VULNERABILITIES BY ANALOGY

Why is a raven like a writing desk?

What am I doing?

- I'm going to explain common attack and exploitation techniques, through my power of analogy!
- There are some great common parallels between computer security and the real world
- I will gently guide you from the real world into a high-level technical understanding

- Goal: Lay the groundwork of understanding attacks and vulnerabilities for future

- We will also talk about some of the common standards and groupings of threats, vulnerabilities, weaknesses, and attack patterns (OWASP, CWE, CAPEC, etc.)

VULNERABILITIES

the failures

INJECTION FLAWS

Humans + code = sadness

Pizza Robot

Goal:

- Deliver pizza
- Greet human
- Return to pizzeria

Process

1. Human goes to a website
2. Makes their order
3. Enters their name “Joe”
4. The pizza is made and placed in delivery robot
5. Delivery robot is programmed with commands to get to the house
6. Delivery robot delivers pizza and says “Greetings, Joe”
7. Delivery robot returns to base

Forward: 50 ft
Turn: Right
Forward: 300 ft
Turn: Left
Forward: 10 ft
Turn: Left
Forward: 5 ft
Greet: Joe
Deliver: Pizza
Return

Hijacking a Pizza Robot

Forward: 50 ft
Turn: Right
Forward: 300 ft
Turn: Left
Forward: 10 ft
Turn: Left
Forward: 5 ft
Greet: Joe
Deliver: Pizza
Return

Expected:
Joe
Unexpected:
Joe
Turn: Left
Forward: 1 ft
Turn: Left
Forward: 1 ft

Forward: 50 ft
Turn: Right
Forward: 300 ft
Turn: Left
Forward: 10 ft
Turn: Left
Forward: 5 ft
Greet: **Joe**
Turn: Left
Forward: 1 ft
Turn: Left
Forward: 1 ft
Deliver: Pizza
Return

What's happening!?

- Everything in White is "Code" – programmer supplied
  - *Code is simply special text that tells a system what to do*
  - *GPS for a computer*
- Everything in Red is "Data" – user supplied
  - *Data is anything else: text, photos, etc.*
- The programmer assumed the name would not include "Code"
  - *Nobody's named "Turn" or "Forward" right?*
- When the user supplied those things the robot wrongly interpreted them as "Code"
- This is fundamentally the same thing that happens in XSS, SQLi, Buffer Overflows, XML injection, and more!

Forward: 50 ft
Turn: Right
Forward: 300 ft
Turn: Left
Forward: 10 ft
Turn: Left
Forward: 5 ft
Greet: **Joe**
Turn: Left
Forward: 1 ft
Turn: Left
Forward: 1 ft
Deliver: Pizza
Return

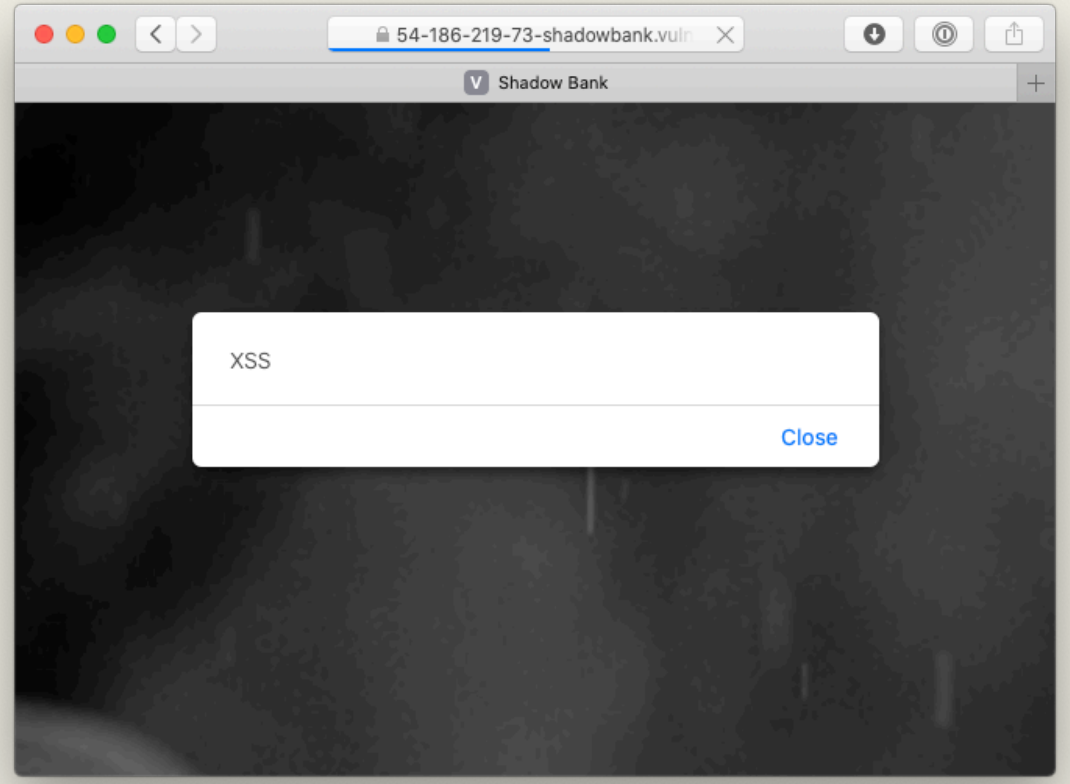
XSS & SQLI

Time to get real

Cross Site Scripting (XSS)

