

Web Application Penetration Test Report

PREPARED 4/25/2023



Prepared For:

Michael Scott

Dunder Mifflin:

1725 Slough Avenue in Scranton, PA.

Table of Contents

1. Summary of Changes	1
3. How to Use This Report	2
4. Executive Summary	3
5. Positive Observations	4
6. Key Recommendations	4
7. Scope	4
8. Primary Findings	5
8.1 DM-01 – Stored Cross-Site Scripting	6
8.2 DM-02 – Privilege Escalation	8
8.3 DM-03 – Tokens Stored in Local Storage	10
8.4 DM-04 – Missing HttpOnly Flag on Cookie	11
8.5 DM-05 – Missing Secure Flag on Cookie	12
8.6 DM-06 – Username Enumeration	13
8.7 DM-07 – HTML Injection	14
8.8 DM-08 – Deprecated TLS Implementation	15
8.9 DM-09 – Weak Password Requirements	16
8.10 DM-10 – Information Disclosure	18
9. CVSS v3.0 Reference Table	20
10. Acronym Reference Chart	20
11. Tools Used	21

1. Summary of Changes

Change Information	Reason for Change	Date
Version 1.0	Initial Release	4/22/2023

Table 1: Summary of Changes

3. How to Use This Report

This document was prepared in accordance with cybersecurity best practices. Remediation recommendations, where applicable, are found within each related section. These recommendations specifically address the security concerns discussed in the respective sections. Wherever possible, screenshots, code samples, or similar documentation is included to demonstrate methods and findings encountered during the period of performance of this test.

This report represents a point-in-time snapshot of assets that underwent testing. In accordance with cybersecurity best practices, regular security assessments should be commissioned, especially after major changes to application source code or infrastructure. This report should not be considered absolute in nature as restrictions on time, economics, and resources contribute to a limited perspective that an assessment of this type provides.

4. Executive Summary

From April 6th through April 20th, 2023, SecureTrust performed a penetration test of the Dunder Mifflin web application. The tools used to conduct the penetration test consisted of publicly available tools combined with commercial and proprietary software. For this project, both manual and automated testing methods were used to detect and/or validate the existence of exploitable vulnerabilities.

The objective of this penetration test was to compromise and/or otherwise gain unauthorized access to all resources located within the site. SecureTrust Security uses the Web Security Testing Guide methodology for web application penetration testing. This framework ensures that the application receives full, comprehensive coverage during testing.

The testing efforts resulted in a total of two high, five medium, and two low severity findings - nine in total. A stored cross-site scripting was found that would allow an authenticated attacker to inject malicious JavaScript into the application which will be executed each time an affected page is rendered.

Additionally, a privilege escalation vulnerability was found that would allow a user with the ability to edit roles of arbitrary users to escalate their privileges to SuperAdmin – functionality which appears to be hidden from the front end of the application.

Overall, the remediation efforts should require minimal effort. For example, several medium findings require simple configuration changes or updates. However, remediation for the two high findings will likely require edits to the application's front and back-end code.

The team employed a variety of advanced testing techniques, adhering to industry-standard methodologies such as the OWASP Top Ten. The assessment focused on critical areas including input validation, authentication and authorization mechanisms, session management, and business logic vulnerabilities. Particular attention was given to testing for common web application vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

5. Positive Observations

During the penetration test of Dunder Mifflin's web application, several positive observations were noted. The application's error handling procedures were commendable, displaying generic error messages instead of verbose output, thus preventing potential information leakage. The use of up-to-date and secure third-party components further fortified the application against vulnerabilities associated with outdated libraries.

The inclusion of comprehensive logging and monitoring mechanisms was another significant win, enabling the timely detection and response to potential security incidents. Lastly, the presence of a well-documented API with clear security guidelines showcased the team's thorough understanding of API security best practices. These positive aspects not only demonstrate the application's resilience against common web threats but also highlight the proactive approach taken by the development and security teams in maintaining a secure and robust web environment.

6. Key Recommendations

From the comprehensive penetration test conducted on the Dunder Mifflin web application, several crucial recommendations have emerged. A primary concern is the stored Cross-Site Scripting vulnerability, which necessitates stringent input filtering and output encoding to prevent malicious script injections. The discovery of privilege escalation vulnerabilities highlights the need for enhanced backend API security to restrict unauthorized role modifications. Addressing tokens stored in local storage is critical, with a shift recommended towards storing sensitive tokens in HttpOnly and Secure cookies. Furthermore, the application's missing HttpOnly and Secure flags on cookies demand immediate attention to safeguard against client-side script access and data interception over unsecured connections. The identification of username enumeration risks calls for the implementation of generic error messaging and the removal of specific error details from server responses. HTML injection vulnerabilities can be mitigated by applying rigorous input validation and encoding. The deprecated TLS implementation poses a significant security risk, necessitating an upgrade to TLS 1.2 or higher. Weak password requirements currently undermine the application's security, and a revision in line with NIST best practices is recommended to enhance password strength. Finally, the exposure of versioning information in server headers and error messages should be minimized to reduce the risk of targeted attacks.

