

Secure Coding

UBNetDef, Fall 2023
Week 15

Presenters:
Dikshit Khandelwal [DK]
Kyle Lemma

What is secure coding?

- Secure coding is the practice of writing code for systems, applications, and web services in such a way as to ensure the confidentiality, integrity, and availability of information and services.
- It's a proactive approach to ensure software is built to be as impervious to attacks as possible.



Importance of Secure Coding

- It is the practice of writing software in a way that guards against the introduction of vulnerabilities and protects against exploitation by malicious actors.



Confidentiality: Secure coding is fundamental in maintaining confidentiality. It involves implementing security measures in software to protect sensitive data from unauthorized access and breaches. Encryption, access controls, and secure authentication mechanisms are key examples.



Integrity: It ensures the integrity of data by preventing unauthorized modifications. Secure coding practices like input validation, error handling, and secure database interactions safeguard data accuracy and consistency.



Availability: Secure coding also contributes to the availability of systems and data. By mitigating risks like denial-of-service attacks and ensuring robust error handling, it helps maintain reliable access to resources when needed.

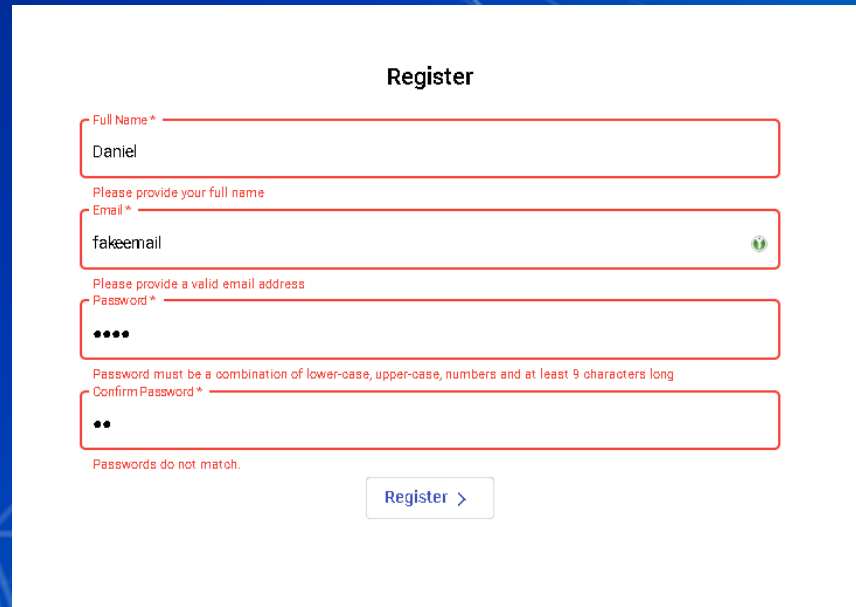
Principles of Secure Coding

- Principles of Secure Coding are fundamental rules and practices that software developers follow to make sure their software is safe from security threats and attacks.
- These principles include things like checking user input, protecting data, and controlling who can access the software.
- They help create software that is less likely to be hacked or have security problems.



1. Input Validation

- Input validation is the practice of checking and validating all user inputs to ensure they meet expected criteria.
- It prevents malicious inputs that can exploit vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), and command injection.



The image shows a registration form titled "Register" with the following fields and errors:

- Full Name ***: Input "Daniel". Error: "Please provide your full name".
- Email ***: Input "fakeemail". Error: "Please provide a valid email address".
- Password ***: Input masked with "••••". Error: "Password must be a combination of lower-case, upper-case, numbers and at least 9 characters long".
- Confirm Password ***: Input masked with "••". Error: "Passwords do not match".

A "Register >" button is located at the bottom of the form.

