Design and apply context-aware access controls based on user identity, device posture, location, and application risk.

Team details

S.NO	NAME	COLLEGE	CONTACT
1	**KISHORE KUMAR M** (Team leader)	UNITED COLLEGE OF ARTS AND SCIENCE	kishoremskj04@gmail.com
2	**KARTHIKEYAN B** (Member)	UNITED COLLEGE OF ARTS AND SCIENCE	karthikeyan25034@gmail.com
3	**KARTHIKEYAN K** (Member)	UNITED COLLEGE OF ARTS AND SCIENCE	karthik2822c@gmail.com
4	**KOWSALYA A** (Member)	UNITED COLLEGE OF ARTS AND SCIENCE	kowsalyaravi482@gmail.com

INTRODUCTION:

In an era where the digital landscape is constantly evolving, organizations face an ever-increasing threat from cyberattacks. The need for swift and effective incident response has become paramount to safeguard sensitive data, protect digital assets, and maintain the trust of stakeholders. In response to this growing challenge, we present a visionary project: the development of an AI-driven incident response platform that empowers cybersecurity teams with the tools to automate and enhance incident triage and response tasks.

The traditional network perimeter has all but dissolved in the modern digital landscape. With the mass adoption of cloud services, remote work, and personal mobile devices (BYOD), corporate resources are no longer confined to a single office building. This shift has rendered the conventional "castle-and-moat" security model—where anyone inside the network is trusted by default—obsolete and dangerously insufficient. Modern organizations now face the complex challenge of securing access to sensitive data from a diverse and ever-changing array of locations, devices, and user environments. To address this, security strategies must evolve from static, location-centric models to dynamic, identity-centric frameworks that assess the risk of every access request in real-time.

This project focuses on the design and implementation of a context-aware access control system to solve this exact challenge. By moving beyond simple username and password verification, this system will evaluate a rich set of contextual signals before granting access to applications and data. These signals include the user's identity and role, the security posture of their device (e.g., is it patched, encrypted, and managed?), their geographic location, and the inherent risk level of the application they are attempting to use. By analyzing these factors in unison, the system can make a granular, real-time decision: grant full access, restrict access to certain features, require step-up authentication (like multi-factor authentication), or block the request entirely.

Ultimately, this project aims to build a security architecture that is both robust and frictionless. For example, a trusted employee accessing a low-risk application from a secured corporate laptop on the office network might have a seamless login experience. Conversely, that same employee attempting to access a critical financial system from an unpatched personal device in a foreign country would be met with additional security challenges or be completely blocked. This intelligent, adaptive approach not only significantly reduces the attack surface and protects against compromised credentials but also enables a secure and productive work environment for a modern, mobile workforce.

ABSTRACT:

The increasing adoption of cloud services, mobile computing, and hybrid work models has fundamentally broken the traditional network perimeter, rendering legacy access control models ineffective. Organizations now struggle to protect sensitive data from a growing attack surface, where threats can originate from compromised endpoints, unsafe locations, or malicious insiders. Static, binary decisions (allow or deny) are no longer sufficient to mitigate the sophisticated risks present in modern distributed environments. There is a critical need for a dynamic security framework that can evaluate the full context of every access request in real-time.

This project presents the design and application of a context-aware access control system based on user identity, device posture, geolocation, and application risk. The objective is to develop a policy engine that ingests multiple contextual signals to calculate a real-time risk score for each access attempt. The methodology involves defining granular policies that map specific context combinations (e.g., "managed device" + "corporate network" + "low-risk app") to corresponding access decisions, such as granting full access, enforcing step-up authentication (MFA), limiting session functionality, or denying access entirely.
The expected outcome is an adaptive security model that significantly reduces the organizational attack surface and mitigates the risk of credential compromise.

By ensuring that access decisions are dynamically aligned with the perceived threat level of each request, the system enhances security without introducing unnecessary friction for low-risk, trusted users. This approach provides a robust framework for enabling secure and productive access to resources in an increasingly perimeter-less and dynamic digital world.

Use Case Scenarios

Scenario 1: Finance Department Access to ERP System
A senior accountant in the finance department needs to access the corporate Enterprise Resource Planning (ERP) system to process monthly payroll. The user connects from a remote location using their company-managed laptop. Upon launching the ZPA Client Connector, they authenticate via Azure Active Directory with multi-factor authentication (MFA). ZPA evaluates the user's identity, group membership (Finance_ERP_Users), device posture (antivirus active, disk encryption enabled), and connection context. The policy engine determines that the user is authorized and brokers a secure, direct micro-tunnel to the ERP application only. The user can interact with the ERP interface, but has no visibility or network-level access to any other internal systems, such as development servers or the HR database.
Output:

- User authenticated and authorized based on identity and group membership
- Direct micro-tunnel established specifically to the ERP application
- All other internal applications and network segments remain invisible and inaccessible

Scenario 2: Contractor Access to Specific Project Repository
An external software consultant is hired for a three-month project and requires access only to a specific Git repository hosted on an internal server. The contractor is onboarded as a guest user in the identity provider and assigned to the "Project_X_Contractors" group. When the contractor connects using the ZPA client, the system verifies their identity and checks the device for basic security compliance (OS patch level, firewall enabled). ZPA enforces a policy that grants access exclusively to the Git repository application (port 22/SSH or port 443/HTTPS) during business hours (9 AM to 5 PM). The contractor can clone, push, and pull code, but cannot discover or connect to any other internal resources, including the corporate intranet or file servers.
Output:

- Time-bound and role-specific access enforced
- Contractor restricted to a single application (Git repository)
- Lateral movement to adjacent systems completely blocked

Scenario 3: Executive Access to Sensitive Financial Documents
A company VP needs to access a folder containing confidential M&A documents stored in a corporate SharePoint site. The user connects from a hotel business center while traveling internationally, using their company-managed tablet. Upon launching the ZPA Client Connector, they authenticate via Azure Active Directory with MFA. ZPA evaluates the user's identity, executive group membership, and device posture (which is compliant). However, the geolocation (unusual foreign country) and the application risk (highly sensitive document repository) trigger an elevated risk score. The policy engine grants access to the SharePoint site, but enforces additional restrictions: download and print capabilities are disabled, and the session is monitored for unusual behavior (e.g., mass download attempts). An alert is also sent to the security team for high-risk executive login. Output:

- User authenticated with MFA and device compliant, but flagged for unusual location
- Read-only access granted with download/print restrictions enforced
- Security team alerted to potential credential misuse or espionage attempt

Scenario 4: Remote Support Technician Access to Internal Tools
An IT support technician needs to access the corporate ServiceNow instance and a remote desktop tool to assist an employee. The technician connects from their home office using a company-managed desktop. Upon launching the ZPA Client Connector, they authenticate via Azure Active Directory with MFA. ZPA evaluates the user's identity, group membership (IT_Support), device posture (fully compliant), and connection type (home office, which is whitelisted for IT staff). Because the user, device, and location all meet the信任baseline, the policy engine grants direct micro-tunnel access to both ServiceNow and the specific remote desktop jump host used for employee support. The technician can perform their duties efficiently, but still has no access to finance systems, HR databases, or source code repositories.

Output:

- User authenticated and fully authorized based on trusted context
- Direct micro-tunnels established to ServiceNow and the remote desktop jump host
- Least-privilege access maintained; unrelated internal systems remain invisible

Scenario 5: High-Risk Application Access from a Low-Trust Location
A marketing coordinator needs to access the corporate Salesforce instance to update a campaign. The user connects from a coffee shop using their company-managed smartphone. Upon launching the ZPA Client Connector, they authenticate via Azure Active Directory with MFA. ZPA evaluates the user's identity, group membership (Marketing_Salesforce_Users), device posture (managed and compliant), but flags the public Wi-Fi network as a high-risk connection. The policy engine determines that while the user and device are trusted, the network environment introduces unacceptable risk for direct access to a customer-facing CRM containing sensitive lead data. Instead of blocking access entirely, the user is routed through a secure ZPA browser isolation service, rendering the Salesforce session in a disposable container that prevents any data from persisting on the device or being exposed to the public network.

Output:

- User authenticated and device compliant, but network location deemed high-risk
- Access granted via browser isolation to prevent data leakage on public Wi-Fi
- Session is rendered in a secure container with copy/paste and download controls enforced

Stage – 1

Title:

Design and apply context-aware access controls based on user identity, device posture, location, and application risk.

Overview:

In response to the evolving threat landscape and the dissolution of the traditional network perimeter, modern enterprises are rapidly shifting away from legacy VPN-centric remote access models. These conventional approaches often grant excessive network-level access, creating a significant attack surface and increasing the risk of lateral threat movement. To address these challenges, this project centers on the implementation of a Zero Trust security architecture, fundamentally operating on the principle of "never trust, always verify." By moving beyond location-based access, we aim to establish a security posture where trust is no longer implicit, and access is determined by granular context rather than network connectivity.

The core of this initiative involves the deployment of Zscaler Private Access (ZPA), a cloud-delivered Zero Trust Access solution designed to create identity-aware, application-specific connectivity. Unlike traditional VPNs that place users on the internal network, ZPA leverages a "service-initiated" model to establish direct, one-to-one connections between authorized users and specific applications, completely hiding the corporate network and IP address space from the internet. This ensures that users connect exclusively to the applications they are permitted to access, based on continuous identity verification and context, rendering the applications invisible to unauthorized users and preventing common attack vectors such as port scanning and distributed denial-of-service (DDoS) attacks.

Ultimately, this project will enforce a least-privilege access model, ensuring users and devices are granted the minimum required access necessary to perform their functions. By decoupling application access from network access, we will eliminate the risk of lateral movement across the network, as users have no visibility or connectivity to other hosts. This strategic move not only significantly enhances security by reducing the attack surface but also simplifies the user experience and reduces operational overhead associated with legacy remote access infrastructure, positioning the organization at the forefront of secure and efficient access management.

List of Vulnerability:

S. No.	Vulnerability Name	CWE No.
1.	Cross-site scripting (stored)	79
2.	Broken Access Control	285
3.	SQL Injection	89
4.	Broken Authentication	285
5.	Insecure Direct Object Reference	639
6.	Security Misconfiguration	732
7.	Cross-Site Request Forgery	352
8.	Cleartext Transmission of Sensitive Information	319
9.	ClickJacking	1021
10.	webserver allows password auto-completion	310
11.	Cross-site scripting (DOM-based XSS)	79
12.	JQuery 1.2 < 3.5.0 Multiple XSS	79
13.	Information Exposure (Web Server HTTP Header Internal IP Disclosure)	200
14.	Information Exposure (SSL (Multiple Issues))	200
15.	Information Exposure (HTTP (Multiple Issues))	200
16.	OS Identification Failed	200

Report:

1. Vulnerability name: Cross site scripting

   (stored) CWE: 79

   OWASP category: A03:2021 -Injections

   Description: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser.

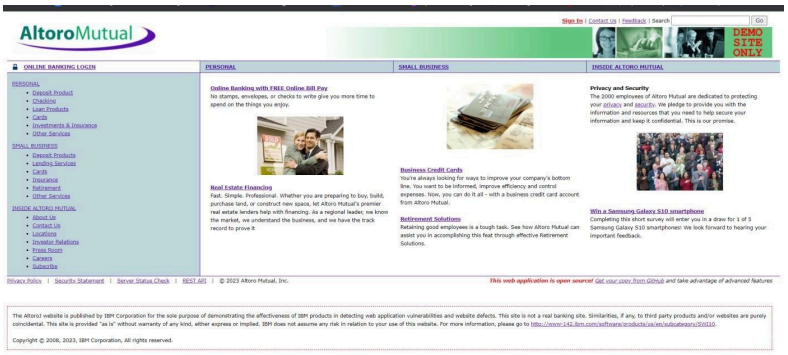
   Business Impact: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

   Vulnerability path: http://testfire.net/

   Vulnerability parameter:
   http://testfire.net/search.jsp?query=%27%3E%3Cscript%3Ealert%28%27hacked%27%29%3C%2Fscrip t%3E

   Steps to Reproduce:

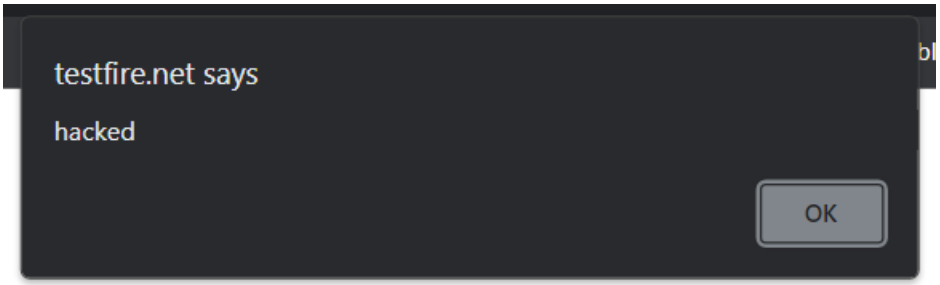
   - Access the URL



- In the search box we will input some code to perform the vulnerability.

- The Script we will be inputting is '><script>alert('hacked')</script>

- This displays a harmless pop up alert box with the text saying 'hacked'



Recommendation:

- When interacting with databases, use parameterized statements or prepared statements to avoid SQL injection, which can be a vector for XSS.
- Sanitize and validate all user inputs and ensure that any data displayed on the web page is properly encoded. Use output encoding libraries or functions to prevent script injection.

2. Vulnerability name: Broken access

control CWE: 285

Description: When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information exposures, denial of service, and arbitrary code execution.

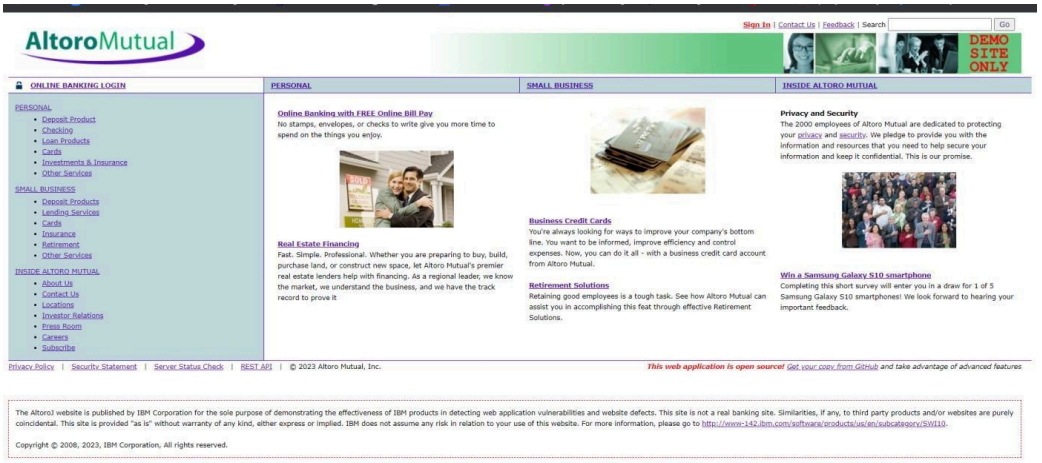
Business Impact: wide-ranging business impact, including financial losses, reputation damage, legal consequences, operational disruption, and loss of customer trust. Addressing this vulnerability is crucial to protect the organization and its stakeholders from these potential negative effects.

Vulnerability path: http://testfire.net/

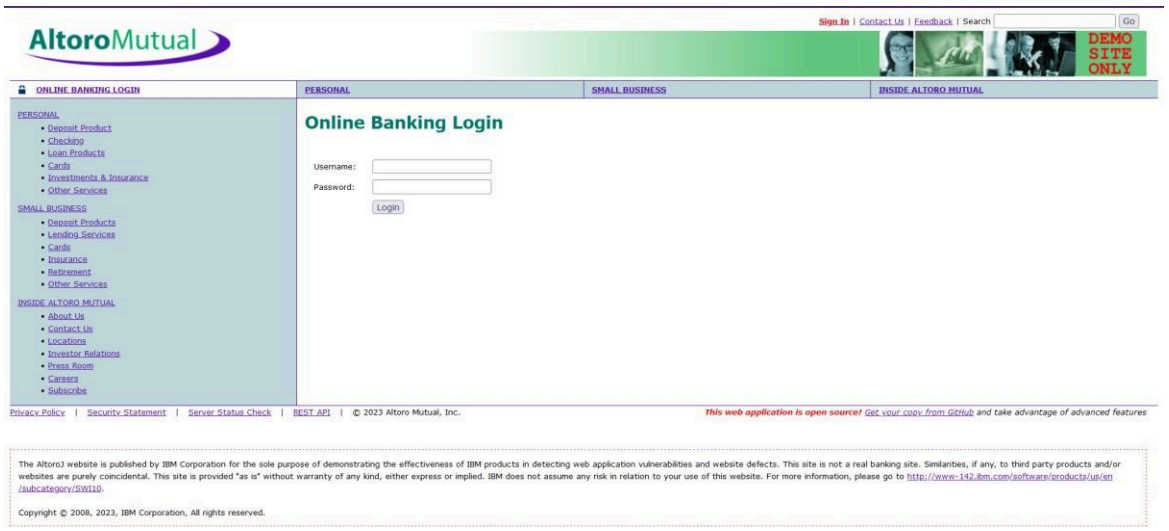
Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

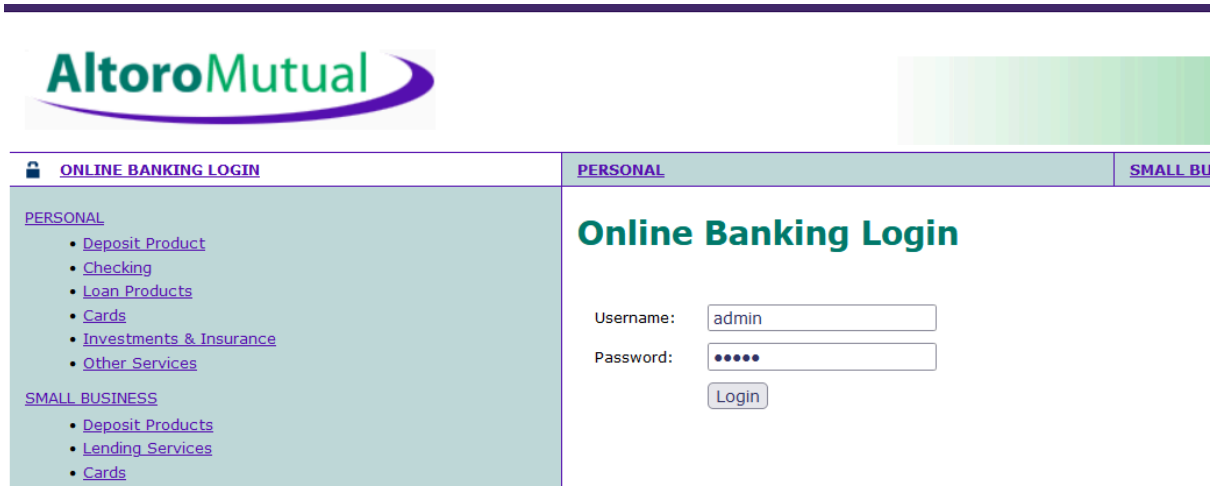
- Access the URL



- Now we will try to sign in to this website with admin privileges



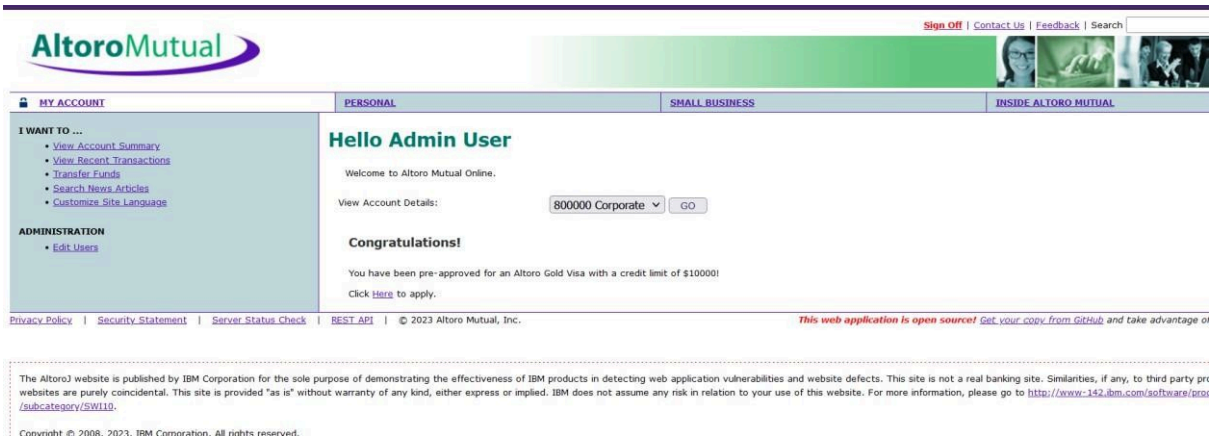
- Also, we will be using burp suite to get requests from the website and know additional information. We use 'admin' for the username and password.



