

# 25 Most Dangerous Software Weaknesses

Pete Freitag, Foundeo Inc.

*foundeo*  
inc.

# About Me

## Pete Freitag

*foundeo*  
inc.

- 25+ Years ColdFusion Experience
- Company: Foundeo Inc.
  - Products: FuseGuard, HackMyCF, Fixinator
  - Consulting: Code Reviews, Server Review, CFML Security Training
- You might also know me from:
  - Lockdown Guides CF9 - CF2025
  - [CFDocs.org](http://CFDocs.org), [cfscript.me](http://cfscript.me), [cfbreak.com](http://cfbreak.com)
  - blog: [petefreitag.com](http://petefreitag.com)
  - twitter/github: @pfreitag

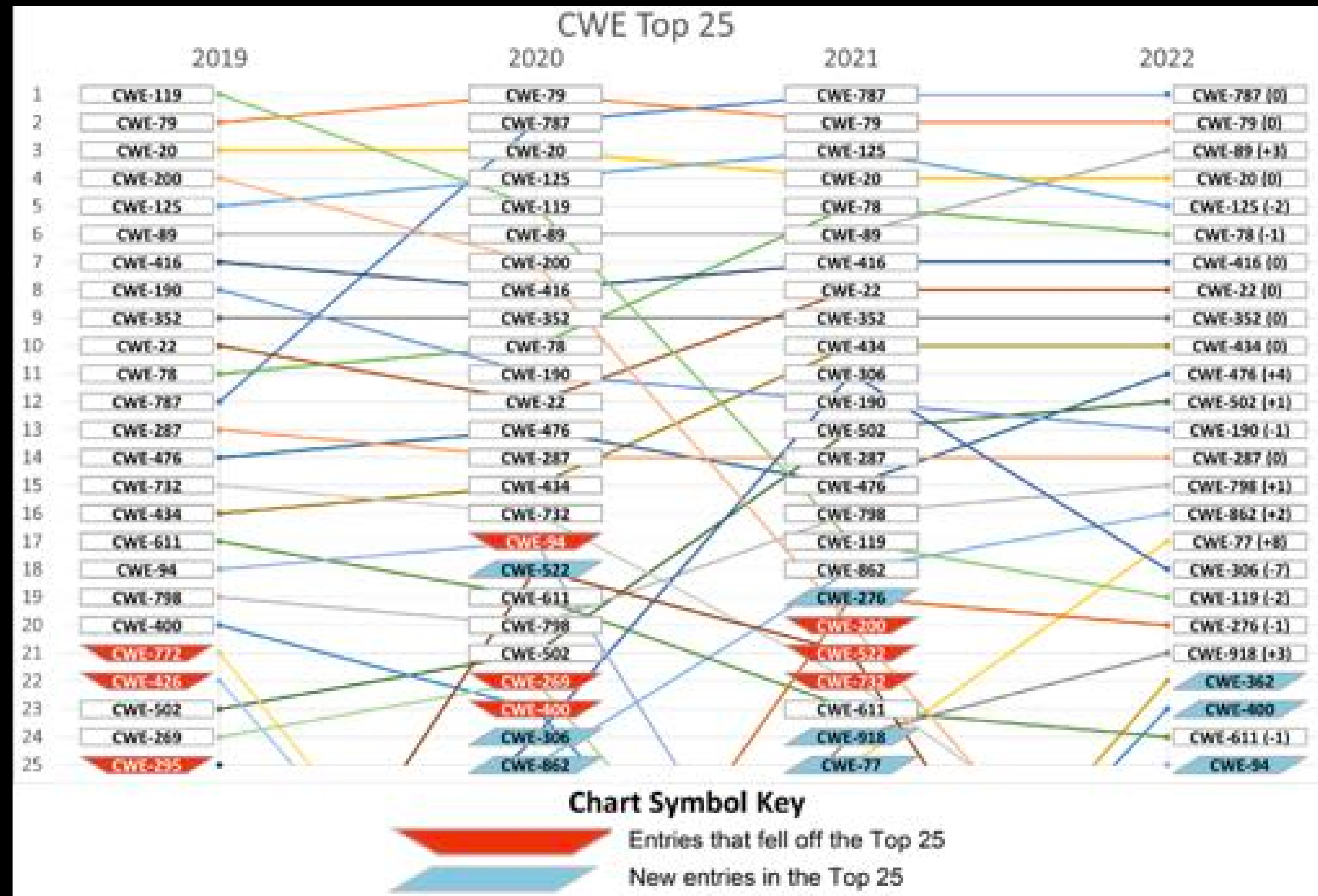
# CWE Top 25



- CVE's published between June 2023 - June 2024
- 31,770 CVE's!
- Ranking Factors:
  - Frequency
  - Severity
  - Danger

Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2023
1	<a href="#">CWE-79</a>	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	56.92	3	+1
2	<a href="#">CWE-787</a>	Out-of-bounds Write	45.20	18	-1
3	<a href="#">CWE-89</a>	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	35.88	4	0
4	<a href="#">CWE-352</a>	Cross-Site Request Forgery (CSRF)	19.57	0	+5
5	<a href="#">CWE-22</a>	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	12.74	4	+3
6	<a href="#">CWE-125</a>	Out-of-bounds Read	11.42	3	+1
7	<a href="#">CWE-78</a>	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.30	5	-2
