

Securing ColdFusion Applications

Pete Freitag, Foundeo Inc.

About Pete

- Guy who wrote the ColdFusion Lockdown Guides CF9-CF2021
- My Company: Foundeo Inc.
 - **Consulting:** Code Reviews, Server Reviews, Development
 - **FuseGuard:** Web App Firewall for CFML
 - **HackMyCF:** Server Security Scanner
 - **Fixinator:** Code Security Scanner
- Blog (petefreitag.com), Twitter ([@pfreitag](https://twitter.com/pfreitag)), #CFML Slack
 - I will post these slides on my blog
- Using CFML since late 90s

How is 2021 Going?

- SolarWinds: (end of 2020) at least 200 companies, gov orgs impacted
- Microsoft Exchange Hack: at least 30,000 US Companies [[link](#)]
- Colonial Pipeline: gas pipeline forced to shut down and causing shortages.

Takeaways

- We're all impacted
- Even the biggest, wealthiest, smartest companies still have security vulnerabilities.
- Absolute or Perfect Security does not exist
 - And probably never will!
- We can't ignore it
- Probably a good time to talk to stakeholders about improving security

What we know

Laying out the facts

- Security breaches are skyrocketing
- More vulnerabilities are being discovered in the software / hardware we use
 - The number of CVEs published nearly tripled from 2015 (6k) vs 2020 (18k)
 - Staying up to date is hard
- Security is hard
- Humans consistently fail

“Assume Breach”

Does this change how you would build / deploy your applications?

Can we easily redeploy?

Are we using principal of least privilege to minimize the impact of an attack?

Can we easily revoke access?

Am I writing code with security in mind?

“Assume Breach” != “Assume Beach”



Don't go it alone

- Automate with CI / CD
- Leverage security tools
- Unit / Integration Tests
- Get support from colleagues, boss





Top 10 CF Security Risks

Because everyone loves top 10 lists!

CF Security Top 10

- 10 - Old Code
- 09 - Failure to Delegate
- 08 - Security Configuration
- 07 - Injection
- 06 - XSS
- 05 - Authentication / Authorization
- 04 - Remote Code Execution
- 03 - SQL Injection
- 02 - Insecure File Uploads
- 01 - Unpatched Known Vulnerabilities



Other Top 10 Lists

- OWASP Top 10 Web Application Security Risks:
 - <https://owasp.org/www-project-top-ten/>
- OWASP Top 10 Proactive Controls
 - <https://owasp.org/www-project-proactive-controls/>

Tip: There are more than 10 types of security risks, so these lists are just starting points

10 : Old Code

You Might Be Using...

“Home Made Version Control”

- index_2.cfm
- index.old.cfm
- index-backup.cfm
- index-2007-03-04.cfm
- index-copy.cfm
- folder_backup2009/

Version Control

- Those backup folders and files are probably full of vulnerabilities.
- Version Control Server keeps backups of all your code and all changes you have ever made to it.
- Sync server source code with version control.
 - Identify if someone changed something on the server.

Version Control

- Spend some time to identify unused code.
- Delete it!
- Version control has your back, if you deleted something you can recover it from the repository.

“There are Lots of Fads in Software Development, Version control is not one of them.”

Ways To Identify OBSOLETE Code

- Unix / Linux / Mac

```
$> find /wwwroot/ -mtime +365
```

- Windows

```
C:\>forfiles -p "C:\web" -s -m *.* /D -365 /C  
"cmd /c echo @path"
```

Ways To Identify OBSOLETE Code

- Unix / Linux / Mac

```
$> find /wwwroot/ -atime +365
```

- The `atime` (last accessed time) timestamp may be disabled on your server for performance (if drive was mounted with `noatime` flag).
- RHEL mounts drives with the `relatime` flag by default, which is not real time but may be sufficient for these purposes.