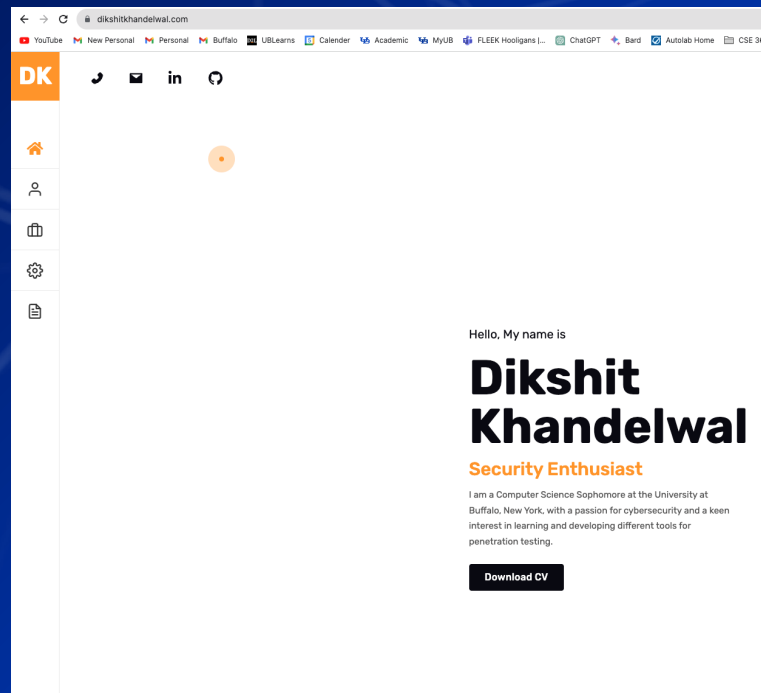
Input Validation Importance

- **Prevent Attacks:** Input validation stops attackers from injecting harmful data into your application.
- **Data Integrity:** It ensures that the data your application processes is safe

Name:	<input type="text" value="John Doe"/>
Credit card number:	<input type="text" value="1234-5678-2300-9000"/>
E-Mail:	<input type="text" value="<script>alert('This is XSS')</script>"/>
Phone number:	<input type="text" value="1234567"/>
Address:	<input type="text" value="1337 Make Believe Ln"/>
<input type="button" value="update"/>	

2. Output Encoding

- Output encoding ensures that data sent to users is safe and can't be used for malicious purposes. It prevents attackers from injecting harmful content like XSS attacks.



```
<div>
<h6>Hello, My name is</h6>
<h1>Dikshit Khandelwal</h1>
  <p>Security Enthusiast</p>
  <p>I am a Computer Science Sophomore at the
  University at Buffalo, New York, with a passion for cybersecurity
  and a keen interest in learning and developing
  different tools for penetration testing.</p>
</div>
```

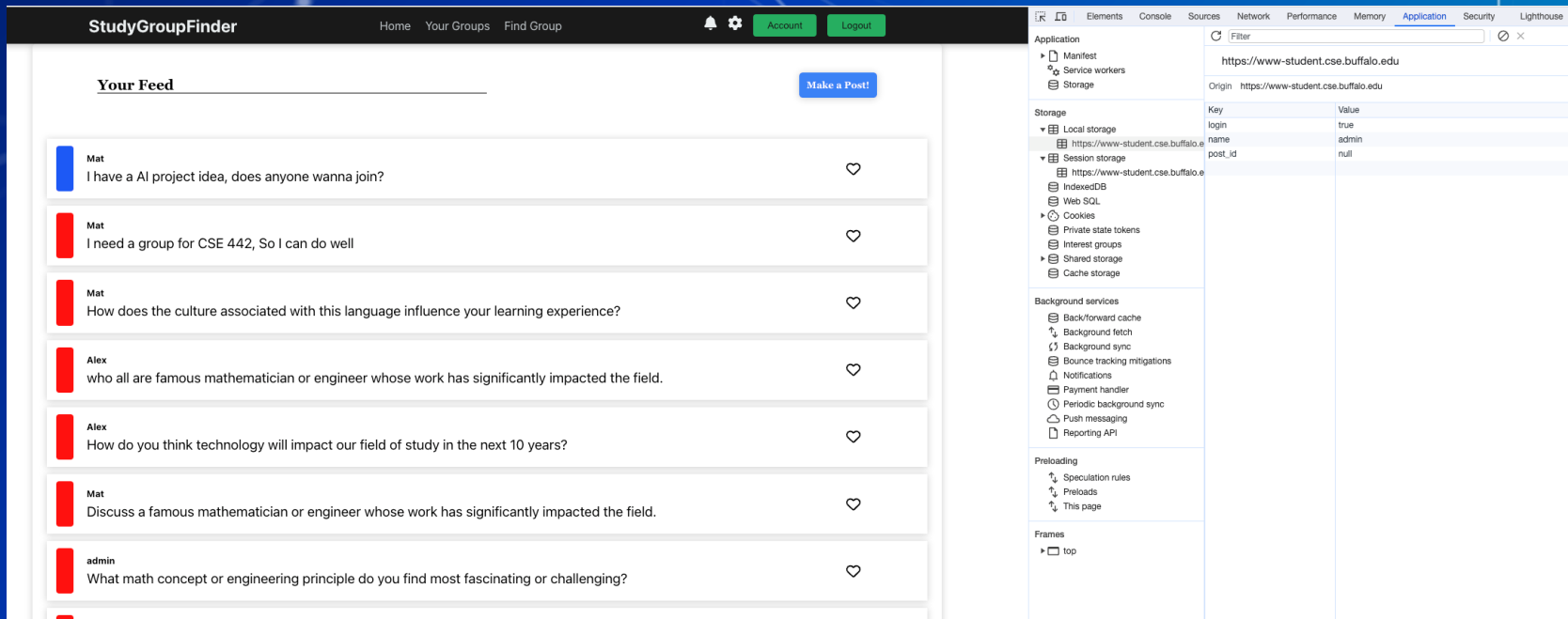

Output Encoding Importance:

