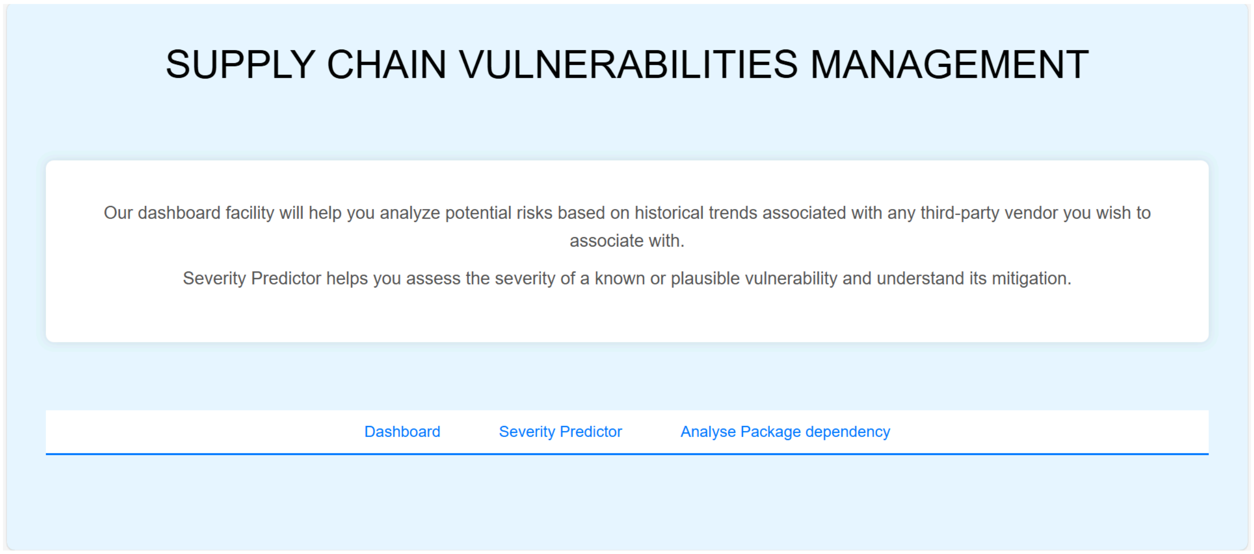
DOCUMENTATION OF THE IMPLEMENTATION:

The implementation involves 3 key components:

1. Vulnerability analysis dashboard
2. NPM audit visualiser
3. Vulnerability severity predictor model



SUPPLY CHAIN VULNERABILITIES MANAGEMENT

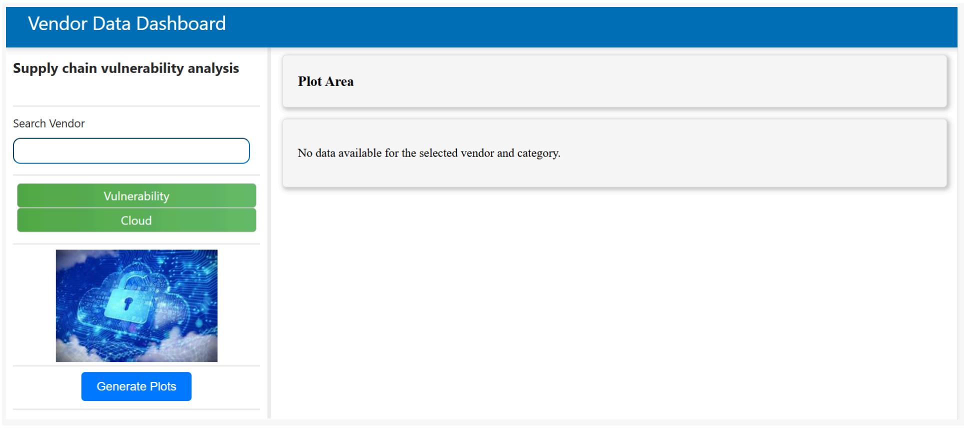
Our dashboard facility will help you analyze potential risks based on historical trends associated with any third-party vendor you wish to associate with.

Severity Predictor helps you assess the severity of a known or plausible vulnerability and understand its mitigation.