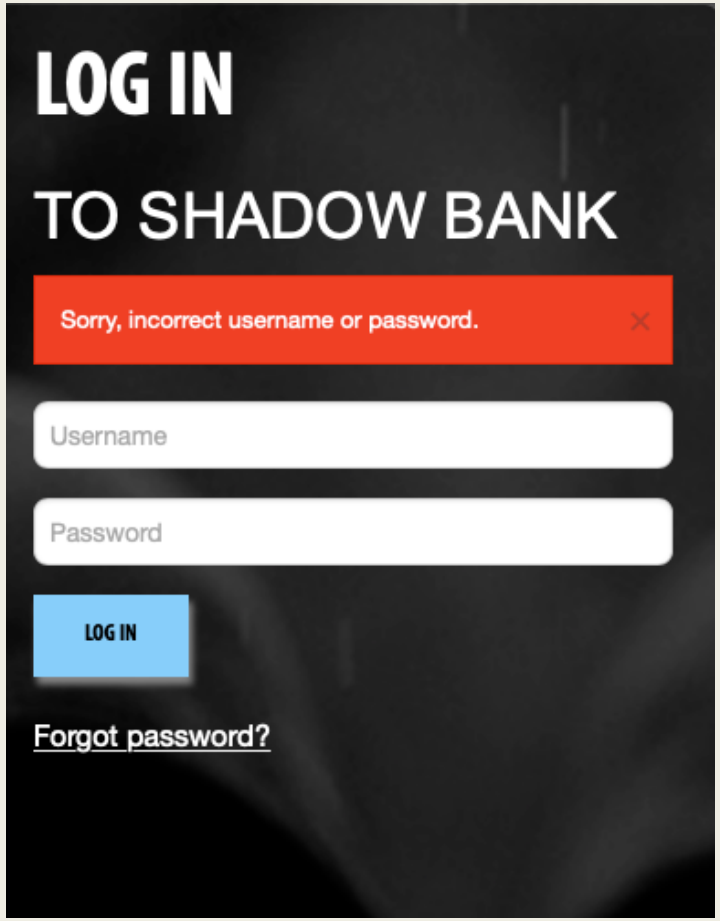
Mixing Code and Data using **control characters**
in the webpage

- Try this anywhere you control a value on the page
  - *HTML*
  - *JavaScript*
  - *Headers*
- How is your input being encoded?
- Test Cases
  - *Change your input*
  - *Try <marquee>*
  - *Try <script>alert('XSS')</script>*



What Can You Do with XSS?

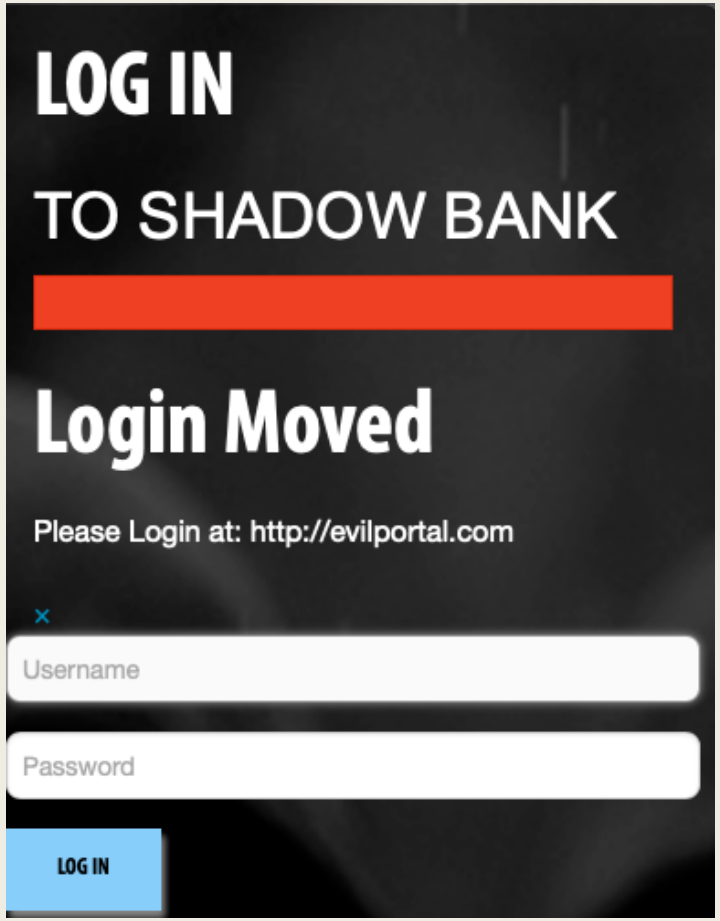
loginError.action?errorMsg=Sorry%2C+incorrect+username+or+password.



What Can You Do with XSS?

loginError.action?errorMsg=

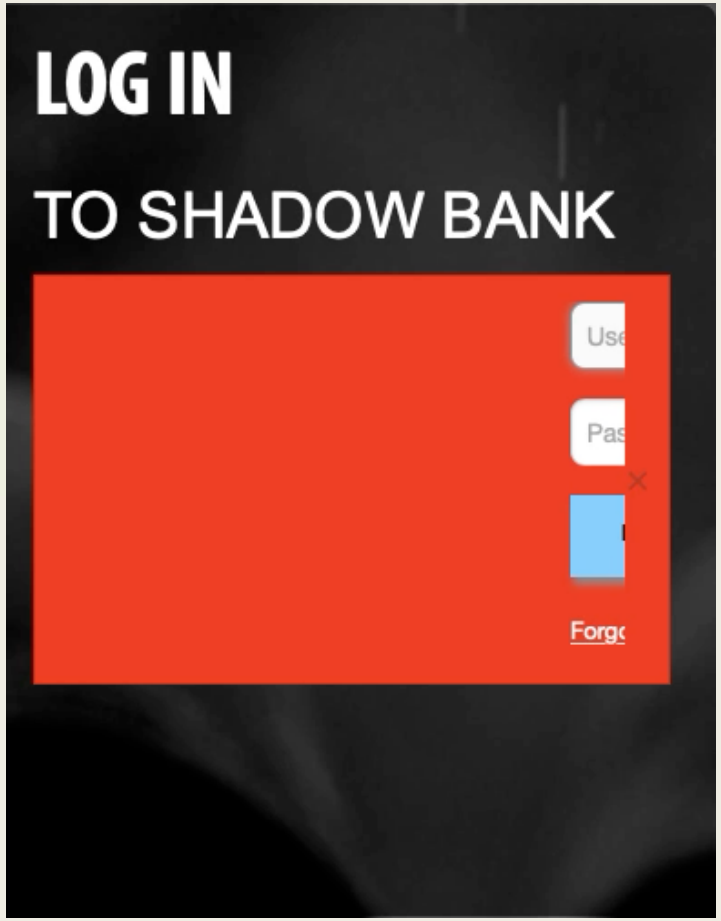
<h1>Login Moved</h1> Please Login at: http://evilportal.com