- **Prevent XSS:** Output encoding safeguards your application against Cross-Site Scripting (XSS) attacks.
- **Protect Users:** It ensures that user-generated content is displayed safely.

```
<div>
  <h6>Hello, My name is</h6>
  <h1 id="username">Dikshit Khandelwal</h1>
  <script>
    var userName = document.getElementById('username').innerText;
    document.write('<script>alert("Welcome, ' + userName + '")</script>');
  </script>
  <p>Security Enthusiast</p>
  <p>I am a Computer Science Sophomore at the
    University at Buffalo, New York, with a passion for cybersecurity
    and a keen interest in learning and developing
    different tools for penetration testing.</p>
</div>
```


3. Authentication

- In secure coding, authentication involves writing code that accurately verifies user identities. The code written to handle user logins, session management, and identity verification must be secure against various attack vectors.
- <https://www-student.cse.buffalo.edu/CSE442-542/2023-Fall/cse-442e/#/>



Authentication Importance

■ Use/Importance:

- **Access Control:** It prevents unauthorized access to sensitive resources and actions.
- **User Trust:** Proper authentication and password management build user trust in your application.



4. Session Management

- Secure session management protects user session tokens during an active session.

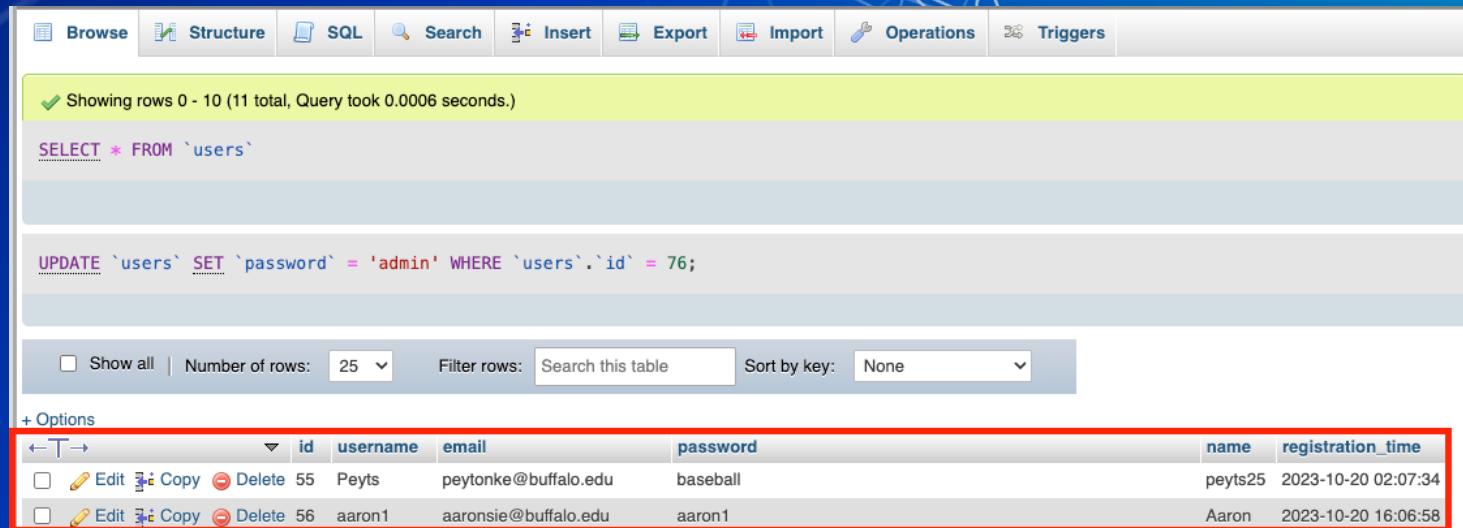
4. Session Management

■ Use/Importance:

- **Prevent Session Hijacking:** It prevents attackers from stealing session tokens and taking over user sessions.
- **Data Privacy:** Secure session management ensures that user data remains private.

5. Cryptographic Practices

- Access Cryptography in software development involves using algorithms and cryptographic keys to encrypt and decrypt data.
- It's essential for protecting sensitive data like user credentials, personal information, and financial transactions, both at rest (stored data) and in transit (data being transmitted).



