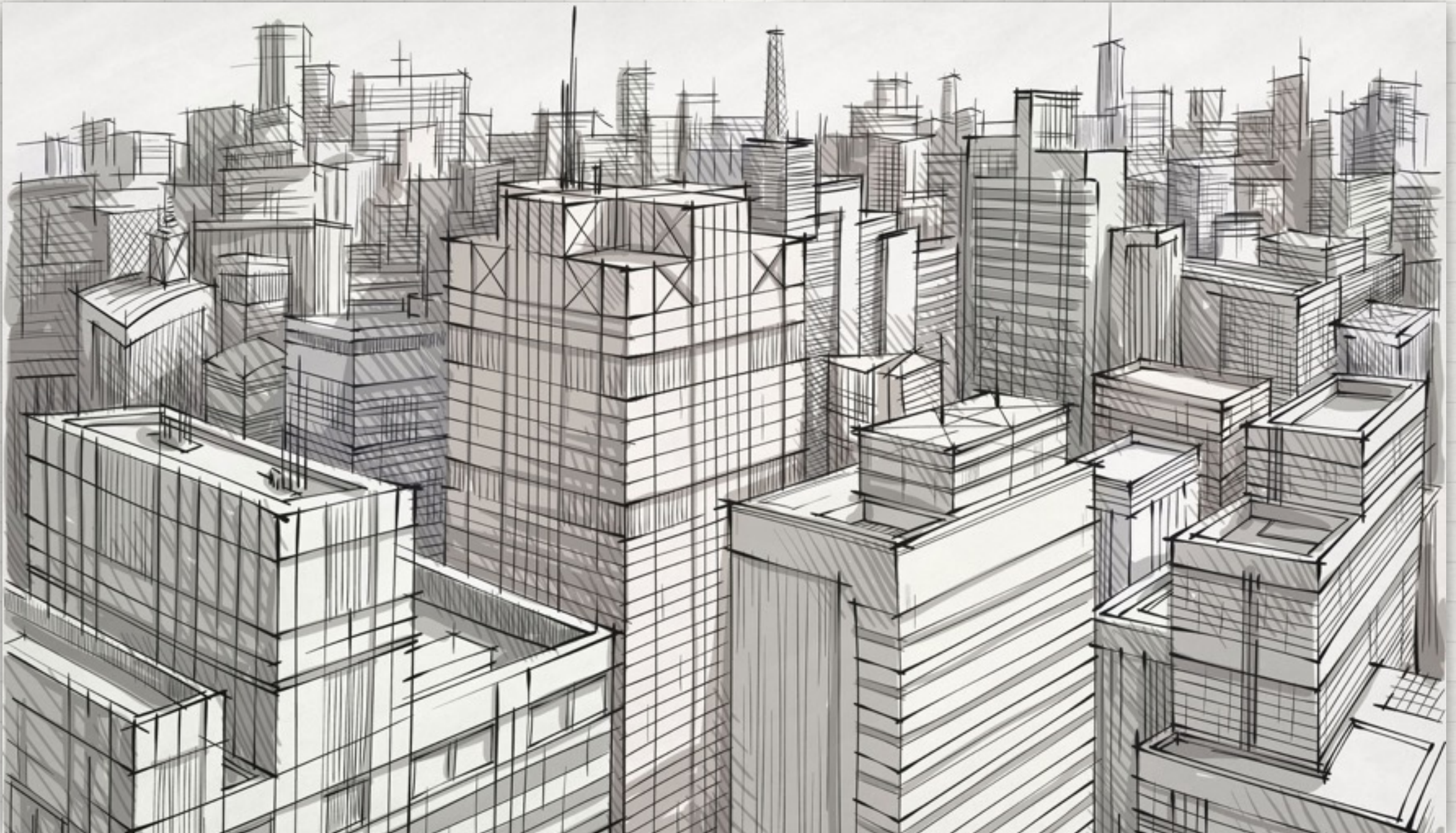


APPROACHES TO SECURE CFML CODE

PETE FREITAG, FOUNDEO INC.



foundeo

ABOUT PETE

- Guy who wrote the ColdFusion Lockdown Guides CF9-CF2018
- 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), #CFML Slack
 - I will post these slides on my blog
- Using CFML since late 90s

A FEW RECENT HEADLINES

FROM 2020

- Microsoft: 250 million customer support logs from misconfigured elasticsearch servers [\[link\]](#)
- MGM Resorts: 10.6 million customer records including names, addresses, dob posted to a hacking forum. [\[link\]](#)
- Tupperware: Hackers added code to checkout page to collect payment info. [\[link\]](#)
- Marriott: 5.2 million customer records including names, addresses, phone numbers, dob. [\[link\]](#)
- Last year a 3-year-old boy repeatedly entered the wrong password, locked up his dad's iPad until the year 2067