1.Vulnerability analysis dashboard

AN extensive dashboard based on data of vulnerabilities recorded in various third-party vendors through the years 2014-2022 is curated.

The user can input the name of any third-party vendor whose vulnerability trends they want to understand.
If the vendor specific details are not found in the database it displays a message-”No data available for the selected vendor and category.”



Vendor Data Dashboard

Supply chain vulnerability analysis

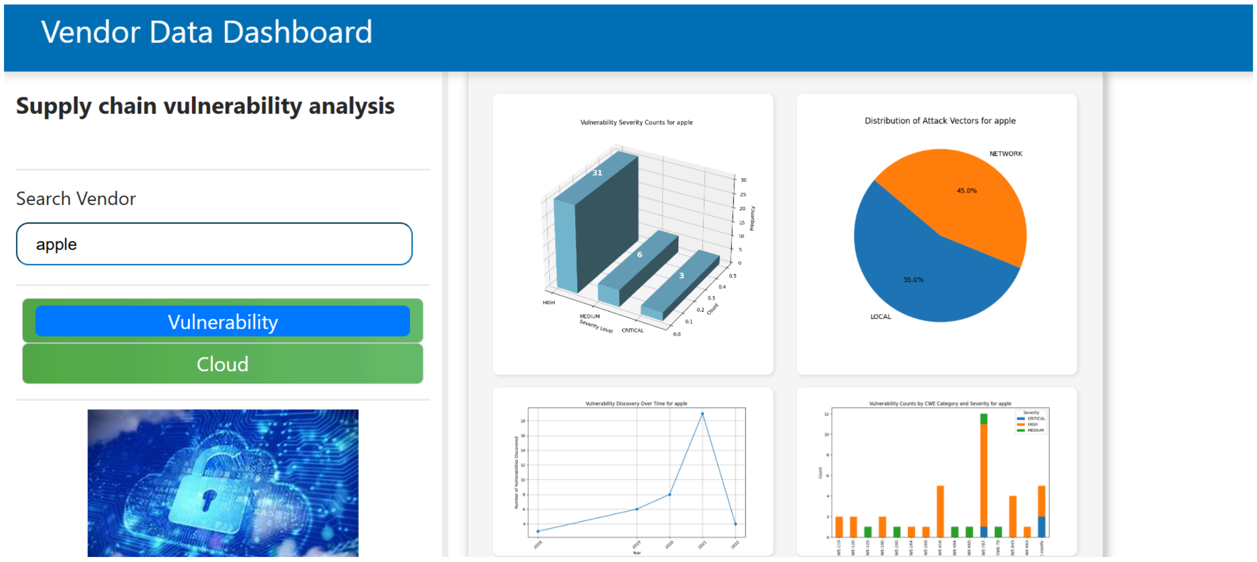
Plot Area

No data available for the selected vendor and category.

Different aspects of the vulnerability visualization are provided.The visualizations are broadly classified into 2 components-

- One is studying the vulnerability based on its type,severity ,CWE-category etc,i.e analyzing the vulnerability under a radar of possible characteristics and effects.
  The four graphs covered are:
  1. Severity-based bar chart
  2. Distribution of attack vectors-network or local
  3. Time graph of vulnerability count over the years
  4. Stacked bar chart on CWE-classification