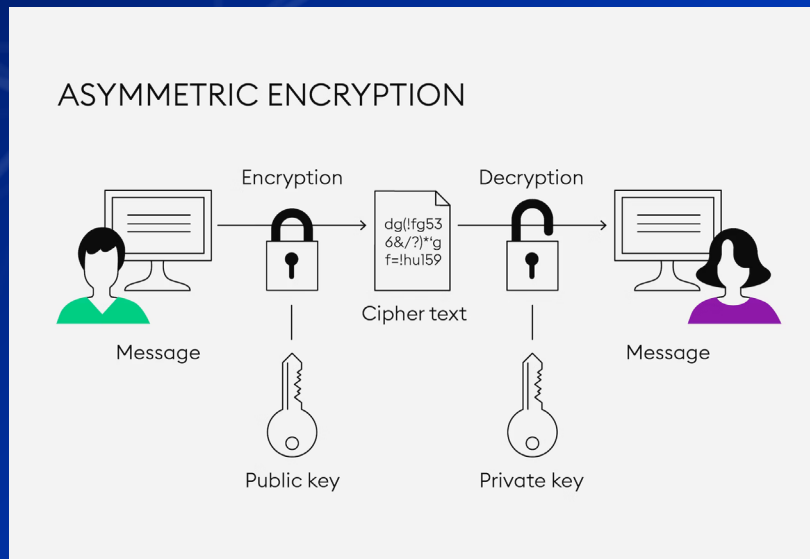
The screenshot shows a database management tool interface with a menu bar (Browse, Structure, SQL, Search, Insert, Export, Import, Operations, Triggers) and a status bar indicating 11 rows shown. The SQL editor contains two queries: a SELECT statement and an UPDATE statement. Below the editor are controls for showing all rows, number of rows (25), filter rows, and sort by key (None). A table of user data is displayed at the bottom, with a red border around the data rows.

	id	username	email	password	name	registration_time
<input type="checkbox"/>	55	Peyts	peytonke@buffalo.edu	baseball	peyts25	2023-10-20 02:07:34
<input type="checkbox"/>	56	aaron1	aaronsie@buffalo.edu	aaron1	Aaron	2023-10-20 16:06:58

Cryptographic Practices Importance

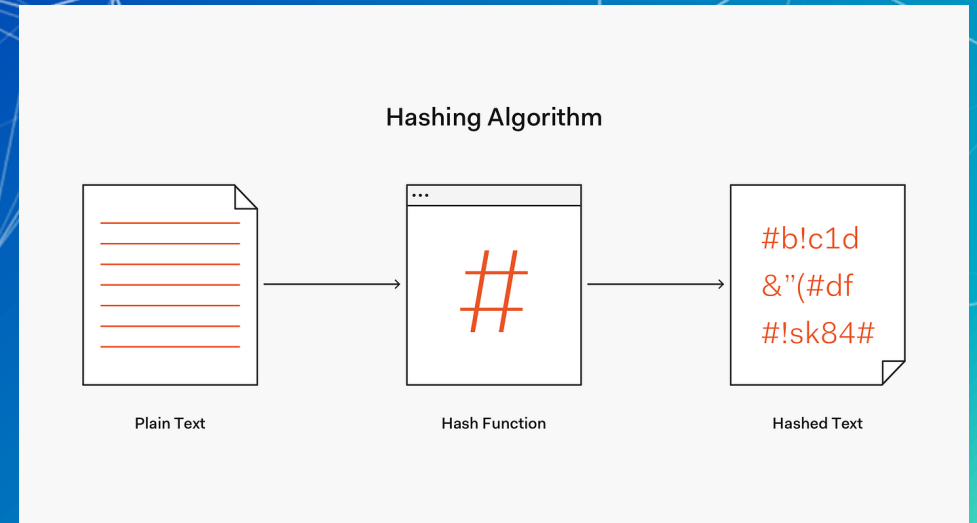
Confidentiality

- Cryptography in software development involves using algorithms and cryptographic keys to encrypt and decrypt data to maintain the confidentiality



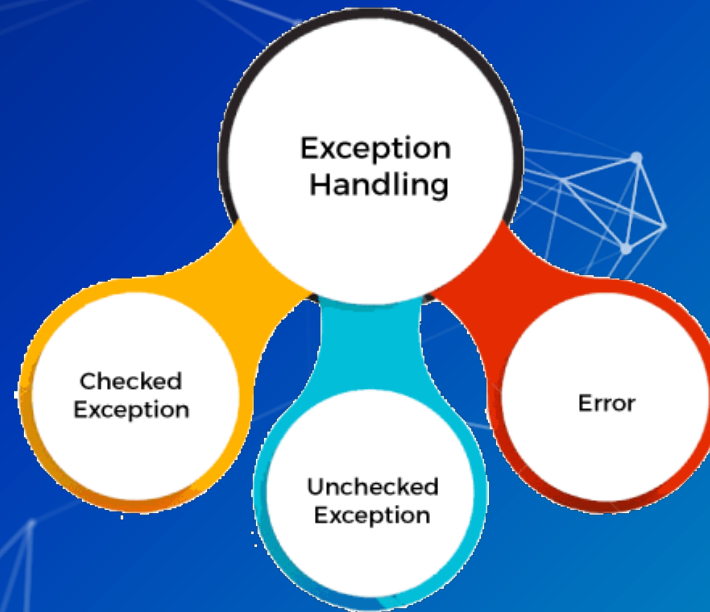
Integrity

- Use cryptographic hashing for data integrity checks and password storage. Implement algorithms like SHA-256 or SHA-3.