09 : Failure to Delegate

Delegate Risky Areas

When possible

- Payment Processing
 - Delegate to payment gateway hosted payment page
 - JS SDK (Braintree, Stripe, etc), may lessen PCI compliance requirements
- Authentication
 - Enterprise: Active Directory, LDAP, etc
 - Social: Google, Apple, Facebook, Twitter, MS, etc.
- Other Areas: Encryption, Secrets, Key Management

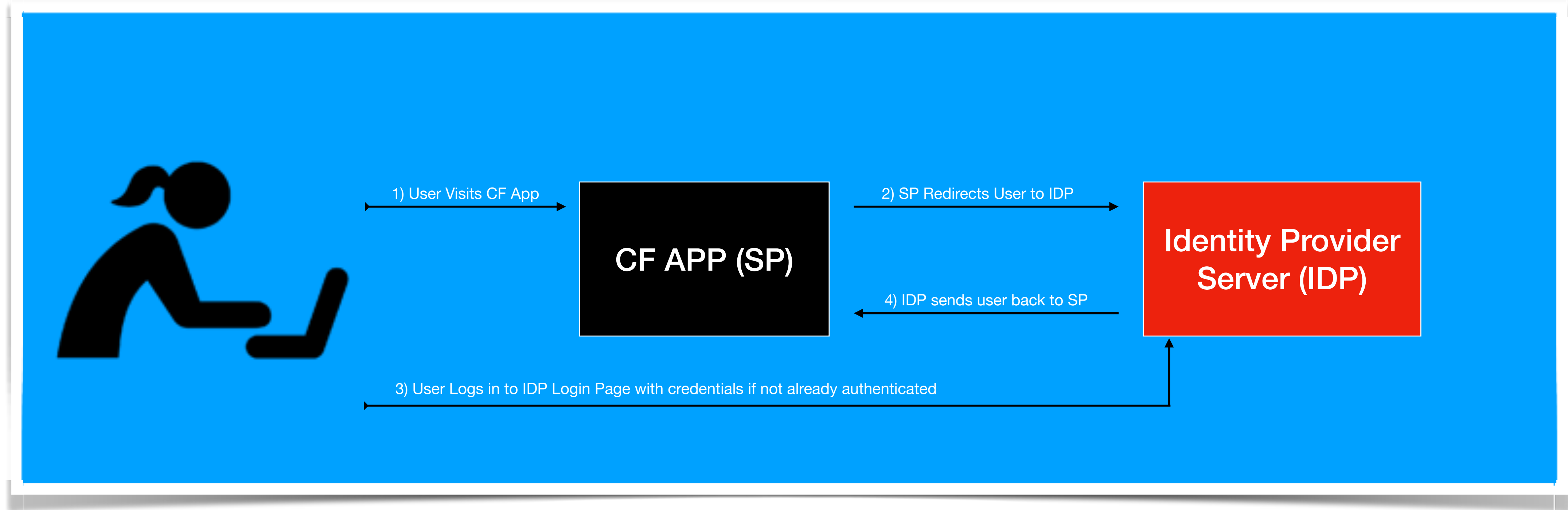
Authentication

Single Sign On (SSO)

SAML, OAUTH, etc

- DELETE your authentication code!
 - DIY Auth code notoriously riddled with security weaknesses
 - Eliminate storing passwords, forgot passwords, etc.
- Users will notice and will like it.
 - Users do not need separate passwords for your app
- Both a Security Improvement and User Experience Improvement

How does SAML work?



SSO Implementation

- ColdFusion 2021 SAML Integration
 - `InitSAMLAuthRequest()`, `ProcessSAMLResponse()`,
`InitSAMLLogoutRequest()`
 - <https://helpx.adobe.com/coldfusion/using/saml-coldfusion.ug.html>
- Java Libraries such as java-saml, or opensaml
- Web Server Extensions act as a facade, provide authenticated user id as CGI variable.