7. Scope

URL(s)	https://dundermifflin.com
--------	---------------------------

8. Primary Findings

Below is a table of the finding with their associated severities.

ID	Finding	Severity
DM-01	Stored Cross-Site Scripting	High
DM-02	Privilege Escalation	High
DM-03	Tokens Stored in Local Storage	Medium
DM-04	Http Only Flag Missing	Medium
DM-05	SSL Flag Missing	Medium
DM-06	Username Enumeration	Medium
DM-07	HTML Injection	Medium
DM-08	Deprecated TLS Version	Low
DM-09	Weak Password Requirements	Low
DM-10	Information Disclosure	Informational

Table 2: Table of findings.

8.1 DM-01 – Stored Cross-Site Scripting

Current Rating	CVSS
HIGH	8.1

Description of Finding:

A stored Cross-Site Scripting vulnerability was discovered during testing as an authenticated user. A stored XSS is the most damaging of type of XSS. It occurs when a malicious script is injected directly into an application where it persists until it is manually removed.

To demonstrate the severity, the payload used, forced the SecureTrust testers' browser to pop an alert box and display cookie information. This can also be used to redirect attackers to a malicious page in an attempt to steal session information.

The SecureTrust tester injected the payload `` into the proposition name field within the Home/Propositions resource.

Home / Propositions / ``

`` Back

Details Requirements Contests/Elections Attachment

Status / Relations Edit

Proposition ID: 23 Status: Active Residency:

Name Edit

Proposition Name 1: `` Proposition Name 2:

Figure 1- Editing proposition name with malicious payload

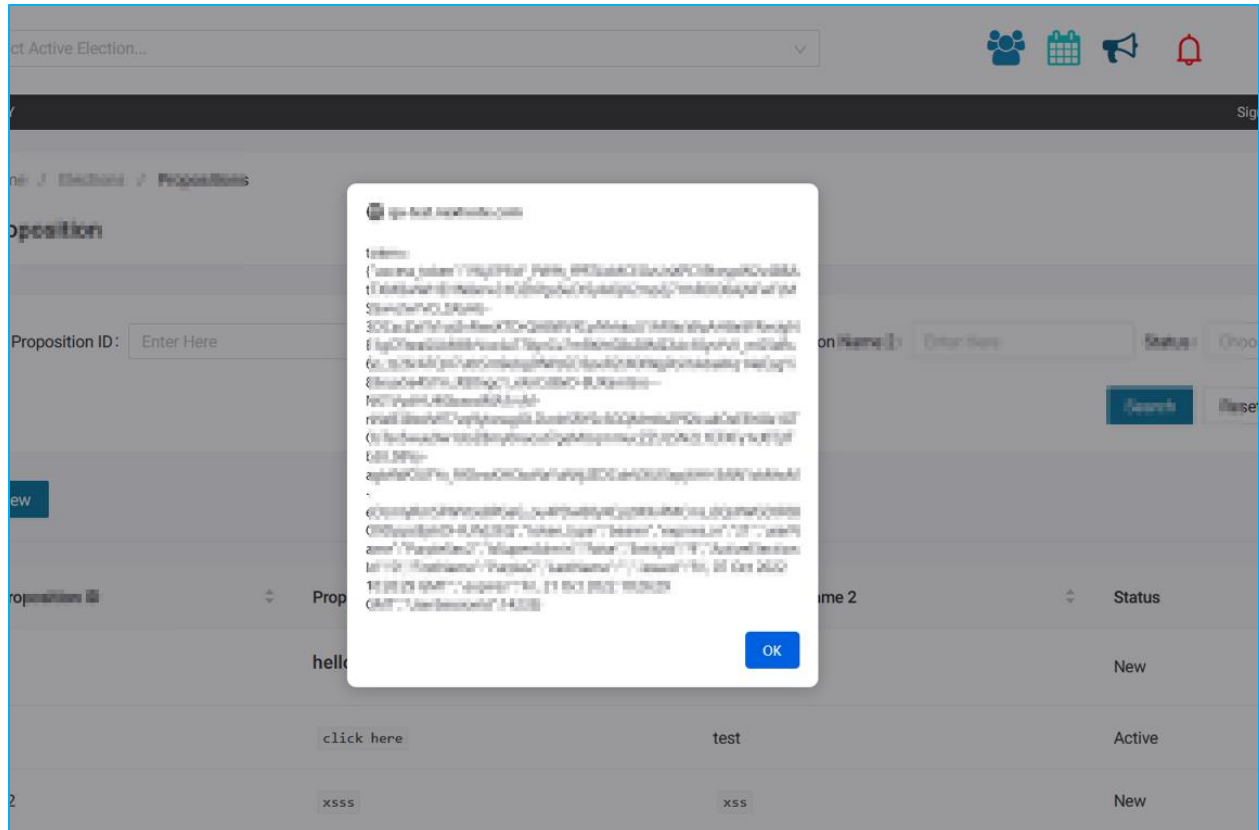


Figure 2 - Application executing JavaScript payload

Recommendation:

Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

8.2 DM-02 – Privilege Escalation

Current Rating	CVSS
HIGH	8.8

Description of Finding:

A user with permissions to edit roles may modify their request to add the SuperAdmin role to a targeted user. Normally, the SuperAdmin role is unable to be modified via the front end of the application.

In the following request, the test user was escalated to a SuperAdmin via a modified POST request to the `/api/User/SaveRoles` endpoint.

```

POST /api/User/SaveRoles HTTP/2
Host: ip:192.168.1.100:8080
Content-Length: 539
Sec-Ch-Ua: "Not:A=Brand";v="99", "Chromium";v="106"
EntityId: 9
Sec-Ch-Ua-Mobile: 70
...
{"User ID": 10,
"Role ID": 1,
"Role Name": "SuperAdmin",
"Checked": true
}

1 HTTP/2 200 OK
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Content-Length: 103
5 Content-Type: application/json; charset=utf-8
6 Expires: -1
7 Server: Microsoft-IIS/10.0
8 X-AspNet-Version: 4.0.30319
9 X-Powered-By: ASP.NET
10 Access-Control-Allow-Origin: *
11 Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
12 Access-Control-Allow-Headers: *
13 Date: Sat, 15 Oct 2022 12:58:16 GMT
14
15 {
  "Result": true,
  "ResultCode": 1000,
  "Data": "",
  "Message": "SUCCESS",
  "ResultCodeStr": "Success",
  "SourceId": ""
}

```

Figure 3 - Malicious Request modifying user roles.

The modification consisted of adding the following data to the JSON body of the request.

```
{
  "UserId": 68,
  "RoleId": 1,
  "RoleName": "SuperAdmin",
  "Checked": true
}
```

Figure 4 - Added JSON data

This assigned the SuperAdmin role to the test user, enabling the user to login to the SuperAdmin panel.

The screenshot shows the SuperAdmin panel interface. At the top right, there is a "Sign Out" button. Below the header, there are two lines of red text indicating a warning about user permissions. The main content area is divided into several sections:

- To Install new:** A list of steps with corresponding buttons:
 - Step 1: Init Settings (Init Settings)
 - Step 2a: Add/Configure Any Users & Roles via api
 - Step 3: Update Menu and Pre-configured Roles Step 2b: Update Super Admin Roles Linking (Reset Menus)
 - Step 4: Update Language Settings: SQL Configuration (Update Language Configuration)
 - Step 5: Update System Settings: SQL Configuration (Update System Settings via SQL Scripts)
 - Step 6: Update System Settings: VoterStatus, Voter Status Reason, Template Configuration (Update System Settings)
- To Update the System after deployment:** A list of steps with corresponding buttons:
 - Step 1: Add/Configure Users & Roles....
 - Step 2: Update Language Settings: SQL Configuration (Update Language Configuration)
 - Step 3: Update System Settings: SQL Configuration (Update System Settings via SQL Scripts)
 - Step 4: Update System Settings: VoterStatus, Voter Status Reason, Template Configuration (Update System Settings)
 - Step 5: Update Menu (Update Menu)
 - Step 6: Update Super Admin Roles Linking (Super Admin Roles Linking)
 - Step 7: Update Templates Only (Optional if not running Step 4) Update Templates only
- To update store procedures changed:** (Update Store Procedures)
- To update function operation list:** (Reset Menus, Reset Users)
- County to reset data:** A dropdown menu set to "Chenango" with buttons for (Reset Language, Reset Settings, Reset Data Source, Refresh Voter Address GIS Layer Data).
- Other buttons:** (Update Menu, Update Administrator Menu, Refresh Table Index, Clear Cache Data, WEB Prototype, API Help Document »).