What Can You Do with XSS?

loginError.action?errorMsg=

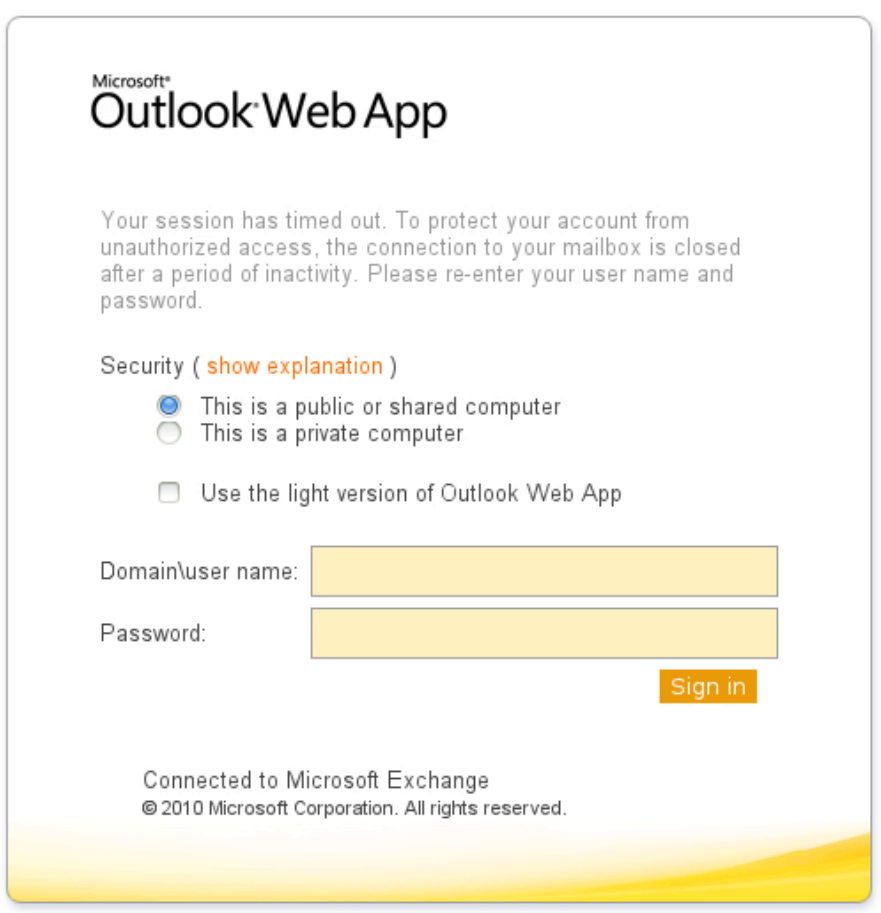
<marquee>



What Can You Do with XSS?

loginError.action?errorMsg=

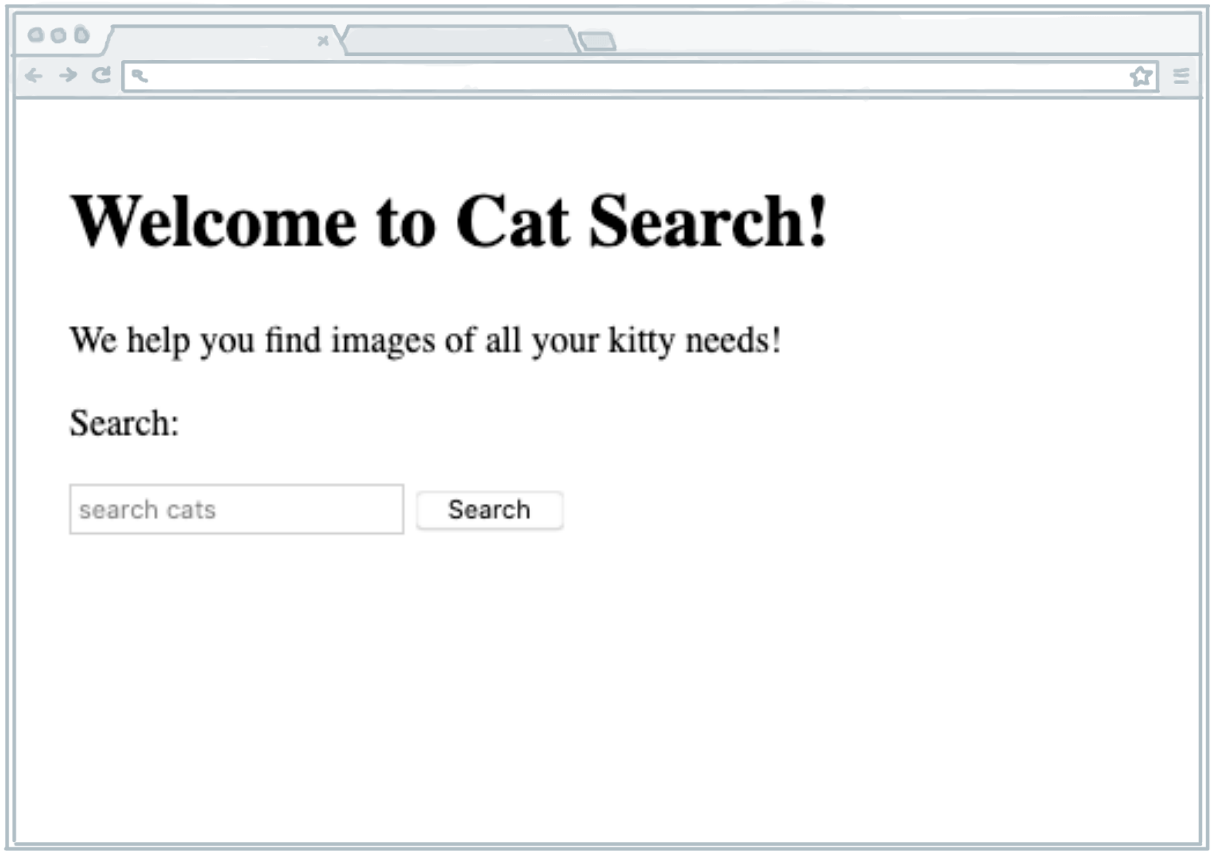
<script>document.location='http://evilhacker.org'</script>