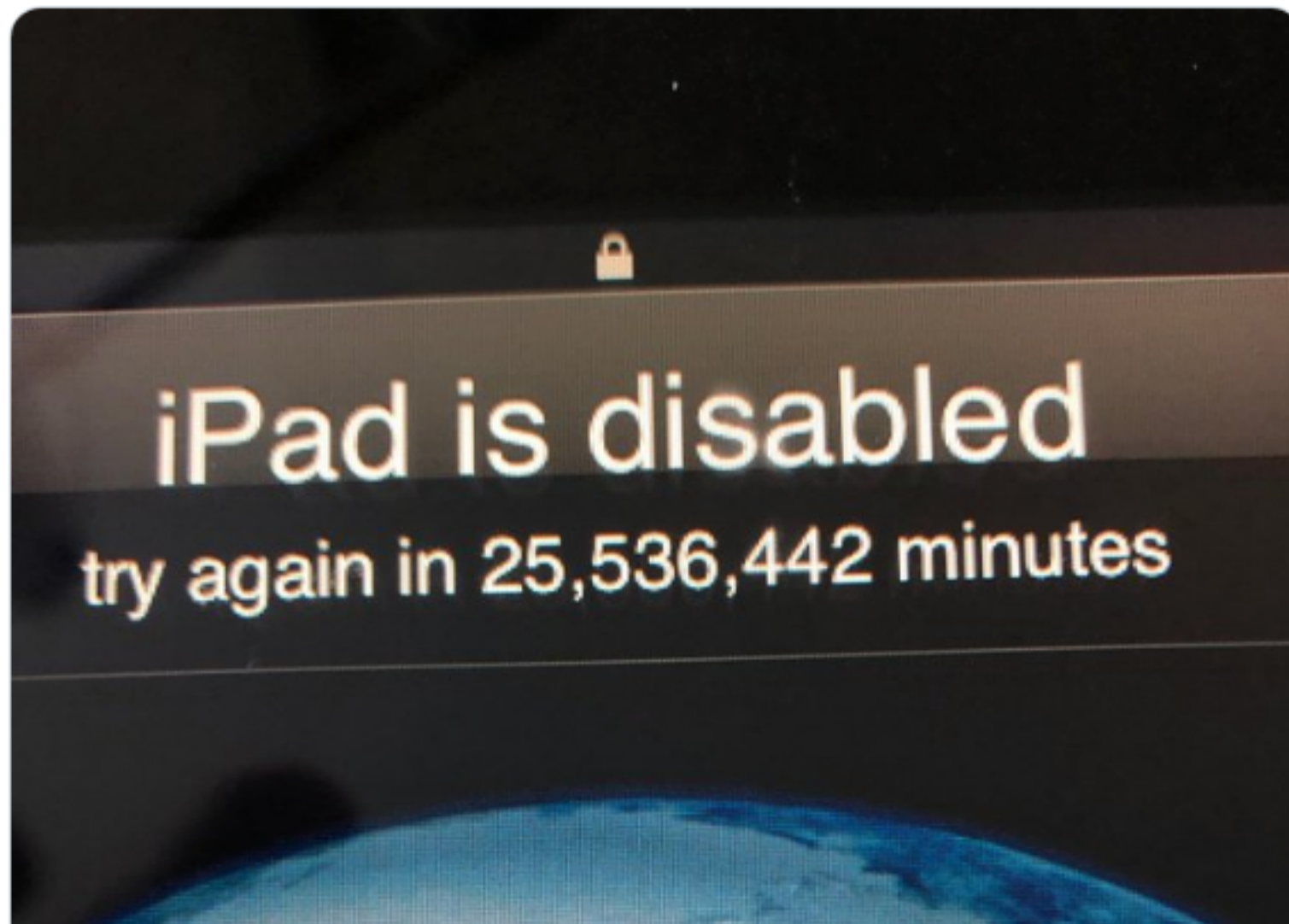


Evan Osnos ✓

@eosnos



Uh, this looks fake but, alas, it's our iPad today after 3-year-old tried (repeatedly) to unlock. Ideas?



TAKEAWAYS

- You have all been impacted by data breaches
- 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

DO YOU HAVE TO
WORK WITH
AN OLD &
LARGE
CODEBASE?



MATURE CODEBASES

TYPICALLY

- Have **thousands** of source code files
- Has code you hope you don't have to see again.
- Can take weeks, but often months of work to properly secure.
- Can be hard to fix, brittle
- Probably uses outdated techniques

DIFFERENT APPROACHS

FOR SECURING A LARGE CODEBASE

- **Focus Mode** - Spend several weeks dedicated to identifying & fixing vulnerabilities.
- **Prioritize** - Spend time identifying the most critical vulnerabilities and patch less critical vulnerabilities as you see them.
- **As you go** - As you work on files fix vulnerabilities as you see them. You may miss some vulnerabilities with this approach.
- **Hire Someone to Find or Fix issues** - Can work well when you are too busy to find or fix issues.

HOW DO YOU START?

SECURING THAT CODE

STEP 1: DELETE THE CODE!

YOU MIGHT
HAVE A LOT
OF CODE THAT
NEVER RUNS



OLD CODE IS
OFTEN FULL
OF SECURITY
HOLES



YOU MIGHT BE USING...

HOMEMADE 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

IDENTIFY UNUSED CODE

- 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

FINDING OLD FILES (MODIFICATION DATE)

- 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

FINDING OLD FILES (BY ACCESS DATE)

- 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.

PATCH THAT SERVER

- Use ColdFusion 2016 or greater. 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)



PATCH THAT SERVER

FIX VULNERABILITIES

- 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

LOCKDOWN THE SERVER

MITIGATES POTENTIAL IMPACT OF A VULNERABILITY

- What user is the JVM running as?
 - If your CFML server is running as SYSTEM or root then the attacker can do a lot more harm.
- 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?

