



ABACUS

PENETRATION TESTING REPORT

CONFIDENTIAL // ABACUSTECHNOLOGY.COM

CLIENT



ASSESSMENT DATE

2026Q2

REPORT VERSION

1

CLASSIFICATION

CONFIDENTIAL

██████████ CES Customer Portal

2026Q2 / 1

Project Objective

Abacus was contracted by ██████████ to perform scoped web application testing between the dates of 5/19/2026, and 6/5/2026, on ██████████ web application.

Objectives of this Exercise:

- Replicate the vantage point of a malicious actor and identify high-risk vulnerabilities of ██████████ web application.
- Bring to the forefront the weakest areas of ██████████ security posture by exploiting the highest risk vulnerabilities and attack vectors found within the in-scope web application.
- Provide full due diligence information to ██████████ such as vulnerability details, risk mitigation recommendations, reconnaissance processes, exploitation methods, information system statistics, any compromised credentials, any acquired system configuration files, network diagrams, and more.

Abacus performed testing in accordance with the OWASP (Open Web Application Security Project) Web Security Testing guidelines. The findings in this report are limited to those that were within the scope of the Rules of Engagement. The penetration testing and this report are not exhaustive, as the security landscape is constantly evolving. Furthermore, resource restrictions and the need to avoid impacting business operations limit the testing that may be performed.

This report reflects the security position of ██████████ in-scope web application, as seen during the dates in which the testing was conducted. Future changes to any applications or infrastructure will change the security position of the environment from its current state. Additionally, as time passes, new types of attacks may arise which were previously unknown to the security industry. Considering this, Abacus recommends that all information systems undergo periodic security testing within a reasonable timeframe.

Table of Contents

Project Objective..... 2

Table of Contents..... 3

 Web Application Penetration Testing Methodology 4

 Risk Scoring 5

 Risk Results Key..... 6

Penetration Testing Summary 8

Web Application Penetration Testing Narrative..... 9

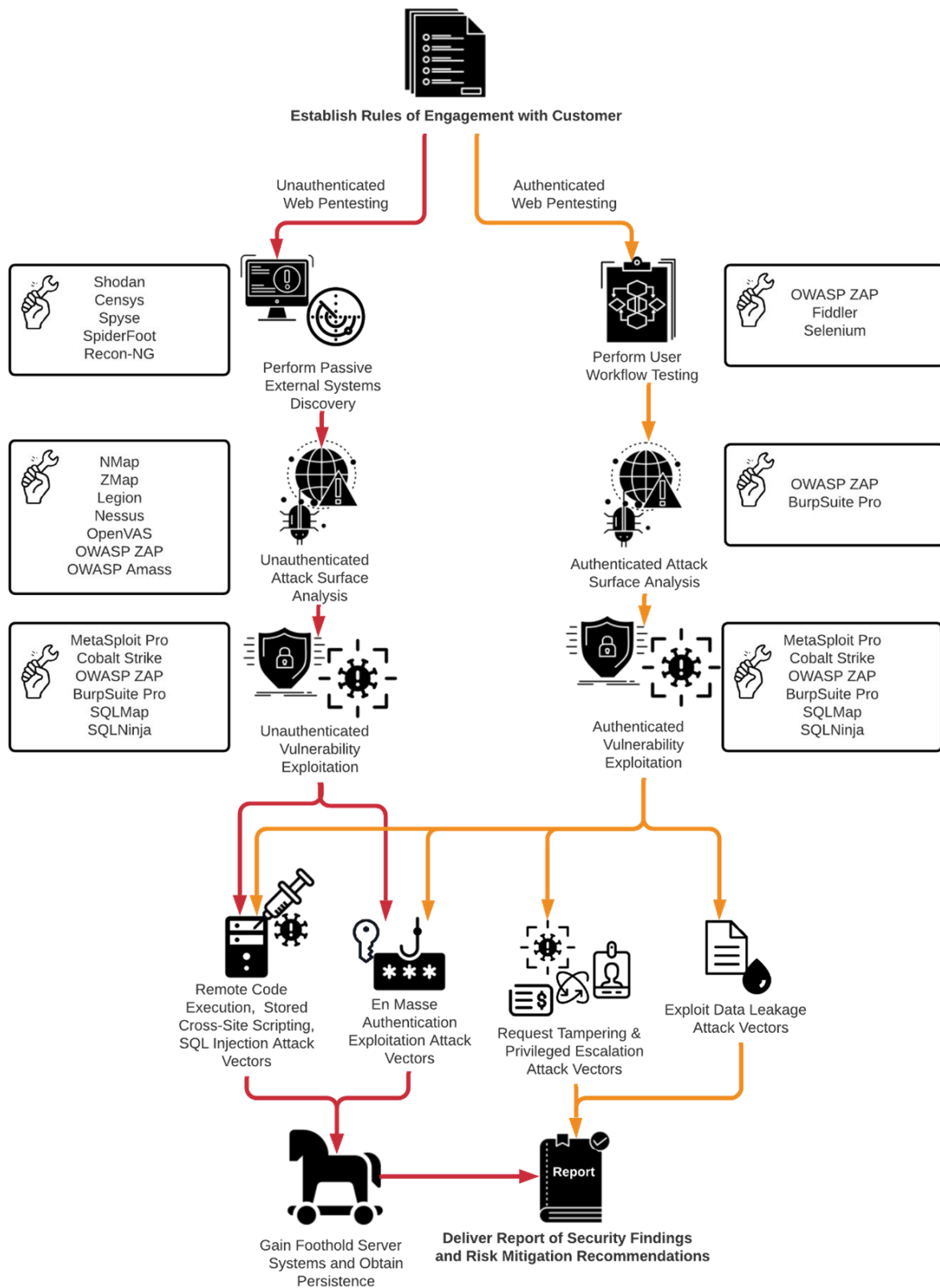
Risk Details 27

Conclusion 60

About the Security Testers:..... 60

Appendix A - OWASP Web Application Category Definitions..... 61

Web Application Penetration Testing Methodology



Risk Scoring

To present the findings in a manner easily digested and prioritized, Abacus uses the traditional aspects of Magnitude of Impact and Ease of Exploitation to determine the Level of Risk associated with individual findings identified in this security assessment.

Magnitude of Impact

The Magnitude of Impact rating is a measure of the damage each weakness could inflict on the organization and is based on access, data, etc., obtained through exploitation (i.e., level of access gained, types of information accessed, the strength of security measures bypassed)

Rating	Definition
HIGH	Exploitation could seriously affect system confidentiality, availability, integrity, or authentication. Resources to mitigate high risks should receive priority.
MODERATE	Exploitation could moderately affect system confidentiality, availability, integrity, or authentication.
LOW	Exploitation could minimally affect system confidentiality, availability, integrity, or authentication. The presence of multiple low impact findings could result in a finding with a higher impact rating.

Ease of Exploitation

The Ease of Exploitation rating measures the degree of difficulty for exploiting each finding. The higher the rating, the easier it is to exploit the finding and, therefore, the more likely it is to be exploited

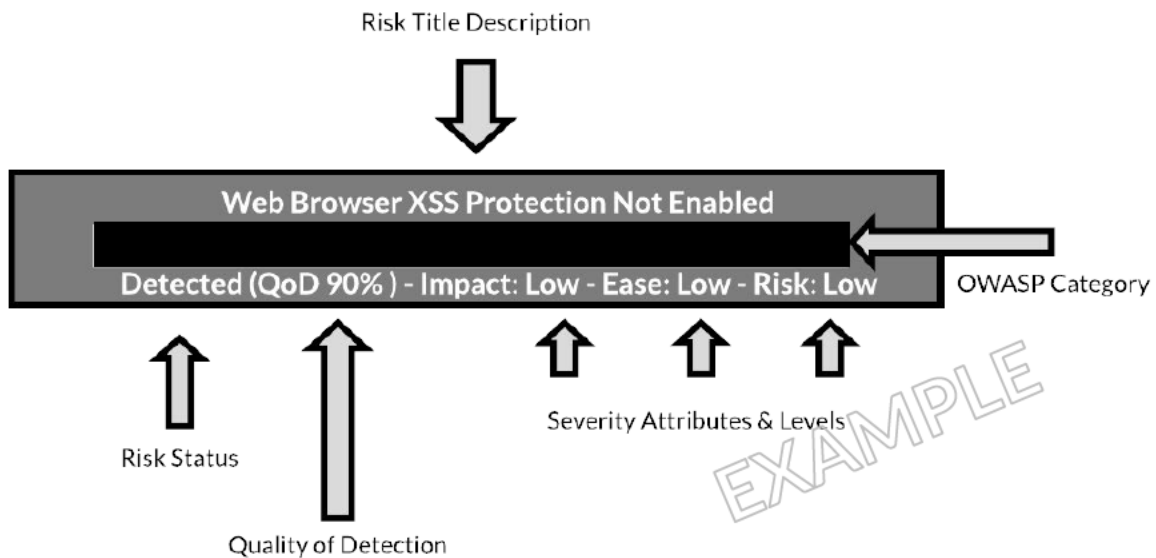
Rating	Definition
HIGH	Exploitation is easily accomplished using freely and readily available tools, techniques, and exploits.
MODERATE	Exploitation is moderately accomplished using a combination of freely available and custom tools, techniques, and exploits.
LOW	Exploitation is difficult to accomplish and requires a significant investment of time to develop custom tools, techniques, and exploits.

Level of Risk

		MAGNITUDE OF IMPACT		
EASE OF EXPLOITATION		HIGH	MODERATE	LOW
	HIGH	HIGH	MODERATE	MODERATE
	MODERATE	MODERATE	MODERATE	LOW
	LOW	MODERATE	LOW	LOW

Security Findings that are noteworthy but do not have an associated risk are labeled as Informational. Informational Security Findings are not inherently usable as attack vectors but may have associated operational excellence recommendations for implementing additional technical safeguards and controls.

Risk Results Key



- **Risk Title Bar**

- **Risk Title Description**
 - Contains a brief description of the pertinent risk.
- **MITRE ATT&CK Tactic**
 - Defines the MITRE ATT&CK Tactic category to which the risk is most closely associated to.
- **Risk Status**
 - Specifies if the risk was detected or exploited. In many situations, a high-severity risk is detected but not exploited due to the potential service impact to a production system and subsequent violation of the project Rules of Engagement (RoE).
 - In some cases of a service-impactful risk, a proof-of-concept exploit can be safely performed.
- **QoD (Quality of Detection)**
 - For "Detected" but not exploited risks, QoD is a value between 0% and 100% (i.e., confidence) that ranks the reliability of the executed network vulnerability test or fingerprinted information system.
- **Impact**
 - The Impact rating is a measure of the damage each weakness could inflict on the organization and is based on access, data, etc., obtained through exploitation.
- **Ease**
 - The Ease rating measures the degree of difficulty for exploiting each finding. The higher the rating, the easier it is to exploit the finding and, therefore, the more likely it is to be exploited.
- **Risk**
 - The Risk rating is based largely upon the potential negative impact to the pertinent environment combined with the ease of exploitation as illustrated in the matrix on the previous page.

Insecure SSL/TLS Protocol Versions and Cipher Suites

- Cryptographic Failures

Exploited (POC): **Yes** | Impact: Moderate | Ease: Moderate | Risk: Moderate

Summary:

The affected endpoints supported TLS v1.0 and TLS v1.1, which are out of date and considered a weak form of encryption. This may allow a malicious actor to sniff web server traffic and decrypt the content of an encrypted session, gaining access to potentially sensitive information.

Risk Detection Result(s):

Accepted	TLSv1.2	112 bits	Curve P-256 DHE 256
Accepted	TLSv1.2	112 bits	DHE 2048 bits
Accepted	TLSv1.2	128 bits	
Accepted	TLSv1.2	256 bits	
Accepted	TLSv1.2	128 bits	
Accepted	TLSv1.2	256 bits	
Accepted	TLSv1.2	128 bits	
Accepted	TLSv1.2	256 bits	
Accepted	TLSv1.2	112 bits	
Preferred	TLSv1.1	128 bits	Curve P-256 DHE 256
Accepted	TLSv1.1	256 bits	Curve P-256 DHE 256
Accepted	TLSv1.1	128 bits	DHE 2048 bits
Accepted	TLSv1.1	256 bits	DHE 2048 bits
Accepted	TLSv1.1	112 bits	Curve P-256 DHE 256
Accepted	TLSv1.1	112 bits	DHE 2048 bits
Accepted	TLSv1.1	128 bits	
Accepted	TLSv1.1	256 bits	
Accepted	TLSv1.1	112 bits	
Preferred	TLSv1.0	128 bits	Curve P-256 DHE 256
Accepted	TLSv1.0	256 bits	Curve P-256 DHE 256
Accepted	TLSv1.0	128 bits	DHE 2048 bits
Accepted	TLSv1.0	256 bits	DHE 2048 bits
Accepted	TLSv1.0	112 bits	Curve P-256 DHE 256
Accepted	TLSv1.0	112 bits	DHE 2048 bits
Accepted	TLSv1.0	128 bits	
Accepted	TLSv1.0	256 bits	
Accepted	TLSv1.0	112 bits	

Risk Mitigation Recommendation(s):

- Update TLS configurations to meet modern or intermediate configuration standards. This is the recommended configuration for most services that do not need compatibility with legacy clients, such as Windows XP or old versions of OpenSSL, as it is highly secure and compatible with nearly every client released in the last five (or more) years.

Risk Body

- Summary**
 - Contains a high-level overview of the risk description.
- Risk Detection Result(s)**
 - Demonstrates evidence that the risk exists, showcases exploitation if exploitation was performed.
- Risk Mitigation Recommendation(s)**
 - Gives recommendations on how to best mitigate this risk.
- Reference(s)**
 - Lists useful outside references that elaborate on the risk and/or provide additional mitigation recommendations.
- Affected Endpoint(s)**
 - Lists every endpoint detected as being affected by this risk.

Penetration Testing Summary



Organization	[Redacted]		
Point of Contact Name	Vulnerability Management Team	Security Testing Timeframe	May 19th, 2026 - June 5th, 2026
Point of Contact Phone	N/A	Point of Contact Email	[Redacted]

Total Count of Security Findings:

2	1	4	4
High	Moderate	Low	Informational

Summary of Risk Results	
Risk Title Description	Affected Endpoint(s)
Missing or Misconfigured Access Controls	[Redacted]
Exposed Personally Identifiable Information (PII)	[Redacted]
Session Cookies Are Not Invalidated Upon Logout	[Redacted]
Session Cookies Missing HttpOnly, Secure, and/or SameSite Attributes	[Redacted]
Excessive Externally Exposed Ports	[Redacted]
Misconfigured Content-Security-Policy Header	[Redacted]
Weak File Upload Protections	[Redacted]
Outdated JavaScript Libraries	[Redacted]
Missing Security Header: Referrer Policy	[Redacted]
Webserver Technology Exposed via Server Header	[Redacted]
Commented out Functionality Exposed	[Redacted]

Web Application Penetration Testing Narrative

The scope of this penetration testing engagement was specific to the Customer Portal, which was accessible via the following in-scope URL(s):

- [https://cesadministrationuat\[REDACTED\].com/](https://cesadministrationuat[REDACTED].com/)*

Abacus utilized the following provisioned virtual machines to conduct all penetration testing:

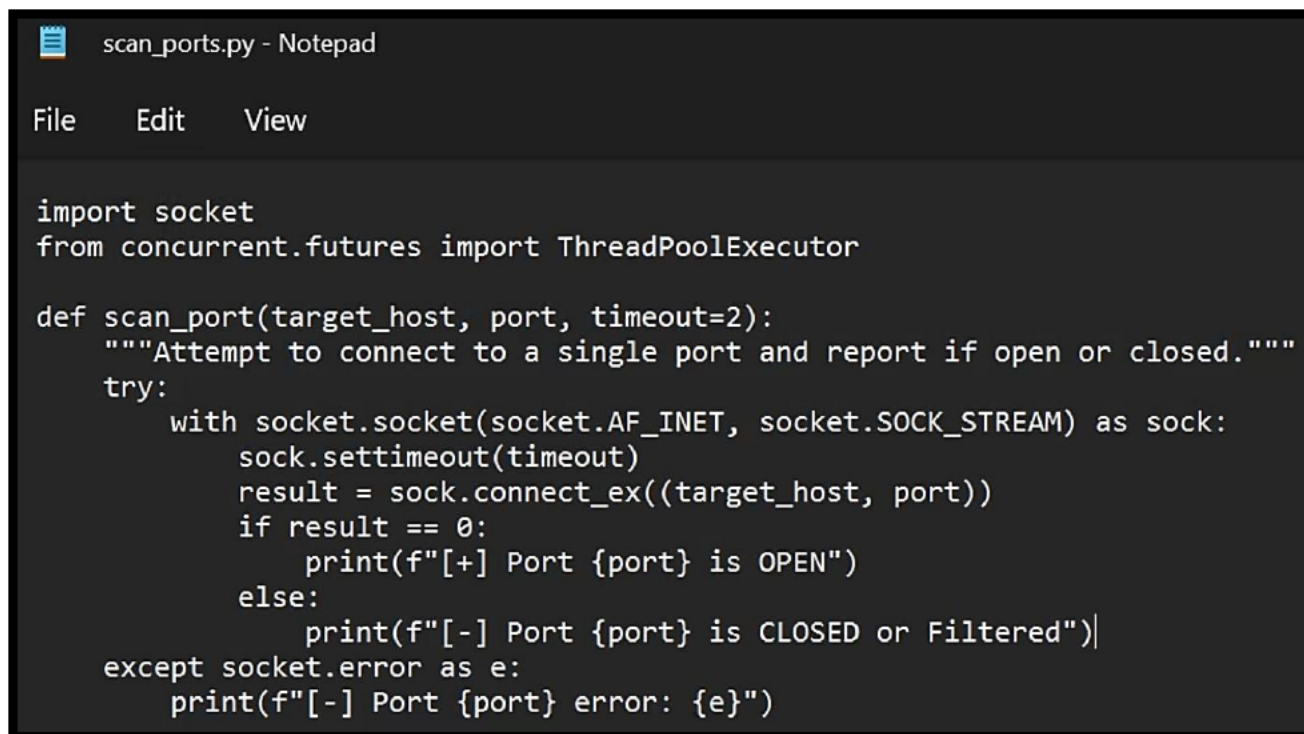
Machine Name	Source IP Address	MAC Address
[REDACTED]	[REDACTED]	[REDACTED]

[REDACTED] application team provided Abacus with an in-depth application walkthrough demonstrating the application's main functionality, business use cases, and general architecture before the penetration testing began. Additionally, the [REDACTED] team provided the following user account(s) to facilitate authenticated penetration testing:

Username	Role
[REDACTED]	Administrator

Abacus began the engagement by employing both passive and active reconnaissance to analyze the attack surface of the web application, APIs, and their supporting infrastructure. The purpose of these phases was to gain an understanding of key attack vectors prior to launching any active vulnerability scanning or attempted exploits against the in-scope systems.

The passive information gathering process involved utilizing open-source web services such as VirusTotal and DNS Twist to identify indicators of compromise (IoCs) and social engineering risks. In addition to this, an in-house port scanning tool was used to identify internet-exposed ports and services. There were no IoCs across the application and no flags for malicious activity. Abacus noted the application had the expected ports 80 and 443 open. However, port 25, used for Simple Mail Transfer Protocol (SMTP), was also open. While Abacus did not have the full context on the usage of this port, it is still recommended to review and close any unnecessary ports.



```
scan_ports.py - Notepad

File Edit View

import socket
from concurrent.futures import ThreadPoolExecutor

def scan_port(target_host, port, timeout=2):
    """Attempt to connect to a single port and report if open or closed."""
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.settimeout(timeout)
            result = sock.connect_ex((target_host, port))
            if result == 0:
                print(f"[+] Port {port} is OPEN")
            else:
                print(f"[-] Port {port} is CLOSED or Filtered")
    except socket.error as e:
        print(f"[-] Port {port} error: {e}")
```

Windows Notepad - In-house script used to check a web application for open ports


```
PS C:\OneDrive - python .\scan_ports.py
Scanning ports on
[+] Port 80 is OPEN
[+] Port 443 is OPEN
[+] Port 25 is OPEN
[-] Port 110 is CLOSED or Filtered
[-] Port 135 is CLOSED or Filtered
```

scan_ports.py - The web application had port 25 open

0

/ 92

Community Score

No security vendors flagged this URL as malicious

Reanalyze Search More

Status 200

Last Analysis Date a moment ago

VirusTotal - The web application had no flags present for malicious activity

dnstwist

phishing domain scanner

Scan

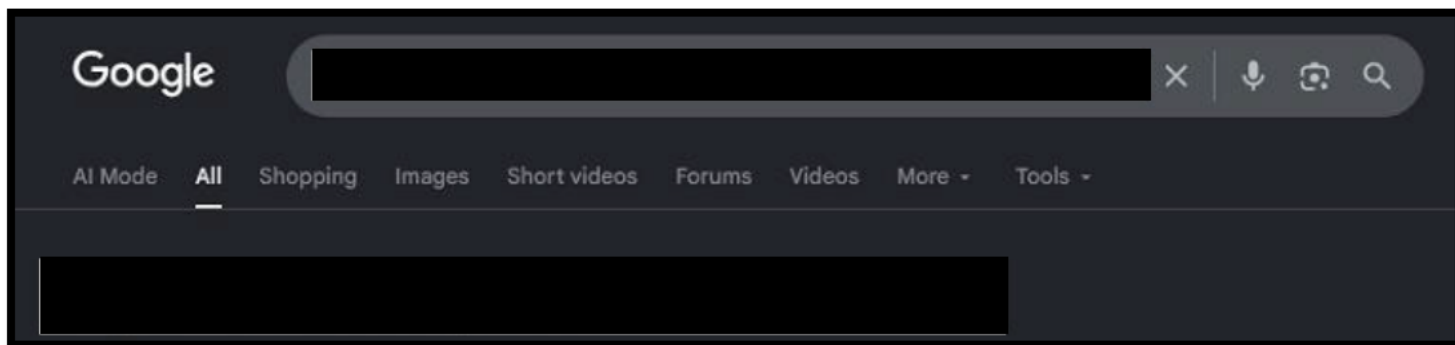
Scanned 5079 permutations. Found 37 registered: [share it](#) or download as [CSV](#) [JSON](#)

PERMUTATION	IP ADDRESS	NAME SERVER	MAIL SERVER

addition

DNS Twist - No active domain impersonations were identified

Google Dorking is the process of using advanced search operators to find sensitive information publicly available on the internet. By crafting specific search queries, Abacus attempted to locate exposed files, directories, passwords, and other sensitive data that may have been inadvertently indexed by search engines. Through this method, no sensitive files were uncovered.



Google Dorking - No sensitive documents or files were exposed on the internet

Abacus performed active reconnaissance with common tools such as SSLyze and WAFW00F to identify cryptographic issues, insecure TLS/SSL ciphers, and to check the presence of security controls like a web application firewall (WAF). The web application utilized secure TLS protocols and cipher suites. Additionally, a *BIG-IP AppSec Manager WAF* was identified in front of the web application.

```
* TLS 1.2 Cipher Suites:
  Attempted to connect using 156 cipher suites.

  The server accepted the following 3 cipher suites:
  [REDACTED]

  The group of cipher suites supported by the server has the following properties:
  [REDACTED] OK - Supported
  [REDACTED] OK - Not Supported

* TLS 1.3 Cipher Suites:
  Attempted to connect using 5 cipher suites.

  The server accepted the following 3 cipher suites:
  [REDACTED] (256 bits)
  [REDACTED] (256 bits)
  [REDACTED] (256 bits)
```

SSLyze - The application utilized secure cipher suites

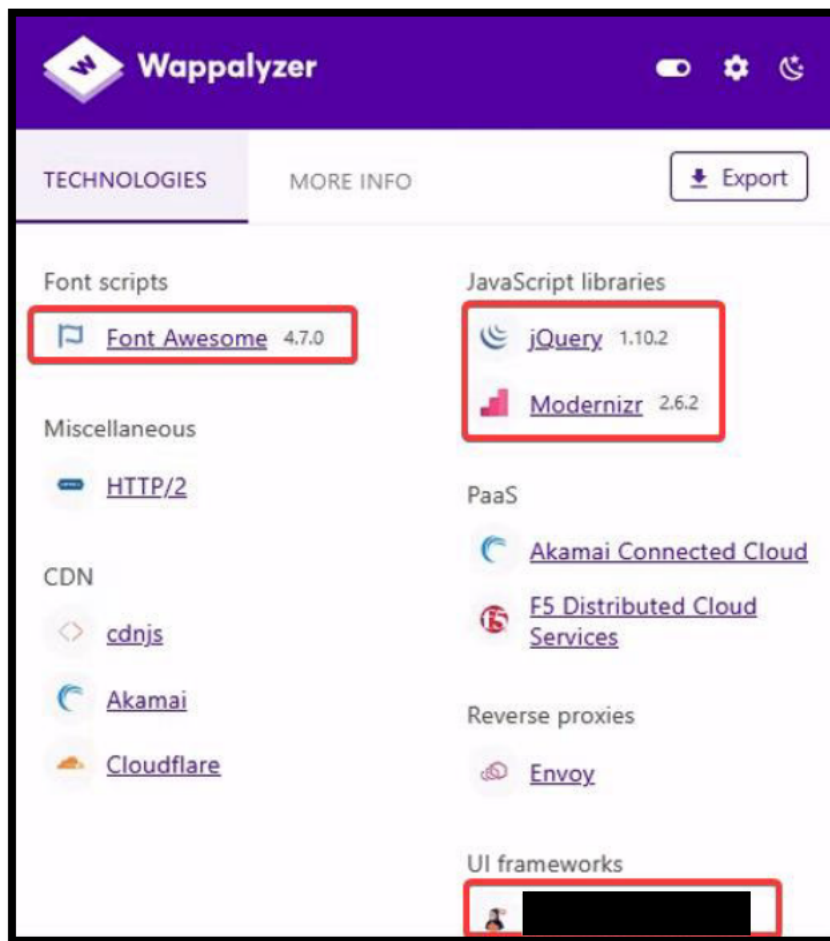
```
~ WAFW00F : v2.3.1 ~
The Web Application Firewall Fingerprinting Toolkit

[*] Checking [REDACTED].com
[+] The site [REDACTED].com is behind BIG-IP AppSec Manager (F5 Networks) WAF.
[~] Number of requests: 2
```

WAFW00F - The application was behind a BIG-IP AppSec Manager WAF

Abacus then stepped through the application to understand its functionality by utilizing a provisioned user account, [REDACTED]. This traffic was proxied through Burp Suite Professional, which is a commercial software that allows you to intercept, modify, and replay web requests. During this process, passive and active vulnerability scanners were aiding in the detection of sensitive information in source files, server response headers, and identifying third-party web technologies in use.

