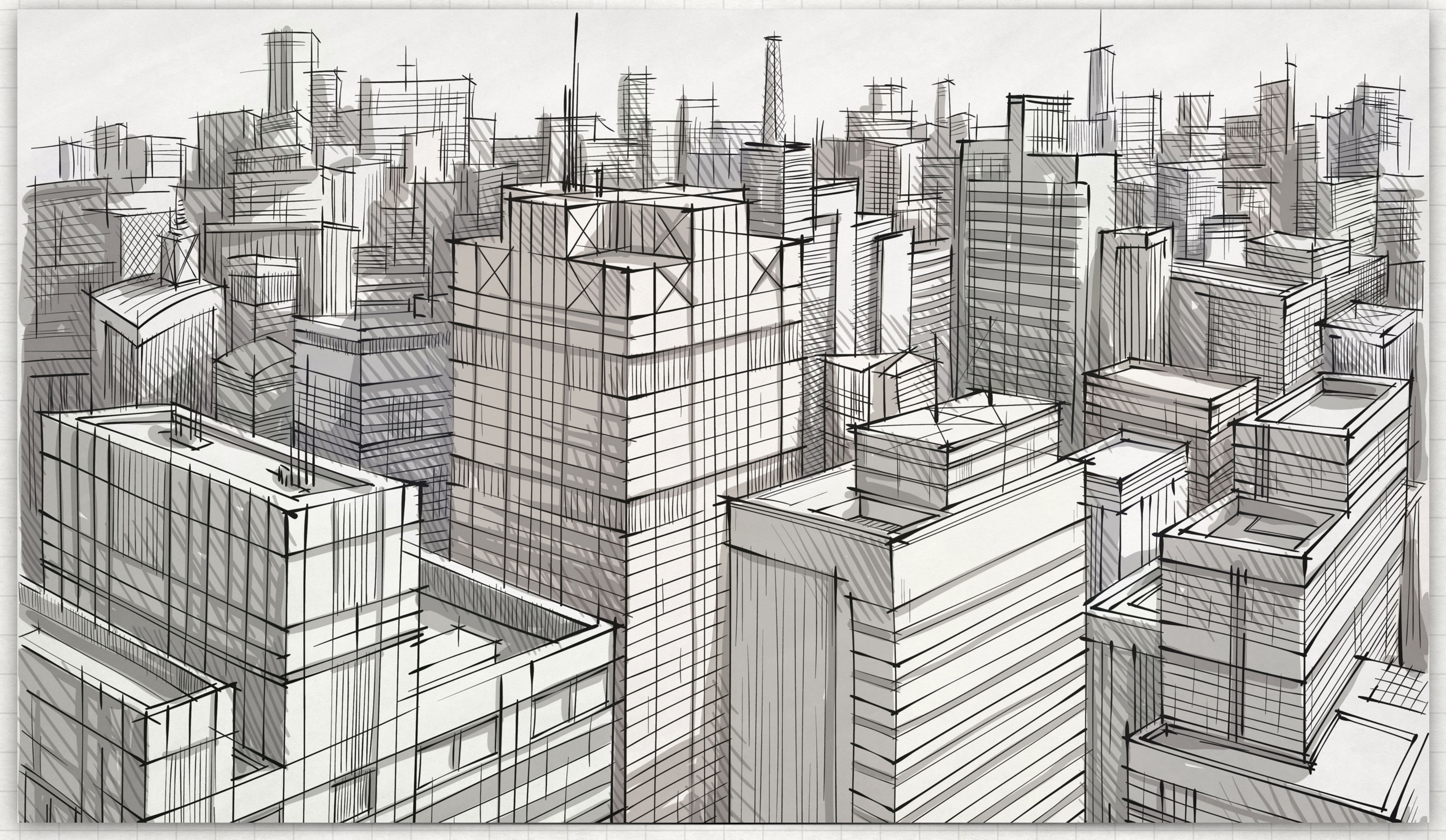
SECURING MATURE CFML CODEBASES

PETE FREITAG, FOUNDEO INC.



ABOUT PETE

- My Company: Foundeo Inc.
  - **Consulting**: Code Reviews, Server Reviews, Development
  - **FuseGuard**: Web App Firewall for CFML
  - **HackMyCF**: Server Security Scanner
- Blog (petefreitag.com), Twitter (@pfreitag), #CFML Slack
- Using CFML since late 90s

AGENDA

- Large Codebase Challenges
- How do you get started?
    - Low Hanging Fruit
    - Things to focus on
- What's Next?
- Disclaimer: This approach may not be appropriate for all scenarios. This is a generalized approach which I have found can work well for many.