8	<a href="#">CWE-416</a>	Use After Free	10.19	5	-4
9	<a href="#">CWE-862</a>	Missing Authorization	10.11	0	+2
10	<a href="#">CWE-434</a>	Unrestricted Upload of File with Dangerous Type	10.03	0	0
11	<a href="#">CWE-94</a>	Improper Control of Generation of Code ('Code Injection')	7.13	7	+12
12	<a href="#">CWE-20</a>	Improper Input Validation	6.78	1	-6
13	<a href="#">CWE-77</a>	Improper Neutralization of Special Elements used in a Command ('Command Injection')	6.74	4	+3
14	<a href="#">CWE-287</a>	Improper Authentication	5.94	4	-1
15	<a href="#">CWE-269</a>	Improper Privilege Management	5.22	0	+7
16	<a href="#">CWE-502</a>	Deserialization of Untrusted Data	5.07	5	-1
17	<a href="#">CWE-200</a>	Exposure of Sensitive Information to an Unauthorized Actor	5.07	0	+13
18	<a href="#">CWE-863</a>	Incorrect Authorization	4.05	2	+6
19	<a href="#">CWE-918</a>	Server-Side Request Forgery (SSRF)	4.05	2	0
20	<a href="#">CWE-119</a>	Improper Restriction of Operations within the Bounds of a Memory Buffer	3.69	2	-3
21	<a href="#">CWE-476</a>	NULL Pointer Dereference	3.58	0	-9
22	<a href="#">CWE-798</a>	Use of Hard-coded Credentials	3.46	2	-4
23	<a href="#">CWE-190</a>	Integer Overflow or Wraparound	3.37	3	-9
24	<a href="#">CWE-400</a>	Uncontrolled Resource Consumption	3.23	0	+13
25	<a href="#">CWE-306</a>	Missing Authentication for Critical Function	2.73	5	-5