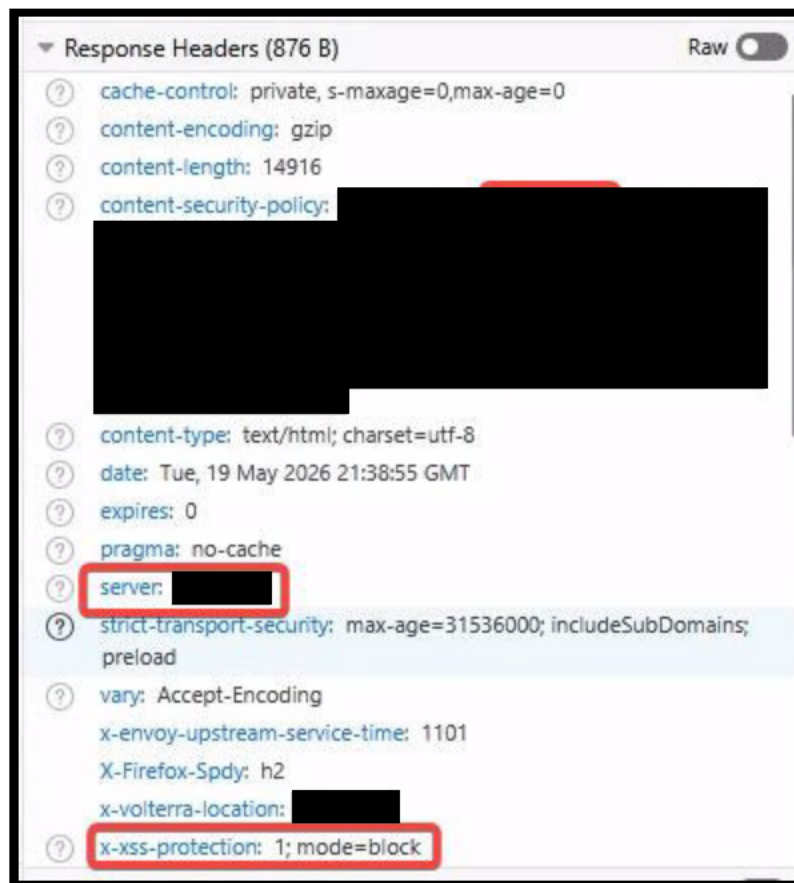
The Wappalyzer extension and Burp Suite's passive scanning helped fingerprint these technologies in use. It was found that outdated versions of Font Awesome, jQuery, Modernizr and Kendo UI were in use. While the versions implemented had no direct vulnerabilities at the time of this writing, it is still recommended to update the technologies to their respective latest and stable versions.



Wappalyzer – The application utilized outdated frameworks and libraries

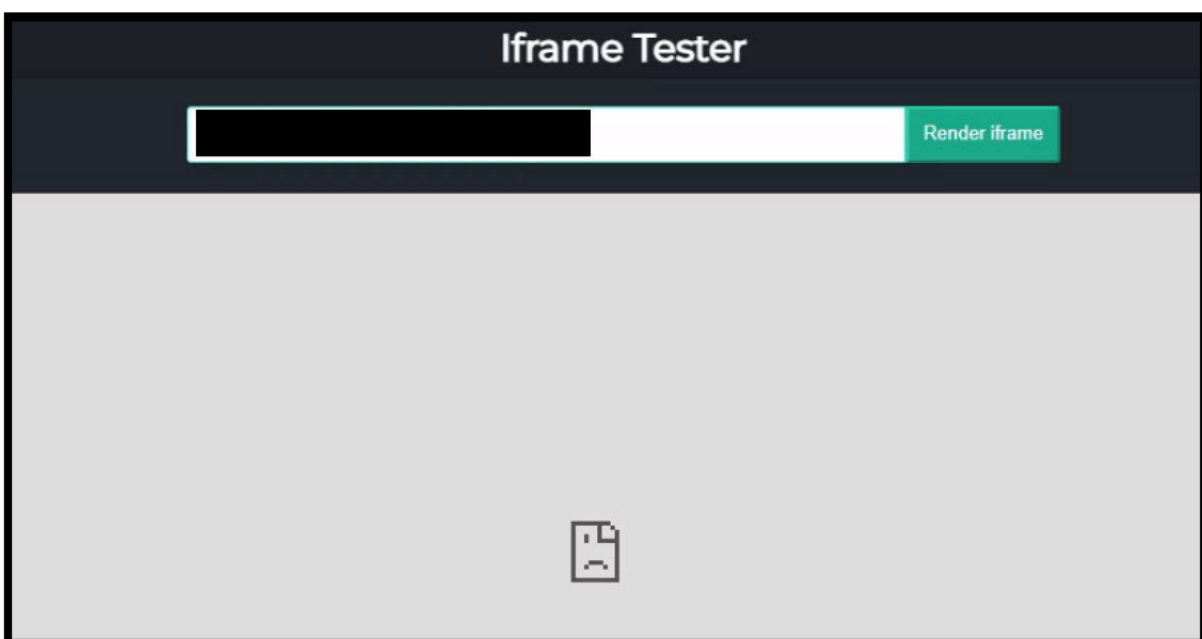
The application's response headers were analyzed to identify potential security misconfigurations, such as missing or improperly configured security headers, caching directives, or information disclosures. A Content-Security-Policy (CSP) header provides an explicit policy specifying acceptable origins from which various site components may be loaded. This adds an important layer of security that helps to protect against certain classes of attacks, such as Cross-Site Scripting (XSS), Clickjacking, and Sub-resource Integrity violations. In most cases, a well-defined CSP can completely take the place of older security headers such as "X-Frame-Options" and "X-XSS-Protection". The web application had a misconfigured CSP, implementing the 'unsafe-eval' directive without a nonce value, which could lead to the untrusted execution of JavaScript or style sheets. It is of note that the application also included 'X-XSS-Protection' in the response headers, which is considered deprecated in favor of a well-defined CSP.

Additionally, the referrer-policy header was missing, and server information was exposed. The referrer-policy header is designed to prevent cross-domain referrer leakage. The server header provides information to a malicious actor who may tailor attacks specific to those technologies. It is recommended to remove this header from all server responses to limit the available information provided to a malicious actor.



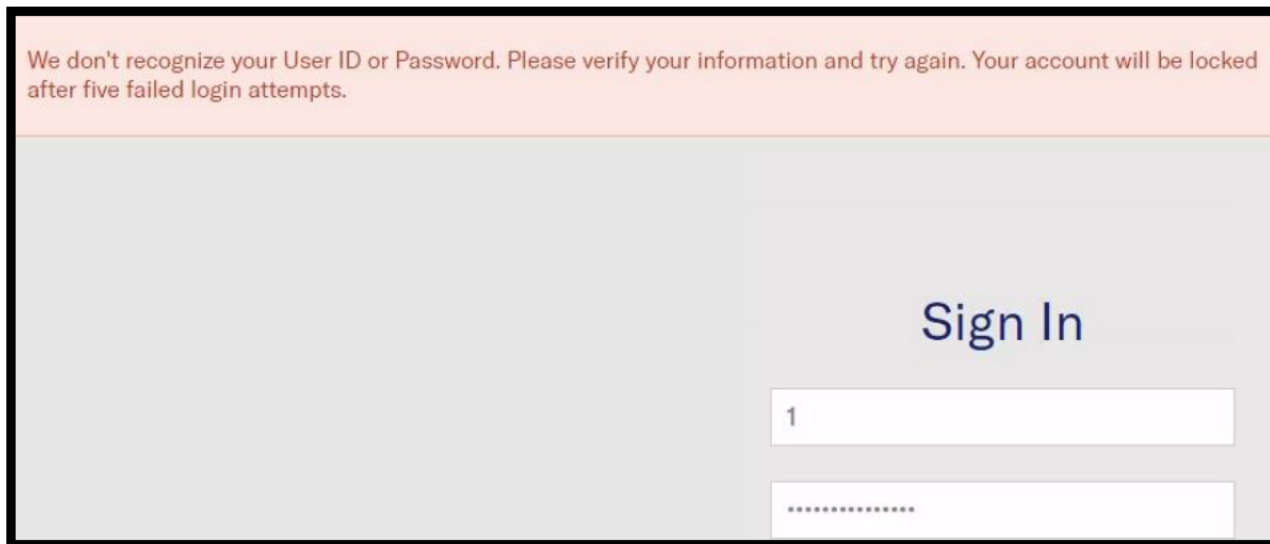
Dev Tools – The application had a misconfigured CSP header, missing a referrer header, and exposed server information

A missing or misconfigured CSP header could lead to attacks such as clickjacking using iFrames. Clickjacking is an attack that allows a malicious actor to embed a web page, into an iFrame. A malicious actor could leverage this along with social engineering techniques to trick users into entering credentials that would be stolen by the attacker or performing some other unintended action. To test for this vulnerability, Abacus used a tool called iFrame tester to demonstrate that the application did not allow iFrames to be embedded.



iFrame Tester – The application did not allow iFrame to be embedded

Abacus then attempted to actively test the application's authentication features. Authentication testing involves credential security, lockout mechanisms, password policies, and authentication bypasses. The application had a proper generic message in place when an invalid username or password was used.



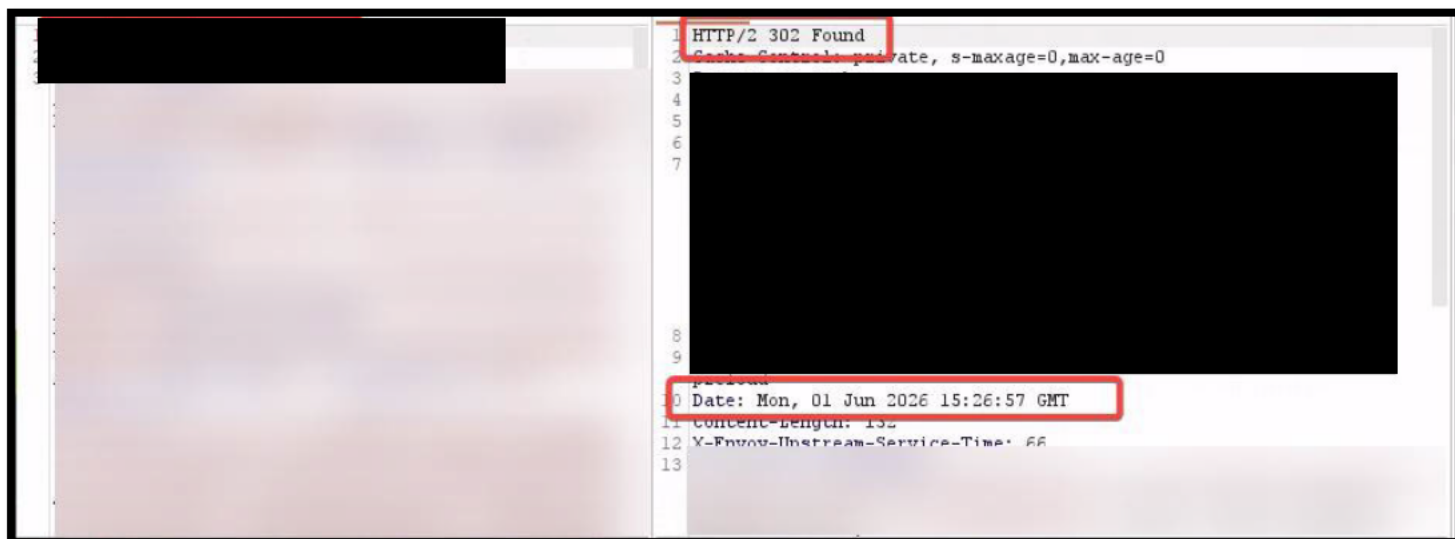
Customer Portal – The application gave a generic error message when an invalid username or password was used

Abacus then tested authorization and session management for the web application. Authorization testing identifies issues with how user permissions are enforced, which resources they can access, and which actions they can perform. Session management testing identifies issues with how user sessions are created and terminated, along with session cookie and token configurations. The application utilized cookies for authentication and session management.

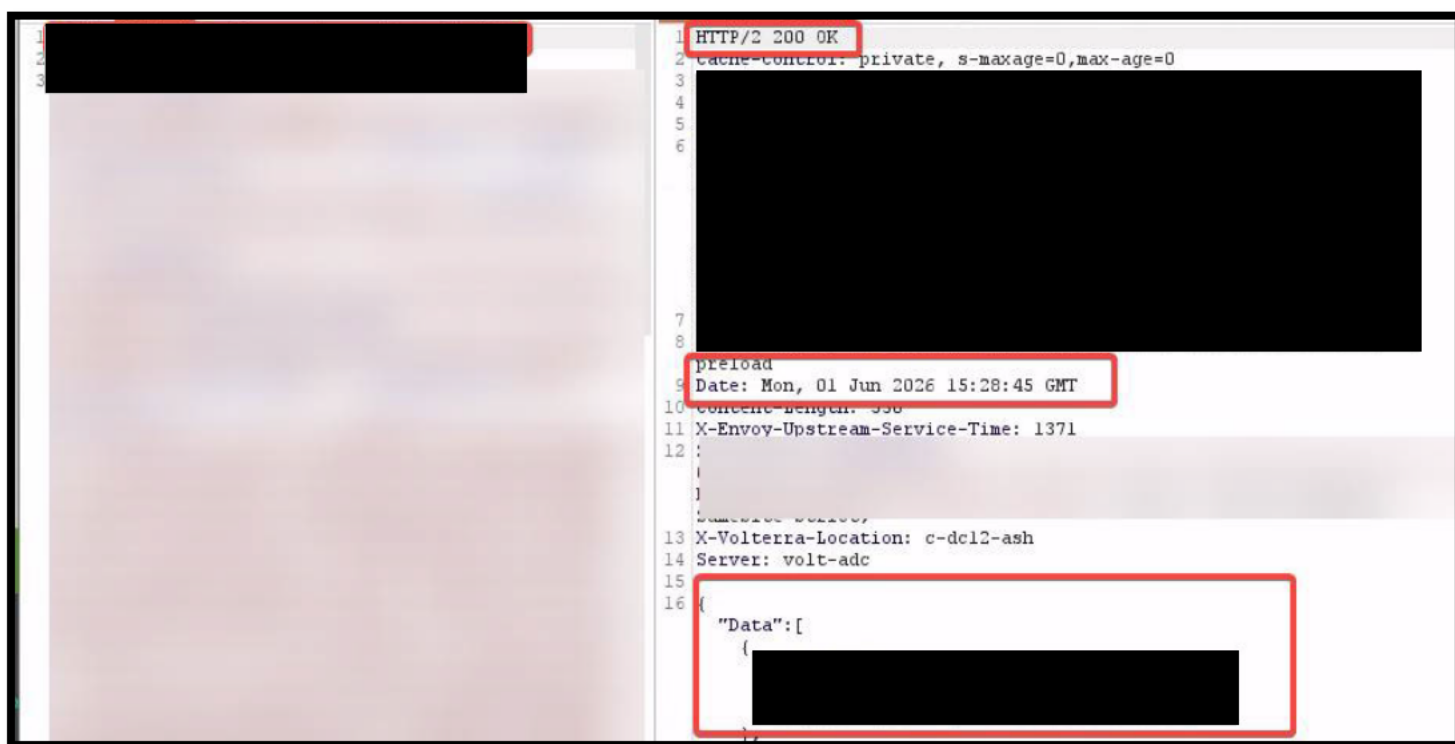
The logout functionality was tested for the proper invalidation of any session cookies or authorization tokens. The expected behavior is that any session cookies or tokens should be invalidated upon logout and should not be able to be replayed after logout. The web server did not properly invalidate the required cookie when utilizing the logout functionality. This was validated by logging out from the application and replaying a previous request. A malicious actor who has obtained access to a user's cookie would be able to perform Session Hijacking even after the user logs out from the application.



Burp Suite - Demonstrated a base request at 15:25:15 GMT



Burp Suite - Successful logout at 15:26:57 GMT

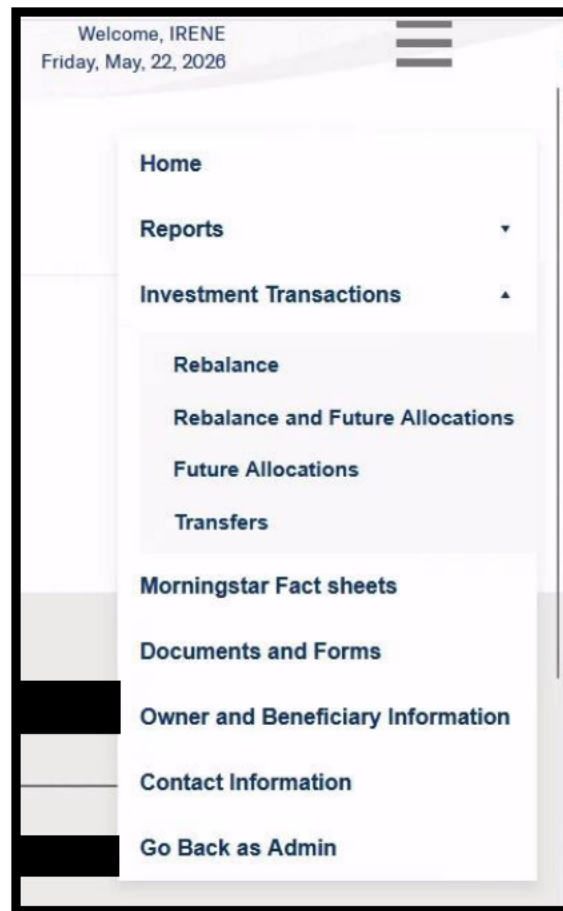


Burp Suite - GET request was successfully replayed at 15:28:45 GMT

Abacus tested authorization controls throughout the application to identify instances of broken access controls. This was done by both repeating requests while unauthenticated and using a lower-privileged user account's session cookies to determine if the application enforced proper access controls. This phase of testing included cross-role tampering to determine if an impersonated user could access the [REDACTED] user's admin pages. This was done by using the impersonate feature and force-browsing to admin-specific pages. Even with the attempt to modify an impersonated user's role to administrator, the application had proper access controls in place for pages such as Fund Notifications and Disruptive Trading. However, an impersonated user was able to access the Participant Policies page and make changes to a participant policy. All of these pages had a direct route from the admin menu, but an impersonated user did not have these options in their drop-down menu.



Customer Portal – Admin user had menu access to Fund Notifications, Disruptive Trading, and Participant Policies



Customer Portal – Impersonated User did not have menu access to Fund Notifications, Disruptive Trading, and Participant Policies



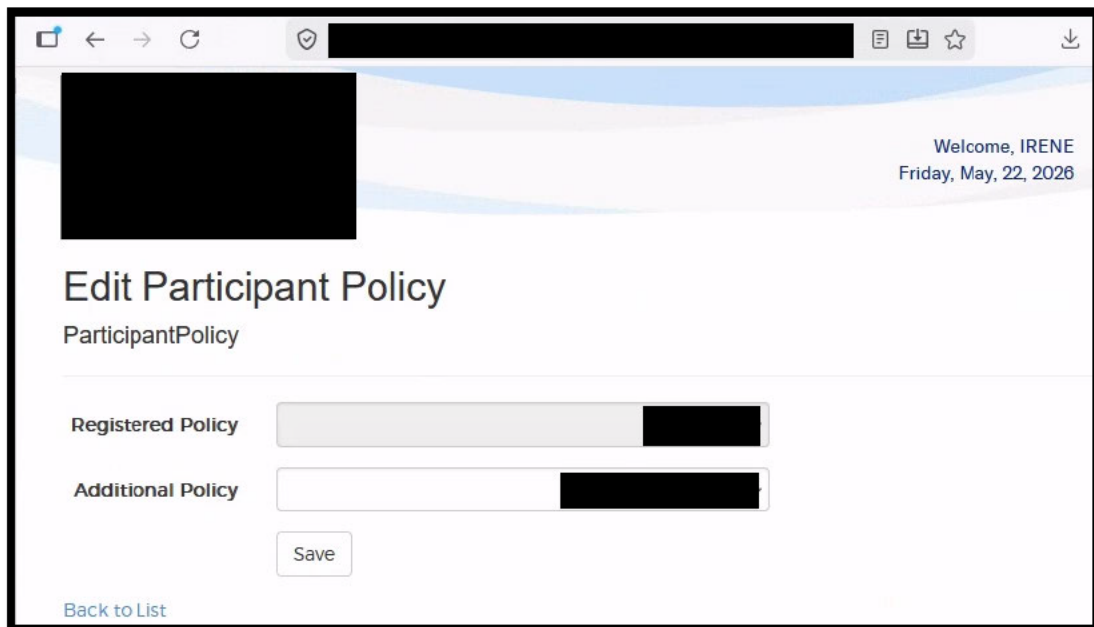
Burp Suite – Attempted to impersonate a user and modify the role to Administrator



Customer Portal – The impersonated user did not have access to CESANotifications



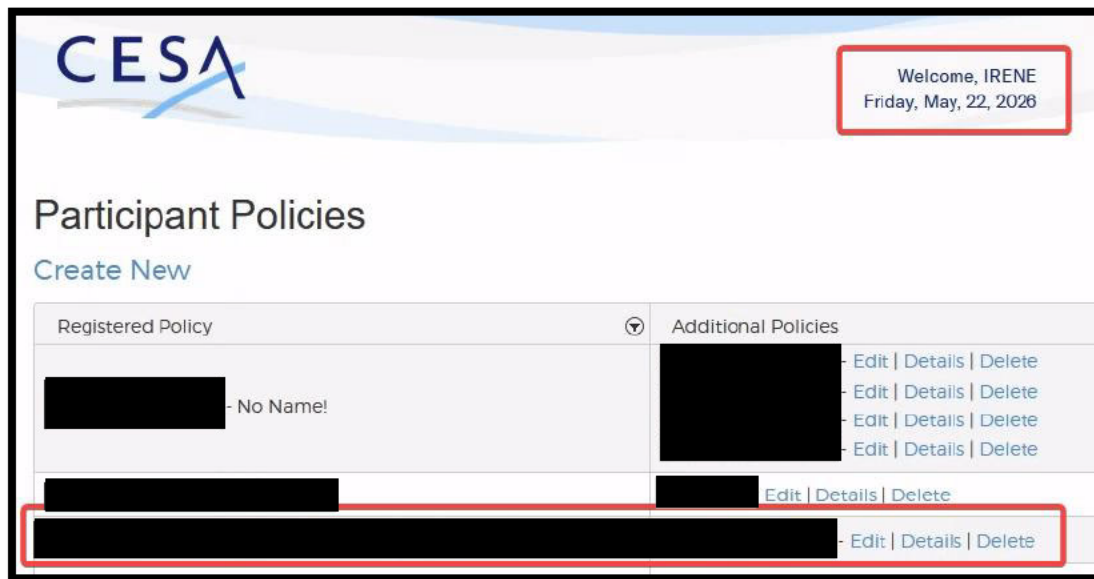
Customer Portal – The impersonated user did not have access to Disruptive Trading



