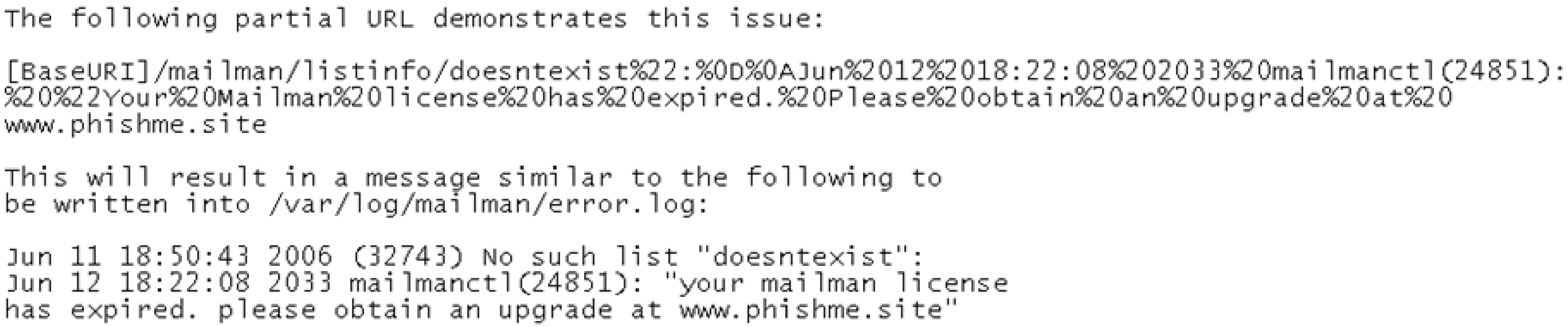
- **Possibly fooling later log viewers:**
  - badguy “couldn’t” have done later actions
  - Make it appear things okay or confuse causes
  - Frame someone else

Real World Example – Log Spoofing



Mailman, the GNU Mailing List Manager

CVE-2006-4624:



```
The following partial URL demonstrates this issue:

[BaseURI]/mailman/listinfo/doesntexist%22:%0D%0AJun%2012%2018:22:08%202033%20mailmanctl(24851):
%20%22Your%20Mailman%20license%20has%20expired.%20Please%20obtain%20an%20upgrade%20at%20
www.phishme.site

This will result in a message similar to the following to
be written into /var/log/mailman/error.log:

Jun 11 18:50:43 2006 (32743) No such list "doesntexist":
Jun 12 18:22:08 2033 mailmanctl(24851): "your mailman license
has expired. please obtain an upgrade at www.phishme.site"
```

SA0013 – public advisory. (2006, September 13). Retrieved February 21, 2011, from http://moritz-naumann.com/adv/0013/mailmanmulti/0013.txt

CWE-779: Logging of Excessive Data

- Description: The software logs too much information, making log files hard to process and possibly hindering recovery efforts or forensic analysis after an attack.
- While logging is a good practice in general, and very high levels of logging are appropriate for debugging stages of development, too much logging in a production environment might hinder a system administrator's ability to detect anomalous conditions.
- This can provide cover for an attacker while attempting to penetrate a system, clutter the audit trail for forensic analysis, or make it more difficult to debug problems in a production environment.

Sources:

1. CWE-779: Logging of Excessive Data. MITRE CWE. https://cwe.mitre.org/data/definitions/779.html. Retrieved July 20, 2021.

Real World Example – Attack on Microsoft Event Log

- One feature of the IE and Windows integration is an Internet Explorer-specific Event Log that is present on all current Windows operating systems. This IE-specific Event Log has a distinct set of permissions that enable two exploits against Windows systems: LogCrusher and OverLog

Sources:

1. Dolev Taler. The Logging Dead: Two Event Log Vulnerabilities Haunting Windows. https://www.varonis.com/blog/the-logging-dead-two-windows-event-log-vulnerabilities. May 24, 2022. Retrieved Nov 07, 2022.

Display attacks

- **Many displays simulate long-gone consoles**
  - ESCAPE + codes can change color, erase screen, sometimes even send a screen content back
  - Result: Merely *displaying* a filename or file content can cause command execution
- **Many systems store info in HTML/XML**
  - May include Javascript, etc., that is executed on display
- **Consider encoding data that users/admins might directly display later.**

Bash escape sequences

```
\033[XXXm where XXX is a series of semicolon-separated parameters
print("\033[39m\\033[49m                  - Reset color")
print("\\033[2K                           - Clear Line")
print("\\033[21m                          - Bold off")
```

See link below for more.

Sources:

1. Richard. List of ANSI color escape sequences. https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences. Oct 19, 2015. Retrieved Nov 08, 2023.

Next time …

- **Error Handling and Logging related issues - Defenses**