Figure 5 - User logged into SuperAdmin panel

Recommendation:

Implement a back-end API fix to validate user's authorization to add SuperAdmin role.

8.3 DM-03 – Tokens Stored in Local Storage

Current Rating	CVSS
MEDIUM	4.3

Description of Finding:

Sensitive session information stored in HTML5 local storage is vulnerable to XSS. Session tokens, IDs, etc. stored as cookies with the HttpOnly flag are a more secure method of storage.

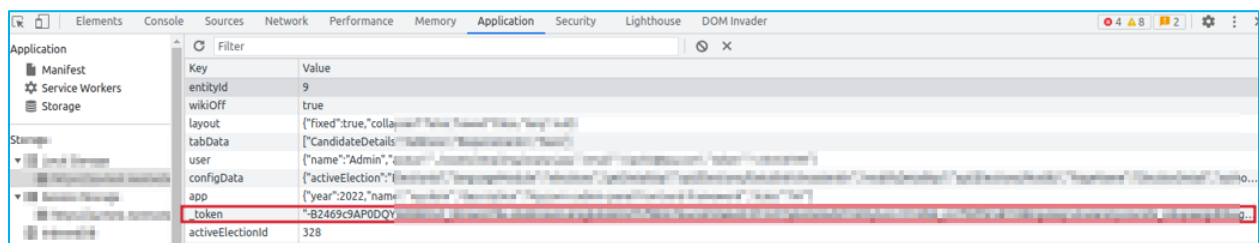


Figure 6 - "token" stored in local storage

Recommendation:

Store sensitive tokens as strictly cookies with the necessary security flags instead of in local storage.

8.4 DM-04 – Missing HttpOnly Flag on Cookie

Current Rating	CVSS
MEDIUM	4.3

Description of Finding:

The tester observed the HttpOnly flag to be missing from the "token" cookie. HttpOnly is an additional flag included in a Set-Cookie HTTP response header. If supported by the browser, using the HttpOnly flag when generating a cookie helps mitigate the risk of client-side script accessing the protected cookie. If a browser that supports HttpOnly detects a cookie containing the HttpOnly flag, and client-side script code attempts to read the cookie, the browser returns an empty string as the result. This causes the attack to fail by preventing the malicious (usually XSS) code from sending the data to an attacker's website.

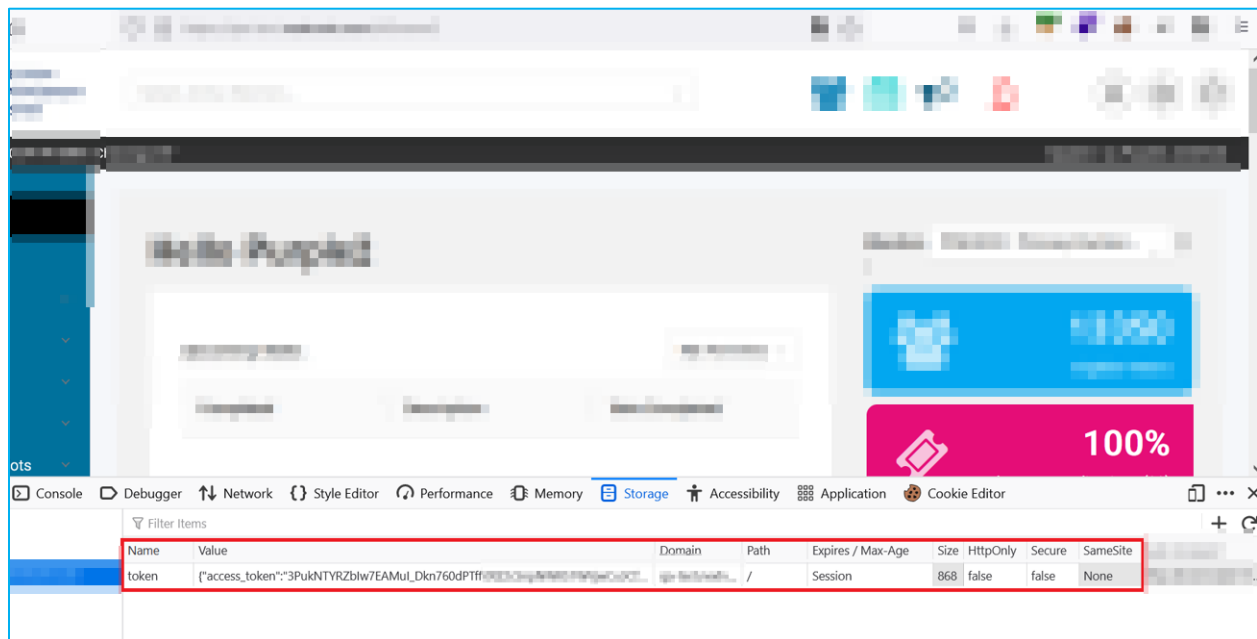


Figure 7 - "token" missing HttpOnly Flag

8.6 DM-06 – Username Enumeration

Current Rating	CVSS
MEDIUM	5.3

Description of Finding:

Upon inspecting authentication logic at <https://duntermifflin.com/login>, the tester identified an error message in the response from the server. The error stated "USERNOTEXIST".