# Trends of the CWE Top 25



Rank	ID	Name	Score	CVEs in KEV	Rank Change vs. 2023
26	<a href="#">CWE-770</a>	Allocation of Resources Without Limits or Throttling	2.65	0	+3
27	<a href="#">CWE-668</a>	Exposure of Resource to Wrong Sphere	2.56	0	+13
28	<a href="#">CWE-74</a>	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	2.10	0	+19
29	<a href="#">CWE-427</a>	Uncontrolled Search Path Element	2.08	0	-2
30	<a href="#">CWE-639</a>	Authorization Bypass Through User-Controlled Key	2.05	0	+8
31	<a href="#">CWE-532</a>	Insertion of Sensitive Information into Log File	1.99	0	+14
32	<a href="#">CWE-732</a>	Incorrect Permission Assignment for Critical Resource	1.94	0	-1
33	<a href="#">CWE-601</a>	URL Redirection to Untrusted Site ('Open Redirect')	1.85	0	-1
34	<a href="#">CWE-362</a>	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	1.75	2	-13
35	<a href="#">CWE-522</a>	Insufficiently Protected Credentials	1.71	0	0
36	<a href="#">CWE-276</a>	Incorrect Default Permissions	1.68	0	-11
37	<a href="#">CWE-203</a>	Observable Discrepancy	1.61	0	+14
38	<a href="#">CWE-59</a>	Improper Link Resolution Before File Access ('Link Following')	1.40	0	+1
39	<a href="#">CWE-843</a>	Access of Resource Using Incompatible Type ('Type Confusion')	1.38	6	+7
40	<a href="#">CWE-312</a>	Cleartext Storage of Sensitive Information	1.37	0	+3

# "On the Cusp" 2024 List

# Over 900 CWE's

*foundeo*  
inc.

- Some are very similar (as we'll see)
- Some are more specific cases of others (child of)

# #25 - Missing Authentication for Critical Function

## CWE-306

- Authentication = User is who they attest to be
- Authorization = User has permission to perform action
- Without Authentication you cannot have Authorization
- Tips: Check for authentication in Application.cfc/cfm or use a framework
- Check your app to make sure it requires authentication where it should. For bonus points automate this check in unit / integration tests.

# Do This

```
component {  
  
    function onRequestStart() {  
        checkAuth();  
    }  
  
}
```

- Use Application.cfc onRequestStart or onRequest
- Frameworks
- Application.cfm

# Avoid This

```
<cfinclude template="auth.cfm">
```

 On top of every file



# #24 Uncontrolled Resource Consumption

## CWE-400

- Denial of Service
  - Check Queries
    - Use LIMIT / TOP on Queries
  - Check Loops
    - Add a maximum iteration when looping over an untrusted input
- HashDOS - make sure post parameter limit is not set too high in CF Admin

# #23 - Integer Overflow or Wraparound

## CWE-190

- Max value of a 32 bit unsigned integer is 4,294,967,295
- What happens when you add 1?
  - In MySQL < 5.5.5 it silently wraps around to 0

# #22 - Use of Hard-coded Credentials

CWE-798

- API Keys / Passwords in Code
- Embedded / Hard Coded Certificates

# #22 - Hard Coded Credentials



## Avoiding Hard Coded Credentials

- Environment Variables
- Docker Secrets
- Secure Key Store Services:
  - Self Hosted: Hashicorp Vault
  - AWS: EC2 Parameter Store, KMS, Secrets Manager
  - Azure: Key Vault
  - GCP: Key Vault, Secret Manager

**#21 - NULL Pointer  
Dereference  
CWE-476**

**#20 Improper Restriction of Operations  
within the Bounds of a Memory Buffer  
CWE-119**

**Thanks Java**

“Java is said to be **memory-safe** because its runtime error detection checks array bounds and pointer dereferences.”

# #19 - Server-Side Request Forgery (SSRF)

CWE-918

- SSRF Happens When your server makes a HTTP request to an arbitrary URL
- Can allow attacker to hit other http services behind the firewall (dbs, caches)
- Cloud Metadata APIs (eg: 169.254.169.254) can leak access keys or other sensitive info:
  - Tip: on AWS Disable IMDSv1 and use IMDSv2 instead

# #19 - SSRF

## Some Functions / Tags That Can Request a URL

- cfhttp
- PDF: cfdocument / cfhtmltopdf (within HTML: img, iframe, etc)
- Images: isImageFile
- XML: XmlParse, XmlSearch, XmlValidate
- Additional List: <https://hoyahaxa.blogspot.com/2021/04/ssrf-in-coldfusioncfml-tags-and.html>

# #18 - Incorrect Authorization

CWE-863



- Authorization checks exist, but are not functioning properly

```
//make sure they are an admin
if (isLoggedIn() || isAdmin()) {
    doAdiminStuff();
}
```