Customer Portal – The impersonated user was able to force browse to ParticipantPolicies


```
11 Referer:
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19
20
16
17 <html>
18   <head>
19     <title>
20       Object moved
21     </title>
22   </head>
23   <body>
24     <h2>
25       Object moved to <a href="/ParticipantPolicies">
26         here
27       </a>
28     </h2>
29   </body>
30 </html>
```

Burp Suite – The impersonated user was able to modify a Participant Policy (1 of 2)



Customer Portal – The impersonated user was able to modify a Participant Policy (2 of 2)

Then, cookies were tested for proper access control if removed or tampered with. This method also identified which cookies were the true session cookies of the web application. True session cookies are the cookies that are required for a successful request. The application properly revoked access when cookies were removed from the request and the 'am-auth-jwt' and 'AspNet.ApplicationCookie' were identified as the true session cookies of the application.

```
1 GET [redacted]
2 Host: [redacted]
3 Cookie: StateCode=; CompanyUserRole=;
  WebTrendUserRole=; OptanonConsent=
  OptanonAlertBoxClosed=
  FPTType=; BrokerType=; TS015e47d9=
  USER_DISPLAYNAME=; OCP_ROLE=;
  firstName=; lastName=; TS01dc4fc6=
  TAL11CDg=
4 HTTP/2 200 OK
5 cache-control: private, s-maxage=0,max-age=0
6 Pragma: no-cache
7 Content-Type: text/html; charset=utf-8
8 Expires: 0
9 Vary: Accept-Encoding
10 X-Xss-Protection: 1; mode=block
11 Strict-Transport-Security: max-age=;
  includeSubDomains; preload
12 Date: Tue, 19 May 2026 21:45:15 GMT
13 Content-Length: 66055
```

Burp Suite – Demonstrated a base request with valid cookies


```
11 Referer: [REDACTED]
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19
20 RequestVerificationToken=
[REDACTED]
PolicyNumber=173451126&FirstName=TIMOTHY&LastName=
DECKER+<script>prompt(1)</script>&Email=
tdecke1352440a01.com&Phone=3098265599

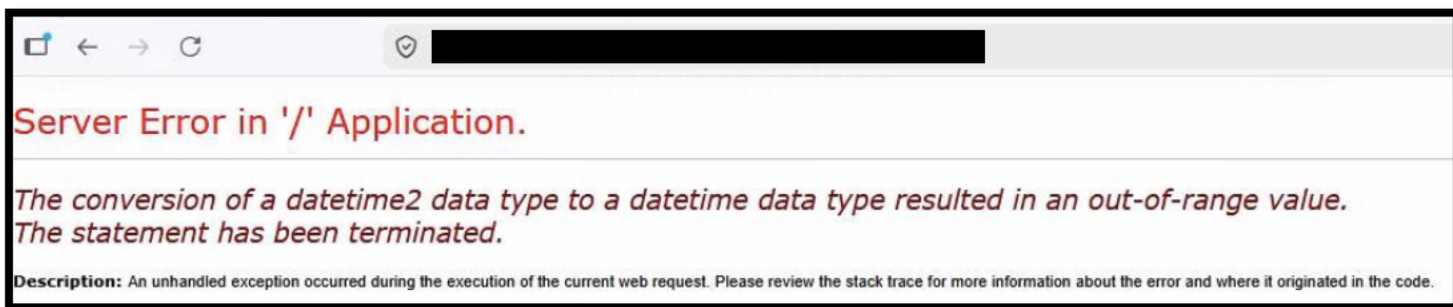
10 <html>
  <head>
    <title>
      Request Rejected
    </title>
  </head>
  <body>
    The requested URL was rejected. Please consult
    with your administrator.<br>
    <br>
    your support ID:
    [REDACTED] <br>
    <br>
    <a href='javascript:history.back();'>
      [Go Back]
    </a>
  </body>
</html>
```

Burp Suite – The application did not allow script tags

```
1 POST [REDACTED]
2 Host: [REDACTED]
3 Cookie: [REDACTED]

{
  "Id": [REDACTED],
  "Username": [REDACTED],
  "Roles": "Participant",
  "FirstName": [REDACTED],
  "LastName": [REDACTED] ((7*7))",
  "PolicyNumber": [REDACTED]
}
```

Burp Suite – The application accepted an SSTI payload but did not make any calculations



Server Error in '/' Application.

The conversion of a datetime2 data type to a datetime data type resulted in an out-of-range value. The statement has been terminated.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Customer Portal – The application had proper validations in place for date range values

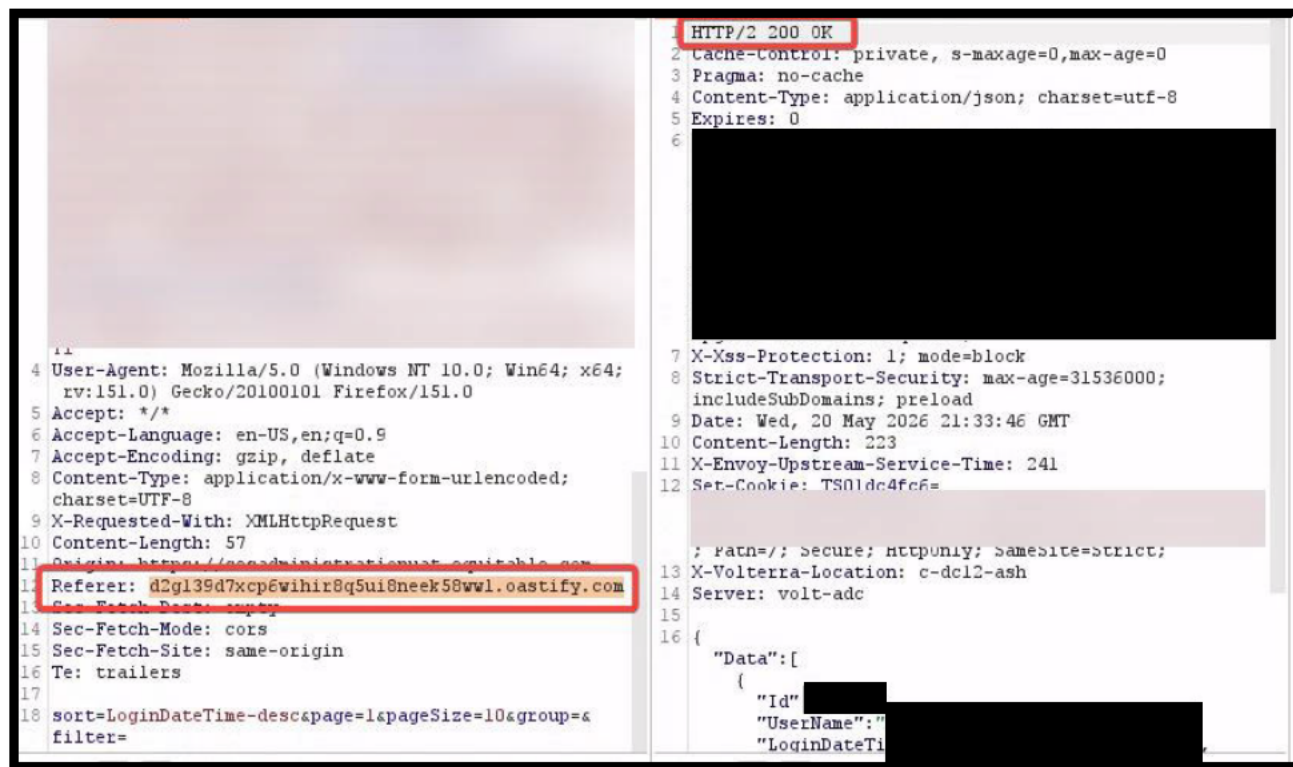
Documents and Forms

%%%

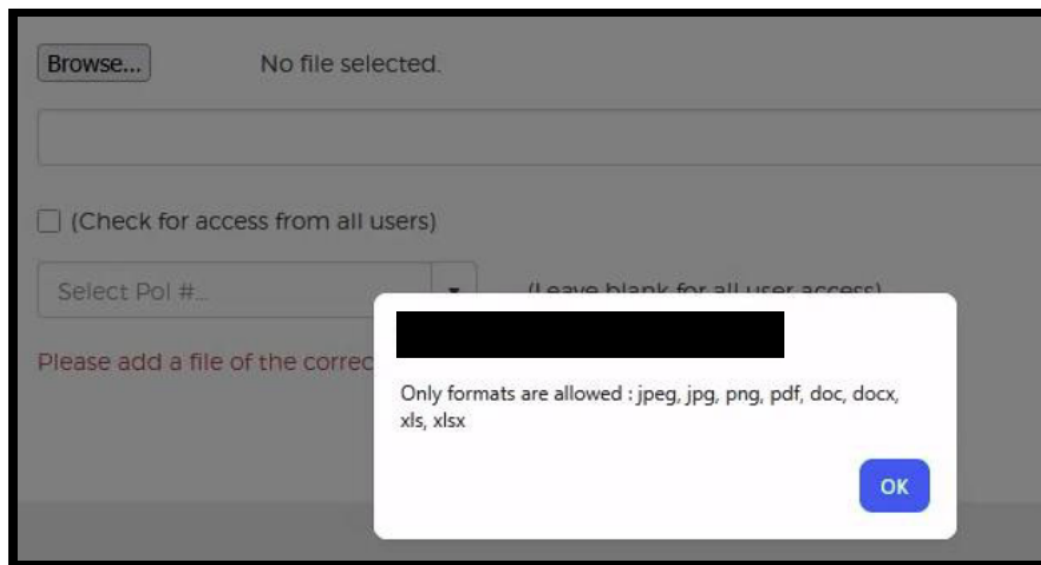
No matching documents

Customer Portal – The application did not process SQL Syntax

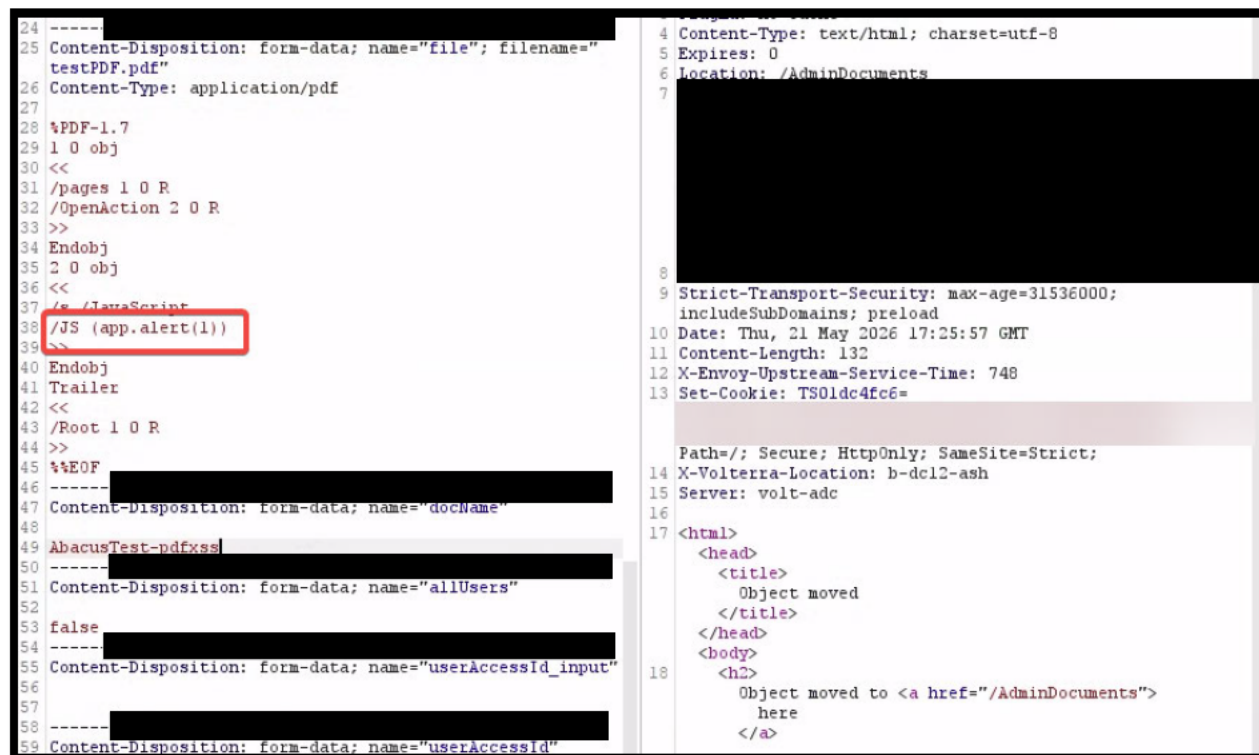
Server-Side Request Forgery (SSRF) vulnerabilities can occur whenever a web application fetches a remote resource without validating the user-supplied URL. It allows a malicious actor to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall or VPN. The application was not vulnerable to SSRF as there were no 'DNS' or 'HTTP' responses logged to the malicious collaborator URL.



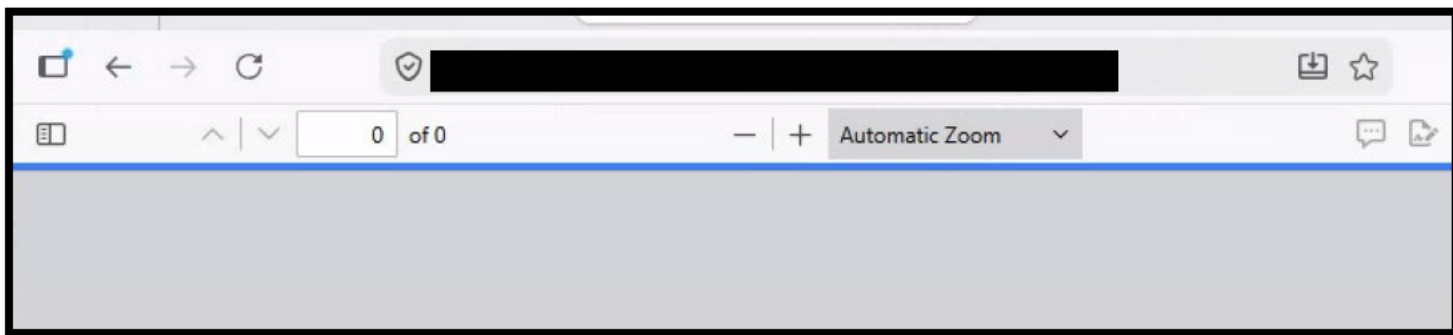
Specific intended workflows were also evaluated to determine if the implementations had any logical flaws. One example of this workflow testing was the file upload functionality in Documents. When an invalid file was used, the application output an alert showcasing the file formats it accepts. Because PDFs were allowed, Abacus attempted to upload a malicious PDF file containing an XSS attack. The file was accepted and could be accessed using the route given in details, but no alerts were triggered. It is of note that CSV files were not accepted and the application successfully blocked an attempt to upload a CSV using extension modification. Weak file upload protections were identified when it came to using a Canary Token, which is used to detect if the server will allow unauthorized access. After successfully uploading a PDF containing a canary token, and deleting the document, a notification was triggered almost a week later, showcasing that the file was accessed. Successful exploitation could result in remote file inclusion that may lead to cross-site scripting (XSS) attacks or malicious file uploads.



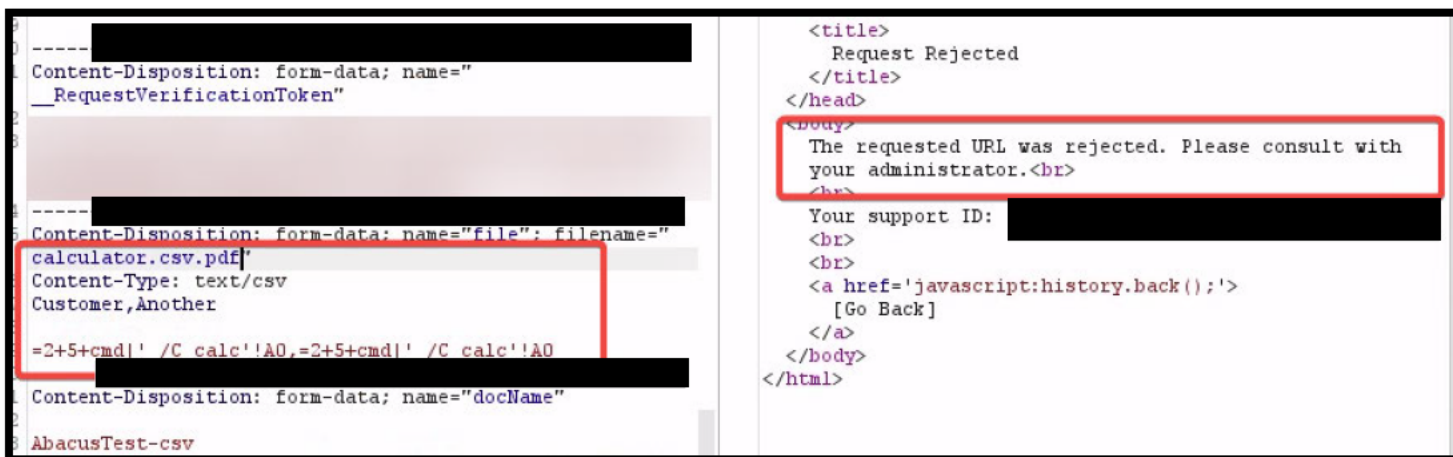
Customer Portal – The application revealed information on accepted file extensions



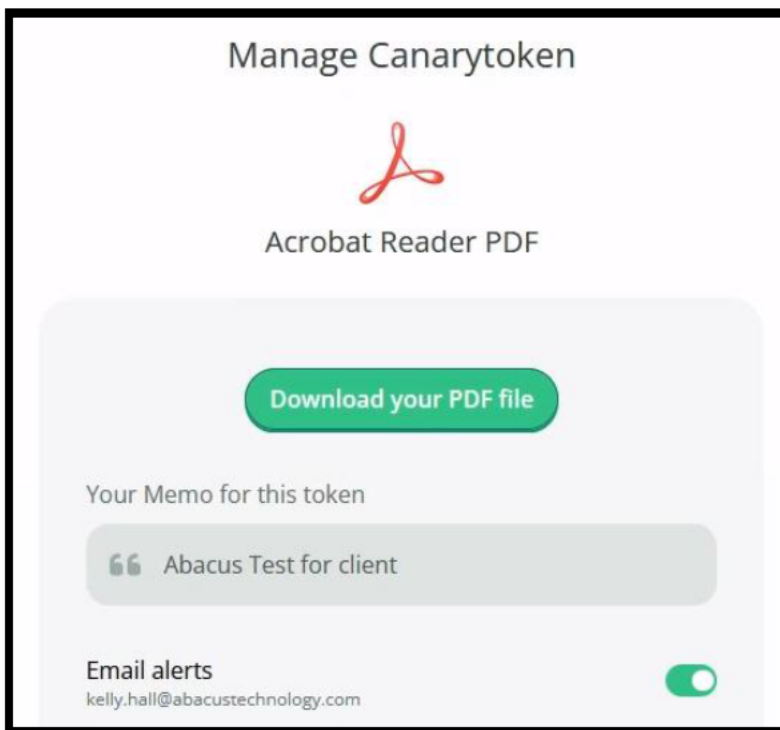
Burp Suite – The application accepted a PDF with an XSS attack



Customer Portal – Accessing the PDF did not trigger any alerts



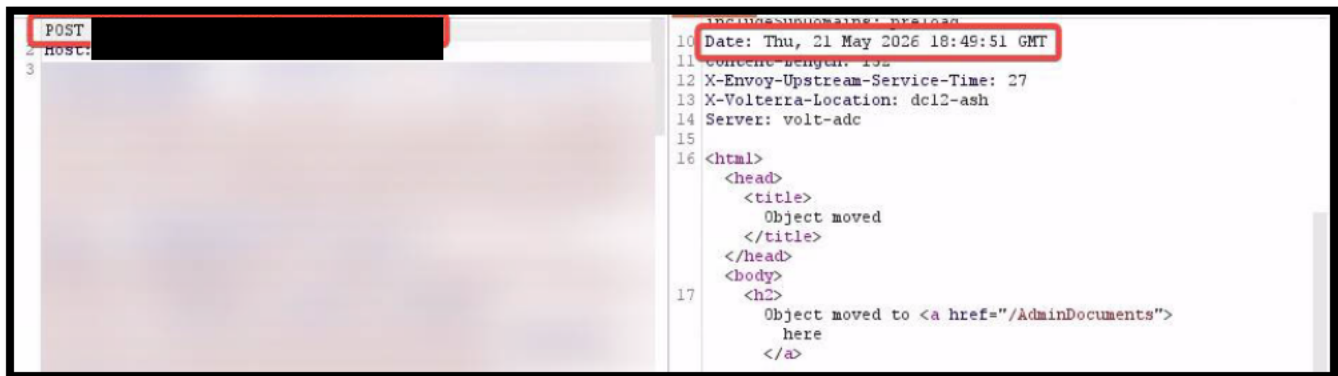
Burp Suite – The application did not allow invalid extensions to be uploaded



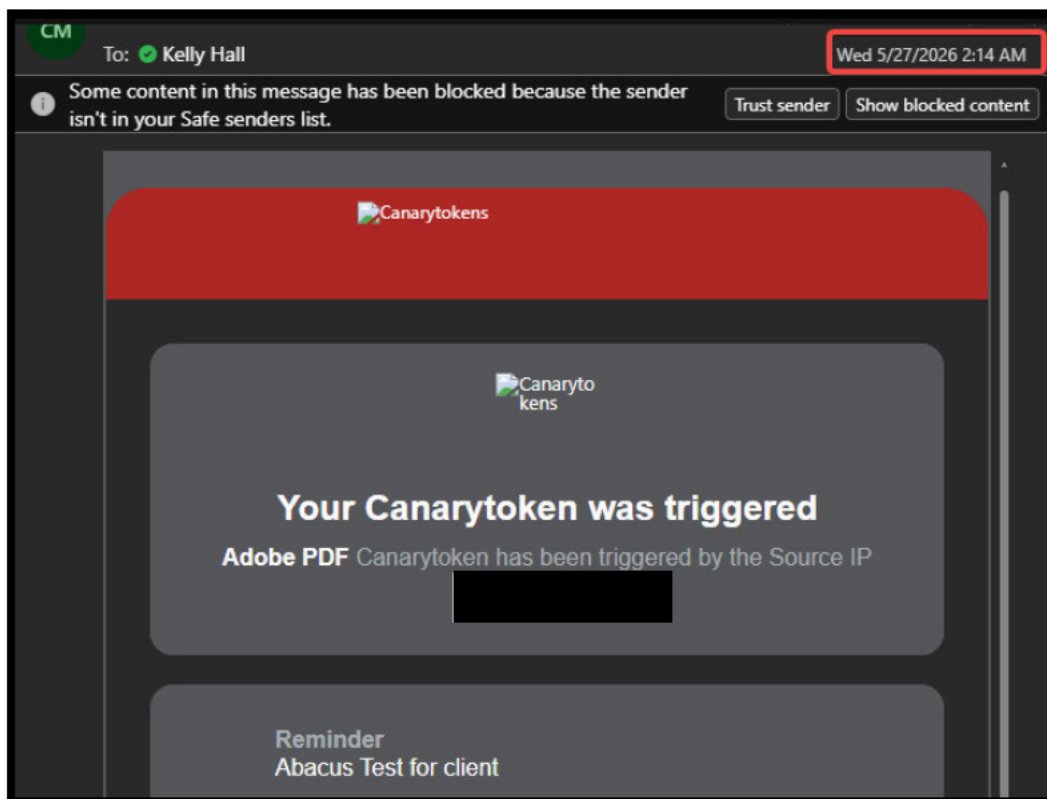
Canarytokens.org – A PDF file containing a canary token was created



Burp Suite – The application allowed the PDF file containing a canary token to be uploaded under the ID of 14324

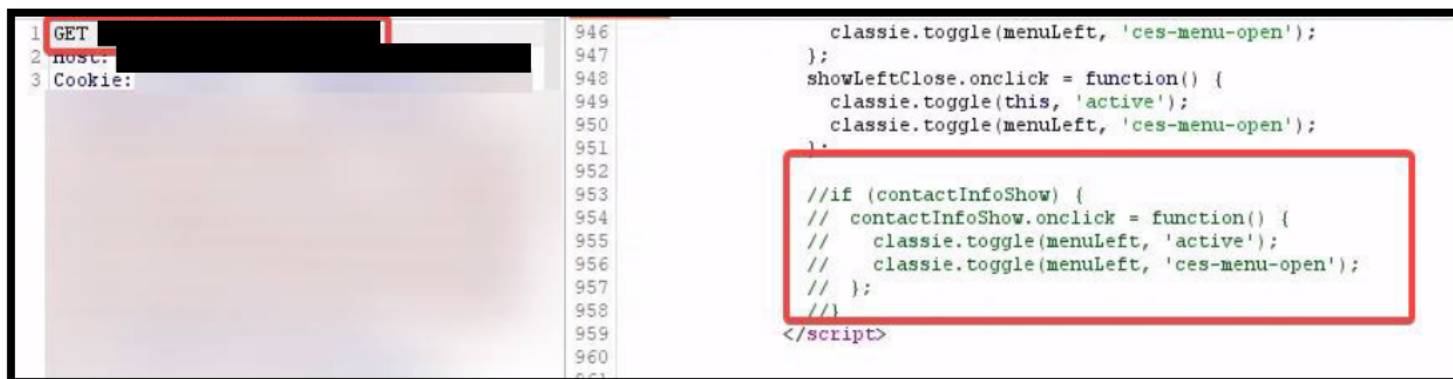


Burp Suite – The PDF file containing a canary token was deleted on May 21, 2026



Outlook – The canary token was triggered on May 27, 2026

Abacus continued to step through the workflow of the application, such as reviewing the source code for logging and debugging information, and reviewing responses for any exposed sensitive information. The source code of the dashboard page revealed unused functionality. Abacus attempted to intercept this request and un-comment the functionality, but was unable to find any notable changes in the application. While this was not exploited, it is still recommended to remove any unused code from the application as a malicious actor could access future and potentially sensitive functionality.