“

NEARLY 60% OF BREACHES
DUE TO UN-PATCHED
VULNERABILITY

— *ServiceNow Survey*

”

EQUIFAX BREACH

2017

- 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

THIRD PARTY LIBRARIES

- 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)

POLL Q2

CONTINUOUS SECURITY

YOU NEED VERSION CONTROL



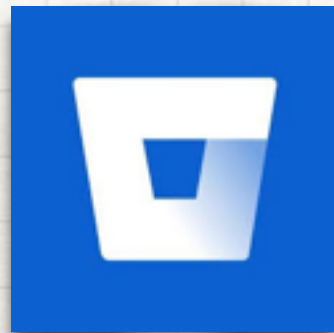
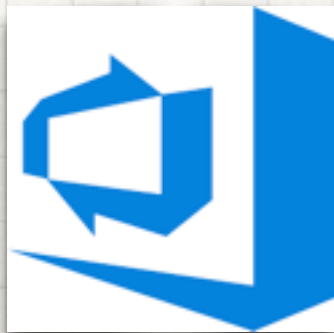
GitHub



GitLab

**LEARN YET ANOTHER
MARKUP LANGUAGE**

A CI PLATFORM

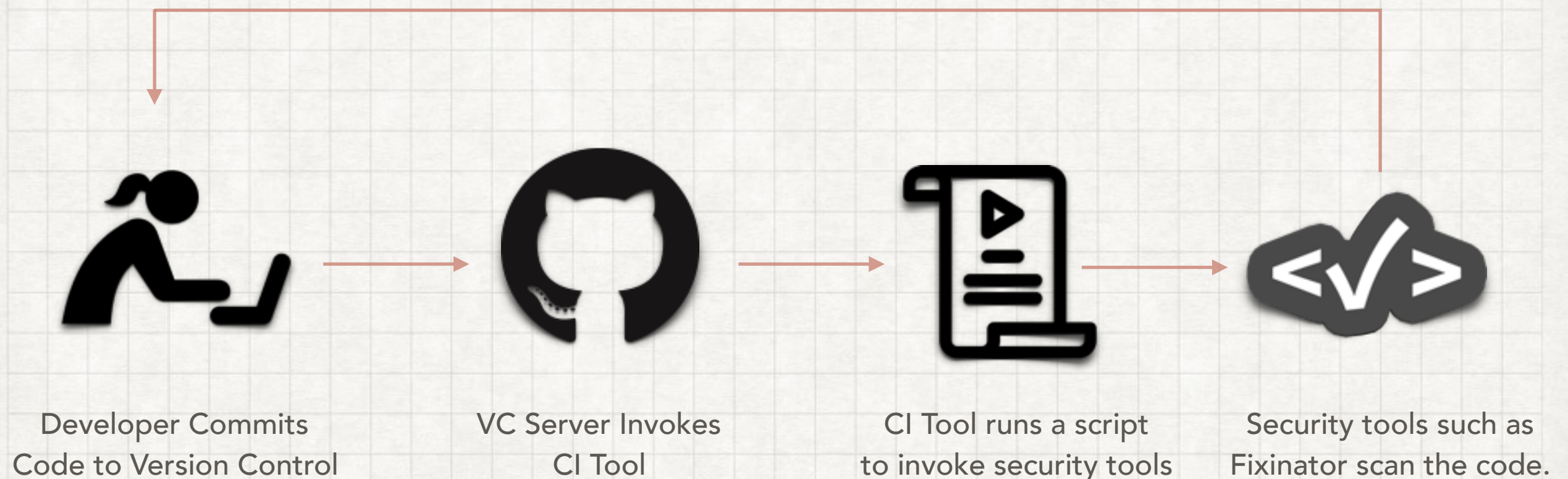


GitLab

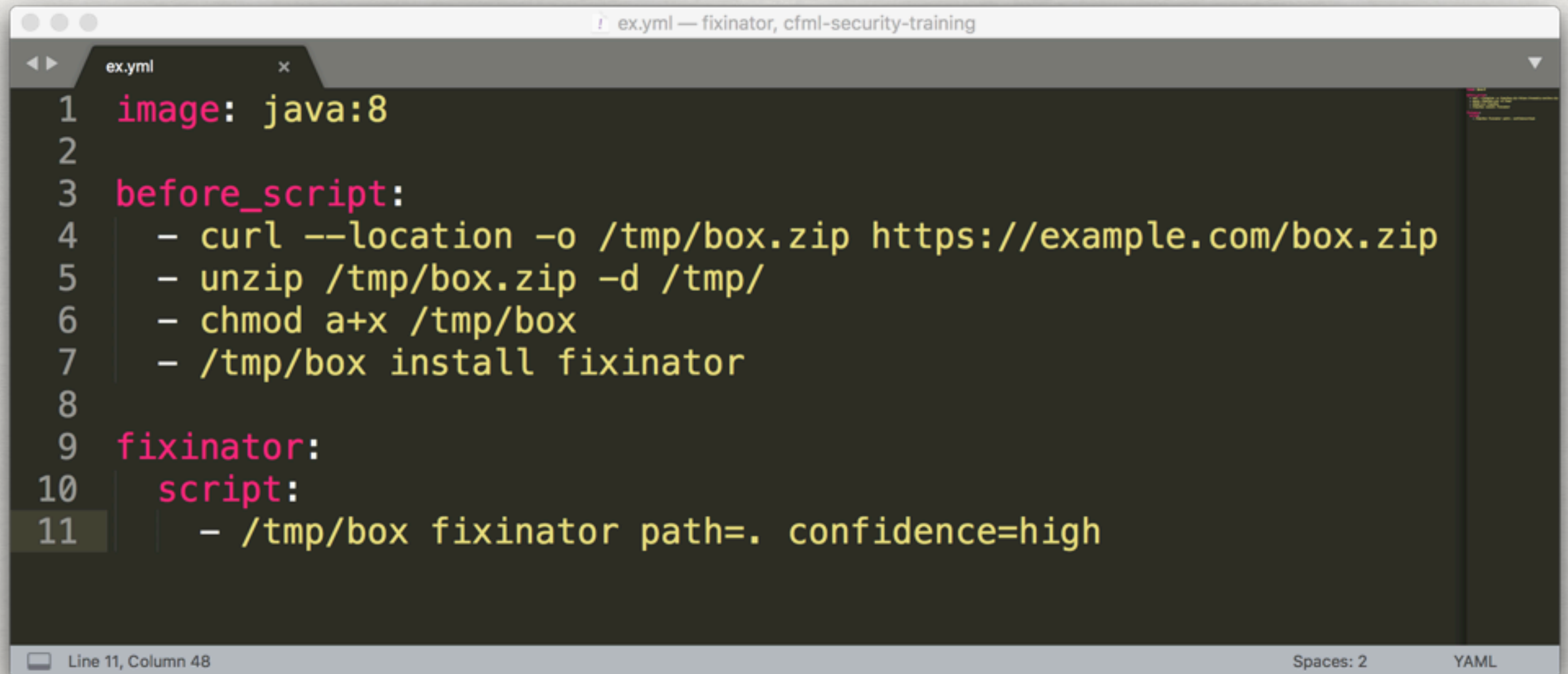
or



HOW IT WORKS



STOP, YAML TIME



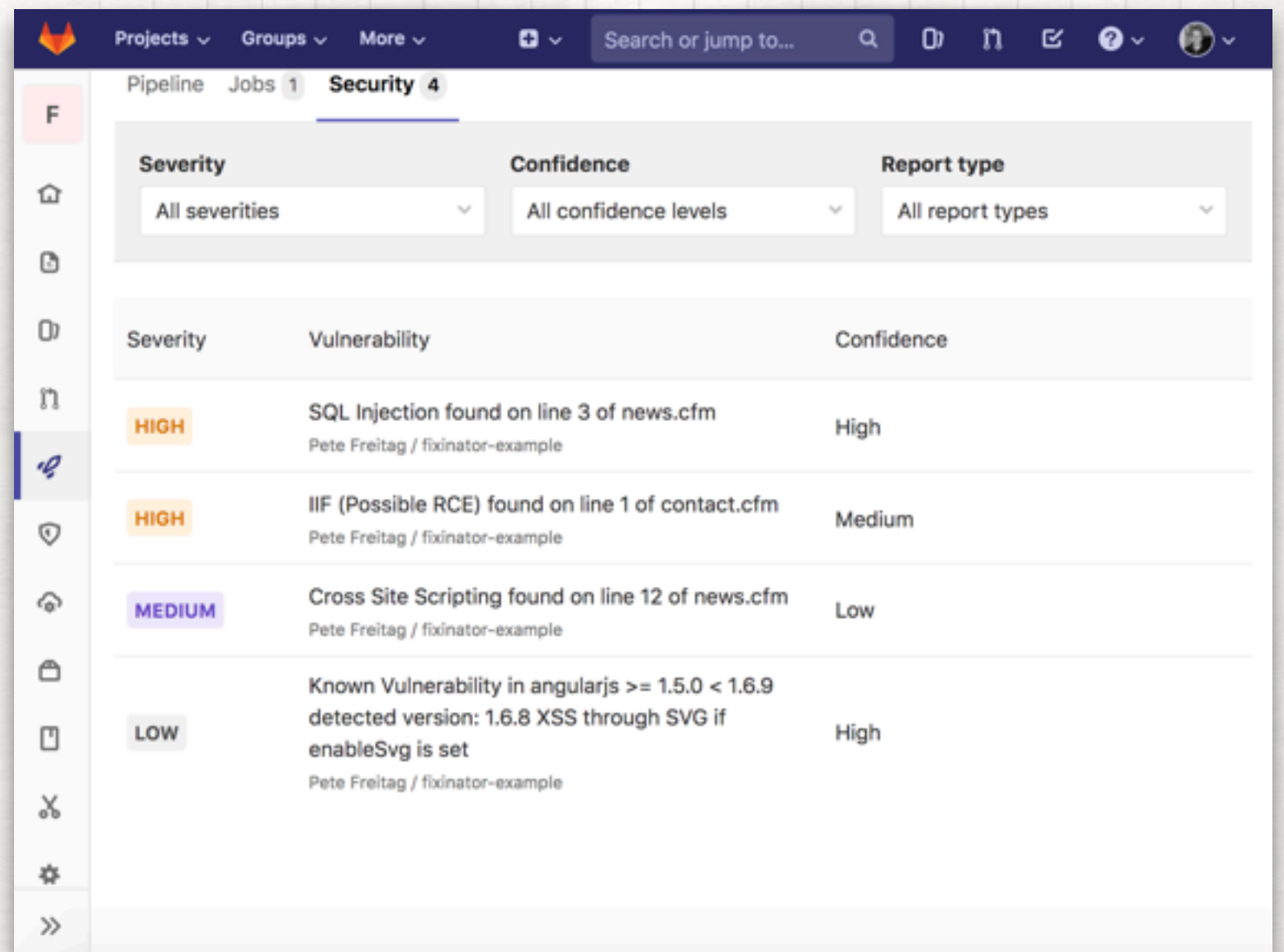
```
ex.yml — fixinator, cfml-security-training
1  image: java:8
2
3  before_script:
4    - curl --location -o /tmp/box.zip https://example.com/box.zip
5    - unzip /tmp/box.zip -d /tmp/
6    - chmod a+x /tmp/box
7    - /tmp/box install fixinator
8
9  fixinator:
10   script:
11     - /tmp/box fixinator 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 are tabs for 'Pipeline', 'Jobs 1', and 'Security 4'. Below these are filters for 'Severity' (All severities), 'Confidence' (All confidence levels), and 'Report type' (All report types). The main table lists four vulnerabilities with their severity levels (HIGH, MEDIUM, LOW) and confidence levels (High, Medium, Low).

Severity	Vulnerability	Confidence
HIGH	SQL Injection found on line 3 of news.cfm Pete Freitag / fixinator-example	High
HIGH	IIF (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/>

SHIFTING LEFT

A DevOps term, give developers ability to fix issues early in development

GETTING STARTED WITH CI

RESOURCES

- 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>

IMPLEMENT A WAF

WEB APPLICATION FIREWALLS

- Inspect HTTP Request or Response
 - Block or log malicious requests
 - Provides Defense in Depth
- Several options
 - Hardware Based
 - Software Based / Application Level
 - FuseGuard

HOW DO YOU START SECURING THAT LARGE CFML CODEBASES?

**STEP 2: IDENTIFY HIGH RISK
VULNERABILITIES IN YOUR CODE.**

HIGH RISK VULNERABILITIES

TAKE CARE OF THESE FIRST

- File Uploads
- Remote Code Execution / Dynamic Evaluation Issues
- SQL Queries (SQL Injection)
- File System Access / Path Traversals
- Dynamic Process Execution (CFEXECUTE)
- Anything that can fully compromise server

HOW I CLASSIFY VULNERABILITIES

**Compromises
the server(s) directly**

**Compromises
users**

Both are important but where do you start?

HOW I CLASSIFY VULNERABILITIES

**Compromises
the server(s) directly**

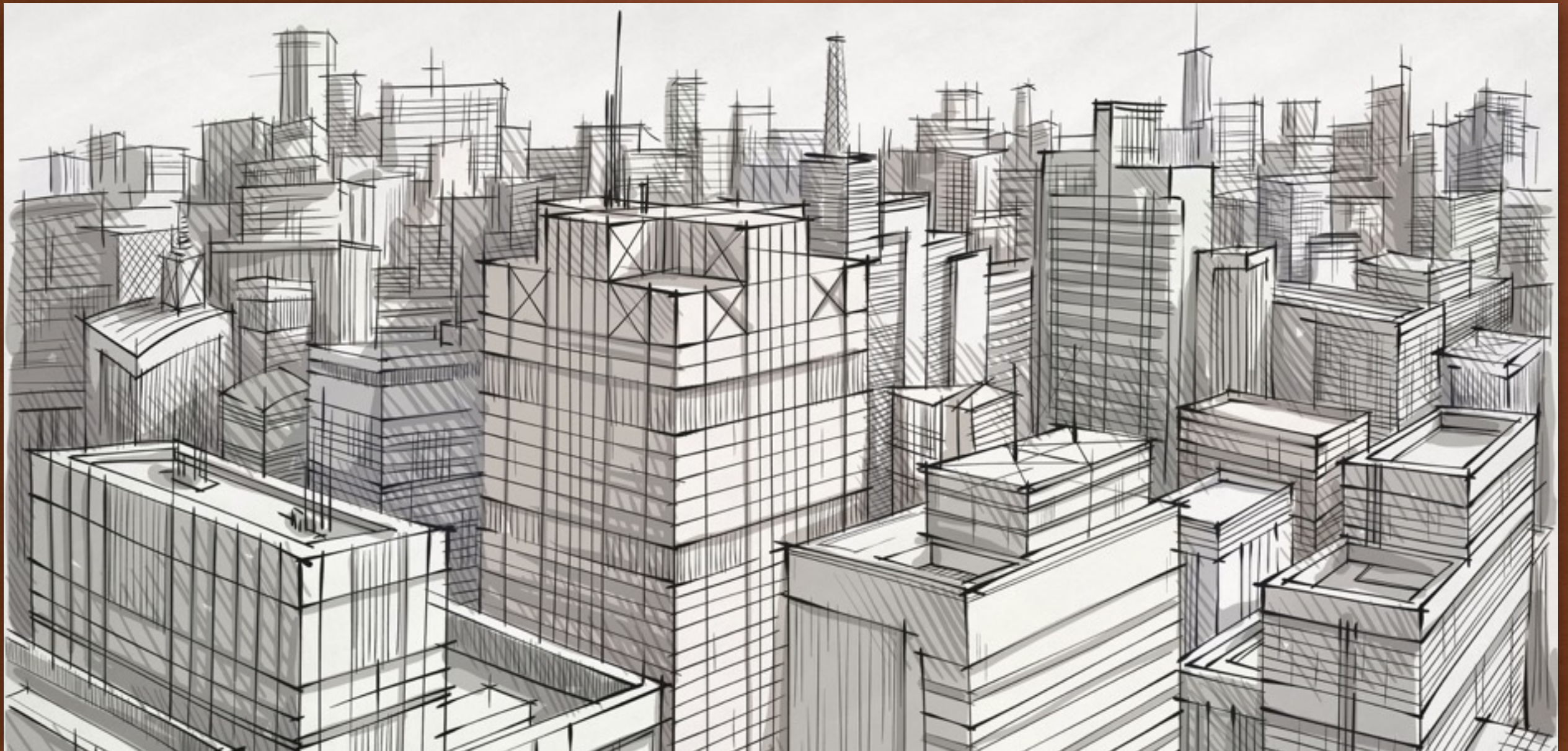
Examples:
SQL Injection
File Upload / Access
Remote Code Execution

**Compromises
users**

Examples:
XSS
CSRF
Session Hijacking

Both are important but where do you start?

REMOTE CODE EXECUTION VIA EVALUATE



COMMON LEGACY EVALUATE

CODE EXAMPLE

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

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

**EVALUATE
EXAMPLE**

FIXING LEGACY EVALUATE EXAMPLE

USE BRACKET NOTATION