Secrets / Key Management

Avoid using source code for storing secrets

- Secret / Key Management APIs / Services
 - Provides auditing, logging, rotation
 - Built by world class security experts, let's delegate to them!
- Cloud Solutions
 - AWS: KMS, Secrets Manager, EC2 Parameter Store
 - Azure: Key Vault
- On Premise
 - Hashicorp Vault

08: Secure Configuration

Secure Configuration

“Assume Breach” driven

- Use Principal of Least Privilege
- Follow security best practices such as:
 - Lockdown Guides for ColdFusion
 - STIGs
 - CIS Benchmark Guides

Lockdown ColdFusion

- What user is the JVM running as?
 - If your CFML server is running as SYSTEM or root?
 - Assume Breach - limit impact if breached.
- What permission does the user have?
 - If CFML server user only has readonly access to web root and CFML server install directory then less harm can be done (easily).
 - Does CFML server need full write access to web root? or just one or two directories?

Lockdown Database

- Limit what the datasource DB user can do
 - Is it a readonly database or table, then it only needs SELECT permission.
 - Don't use the root, DBA, SA account!
- Assume Breach, Principal of Least Privilege

07 : Injection Attacks

PDF Injection

Risks: SSRF (Server-Side Request Forgery), RCE

Applies to: cfdocument, cfhtmltopdf

```
<cfdocument>
  <p>Hi #url.name#</p>
</cfdocument>
```

Vulnerable Example

pdf.cfm?name=

```
<cfdocument>
  <p>Hi #encodeForHTML(url.name)#</p>
</cfdocument>
```

Safe Example

Scope Injection

Taking advantage of scope cascading in CFML

- Suppose you have a variable **session.userID**
 - Now assume the variable is not yet defined in the `session` scope (user not yet logged in)
 - If I request `/admin/page.cfm?session.userID=1`
 - CF will check `session` scope first, but if not defined will check all other scopes: `url`, `form`, `cookie`, etc.
 - Here `url.session.userID` is defined and would be used
- Demo

Preventing Scope Injection

- Can happen with any variable that is not defined, not just `session`
- Ensure that variables are properly defined
 - Set defaults in `onSessionStart`, `onApplicationStart`
 - Use `structKeyExists`, or `session.keyExists("userID")`
- `Application.cfc` setting `this.searchImplicitScopes = false`
- Learn more: <https://www.petefreitag.com/item/834.cfm>

Other Injection Attacks

- XML Entity Injection
- XPath Injection
- LDAP Injection
- Command Injection
- Comment Injection
- See OWASP for details.

06 : XSS

XSS or Cross Site Scripting

```
<cfoutput>
  <p>Hi #url.name#</p>
</cfoutput>
```

Vulnerable Example

example.cfm?name=<script>doSomething()</script>

```
<cfoutput>
  <p>Hi #encodeForHTML(url.name)#</p>
</cfoutput>
```

Safe Example

Mitigating XSS

- Use encoder functions when possible:
 - Functions: `encodeForHTML`, `encodeForHTMLAttribute`, `encodeForJavaScript`, `encodeForCSS`, etc.
 - Or use `<cfoutput encodefor="html">#url.name#</cfoutput>`
- When encoder functions can't be used, use: `isSafeHTML`, or `getSafeHTML`
- Add Content-Security-Policy headers

XSS

A pesky problem

- XSS vulnerabilities are among the most prevalent
- Time consuming to fix on very large legacy applications.

05 : Authentication / Authorization

Authentication

Proving you are who you say you are

- See #9, if possible delegate this to an identity provider (SAML, oauth, etc).
- Otherwise you'll need to consider:
 - Password storage (adaptive one way functions)
 - Credential stuffing attacks, rate limiting
 - Password reset flows
 - 2FA Implementations
 - Timing Attacks

Authorization

Does this user have permission to perform an action

- Most common problem: Missing Authorization
 - Test by attempting to access resources your user should not have access to.
 - Forgot to check if user has permission
- Authorization issues are hard to detect with automated tools such as code scanners.
 - Create integration tests that check to make sure users can't access resources they shouldn't be able to.