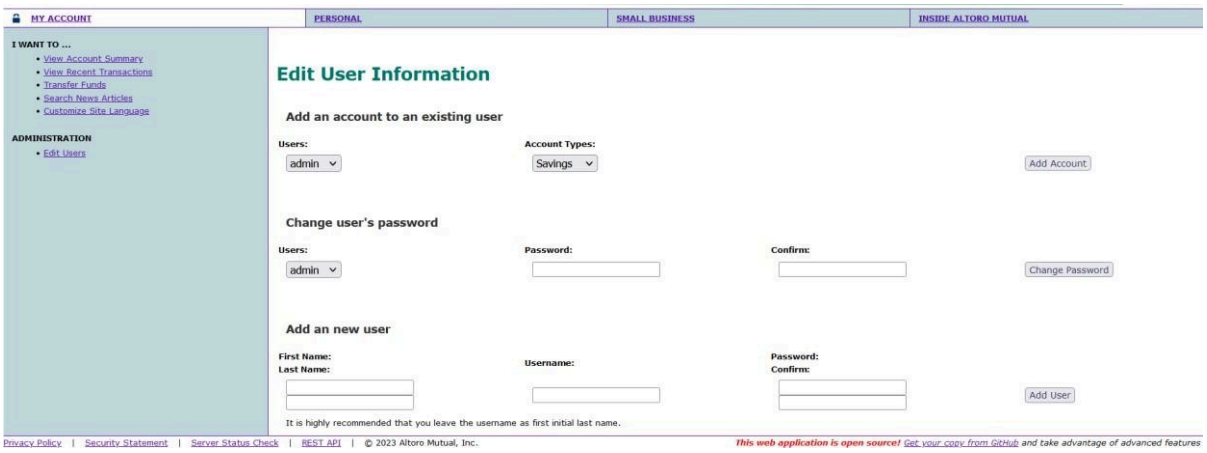
- This request has been received in the burp suite with the username and password as well in clear text.

```
Pretty   Raw   Hex
1  POST /doLogin HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 37
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/login.jsp
12 Cookie: JSESSIONID=B6ADB36ACD5C83083787343A1F97F853; AltoroAccounts="0DAwMDAwfkNvcnBvcmF0ZX45LjQ30TA1MTE2MUU3fDgwMDAwMX5DaGVja2luZ34tNC4yMjc0NjkONTZFN3w="
13 Upgrade-Insecure-Requests: 1
14
15 uid=admin&passw=admin&btnSubmit=Login
```

- now we click on forward request in the burp suite and then we will be redirected to the admin user details.



- We can even edit the other users data and delete their login details as well.



Recommendations:

- Ensure that the application has a robust authentication and authorization mechanism in place. Users should be authenticated before any sensitive actions are performed, and authorization checks should be conducted to verify that users have the necessary permissions for the requested operation.
- Ensure that session management is secure. Use strong session IDs, enforce session timeouts, and regenerate session tokens after login. Implement session fixation protection to prevent session hijacking.

3. Vulnerability name: SQL

   injection CWE: 89

   Description: The product constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

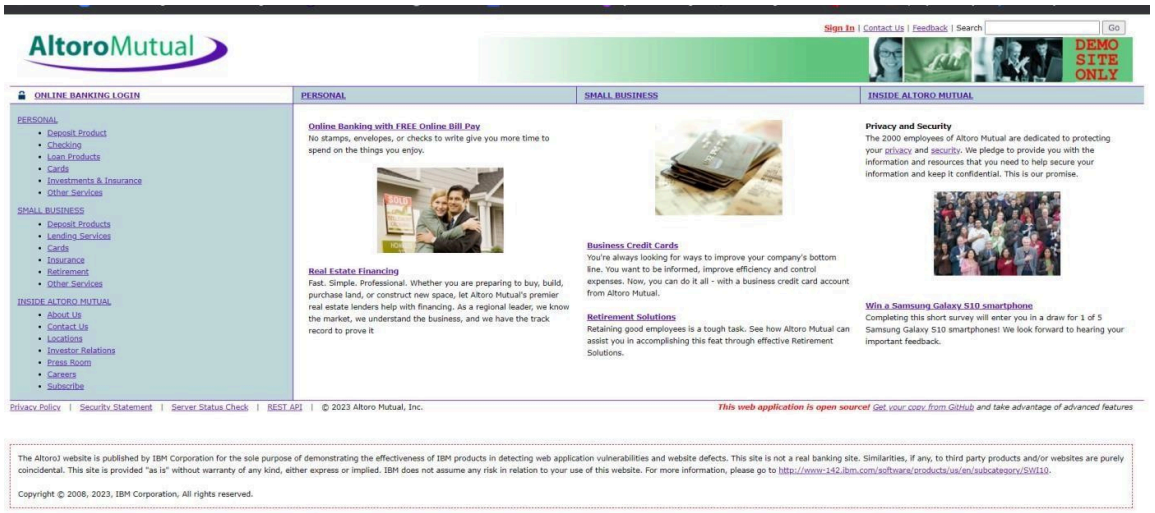
Business Impact: In summary, it is crucial to underscore that CWE-89, known as SQL Injection, can exert a profound and diverse business impact. This encompasses critical facets such as data breaches, financial setbacks, harm to reputation, legal ramifications, and operational turmoil. Hence, the imperative of preventing and remedying SQL injection vulnerabilities cannot be overstated, as it is indispensable for fortifying the security and continuity of an organization's applications and data.

Vulnerability path: http://testfire.net/

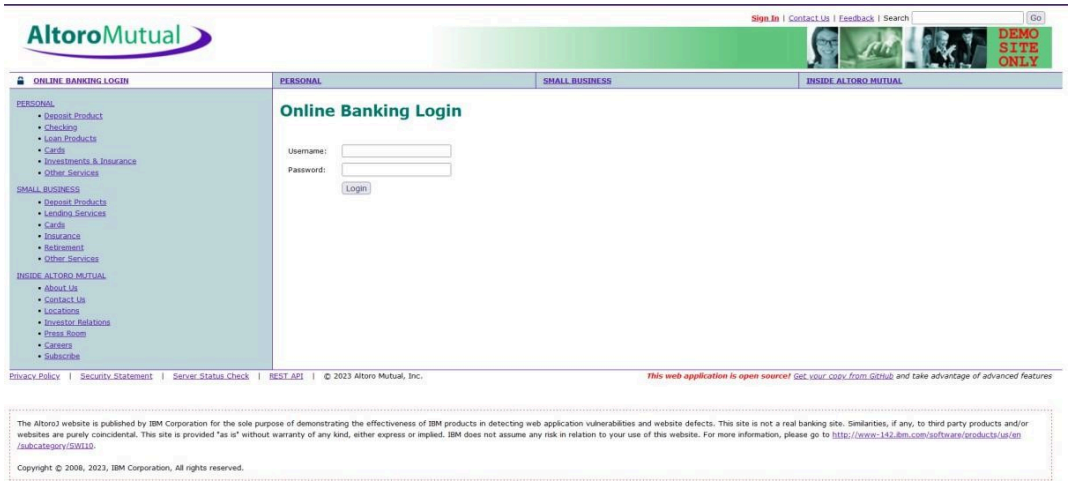
Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

- Access the URL



- Now we will try to sign in to this website with admin privileges but using SQL injection



- Also, we will be using burp suite to get requests from the website and know additional information.

Intercept | HTTP history | WebSockets history | Proxy settings

Request to http://testfire.net:80 [65.61.137.117]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
1  GET / HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Connection: close
8  Cookie: JSESSIONID=AFA9012F50640AF57ED2D63F8959DEE0
9  Upgrade-Insecure-Requests: 1
10
11
```

Dashboard | Target | Proxy | Intruder | Repeater | Collaborator | Sequencer | Decoder | Comparer | Logger | Organizer | Extensions | Learn

Intercept | HTTP history | WebSockets history | Proxy settings

Request to http://testfire.net:80 [65.61.137.117]

Forward | Drop | Intercept is on | Action | Open browser

Pretty | Raw | Hex

```
1  POST /doLogin HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/118.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 45
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/login.jsp
12 Cookie: JSESSIONID=AFA9012F50640AF57ED2D63F8959DEE0
13 Upgrade-Insecure-Requests: 1
14
15 uid=%27+or+1%3D1--&passw=1234&btnSubmit=Login
```

- Click forward request multiple times to proceed to the login page.

- With this we can know that sql injection worked and we got the admin privileges.

Recommendations:

- If parameterized statements are not feasible, use proper input validation and escaping mechanisms to sanitize user inputs before they are used in SQL queries. This helps prevent malicious code injection.
- Utilize stored procedures for database operations. This can help separate SQL code from application code and reduce the risk of SQL injection.
- Implement strict input validation for all user inputs, ensuring that data adheres to the expected format and structure.

4. Vulnerability name: Broken

authentication CWE: 285

Description: The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.

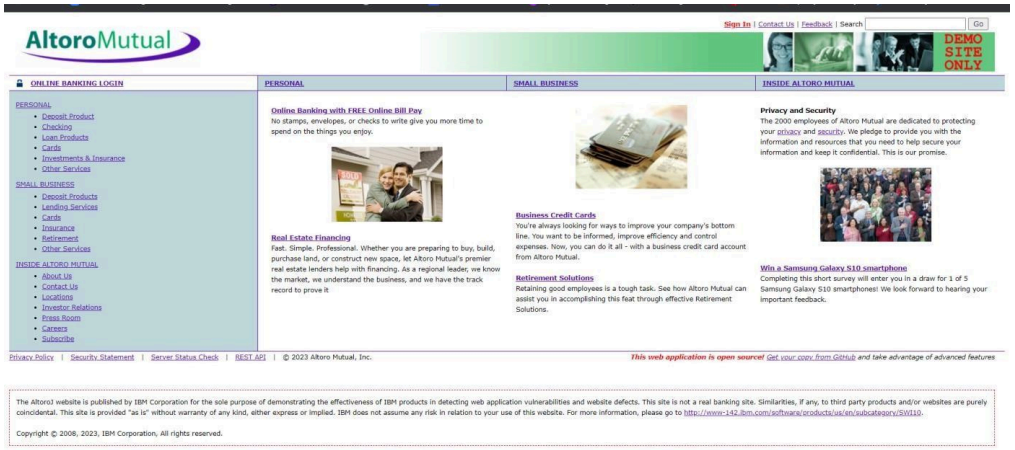
Business Impact: To effectively mitigate the business impact of CWE-285, it is imperative that organizations place a strong emphasis on bolstering their authentication and session management practices. This should encompass the adoption of multi-factor authentication, the secure storage of credentials, meticulous session handling, and a commitment to conducting routine security assessments and testing. The rectification of these vulnerabilities stands as a critical imperative, safeguarding sensitive data, user identities, and the organization's overarching security stature and reputation.

Vulnerability path: http://testfire.net/

Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

- Access the URL



- Now we will try to login using some different approach.

- As we know which users are present in the database of this website by using the admin privileges. We can directly access a particular user by simply knowing their username; we will add some characters after his user name as a sql injection to simply bypass the password.

- This leads us to the details of this person's account.

Recommendations:

- Implement strong authentication mechanisms, such as multi-factor authentication (MFA) or two-factor authentication (2FA), to ensure that only authorized users can access the system.
- Implement proper authorization controls to ensure that users have the necessary permissions to access specific functions or data. Use role-based access control (RBAC) to manage user privileges effectively.
- Deploy intrusion detection and prevention systems (IDS/IPS) to monitor and block suspicious activities related to authentication and authorization.

5. Vulnerability Name: Insecure Direct object Reference

CWE: 639

OWASP Category: A01: Broken Access Control

Description:
Insecure Direct Object Reference (IDOR) is a vulnerability that arises when attackers can access or modify objects by manipulating identifiers used in a web application's URLs or parameters. It occurs due to missing access control checks, which fail to verify whether a user should be allowed to access specific data.

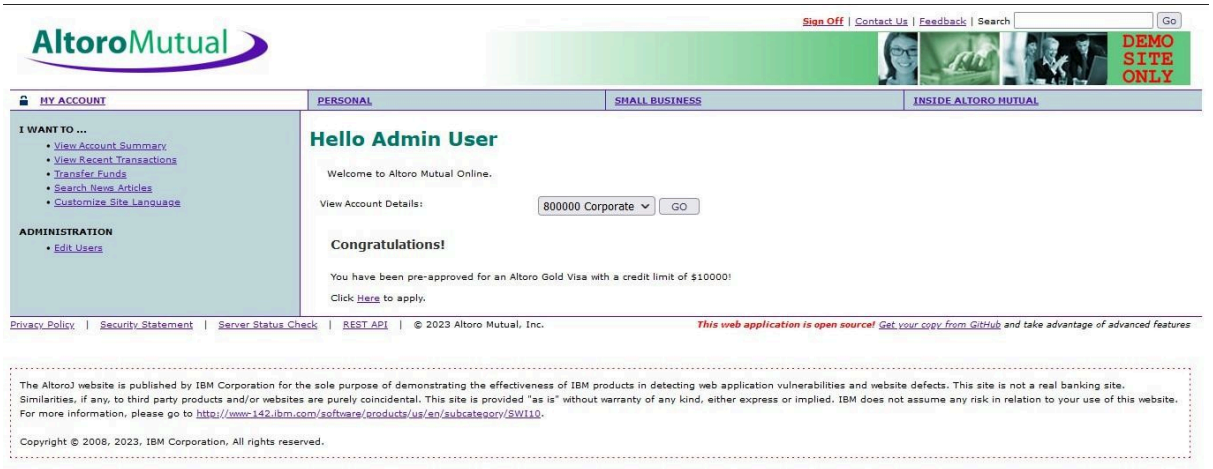
Business Impact:
To effectively mitigate the business impact of CWE-639, organizations must make it a top priority to fortify their access control and authorization mechanisms. This entails the implementation of robust security measures, the regular conduct of comprehensive security assessments, and the deployment of intrusion detection systems to promptly identify and counter unauthorized access attempts. These measures stand as absolutely critical in the defense of sensitive data, the overall security of systems, and the preservation of the organization's esteemed reputation.

Vulnerability Path: http://testfire.net

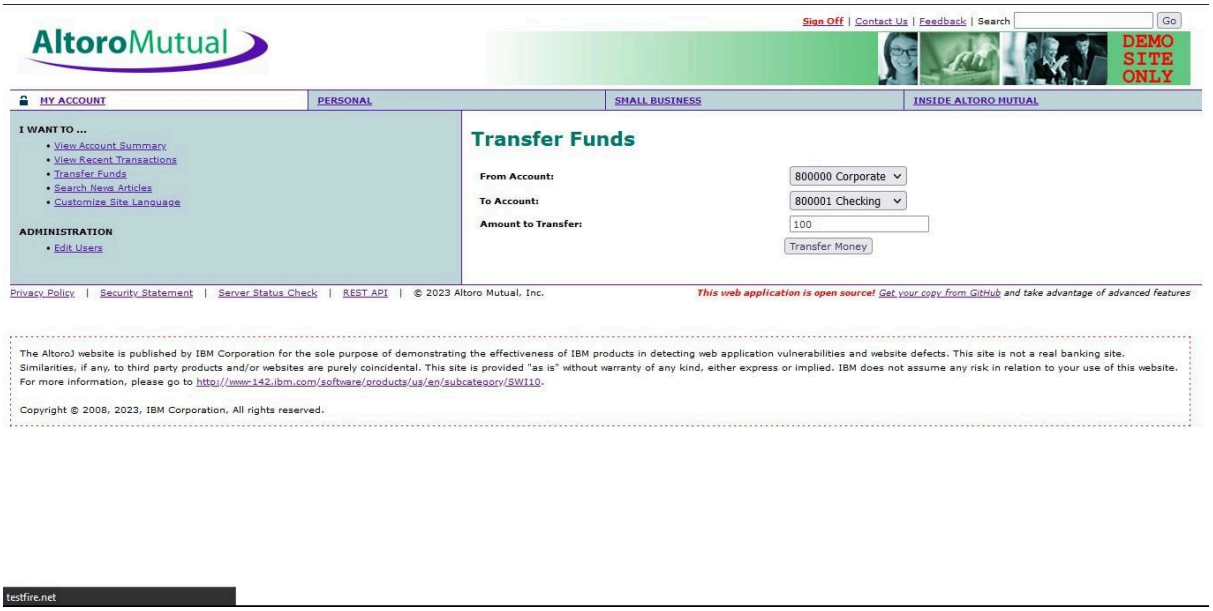
Vulnerability Parameter: http://testfire.net/bank/transfer.jsp

Steps to Reproduce:

- Go to the domain.
- Sign into the site.



- Open “Transfer Money” on the left side and fill in the details. On the intercept and click transfer



- Open burp and notice the change.