```
<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

- 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?**

IIF

IF YOU ARE USING IIF STOP USING IIF

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

The second and third arguments are evaluated dynamically!

IIF EXAMPLE

FIXING IIF

USE TERNARY OPERATOR (CF9+)

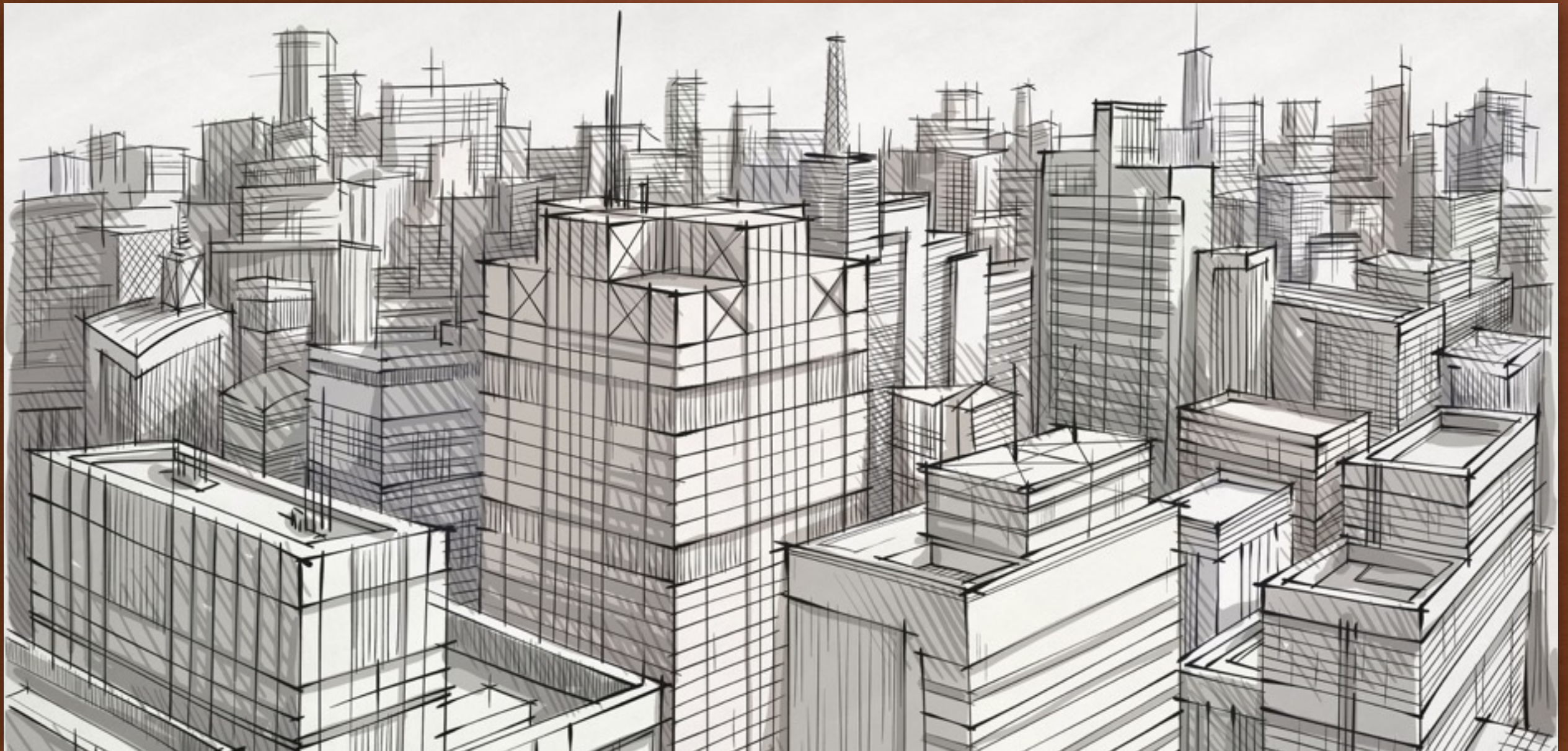
```
Hi #(!len(url.name)) ? "Friend" : url.name#
```