6. Error Handling and Logging

- Proper error handling ensures that errors do not expose sensitive information, and secure logging records of important security events.



Confidentiality

Prevent Data Leakage: It prevents error messages from revealing sensitive information.

Reset your password

User not found

Email

Email Reset Link

Don't have an account? [Register](#)

Availability

Ensure Continuous Service: Proper error handling maintains application uptime by gracefully managing exceptions, thus preserving the availability of the software for users.

```
a.py
1 num1 = 10
2 num2 = 0
3 result = num1 / num2
4 print("The result is", result)
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Ⓞ (base) dikshit@Dikshits-MacBook-Pro ~ % /Users/dikshit/anaconda3/bin/python /Users/dikshit/a.py
Traceback (most recent call last):
  File "/Users/dikshit/a.py", line 3, in <module>
    result = num1 / num2
            ~~~~~
ZeroDivisionError: division by zero
Ⓞ (base) dikshit@Dikshits-MacBook-Pro ~ %
```


Effective Error handling

Confidentiality

RESET PASSWORD

If the email address you provided exists in our records, you'll receive an email to reset your password.

Please check your inbox and follow the instructions provided.

No email? [Resend](#)

Availability

```
a.py
1  try:
2      num1 = 10
3      num2 = 0
4      result = num1 / num2
5      print("The result is", result)
6  except ZeroDivisionError:
7      print("Error: Cannot divide by zero.")
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● (base) dikshit@Dikshits-MacBook-Pro ~ % /Users/dikshit/anaconda3/bin/python /Users/dikshit/a.py
Error: Cannot divide by zero.
○ (base) dikshit@Dikshits-MacBook-Pro ~ % █
```

Effective Logging

The background features a complex, abstract geometric pattern of white lines and dots. The pattern consists of interconnected lines forming various polygons and shapes, with small white dots at the vertices. The background color transitions from a deep blue on the left to a lighter, teal-green on the right.

Some additional Principles

1. Data Protection
2. Communication Security
3. System Configuration
4. Database Security
5. File Management
6. Memory Management
7. General Coding Practices
8. and more

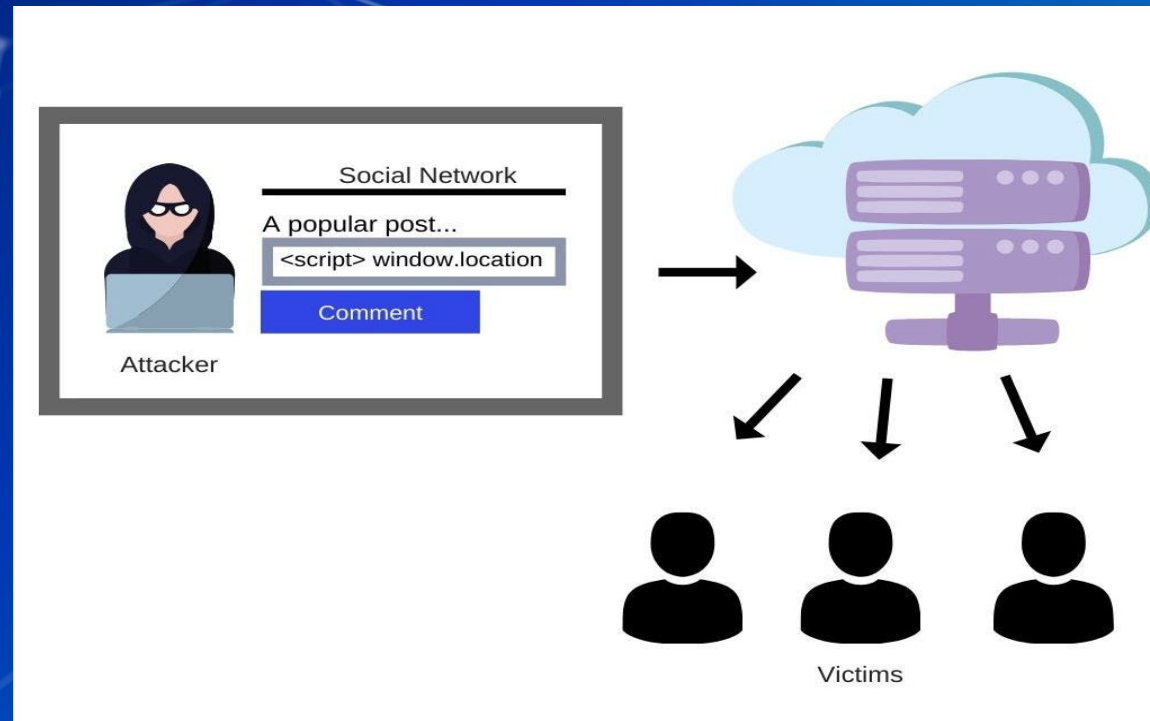
OWASP

- The Open Web Application Security Project (OWASP) Top 10 Web Application Security Vulnerabilities is a list of the most common and critical security weaknesses in web applications. The list is updated annually and is based on the input of security experts from around the world.