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- Probably uses outdated techniques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

- **Beast 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.

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 11 or greater, CF10 Ended May 2017 (CF9 and below are no longer supported and no longer patched by Adobe).

- 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?

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 CODE

~~STEP 1: DELETE THE CODE!~~

STEP 1: LOW HANGING FRUIT

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?

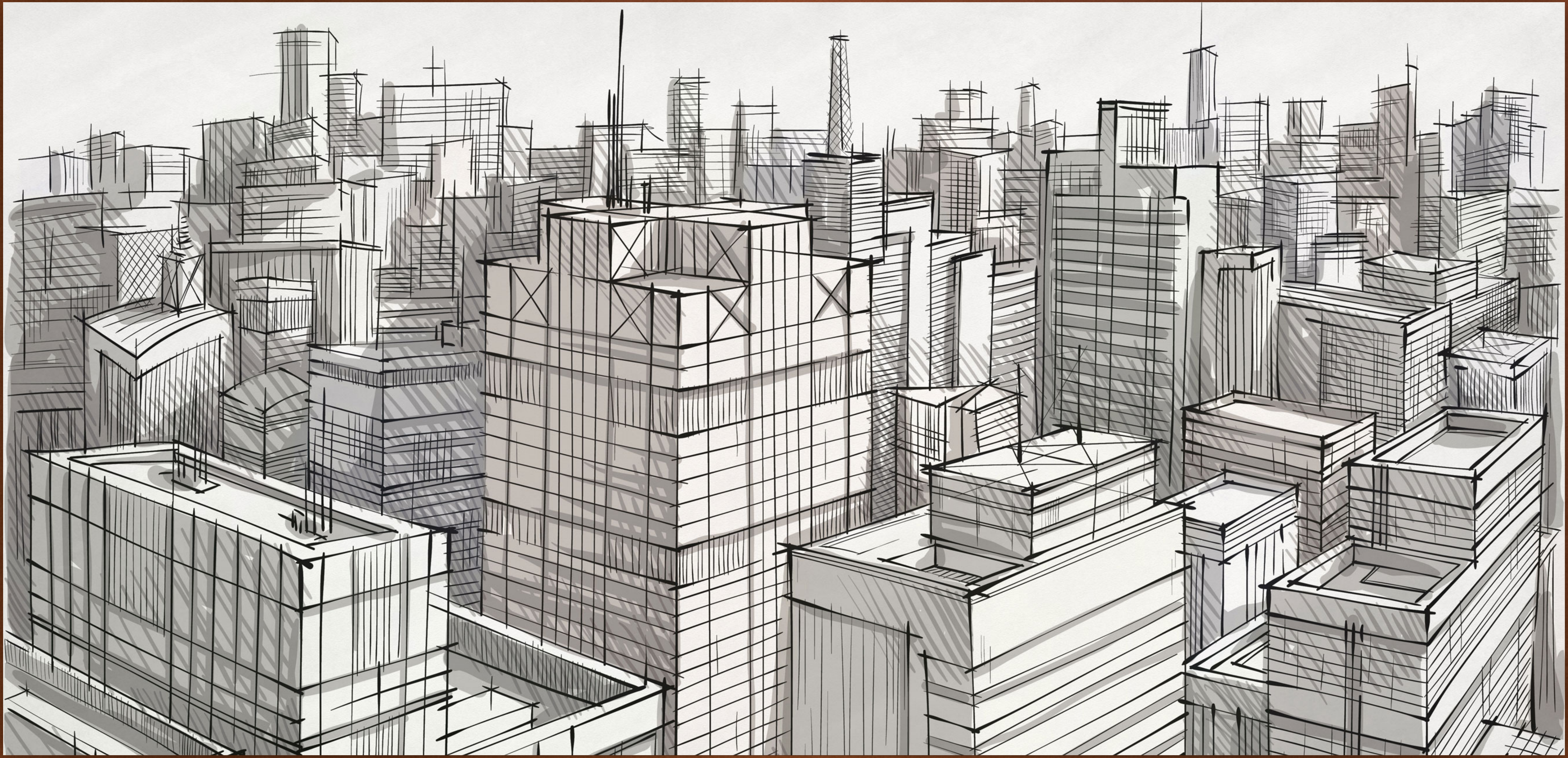
HOW I CLASSIFY VULNERABILITIES

Compromises the server(s) directly	**Compromises users**
Examples: SQL Injection File Upload / Access Remote Code Execution	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 = "Wednesday">
<cfset day_2 = "Thursday">
<cfset day_3 = "Friday">