ELVIS OPERATOR (CF11+)

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

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

COMMON YET DANGEROUS FILE UPLOADS



FILE UPLOADS

3 CORE RULES

FILE UPLOADS

RULES # 1



Never trust a MIME!

FILE UPLOADS RULE #1

NEVER TRUST A MIME

- CF10 added strict attribute to cffile 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** against a whitelist
 - CF10+ allows you to specify file extensions in accept attribute

FILE UPLOADS

RULE #3

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

FILE UPLOADS

ADDITIONAL TIPS

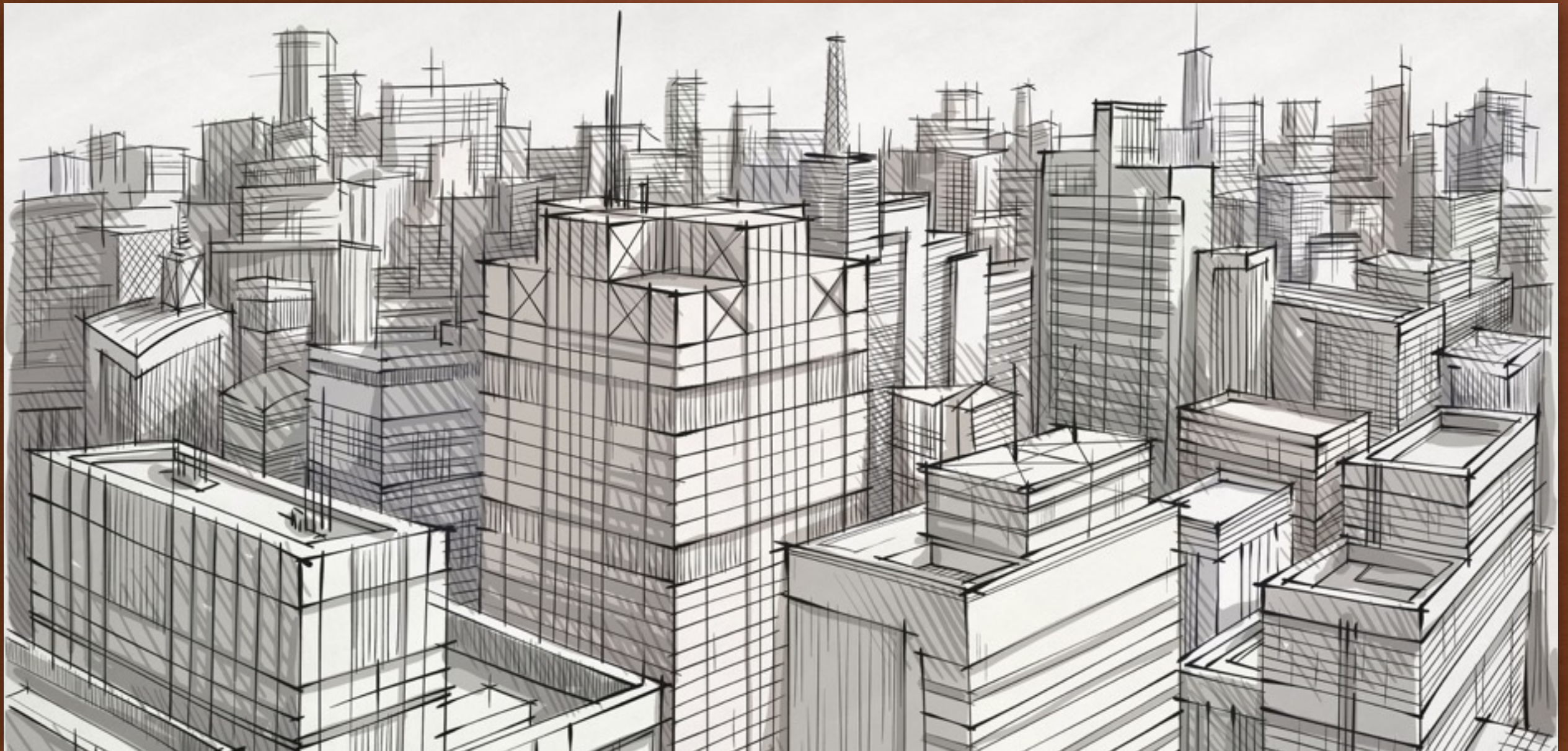
- Inspect file content: `fileGetMimeType`, `isImageFile`, `isPDFFile`, etc
- Upload to static content server (s3 for example)
 - Upload directly to s3: <https://www.petefreitag.com/item/833.cfm>
- Make sure directory serving uploaded files cannot serve dynamic content.
- File Extension Whitelist on Web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