However, when the correct username was provided, the application returned a message that the password was incorrect, implying the username was accurate. This verbose error messaging could allow an attacker to utilize a tool or custom script to enumerate valid usernames.

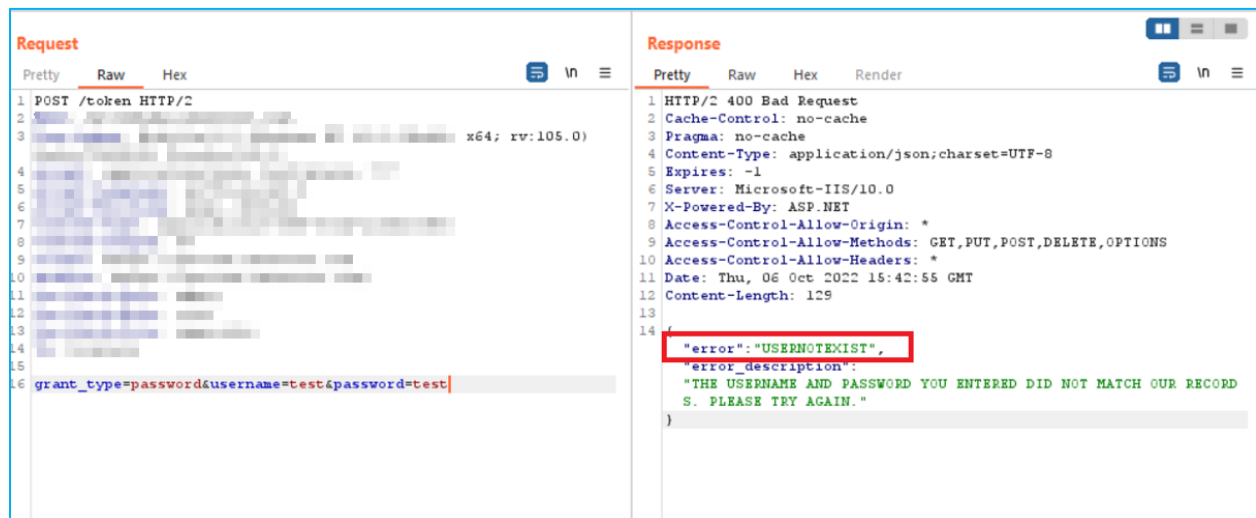


Figure 9 - Server responds with "USERDONOTEXIST" when attempting to authenticate as an invalid user

Recommendation:

Consider using a generic error message stating "Either the username OR password did not match our records. Please try again." Additionally, remove the JSON error message in the server response.

8.7 DM-07 – HTML Injection

Current Rating	CVSS
MEDIUM	4.3

Description of Finding:

Similar to Cross-site Scripting, an HTML injection happens when the payload supplied by the malicious user as part of untrusted input is executed client-side by the web browser as part of the HTML code of the web application. Stored HTML injection, the payload is stored by the web server and delivered later potentially to multiple users. An attacker may inject malicious HTML that aims to harm the reputation of the page, for example, for political or personal reasons.