When is XSS Possible?

Whenever a page includes unsanitized user input

```
<html>
<body>
<h1>Welcome to Cat Search!</h1>
<p>We help you find images of all your kitty needs!</p>
<p>Search:
<form>
  <input type='text' placeholder='search cats'></input>
  <input type='submit' value="Search">
</form>
</p>
</body>
</html>
```



When is XSS Possible?

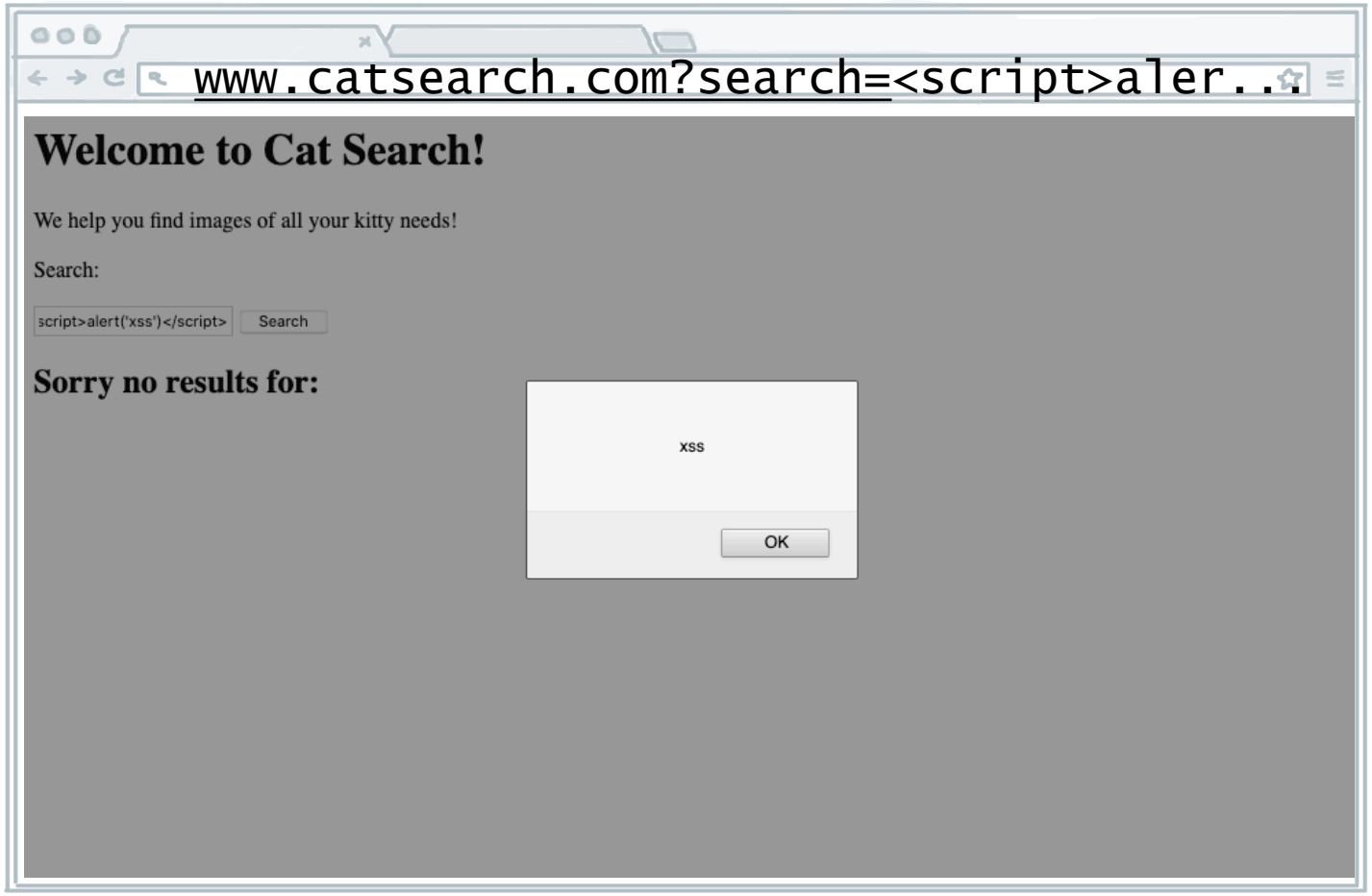
```html
<html>
<body>
<h1>Welcome to Cat Search!</h1>
<p>We help you find images of all your kitty needs!</p>
<p>Search:
<form>
  <input type='text' placeholder='search cats'></input>
  <input type='submit' value="Search">
</form>
</p>
<h2>Results for: fluffy</h2>
<p>
  <img src='cat1.jpg'>
  <img src='cat2.jpg'>
  <img src='cat3.jpg'>
</p>
</body>
</html>
```