```
Pretty   Raw   Hex
1  POST /bank/doTransfer HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 78
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/bank/transfer.jsp
12 Cookie: JSESSIONID=17BCE438BBB750A14D269616C7623BB0; AltoroAccounts=
   "ODAwMDAwfkNvcnBvcmFOZX4xLjYzNTczNTM1NjM10DcONEUy0Xw4MDAwMDF+Q2hlY2tpbmd+LTEuNjM1NzM1MzU2MzU4NzQORTI5fA=="
13 Upgrade-Insecure-Requests: 1
14
15 fromAccount=800000&toAccount=800001&transferAmount=100&transfer=Transfer+Money
```

- In the 15th line change the amount from 100 to 1000 and click forward

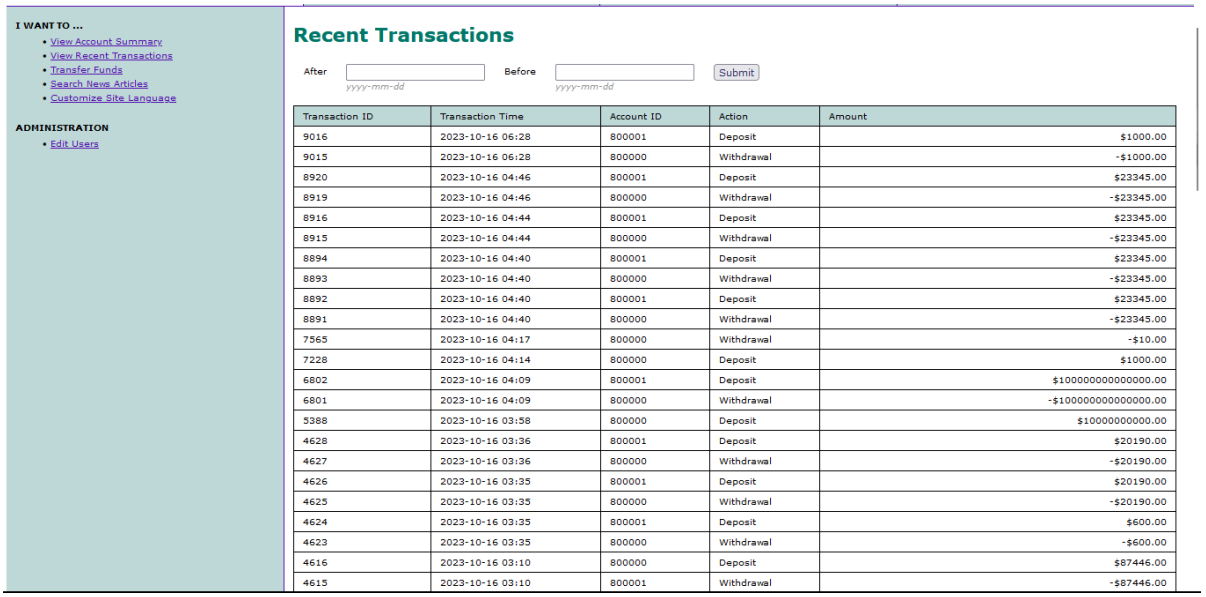
```
fromAccount=800000&toAccount=800001&transferAmount=100&transfer=Transfer+Money

fromAccount=800000&toAccount=800001&transferAmount=1000&transfer=Transfer+Money
```

Pretty Raw Hex

```
1 GET /v1/tiles HTTP/1.1
2 Host: contile.services.mozilla.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Te: trailers
8 Connection: close
9
10
```

- Look at the site, we can notice the msg that shows the transfer of 1000.

- Off the Intercept and open "View Recent Transactions".



Transaction ID	Transaction Time	Account ID	Action	Amount
9016	2023-10-16 06:28	800001	Deposit	$1000.00
9015	2023-10-16 06:28	800000	Withdrawal	-$1000.00
8920	2023-10-16 04:46	800001	Deposit	$23345.00
8919	2023-10-16 04:46	800000	Withdrawal	-$23345.00
8916	2023-10-16 04:44	800001	Deposit	$23345.00
8915	2023-10-16 04:44	800000	Withdrawal	-$23345.00
8894	2023-10-16 04:40	800001	Deposit	$23345.00
8893	2023-10-16 04:40	800000	Withdrawal	-$23345.00
8892	2023-10-16 04:40	800001	Deposit	$23345.00
8891	2023-10-16 04:40	800000	Withdrawal	-$23345.00
7565	2023-10-16 04:17	800000	Withdrawal	-$10.00
7228	2023-10-16 04:14	800000	Deposit	$1000.00
6802	2023-10-16 04:09	800001	Deposit	$100000000000000.00
6801	2023-10-16 04:09	800000	Withdrawal	-$100000000000000.00
5388	2023-10-16 03:58	800000	Deposit	$10000000000.00
4628	2023-10-16 03:36	800001	Deposit	$20190.00
4627	2023-10-16 03:36	800000	Withdrawal	-$20190.00
4626	2023-10-16 03:35	800001	Deposit	$20190.00
4625	2023-10-16 03:35	800000	Withdrawal	-$20190.00
4624	2023-10-16 03:35	800001	Deposit	$600.00
4623	2023-10-16 03:35	800000	Withdrawal	-$600.00
4616	2023-10-16 03:10	800000	Deposit	$87446.00
4615	2023-10-16 03:10	800001	Withdrawal	-$87446.00

Recommendation:

- Utilize parameterized queries or prepared statements when interacting with databases. This prevents user-controlled input from being executed as SQL code.
- Implement a Content Security Policy to mitigate the risk of cross-site scripting (XSS) attacks, which can be used to execute unauthorized queries.
- Implement account lockout mechanisms that temporarily suspend accounts after a specified number of unauthorized access attempts to prevent brute force attacks.

6. Vulnerability name: Security

Misconfiguration CWE:732

Description: The product specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

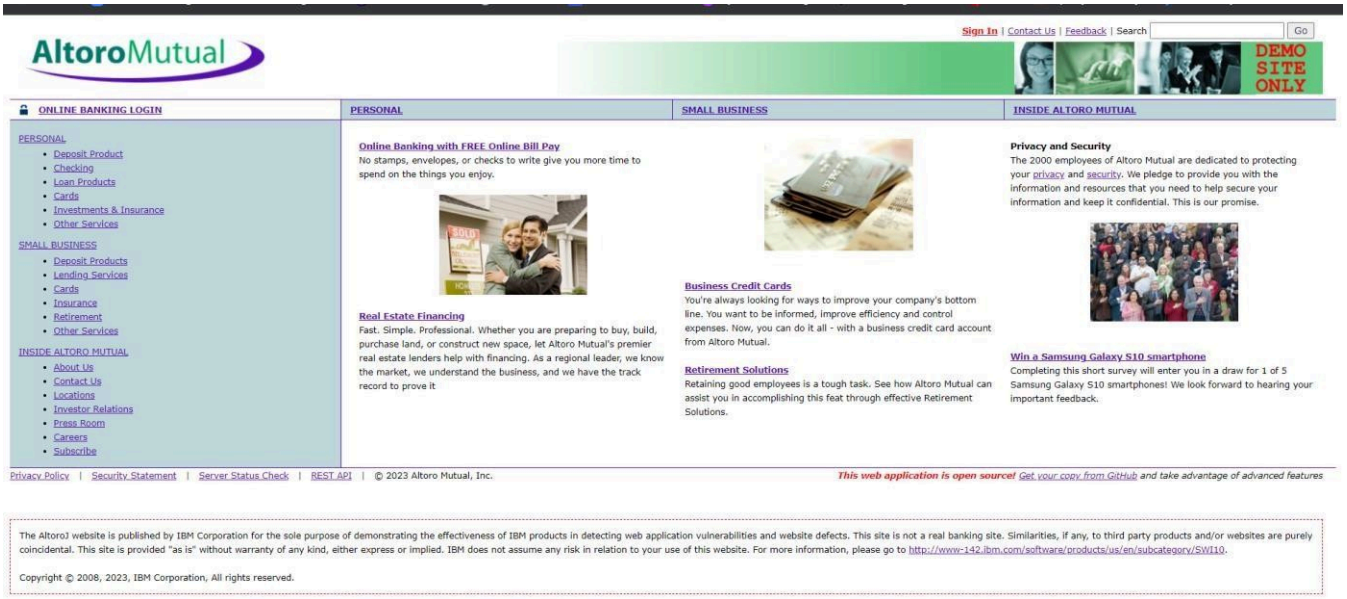
Business Impact: To effectively lessen the business consequences stemming from CWE-732, organizations must place a strong emphasis on bolstering their configuration management practices. This involves the establishment of secure default settings, the routine conduct of comprehensive security evaluations, and the deployment of intrusion detection systems to promptly detect and address misconfigurations and vulnerabilities. These actions are of paramount importance in safeguarding sensitive data, fortifying system security, and preserving the organization's esteemed reputation.

Vulnerability path: http://testfire.net/

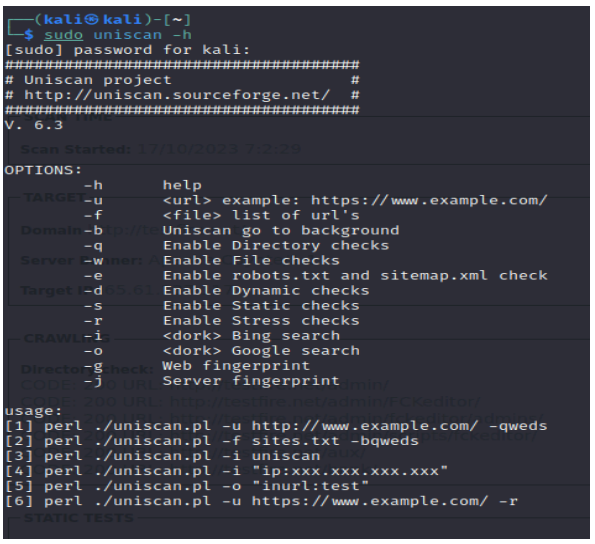
Vulnerability parameter: http://testfire.net/login.jsp

Steps to Reproduce:

- Access the URL



- We will be using the uniscan tool which is available in the kali.



- We will use this tool to scan for hidden directories that can be access due to poor security measures taken in this website.

```
┌──(kali㉿kali)-[~]
└─$ sudo uniscan -u http://testfire.net/ -qweds
[sudo] password for kali:
######################################
# Uniscan project                    #
# http://uniscan.sourceforge.net/    #
######################################
V. 6.3


Scan date: 17-10-2023 7:18:22
==============================================================================
| Domain: http://testfire.net/
| Server: Apache-Coyote/1.1
| IP: 65.61.137.117
==============================================================================
|
| Directory check:
| [+] CODE: 200 URL: http://testfire.net/admin/fckeditor/admins/
| [+] CODE: 200 URL: http://testfire.net/admin/
| [+] CODE: 200 URL: http://testfire.net/admin/scripts/fckeditor/
| [+] CODE: 200 URL: http://testfire.net/admin/FCKeditor/
| [+] CODE: 200 URL: http://testfire.net/aux/
| [+] CODE: 200 URL: http://testfire.net/bank/
==============================================================================
|
```

- We found directories that can be accessed easily. Also all the files that are present in the website has also been found.

```
| File System
| File check:
| [+] CODE: 200 URL: http://testfire.net/admin/account.asp
| [+] CODE: 200 URL: http://testfire.net/admin/account.html
| [+] CODE: 200 URL: http://testfire.net/admin/admin.php
| [+] CODE: 200 URL: http://testfire.net/admin/admin_phpinfo.php4
| [+] CODE: 200 URL: http://testfire.net/admin/account.php
| [+] CODE: 200 URL: http://testfire.net/admin/admin.shtml
| [+] CODE: 200 URL: http://testfire.net/admin/aindex.htm
| [+] CODE: 200 URL: http://testfire.net/admin/auth.php
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configscreen.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configsite.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configsql.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cfg/configtache.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/cms/htmltags.php
| [+] CODE: 200 URL: http://testfire.net/admin/config.php
| [+] CODE: 200 URL: http://testfire.net/admin/contextAdmin/contextAdmin.html
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.asp
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.php
| [+] CODE: 200 URL: http://testfire.net/admin/controlpanel.html
| [+] CODE: 200 URL: http://testfire.net/admin/cplogfile.log
| [+] CODE: 200 URL: http://testfire.net/admin/cp.php
| [+] CODE: 200 URL: http://testfire.net/admin/cp.html
| [+] CODE: 200 URL: http://testfire.net/admin/database/wwForum.mdb
| [+] CODE: 200 URL: http://testfire.net/admin/credit_card_info.php
| [+] CODE: 200 URL: http://testfire.net/admin/datasource.asp
| [+] CODE: 200 URL: http://testfire.net/admin/db.php
| [+] CODE: 200 URL: http://testfire.net/admin/home.asp
| [+] CODE: 200 URL: http://testfire.net/admin/home.php
| [+] CODE: 200 URL: http://testfire.net/admin/index.asp
| [+] CODE: 200 URL: http://testfire.net/admin/index.html
| [+] CODE: 200 URL: http://testfire.net/admin/index.php
| [+] CODE: 200 URL: http://testfire.net/admin/phpinfo.php
| [+] CODE: 200 URL: http://testfire.net/admin/settings.inc.php+
| [+] CODE: 200 URL: http://testfire.net/admin/system_footer.php
| [+] CODE: 200 URL: http://testfire.net/admin/upload.php
| [+] CODE: 200 URL: http://testfire.net/admin/templates/header.php
| [+] CODE: 200 URL: http://testfire.net/admin/wg_user-info.ml
| [+] CODE: 200 URL: http://testfire.net/admin/script.php
| [+] CODE: 200 URL: http://testfire.net/login.jsp
==============================================================================
|
```

- Let us go through one webpage to see the vulnerability

- Here we can see the sensitive data of job applications which can help an attacker.

Recommendations:

- Implement a change management process to document and review all changes to system configurations. This helps prevent unauthorized or unintended alterations.
- Follow security guidelines provided by software and hardware vendors to ensure secure configurations.
- Regularly perform compliance checks against industry standards and regulations to ensure that configurations remain in compliance with security requirements.

7. Vulnerability Name: Cross-Site Request Forgery

CWS: 352

OWASP Category: A08: Cross-Site Request Forgery

Description:

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Business Impact:

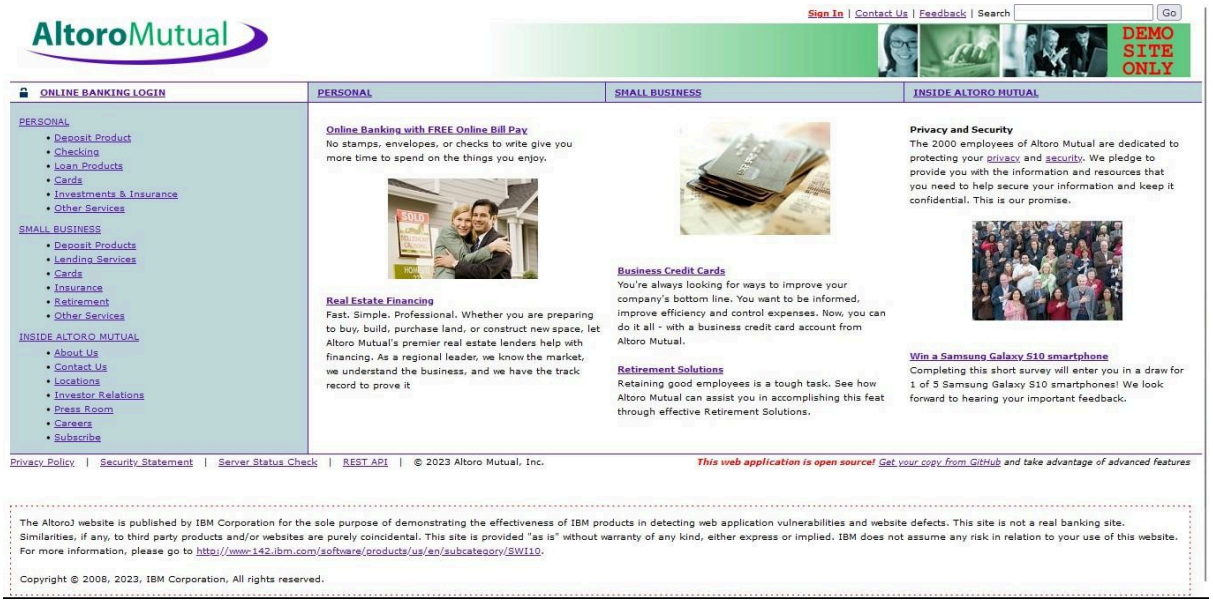
To effectively reduce the business repercussions of CWE-352, organizations must prioritize secure coding practices, including the incorporation of anti-CSRF tokens, while also conducting routine security assessments and testing to detect and rectify CSRF vulnerabilities. Furthermore, educating users on secure browsing practices is pivotal in preventing CSRF attacks. These actions are of paramount importance in the protection of data and in upholding the trust of customers and partners.

Vulnerability Path: http://testfire.net/

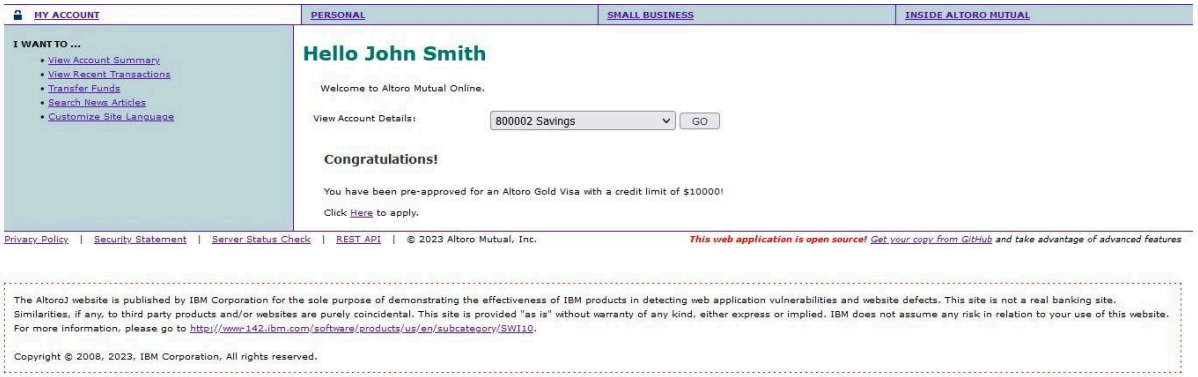
Vulnerability Parameter: http://testfire.net/bank/transfer.jsp

Steps to Reproduce:

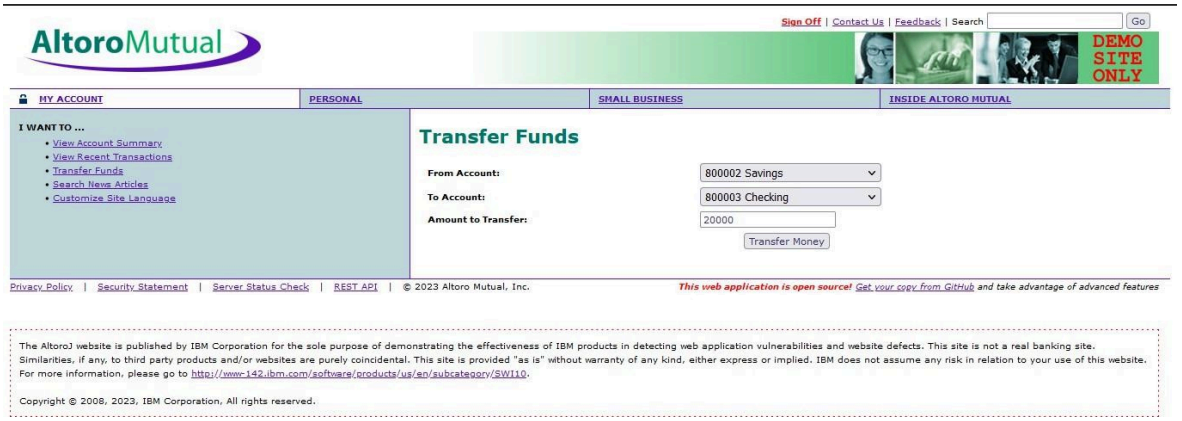
- Open https://testfire.net/.

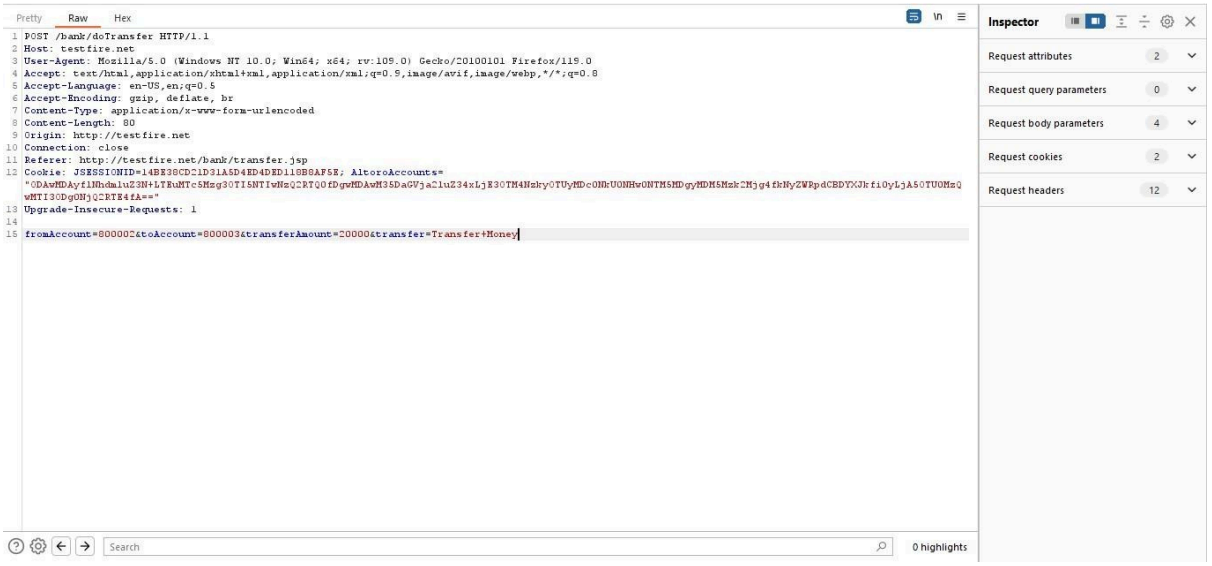


- Login to it using default credentials, and you will find two accounts.

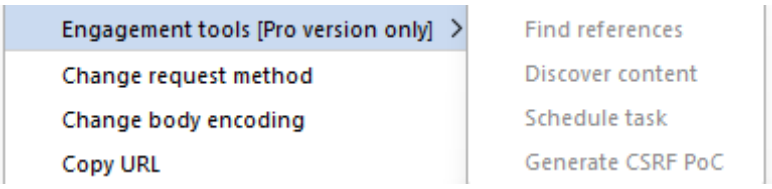


- Capture the request in any Intercepting proxy (Burp Suite, Charles Proxy etc.) while transferring the amount from one account to another as shown in the below picture.





- Generate CSRF POC using Burp Suite Engagement tools (by modifying the account numbers accordingly)

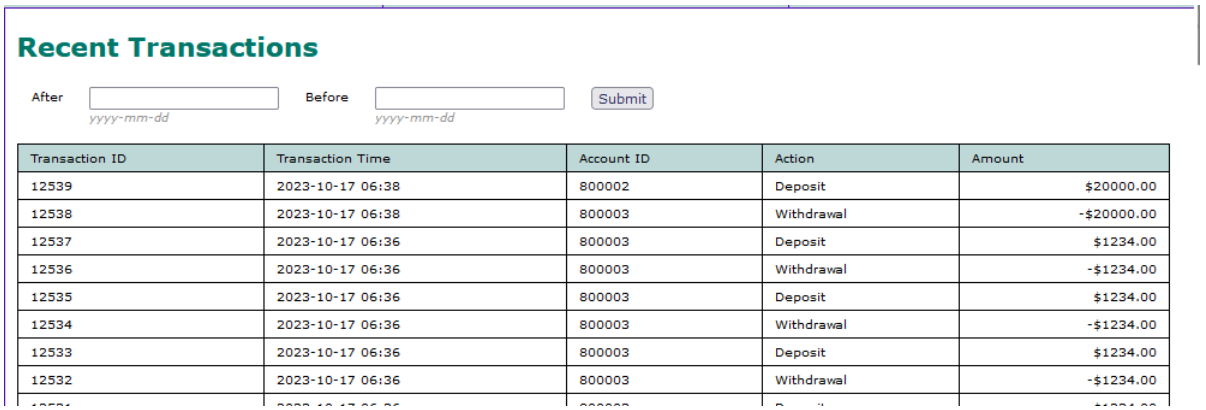


- As the victim is already logged-in to the site an attacker can send the crafted malicious form/URL to the user to his email/text/by any other means and trick the victim using his social engineering skills/techniques like below.