NEW FILE UPLOAD FEATURES

IN LATEST CF HOTFIX

- 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

FILE SYSTEM ACCESS & PATH TRAVERSAL



PATH TRAVERSAL

VULNERABLE CODE EXAMPLE

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


PATH TRAVERSAL EXAMPLE

FIXING PATH TRAVERSALS

TIPS

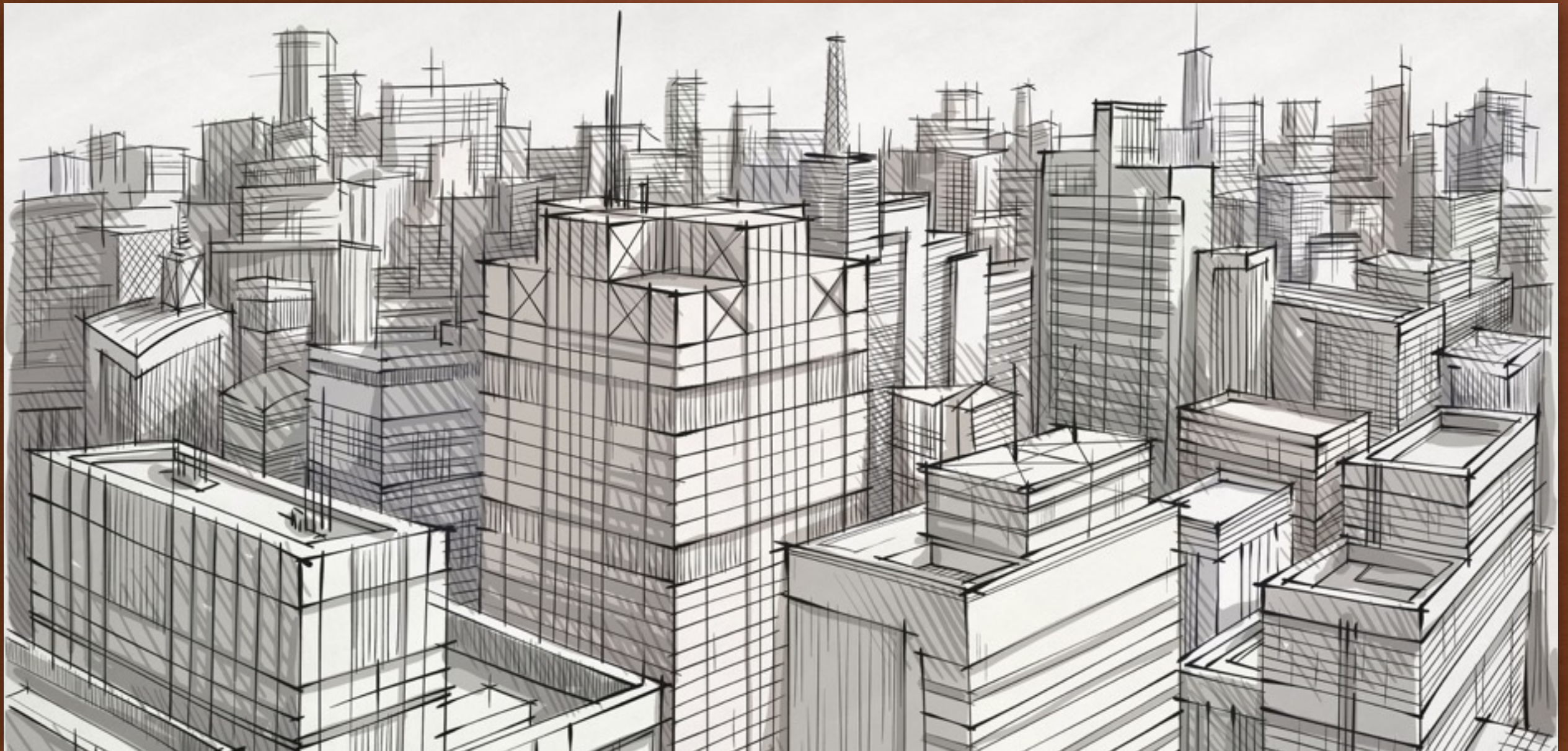
- Avoid variables in paths
 - If you really need to use a variable strip out everything except a-z0-9
- Use the CF11 Application.cfc setting `this.compileExtForInclude` setting.

FINDING FILE ACCESS ISSUES

CAN BE TIME CONSUMING

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

SQL INJECTION



CLASSIC SQL INJECTION

CODE EXAMPLE

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

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

FIXING SQL INJECTION

CODE EXAMPLE

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


SCRIPT BASED

SQL INJECTION

```
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`, `ormExecute query`
- Use Static Code Analyzer (CFBuilder 2016+)
- Fixinator can find, and fix them for you
- Fix when you see one as you work

SECURING LEGACY CFML

**STEP 3: FIX ADDITIONAL
VULNERABILITIES IN YOUR CODE.**

WHAT'S NEXT

TO REVIEW

- Session Handling (sessionRotate, sessionInvalidate)
- Scope Injection
- Authentication / Authorization / Forgot / Remember Me Code
- Cross Site Scripting
 - CF2016 <cfoutput encodefor="html">
- Cross Site Request Forgery
- Timing Attacks
- Visit [OWASP.org](https://www.owasp.org) for tons of info about web application vulnerabilities

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