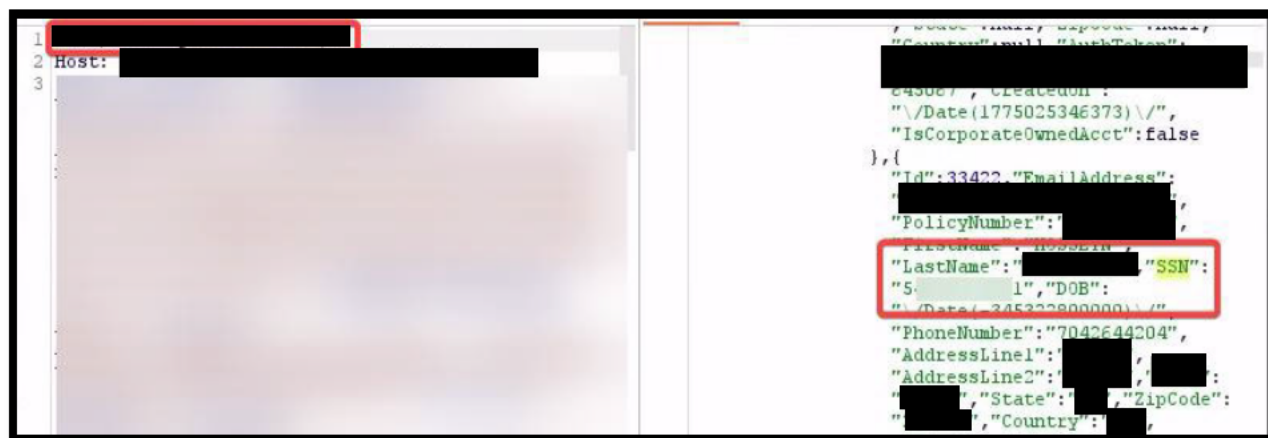


Burp Suite – The Dashboard page contained commented-out functionality

Additionally, sensitive information such as Personally Identifiable Information (PII) and Policy Numbers was found in plain text. The /PendingInvites endpoint retrieved user information, but not all of it was needed for the page. Information such as Social Security Numbers (SSNs) and addresses was found. Sensitive information that can identify an individual should be encrypted or removed if it is not necessary in the application. The Future Allocations page of an impersonated user showcased the last 4 digits of the policy number, masking the remaining numbers. However, on the Owner and Beneficiary Details page of the same impersonated user, the policy number is revealed in plain text. While this could be intended functionality, sensitive information should be masked throughout the application.

Email/Username	Pol #	First	Last	Invite Date
[redacted]	[redacted]	[redacted]	[redacted]	07/13/2021 - 12:11 PM
[redacted]	[redacted]	[redacted]	[redacted]	12/29/2020 - 2:30 PM
[redacted]	[redacted]	[redacted]	[redacted]	10/09/2020 - 11:4 PM
[redacted]	[redacted]	[redacted]	[redacted]	12/30/2019 - 4:57 PM

Customer Portal – The /PendingInvites page did not display SSN information



Burp Suite – The /PendingInvites request retrieved SSN and other user PII



Customer Portal – The Future Allocations page masked the policy number value



Customer Portal – The Owner and Beneficiary Details page output the policy number in plain text

Additional business logic testing was performed to identify if the application was susceptible to brute-force attacks that could impact the application. Without proper rate-limiting controls in place, a malicious actor may be able to repeatedly submit requests to an endpoint without slowing down their rate of connection. This could enable or encourage attacks such as enumeration or Denial-of-Service (DoS) attacks to gain access to additional information. For this test, the `/ParticipantPolicies/Edit/*` endpoint was tested, modifying the ID value with up to 20 requests at a rate of 10 requests per second. While the application did not block the brute-force attempt entirely, the requests were throttled, causing the attack of 20 requests to take 10 seconds instead of 2.

Request	Payload	Status	Time of day	Error	Timeout	Length	Comment
9	8	200	18:45:17 22 May 2026	☐	☐	199898	
10	9	200	18:45:18 22 May 2026	☐	☐	199896	
11	10	200	18:45:19 22 May 2026	☐	☐	199918	
12	11	200	18:45:20 22 May 2026	☐	☐	199898	
13	12	200	18:45:21 22 May 2026	☐	☐	199918	
14	13	200	18:45:22 22 May 2026	☐	☐	199918	
15	14	200	18:45:22 22 May 2026	☐	☐	199919	
16	15	200	18:45:23 22 May 2026	☐	☐	199919	
17	16	200	18:45:24 22 May 2026	☐	☐	199939	
18	17	200	18:45:25 22 May 2026	☐	☐	199899	
19	18	200	18:45:26 22 May 2026	☐	☐	199919	
20	19	200	18:45:27 22 May 2026	☐	☐	199899	

Request	Response
GET /	[REDACTED]
Host:	[REDACTED]

Burp Suite – The application throttled but did not block the brute-force attempt

Risk Details

Missing or Misconfigured Access Controls

A01:2025 - Broken Access Control

Detected: (QoD 100%) - Impact: High - Ease: High - Risk: High

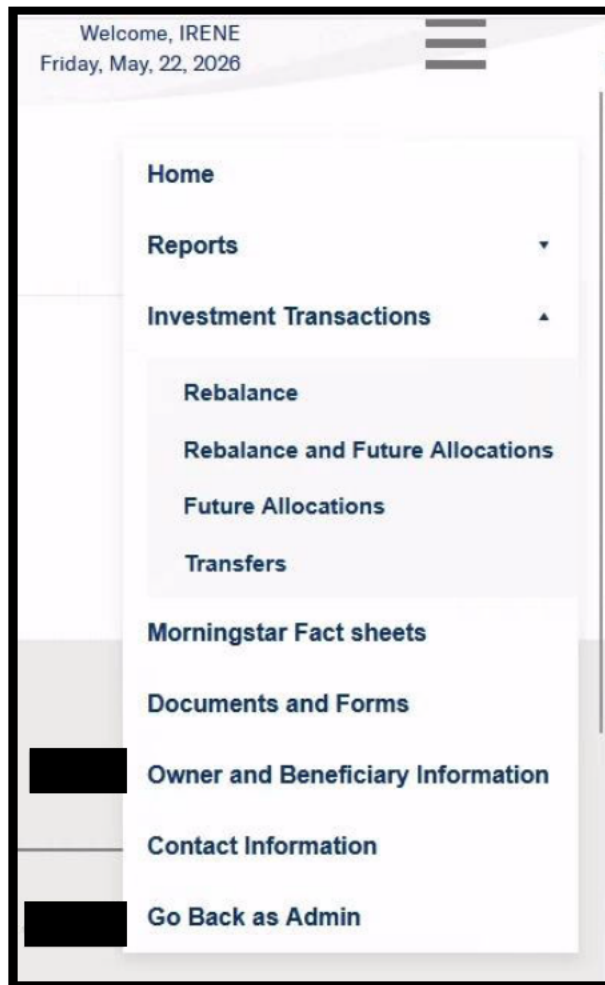
Summary:

The application's /Participant/Policies/Edit/* endpoint was accessible by the force browsing method. With access controls not properly in place, Abacus was able to impersonate a user, access the page, and modify a participant policy. Additionally, the Participant Policies are accessible by easily guessable Object Identifiers. Object identifiers are often unique IDs that are used to directly access a resource, such as a specific file or user account details. With low-entropy and easily guessable ID values, a malicious actor may attempt to brute-force or guess other object ID values that they should not have access to. If access controls are not properly enforced on these objects, this is known as an Insecure Direct Object Reference (IDOR). Because of this, a user can modify any participant policy by guessing the ID value.

Risk Detection Result(s):



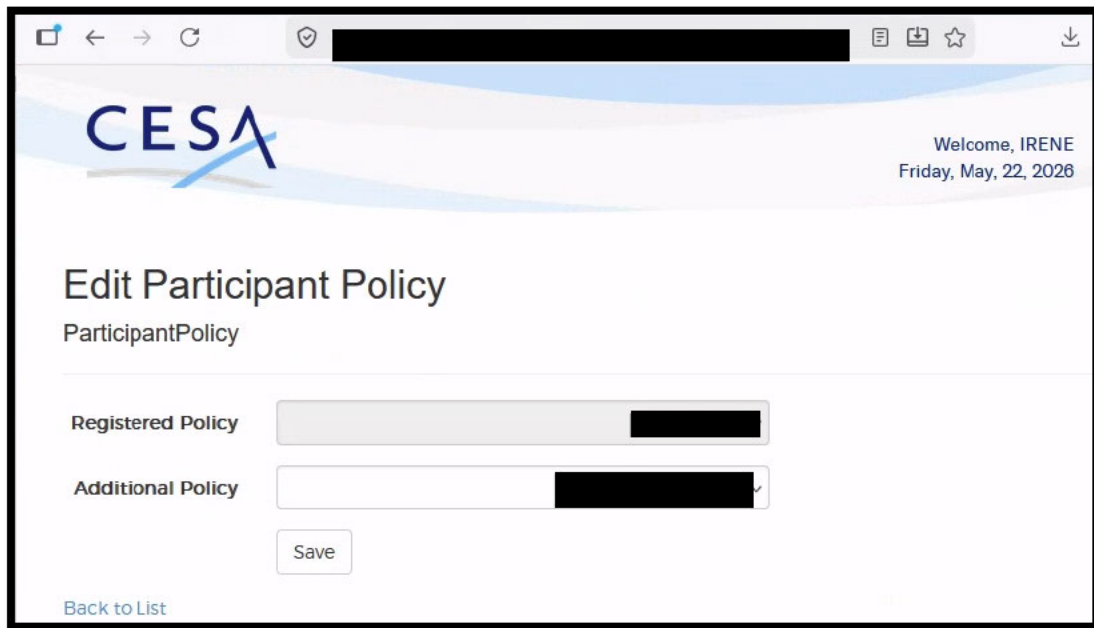
Customer Portal – Admin user had menu access to Pending Invites, Disruptive Trading, and Participant Policies



Customer Portal – Impersonated User did not have menu access to Pending Invites, Disruptive Trading, and Participant Policies



Burp Suite – Attempted to impersonate a user and modify the role to Administrator



Customer Portal – The impersonated user was able to force browse to ParticipantPolicies



Burp Suite – The impersonated user was able to modify a Participant Policy (1 of 2)

Risk Mitigation Recommendation(s):

- Implement access controls on the API endpoints to restrict low-privileged or unauthorized users from viewing or modifying information they should not have access to.
- Deny access to endpoints by default until the user's session cookies or tokens are checked for proper authorization to perform a delete operation.
- Implement session and identity management features into the application.
- Implement JSON web tokens (JWTs) or session cookies that are validated on the back end to be used for authorization throughout the available endpoints.
- Utilize session-specific API keys for authorization, authentication, and accounting purposes.

Reference(s):

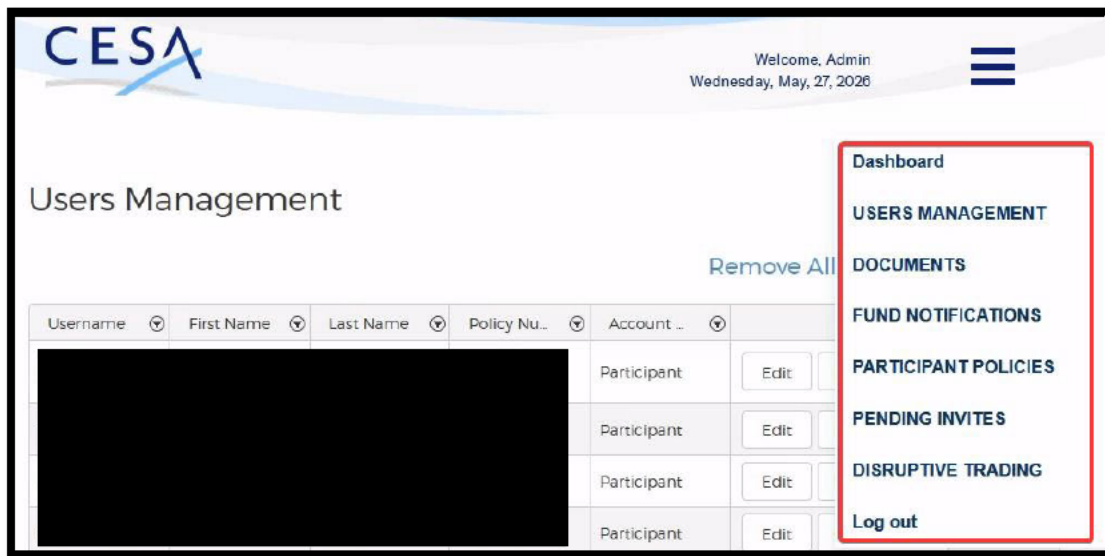
https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html#validate-the-permissions-on-every-request
https://owasp.org/Top10/A01_2021-Broken_Access_Control/
<https://owasp.org/API-Security/editions/2023/en/Oxa5-broken-function-level-authorization/>

Affected Endpoint(s):

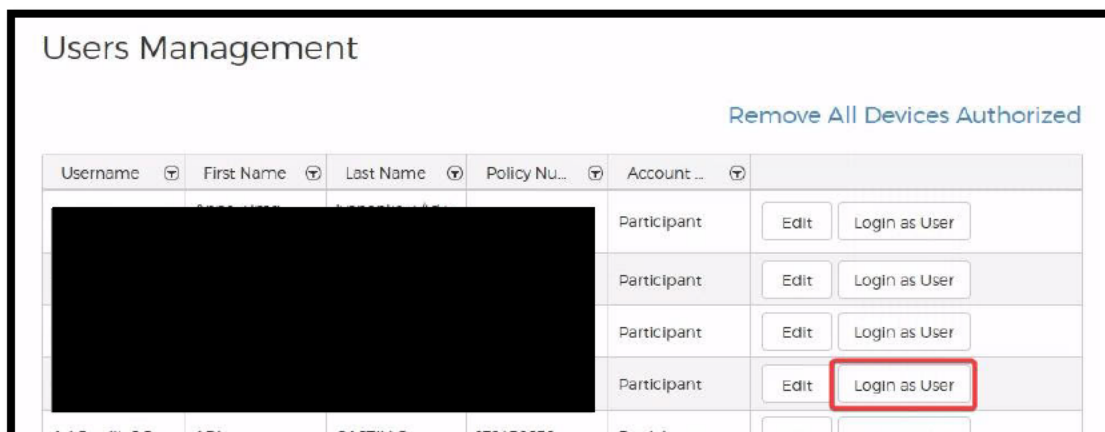
[Redacted]

Risk Replication Steps:

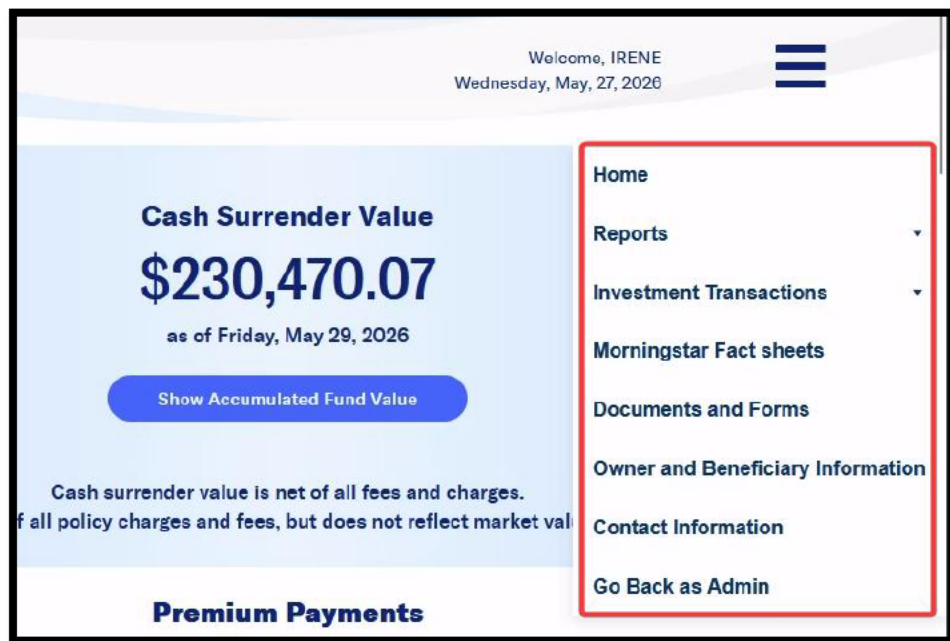
1. Review the pages and functionality admin has access to



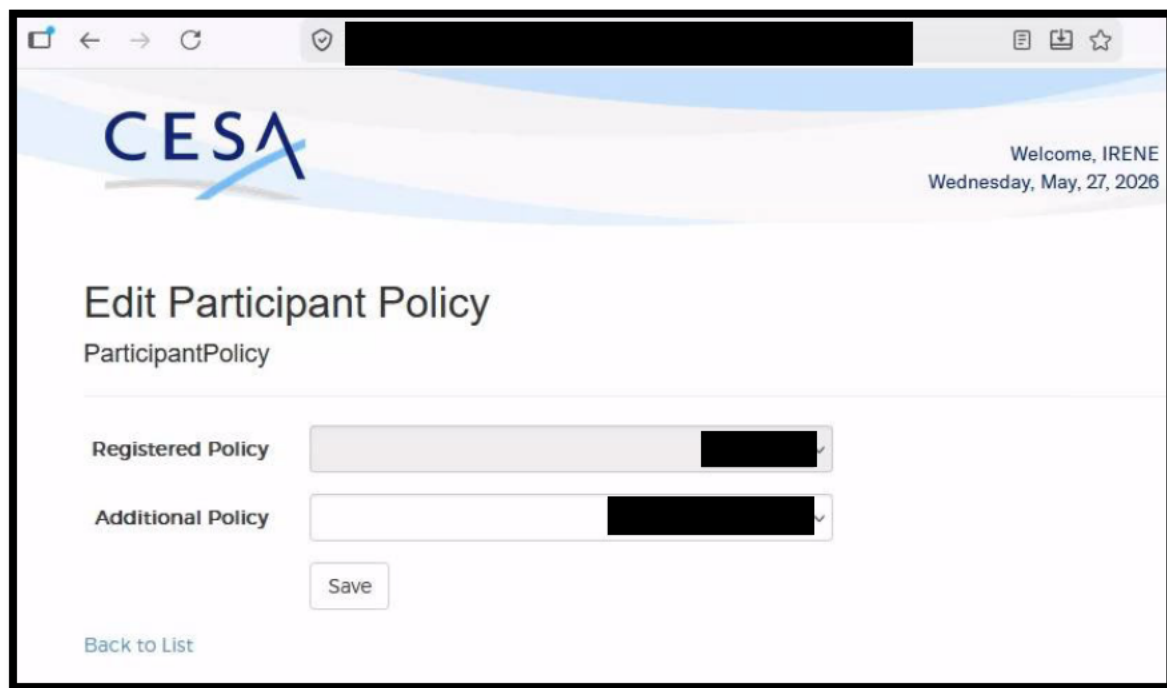
2. Under Users Management, impersonate a user



- Review what the impersonated user has access to



- Access an Admin page while impersonating a user

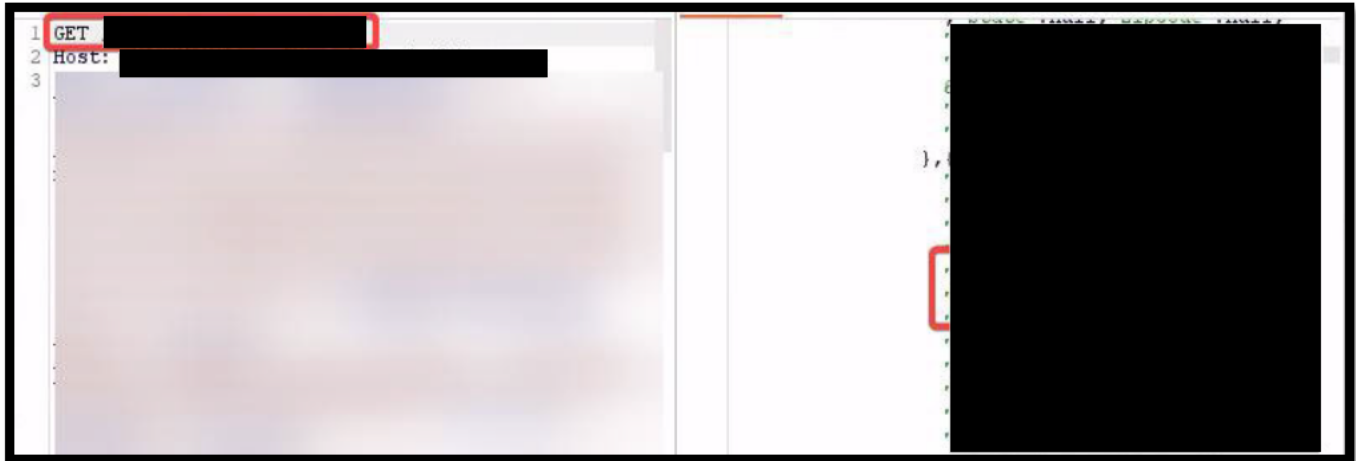


Exposed Personally Identifiable Information (PII)
A04:2025 - Cryptographic Failures
Detected: (QoD 100%) - Impact: High - Ease: High - Risk: High

Summary:

The `/PendingInvites` endpoint exposed Social Security Numbers (SSN) and other sensitive personally Identifiable Information (PII) in plain text. The `PendingInvites` page had no need for SSN but retrieved from the server all information on each participant, including SSNs in plain text. Additionally, on the `Future Allocations` page, it was found that the policy number was masked but on another page, `Owner and Beneficiary Details`, the policy number was displayed in plain text.

Risk Detection Result(s):



Burp Suite - The `/PendingInvites` request retrieved SSN and other user PII



Customer Portal - The `Future Allocations` page masked the policy number value



Customer Portal - The `Owner and Beneficiary Details` page output the policy number in plain text

Risk Mitigation Recommendation(s):

- Obfuscate or encrypt SSNs and other sensitive PII that are passed in a request and response.

Reference(s):

https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html
<https://owasp.org/API-Security/editions/2019/en/0xa3-excessive-data-exposure/>

Affected Endpoint(s):



Risk Replication Steps:

1. In Burp Suite's HTTP history, navigate to a /PendingInvites request and review the user information in the response

The screenshot displays the Burp Suite interface. At the top, the 'Proxy' tab is selected, and 'HTTP history' is highlighted. A message states 'Logging of out-of-scope Proxy traffic is disabled' with a 'Re-enable' button. Below this, a filter is applied: 'Filter: Hiding out of scope items; hiding CSS, image and general binary content; matching expression SSN'. The HTTP history table shows several requests, with the one at index 314 (GET /PendingInvites) highlighted in orange. Below the table, the 'Request' and 'Response' panels are visible. The 'Request' panel shows a GET request to a redacted host. The 'Response' panel shows a JSON response with a red box highlighting a section containing sensitive information, including a field labeled 'SSN'.