```
Limited Time Offer Get a $100 Gift Card Now!!  (Evil :P)
```

- User thinks, he could use a free $100 gift card! What could go wrong? You open the email and click the link/open the attachment.

- Once the user opens the form a transaction is automatically performed from his account without his knowledge as the user is already logged in.



Recent Transactions

After yyyy-mm-dd Before yyyy-mm-dd Submit

Transaction ID	Transaction Time	Account ID	Action	Amount
12539	2023-10-17 06:38	800002	Deposit	$20000.00
12538	2023-10-17 06:38	800003	Withdrawal	-$20000.00
12537	2023-10-17 06:36	800003	Deposit	$1234.00
12536	2023-10-17 06:36	800003	Withdrawal	-$1234.00
12535	2023-10-17 06:36	800003	Deposit	$1234.00
12534	2023-10-17 06:36	800003	Withdrawal	-$1234.00
12533	2023-10-17 06:36	800003	Deposit	$1234.00
12532	2023-10-17 06:36	800003	Withdrawal	-$1234.00

Recommendations:

- Implement Synchronizer Token Patterns to include unique, randomly generated tokens in each HTTP request and validate these tokens on the server side to verify the authenticity of the request.
- Utilize the SameSite attribute in cookies to restrict their usage to same-site requests only, preventing them from being sent along with cross-site requests and effectively mitigating

CSRF attacks.

- Utilize framework-specific protections and security features to prevent CSRF attacks. Many modern web frameworks have built-in mechanisms and libraries for handling CSRF vulnerabilities effectively.
- Implement a Content Security Policy to restrict the sources from which various types of content can be loaded. By specifying the trusted sources of content, you can minimize the risk of malicious code execution and reduce the likelihood of successful CSRF attacks.

8. Vulnerability name: Cleartext Transmission of Sensitive

   Information CWE: 319

   Description: The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors. Many communication channels can be "sniffed" (monitored) by adversaries during data transmission. For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data. As a result, network traffic could be sniffed by adversaries, spilling security-critical data.

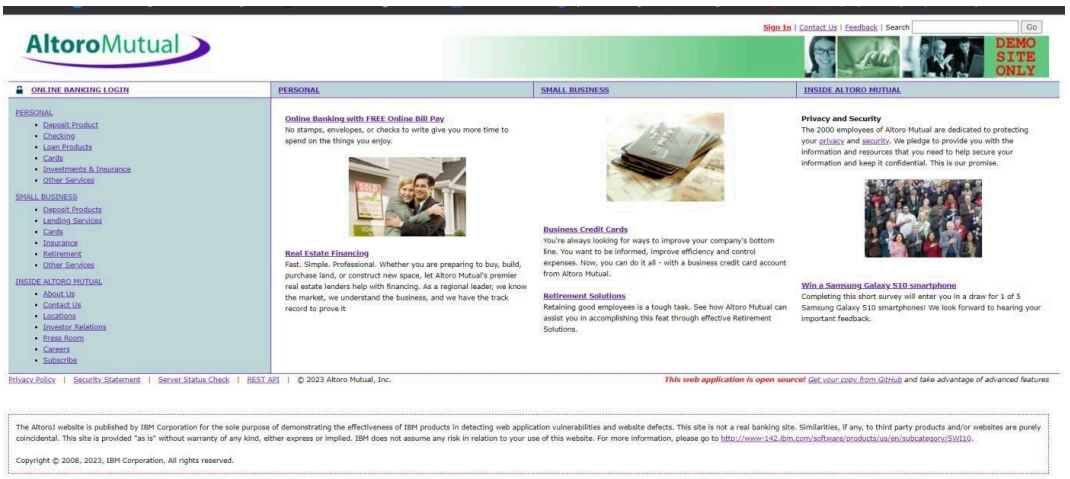
   Business Impact: To effectively reduce the business consequences associated with CWE-319, organizations must prioritize the adoption of secure data transmission practices. This includes the utilization of encryption and robust, secure protocols. The routine conduct of security assessments and testing is pivotal in pinpointing and remedying vulnerabilities linked to data transmission. Furthermore, the education of users on secure data handling practices plays a vital role in proactively preventing data exposure incidents. These actions are of paramount importance in the protection of sensitive data and in preserving the trust of both customers and partners.

   Vulnerability path: http://testfire.net/

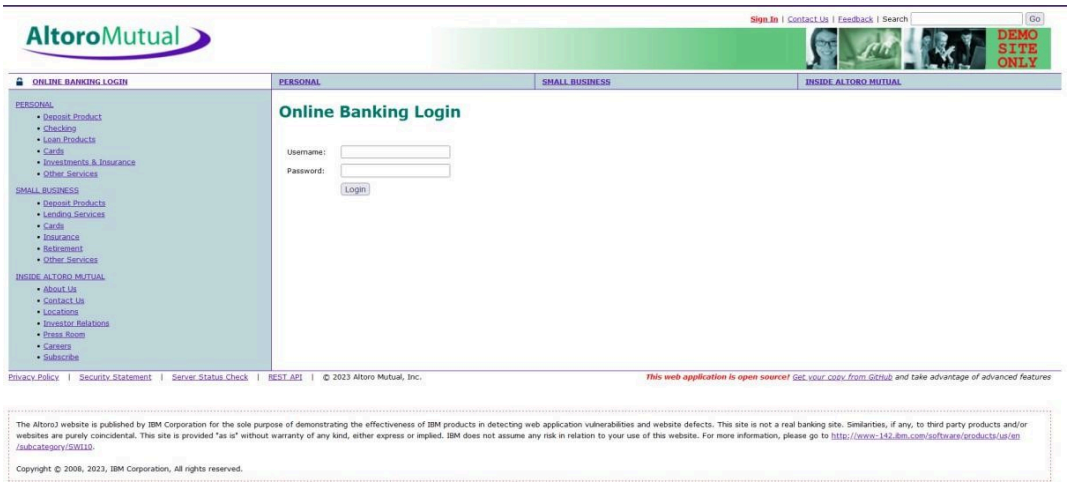
   Vulnerability parameter: http://testfire.net/login.jsp

   Steps to reproduce:

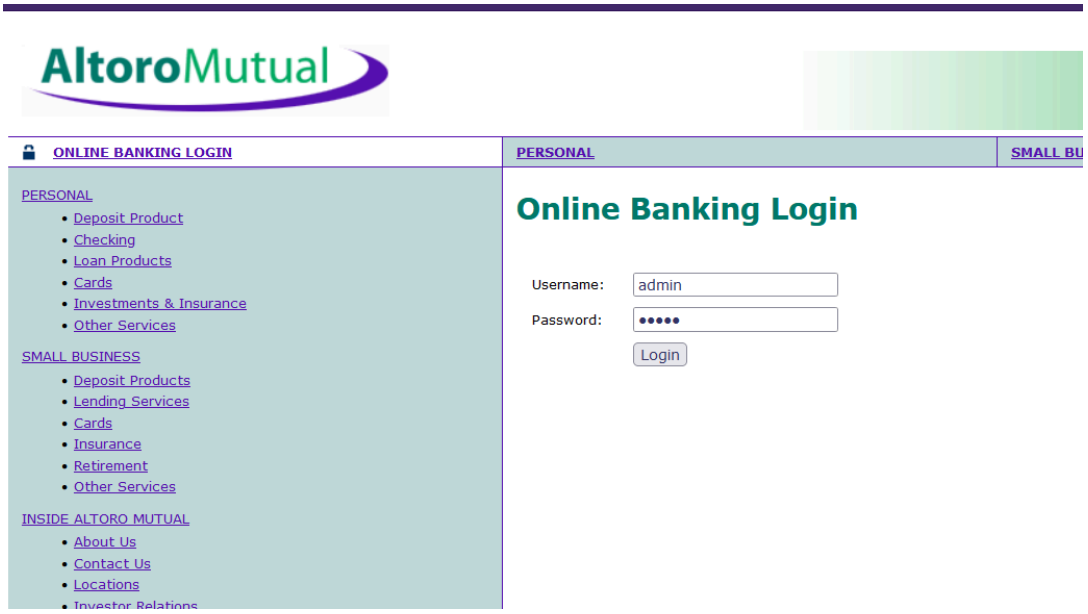
   - Access the URL



   - Now we will try to sign in to this website with admin privileges



- Also, we will be using burp suite to get requests from the website and know additional information.
- We use 'admin' for the username and password.



- This request has been received in the burp suite with the username and password as well in clear text.

```
Pretty   Raw   Hex
1  POST /doLogin HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 37
9  Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/login.jsp
12 Cookie: JSESSIONID=B6ADB36ACD5C83083787343A1F97F853; AltoroAccounts="0DAwMDAwfkNvcnBvcmF0ZX45LjQ30TA1MTE2MUU3fDgwMDAwMX5DaGVja2luZ34tNC4yMjc0NjkONTZFN3w="
13 Upgrade-Insecure-Requests: 1
14
15 uid=admin&passw=admin&btnSubmit=Login
```

- Now we click on forward request in the burp suite and then we will be redirected to the admin user details. Here in the burp suite, we can clearly see the login details in clear text. this is the clear indication of the vulnerability which can lead to data breach, monitored, and manipulated as well.

Recommendations:

- Educate users and employees on secure data handling practices, such as recognizing secure websites (look for "https") and avoiding insecure Wi-Fi networks.
- Employ data masking or redaction techniques to replace sensitive data with placeholders or cryptographic representations during transmission.
- Implement encryption for data in transit. Use secure encryption protocols such as TLS/SSL

to protect sensitive information during transmission.

- Use secure communication protocols for transmitting data, such as HTTPS for web applications, and ensure that the selected protocols are kept up-to-date.

9. Vulnerability name: Click

   Jacking CWE: 1021

   Description: The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the user is interacting with.

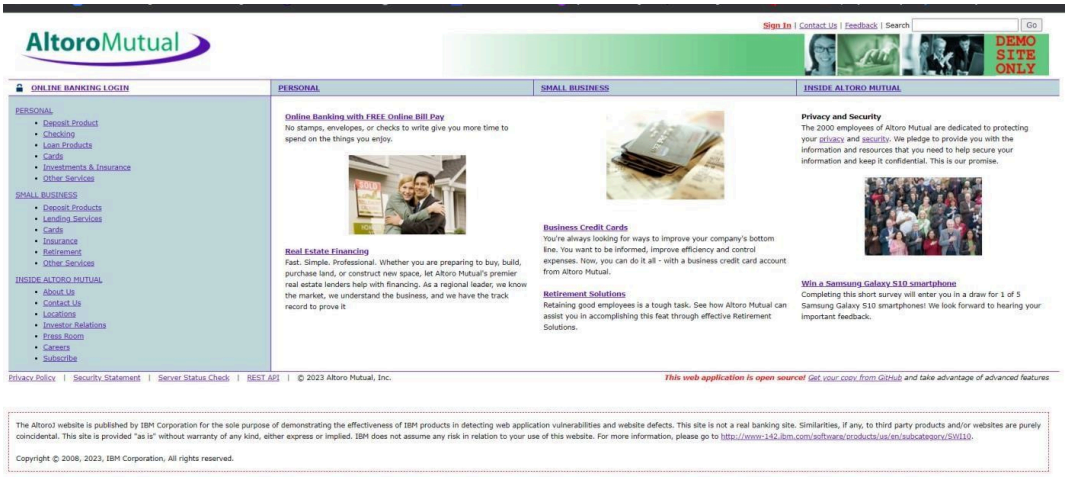
   Business Impact: To effectively reduce the business consequences resulting from CWE-1021, organizations must prioritize the implementation of protective measures, such as frame-busting code. Simultaneously, educating users on safe browsing practices plays a crucial role in preventing clickjacking incidents. Additionally, routine security assessments and testing are pivotal for identifying and mitigating vulnerabilities associated with clickjacking. These actions are of paramount importance in upholding user trust, ensuring data protection, and safeguarding the organization's reputation.

   Vulnerability path: http://testfire.net/

   Vulnerability parameter: http://testfire.net/

   Steps to reproduce:

   - Access the URL

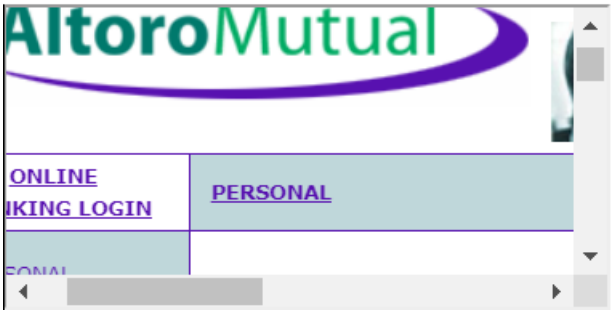


   - Then take the URL and use it for the html code
   - we will be writing some html code to perform this vulnerability. we will be writing the code in vs code for better flexibility and functionality.