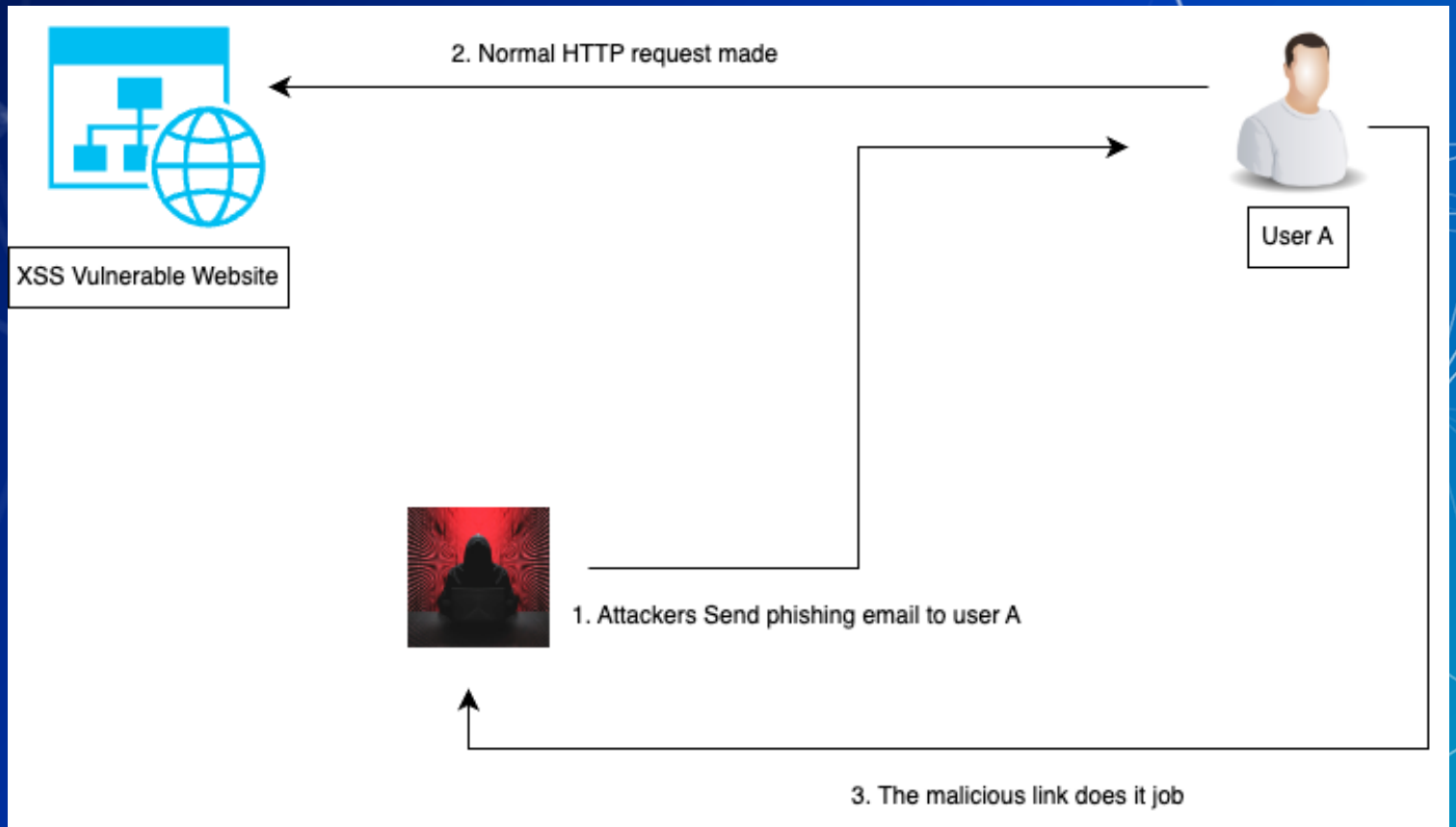
1. Cross-site scripting (XSS)

- XSS vulnerabilities allow attackers to inject malicious code into an application's web pages. This can allow the attacker to steal sensitive data, redirect users to malicious websites, or take control of the user's browser.