<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#
```

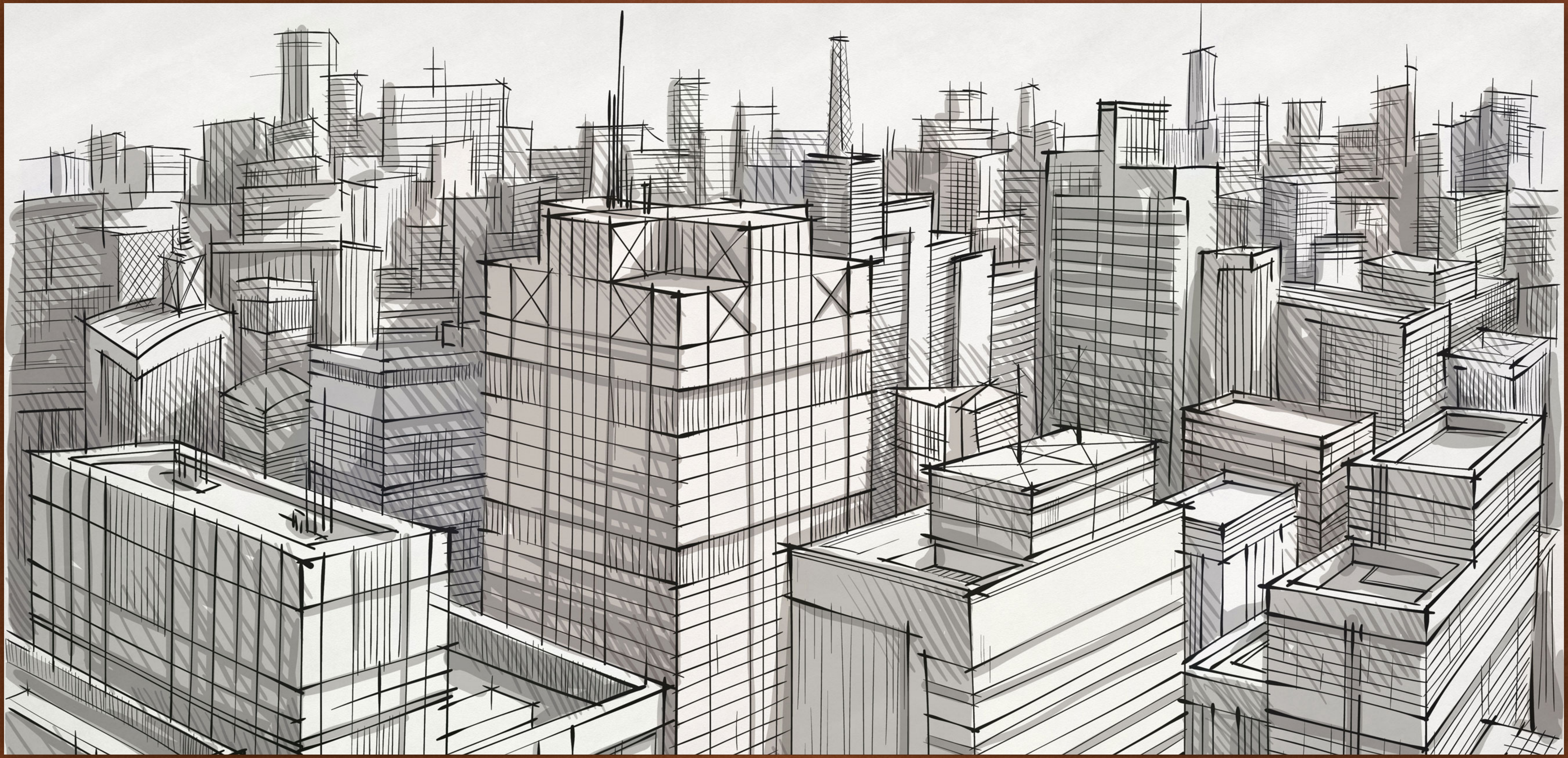
ELVIS OPERATOR (CF11+)

```
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?
- Lucee also has strict attribute but it doesn't seam to do much.