```
<html>
<body>
<title>Click jacking vulnerability</title>
<h2>This website is vulnerable to clickjacking</h2>
<iframe  src="http://testfire.net/"></iframe>
</body>
</html>
```

   - After the code has been written then we will be executing the code in the browser.

This website is vulnerable to clickjacking



- From this image we can see that the vulnerability has been found.

Recommendation:

- Set the X-Frame-Options HTTP response header to deny or same-origin to control how your site can be framed. This is supported by most modern browsers.
- Utilize frame-busting JavaScript code in web applications to prevent the embedding of your site within malicious iframes. This code can disrupt clickjacking attempts.
- Implement a Content Security Policy to restrict which domains can embed your site in iframes. This can help prevent unauthorized framing.
- Implement additional security controls to prevent UI redress attacks, such as clickjacking, within your web application.

10. Vulnerability name: web server allows password

    auto-completion CWE: 310

    Description:
    Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.

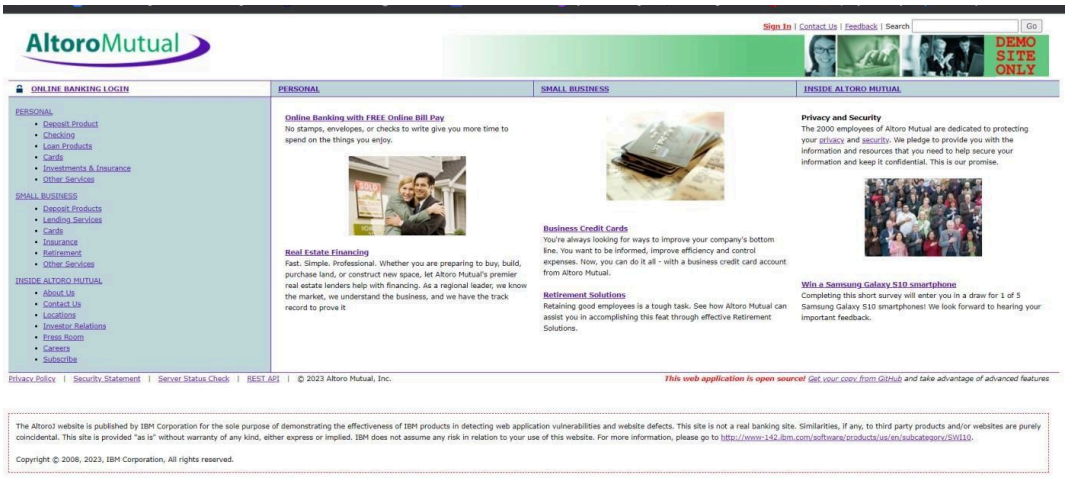
    Business impact:
    To effectively lessen the business consequences associated with CWE-310, organizations must prioritize the adoption of secure cryptographic practices. This entails ensuring proper password storage and robust encryption key management. Simultaneously, conducting routine security assessments and testing is pivotal in detecting and mitigating vulnerabilities related to cryptographic issues. Striving for compliance with pertinent data protection regulations and industry standards is equally essential. These actions are of paramount importance in the protection of sensitive data, the upholding of user trust, and the preservation of the organization's esteemed reputation.

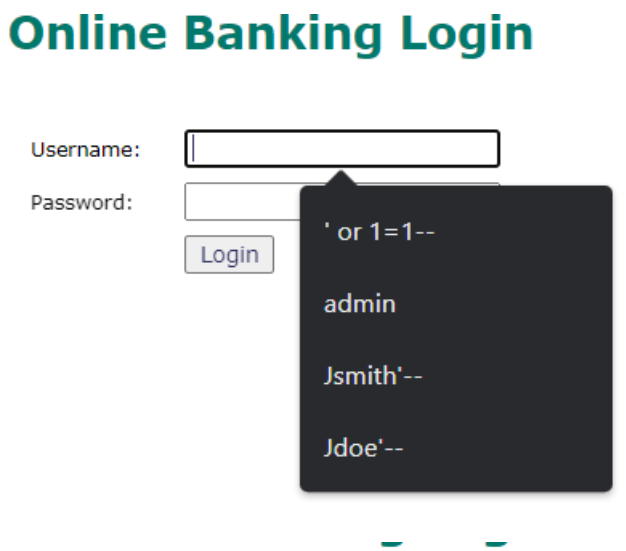
    Vulnerability path: http://testfire.net/

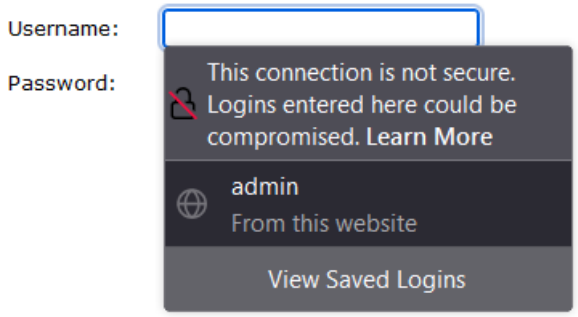
    Vulnerability parameter: http://testfire.net/login.jsp

    Steps to reproduce:

    - Access the URL







- From this image we can see the usernames and the passwords getting auto filled. This is a potential vulnerability as this can be a doorway for attackers.

Responsibilities:

- Implement secure password storage mechanisms, such as using strong and salted cryptographic hashing algorithms like bcrypt or scrypt.
- Implement robust encryption key management practices, including secure key storage, key rotation, and access controls.
- Utilize well-established and proven cryptographic libraries and algorithms for encryption and decryption.
- Use secure encryption protocols, like TLS/SSL, for data transmission over networks to protect data in transit.

11. Vulnerability name: Cross-site scripting (DOM based

XSS) CWE: 79

OWASP category: A03: Injections

Description: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser.

DOM-XSS: DOM stands for Document Object Model and is a programming interface for HTML and XML documents. It represents that programs can change the document structure, style and content. A webpage is a document and this document can be either displayed in the browser window or as the HTML Source.

Business Impact: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area displayed to many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

Vulnerability path: https://vit.ac.in/

Vulnerability parameter: https://vit.ac.in/

Steps to Reproduce:

- Access the URL

- We will try to perform the DOM XSS attack which is displaying the source code of the webpage.

- Now we can access the source code of the webpage in two ways by clicking on the inspect option or by simply pressing Ctrl + U.

```
Line wrap
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="alternate" type="application/rss+xml" title="VIT RSS" href="https://vit.ac.in/rss.xml" />
<link rel="shortcut icon" href="https://vit.ac.in/sites/all/themes/vittheme/favicon.ico" type="image/vnd.microsoft.icon" />
<meta name="description" content="One of the Best Educational Institution in India. Maintained an excellent placement records and gives training students to get
<meta name="abstract" content="VIT, the top university in India was established with the aim of providing quality higher education on par with international stan
<meta name="keywords" content="vit, vellore institute of technology, viteee, vit university, vit vellore, top engineering colleges in india, top 10 engineering c
<meta name="generator" content="Drupal" />
<link rel="canonical" href="https://vit.ac.in/" />
<meta property="og:site_name" content="Vellore Institute of Technology" />
<meta property="og:type" content="website" />
<meta property="og:url" content="https://vit.ac.in/" />
<meta property="og:title" content="Vellore Institute of Technology | A Place to Learn, Chance to grow" />
<meta property="og:description" content="One of the Best Educational Institution in India. Maintained an excellent placement records and gives training students
<meta property="og:image" content="https://vit.ac.in/sites/all/themes/vittheme/images/og-vit-updated.jpg" />
<meta property="og:image:url" content="https://vit.ac.in/sites/all/themes/vittheme/images/og-vit-updated.jpg" />
<meta name="dcterms.title" content="VIT" />
<meta name="dcterms.type" content="Text" />
<meta name="dcterms.format" content="text/html" />
<meta name="dcterms.identifier" content="https://vit.ac.in/" />
<meta name="facebook-domain-verification" content="9bbekg8iy8yy964rj4euqpwennaxoc" />
      <title>Vellore Institute of Technology | A Place to Learn, Chance to grow</title>
  <style type="text/css" media="screen">
@import url("https://vit.ac.in/sites/all/themes/vittheme/css/bootstrap.min.css?rdiap4");
@import url("https://vit.ac.in/sites/all/themes/vittheme/css/style.css?rdiap4");
@import url("https://vit.ac.in/sites/all/themes/vittheme/css/theme.css?rdiap4");
@import url("https://vit.ac.in/sites/all/themes/vittheme/css/combine.css?rdiap4");
@import url("https://vit.ac.in/sites/all/themes/vittheme/css/responsive.css?rdiap4");
</style>
<meta name="author" content="VIT University">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="language" content="English">
<meta name="copyright" content="https://vit.ac.in/"/>
<meta name="geo.region" content="IN-TN" />
<meta name="geo.placename" content="Vellore" />
```

- Many programs can help to alter this source code to find vulnerable spots.

- So, getting access to the source code of the webpage can lead to potential vulnerabilities as attackers can abuse the loop holes, exploits, vulnerabilities in the code and act accordingly.

Recommendation:

- When interacting with databases, use parameterized statements or prepared statements to avoid SQL injection, which can be a vector for XSS.
- Sanitize and validate all user inputs and ensure that any data displayed on the web page is properly encoded. Use output encoding libraries or functions to prevent script injection.
- Avoid using DOM manipulation methods that can introduce vulnerabilities. Be cautious with inner HTML, document. Write, and other methods that can execute scripts.
- Consider using well-established JavaScript frameworks and libraries that include built-in security features to prevent DOM XSS.

12. Vulnerability name: JQuery 1.2 < 3.5.0 Multiple XSS

    CWE: 79

    OWASP category: A03: Injections

    Description: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser.

    Stored XSS (Persistent XSS): In this type of attack, the malicious script is permanently stored on a server or in a database, often in user-generated content like comments or forum posts. When other users view the affected content, the script executes in their browsers, potentially compromising their data or sessions.

    Reflected XSS: Reflected XSS occurs when the malicious script is embedded in a URL and immediately executed when a victim clicks on a manipulated link. The script is not stored on a server but is reflected off a web application, making it a one-time attack.

    DOM-based XSS: DOM stands for Document Object Model and is a programming interface for HTML and XML documents. It represents that programs can change the document structure, style and content. A webpage is a document and this document can be either displayed in the browser window or as the HTML Source.

Business Impact: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area displayed to many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

Vulnerability path: https://vit.ac.in/

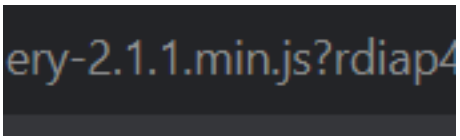
Vulnerability parameter: https://vit.ac.in/sites/all/themes/vittheme/js/jquery-2.1.1.min.js?rdiap4

Steps to Reproduce:

- Access the URL

- Try to access the URL https://vit.ac.in/sites/all/themes/vittheme/js/jquery- . This will lead us to the webpage with jquery script in it.

- We can also see the jquery version that has been used in the website



- We can access the source code of the jquery script that has been used in making the website, which can lead to multiple potential threats

Recommendation:

- The most important recommendation is to update to the latest version of jQuery. Newer versions typically include security patches, bug fixes, and performance improvements.
- Stay vigilant about jQuery updates and security announcements. Subscribe to relevant security mailing lists or forums to receive notifications about new releases.
- A Content Security Policy (CSP) can help mitigate various types of client-side vulnerabilities, including Cross-Site Scripting (XSS). Consider implementing a CSP in your web application to restrict the sources from which scripts can be loaded.
- If updating jQuery is not immediately possible, review your code for potential security issues. Ensure that you are escaping or sanitizing user inputs and that you are using safe coding practices to prevent XSS and other security vulnerabilities.

13. Vulnerability name: Information Exposure (Web Server HTTP Header Internal IP

Disclosure) CWE: 200

Description: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

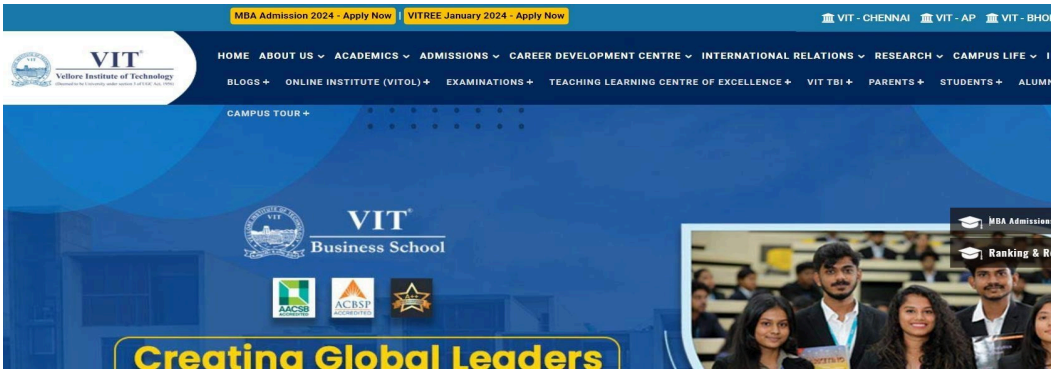
Business Impact: To effectively reduce the business consequences associated with CWE-200, organizations must prioritize the implementation of robust data access controls, secure data handling practices, and privacy protection measures. Regular security assessments and testing are instrumental in detecting and rectifying vulnerabilities related to information exposure. Furthermore, strict adherence to data protection regulations and industry standards is indispensable. These actions are of paramount importance in the protection of sensitive data, the preservation of user trust, and the upholding of the organization's esteemed reputation.

Vulnerability path: https://vit.ac.in/

Vulnerability parameter: https://vit.ac.in/

Steps to Reproduce:

- Access the URL



- We will be using a tool to perform a scan on this website. (Nessus Scanner)

LOW **Web Server HTTP Header Internal IP Disclosure**

Description

This may expose internal IP addresses that are usually hidden or masked behind a Network Address Translation (NAT) Firewall or proxy server.

There is a known issue with Microsoft IIS 4.0 doing this in its default configuration. This may also affect other web servers, web applications, web proxies, load balancers and through a variety of misconfigurations related to redirection.

Solution

Apply configuration suggested by vendor.

- Through this tool we can see the vulnerability in the scan along with detailed information regarding that vulnerability.

```
Nessus was able to exploit the issue using the following request :

GET / HTTP/1.0
Accept-Charset: iso-8859-1,utf-8;q=0.9,*;q=0.1
Accept-Language: en
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*



This produced the following truncated output (limited to 10 lines) :
------------------------------ snip ------------------------------
Location: https://10.10.7.35/
Content-Length: 0


------------------------------ snip ------------------------------
less...
```

- Now this image tells us the internal IP address of the target website, the attackers can easily exploit the website using the ip address.

Recommendation:

- Carefully review your web server's configuration files (e.g., Apache's httpd.conf, Nginx's nginx.conf) to identify anywhere internal IP addresses are inadvertently included in HTTP response headers.
- Remove or replace any internal IP addresses in the server configuration files with appropriate placeholders or public IP addresses.
- If you use proxy servers or load balancers, ensure that they are configured to hide internal IP addresses and only expose public IP addresses instances in HTTP headers.
- Review and configure HTTP response headers (e.g., Server, X-Powered-By) to limit the exposure of server and technology details, reducing the potential for disclosing internal IP addresses.

14. Vulnerability name: Information Exposure (SSL (Multiple

Issues)) CWE: 200

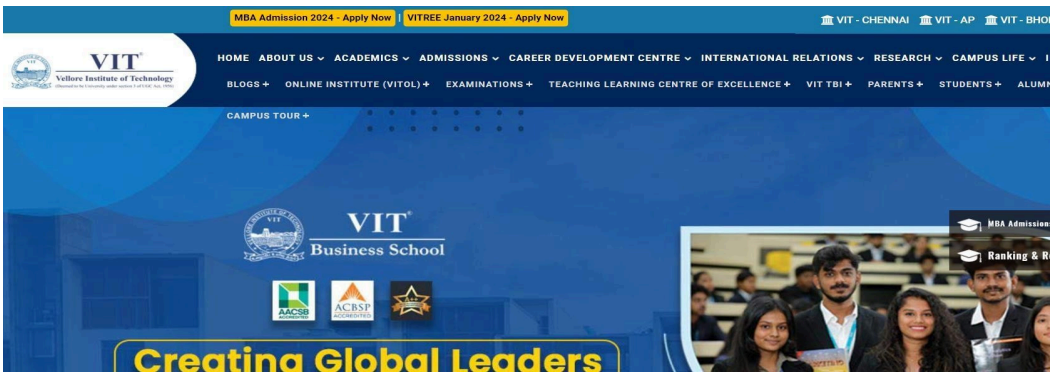
Description: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

Business Impact: To effectively reduce the business consequences arising from SSL issues, organizations must give utmost priority to establishing a secure SSL/TLS configuration, conducting frequent security assessments and testing. Adhering to data protection regulations and industry standards is imperative. These actions play a pivotal role in safeguarding sensitive data, upholding user trust, and safeguarding the organization's esteemed reputation.

Vulnerability path: https://vit.ac.in/

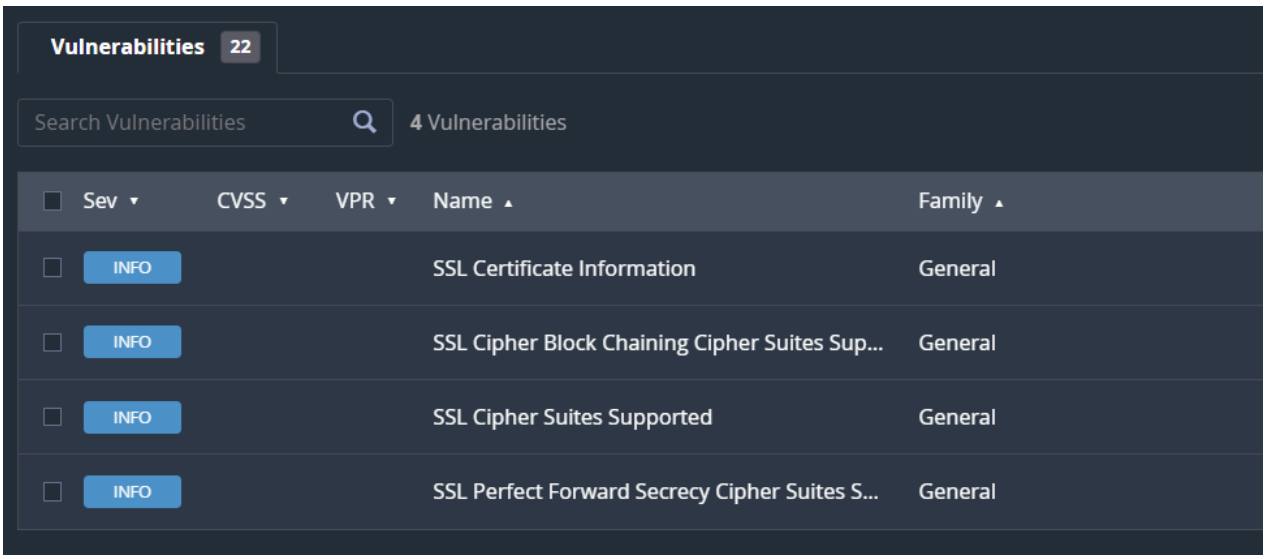
Vulnerability parameter: https://vit.ac.in/

Steps to Reproduce:



- Access the URL

- We will be using a tool to perform a scan on this website. (Nessus Scanner)



- Through this tool we can see the vulnerability in the scan along with detailed information regarding that vulnerability.

INFO SSL Certificate Information

Description

This plugin connects to every SSL-related port and attempts to extract and dump the X.509 certificate.

Output