# #17 - Exposure of Sensitive Info to Unauthorized Actor

## CWE-200

- Disclosing which email addresses are valid on login requests.
- Disclosing Error Details
  - Use `onError` in `Application.cfc`, set global error handler

# #16 - Deserialization of Untrusted Data

## CWE-502

- Java Deserialization Vulnerabilities
  - Has the ability to cause remote code execution if malicious content is added to the serialized class.
- Avoid Untrusted Input to ColdFusion's Deserialize Function
- Block the flash remoting endpoints on older CF
- JSON Deserialization - Consider Validating JSON with a JSON schema first

# #15 - Improper Privilege Management

CWE-269

- Similar to #18 (CWE-863) Incorrect Authorization

# #14 - Improper Authentication

CWE-287



- So many ways Authentication can go wrong...
  - Weak Passwords, Credential Stuffing, Weak Session Cookie Config
- Use SSO
  - Most orgs now have the ability a SSO provider, either through ActiveDirectory, Google Apps, Okta, etc.
  - You can use SAML to integrate with the identity provider in your ColdFusion Apps. SAML Features added to CF2021

# #13 - Command Injection

## CWE-77 Improper Neutralization of Special Elements used in a Command

- Take care when using `cfexecute`, or other APIs that may wrap a native command

```
<cfexecute name="c:\bin\tool.exe" arguments="-n #url.n#">
```

# #12 - Improper Input Validation

## CWE-20

- This is a catch all CWE
  - Almost all vulnerabilities are caused by failing to validate an input!
  - TLDR: Add validation, improve security

# #11 - Improper Control of Generation of Code

## CWE-94 Code Injection

- A few different ways this can happen in CFML, most common:
  - Evaluate
  - IIF
  - cfinclude

# #11 Fixing RCE

## Replace IIF with Ternary Operator

```
<cfset greet = iif( len(name), de("Hi #name#"), de("Hi") )>
```



```
<cfset greet = ( len(name) ) ? "Hi #name#" : "Hi">
```



# #11 Fixing RCE

## Fixing Evaluate

```
evaluate("url.#name#")
```



```
url[name]
```

Rid your code of evaluate() - terrible for both performance and security

# #11 Fixing RCE

## Fixing Evaluate

```
#evaluate("x+y")#
```



```
#x+y#
```

Rid your code of evaluate() - terrible for both performance and security

# #11 RCE

## Good News / Bad News



- Good News
  - Easy to find
  - Easy to fix
- Bad News
  - Very Dangerous
  - Might have a lot if your code was written early 2000's

# #10 - Unsafe File Uploads

## CWE-434 Unrestricted Upload of File with Dangerous Type

- Regularly review all your file upload code, and make sure that it:
  - Always checks the file extensions of uploaded files against a list of allowed extensions. Use the `allowedExtensions` attribute of `cfile`.
  - Does not upload directly under the web root (at least before validation)
  - Don't rely on mime type checks alone, they can be bypassed!
  - Set `this.blockedExtForFileUpload` to full list of executable extensions.

# #9 - Missing Authorization



## CWE-862

- Does your code check that the user is allowed to perform the requested function?
- IDOR: Insecure Direct Object Reference: `document.cfm?id=123`
- Sounds simple but these types of issues fall under the radar, because “it works”
  - Need to test that it “doesn’t work” for X role
  - No easy way to do this, but you can write tests

# #8 - Path Traversal

## CWE-22 Improper Limitation of a Pathname to a Restricted Directory

- Path Traversal can happen whenever you construct a file path with unsafe variables.

- Example:

```
<cfinclude template="#url.name#">
```

**#8 - Use After Free**  
**CWE-416**

**Thanks Java**

# #7 - OS Command Injection

**CWE-78 - Improper Neutralization of Special Elements used in an OS Command**

- Similar / Child of to #13, CWE-77
  - Command vs OS Command
  - Same Protections Apply



**#5 - Out-of-bounds**

**Read**

**CWE-125**

**Thanks Java**

# #4 - Cross-Site Request Forgery (CSRF)

CWE-352

- Causing a request to be made by an **authenticated** and **authorized** user's browser to perform an unwanted action.

