Pentest Report: [www.example.com]

Date 2022-03-20

Prepared By Joseph Lee

Ripple Software Consulting LLC

Email joseph@ripplesoftware.ca

Mobile 778-725-3206

CONFIDENTIALITY

Ripple Software Consulting

465 Dundas St. W

Toronto, Ontario Canada

Attention: Contracts Manager

DISCLAIMERS

[Table Of Contents

[1. Purpose

The purpose of the penetration test was to create evidence that can be used for SOC-2 compliance. This includes the framework used for the penetration test, and the report generated by implementing the penetration testing framework. This document *'Pentest Report: [www.example.com]'* represents the report generated by the framework. The original framework used can be found as the document titled *'Penetration Testing Framework'*.

[2. Scope

The scope for the pentest was identified as the subdomain *'[www.example.com]'* and the web-application hosted at that subdomain. The subdomain *'[www.example.com]'* will hereby be referred to as the **target domain**. The hosted web-application will hereby be referred to as the **target application**.

The goal of the test is to prioritize the identification of OWASP Top Ten vulnerabilities, external service vulnerabilities, and identify any behavior on the target application that does not comply with industry best practices according to NIST Cyber Security Framework (CSF).

[3. Summary of Findings

In performing a detailed application penetration study against the target domain and target application several severe vulnerabilities were identified. Throughout this report we provide brief descriptions of each testing category and provide more detailed where findings were negative.

Table 1 below table shows a list of all files and external links used in this report and a description of the file contents.

[www.example.com]-202203182222.xlsx	Report of DNS information generated from DNSDumpster.com
report-cc3aa081-53ce-447d-b1d9-4277d384b894.pdf	OpenVas scan of IP addresses [REDACTED] and 104.21.36.49.
www.example.com.xml	XML formatted output of Nmap intense scan of IP addresses [REDACTED] and 104.21.36.49.
https://spyse.com/search?target=domain&query=[www.example.com]%20	SPYSE report for scan of the domain target domain.
domain-map-[www.example.com].png	A DNS map image provided by DNSDumpster
subdomain-map-example.com.png	A subdomain to IP address map provided by DNSDumpster
paywall-js-exposed-[www.example.com].png	An image of HTML source code served from the homepage
angularowasptop10.pdf	An advisory on mitigation of OWASP Top Ten vulnerabilities in AngularJS.
OWASPLondon20170727_AngularJS.pdf	An advisory on mitigation of OWASP Top Ten vulnerabilities in AngularJS.

Table 1: List of all files provided with the pentest report

Table 2 below shows a breakdown of the vulnerabilities identified based on category and severity of risk. This table is followed by a detailed breakdown outlining each category. In the table below, a vulnerability listed under 'Pending' has been reported, where a vulnerability listed under 'Fixed', is a vulnerability that has been satisfactorily mitigated.

Unencrypted connections are enabled on all target website pages	High	Allowing unencrypted connections exposes sensitive data such as user credentials and all other information accessed or submitted to the website to MITM attackers
Weak policy for password requirements allows passwords to have very weak keyspace	High	This allows users to set weak passwords that can be easily brute-forced by an attacker. NIST advisory can be read here.
Weak registration implementation allows blocking email addresses from ability to register	High	This allows an attacker to block legitimate users from registering an account on the site

Certificate issued is wildcard	Medium	This allows a MITM attacker to spoof any subdomain with a stolen certificate . The risks can be read here.
HTTP Strict Transport Security (HSTS) header is not configured	Medium	This target application is not protect against MITM attacks such as protocol downgrade attacks and cookie hijacking
Specifying port 80, 8080, and 8334 from a browser results in error pages	Medium	Accessing these ports directly should result in a forward of the request to port 443. This is a cosmetic or convenience issue, although further penetration testing should be done to identify the extent that services on these ports is unavailable.
Support for TLS 1.0 and 1.1	Low	TLS 1.0 and 1.1 are associated with exploits that can allow an attacker to steal information
Support for weak encryption ciphers	Low	Weak ciphers represent a higher risk of connections to the server being decrypted
Web server is not configured to properly process URL requests that contain an explicit port	Low	While this vulnerability is mostly cosmetic it should be addressed in the case that specific ports are included in the URL that may allow a user who is provided with such a URL to believe the website is not working properly
Missing HTTP security headers	Low	Missing X-Content-Type-Options Header , Anti-clickjacking Header, and missing cache-control header. This is a weak HTTP header configuration, and these headers should be added and configured properly.