```
Subject Name:

Common Name: *.vit.ac.in

Issuer Name:

Country: GB
State/Province: Greater Manchester
Locality: Salford
Organization: Sectigo Limited
Common Name: Sectigo RSA Domain Validation Secure Server CA

Serial Number: 46 53 67 B6 23 C5 BE EE B9 6E E2 C0 5F 46 C9 F7

Version: 3

Signature Algorithm: SHA-256 With RSA Encryption

Not Valid Before: Sep 04 00:00:00 2023 GMT
Not Valid After: Aug 03 23:59:59 2024 GMT

Public Key Info:

Algorithm: RSA Encryption
Key Length: 2048 bits
```

- Now this image tells us the type of encryption used in the target website with the public key details as well.

INFO SSL Cipher Block Chaining Cipher Suites Supported

Description

The remote host supports the use of SSL ciphers that operate in Cipher Block Chaining (CBC) mode. These cipher suites offer additional security over Electronic Codebook (ECB) mode, but have the potential to leak information if used improperly.

INFO SSL Cipher Suites Supported

Description

This plugin detects which SSL ciphers are supported by the remote service for encrypting communications.

INFO SSL Perfect Forward Secrecy Cipher Suites Supported

Description

The remote host supports the use of SSL ciphers that offer Perfect Forward Secrecy (PFS) encryption. These cipher suites ensure that recorded SSL traffic cannot be broken at a future date if the server's private key is compromised.

- Through this we can understand what different types of encryptions are used in the website.

```
Here is the list of SSL CBC ciphers supported by the remote server :

  High Strength Ciphers (>= 112-bit key)

    Name                          Code            KEX          Auth     Encryption                MAC
    ----------------------        ----------      ---          ----     ---------------------     ---
    ECDHE-RSA-CAMELLIA-CBC-128    0xC0, 0x76      ECDH         RSA      Camellia-CBC(128)         SHA256
    ECDHE-RSA-CAMELLIA-CBC-256    0xC0, 0x77      ECDH         RSA      Camellia-CBC(256)         SHA384
    DHE-RSA-AES128-SHA            0x00, 0x33      DH           RSA      AES-CBC(128)              SHA1
    DHE-RSA-AES256-SHA            0x00, 0x39      DH           RSA      AES-CBC(256)              SHA1
    DHE-RSA-CAMELLIA128-SHA       0x00, 0x45      DH           RSA      Camellia-CBC(128)         SHA1
    DHE-RSA-CAMELLIA256-SHA       0x00, 0x88      DH           RSA      Camellia-CBC(256)         SHA1
    DHE-RSA-SEED-SHA              0x00, 0x9A      DH           RSA      SEED-CBC(128)             SHA1
    ECDHE-RSA-AES128-SHA          0xC0, 0x13      ECDH         RSA      AES-CBC(128)              SHA1
    ECDHE-RSA-AES256-SHA          0xC0, 0x14      ECDH         RSA      AES-CBC(256)              SHA1
    AES128-SHA                    0x00, 0x2F      RSA          RSA      AES-CBC(128)              SHA1
    AES256-SHA                    0x00, 0x35      RSA          RSA      AES-CBC(256)              SHA1
    CAMELLIA128-SHA               0x00, 0x41      RSA          RSA      Camellia-CBC(128)         SHA1
    CAMELLIA256-SHA               0x00, 0x84      RSA          RSA      Camellia-CBC(256)         SHA1
    SEED-SHA                      0x00, 0x96      RSA          RSA      SEED-CBC(128)             SHA1
    DHE-RSA-AES128-SHA256         0x00, 0x67      DH           RSA      AES-CBC(128)              SHA256
    DHE-RSA-AES256-SHA256         0x00, 0x6B      DH           RSA      AES-CBC(256)              SHA256
    DHE-RSA-CAMELLIA128-SHA256    0x00, 0xBE      DH           RSA      Camellia-CBC(128)         SHA256
    DHE-RSA-CAMELLIA256-SHA256    0x00, 0xC4      DH           RSA      Camellia-CBC(256)         SHA256
    ECDHE-RSA-AES128-SHA256       0xC0, 0x27      ECDH         RSA      AES-CBC(128)              SHA256
    ECDHE-RSA-AES256-SHA384       0xC0, 0x28      ECDH         RSA      AES-CBC(256)              SHA384
    RSA-AES128-SHA256             0x00, 0x3C      RSA          RSA      AES-CBC(128)              SHA256
    RSA-AES256-SHA256             0x00, 0x3D      RSA          RSA      AES-CBC(256)              SHA256
    RSA-CAMELLIA128-SHA256        0x00, 0xBA      RSA          RSA      Camellia-CBC(128)         SHA256
    RSA-CAMELLIA256-SHA256        0x00, 0xC0      RSA          RSA      Camellia-CBC(256)         SHA256
```

Recommendation:

- Configure your web server to use the latest and strongest encryption protocols, such as TLS 1.2 or higher. Disable outdated and insecure protocols like SSLv3.
- Implement Perfect Forward Secrecy to ensure that even if the private key is compromised, previously recorded encrypted traffic remains secure.
- Safeguard your private keys by storing them securely and ensuring they are not accessible to unauthorized parties.

15. Vulnerability name: Information Exposure (HTTP (Multiple

Issues)) CWE: 200

Description: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

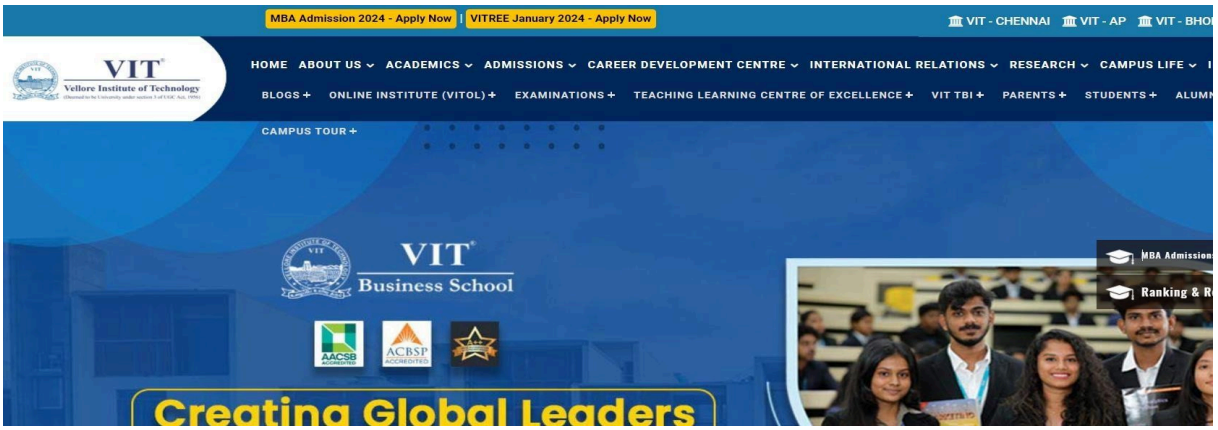
Business Impact: To effectively reduce the business repercussions of HTTP issues, organizations must place a strong emphasis on ongoing monitoring and enhancement of their web services. Consistent performance testing and security assessments, coupled with proactive maintenance, are indispensable to ensure a seamless and secure online presence. Furthermore, offering timely and helpful customer support and transparent communication when HTTP issues arise is instrumental in mitigating adverse effects on user trust and satisfaction.

Vulnerability path: https://vit.ac.in/

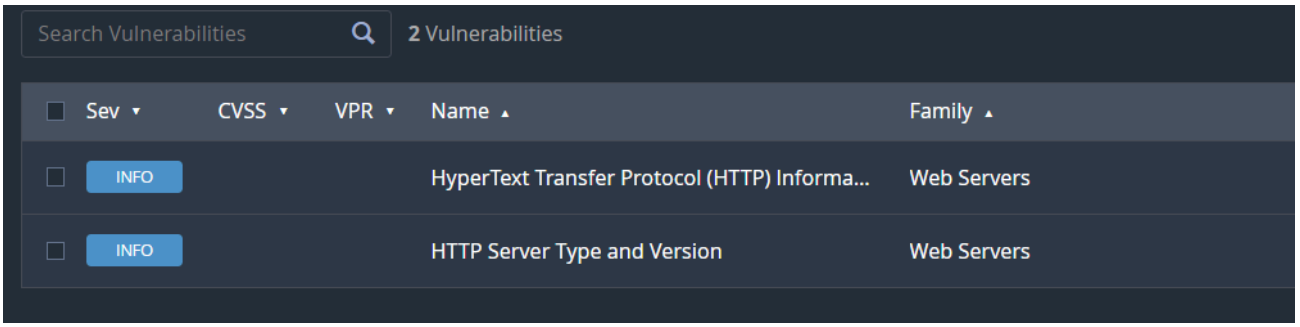
Vulnerability parameter: https://vit.ac.in/

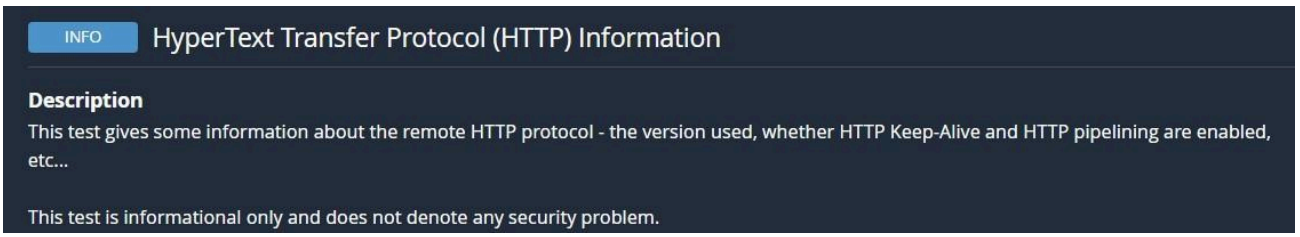
Steps to Reproduce:

- Access the URL



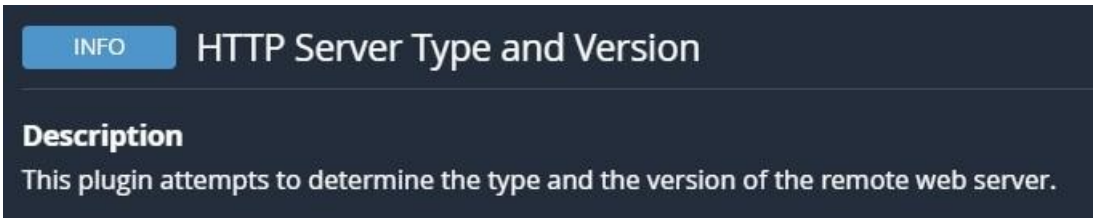
- We will be using a tool to perform a scan on this website. (Nessus Scanner)






```
Response Code : HTTP/1.1 307 Moved Temporarily

Protocol version : HTTP/1.1
SSL : no
Keep-Alive : no
Options allowed : (Not implemented)
Headers :

  Location: https://vit.ac.in/
  Content-Length: 0

Response Body :
```

- From the above image we can see the details of protocol version of the website and its status.




```
The remote web server type is :

Apache
```

- We can see the type of remote webserver here.
- All this information can lead to multiple potential threats such as data breaches by hackers.

Recommendation:

- Minimize the use of unnecessary redirects as they can increase page load times. Use 301 (permanent) redirects for consistent content.
- Regularly check for broken links (404 errors) and fix them to ensure users find the content they are looking for.
- Properly configure CORS to control which domains are allowed to make requests to your server.
- Migrate your website from HTTP to HTTPS to ensure secure data transmission. This is essential for user trust and search engine ranking.

16. Vulnerability name: OS Identification

Failed CWE: 200

Description: OS identification failed refers to the inability of a system or software to accurately recognize the operating system running on a device or network. This issue can result from various factors such as misconfigurations, network issues, or compatibility problems. The failure to identify the OS correctly can lead to security vulnerabilities, hindered software updates, and potential operational disruptions. Resolving this issue promptly is crucial to ensure accurate security measures and the smooth functioning of the system or network.

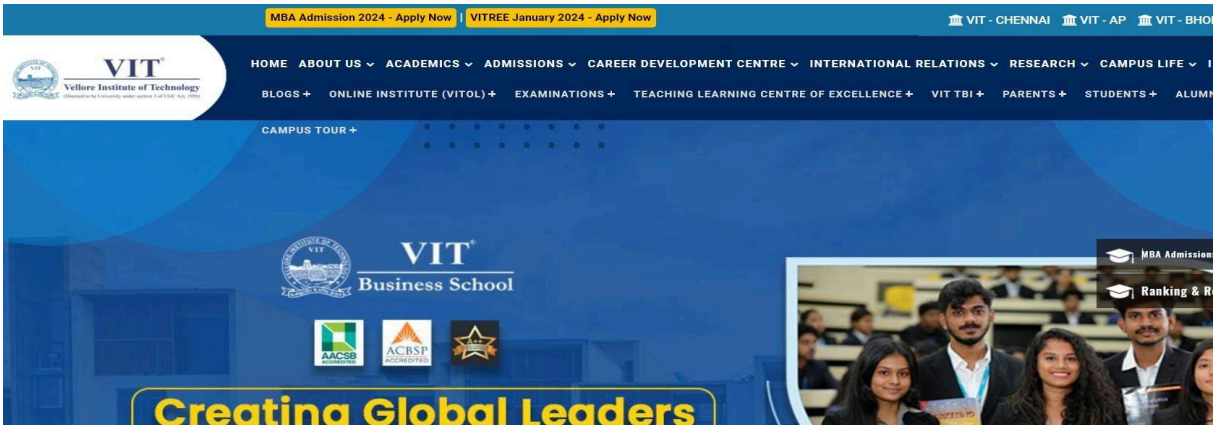
Business Impact: It may lead to operational inefficiencies, potential security vulnerabilities, non-compliance with industry regulations, and a possible loss of customer trust. Such failures could result in downtime, compromised data security, regulatory penalties, and damage to the company's reputation, potentially leading to financial losses and decreased market competitiveness. Resolving this issue promptly is crucial to mitigate these risks and maintain the smooth functioning of the organization.

Vulnerability path: https://vit.ac.in/

Vulnerability parameter: https://vit.ac.in/

Steps to Reproduce:

- Access the URL



- We will be using a tool to perform a scan on this website. (Nessus Scanner)

INFO **OS Identification Failed**

Description

Using a combination of remote probes (TCP/IP, SMB, HTTP, NTP, SNMP, etc), it was possible to gather one or more fingerprints from the remote system. Unfortunately, though, Nessus does not currently know how to use them to identify the overall system.

```
If you think these signatures would help us improve OS fingerprinting,
please send them to :

  os-signatures@nessus.org

Be sure to include a brief description of the device itself, such as
the actual operating system or product / model names.