  Considering apple as an example-



Vendor Data Dashboard

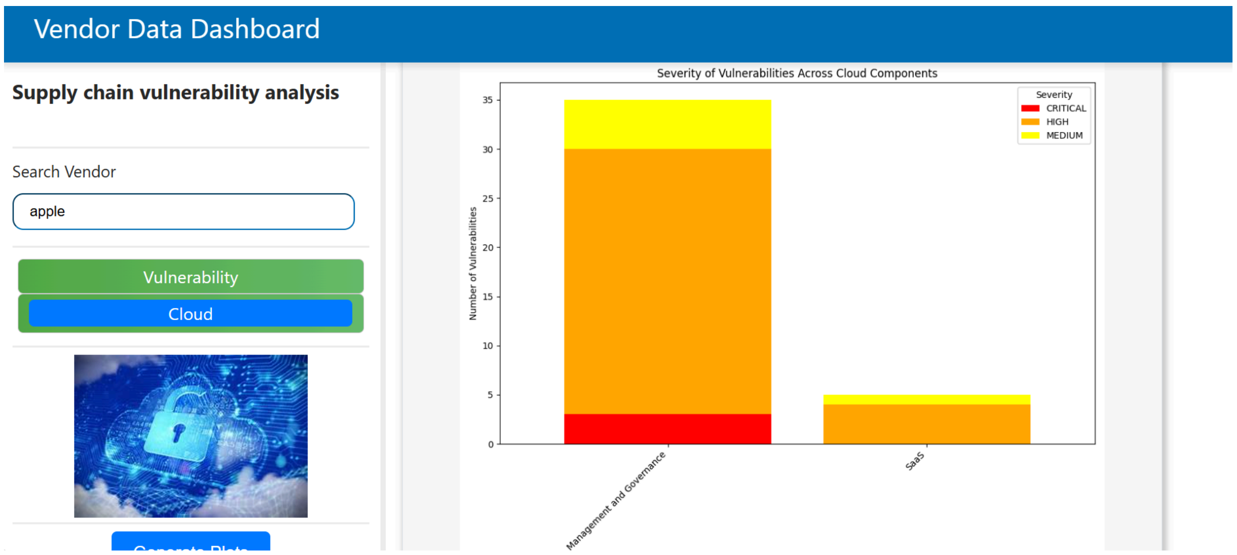
Supply chain vulnerability analysis

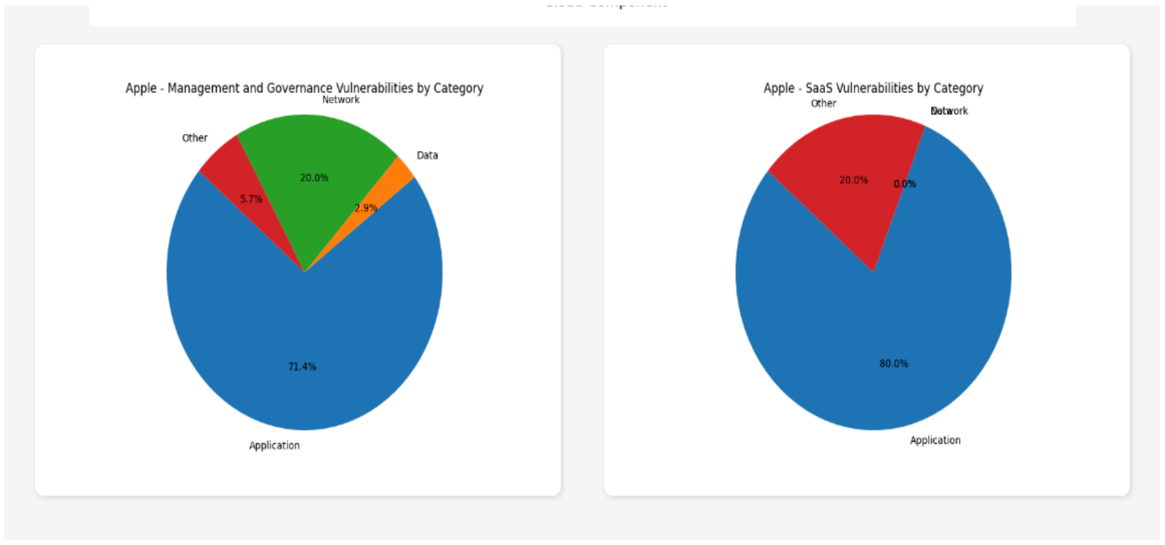
Search Vendor

apple

Vulnerability
Cloud
Vulnerability Severity Counts for apple
31
6
3
HIGH
MEDIUM
CRITICAL
Severity Level
Frequency
Distribution of Attack Vectors for apple
NETWORK
45.0%
55.0%
LOCAL
Vulnerability Discovery Over Time for apple
Year
Vulnerability Counts by CWE Category and Severity for apple
Severity
CRITICAL
HIGH
MEDIUM
Count