#	Host	Met...	URL	Params	Edited	Status	Length	MIME
1		GET	/			200	67067	HTML
104		GET	/			200	66891	HTML
105		GET	/			200	114069	script
116		GET	/		✓	200	2086103	script
121		GET	/		✓	200	114069	script
122		GET	/		✓	200	2086103	script
314		GET	/PendingInvites			200	638838	HTML
350		GET	/			200	66870	HTML
411		GET	/			200	66893	HTML

Request

```
1 GET /PendingInvites HTTP/1.1
2 Host: [REDACTED]
3
```

Response

```
{
  "IsCorporateOwnedAcct": false,
  "SSN": "[REDACTED]",
  "Email": "[REDACTED]",
  "Phone": "[REDACTED]",
  "Address": "[REDACTED]",
  "City": "[REDACTED]",
  "State": "[REDACTED]",
  "Zip": "[REDACTED]",
  "Country": "[REDACTED]",
  "IsVerified": true,
  "IsActive": true,
  "IsDeleted": false
}
```


2. While impersonating a user, navigate to Future Allocations to review the masked value



Welcome, IRENE
Wednesday, May, 27, 2026



XXXXX1137

Rebalance and Future Allocations

The Rebalance is not an option due to your policy being in an MSO segment or having a loan.

Please use the Transfer Option to make changes.

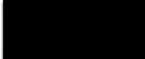
3. While impersonating a user, navigate to Owner and Beneficiary Details to review the policy number unmasked



Welcome, IRENE
Wednesday, May, 27, 2026



Owner and Beneficiary Details

Insured Name and Policy Number	Owner	Primary Beneficiary	Relationship	Contingent Beneficiary	Relationship
			Husband		

Session Cookies Are Not Invalidated Upon Logout

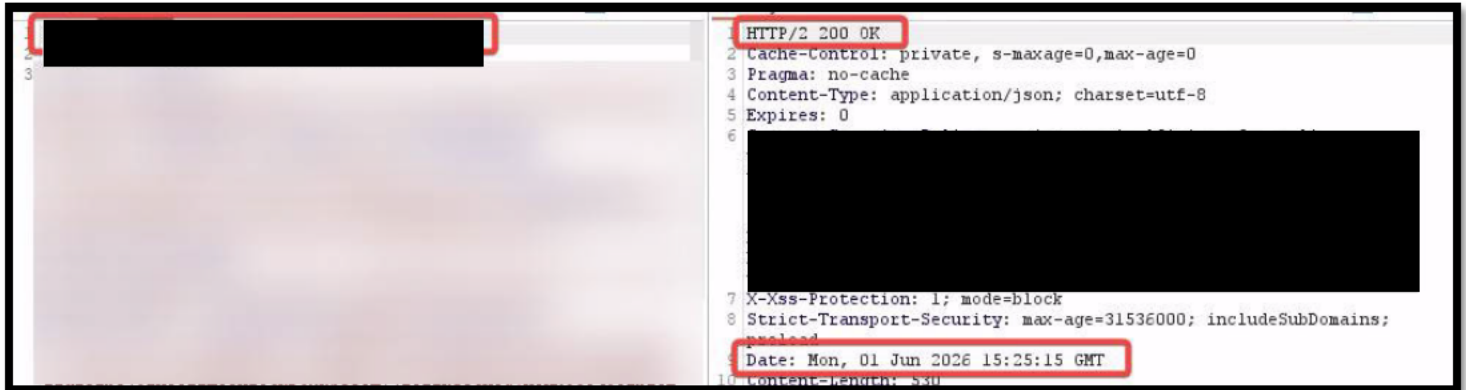
A07:2025 - Authentication Failures

Detected: (QoD 100%) - Impact: Moderate - Ease: Moderate - Risk: Moderate

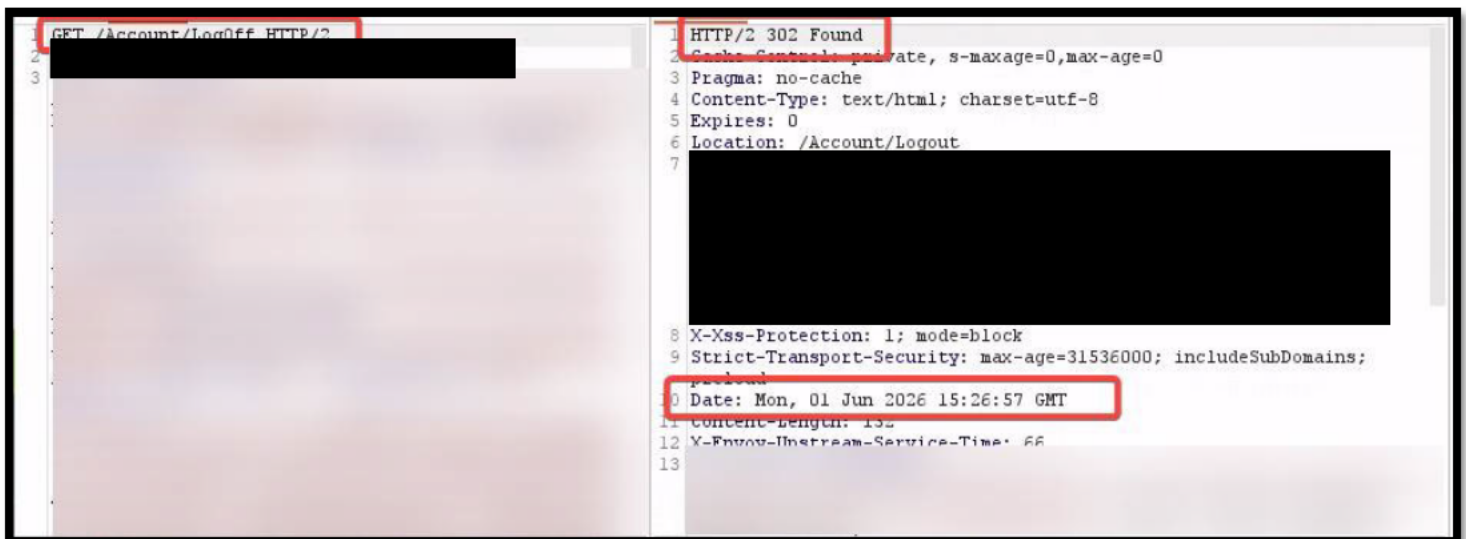
Summary:

The logout function does not expire session cookies, which are used for authentication. This was validated by logging out, clearing the browser cache, and reusing a previous cookie. Abacus also created a new session, and a new cookie was provided, but the previous one was still functional for all authenticated requests, which presents the opportunity for session hijacking.

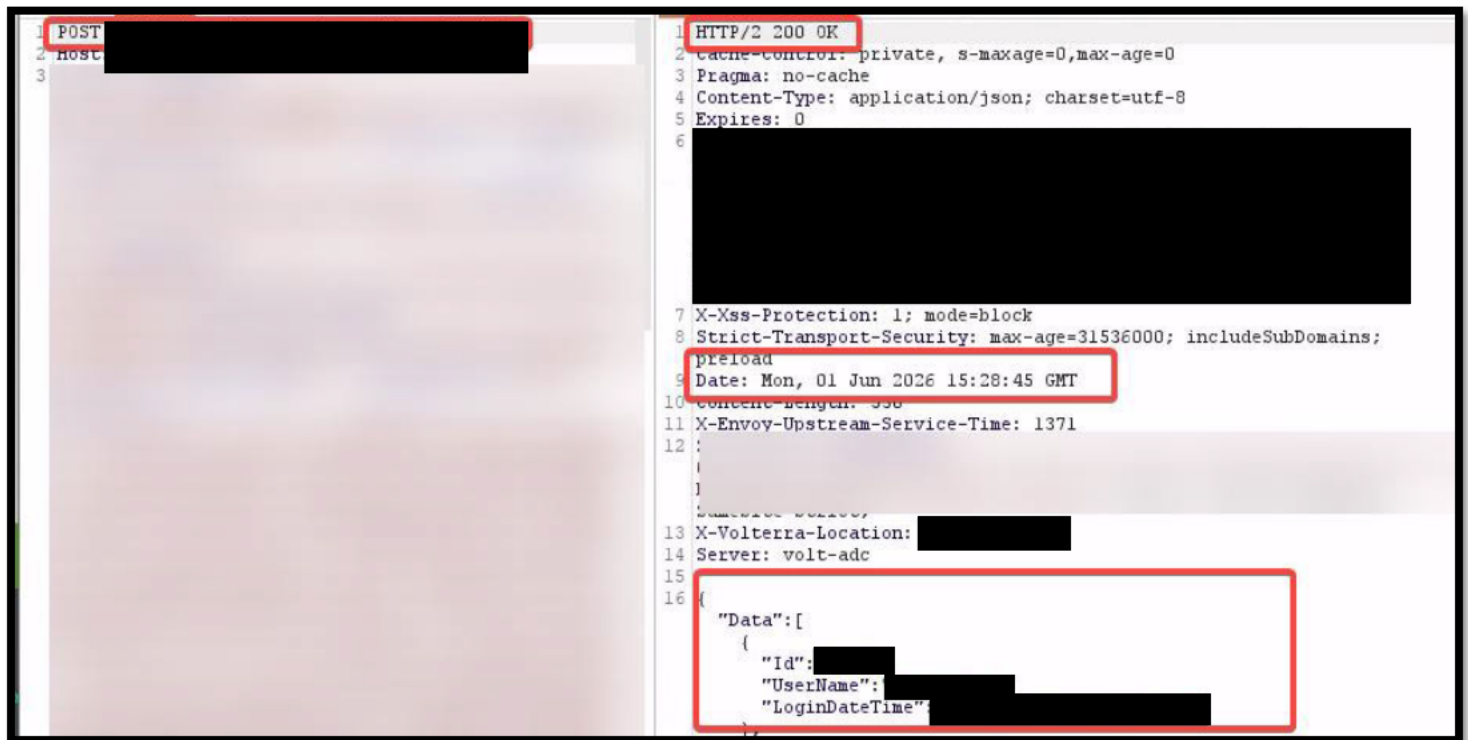
Risk Detection Result(s):



Burp Suite - Demonstrated a base request at 15:25:15 GMT



Burp Suite - Successful logout at 15:26:57 GMT



Burp Suite – GET request was successfully replayed at 15:28:45 GMT

Risk Mitigation Recommendation(s):

- Rewrite the web application code surrounding the logout function, so that cookies are invalidated when the logout function is used.
- Consider implementing a token blacklist in memory to immediately invalidate cookies that are still within their validity period but have been explicitly revoked (i.e. during user logout).

Reference(s):

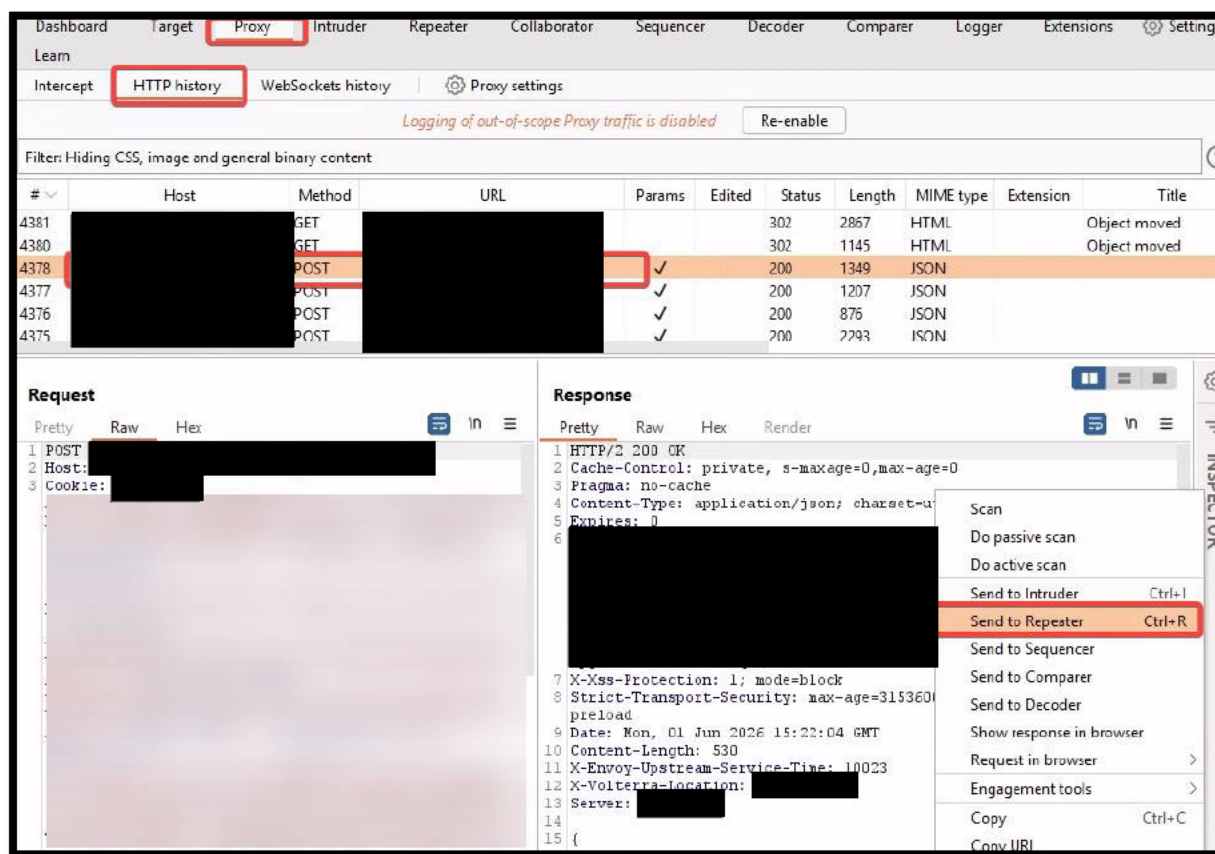
[https://cheatsheetseries.owasp.org/cheatsheets/Session Management Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Session%20Management%20Cheat%20Sheet.html)
<https://www.rfc-editor.org/rfc/rfc7009>

Affected Endpoint(s):

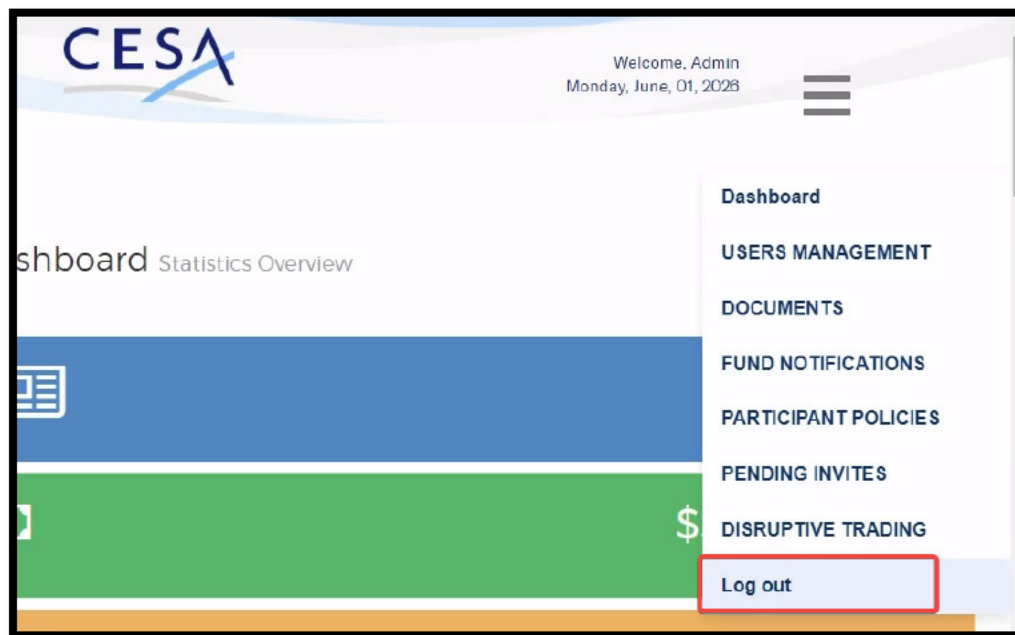
[redacted]

Risk Replication Steps:

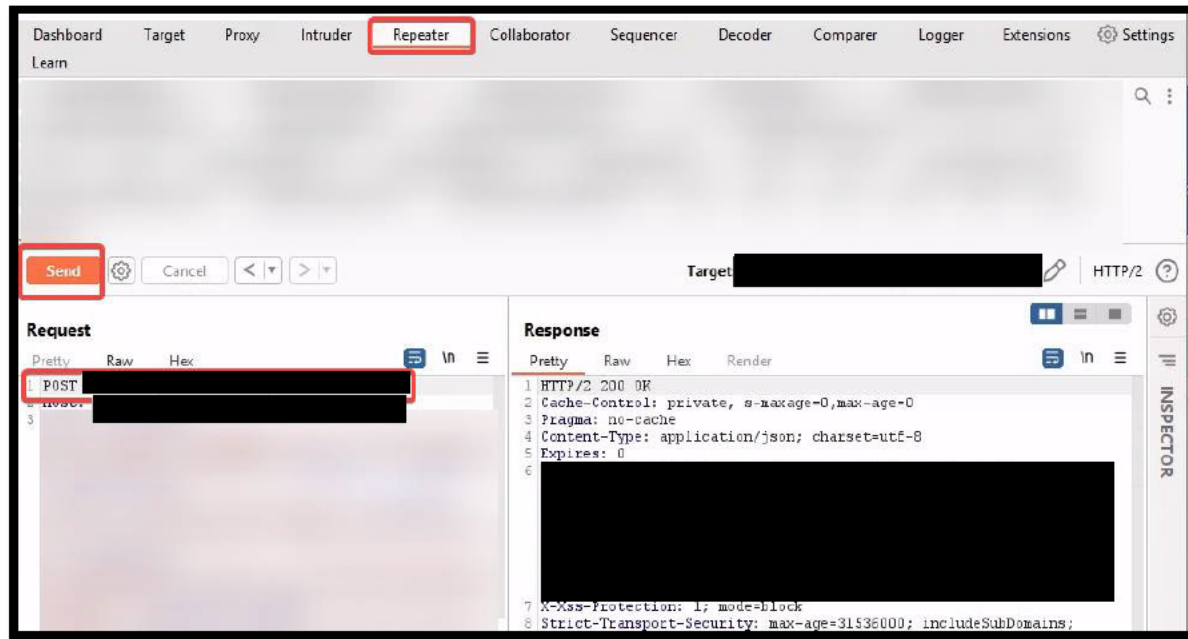
1. In Burp Suite's HTTP history, find a request you would like to test, and send it to Repeater



2. Log out of the application



3. In Burp Suite's Repeater, replay the request by sending it again



Session Cookies Missing HttpOnly, Secure, and/or SameSite Attributes

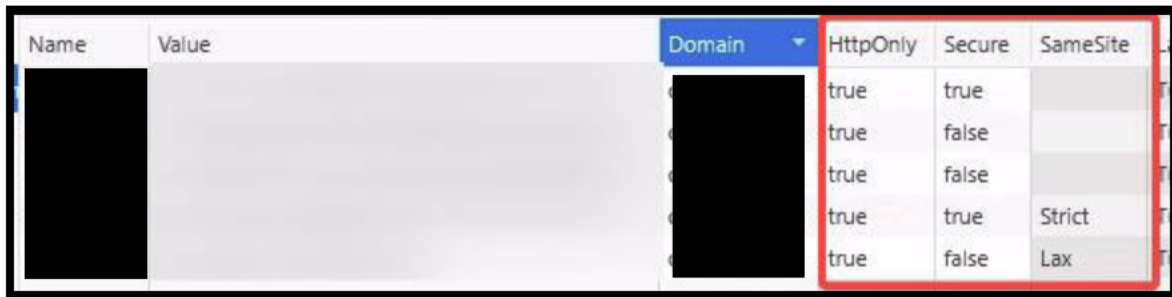
A02:2025 - Security Misconfiguration

Detected: (QoD 100%) - Impact: Low - Ease: Low - Risk: Low

Summary:

Session cookies set by the affected endpoint were missing one or more of the HttpOnly, SameSite, or Secure attributes. Leaving out the HttpOnly attribute means that the cookie may be accessed by client-side JavaScript. The SameSite attribute helps to prevent cookies from being sent to external domains ("cross-origin"), which could leak sensitive information to third parties. The absence of the Secure attribute means that the cookie can be sent over plain-text HTTP. Together, leaving out these attributes may increase the risk of Machine-in-the-Middle, session hijacking, Cross-Site Scripting, and Cross-Site Request Forgery attacks by a malicious actor who can intercept the session cookie or manipulate it with code. These attacks allow various types of access to a user's active logged-in browsing session.

Risk Detection Result(s):

A screenshot of a web browser's developer tools, specifically the 'Cookies' tab. It shows a table of cookies. The columns are 'Name', 'Value', 'Domain', 'HttpOnly', 'Secure', and 'SameSite'. A red rectangle highlights the 'HttpOnly', 'Secure', and 'SameSite' columns for several cookies. The 'HttpOnly' column shows 'true' for all cookies. The 'Secure' column shows 'true' for the first two cookies and 'false' for the others. The 'SameSite' column shows 'Strict' for one cookie and 'Lax' for another, with some cells being empty.

Name	Value	Domain	HttpOnly	Secure	SameSite
			true	true	
			true	false	
			true	false	
			true	true	Strict
			true	false	Lax

Dev Tools - The application's in-scope cookies were missing Secure and/or SameSite attributes

Risk Mitigation Recommendation(s):

- Set all applicable cookies with the appropriate flags.
- With all flags set appropriately, a securely configured cookie could look like the following:
 - Set-Cookie: CookieName=CookieValue; HttpOnly; SameSite=strict; Secure

Reference(s):

<https://owasp.org/www-community/controls/SecureCookieAttribute>

<https://owasp.org/www-community/HttpOnly>

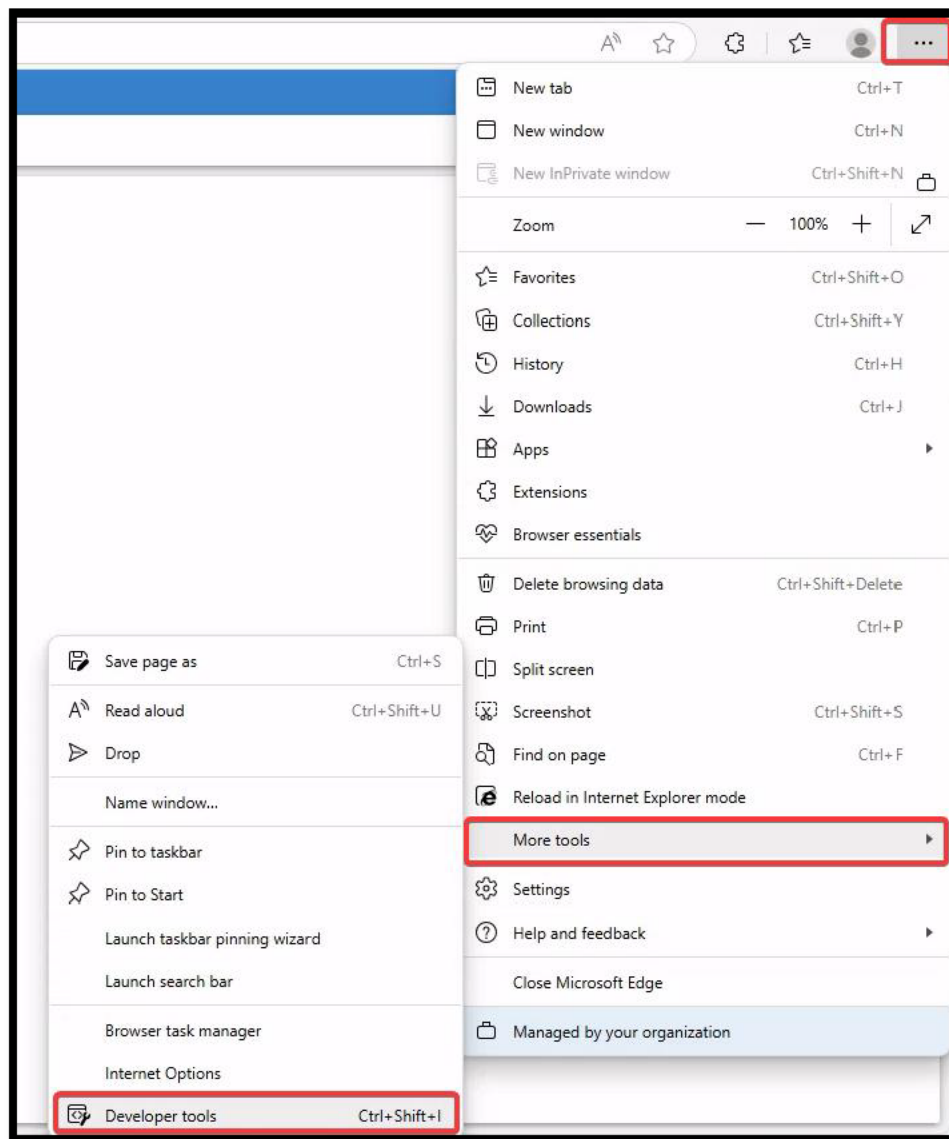
<https://owasp.org/www-community/SameSite>

Affected Endpoint(s):

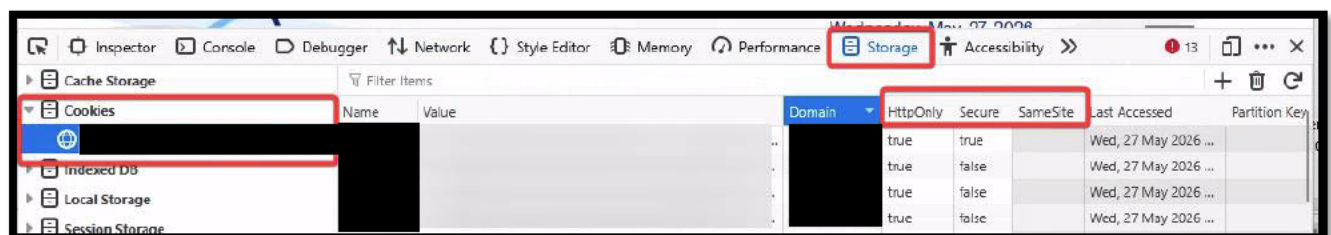
[REDACTED]

Risk Replication Steps:

1. Utilize your browser's Developer Tools



2. Navigate to the *Storage* tab and review the Cookies' HttpOnly, Secure and SameSite security attributes



Excessive Externally Exposed Ports
A06:2025 - Insecure Design
Detected: (QoD 75%) - Impact: Low - Ease: Low - Risk: Low

Summary:

Externally exposed ports and services may allow a malicious actor to gain entry to the underlying infrastructure and perform further attacks against the web application. The web application had port 25 open which is used for Simple Mail Transfer Protocol (SMTP). While Abacus did not have the full context on the usage of this port, it is still recommended to review and close any unnecessary ports.