# #9 - CSRF

## Best Example



- Netflix in 2006 - Remember when they rented DVDs?
  - To Add a Movie to your Queue:
    - Request to: <http://www.netflix.com/AddToQueue>
    - Pass a movie id: **movieid=70011204**

# #9 CSRF

## Netflix Example



Step 1: Create a Web Page With The Following img tag:

```

```

Step 2: Get People to Visit the Page

Step 3: Millions of people added *Sponge Bob Square Pants the Movie* to their Queue

# #4 CSRF

## Fixing CSRF



- SameSite Cookies
- Check HTTP Method (eg: require POST)
- CAPTCHAs - Helpful but causes usability issues / AI
- Inspect Sec-Fetch and Origin Request Headers
- Use a CSRF Token
  - CFML Functions: `CSRFGenerateToken()` and `CSRFVerifyToken()`

# #4 CSRF

## Fixing CSRF Sec-Fetch Headers (Modern Browsers)

```

```

```
GET /AddToQueue?movieid=70011204  
Host: www.netflix.com  
Sec-Fetch-Dest: image  
Sec-Fetch-Site: cross-site  
Sec-Fetch-Mode: no-cors
```

HTTP Request



## #3 - Improper Neutralization of Special Elements used in an SQL Command

### CWE-89 SQL Injection

- Classic Example:

```
<cfquery>
  SELECT story
  FROM news
  WHERE id = #url.id#
</cfquery>
```

# #3 SQL Injection

## Fixing SQL Injection



- Use cfqueryparam

```
<cfquery>
  SELECT story
  FROM news
  WHERE id = <cfqueryparam value="#url.id#">
</cfquery>
```



# #3 SQL Injection

With queryExecute

```
queryExecute("SELECT story  
FROM news  
WHERE id = #url.id#");
```



```
queryExecute("SELECT story  
FROM news  
WHERE id = :id", {id=url.id} );
```