SSLcert:!:i/CN:Sectigo RSA Domain Validation Secure Server CAi/O:Sectigo
Limiteds/CN:*.vit.ac.in
02ed800c9e7f764615d7e9ffd74b643a6cace446

SinFP:!:
   P1:B10113:F0x12:W65535:O0204ffff:M496:
   P2:B10113:F0x12:W65535:O0204ffff010303030101040201010 80affffffff44454144:M1300:
   P3:B00000:F0x00:W0:O0:M0
   P4:190701_7_p=80R
```

- From the above image, we can see the details of SSL Certificate Information and SinFP Information.
- If it is released without proper authorization, it can lead to serious consequences such as increased vulnerability to cyberattacks, potential data breaches, privacy violations, and legal repercussions.
- All this information can lead to multiple potential threats such as data breaches by hackers.

Recommendations:

- Employ robust data encryption methods for both data in transit and at rest.
- Implement strict access controls and authentication protocols to limit unauthorized access.
- Use secure communication channels, such as encrypted email and file transfer protocols.
- Provide comprehensive training to employees on data security best practices.
- Conduct regular security audits and assessments to identify and address vulnerabilities promptly.
- Ensure compliance with relevant data protection regulations and have a well-defined incident response plan in place.

NESSUS SCAN REPORT:

testfire.net

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	3	1	24

Vulnerabilities — Total: 28

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
MEDIUM	6.5	-	142960	HSTS Missing From HTTPS Server (RFC 6797)
MEDIUM	6.5	-	104743	TLS Version 1.0 Protocol Detection
MEDIUM	6.5	-	157288	TLS Version 1.1 Protocol Deprecated
LOW	3.7	4.5	83875	SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam)
INFO	N/A	-	46180	Additional DNS Hostnames
INFO	N/A	-	39446	Apache Tomcat Detection
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	54615	Device Type
INFO	N/A	-	84502	HSTS Missing From HTTPS Server
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	11936	OS Identification
INFO	N/A	-	10919	Open Port Re-check
INFO	N/A	-	56984	SSL / TLS Versions Supported
INFO	N/A	-	10863	SSL Certificate Information
INFO	N/A	-	95631	SSL Certificate Signed Using Weak Hashing Algorithm (Known CA)
INFO	N/A	-	70544	SSL Cipher Block Chaining Cipher Suites Supported

INFO	N/A	-	21643	SSL Cipher Suites Supported
INFO	N/A	-	57041	SSL Perfect Forward Secrecy Cipher Suites Supported
INFO	N/A	-	94761	SSL Root Certification Authority Certificate Information
INFO	N/A	-	156899	SSL/TLS Recommended Cipher Suites
INFO	N/A	-	22964	Service Detection
INFO	N/A	-	25220	TCP/IP Timestamps Supported
INFO	N/A	-	121010	TLS Version 1.1 Protocol Detection
INFO	N/A	-	136318	TLS Version 1.2 Protocol Detection
INFO	N/A	-	10287	Traceroute Information

* indicates the v3.0 score was not available; the v2.0 score is shown

vit.ac.in

0	0	1	1	26
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities — Total: 28

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
MEDIUM	6.1	5.7	136929	JQuery 1.2 < 3.5.0 Multiple XSS
LOW	3.1	2.2	10759	Web Server HTTP Header Internal IP Disclosure
INFO	N/A	-	48204	Apache HTTP Server Version
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	106658	JQuery Detection
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	42823	Non-compliant Strict Transport Security (STS)
INFO	N/A	-	50350	OS Identification Failed
INFO	N/A	-	66334	Patch Report
INFO	N/A	-	56984	SSL / TLS Versions Supported
INFO	N/A	-	10863	SSL Certificate Information
INFO	N/A	-	70544	SSL Cipher Block Chaining Cipher Suites Supported
INFO	N/A	-	21643	SSL Cipher Suites Supported
INFO	N/A	-	57041	SSL Perfect Forward Secrecy Cipher Suites Supported
INFO	N/A	-	94761	SSL Root Certification Authority Certificate Information
INFO	N/A	-	156899	SSL/TLS Recommended Cipher Suites

INFO	N/A	-	22964	Service Detection
INFO	N/A	-	42822	Strict Transport Security (STS) Detection
INFO	N/A	-	25220	TCP/IP Timestamps Supported
INFO	N/A	-	84821	TLS ALPN Supported Protocol Enumeration
INFO	N/A	-	136318	TLS Version 1.2 Protocol Detection
INFO	N/A	-	10287	Traceroute Information
INFO	N/A	-	100669	Web Application Cookies Are Expired
INFO	N/A	-	10386	Web Server No 404 Error Code Check
INFO	N/A	-	10302	Web Server robots.txt Information Disclosure

* indicates the v3.0 score was not available; the v2.0 score is shown

"With the Nessus scan report now at our disposal, we're equipped with critical data on the existing vulnerabilities and potential threats. This invaluable information will serve as a foundation for our project planning phase, as we chart the course to develop our AI-driven incident response platform with a well-informed strategy and timeline."

Stage – 2

Overview: -

An overview of a Nessus scan typically involves the following key steps and aspects:

1. Preparation: Define Objectives: Determine the scope and objectives of the scan. What systems or networks do you want to assess, and what specific vulnerabilities are you looking for?

   System Access: Ensure that you have the necessary permissions and access rights to scan the target systems or networks.

2. Configuration: Set Up Scan Policy: Create a scan policy that defines the parameters of the scan, such as the types of vulnerabilities to check for, scanning frequency, and specific configurations.

   Define Target Assets: Specify the IP addresses, hostnames, or network ranges of the systems you want to scan.

3. Scanning: Initiate Scan: Start the Nessus scan using the configured policy and target assets. Nessus will systematically scan the target systems for vulnerabilities based on its plugins and policies.

4. Vulnerability Identification: Nessus performs a comprehensive assessment, checking for vulnerabilities, misconfigurations, open ports, and other security issues. As the scan progresses, it categorizes vulnerabilities by severity, making it easier to prioritize remediation efforts.

5. Reporting: Nessus generates detailed reports that list the vulnerabilities discovered during the scan. Reports typically include information about the severity of each vulnerability, recommended actions for remediation, and supporting evidence. These reports can be customized and exported in various formats, such as PDF, HTML, or CSV.

6. Remediation: Review Findings: IT and security teams review the Nessus scan reports to prioritize and address the identified vulnerabilities.

   Remediation Steps: Take corrective actions to fix or mitigate the vulnerabilities. This may involve applying patches, changing configurations, or implementing security measures.

7. Follow-up: Regular Scanning: Conduct periodic Nessus scans to continuously monitor the security posture of the systems and networks. This helps identify new vulnerabilities and track progress in mitigating existing ones.

   Integration: Integrate Nessus with other security tools or workflows to automate the vulnerability management process and streamline the response to new findings.

8. Compliance Auditing (if applicable): If compliance checks are part of your scan policy, review compliance results and take actions to ensure systems meet the required standards and regulations.

9. Documentation: Maintain records of scan results, remediation efforts, and compliance checks for audit and reporting purposes.

10. User Training: Ensure that your IT and security teams are trained in using Nessus effectively and interpreting scan results.

Nessus scans are a critical component of proactive security, helping organizations identify and address vulnerabilities in their IT infrastructure to reduce the risk of security breaches and data compromises. The frequency and depth of scans can vary depending on an organization's security strategy and requirements.

Nessus is a popular vulnerability scanning tool used for identifying security weaknesses in computer systems and networks. It is widely utilized by security professionals and organizations to assess the security posture of

their systems and to proactively identify potential vulnerabilities that could be exploited by attackers. Here's what I understand about Nessus scans:

1. Vulnerability Assessment: Nessus performs a vulnerability assessment by scanning target systems or networks to identify security issues, such as missing patches, misconfigurations, weak passwords, and other potential weaknesses that could be exploited by attackers.

2. Network Scanning: Nessus can scan a wide range of network devices, including servers, routers, switches, firewalls, and more. It examines network services, open ports, and software running on target systems.

3. Plugin Architecture: Nessus uses a plugin-based architecture, allowing it to check for a wide variety of vulnerabilities. These plugins are regularly updated to include the latest security checks and support for new vulnerabilities.

4. Compliance Auditing: It can also be used to assess systems against various compliance standards and regulatory requirements, helping organizations ensure that their systems meet specific security standards, like PCI DSS or CIS benchmarks.

5. Customization: Nessus scans can be customized to suit the specific needs of an organization. Users can define scan policies, select the target systems, and specify the types of vulnerabilities to scan for.

6. Reporting: Nessus provides detailed reports that highlight vulnerabilities, their severity, and remediation recommendations. These reports are essential for IT and security teams to prioritize and address vulnerabilities.

7. Scanning Frequency: Organizations often perform regular Nessus scans to continuously monitor and improve their security posture. This is crucial for staying ahead of evolving threats and vulnerabilities.

8. Integration: Nessus can be integrated with other security tools and platforms, allowing organizations to automate the scanning process and streamline their vulnerability management.

9. Licensing: Nessus offers both free and commercial versions. The free version is limited in its functionality, while the commercial version provides more features and support.

10. User-Friendly Interface: Nessus offers an easy-to-use web-based interface, making it accessible to a wide range of users, from security experts to less experienced IT professionals.

It's important to note that Nessus is a valuable tool for identifying vulnerabilities, but it doesn't fix these vulnerabilities itself. It's up to the organization's IT and security teams to take the necessary steps to remediate the identified issues to enhance their security.

S. No	Vulnerability Name	Severity	Plugins
1.	Cross-site scripting (stored)	Low	Web application Scanner
2.	Broken Access Control	Medium	SSH weak HMAC Algorithms Supported
3.	SQL Injection	Medium	N/A
4.	Broken Authentication	Low	HSTS Missing from Server
5.	Insecure Direct Object Reference	Info	Service Detection
6.	Security Misconfiguration	Low	SSL cipher Block Chaining cipher suites
7.	Cross-Site Request Forgery	Low	HTTP Form- Based Authentication Detected

8.	Cleartext Transmission of Sensitive Information	Medium	Cleartext Transmission of Sensitive Information over HTTP
9.	ClickJacking	Info	HTML Rendering Layer Vulnerability

10.	webserver allows password auto-completion	Low	SSL/TLS Diffie-Hellman Modulus<=1024 bits(Logjam)

Target Website: testfire.net

Target IP Address: 54.83.41.36

Report:

1. VULNERABILITY NAME: Cross site scripting

   (stored) SEVERITY: Low

   PLUGIN: web application

   scanner PORT: N/A

   DESCRIPTION: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser.

   SOLUTION:
   - When interacting with databases, use parameterized statements or prepared statements to avoid SQL injection, which can be a vector for XSS.
   - Sanitize and validate all user inputs and ensure that any data displayed on the web page properly encoded. Use output encoding libraries or functions to prevent script injection.

   BUSINESS IMPACT: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

2. VULNERABILITY NAME: Broken access

   control SEVERITY: medium

   PLUGIN: SSH weak HMAC Algorithms supported

   PORT: N/A

DESCRIPTION: When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information exposures, denial of service, and arbitrary code execution.

SOLUTION:

- Ensure that the application has a robust authentication and authorization mechanism in place. Users should be authenticated before any sensitive actions are performed, and authorization checks should be conducted to verify that users have the necessary permissions for the requested operation.
- Ensure that session management is secure. Use strong session IDs, enforce session timeouts, and regenerate session tokens after login. Implement session fixation protection to prevent session hijacking.

BUSINESS IMPACT: wide-ranging business impact, including financial losses, reputation damage, legal consequences, operational disruption, and loss of customer trust. Addressing this vulnerability is crucial to protect the organization and its stakeholders from these potential negative effects.

3. VULNERABILITY NAME: SQL injection

   SEVERITY: medium

   PLUGIN: N/A (typically would be a security testing tool)

   PORT: port 80/443

   DESCRIPTION: The product constructs all or part of an SQL command using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

   SOLUTION:
   - If parameterized statements are not feasible, use proper input validation and escaping mechanisms to sanitize user inputs before they are used in SQL queries. This helps prevent malicious code injection.
   - Utilize stored procedures for database operations. This can help separate SQL code from application code and reduce the risk of SQL injection.
   - Implement strict input validation for all user inputs, ensuring that data adheres to the expected format and structure.

   BUSINESS IMPACT: In summary, it is crucial to underscore that CWE-89, known as SQL Injection, can exert a profound and diverse business impact. This encompasses critical facets such as data breaches, financial setbacks, harm to reputation, legal ramifications, and operational turmoil.
   Hence, the imperative of preventing and remedying SQL injection vulnerabilities cannot be overstated, as it is indispensable for fortifying the security and continuity of an organization's applications and data.

4. VULNERABILITY NAME: Broken authentication

   SEVERITY: Low

   PLUGIN: HSTS Missing from HTTPS Server (plugin id:142960)

   PORT: port 443

DESCRIPTION: The system's authorization functionality does not prevent one user from gaining access to another user's data or record by modifying the key value identifying the data.

SOLUTION:

- Implement strong authentication mechanisms, such as multi-factor authentication (MFA) or two-factor authentication (2FA), to ensure that only authorized users can access the system.
- Implement proper authorization controls to ensure that users have the necessary permissions to access specific functions or data. Use role-based access control (RBAC) to manage user privileges effectively.
- Deploy intrusion detection and prevention systems (IDS/IPS) to monitor and block suspicious activities related to authentication and authorization.

BUSINESS IMPACT: To effectively mitigate the business impact of CWE-285, it is imperative that organizations place a strong emphasis on bolstering their authentication and session management practices. This should encompass the adoption of multi-factor authentication, the secure storage of credentials, meticulous session handling, and a commitment to conducting routine security assessments and testing. The rectification of these vulnerabilities stands as a critical imperative, safeguarding sensitive data, user identities, and the organization's overarching security stature and reputation.

5. VULNERABILITY NAME: Insecure Direct object

   Reference SEVERITY: Info

   PLUGIN: Service Detection (plugin

   id:22964) PORT: port 80/443

   DESCRIPTION: Insecure Direct Object Reference (IDOR) is a vulnerability that arises when attackers can access or modify objects by manipulating identifiers used in a web application's URLs or parameters. It occurs due to missing access control checks, which fail to verify whether a user should be allowed to access specific data.

   SOLUTION:

   - Utilize parameterized queries or prepared statements when interacting with databases. This prevents user-controlled input from being executed as SQL code.
   - Implement a Content Security Policy to mitigate the risk of cross-site scripting (XSS) attacks, which can be used to execute unauthorized queries.
   - Implement account lockout mechanisms that temporarily suspend accounts after a specified number of unauthorized access attempts to prevent brute force attacks.

   BUSINESS IMPACT: To effectively mitigate the business impact of CWE-639, organizations must make it a top priority to fortify their access control and authorization mechanisms. This entails the implementation of robust security measures, the regular conduct of comprehensive security assessments, and the deployment of intrusion detection systems to promptly identify and counter unauthorized access attempts. These measures stand as critical in the defence of sensitive data, the overall security of systems, and the preservation of the organization's esteemed reputation.

6. VULNERABILITY NAME: Security Misconfiguration

SEVERITY: low

PLUGIN: SSL cipher Block chaining cipher suites

PORT: port 443

DESCRIPTION: The product specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

SOLUTION:

- Implement a change management process to document and review all changes to system configurations. This helps prevent unauthorized or unintended alterations.
- Follow security guidelines provided by software and hardware vendors to ensure secure configurations.
- Regularly perform compliance checks against industry standards and regulations to ensure that configurations remain in compliance with security requirements.

BUSINESS IMPACT: To effectively lessen the business consequences stemming from CWE-732, organizations must place a strong emphasis on bolstering their configuration management practices. This involves the establishment of secure default settings, the routine conduct of comprehensive security evaluations, and the deployment of intrusion detection systems to promptly detect and address misconfigurations and vulnerabilities. These actions are of paramount importance in safeguarding sensitive data, fortifying system security, and preserving the organization's esteemed reputation.

7. VULNERABILITY NAME: Cross-Site Request

   Forgery SEVERITY: Low

   PLUGIN: HTTP Form-Based Authentication

   Detected PORT: port 8080/443

   DESCRIPTION: Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

   SOLUTION:

   - Implement Synchronizer Token Patterns to include unique, randomly generated tokens in each HTTP request and validate these tokens on the server side to verify the authenticity of the request.
   - Utilize the Same Site attribute in cookies to restrict their usage to same-site requests only, preventing them from being sent along with cross-site requests and effectively mitigating CSRF attacks.

- Utilize framework-specific protections and security features to prevent CSRF attacks. Many modern web frameworks have built-in mechanisms and libraries for handling CSRF vulnerabilities effectively.
- Implement a Content Security Policy to restrict the sources from which various types of content can be loaded. By specifying the trusted sources of content, you can minimize the risk of malicious code execution and reduce the likelihood of successful CSRF attacks.

BUSINESS IMPACT: To effectively reduce the business repercussions of CWE-352, organizations must prioritize secure coding practices, including the incorporation of anti-CSRF tokens, while also conducting routine security assessments and testing to detect and rectify CSRF vulnerabilities. Furthermore, educating users on secure browsing practices is pivotal in preventing CSRF attacks. These actions are of paramount importance in the protection of data and in upholding the trust of customers and partners.

8. VULNERABILITY NAME: Cleartext Transmission of Sensitive

   Information SEVERITY: Medium

   PLUGIN: Cleartext Transmission of Sensitive Information over

   HTTP PORT: port 443

   DESCRIPTION: The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors. Many communication channels can be "sniffed" (monitored) by adversaries during data transmission. For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data. As a result, network traffic could be sniffed by adversaries, spilling security-critical data.

   SOLUTION:
   - Educate users and employees on secure data handling practices, such as recognizing secure websites (look for "https") and avoiding insecure Wi-Fi networks.
   - Employ data masking or redaction techniques to replace sensitive data with placeholders or cryptographic representations during transmission.
   - Implement encryption for data in transit. Use secure encryption protocols such as TLS/SSL to protect sensitive information during transmission.
   - Use secure communication protocols for transmitting data, such as HTTPS for web applications, and ensure that the selected protocols are kept up to date.

   BUSINESS IMPACT: To effectively reduce the business consequences associated with CWE-319, organizations must prioritize the adoption of secure data transmission practices. This includes the utilization of encryption and robust, secure protocols. The routine conduct of security assessments and testing is pivotal in pinpointing and remedying vulnerabilities linked to data transmission. Furthermore, the education of users on secure data handling practices plays a vital role in proactively preventing data exposure incidents. These actions are of paramount importance in the protection of sensitive data and in preserving the trust of both customers and partners.

9. VULNERABILITY NAME: Clickjacking

SEVERITY: Info

PLUGIN: HTML Rendering Layer

Vulnerability PORT: port 80/443

DESCRIPTION: The web application does not restrict or incorrectly restricts frame objects or UI layers that belong to another application or domain, which can lead to user confusion about which interface the user is interacting with.

SOLUTION:

- Set the X-Frame-Options HTTP response header to deny or same-origin to control how your site can be framed. This is supported by most modern browsers.
- Utilize frame-busting JavaScript code in web applications to prevent the embedding of your site within malicious frame's. This code can disrupt clickjacking attempts.
- Implement a Content Security Policy to restrict which domains can embed your site in iframes. This can help prevent unauthorized framing.
- Implement additional security controls to prevent UI redress attacks, such as clickjacking, within your web application.

BUSINESS IMPACT: To effectively reduce the business consequences resulting from CWE-1021, organizations must prioritize the implementation of protective measures, such as frame-busting code. Simultaneously, educating users on safe browsing practices plays a crucial role in preventing clickjacking incidents. Additionally, routine security assessments and testing are pivotal for identifying and mitigating vulnerabilities associated with clickjacking. These actions are of paramount importance in upholding user trust, ensuring data protection, and safeguarding the organization's reputation.

10. VULNERABILITY NAME: web server allows password auto-completion

SEVERITY: Low

PLUGIN: SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam)

PORT: port 443

DESCRIPTION: Weaknesses in this category are related to the design and implementation of data confidentiality and integrity. Frequently these deal with the use of encoding techniques, encryption libraries, and hashing algorithms. The weaknesses in this category could lead to a degradation of the quality data if they are not addressed.

SOLUTION:

- Implement secure password storage mechanisms, such as using strong and salted cryptographic hashing algorithms like crypt or script.
- Implement robust encryption key management practices, including secure key storage, key rotation, and access controls.
- Utilize well-established and proven cryptographic libraries and algorithms for encryption and decryption.

- Use secure encryption protocols, like TLS/SSL, for data transmission over networks to protect data in transit