When is XSS Possible?

```html
<html>
<body>
<h1>Welcome to Cat Search!</h1>
<p>We help you find images of all your kitty needs!</p>
<p>Search:
<form>
  <input type='text' placeholder='search cats'></input>
  <input type='submit' value="Search">
</form>
</p>
<h2>Sorry no results for: sadlfkjsadflkjsdaf</h2>
</body>
</html>
```

When is XSS Possible?

```html
<html>
<body>
<h1>Welcome to Cat Search!</h1>
<p>We help you find images of all your kitty needs!</p>
<p>Search:
<form>
  <input type='text' placeholder='search cats'></input>
  <input type='submit' value="Search">
</form>
</p>
<h2>Sorry no results for: <script>alert('xss')</script></h2>
</body>
</html>
```



SQL Injection

- **Mixing Code and Data** using **control characters** *in Database Queries*
- Try this on any input you think may use the database
  - *Textboxes, URL Parameters, dropdowns, hidden fields*
- Start small, build more complex SQL Queries to manipulate the database
- Test Cases
  - *Does ' Produce an error message?*
  - *Think about how to manipulate the SQL command*

```
SELECT * FROM USERS WHERE Username = 'joe' AND Password = 'P4S$WorD1';
```

Logging in with SQL Injection

Input Values	
Username	joe
Password	P4S$WorD1

SELECT * FROM USERS
WHERE Username = '**joe**' AND
Password = '**P4S$WorD1**';

Commentary:

Assuming correct username and password the user is logged in

Logging in with SQL Injection

Input Values	
Username	joe'
Password	P4S$WorD1

SELECT * FROM USERS
WHERE Username = '**joe**'' AND
Password = '**P4S$WorD1**';

```
com.fjordengineering.store.util.SecureSQLException
```

Commentary:

Errant single quote causes a parsing error.
Error returned to user.

Logging in with SQL Injection

Input Values	
Username	joe’#
Password	P4S$WorD1

SELECT * FROM USERS
WHERE Username = **'joe'#**' AND
Password = '**P4S$WorD1**';

```
Login Success: User = joe
```

Commentary:

Password check is commented out. Username is checked and attacker is logged in as ‘joe’

Logging in with SQL Injection

Input Values	
Username	joe’ OR 1=1 #
Password	P4S$WorD1

SELECT * FROM USERS
WHERE Username = '**joe**' OR 1=1 #'
AND Password = '**P4S$WorD1**';

Commentary:

Password check is commented out. Username is checked and attacker is logged in as ‘joe’

Everything after the # is disregarded

Logging in with SQL Injection

Input Values	
Username	joe' OR 1=1 #
Password	P4S$WorD1

```
SELECT * FROM USERS
WHERE Username = 'joe' OR 1=1;
```

Commentary:

Password check is commented out. Username is checked and attacker is logged in as 'joe'

```
SELECT * FROM USERS
WHERE Username = 'joe' OR TRUE;
```

1=1 is always TRUE, so we can replace that

Logging in with SQL Injection

Input Values	
Username	joe' OR 1=1 #
Password	P4S$WorD1

```
SELECT * FROM USERS
WHERE Username = 'joe' OR 1=1;
```

Commentary:

Password check is commented out. Username is checked and attacker is logged in as 'joe'

```
SELECT * FROM USERS
WHERE Username = 'joe' OR TRUE;
```

```
SELECT * FROM USERS
WHERE TRUE;
```

Anything OR TRUE is always TRUE

Logging in with SQL Injection

Input Values	
Username	joe' OR 1=1 #
Password	P4S$WorD1

```
SELECT * FROM USERS
WHERE Username = 'joe' OR 1=1;

SELECT * FROM USERS
WHERE Username = 'joe' OR TRUE;

SELECT * FROM USERS
WHERE TRUE;

SELECT * FROM USERS;
```

Commentary:

Password check is commented out. Username is checked and attacker is logged in as 'joe'

OR 1=1 # short circuits the entire where clause in this case

INJECTION FLAWS ALLOW AN ATTACKER TO INJECT THEIR OWN CODE INTO THE PROGRAM

BROKEN AUTHENTICATION

Check ID at the door

IS A HI-VIS VEST MORE POWERFUL THAN ID?

FREE MOVIES

ENTRANCE TO THE ZOO


COLDPLAY?
I wasn't a big fan of Coldplay before I saw them in hi-vis

Authentication Issues

- Many opportunities to make mistakes
  - *Not checking credentials properly*
  - *Not storing credentials properly*
  - *Not protecting authentication tokens properly*
  - *Loss of credentials*
  - *Password reuse*
  - *Phishing*
  - *Failure to use 2FA*
  - *Cookie issues*
  - *XSS*
  - *CSRF*
  - *…*