FILE UPLOADS

RULE #2

- Always validate the **file extension** against a whitelist
  - CF10+ allows you to specify file extensions in accept attribute
  - Lucee also supports extensions in accept attribute
  - In Lucee you can mix MIME types and extensions, don't do that.

FILE UPLOADS

RULE #3

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

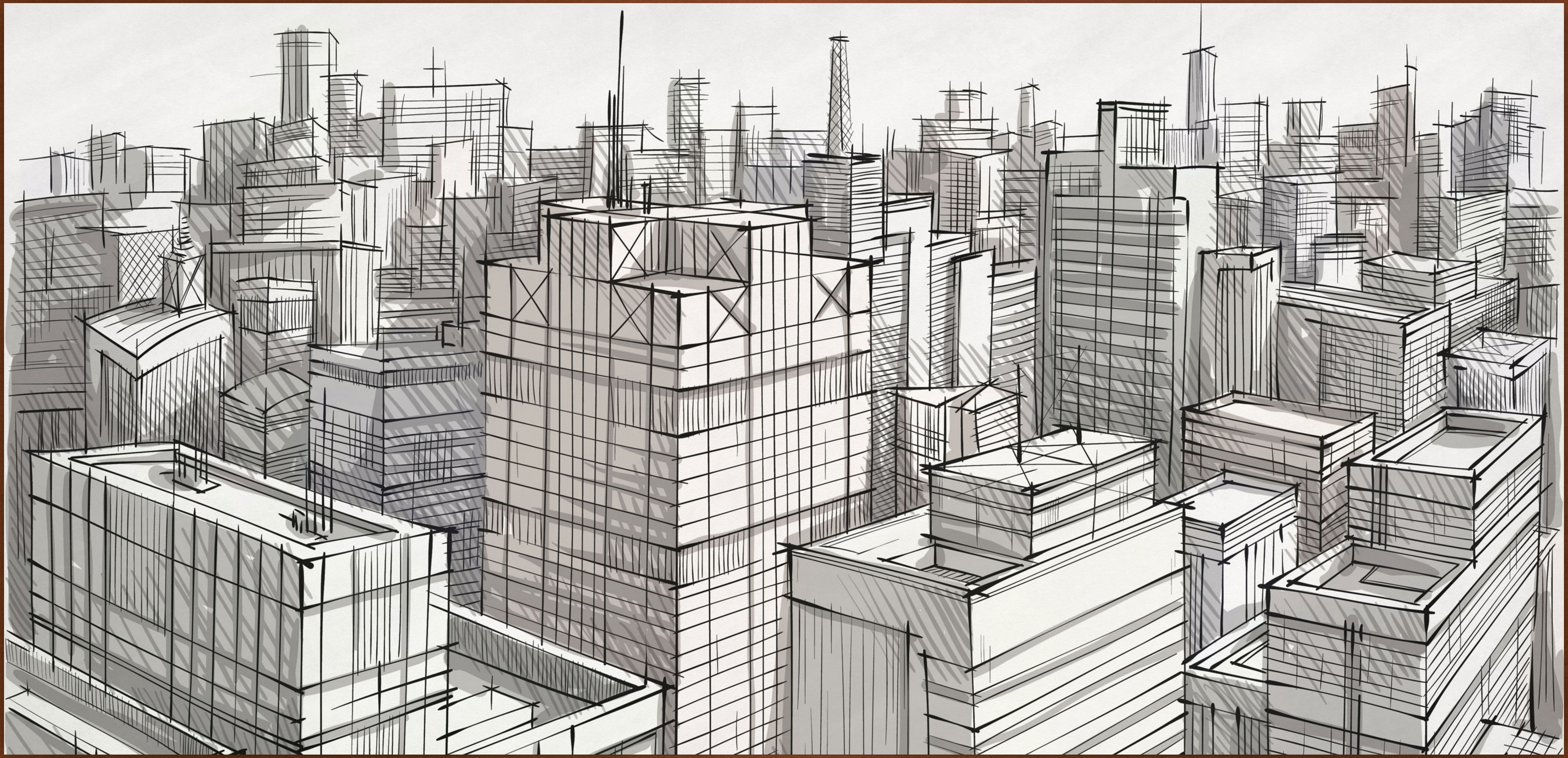
FILE UPLOADS

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/

FILE SYSTEM ACCESS &

PATH TRAVERSAL



PATH TRAVERSAL

VULNERABLE CODE EXAMPLE

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