Risk Detection Result(s):

```
PS C:\Users\Kelly.Hall\OneDrive - Equitable\Documents> python .\scan_ports.py
Scanning ports on 159.60.147.224...

[+] Port 80 is OPEN
[+] Port 443 is OPEN
[+] Port 25 is OPEN
[-] Port 110 is CLOSED or Filtered
[-] Port 135 is CLOSED or Filtered
```

scan_ports.py - The web application had port 25 open

Risk Mitigation Recommendation(s):

- Review open ports and services.
- If the service needs to be externally exposed, consider filtering requests based on IP allow-listing at the system or network's firewall.

Reference(s):

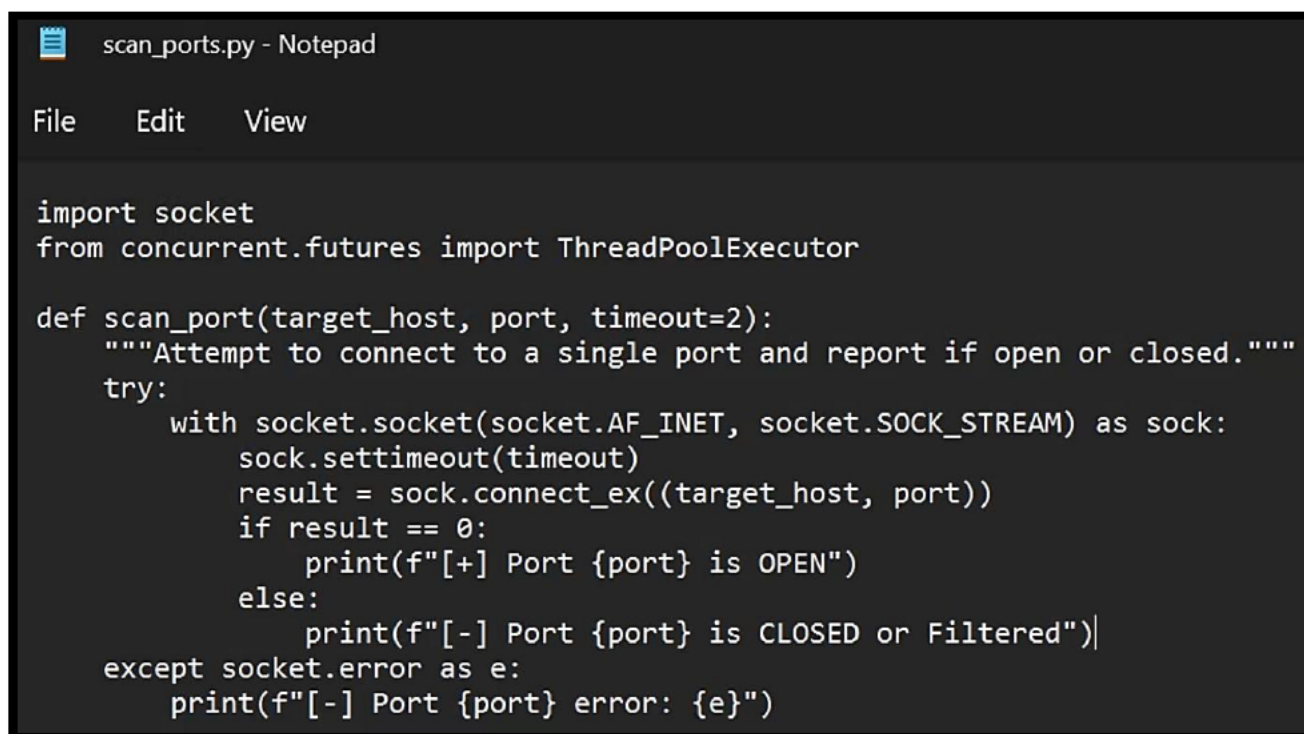
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/01-Test_Network_Infrastructure_Configuration

Affected Endpoint(s):

[https://cesadministrationuat\[REDACTED\].com/](https://cesadministrationuat[REDACTED].com/)*

Risk Replication Steps:

1. Utilize a tool such as Shodan or a python script as shown below that checks the application for open ports



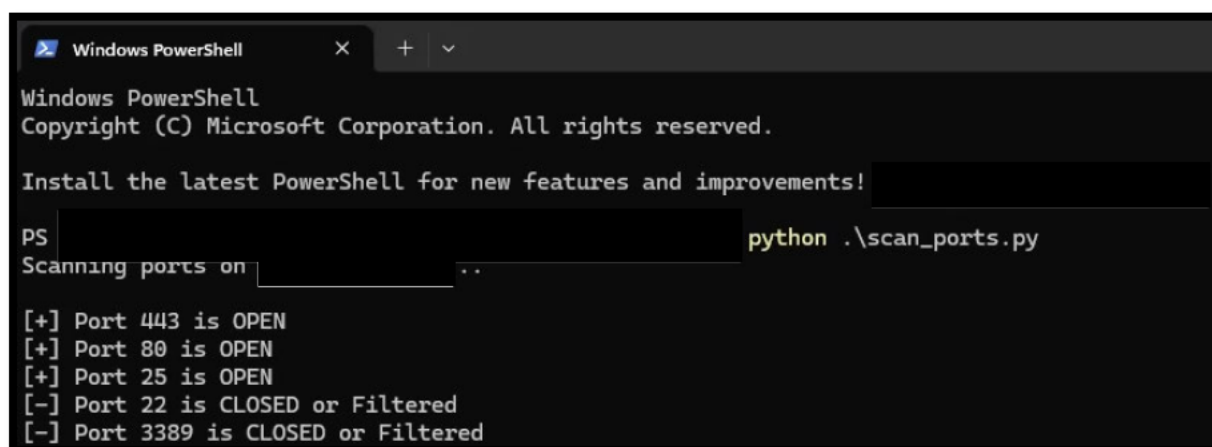
```
scan_ports.py - Notepad

File Edit View

import socket
from concurrent.futures import ThreadPoolExecutor

def scan_port(target_host, port, timeout=2):
    """Attempt to connect to a single port and report if open or closed."""
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
            sock.settimeout(timeout)
            result = sock.connect_ex((target_host, port))
            if result == 0:
                print(f"[+] Port {port} is OPEN")
            else:
                print(f"[-] Port {port} is CLOSED or Filtered")
    except socket.error as e:
        print(f"[-] Port {port} error: {e}")
```

2. Review the results



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!

PS > python .\scan_ports.py
Scanning ports on 10.10.10.10 ..

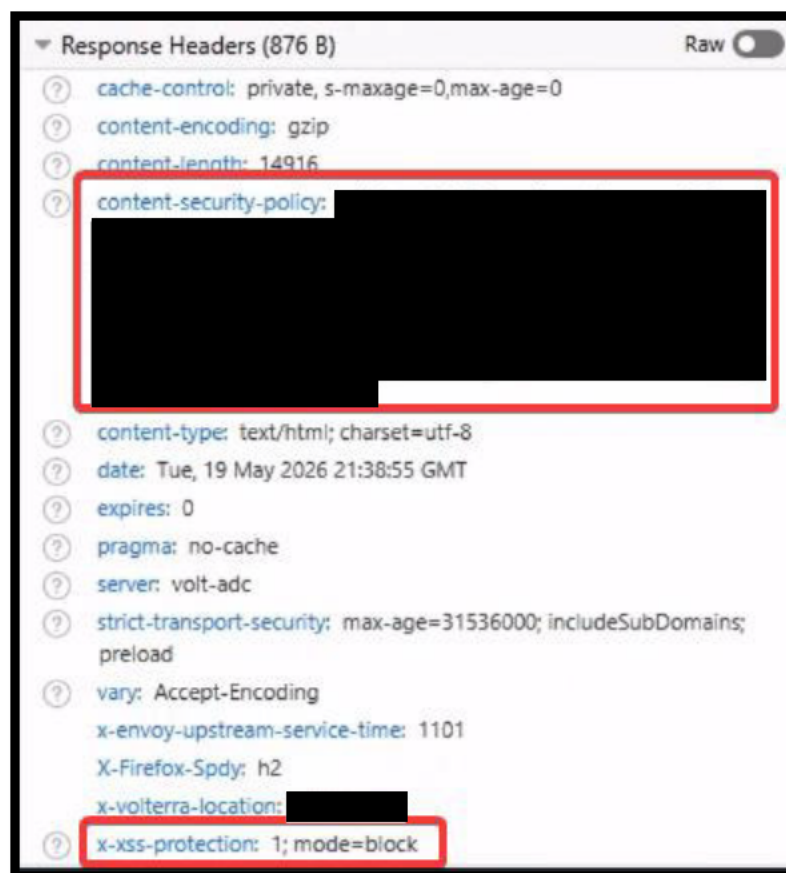
[+] Port 443 is OPEN
[+] Port 80 is OPEN
[+] Port 25 is OPEN
[-] Port 22 is CLOSED or Filtered
[-] Port 3389 is CLOSED or Filtered
```

Misconfigured Content-Security-Policy Header
A02:2025 - Security Misconfiguration
Detected: (QoD 100%) - Impact: Low - Ease: Low - Risk: Low

Summary:

The web application's response headers included a "Content-Security-Policy" (CSP) with insecure directives configured. The application implemented 'unsafe-inline' directives without a nonce value, which could lead to the untrusted execution of JavaScript or style sheets. A CSP header provides an explicit policy specifying acceptable origins from which various site components may be loaded. This adds an important layer of security that helps to protect against certain classes of attacks, such as Cross-Site Scripting (XSS), Clickjacking, and Subresource Integrity violations. In most cases, a well-defined CSP can completely take the place of older security headers such as "X-Frame-Options" and "X-XSS-Protection". It is of note that the application included 'X-XSS-Protection' in its response headers, which is considered deprecated in favor of a well-defined CSP.

Risk Detection Result(s):



Dev Tools – The application had a misconfigured CSP header

Risk Mitigation Recommendation(s):

- Implement the 'default-src', 'base-uri', 'script-src', 'style-src', and 'object-src' directives for all websites, and do not allow for 'unsafe-inline' or 'unsafe-eval' script execution without a nonce.
- Implement form jacking protection for all websites via the 'form-src' directive, which should be set to 'self' or 'none'.
- Implement the 'frame-ancestors' directive across all websites to protect against clickjacking.
- Consider implementing the 'upgrade-insecure-requests' directive to better protect against machine-in-the-middle attacks.
- Consider implementing the 'report-uri' directive for reporting CSP violations.

Reference(s):

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>

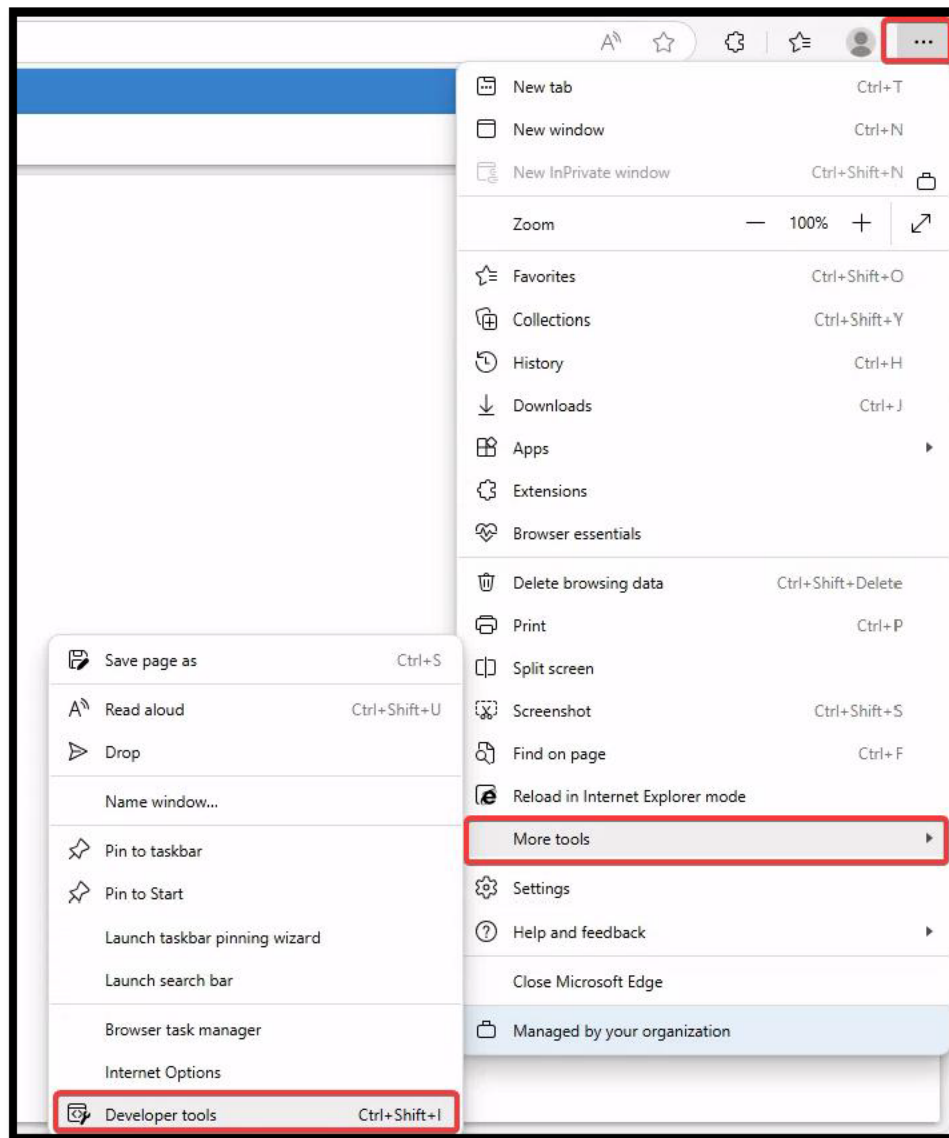
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy>

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Content_Security_Policy_Cheat_Sheet.md

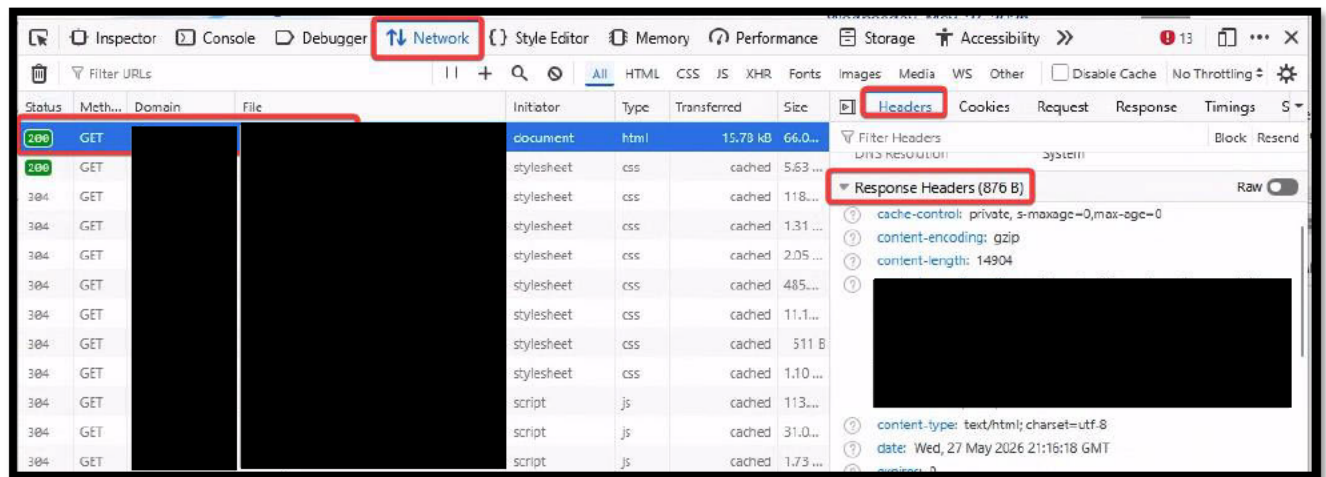
Affected Endpoint(s):

Risk Replication Steps:

1. Utilize your browser's Developer Tools



2. Navigate to the Network tab and review the Dashboard page's Response Headers



Weak File Upload Protections
A02:2025 - Security Misconfiguration
Detected: (QoD 100%) - Impact: Low - Ease: Low - Risk: Low

Summary:

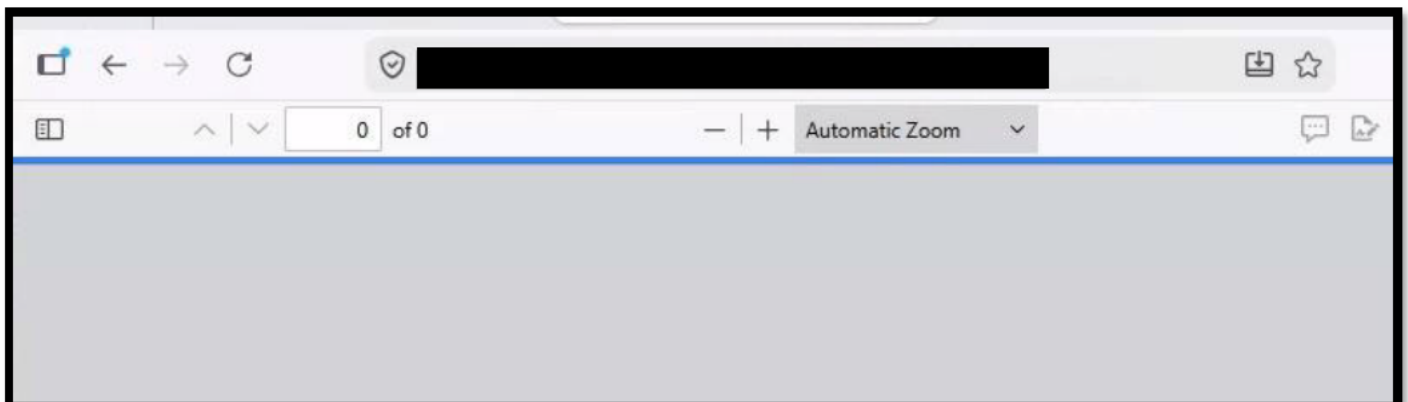
Weak file upload protections were identified in the /AdminDocuments/Create endpoint. This endpoint allowed a PDF file containing malicious code to be uploaded. While no Cross-Site Scripting (XSS) alerts were triggered, this could create an opportunity for exploitation using malicious files and payloads. Additionally, after uploading, the document details page reveals the location of the file, which could be easily accessed by force-browsing to the route. Canary Tokens are used to detect if the server will allow unauthorized access. After successfully uploading a PDF containing a canary token and deleting the document, a notification was triggered almost a week later, showcasing that the file was accessed. Successful exploitation could result in remote file inclusion that may lead to cross-site scripting (XSS) attacks or malicious file uploads.

Risk Detection Result(s):

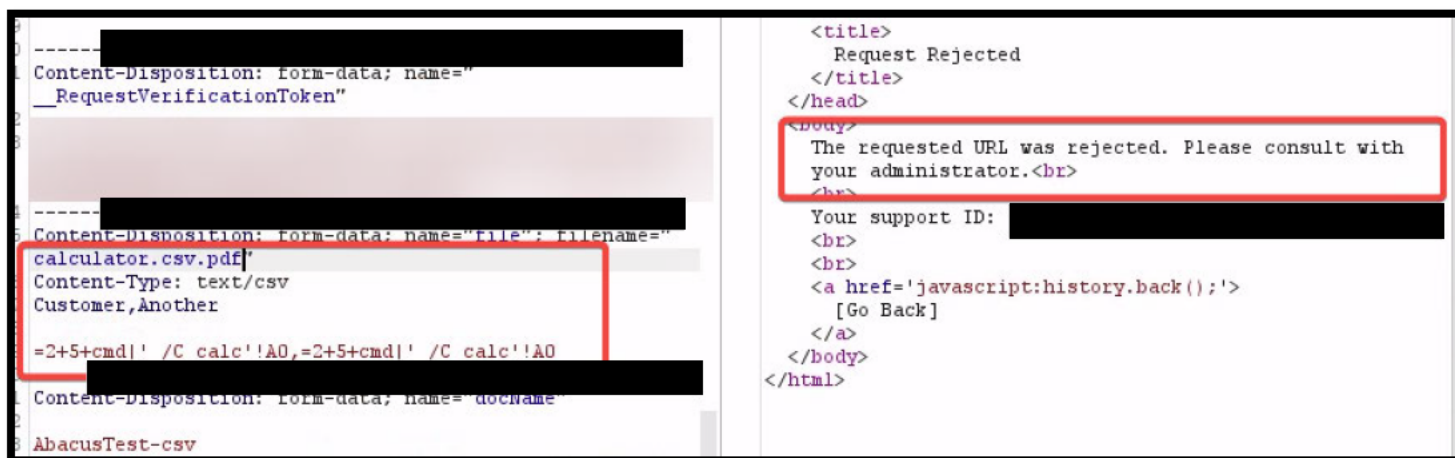
```
24 -----
25 Content-Disposition: form-data; name="file"; filename="
testPDF.pdf"
26 Content-Type: application/pdf
27
28 %PDF-1.7
29 1 0 obj
30 <<
31 /pages 1 0 R
32 /OpenAction 2 0 R
33 >>
34 Endobj
35 2 0 obj
36 <<
37 /s /JavaScript
38 /JS (app.alert(1))
39 >>
40 Endobj
41 Trailer
42 <<
43 /Root 1 0 R
44 >>
45 %%EOF
46 -----
47 Content-Disposition: form-data; name="docName"
48
49 AbacusTest-pdfxss
50 -----
51 Content-Disposition: form-data; name="allUsers"
52
53 false
54 -----
55 Content-Disposition: form-data; name="userAccessId_input"
56
57
58 -----
59 Content-Disposition: form-data; name="userAccessId"