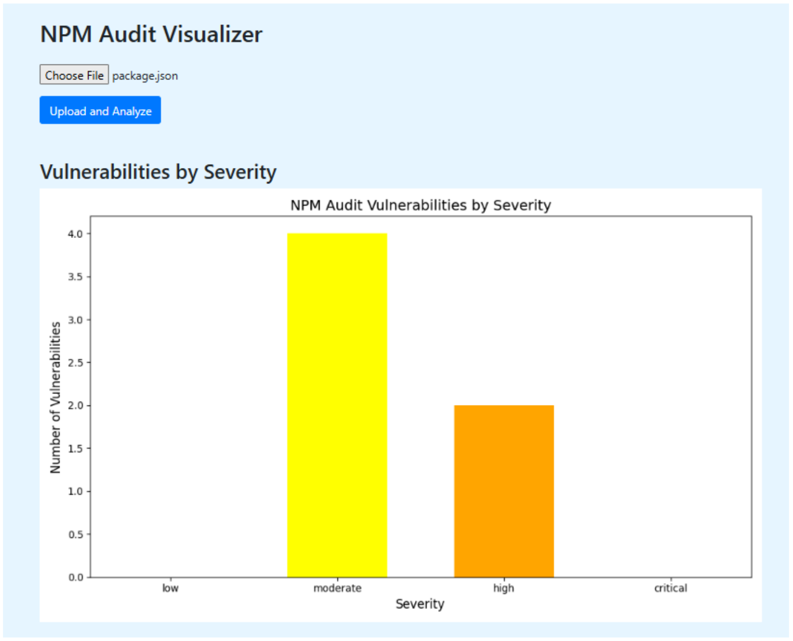
- The second one is cloud component based visualization which include:
  1. A stacked bar chart depicting classification of which cloud component is primarily affected by the attack-PaaS,SaaS,Management and Governance etc.
  2. Specific pie charts in the found category of cloud and a further classification within that domain based on data,network,application etc.

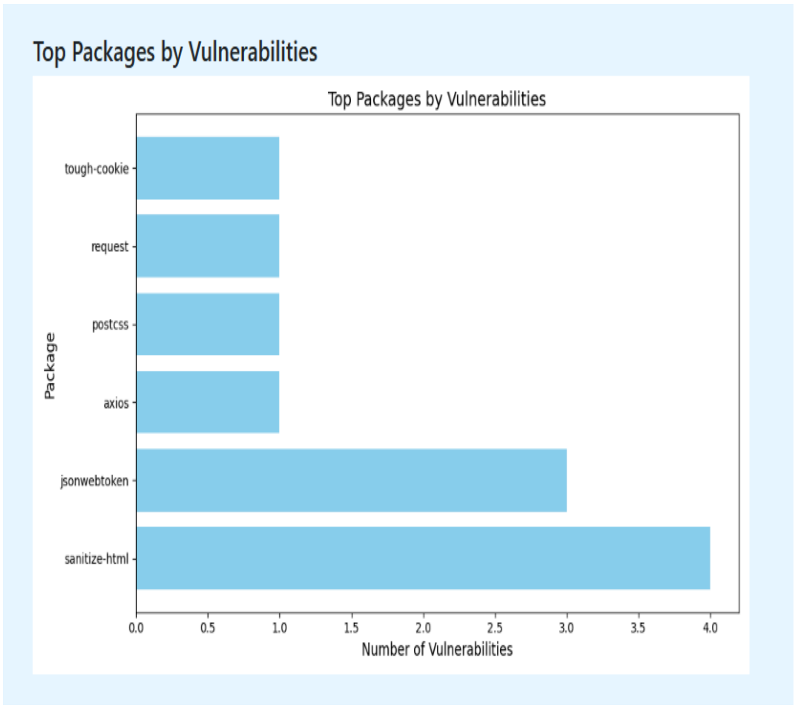

Vendor Data Dashboard
Supply chain vulnerability analysis
Search Vendor
apple
Vulnerability
Cloud
Severity of Vulnerabilities Across Cloud Components
Number of Vulnerabilities
0
5
10
15
20
25
30
35
Severity
CRITICAL
HIGH
MEDIUM
Management and Governance
SaaS