# #3 - SQL Injection

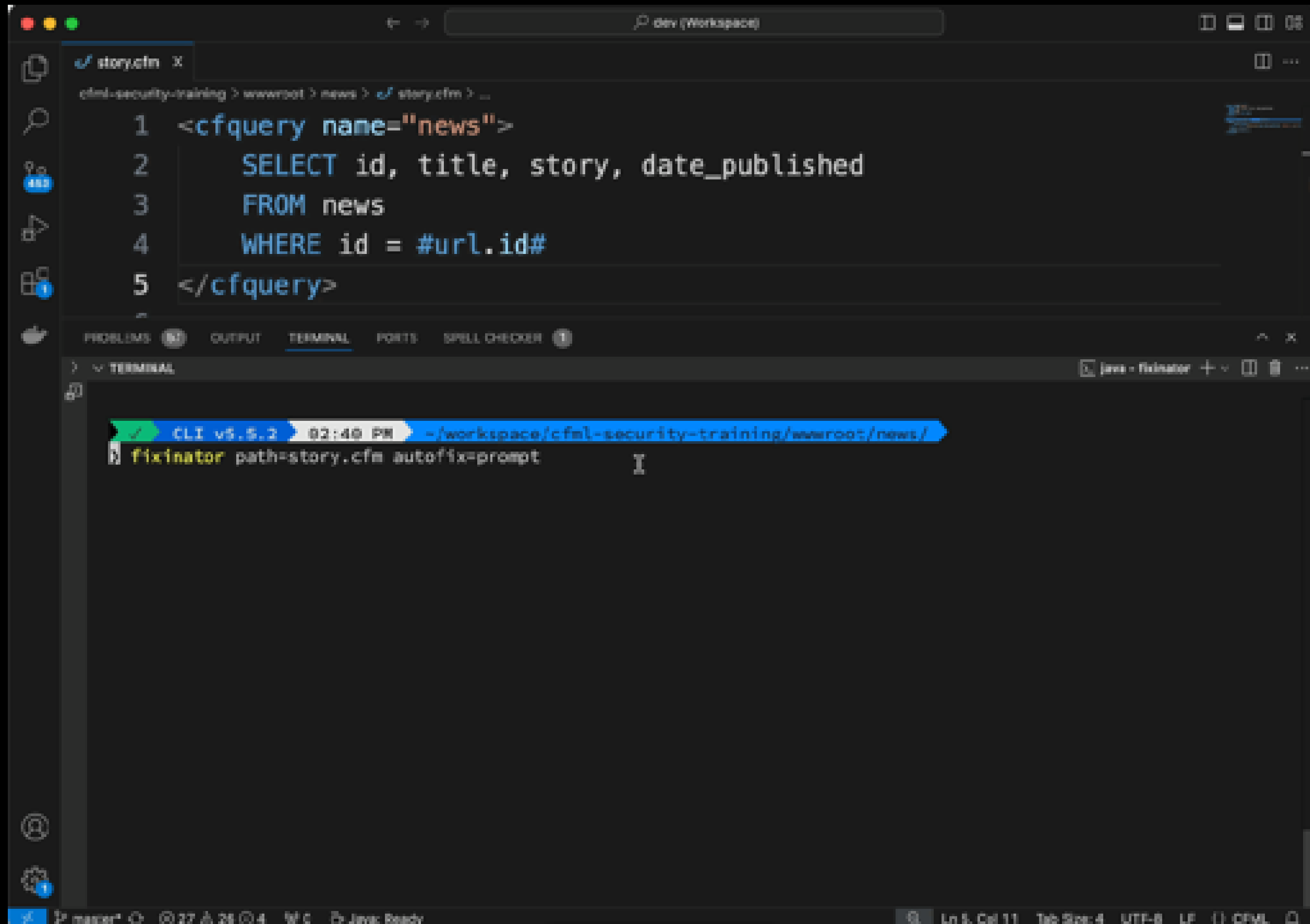
## When Parameters Don't Work

- Places that parameters *may (depending on DB)* not work:
  - ORDER BY clause
  - SELECT TOP n
  - LIMIT / OFFSET
- Validate!
- Use SELECT TOP #int(url.n)#
- Use cfqueryparam whenever you can

# #3 - SQL Injection

## Fixing SQL Injection

- Fixinator can scan your code and fix certain vulnerabilities



```
1 <cfquery name="news">
2     SELECT id, title, story, date_published
3     FROM news
4     WHERE id = #url.id#
5 </cfquery>
```

```
CLI v5.5.2 02:40 PM -/workspace/cfml-security-training/wwwroot/news/
fixinator path=story.cfm autofix=prompt
```



**#2 - Out-of-bounds**

**Write**

**CWE-787**

**Thanks Java**

**#1**

**Any Guesses?**

# #1 XSS

**Improper Neutralization of Input During Web Page Generation**

**CWE-79 Cross-site Scripting / XSS**

# #1 - XSS - Vulnerable Code Example



```
<cfoutput>Hello #url.name#</cfquery>
```

# #1 - Fixing XSS

## Encoder Methods



```
<cfoutput>Hello #encodeForHTML(url.name)#</cfquery>
```

```
<cfoutput encodefor="html">Hello #url.name#</cfquery>
```



# #1 Fixing XSS

## Picking the correct encoder

Context	Method
HTML	<code>encodeForHTML(variable)</code>
HTML Attribute	<code>encodeForHTMLAttribute(variable)</code>
JavaScript	<code>encodeForJavaScript(variable)</code>
CSS	<code>encodeForCSS(variable)</code>
URL	<code>encodeForURL(variable)</code>

# Learn More



- ColdFusion Security Guide
  - <https://foundeo.com/security/guide/>
- Ask Me
- Resources: OWASP, CWE Site

# Thank You!



*foundeo*  
inc.



pete@foundeo.com

Weekly CFML Community Newsletter: [cfbreak.com](http://cfbreak.com)