4 Content-Type: text/html; charset=utf-8
5 Expires: 0
6 Location: /AdminDocuments
7
8
9
10 X-Xss-Protection: 1; mode=block
11 Strict-Transport-Security: max-age=31536000;
includeSubDomains; preload
12 Date: Thu, 21 May 2026 17:25:57 GMT
13 Content-Length: 132
14 X-Envoy-Upstream-Service-Time: 748
15 Set-Cookie: TS01dc4fc6=
16
17 Path=/; Secure; HttpOnly; SameSite=Strict;
18 X-Volterra-Location:
19 Server: volt-adc
20
21 <html>
22 <head>
23 <title>
24 Object moved
25 </title>
26 </head>
27 <body>
28 <h2>
29 Object moved to <a href="/AdminDocuments">
30 here
31 </a>
```

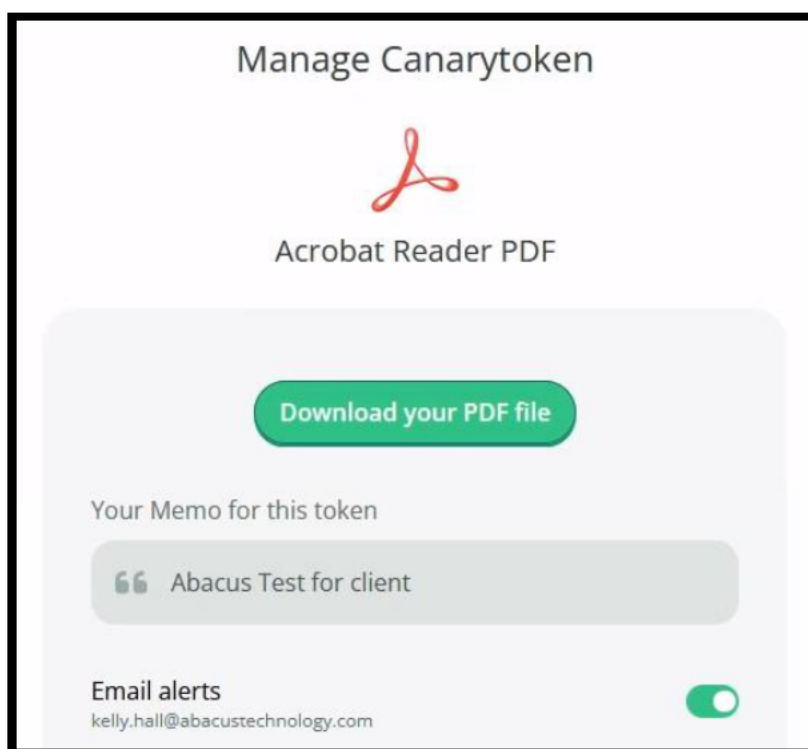
Burp Suite – The application accepted a PDF with an XSS attack



Customer Portal – Accessing the PDF did not trigger any alerts



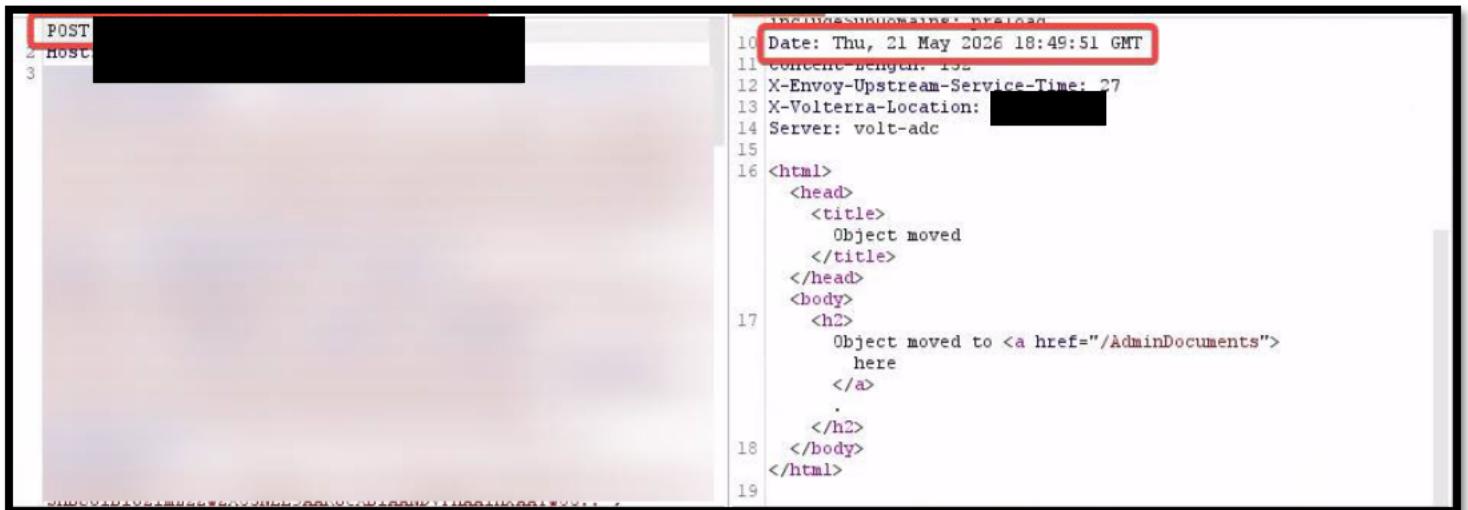
Burp Suite – The application did not allow invalid extensions to be uploaded



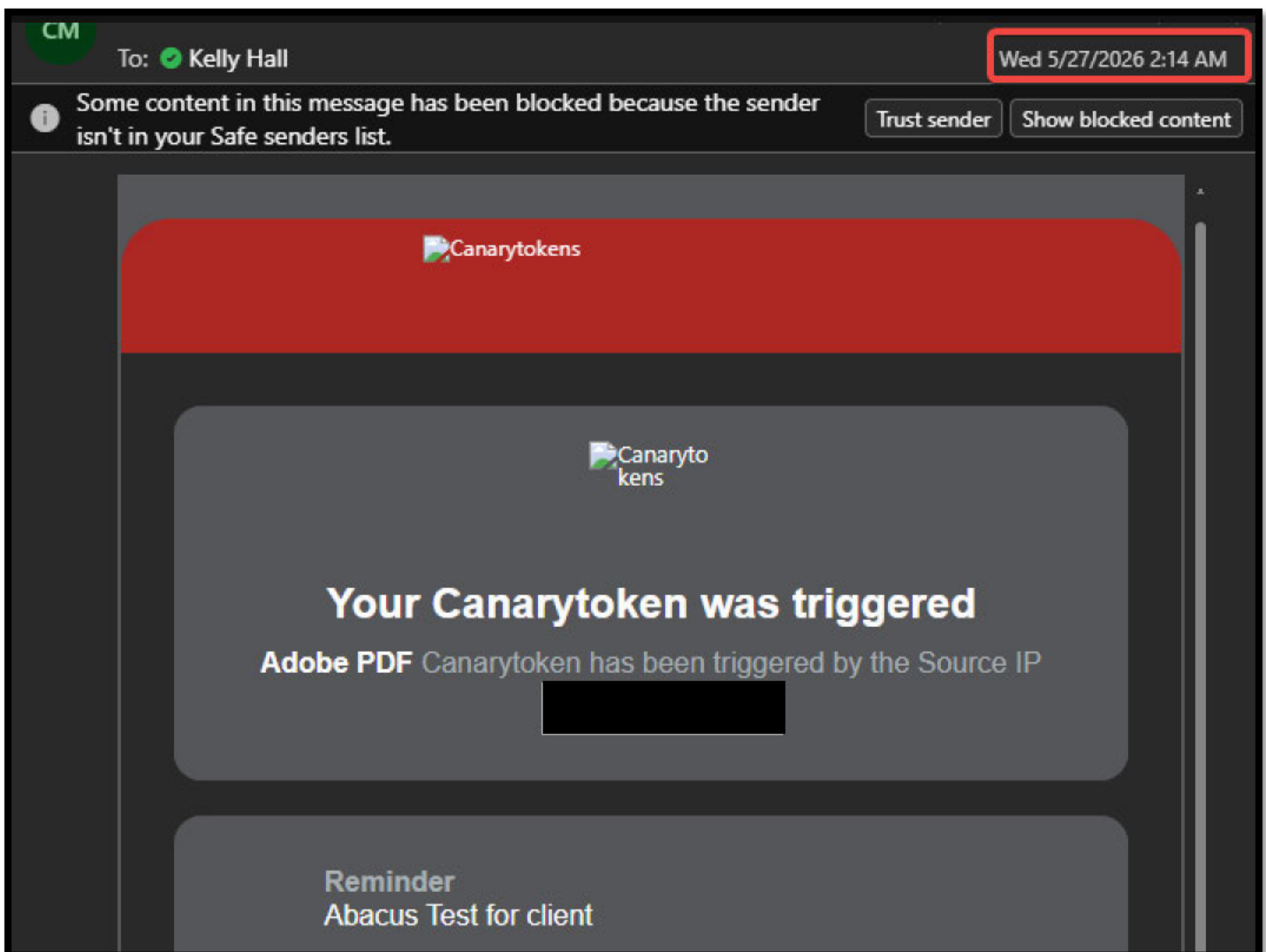
Canarytokens.org – A PDF file containing a canary token was created



Burp Suite – The application allowed the PDF file containing a canary token to be uploaded under the ID of 14324



Burp Suite – The PDF file containing a canary token was deleted on May 21, 2026



Outlook – The canary token was triggered on May 27, 2026

Risk Mitigation Recommendation(s):

- If the intent is for the user to only be able to upload local image files, implement appropriate restrictions.
- Ensure that the data provided is a valid domain name.
- Ensure use of secure deserialization to prevent potential for RCE.
- Consider implementing input validation controls and application logic that would check the vulnerable parameter byte-size and reject all requests over a specific byte-size amount.
- Implement input sanitization that restricts the file type to a MIME type in alignment with the X-Content-Type-Options header specified in this report.

Reference(s):

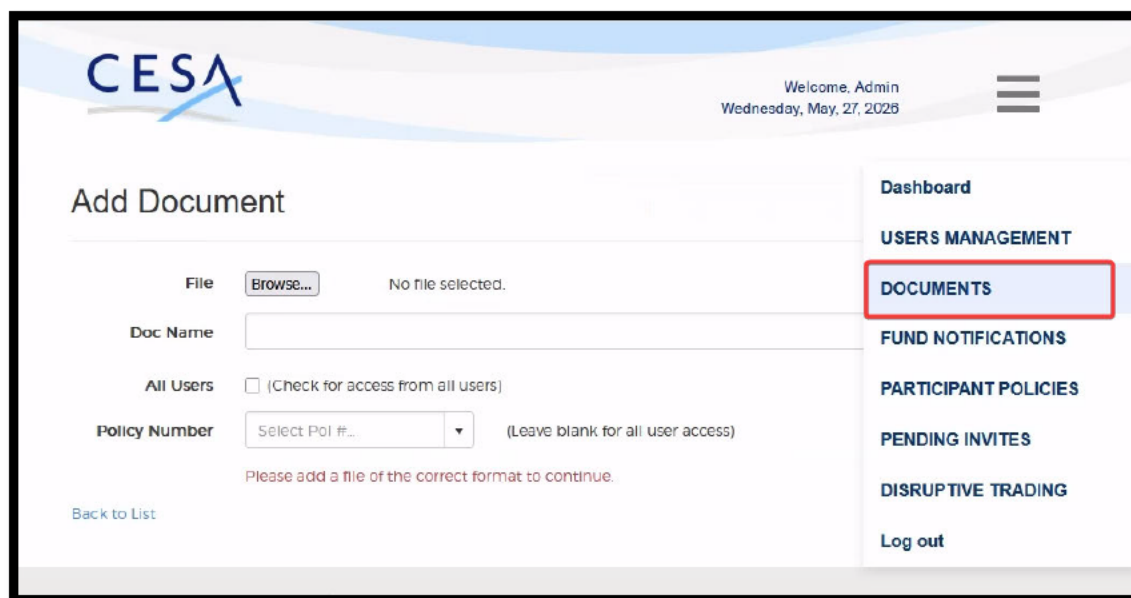
https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html

Affected Endpoint(s):

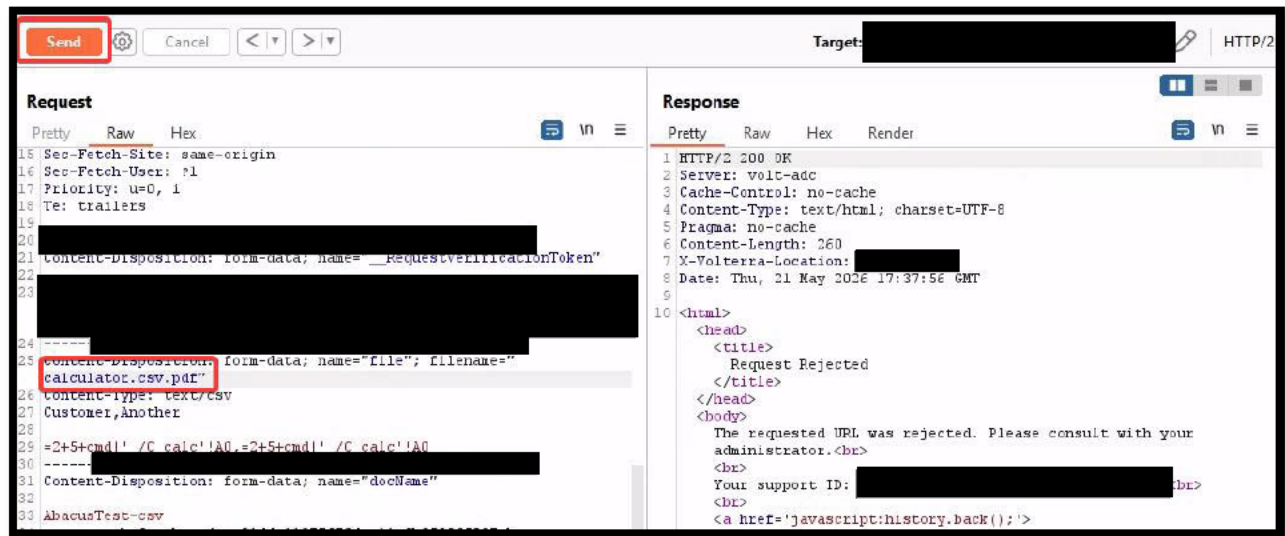
[https://cesadministrationuat\[REDACTED\].com/AdminDocuments/Create](https://cesadministrationuat[REDACTED].com/AdminDocuments/Create),
[https://cesadministrationuat\[REDACTED\].com/pdfDocs/*](https://cesadministrationuat[REDACTED].com/pdfDocs/*)

Risk Replication Steps:

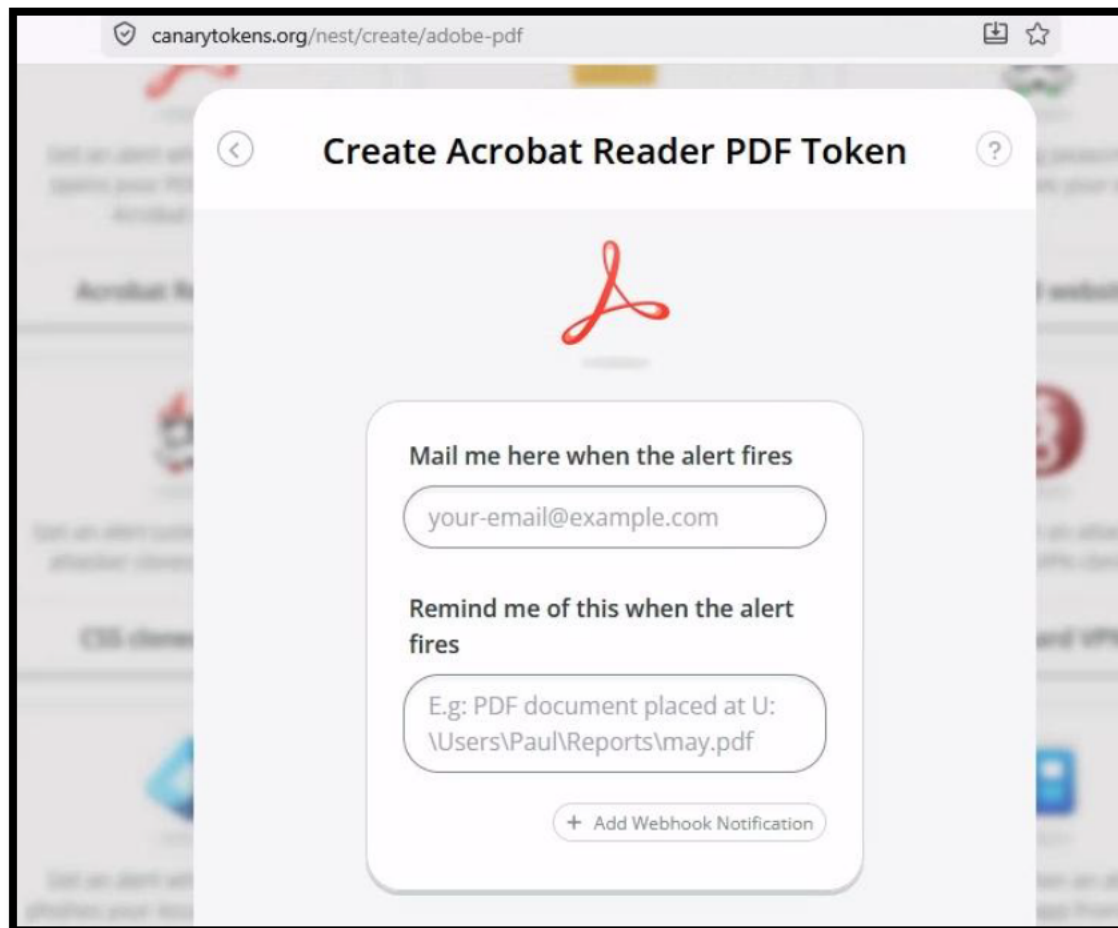
1. As Admin, Navigate to the file upload functionality under Documents



2. Capture the request in Burp Suite's repeater to see how the application responds to different file extensions and file content



3. Navigate to canarytokens.org and create a PDF file containing a canary token



4. Upload the canary token file and wait for an email notification to come through when it is activated

Add Document

File

Browse...

Doc Name

All Users

☐ (Check for access from all users)

Policy Number

Select Pol #...

▼

(Leave blank for all user access)

Create

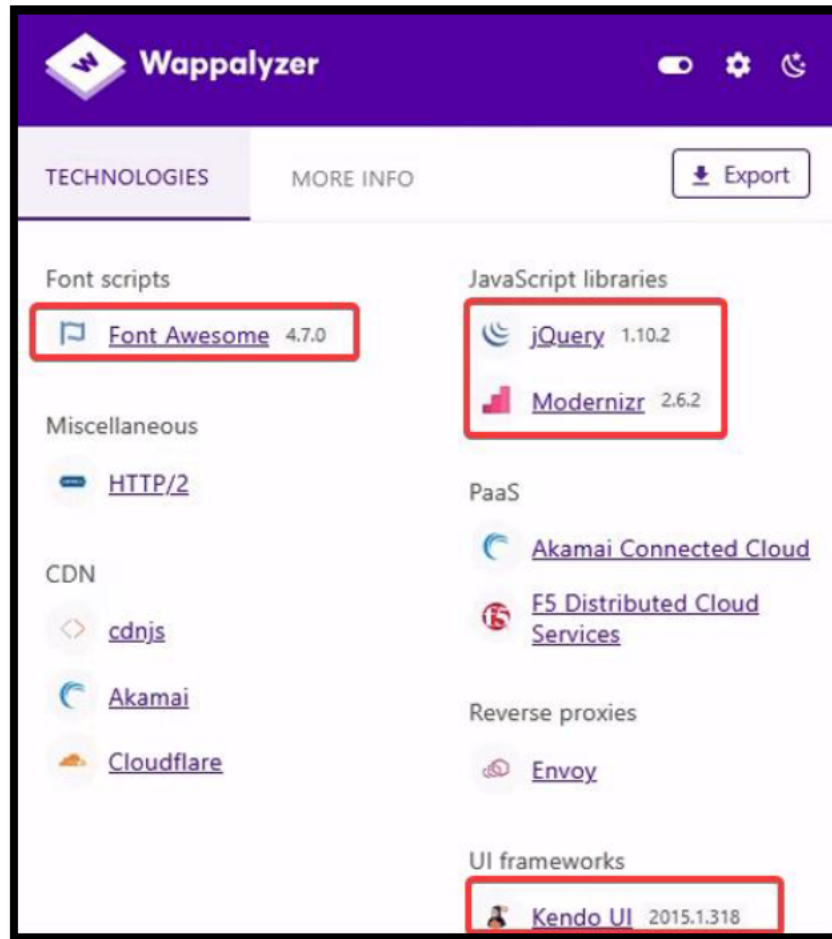
[Back to List](#)

Outdated JavaScript Libraries
A06:2025 - Insecure Design
Informational

Summary:

Out of date versions of Modernizr, and Kendo UI were discovered on the application. While there are no direct vulnerabilities for the current version implemented, it is still recommended to update these libraries to their latest and stable versions.

Risk Detection Result(s):



Wappalyzer – The application utilized outdated frameworks and libraries

Risk Mitigation Recommendation(s):

- Update Modernizr and Kendo UI to the latest version(s), which at the time of this writing are versions 3.13.1 and 2026.2.520, respectively.

Reference(s):

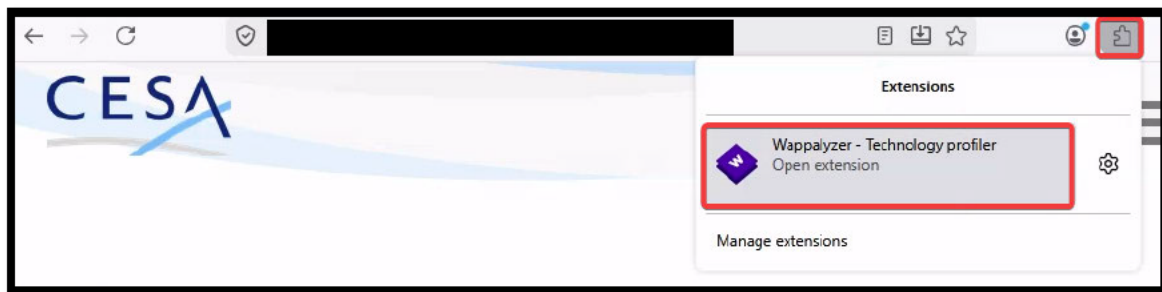
<https://security.snyk.io/package/npm/modernizr>
<https://security.snyk.io/package/npm/kendo-ui-core>

Affected Endpoint(s):

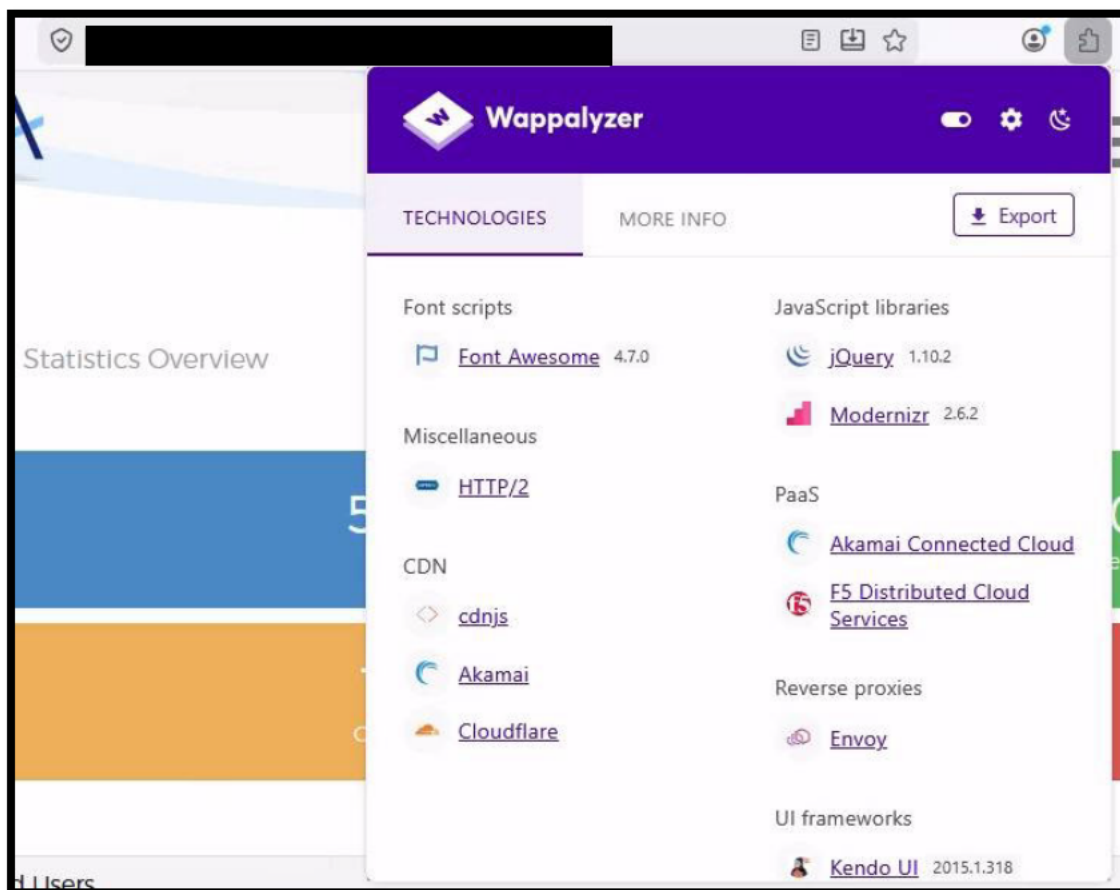
[REDACTED]

Risk Replication Steps:

1. Utilize your browser's Wappalyzer extension



2. Review the framework and library versions implemented

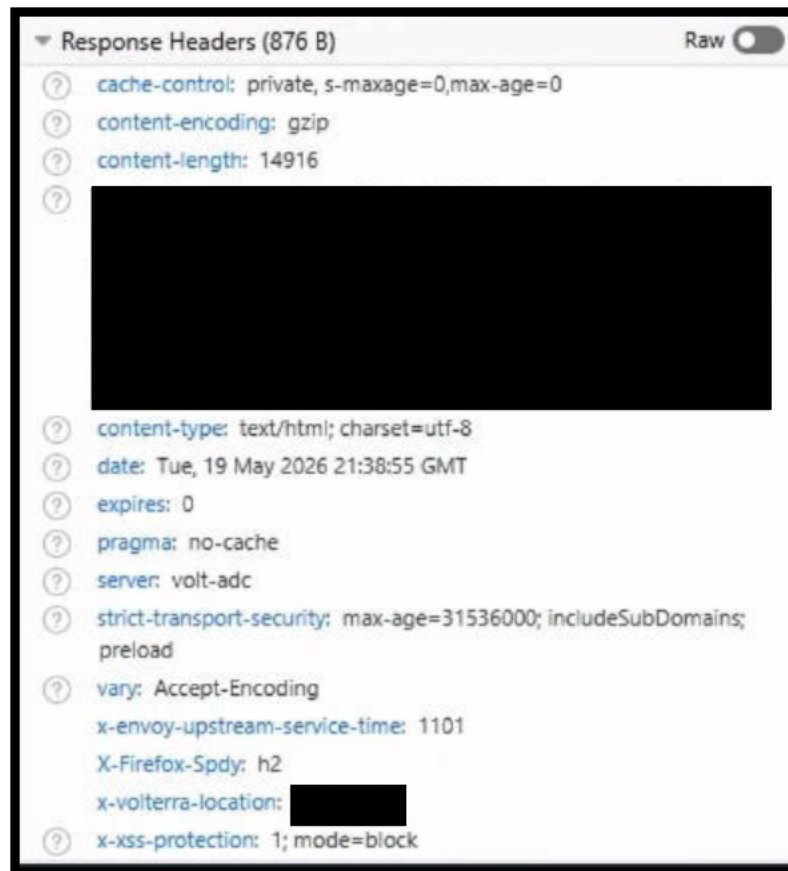


Missing Security Header: Referrer Policy
A06:2025 - Insecure Design
Informational

Summary:

The affected web application did not implement the Referrer-Policy header, which controls how much referrer information the browser will send with each request originating from the current web application. For instance, if a user visits the web page "http://example.com/pricing/" and clicks on a link from that page, going to e.g. "https://www.google.com", the browser will send to Google the full originating URL in the Referrer header, assuming the Referrer-Policy header is not set. The originating URL could be considered sensitive information, and it could be used for user tracking.

Risk Detection Result(s):



Dev Tools – The application was missing a referrer header

Risk Mitigation Recommendation(s):

- Implement a Referrer-Policy by using the Referrer-Policy response header or by declaring it in the meta tags. It is also possible to control referrer information over an HTML element by using the rel attribute.
- In a response header: Referrer-Policy: no-referrer | same-origin | origin | strict-origin | no-origin-when-downgrading
- In a META tag: <meta name=Referrer-Policy value=no-referrer | same-origin/>
- In an element attribute:
- Alternatively:

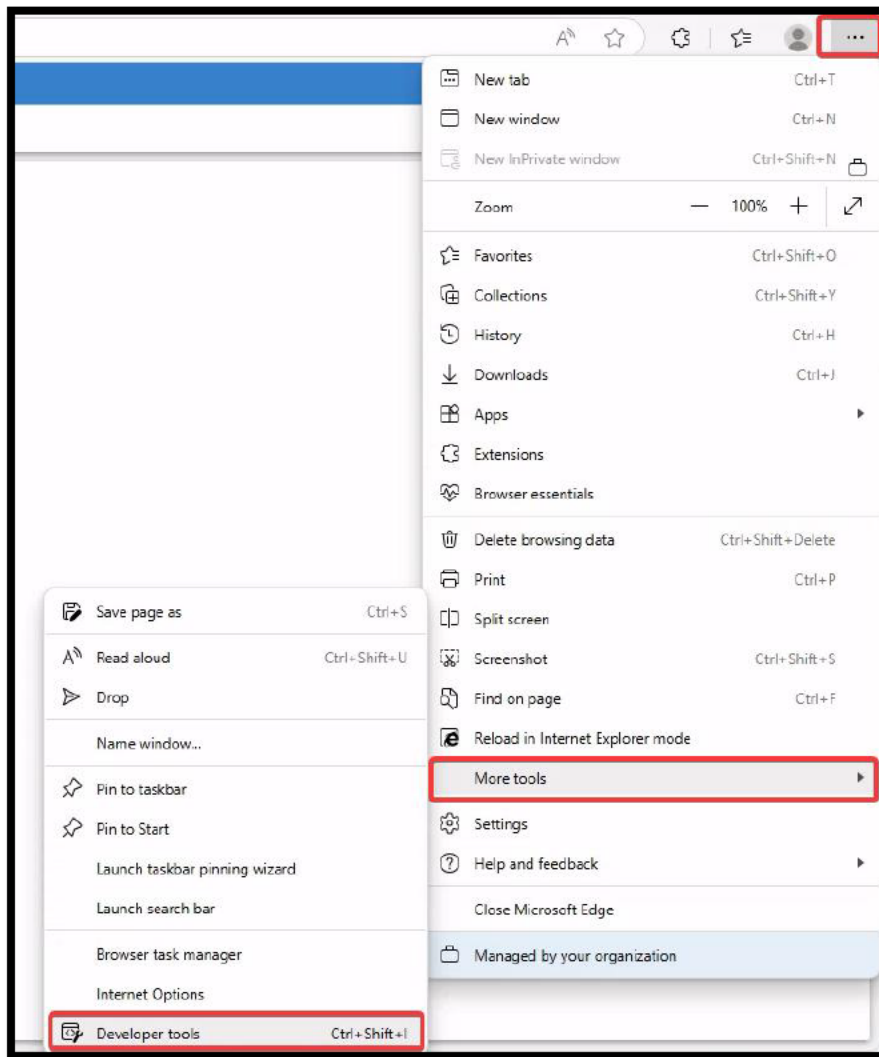
Reference(s):

https://developer.mozilla.org/en-US/docs/Web/Security/Referer_header:_privacy_and_security_concerns
<https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers-Cheat-Sheet.html>

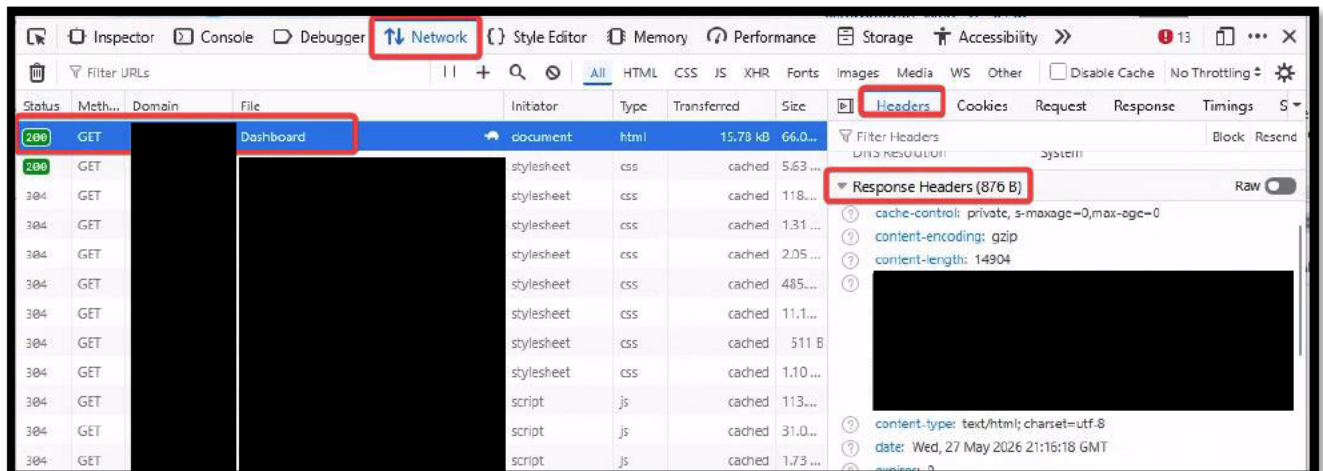
Affected Endpoint(s):

Risk Replication Steps:

1. Utilize your browser's Developer Tools



2. Navigate to the Network tab and review the Dashboard page's Response Headers



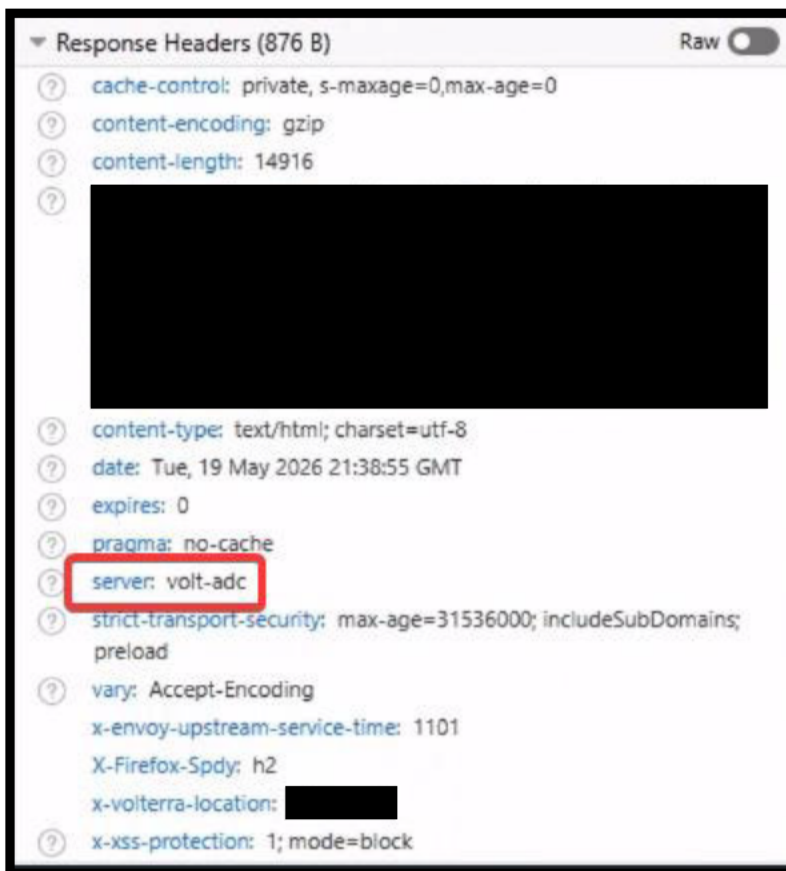
Webserver Technology Exposed via Server Header

A06:2025 - Insecure Design
Informational

Summary:

The server header exposed the technologies used by the webserver as 'volt-adc'. This information exposes the server to attackers. Using the information in this header, attackers can find vulnerabilities easier.

Risk Detection Result(s):



Dev Tools – The application exposed server information via response headers

Risk Mitigation Recommendation(s):

- Remove all server headers from server responses.

Reference(s):

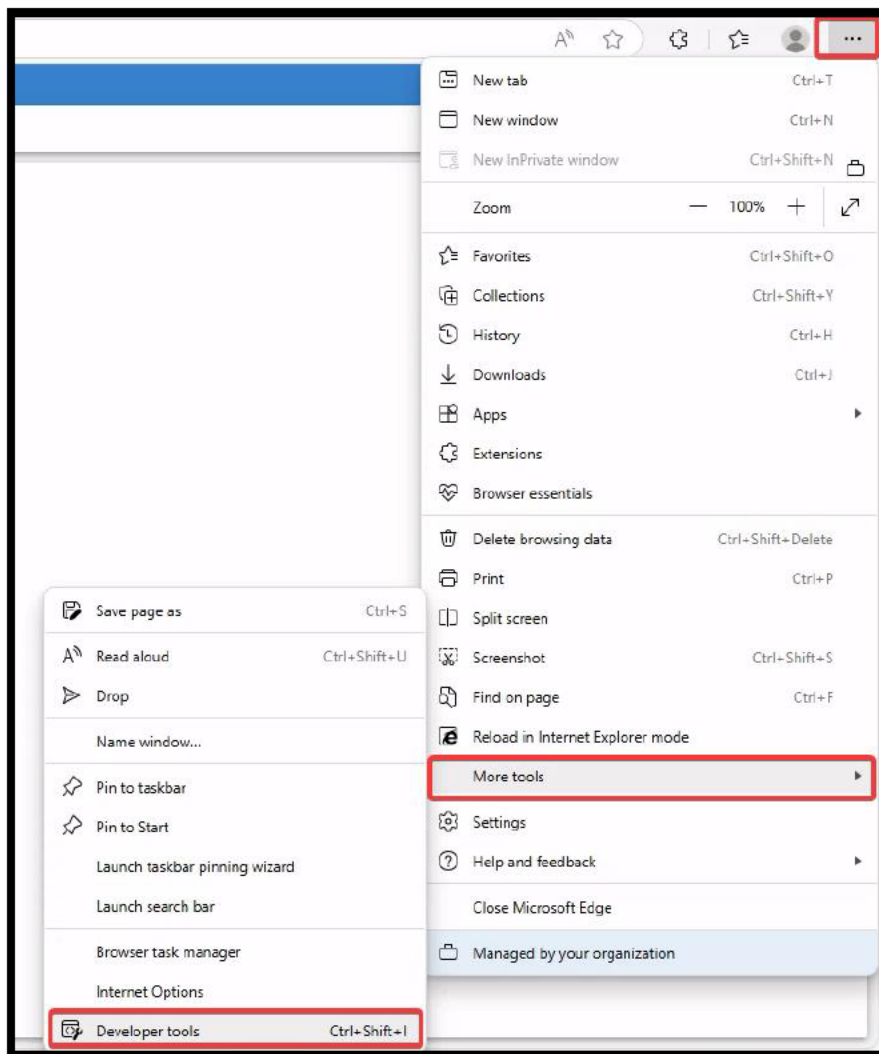
https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers_Cheat_Sheet.html

Affected Endpoint(s):

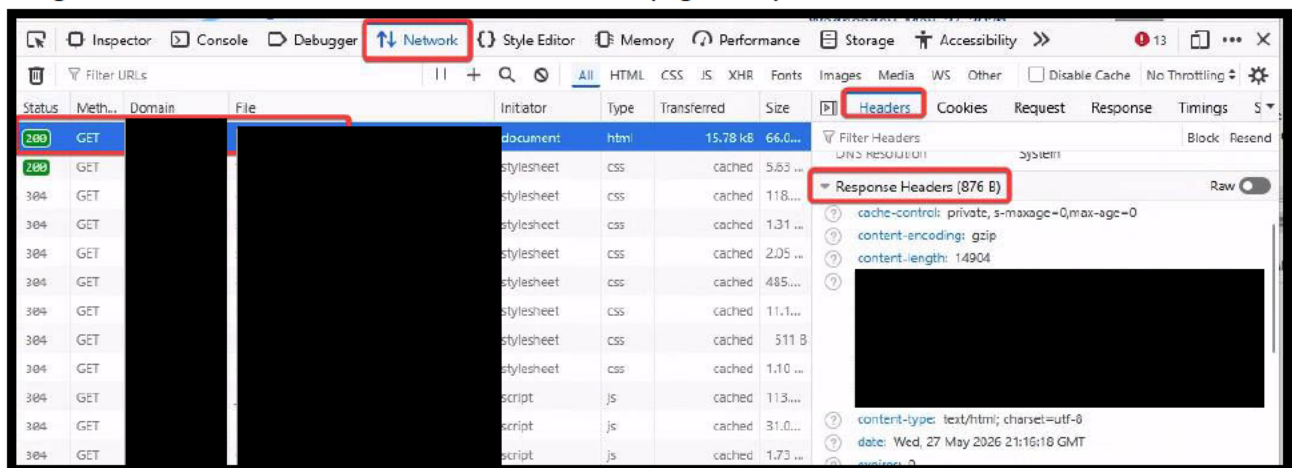
[REDACTED]

Risk Replication Steps:

1. Utilize your browser's Developer Tools



2. Navigate to the Network tab and review the Dashboard page's Response Headers

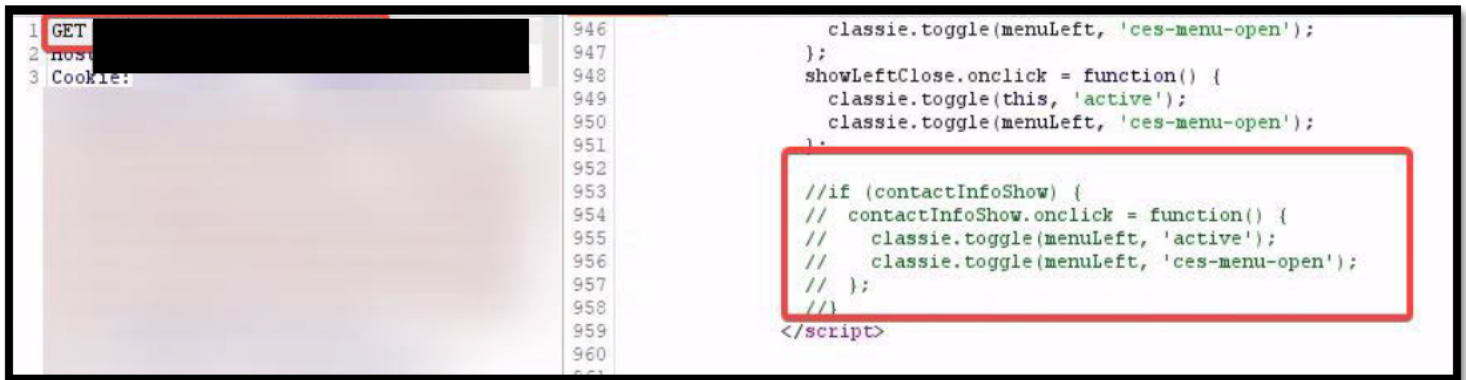


Commented out Functionality Exposed
A06:2025 - Insecure Design
Informational

Summary:

The application's source code revealed developer defined functionality that was commented out and unused. While Abacus was unable to exploit the current commented out functionality, added functionality that's not meant to be used in the application could allow a malicious actor could un-comment it, and access it.

Risk Detection Result(s):