Authentication and Authorization

For more info

- My ColdFusion Security Training Course (foundeo.com)
- OWASP Authentication Cheat Sheet and Authorization Cheat Sheet

04 : Remote Code Execution

Evaluate

```
<cfset day_1 = "Monday">
<cfset day_2 = "Tuesday">
<cfset day_3 = "Wednesday">

<cfoutput>
    #Evaluate("day_#url.day#")#
</cfoutput>
```

Evaluate Example

Fixing Legacy Evaluate Example

```
<cfset day_1 = "Wednesday">
<cfset day_2 = "Thursday">
<cfset day_3 = "Friday">

<cfoutput>
    #variables["day_#url.day#"]#
</cfoutput>
```

Fixing Evaluate Issues

- Search Code for "Evaluate"
- In most cases you should not need to use Evaluate at all, use brackets.
 - If the variable is a query you may need to use `queryName[row][columnName]` notation.
 - Not all cases are super simple to fix, but most are.
- Remove all Evaluate calls from your code.
- Also look at PrecisionEvaluate

Do Any other Functions Evaluate Dynamically?



```
Hi #iif(len(url.name) EQ 0, de("Friend"), de(url.name))#
```

The second and third arguments are evaluated dynamically!

IIF Example

Fixing IIF

```
Hi #(!len(url.name)) ? "Friend" : url.name#
```

ELVIS OPERATOR (CF11+)

```
Hi #url.name?: "Friend" #
```

Elvis Operator tests to see if url.name is defined / not null

03 - SQL Injection

Ye Olde Classic SQL Injection

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

news.cfm?id=0;delete+from+news

Fixing SQL Injection

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

Script Based

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

Vulnerable

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

Not Vulnerable

Finding SQL Injection

- Search codebase for `cfquery`, `queryExecute`, `ormExecuteQuery`, `new Query()`
- Use Static Code Analyzer (CFBuilder / CF Enterprise 2016+)
- Fixinator can find, and fix them for you (quick demo)
- Stop and fix whenever you see one

02 : File Issues

Path Traversal

Path Traversal

```
<cfinclude template="path/#fileName#">
```

Path Traversal example

Fixing Path Traversals

- Avoid variables in paths
 - If you really need to use a variable strip out everything except a-z0-9
- Use the CF11+ Application.cfc setting `this.compileExtForInclude` setting.

File Uploads

3 CORE Rules

File Uploads



Never trust a MIME!

File Uploads Rule #1

- CF10 added strict attribute to cfile action=upload
 - Instead of validating the MIME type that the browser sends it validates the the file content (eg fileGetMimeType()).
 - Can we still get around this?

FILE UPLOADS Rule #2

- Always validate the **file extension**
 - CF10+ allows you to specify file extensions in **accept** attribute
 - You can also specify a file path in the destination with a hard coded extension

```
<cfile action="upload" accept=".png,.jpg">
```

```
<cfile action="upload" destination="/path/to/#int(some.id)#.pdf">
```

FILE UPLOADS Rule #3

- The upload **destination** must be outside of the web root

```
<cffile action="upload" destination="#expandPath("./uploads/")#">
```

```
<cffile action="upload" destination="#getTempDirectory()#">
```

File Uploads

- Inspect file content: `fileGetMimeType`, `isImageFile`, `isPDFFile`, etc
- Upload to static content server (s3 for example)
 - Upload directly to s3: <https://www.petefreitag.com/item/833.cfm>
- Make sure directory serving uploaded files cannot serve dynamic content.
- File Extension Allow List on Web Server (eg IIS Request Filtering)
- `secureupload.cfc`: <https://github.com/foundeo/cfml-security/>

New File Upload Features

- New in CF2018 update 3, CF2016 update 10 & CF11 update 18
- Application.cfc setting: `this.blockedExtForFileUpload`
 - Comma separated list
 - Set to "*" to block all (empty string allows all)
- Set server wide in ColdFusion Administrator

Finding FILE ACCESS ISSUES

- As you can see any code that accesses the file system can potentially be exploited.
- Review all function calls / tags that access file system
 - cfile, cfdocument, cfinclude, cfmodule, cfspreadsheet
 - fileRead, fileWrite, fileOpen, etc