- Verify your users
- Protect their credentials
- Protect credential equivilents

PRIVILEGE ESCALATION

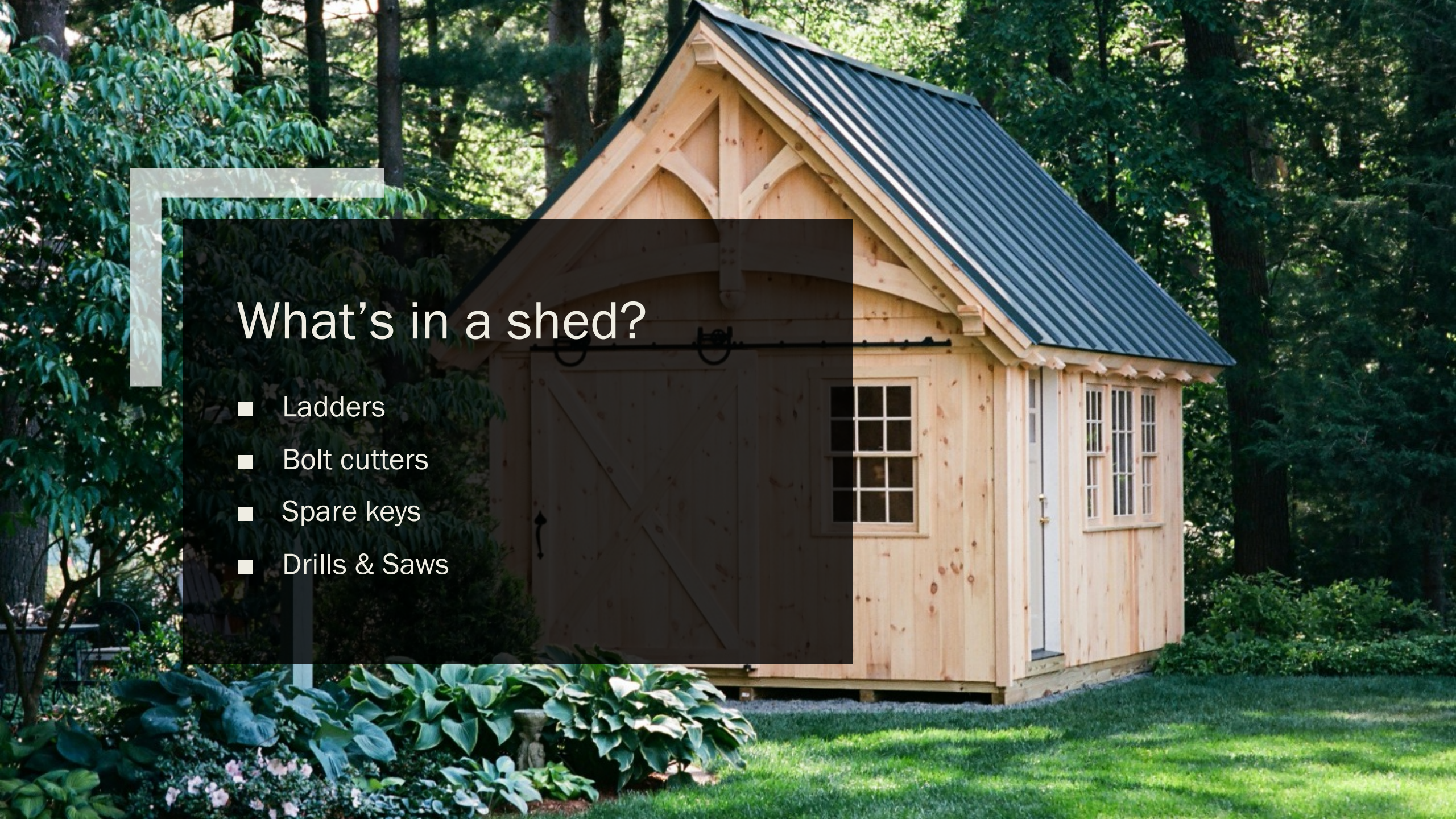
Can I steal your TV through your shed?

I want in here

I can get in here

What’s in a house?

- TV
- Computers
- Electronics
- Money



What’s in a shed?

- Ladders
- Bolt cutters
- Spare keys
- Drills & Saws

Start Here → Go Here

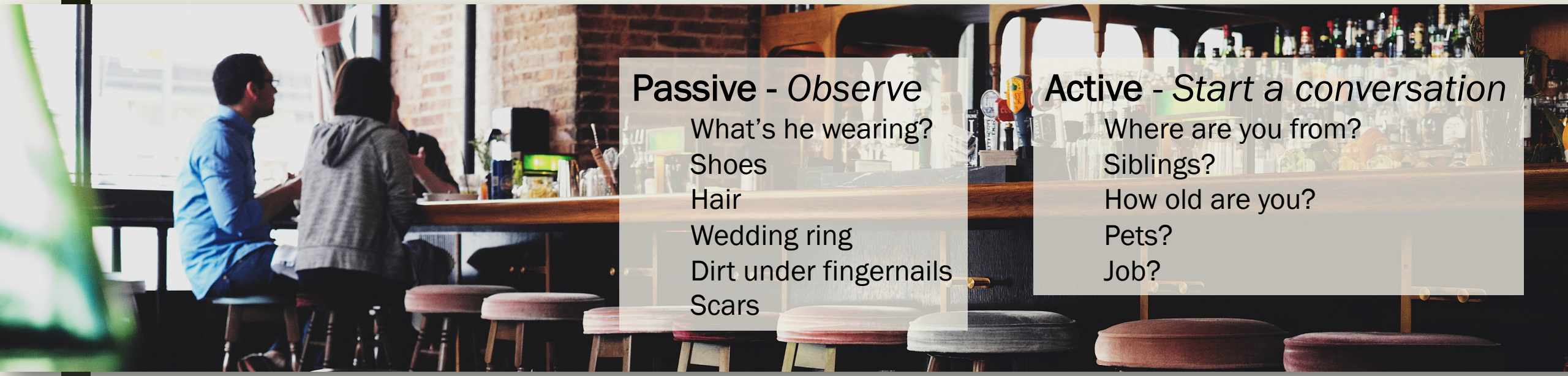
Horizontal vs. Vertical Escalation

- Horizontal Privilege Escalation
  - *Allows one user can access another user's data*
- Vertical Privilege Escalation
  - *Allows a user to increase their privilege level*
  - *Anonymous -> User*
  - *User -> Manager*
  - *Manager –> Administrator*