Table 2: List of all discovered vulnerabilities

[3.1 DNS Information

DNS information was collected from the starting point of the scope, which was to pentest the target domain. The item by item results of information discovered are shown below.

The **nslookup** tool was used to identify the IP addresses resolved to by the target domain. Two IPv4 IP addresses were identified using this method. The two identified IP addresses were **[REDACTED]** and **[REDACTED]** The command output is shown below:

```
$ nslookup [www.example.com]
Non-authoritative answer:
Name:[www.example.com]
Address: [REDACTED]
Name:[www.example.com]
Address: [REDACTED]
```

The whois tool was used to collect ownership information these two identified IP addresses. Identical ownership information was found. The IP addresses were identified as being owned by **Cloudflare Inc**. The truncated command output is shown below.

```
$ whois [REDACTED][REDACTED]
[truncated...]
OrgName:        Cloudflare, Inc.
OrgId:      CLOUD14
Address:        101 Townsend Street
City:       San Francisco
StateProv:      CA
PostalCode:     94107
Country:        US
RegDate:        2010-07-09
Updated:        2021-07-01
Ref:        https://rdap.arin.net/registry/entity/CLOUD14
[truncated...]
```

DNS Zone information was collected from public OSINT web-application tools **SPYSE** and **DNSDumpter**. Links to the respective reports are available: [SPYSE report,]() and DNSDumpster report. Additional information was provided by these sources.

The **SPYSE** report identified some historical DNS information that was scanned on 2021-08-10 including two IP addresses **[REDACTED]** and **[REDACTED]** and nameserver records **ns1.brandbucket.com** and **ns2.brandbucket.com**. These IP addresses were identified as Amazon AES servers. Since ownership of these domains cannot be attributed or associated within the scope of the pentest, they were not scanned for further information.

The **DNSDumpster** report identified additional DNS records such as the current nameserver and MX records, and a list of additional subdomains associated with the base domain for the target domain. The map of subdomains is available as an image in the file *'subdomain-map-example.com.png'*, a map of the domain is available as an image in the file *'domain-map-[www.example.com].png'*. and the list of subdomains as a spreadsheet is included in the file *'[www.example.com]-202203182222.xlsx'*. Since the identified MX records and subdomains were not included in the scope of the penetration test, no additional information was gathered about them.

[3.2 TLS/SSL Certificates

The Qualys SSL Labs online tool was used to collect SSL/TLS certificate information about the target domain. The results included the identification of two IPv6 addresses. Those IPv6 addresses were **[REDACTED]** and **[REDACTED]**. Both the previously identified IPv4 address and identified IPv6 addresses received an overall score of B from the Qualys SSL Labs analysis. The original results can be read from the Qualys SSL Labs report. A summary of the findings are included below.

```
This provided certificates support TLS 1.0 and TLS 1.1.
    ●   Advisory - https://www.rfc-editor.org/info/rfc7525
Weak TLS 1.2 ciphers are supported
    ●   TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
    ●   TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
    ●   TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
● TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
```

Known attacks against TLS 1.0 and TLS 1.1 include POODLE, Vulnerabilities associated with TLS 1.0 and TLS 1.1 are identified in CVEs.