01 : Unpatched Known Vulnerabilities

Patch That Server

- Use ColdFusion 2018* or greater.
 - CF2016 ended Feb 2021, CF11 Core Support Ended Apr 2019, CF10 Ended May 2017, CF9 have had no security patches for many many years.
- Windows 2008 (EOL 2015)
- Java 8+, Java 7 (EOL 2015), Java 6 (EOL 2013)

* Core Support for CF2018 Ends July 2023



Patch That Server

- Multiple Denial of Service Vulnerabilities in old versions of Java
- Path Traversal via Null Byte injection JVM (< 1.7.0_40)
- CRLF Injection (CF10+)
- File Uploads “somewhat” more secure (CF10+)
- TLS / SSL Protocol Implementations
- Java 8 Not supported on CF9 and below
- Use HackMyCF to help keep you on top of all this

**“Nearly 60% of Breaches due to
Un-patched Vulnerability”**

ServiceNow Survey

Equifax Breach

- The Equifax breach was caused by using a vulnerable java library: Apache Struts
 - Struts was patched on March 7th 2017
 - Equifax discovered breach on July 29th 2017
 - Equifax applied the patch on July 30th, 2017

One year after the Equifax breach:

"As many as 10,801 organizations—including 57% of the Fortune Global 100—have downloaded known-to-be-vulnerable versions of Apache Struts"

Source: <http://fortune.com/2018/05/07/security-equifax-vulnerability-download/>

UPDATE Known Vulnerable Components

- **Fixinator** - (CFML, JS, JAR) Looks for known vulnerable CFML libraries (eg FCKeditor file upload vulnerability, old custom tags, etc) [commercial]
- **OWASP Dependency Check** - (Java, C, Ruby, Python, NodeJS)
- **RetireJS** - (JS)
- **npm audit** - (JS)