```
1 GET [REDACTED]
2 Host: [REDACTED]
3 Cookie: [REDACTED]

946     classie.toggle(menuLeft, 'ces-menu-open');
947 };
948 showLeftClose.onclick = function() {
949     classie.toggle(this, 'active');
950     classie.toggle(menuLeft, 'ces-menu-open');
951 };
952
953
954 //if (contactInfoShow) {
955 //    contactInfoShow.onclick = function() {
956 //        classie.toggle(menuLeft, 'active');
957 //        classie.toggle(menuLeft, 'ces-menu-open');
958 //    };
959 //}
960 </script>
```

Burp Suite - The Dashboard page contained commented-out functionality

Risk Mitigation Recommendation(s):

- Remove unused code in the production environment.

Reference(s):

https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html

Affected Endpoint(s):

[REDACTED]

Risk Replication Steps:

1. In Burp Suite's HTTP history, find a Dashboard request and review the source code in the response for commented out functionality

The screenshot displays the Burp Suite interface. At the top, the 'Proxy' tab is selected. Below it, the 'HTTP history' tab is active, showing a list of intercepted requests. The first request is highlighted. The 'Request' pane on the left shows a GET request to a dashboard page. The 'Response' pane on the right shows the corresponding HTML response. A red box highlights a section of JavaScript code in the response, which appears to be commented out or partially obscured.

#	Host	Met...	URL	Params	Edited	Status	Length	MIME type	Extension	Title
1		GET				200	67097	HTML		
104		GET				200	66391	HTML		
105		GET		✓		200	114069	script		
116		GET		✓		200	2086103	script		
121		GET		✓		200	114069	script		
122		GET		✓		200	2086103	script		
314		GET				200	638838	HTML		
350		GET				200	66870	HTML		
411		GET				200	66893	HTML		

```
944
945
946 showLeft.onclick = function() {
947   classie.toggle(this, 'active');
948   classie.toggle(menuLeft, 'ces-menu-open');
949 };
950 showLeftClose.onclick = function() {
951   classie.toggle(this, 'active');
952   classie.toggle(menuLeft, 'ces-menu-open');
953 };
954
955 //if (contactInfoShow) {
956 //  contactInfoShow.onclick = function() {
957 //    classie.toggle(menuLeft, 'active');
958 //    classie.toggle(menuLeft, 'ces-menu-open');
959 //  };
960 //}
961
962 </script>
```


Conclusion

While Abacus makes a best effort to rank the severity of vulnerabilities, it is not possible for any assessment to guarantee that low-risk issues cannot be chained together and significantly impact ██████████ systems. All risk ratings in this report are a combination of industry-standard ratings and considerations for unique business risk as identified by Abacus. Special consideration should also be given to the fact that multiple low-risk issues may compound into higher-risk threats.

Having concluded penetration testing activities, Abacus assessed that ██████████ in-scope applications collectively had 2 high-risks, 1 moderate-risks, 4 low-risks, and 4 informational findings. However, no access to the underlying server infrastructure was obtained, nor could any potentially sensitive data be modified.

It is important to note that the risk mitigation recommendations contained in this report reflect industry best practices and Abacus's limited knowledge of ██████████. Abacus does not warrant the compatibility or operational effectiveness of the risk mitigation recommendations provided in this report, as Abacus does not have any understanding of the nuances and caveats of ██████████ network environment. Abacus highly recommends that ██████████ technology department assesses the compatibility and operational effectiveness of the risk mitigation recommendations provided in this report and uses a change management process to validate, plan and implement remediation changes in a UAT environment first.

About the Security Testers:

Founded in 2008, Abacus is an innovative, award-winning IT and Cybersecurity managed service provider offering an enterprise integrated platform specifically designed for the unique needs of the financial & healthcare services industry. Abacus's Red Team has deep industry expertise that provides high-quality security testing and continuous monitoring which is tailored to meet the unique needs of financial & healthcare services firms against evolving threats and compliance demands.



Appendix A - OWASP Web Application Category Definitions

A01:2025 – Broken Access Control

Access control enforces policy to prevent users from acting outside of their intended permissions. Failures typically lead to unauthorized information disclosure, information modification, destruction of all data, or performing a business function outside of the user's permissions.

A02:2025 – Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is often a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

A03:2025 – Software Supply Chain Failures

Software supply chain failures are breakdowns or compromises in the process of building, distributing, or updating software, often caused by vulnerabilities or malicious changes in third-party code, tools, or dependencies. This category is an expansion of the prior "Vulnerable and Outdated Components".

A04:2025 – Cryptographic Failures

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest, or in transit, and requires special precautions when exchanged with the browser.

A05:2025 – Injection

An application is vulnerable to injection attacks when user-supplied data is not validated, filtered, or sanitized by the application, dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter, hostile data is used within object-relational mapping (ORM) search parameters, or when hostile data is directly used or concatenated.

A06:2025 – Insecure Design

Insecure design is a broad category that represents different types of weaknesses, expressed as "missing or ineffective control design." There is a difference between insecure design and insecure implementation. One of the factors that contributes to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus there is a failure to determine what level of security design is required.

A07:2025 – Authentication Failures

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws in order to assume other users' identities temporarily or permanently. This category encompasses weaknesses in multi-factor authentication, session handling, credential storage, and federated identity flows.

A08:2025 – Software or Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is a CI/CD pipeline that doesn't verify integrity, or where an application relies upon plugins, libraries, and modules from untrusted sources, repositories, and content delivery networks (CDNs).

A09:2025 – Security Logging and Alerting Failures

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show that it takes on average over 200 days to detect a breach, and the breach is typically detected by external parties rather than internal processes or monitoring.

A10:2025 – Mishandling of Exceptional Conditions

Mishandling of exceptional conditions encompasses weaknesses arising from improper error handling, logical errors, fail-open behavior, and other vulnerabilities that manifest when systems encounter abnormal states. Exploitation of these conditions can result in sensitive data exposure, denial-of-service conditions, or unintended access granted through insecure failure modes. Secure applications must be designed not only to resist attack, but to fail safely and predictably under adverse conditions.