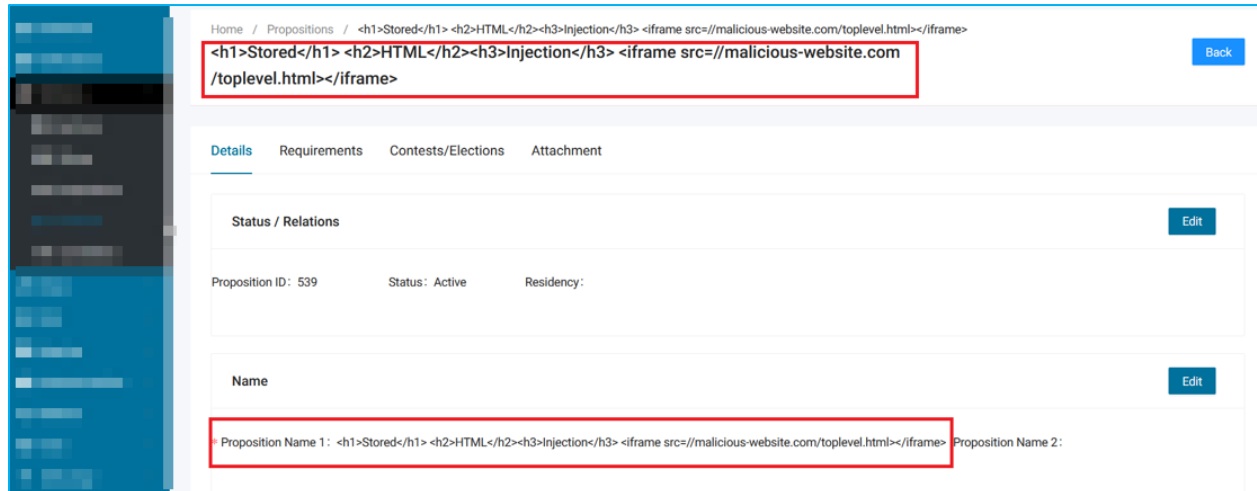


Figure 10 - Malicious HTML stored server side as the "Proposition Name"

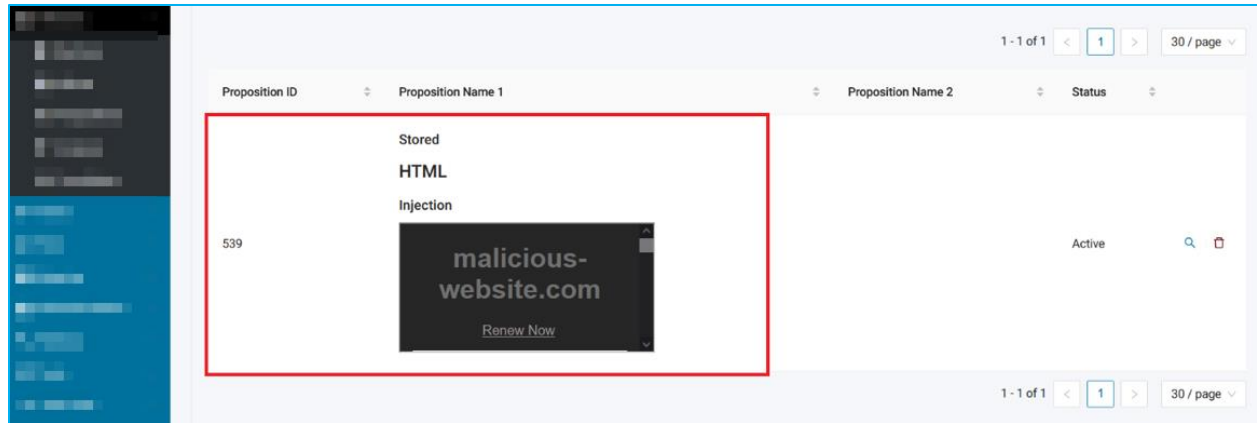


Figure 11 - Malicious HTML rendered in the victim browser including a link to an external website

Recommendation:

Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

8.8 DM-08 – Deprecated TLS Implementation

Current Rating	CVSS
LOW	2.7

Description of Finding:

The web server at dundermifflin.com supports encryption through TLS 1.0, which was formally deprecated in March 2021 due to inherent security issues. In addition, TLS 1.0 is not considered to be "strong cryptography" as defined and required by the PCI Data Security Standard 3.2.(1) when used to protect sensitive information transferred to or from web sites. According to PCI, "30 June 2018 is the deadline for disabling SSL/early TLS and implementing a more secure encryption protocol – TLS 1.2 or higher.

```

Testing SSL server [redacted] on port 443 using SNI name [redacted]

SSL/TLS Protocols:
SSLv2    disabled
SSLv3    disabled
TLSv1.0  enabled
TLSv1.1  enabled
TLSv1.2  enabled
TLSv1.3  disabled

TLS Fallback SCSV:
Server does not support TLS Fallback SCSV

TLS renegotiation:
Secure session renegotiation supported

TLS Compression:
Compression disabled

Heartbleed:
TLSv1.2 not vulnerable to heartbleed

```

Figure 16 - TLSv1.0 & TLSv1.1 enabled on <https://dundermifflin.com>

Recommendation:

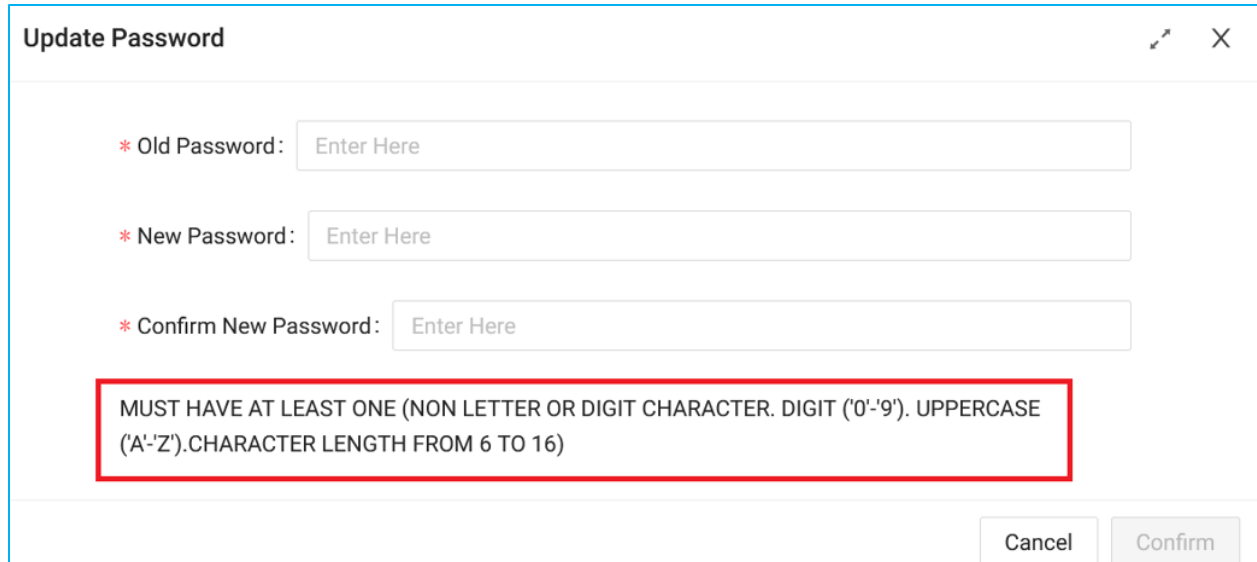
Disable TLS 1.0, TLS 1.1 and replace it with TLS 1.2 or higher.

8.9 DM-09 – Weak Password Requirements

Current Rating	CVSS
LOW	2.7

Description of Finding:

The testing team found the application to be utilizing weak password requirements for its user accounts. The application allows dictionary words that are easily guessable, additionally the application requires a minimum of six (6) characters. This may allow an attacker to brute force, credential stuff, or guess a correct password and gain access to the application



Update Password

* Old Password: Enter Here

* New Password: Enter Here

* Confirm New Password: Enter Here

MUST HAVE AT LEAST ONE (NON LETTER OR DIGIT CHARACTER. DIGIT ('0'-'9'). UPPERCASE ('A'-'Z'). CHARACTER LENGTH FROM 6 TO 16)

Cancel Confirm

Figure 13 - Password requirements of the application, requiring 6 to 16 characters

Recommendation:

NIST has recommendations that are not strict requirements but are seen as best practices.

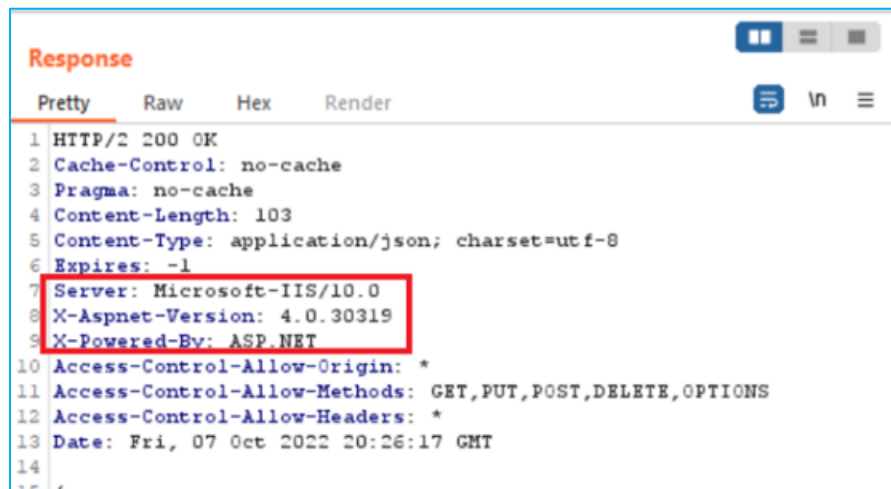
- User-generated passwords should be at least eight characters in length
 - SecureTrust Security recommends thirteen characters.
- Users should be able to create passwords up to at least sixty-four characters
- Prospective passwords should be compared against password breach databases and rejected if there is a match
- Users should be allowed ten failed password attempts before being locked out of a system or service