Continuous Security

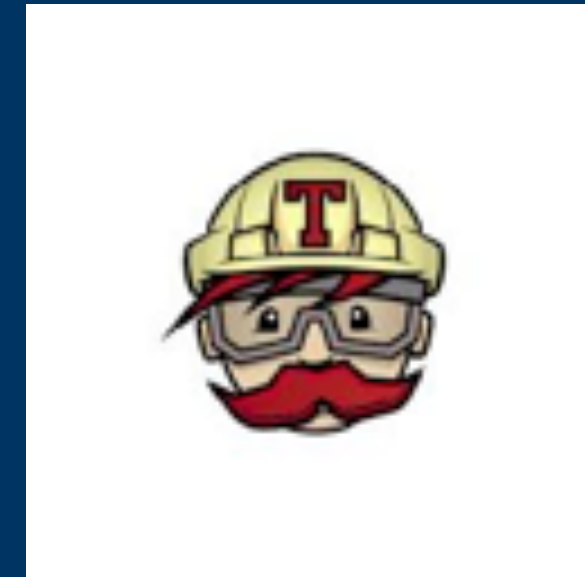
You need Version Control



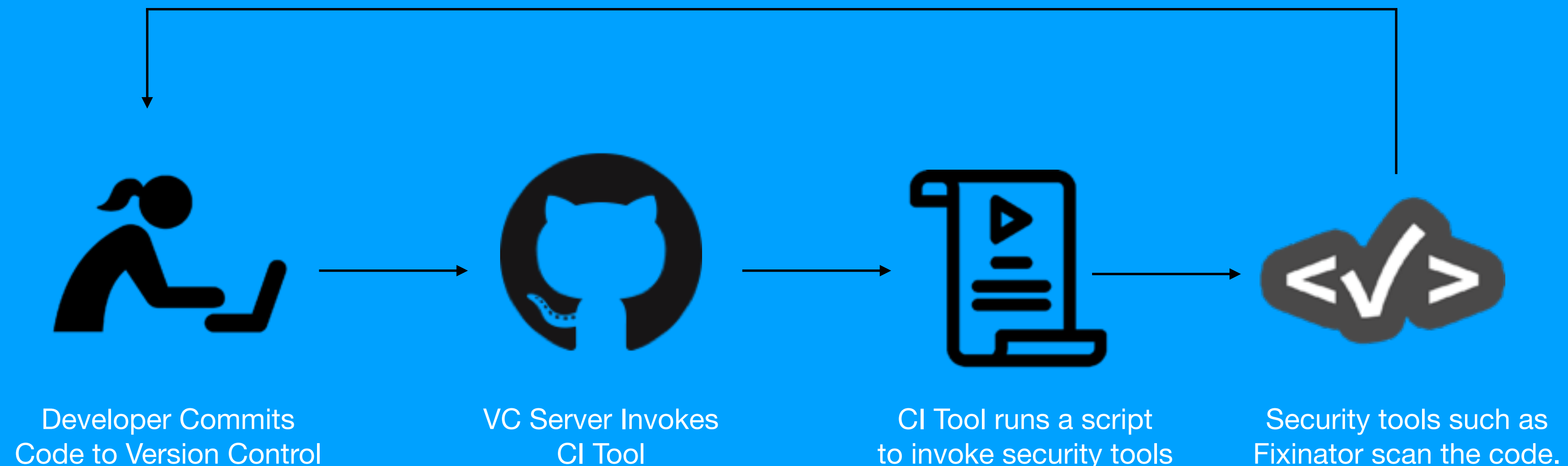
A CI Platform



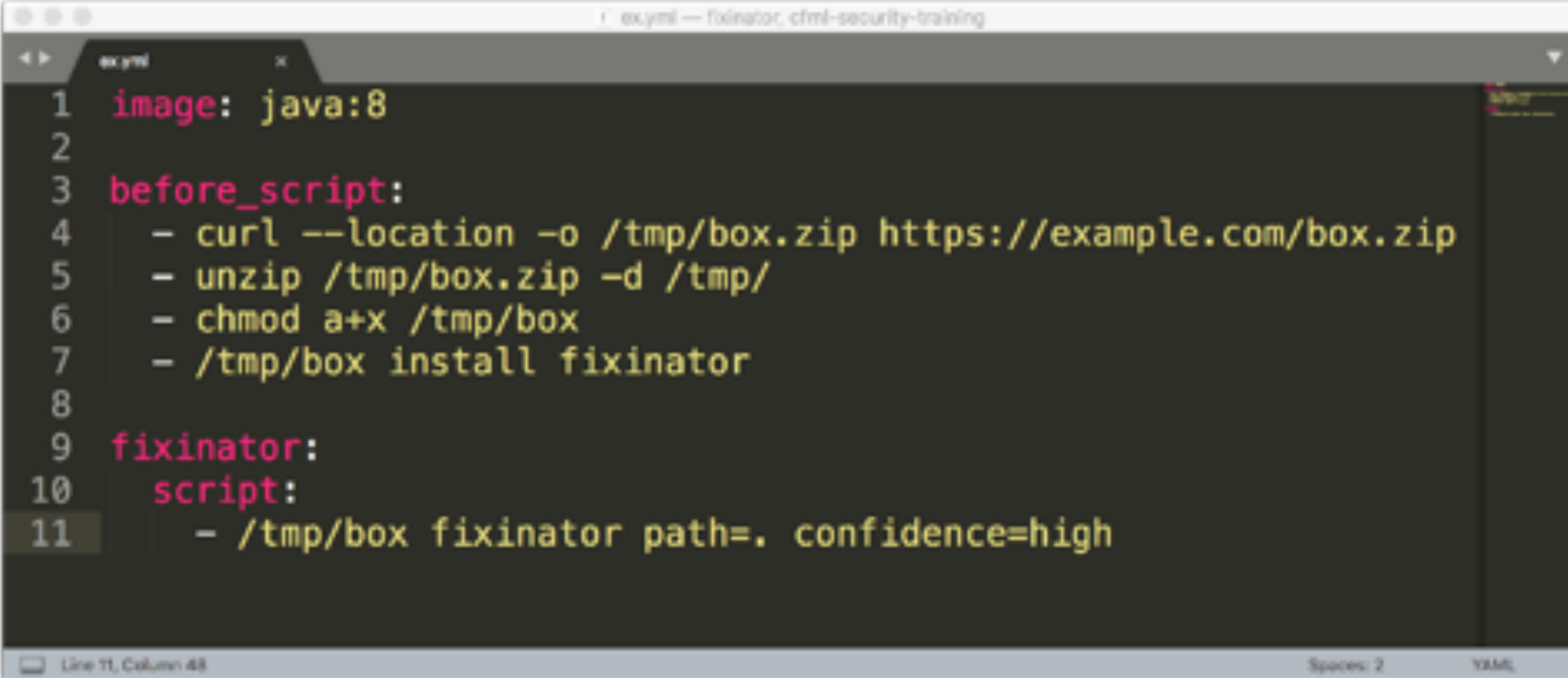
or



Continuous Security Workflow



STOP, YAML TIME

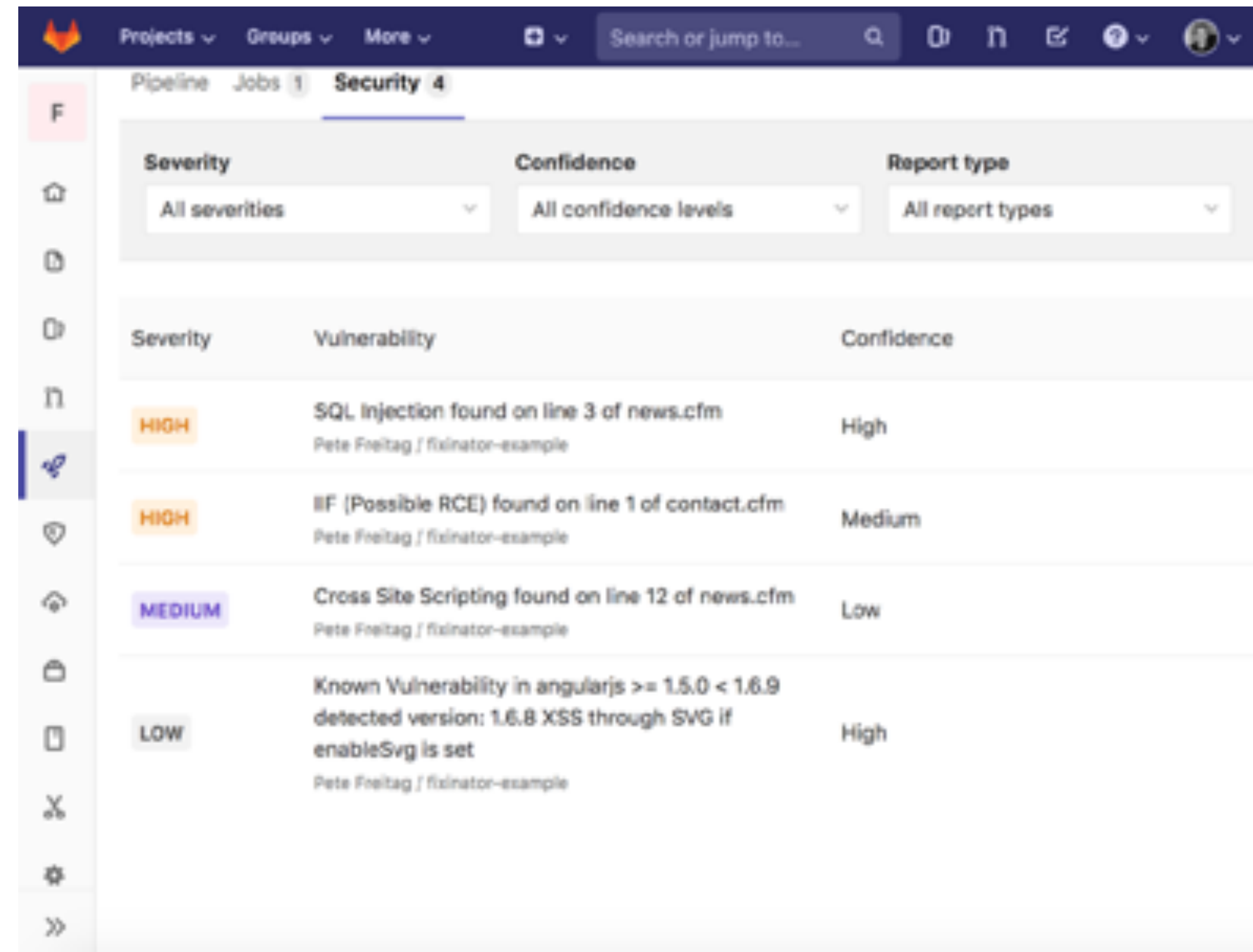


```
ex.yml — fixinator, cfml-security-training
1  image: java:8
2
3  before_script:
4    - curl --location -o /tmp/box.zip https://example.com/box.zip
5    - unzip /tmp/box.zip -d /tmp/
6    - chmod a+x /tmp/box
7    - /tmp/box install fixinator
8
9  fixinator:
10   script:
11     - /tmp/box fixinator path=. confidence=high
```