Demo



Prevent XSS

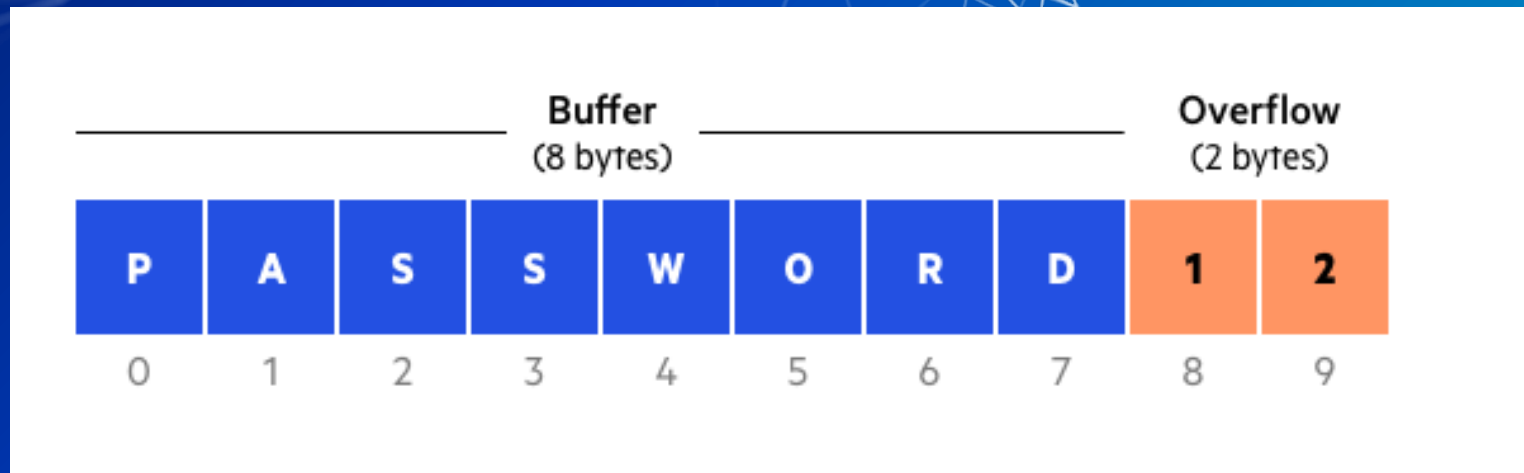
- 1. Data Validation:** Input validation to ensure that only properly formatted data is entered into the system.



- 2. Output Encoding:** When displaying user input or data from untrusted sources, it's important to encode it to prevent any embedded scripts from executing.

Buffer Overflow

- A buffer overflow occurs when the volume of data exceeds the storage capacity of a memory buffer
- Ex.) If a buffer is of size 8 and the data stored into the buffer is 10 there would be a buffer overflow



Preventing Buffer Overflow

- Stack Smashing Tools
- Canary Values
- Using Safe Commands (Specifying Data Lengths)
- Address Randomization

```
*** stack smashing detected ***: terminated
```

```
char *strcpy(char *dest, const char *src);
```

```
char *strncpy(char *dest, const char *src, size_t n);
```



Demo

```
1  #include <stdio.h>
2
3  void main()
4  {
5      int a = 25;
6      int b = 34;
7      char buffer[5];
8      printf("-----Before Gets-----\n");
9      printf("This is a: %d\n", a);
10     printf("This is b: %d\n", b);
11
12     printf("Enter a string\n");
13     gets(buffer);
14
15     printf("-----After Gets-----\n");
16
17     printf("This is decimal a: %d\n", a);
18     printf("This is decimal b: %d\n", b);
19     printf("This is hex a: %x\n", a);
20     printf("This is hex b: %x\n", b);
21
22
23 }
```


How to code securely

There are a number of things that developers can do to prevent common secure coding vulnerabilities. These include:

- **Input validation:** Validate all user input to ensure that it is safe and does not contain malicious code.
- **Output encoding:** Encode all output to prevent cross-site scripting (XSS) attacks.
- **Use strong passwords:** Use strong passwords and store them securely.
- **Testing**
- **More.....**



Testing Agenda

- Types of Testing
- Code Scanning
- What is Git?
- Git Requests

Types of Testing

- Static Testing – Analysis without running the code
- Dynamic Testing – Analysis while running the code with inputs and expecting resulting outputs
- Goal: Break the Code

Static Testing

- Analyzing source code without executing the program
- Manual Static Testing:
 - Inspections
 - Walkthroughs
 - Technical Reviews
- Automatic Static Testing
 - Control Flow Analysis
 - Data Flow Analysis
 - Failure Detection
- Beneficial because lots of time can be saved if defects are detected early on rather than during the later testing process

Examples of Automated Static Testing

- Veracode
 - Standalone Application
 - CI/CD Integration
- SonarQube
 - Standalone Application
 - CI/CD Integration

VERACODE

sonarqube

To benefit from more of SonarQube's features, set up analysis in your favorite CI.