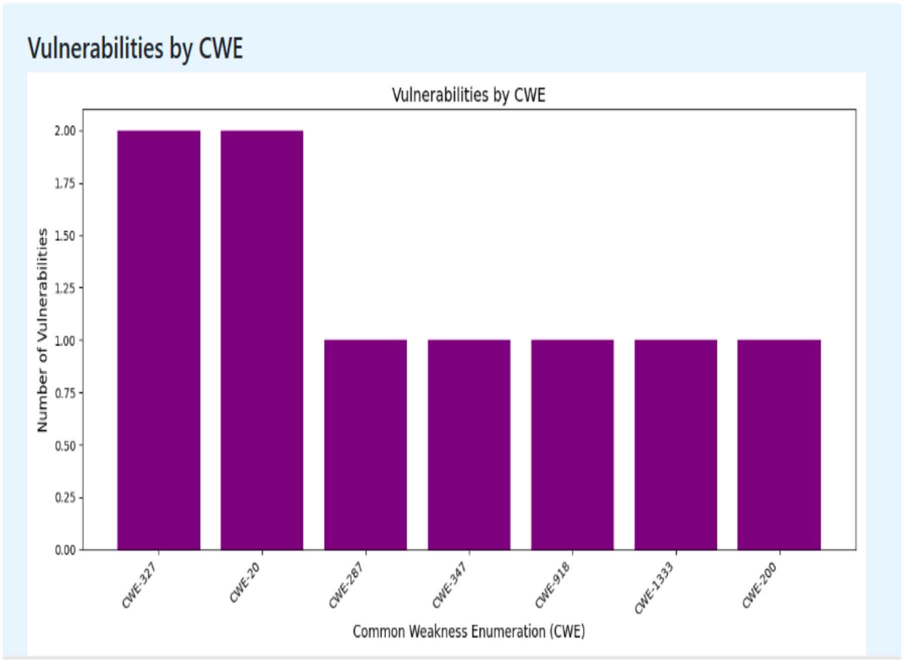


2.NPM visualizer

The feature allows to analyze any possible vulnerability present in the package.json files associated with Node.js projects,and provide corresponding visualizations helping a larger scope of understanding and effective mitigation:

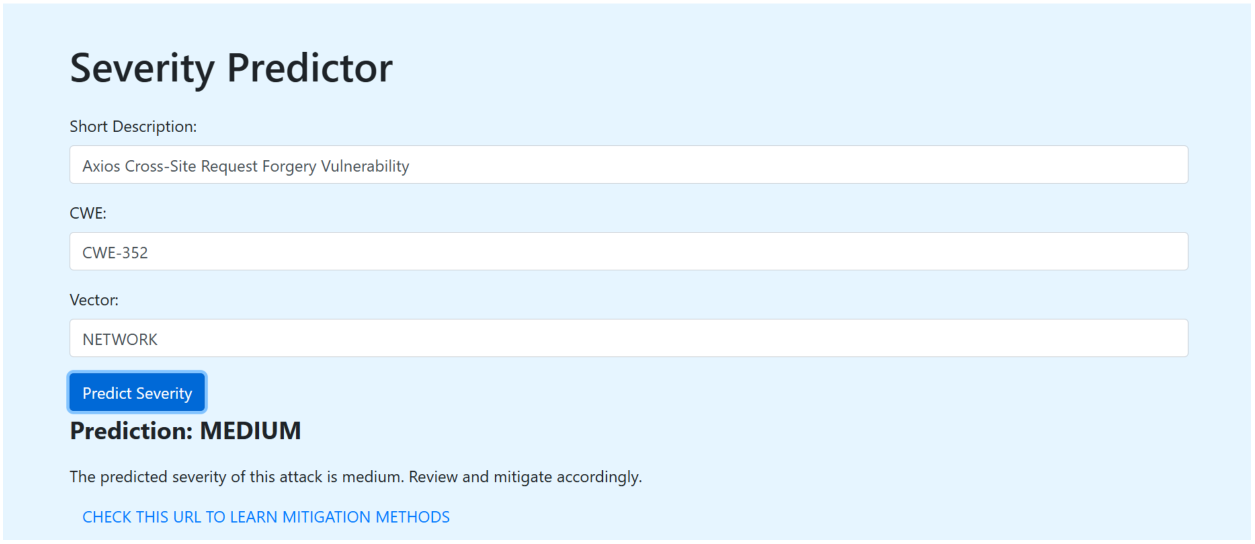

NPM Audit Visualizer
Choose File
package.json
Upload and Analyze
Vulnerabilities by Severity
NPM Audit Vulnerabilities by Severity
Number of Vulnerabilities
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
low
moderate
high
critical
Severity