Line 11, Column 48 Spaces: 2 YAML

This is a script that tells the CI pipeline what tools to use.

GitLab Example



The screenshot shows the GitLab Security interface. At the top, there's a navigation bar with 'Projects', 'Groups', and 'More' dropdowns, a search bar, and user profile icons. Below this, the 'Security' tab is active, showing 4 items. A filter bar allows selecting 'Severity' (All severities), 'Confidence' (All confidence levels), and 'Report type' (All report types). The main table lists vulnerabilities with columns for Severity, Vulnerability, and Confidence.

Severity	Vulnerability	Confidence
HIGH	SQL Injection found on line 3 of news.cfm Pete Freitag / fixinator-example	High
HIGH	IIIF (Possible RCE) found on line 1 of contact.cfm Pete Freitag / fixinator-example	Medium
MEDIUM	Cross Site Scripting found on line 12 of news.cfm Pete Freitag / fixinator-example	Low
LOW	Known Vulnerability in angularjs >= 1.5.0 < 1.6.9 detected version: 1.6.8 XSS through SVG if enableSvg is set Pete Freitag / fixinator-example	High

➡ <https://gitlab.com/pfreitag/fixinator-example/pipelines/>

SHIFT LEFT

A DevOps term:

- Requires continuous integration / testing

- Developers find / fix problems early in development

- Yields higher quality software, better development

It's easier to fix a
bug the same
day you wrote the
code that caused it

GETTING STARTED WITH CI

- Michael Born's Course: Five days of CI with ColdFusion and Bitbucket: <https://learncf.teachable.com/>
- Fixinator Continuous Security Guides: <https://github.com/foundeo/fixinator/wiki/Continuous-Integration-Guide>
 - How to setup CI on GitLab, Bitbucket, TravisCI, Azure DevOps, AWS CodeBuild, CircleCI, and Jenkins.
 - Example: <https://gitlab.com/pfreitag/fixinator-example/pipelines/>
- My CI Presentation ITB2020: <https://www.petefreitag.com/item/902.cfm>
- Brian Sappey's DevWeek Presentation, and [CI / CD WhitePaper for CF2021](#)

Where can I learn more

About Web Application Security?

- OWASP.org - tons of info about web application vulnerabilities
- Foundeo Security Training Class - 6-8 hour class goes more in depth on everything we discussed today.
 - <https://foundeo.com/consulting/coldfusion/security-training/>

Thank You!

Questions?

Pete Freitag pete@foundeo.com
or [@pfreitag](https://twitter.com/pfreitag) on Twitter

foundeo

foundeo.com | fuseguard.com | hackmycf.com | fixinator.app