8.10 DM-10 – Information Disclosure

Current Rating	CVSS
Informational	0.0

Description of Finding:

SecureTrust identified several instances where versioning information is publicly exposed and accessible. Although no vulnerabilities were identified, an attacker can use this information to mount and chain several attacks together. Additionally, if a new vulnerability is discovered with the affected version, this can also be used by an attacker.



```
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Content-Length: 103
5 Content-Type: application/json; charset=utf-8
6 Expires: -1
7 Server: Microsoft-IIS/10.0
8 X-AspNet-Version: 4.0.30319
9 X-Powered-By: ASP.NET
10 Access-Control-Allow-Origin: *
11 Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
12 Access-Control-Allow-Headers: *
13 Date: Fri, 07 Oct 2022 20:26:17 GMT
14
15 /
```

Figure 14 - Response headers containing version information

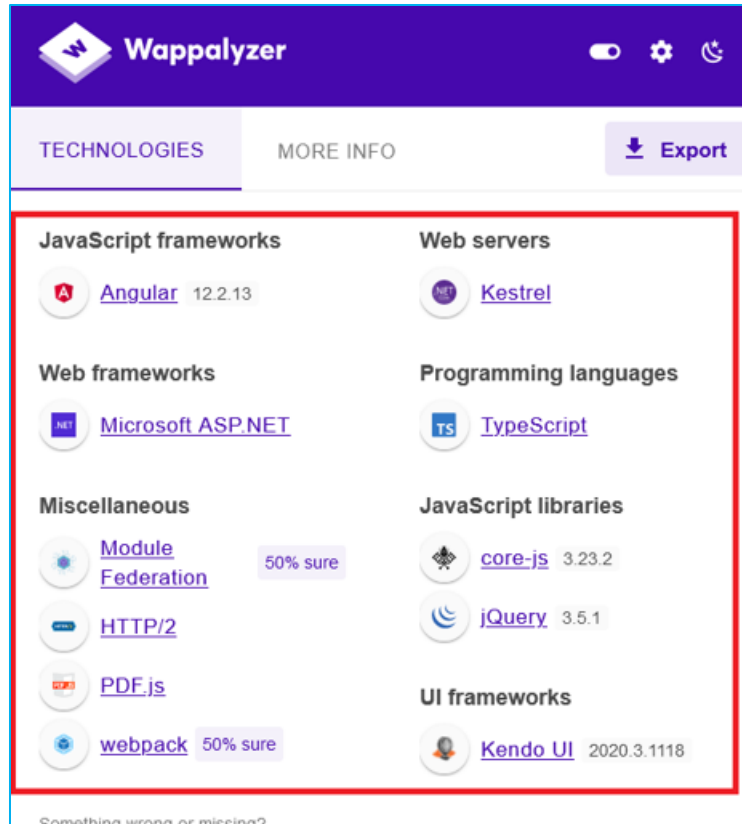


Figure 15 - Browser Plugin detecting framework & library versioning information

Recommendation:

Remove version and banner information from headers. Use a generic error message when a user ends up with a 403 or a 404, this will help avoid displaying version information.

9. CVSS v3.0 Reference Table

Qualitative Rating	CVSS Score
None/Informational	N/A
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

Table 3 : [Common Vulnerability Scoring System Version 3.0](#)

10. Acronym Reference Chart

CVSS	Common Vulnerability Scoring System
OWASP	Open Web Application Security Project
NIST	National Institute of Standards and Technology
CVE	Common Vulnerabilities and Exposures
SDLC	Software Development Life Cycle
XSS	Cross Site Scripting
IDOR	Insecure Direct Object Reference
WAF	Web Application Firewall
IoT	Internet of Things
HTML	HyperText Markup Language
UUID	Universally Unique Identifier
DOS	Denial of Service

Table 7: *Common acronyms.*

11. Tools Used

Tool	Description
Burp Suite	An integrated platform for performing security testing of web applications. Offers a range of tools for scanning, proxying, and analyzing web traffic.
Nikto	An open-source web server scanner which performs comprehensive tests against web servers for multiple items, including potentially dangerous files and programs.
Nmap	While primarily used for network discovery and security auditing, Nmap can be used to scan web servers and applications for vulnerabilities.
SQLmap	An open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws in web applications.
ffuf	Designed for fast web content discovery, a process commonly referred to as fuzzing.
Postman	A powerful tool for building, testing, documenting, and sharing APIs