PATH TRAVERSAL EXAMPLE

FIXING PATH TRAVERSALS

TIPS

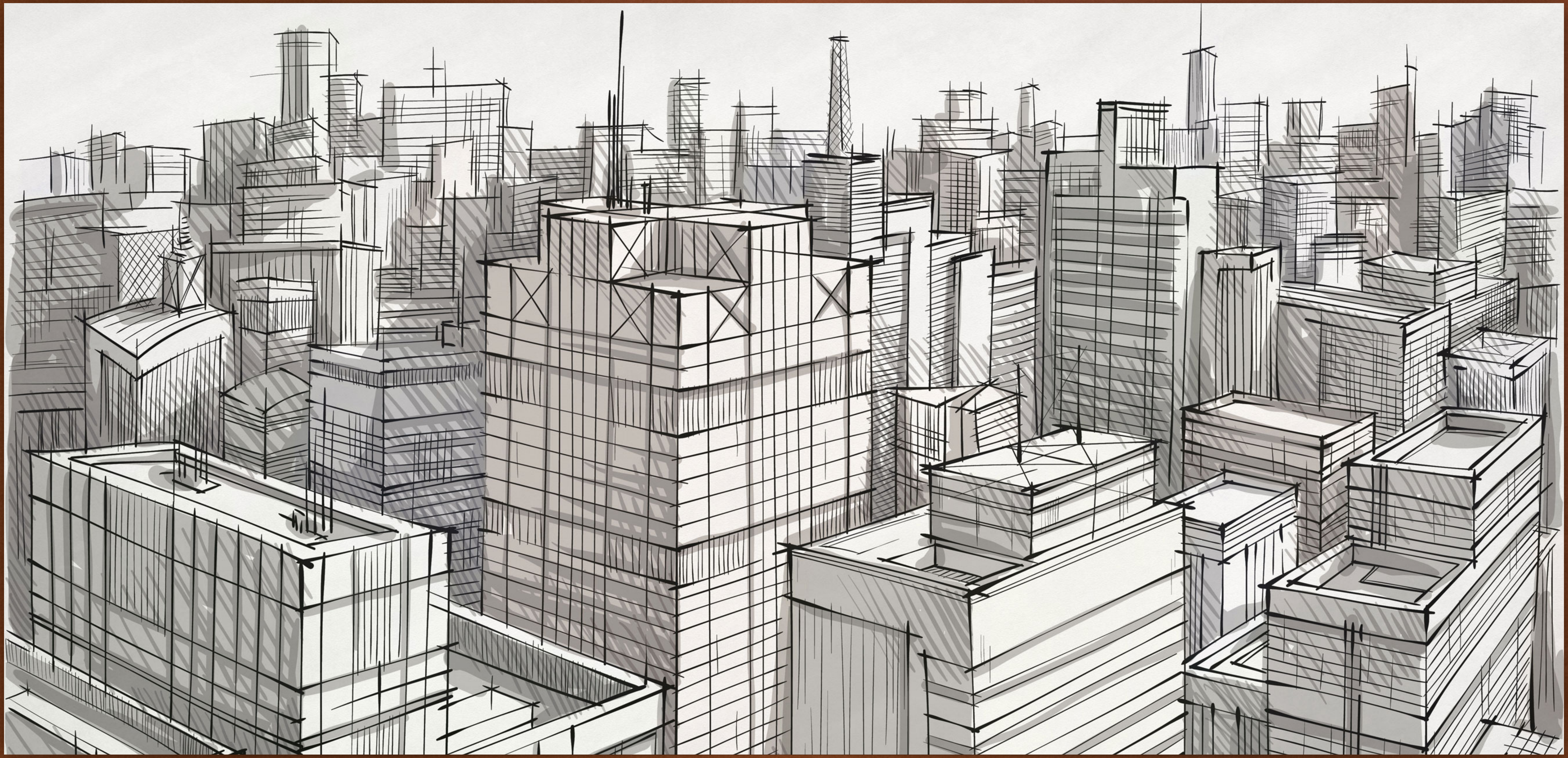
- Avoid variables in paths
    - If you really need to use a variable strip out everything except a-z0-9
- Use the CF11 Application.cfc setting this.compileExtForInclude setting.

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>
```

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)
- 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 for tons of info about web application vulnerabilities

THANK YOU

Questions?

Pete Freitag
pete@foundeo.com

foundeo

foundeo.com | fuseguard.com | hackmycf.com