BUSINESS IMPACT: To effectively lessen the business consequences associated with CWE-310, organizations must prioritize the adoption of secure cryptographic practices. This entails ensuring proper password storage and robust encryption key management. Simultaneously, conducting routine security assessments and testing is pivotal in detecting and mitigating vulnerabilities related to cryptographic issues. Striving for compliance with pertinent data protection regulations and industry standards is equally essential. These actions are of paramount importance in the protection of sensitive data, the upholding of user trust, and the preservation of the organization's esteemed reputation.

S. NO	VULNERABILITY NAME	SEVERITY	PLUGINS
1.	Cross-site scripting (DOM based XSS)	Medium	Multiple XSS
2.	jQuery 1.3<3.5.0 Multiple XSS	Medium	CGI Abuse: XSS
3.	Information exposure (web server HTTP header internal IP disclosure)	Low	Webserver HTTP Header Internal IP Disclosure
4.	Information exposure (SSL (Multiple Issues))	Info	SSL Certificate Information
5.	Information exposure (HTTP (Multiple Issues))	Info	HTTP Information
6.	OS identification failed	Info	TCP/IP Remote Probing

TARGET WEBSITE: vit.ac.in TARGET

IP ADDRESS: 122.184.65.22

Report:

1. VULNERABILITY NAME: Cross-site scripting (DOM based

   XSS) SEVERITY: Medium

   PLUGIN: Multiple

   XSS PORT: port 443

   DESCRIPTION: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser. DOM-XSS: DOM stands for Document Object Model and is a programming interface for HTML and XML documents. It represents that programs can change the document structure, style and content. A webpage is a document, and this document can be either displayed in the browser window or as the HTML Source.

SOLUTION:

- When interacting with databases, use parameterized statements or prepared statements to avoid SQL injection, which can be a vector for XSS.
- Sanitize and validate all user inputs and ensure that any data displayed on the web page is properly encoded. Use output encoding libraries or functions to prevent script injection.
- Avoid using DOM manipulation methods that can introduce vulnerabilities. Be cautious with inner HTML, document. Write, and other methods that can execute scripts.
- Consider using well-established JavaScript frameworks and libraries that include built-in security features to prevent DOM XSS.

BUSINESS IMPACT: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area displayed to many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

2. VULNERABILITY NAME: jQuery 1.2 < 3.5.0 Multiple XSS

   SEVERITY: Medium

   PLUGIN: CGI Abuse: XSS

   PORT: port 443

   DESCRIPTION: It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflection of a malicious script of a web application, onto a user's browser. Stored XSS (Persistent XSS): In this type of attack, the malicious script is permanently stored on a server or in a database, often in user-generated content like comments or forum posts. When other users view the affected content, the script executes in their browsers, potentially compromising their data or sessions. Reflected XSS: Reflected XSS occurs when the malicious script is embedded in a URL and immediately executed when a victim clicks on a manipulated link. The script is not stored on a server but is reflected off a web application, making it a one-time attack. DOM-based XSS: DOM stands for Document Object Model and is a programming interface for HTML and XML documents. It represents that programs can change the document structure, style and content. A webpage is a document, and this document can be either displayed in the browser window or as the HTML Source.

   SOLUTION:

   - The most important recommendation is to update to the latest version of jQuery. Newer versions typically include security patches, bug fixes, and performance improvements.
   - Stay vigilant about jQuery updates and security announcements. Subscribe to relevant security mailing lists or forums to receive notifications about new releases.
   - A Content Security Policy (CSP) can help mitigate various types of client-side vulnerabilities, including Cross-Site Scripting (XSS). Consider implementing a CSP in your web application to restrict the sources from which scripts can be loaded.

- If updating jQuery is not immediately possible, review your code for potential security issues. Ensure that you are escaping or sanitizing user inputs and that you are using safe coding practices to prevent XSS and other security vulnerabilities.

BUSINESS IMPACT: The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. Later, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area displayed to many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

3. VULNERABILITY NAME: Information Exposure (Web Server HTTP Header Internal IP Disclosure)

   SEVERITY: Low

   PLUGIN: Web Server HTTP Header Internal IP Disclosure

   PORT: port 80

   DESCRIPTION: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

   SOLUTION:
   - Carefully review your web server's configuration files (e.g., Apache's httpd.conf, Nginx's nginx.conf) to identify anywhere internal IP addresses are inadvertently included in HTTP response headers.
   - Remove or replace any internal IP addresses in the server configuration files with appropriate placeholders or public IP addresses.
   - If you use proxy servers or load balancers, ensure that they are configured to hide internal IP addresses and only expose public IP addresses instances in HTTP headers.
   - Review and configure HTTP response headers (e.g., Server, X-Powered-By) to limit the exposure of server and technology details, reducing the potential for disclosing internal IP addresses.

   BUSINESS IMPACT: To effectively reduce the business consequences arising from SSL issues, organizations must give utmost priority to establishing a secure SSL/TLS configuration, conducting frequent security assessments and testing. Adhering to data protection regulations and industry standards is imperative. These actions play a pivotal role in safeguarding sensitive data, upholding user trust, and safeguarding the organization's esteemed reputation.

4. VULNERABILITY NAME: Information Exposure (SSL (Multiple Issues))

   SEVERITY: Info

   PLUGIN: SSL certificate

   Information PORT: port 443

DESCRIPTION: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.
SOLUTION:

- Configure your web server to use the latest and strongest encryption protocols, such as TLS 1.2 or higher. Disable outdated and insecure protocols like SSLv3.
- Implement Perfect Forward Secrecy to ensure that even if the private key is compromised, previously recorded encrypted traffic remains secure.
- Safeguard your private keys by storing them securely and ensuring they are not accessible to unauthorized parties.

BUSINESS IMPACT: To effectively reduce the business consequences arising from SSL issues, organizations must give utmost priority to establishing a secure SSL/TLS configuration, conducting frequent security assessments and testing. Adhering to data protection regulations and industry standards is imperative. These actions play a pivotal role in safeguarding sensitive data, upholding user trust, and safeguarding the organization's esteemed reputation.

5. VULNERABILITY NAME: Information Exposure (HTTP (Multiple

   Issues)) SEVERITY: Info

   PLUGIN: HTTP

   information PORT: port 80

   /443

   DESCRIPTION: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information

   SOLUTION:

   - Minimize the use of unnecessary redirects as they can increase page load times. Use 301 (permanent) redirects for consistent content.
   - Regularly check for broken links (404 errors) and fix them to ensure users find the content they are looking for.
   - Properly configure CORS to control which domains are allowed to make requests to your server.
   - Migrate your website from HTTP to HTTPS to ensure secure data transmission. This is essential for user trust and search engine ranking.

   BUSINESS IMPACT: To effectively reduce the business repercussions of HTTP issues, organizations must place a strong emphasis on ongoing monitoring and enhancement of their web services. Consistent performance testing and security assessments, coupled with proactive maintenance, are indispensable to ensure a seamless and secure online presence. Furthermore, offering timely and helpful customer support and transparent communication when HTTP issues arise is instrumental in mitigating adverse effects on user trust and satisfaction.

6. VULNERABILITY NAME: OS Identification

   Failed SEVERITY: Info

PLUGIN: TCP/IP remote

probing PORT: N/A

DESCRIPTION: OS identification failed refers to the inability of a system or software to accurately recognize the operating system running on a device or network. This issue can result from various factors such as misconfigurations, network issues, or compatibility problems. The failure to identify the OS correctly can lead to security vulnerabilities, hindered software updates, and potential operational disruptions. Resolving this issue promptly is crucial to ensure accurate security measures and the smooth functioning of the system or network.

SOLUTION:

- Minimize the use of unnecessary redirects as they can increase page load times. Use 301 (permanent) redirects for consistent content.
- Regularly check for broken links (404 errors) and fix them to ensure users find the content they are looking for.
- Properly configure CORS to control which domains are allowed to make requests to your server.
- Migrate your website from HTTP to HTTPS to ensure secure data transmission. This is essential for user trust and search engine ranking.

BUSINESS IMPACT: It may lead to operational inefficiencies, potential security vulnerabilities, non-compliance with industry regulations, and a possible loss of customer trust. Such failures could result in downtime, compromised data security, regulatory penalties, and damage to the company's reputation, potentially leading to financial losses and decreased market competitiveness. Resolving this issue promptly is crucial to mitigate these risks and maintain the smooth functioning of the organization.

Title:

Design and apply context-aware access controls based on user identity, device posture, location, and application risk.

Context-aware access controls represent the operational pinnacle of a Zero Trust architecture, moving beyond static rules to create a dynamic and adaptive security perimeter around every application. In a modern enterprise, where users are mobile, devices are diverse, and threats are sophisticated, granting access based solely on a username and password is a dangerous oversimplification. Designing and applying context-aware controls means building a policy engine that consumes multiple, real-time signals about an access request and renders an intelligent decision: to allow, to block, to limit, or to challenge. This approach ensures that the security posture is not a flat, always-on gate but a responsive system that tightens or loosens based on the risk of the specific moment.

The design of such a system begins with a deep integration between the access broker, such as Zscaler Private Access (ZPA), and the organization's existing security and identity infrastructure. The first and most fundamental layer is user identity. This is not simply a username, but a rich set of attributes pulled from the corporate Identity Provider (IdP) like Azure AD or Okta. Integration via SAML ensures that when a user authenticates through the ZPA client, they are verified with multi-factor authentication (MFA) by the IdP. ZPA then ingests not just the user's name, but their group memberships (e.g., "Finance – Payroll Managers," "Engineering – Source Code Contributors"), roles, and department. This allows for the creation of granular policies that are intrinsically tied to the user's function within the organization. For instance, a policy can be written in plain language: "Allow members of the 'Finance-Payroll' group to access the 'ADP Payroll' application."

However, a valid identity on a compromised device is still a critical threat. This is where the second layer, device posture, becomes essential. The design incorporates continuous checks on the security health of the endpoint attempting the connection. Through integrations with Endpoint Detection and Response (EDR) tools like CrowdStrike or SentinelOne, and Mobile Device Management (MDM) solutions like JAMF or Intune, ZPA can assess a device's real-time hygiene. The policy engine evaluates factors such as: Is the device's disk encryption (BitLocker/FileVault) enabled? Is the antivirus software active and its definition database up-to-date? Has the operating system been patched to the latest security baseline? Is the device rooted or jailbroken? Is there a high-risk score being reported by the EDR tool due to suspicious activity? If a device fails these checks—for example, a user's laptop has its firewall disabled—the context-aware policy can automatically trigger an action. It might block access entirely, or it might grant "remediation access," routing the user to a captive portal to download updates or contact IT, while preventing them from reaching any business-critical applications. This ensures that a vulnerable endpoint cannot become a beachhead for an attacker to move laterally into the corporate environment.

The third contextual layer is location and network environment. The physical or virtual location from which a user connects provides a powerful signal for risk assessment. A finance director accessing the payroll system from their home office in the same city is a low-risk event. The same user attempting to connect from a country with a high prevalence of cybercrime, or from an anonymizing VPN service, should be treated with extreme suspicion. Context-aware policies can be designed to enforce this logic. For example, access to a source code repository containing intellectual property could be restricted to users connecting from a corporate office or a pre-approved home IP address range. If a legitimate user is traveling, the policy might allow access but require a step-up authentication, such as a hardware token, or grant only read-only permissions. This layer can also consider the security of the network itself; a connection from a public, unencrypted Wi-Fi hotspot might trigger stricter controls than one from a known, secure corporate network.

The final and often most strategic layer is application risk. Not all applications are created equal, and their access policies should reflect their value and sensitivity. In the design phase, applications are categorized and tagged based on their risk profile. An internal wiki used for cafeteria menus would be classified as low-risk, whereas a customer database, a financial trading platform, or a source code repository would be classified as high or critical risk. This tagging allows policies to be layered and dynamic. A single user might have access to multiple applications, but the controls applied to each session will differ based on the application's risk. For the low-risk wiki, a valid user identity and a compliant device might be sufficient. However, for the high-risk financial platform, the policy engine might require all four layers to align: the user must be in a specific group, on a fully patched and managed device, connecting from a trusted location, and may even be subject to session recording or time-of-day restrictions. This ensures that security resources and scrutiny are applied proportionally to the actual risk, preventing "blanket" access and ensuring that the most valuable assets have the most robust defenses.

In practice, applying these controls within ZPA means using the policy builder to create conditional statements. A policy might read: "Allow access to 'Application A' if User is in 'Group X' AND Device Posture is 'Compliant' AND Location is 'Not Restricted' AND Application Risk is 'Low'." Alternatively, a more complex policy for a critical application might be: "Allow access to 'Application B' if User is in 'Group Y' AND Device Posture is 'Highly Secure' AND Location is 'Corporate Network' AND Time is 'Business Hours', otherwise, if Location is 'Remote', require Step-Up Authentication." By combining these signals, the organization creates a security posture that is finely tuned, highly resilient, and fundamentally adaptive to the context of every single access attempt, thereby upholding the core Zero Trust principle of "never trust, always verify."

Conclusion:

Stage 1: In conclusion, web application testing is a crucial process that ensures the security, functionality, and reliability of web-based software. It involves a comprehensive evaluation of various aspects, including user interface, functionality, performance, and security to identify and rectify issues before they impact end-users. This process not only enhances the quality of web applications but also bolsters their resilience against potential threats and vulnerabilities. In an age where online services and applications are integral to our daily lives, effective web application testing is essential to deliver a seamless and secure user experience.

Stage 2: The Nessus report for the specific website has highlighted key security insights. It identified vulnerabilities, categorized by severity, with a focus on critical and high-risk issues that need immediate attention. Recommendations for addressing these vulnerabilities include patching, reconfiguration, and other security measures, with a priority on actions that can have the most significant impact. Compliance with industry standards and best practices is considered, ensuring alignment with recognized security benchmarks like OWASP guidelines. The report underscores the need for continuous monitoring, regular updates, and ongoing security efforts to maintain a strong security posture. In summary, the Nessus report offers a clear view of the website's security state, provides guidance on mitigating vulnerabilities, and stresses the importance of sustained vigilance against evolving threats.

Stage 3:

The implementation of Zscaler Private Access marks a pivotal shift from a vulnerable, network-centric security model to a resilient, identity-centric Zero Trust architecture. By enforcing least-privilege access and rendering private applications invisible to the internet, this project effectively neutralizes the threats of lateral movement and network-based attacks. The solution not only fortifies the organization's security posture against modern cyber threats but also delivers a seamless and high-performance experience for a distributed workforce. Furthermore, the decoupling of access from the network simplifies IT operations, reduces dependency on legacy VPN infrastructure, and provides granular visibility into user-application interactions. Ultimately, this initiative establishes a future-ready foundation for secure digital transformation..

Future Scope:

Stage 1: future scope of web application testing

The future scope of web application testing is promising and evolving rapidly, driven by the continuous growth of the internet and web-based applications.

Here are some key aspects of its future scope:

- Increased Demand: As more businesses and services move online, the demand for web applications continues to grow. This, in turn, increases the demand for web application testing to ensure security, reliability, and functionality.
- Advanced Technologies: The complexity of web applications is rising with the integration of advanced technologies like AI, IoT, and blockchain. Testing these cutting-edge applications will require specialized skills and tools.
- Security Emphasis: With the growing number of cyber threats and data breaches, security testing for web applications will become even more critical. Testers will need to focus on vulnerabilities, data protection, and compliance with regulations like GDPR.
- Mobile and Cross-Browser Testing: Mobile usage is on the rise, and web applications must be responsive and functional on various devices and browsers. Testing for cross-compatibility will be essential.
- Performance Testing: As user expectations for speed and performance increase, load testing, stress testing, and other performance testing types will be crucial to ensure web applications can handle traffic spikes.

Stage 2: future scope of the testing process you understood.

The future scope of the web application testing process, particularly in terms of identifying vulnerabilities, is exceedingly promising due to the continuous growth of digital platforms, the escalating importance of cybersecurity, and the ever-evolving threat landscape. Here are some key aspects that highlight the prospective scope of web application testing for vulnerability discovery:

- AI-Driven Testing: The integration of artificial intelligence (AI) and machine learning (ML) into web application testing tools will revolutionize the way vulnerabilities are identified. AI algorithms can swiftly scan vast amounts of code and data to pinpoint vulnerabilities that might elude human testers.
- IoT and Mobile App Testing: With the proliferation of Internet of Things (IoT) devices and the increasing usage of mobile applications, the scope of web application testing will extend

to encompass these areas. Ensuring the security of IoT devices and the interfaces they use will become critical.

- Blockchain and Cryptocurrency Security: As blockchain technology and cryptocurrencies continue to gain traction, security testing will be imperative to safeguard transactions and digital assets. Ensuring the robustness of blockchain-based applications will be a specialized

niche in web application testing. Serverless Architectures: With the adoption of serverless computing, web application testing will need to adapt to assess the security of serverless functions and APIs. Ensuring that serverless components are free from vulnerabilities will be paramount.

- Container Security: As containerization technologies like Docker and Kubernetes become ubiquitous, web application testing will extend to cover container security. This includes ensuring that container images and orchestration configurations are free from vulnerabilities.

Stage 3: future scope of Aware Access Control

The successful deployment of ZPA lays the groundwork for a broader, more integrated security ecosystem. Future enhancements can be explored to further mature the organization's security capabilities:

- Integration with Zscaler Internet Access (ZIA): Converging Private Access (ZPA) with Internet Access (ZIA) to create a unified Secure Access Service Edge (SASE) platform, managing all web and private traffic through a single policy framework.
- Advanced Device Posture Checking: Moving beyond basic compliance to incorporate real-time risk scores from EDR tools, blocking access if a device shows signs of active compromise.
- Privileged Remote Access: Extending ZPA to third-party vendors and contractors with time-bound, audited access to specific critical systems, replacing jump boxes and bastion hosts.
- IoT/OT Segmentation: Applying the same micro-tunneling principles to Internet of Things (IoT) and Operational Technology (OT) devices to ensure they can only communicate with specific management servers.
- AI-Driven Anomaly Detection: Leveraging machine learning on ZPA logs to detect behavioral anomalies, such as a user accessing an application at an unusual hour or from a strange location, triggering automated MFA challenges or access revocation.
- Cloud Workload Protection: Utilizing ZPA to secure connections between containers and microservices in multi-cloud environments (AWS, Azure, GCP), ensuring east-west traffic between workloads is encrypted and authenticated.
- Automated Provisioning/Deprovisioning: Enhancing the identity provider integration to automatically revoke application access the moment an employee is terminated or changes roles, ensuring near real-time access control.
- Digital Experience Monitoring (DEM): Implementing DEM tools to proactively monitor the end-user experience, identifying network latency or application performance issues before they impact productivity.

Topics Explored:

1. Zscaler
2. Zero trust
3. ZPA
4. VPN
5. MISP
6. College Network Information
7. Deployment of SOC in the College

8. Threat Intelligence
9. Incident Response

Tools Explored:

- Nessus Scanner