INFORMATION DISCLOSURE

I bet that guy is in sales, I can tell by his suit

A guy walks into a bar…



Passive - *Observe*

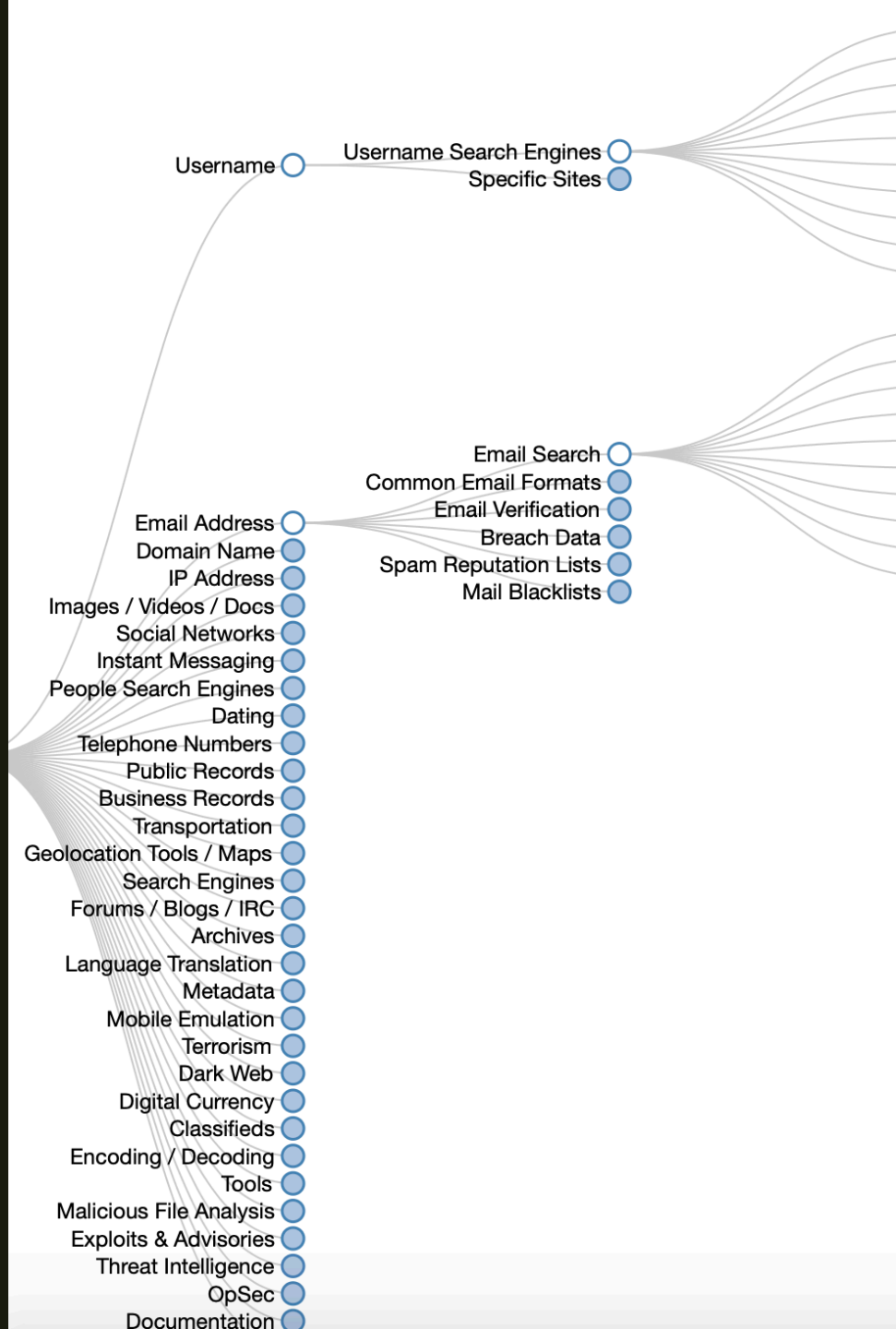
- What's he wearing?
- Shoes
- Hair
- Wedding ring
- Dirt under fingernails
- Scars

Active - *Start a conversation*

- Where are you from?
- Siblings?
- How old are you?
- Pets?
- Job?

Computers give away information all the time

- Hackers gather that information and use it against us every day
- Tools and Databases scan and collect this information for easy querying
- Our job is to protect this information



PARAMETER TAMPERING

Control the *data*

Control the *future*

Let’s find some deals!

- Peel off the tags from some Wonder Bread
- Apply tags to *fancy* bread!

ALWAYS BE NICE TO YOUR MILLENNIALS

Everything a computer does starts with input

Without input a computer will always do the same thing

Input filtering, processing, and blocking sets the stage for everything else

CONFIGURATION ERRORS

Don’t put the locks on the wrong side of the door

Doors, Windows, and Locks

- Installing a door can be difficult to do securely
- Installing a window so it locks automatically
- Don’t forget to lock your doors and windows
- Did you remember all your doors and windows?