Top Packages by Vulnerabilities
Top Packages by Vulnerabilities
Package
tough-cookie
request
postcss
axios
jsonwebtoken
sanitize-html
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Number of Vulnerabilities



3.Severity predictor

A SVC based ml model is designed to predict the severity of a given vulnerability based on three input parameters-
short description of the vulnerability,its CWE category and the attack vector



Severity Predictor

Short Description:

Axios Cross-Site Request Forgery Vulnerability

CWE:

CWE-352

Vector:

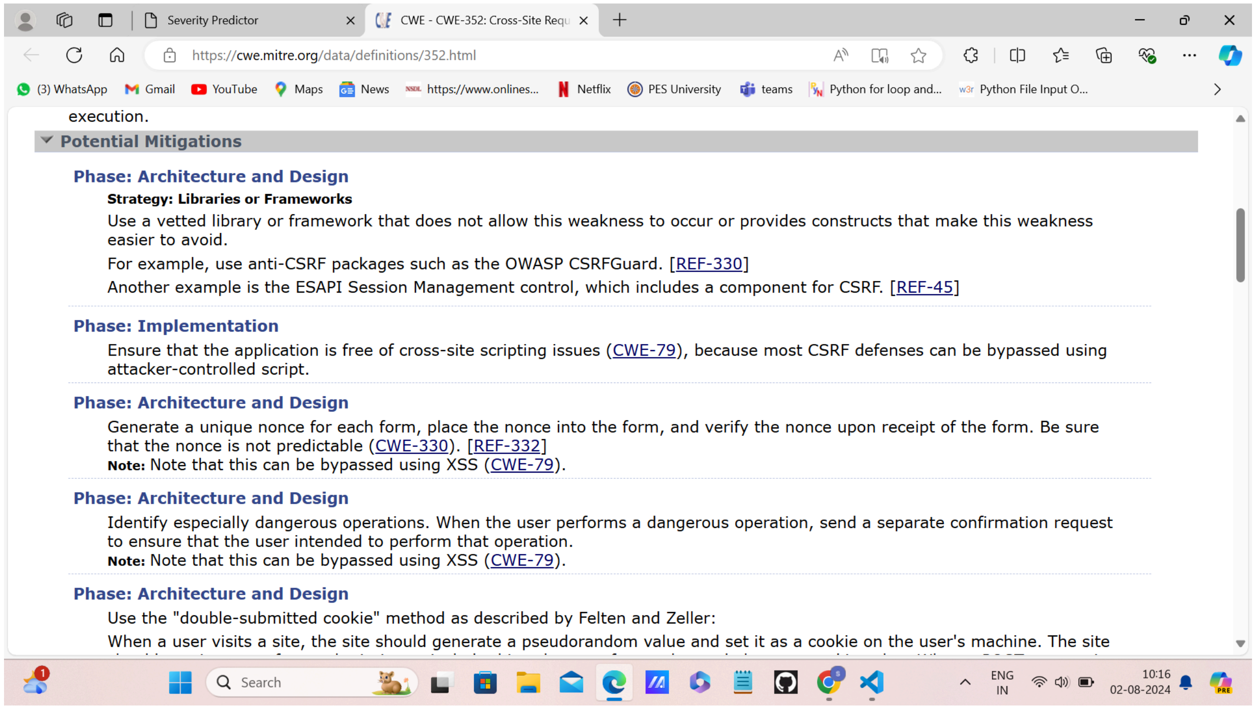
NETWORK

Prediction: MEDIUM

The predicted severity of this attack is medium. Review and mitigate accordingly.

CHECK THIS URL TO LEARN MITIGATION METHODS

Based on the input CWE ,a detailed webpage containing detection and mitigation strategies is made available(the url is the CWE specific url on the official CWE website) for the companies to use.



execution.

Potential Mitigations

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard. [REF-330]

Another example is the ESAPI Session Management control, which includes a component for CSRF. [REF-45]

Phase: Implementation

Ensure that the application is free of cross-site scripting issues (CWE-79), because most CSRF defenses can be bypassed using attacker-controlled script.

Phase: Architecture and Design

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). [REF-332]

Note: Note that this can be bypassed using XSS (CWE-79).