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code

Since January 24, 2023
Started 23 hours ago

Overall Code

0

Bugs

Reliability

A

0

Vulnerabilities

Security

A

2

Security Hotspots

0.0% Reviewed

Security Review

E

1h 23min Debt

36 Code Smells

Maintainability

A



42.4%

Coverage on 83 Lines to cover

27

Unit Tests



0.0%

Duplications on 259 Lines

0

Duplicated Blocks

Dynamic Testing

- Testing code by running the program and providing inputs to then verify
- Setting up simple test cases that meet basic requirements
- Fuzzing – Entering in random incorrect data until something goes wrong
- Re-running test cases over and over to ensure reliability
 - May run tests in random orders so tests are not conditional
- Very beneficial to find defects with quality testing in real-world environments

Examples of Automated Dynamic Testing

- Astra PenTest
 - Used with testing:
 - Web and Mobile Applications, Cloud Infrastructure, API, and Networks
 - Standalone Application and CI/CD integration
- OWASP Zap
 - Used with testing:
 - Web application security testing, network ports, and API testing
 - Standalone Application and CI/CD integration



astra
pentest



OWASP
Zed Attack Proxy

ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	7
Informational	1

Alert Detail

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	https://secureaspnetcoremvc.azurewebsites.net/Home/Privacy
Method	GET
Parameter	X-Frame-Options
URL	https://secureaspnetcoremvc.azurewebsites.net/
Method	GET
Parameter	X-Frame-Options
Instances	2
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
CWE Id	16
WASC Id	15
Source ID	3

What is Git?



Git is a distributed version control system that tracks changes in any set of computer files



Used in organizations to have multiple software engineers be able to work on the same files



Examples:

GitHub

GitLab

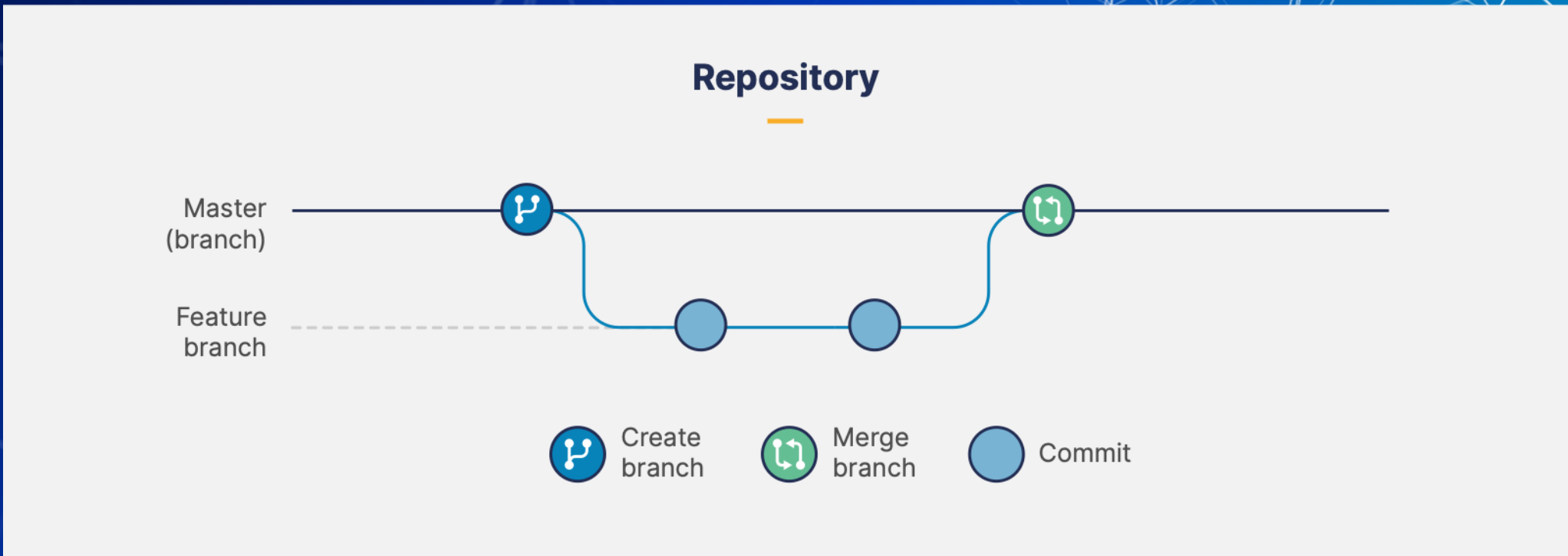
Azure DevOps (Newly Popular)



GitLab

Version Control

How does it work?



Types of Requests

Pull

The process of receiving any new code changes on a branch repository

Push

The process of sending new code changes to a branch repository

Merge

The process of combining one branch with another

Fork

A copy of an existing repository in which the new owner disconnects the codebase from previous committers