Many software systems can be configured securely

- Most software systems don’t come secure by default
- Insecure use of existing components
  - *The door is installed poorly*
- Insecure configuration of components
  - *The lock is misconfigured*
- Insecure defaults are used
  - *The lock has a reused key or default keycode*

MAKING SENSE OF SO MANY ISSUES

Grouping by Threat, Weakness, Attack Pattern

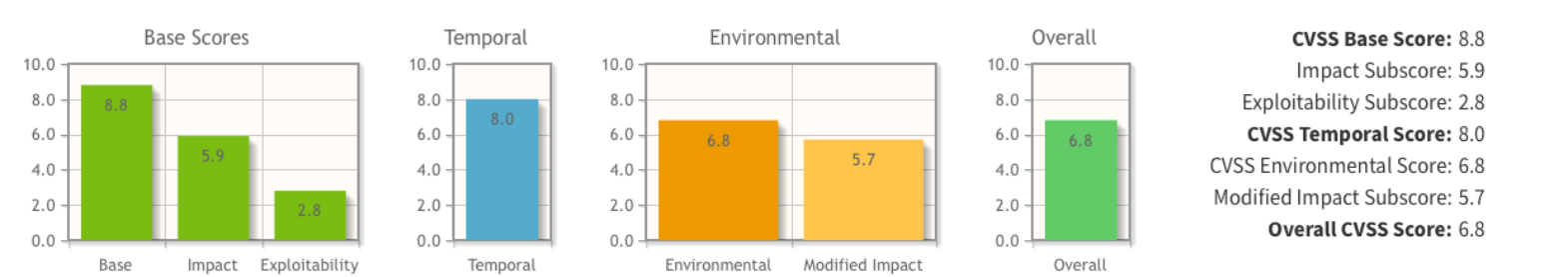
OWASP, CWE, CAPEC, and More!

OWASP
Open Web Application Security Project

- Most famous for the OWASP Top 10
- A semi-updated list of the most critical web application security risks
  - *2004*
  - *2007*
  - *2010*
  - *2013*
  - *2017*
- De-facto standard for basic web application testing

CVSS
Common Vulnerability Scoring System

- A vulnerability scoring calculator
- Included with all our PRs
- Considers Exploitability and Impact metrics
- Can be extended to Temporal and Environmental Metrics
- Exploitability
  - *Attack vector*
  - *Attack complexity*
  - *Privileges Required*
  - *User Interaction*
  - *Scope*
- Impact Metrics
  - *Confidentiality Impact*
  - *Integrity Impact*
  - *Availability Impact*



[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10
[11]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.47
[12]	CWE-787	Out-of-bounds Write	11.08
[13]	CWE-287	Improper Authentication	10.78
[14]	CWE-476	NULL Pointer Dereference	9.74
[15]	CWE-732	Incorrect Permission Assignment for Critical Resource	6.33
[16]	CWE-434	Unrestricted Upload of File with Dangerous Type	5.50
[17]	CWE-611	Improper Restriction of XML External Entity Reference	5.48
[18]	CWE-94	Improper Control of Generation of Code ('Code Injection')	5.36
[19]	CWE-798	Use of Hard-coded Credentials	5.12
[20]	CWE-400	Uncontrolled Resource Consumption	5.04
[21]	CWE-772	Missing Release of Resource after Effective Lifetime	5.04
[22]	CWE-426	Untrusted Search Path	4.40

CWE
Common Weakness Enumeration

- An enumeration of common software weaknesses (think flaws or vulnerabilities)
- CWE Top 25: Recently updated in 2019, methodology changed
- Great for mapping to internal tools
- Important reference
- *Not a test plan*

MITRE | ATT&CK™

- New: Released 2018
- Confusingly named ATT&CK framework
- Focused on enterprise risk (think Attack Sim and Red Teaming)
- Partially maps to our Attack Sim and Red Teaming
- May map to our Cloud CMD+CTRL CyberRanges

ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact

CAPEC
Common Attack Pattern Enumeration and Classification

- An Exhaustive list of every attack possible against any system organized by Domain or Mechanism
- 517 total attack patterns
- Mostly academic
- Great for having a standard language for attacks
- Great for mapping attacks to internal resources
- *Not a test plan*

3000 - Domains of Attack

- Software - *(513)*
  - Exploitation of Trusted Credentials - *(21)*
  - Exploiting Trust in Client - *(22)*
  - Forced Deadlock - *(25)*
  - Leveraging Race Conditions - *(26)*
  - Fuzzing - *(28)*
  - Manipulating User State - *(74)*
  - Man in the Middle Attack - *(94)*
  - Brute Force - *(112)*
  - API Manipulation - *(113)*
  - Authentication Abuse - *(114)*
  - Authentication Bypass - *(115)*
  - Excavation - *(116)*
  - Interception - *(117)*
  - Privilege Abuse - *(122)*
  - Buffer Manipulation - *(123)*
  - Shared Data Manipulation - *(124)*
  - Flooding - *(125)*
  - Pointer Manipulation - *(129)*
  - Excessive Allocation - *(130)*
  - Resource Leak Exposure - *(131)*
  - Parameter Injection - *(137)*
  - Content Spoofing - *(148)*
  - Identity Spoofing - *(151)*
  - Input Data Manipulation - *(153)*
  - Resource Location Spoofing - *(154)*
  - Infrastructure Manipulation - *(161)*
  - File Manipulation - *(165)*
  - Footprinting - *(169)*
  - Action Spoofing - *(173)*
  - Code Inclusion - *(175)*
  - Configuration/Environment Manipulation - *(176)*
  - Software Integrity Attack - *(184)*
  - Reverse Engineering - *(188)*
  - Functionality Misuse - *(212)*
  - Fingerprinting - *(224)*
  - Sustained Client Engagement - *(227)*
  - Privilege Escalation - *(233)*
  - Resource Injection - *(240)*
  - Code Injection - *(242)*
  - Command Injection - *(248)*
  - Protocol Manipulation - *(272)*
  - Information Elicitation - *(410)*
  - Modification During Manufacture - *(438)*
  - Malicious Logic Insertion - *(441)*
  - Contaminate Resource - *(548)*
  - Local Execution of Code - *(549)*
  - Functionality Bypass - *(554)*
  - Object Injection - *(586)*
  - Traffic Injection - *(594)*
  - Obstruction - *(607)*
- Hardware - *(515)*
- Communications - *(512)*
- Supply Chain - *(437)*
- Social Engineering - *(403)*
- Physical Security - *(514)*