Phase: Architecture and Design

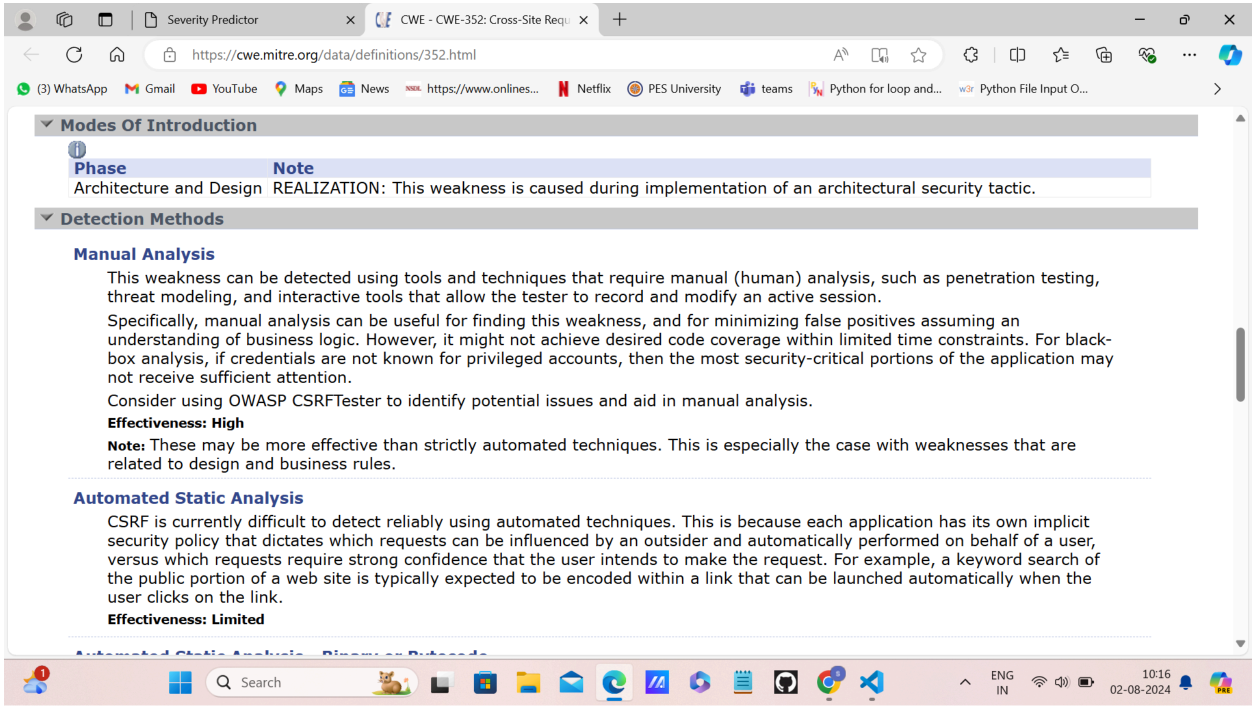
Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note: Note that this can be bypassed using XSS (CWE-79).

Phase: Architecture and Design

Use the "double-submitted cookie" method as described by Felten and Zeller:

When a user visits a site, the site should generate a pseudorandom value and set it as a cookie on the user's machine. The site



Modes Of Introduction

Phase	Note
Architecture and Design	REALIZATION: This weakness is caused during implementation of an architectural security tactic.

Detection Methods

Manual Analysis

This weakness can be detected using tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session.

Specifically, manual analysis can be useful for finding this weakness, and for minimizing false positives assuming an understanding of business logic. However, it might not achieve desired code coverage within limited time constraints. For black-box analysis, if credentials are not known for privileged accounts, then the most security-critical portions of the application may not receive sufficient attention.

Consider using OWASP CSRFTester to identify potential issues and aid in manual analysis.

Effectiveness: High

Note: These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

Automated Static Analysis

CSRF is currently difficult to detect reliably using automated techniques. This is because each application has its own implicit security policy that dictates which requests can be influenced by an outsider and automatically performed on behalf of a user, versus which requests require strong confidence that the user intends to make the request. For example, a keyword search of the public portion of a web site is typically expected to be encoded within a link that can be launched automatically when the user clicks on the link.

Effectiveness: Limited