The **openssl** tool was used to identify particular TLS certificate information. It was observed that the TLS certificate served by the target domain was a wildcard certificate. Wildcard certificates do not provide optimal security since if a certificate is stolen by an attacker, they can use the certificate to enable HTTPS on an arbitrary subdomain related to your base domain. This essentially allows an attacker to spoof services under the guise of your company. The output from the openssl scan is provided below.

```
$ openssl s_client -showcerts -servername [www.example.com] -connect [www.example.com]:443 </dev/null

CONNECTED(00000006)
depth=3 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X2
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = E1
verify return:1
depth=0 CN = *.[www.example.com]
verify return:1
---
Certificate chain
 0 s:CN = *.[www.example.com]
   i:C = US, O = Let's Encrypt, CN = E1
[Truncated...]
---
Server certificate
subject=CN = *.[www.example.com]
issuer=C = US, O = Let's Encrypt, CN = E1
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: ECDSA
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 4513 bytes and written 409 bytes
Verification error: unable to get local issuer certificate
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 256 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 20 (unable to get local issuer certificate)
---

DONE
```

An additional error was given by the openssl tool when requesting certificate information for the target domain: **verify error:num=20:unable to get local issuer certificate.**

Remediation of these security risks includes disabling the identified vulnerable SSL/TLS protocols and weak ciphers on the web-server configuration. However, the risk is considered low.

[3.3 HTTP Headers

The **curl** tool was used to collect HTTP headers from the request to the target domain. This produced a list of HTTP headers (cookies) that were exchanged during a connection made requesting the base target domain. The output is shown below.

Each header cookie was identified and Google search for information about each header cookie was collected. The table below shows the header cookies for which additional information provided security related insight.

CF-RAY	• https://lab.wallarm.com/cloudflare-fixed-an-http-2-smuggling-vulnerability/ • https://hackerone.com/reports/97292
CF-Cache-Status	
X-Amz-Cf-Pop header	
X-Amz-Cf-Id header	

The reported vulnerability in the CF-RAY header resulted in a HTTP/2 smuggling vulnerability. However, it was reported that Cloudflare has since patched against exploitation. Therefore this is not included in the list of discovered vulnerabilities.

[3.4 External Scan

An external scan of all identified IPv4 and IPv6 addresses was conducted to identify all open ports and services, and potential vulnerabilities in the public facing servers. The result for each IP address scanned was identical.

[3.4.1 Nmap Scan

The nmap tool was used to scan the public facing IP addresses for any open ports and associated services. The scan identified that the only open ports were **80**, **443**, **8080**, **8443**.

Each of these open ports were tested for HTTP service using. The results of the nmap scan are included below.

```
$ nmap -T4 -A -v [REDACTED]
[truncated...]
Initiating Connect Scan at 18:19
Scanning [REDACTED] [1000 ports]
Discovered open port 8080/tcp on [REDACTED]
Discovered open port 443/tcp on [REDACTED]
Discovered open port 80/tcp on [REDACTED]
Discovered open port 8443/tcp on [REDACTED]
Completed Connect Scan at 18:20, 41.52s elapsed (1000 total ports)
Initiating Service scan at 18:20
Scanning 4 services on [REDACTED]
Completed Service scan at 18:20, 6.09s elapsed (4 services on 1 host)
[truncated...]
 Host is up (0.024s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE      VERSION
80/tcp   open  http              Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
443/tcp  open  ssl/https         cloudflare
|_http-server-header: cloudflare
|_http-title: 400 The plain HTTP request was sent to HTTPS port
8080/tcp open  http              Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
8443/tcp open  ssl/https-alt cloudflare
|_http-server-header: cloudflare
|_http-title: 400 The plain HTTP request was sent to HTTPS port
[truncated...]
```

[3.4.2 Verify Secure HTTPS Connections

In order to verify the results of the scan, each port was tested on the target domain with Firefox browser for MacOS version 98.0.1 (64-bit) and Chrome browser Version 99.0.4844.74. The results were verified by the browser used.

When using Firefox, using the scheme http:// allowed the browser to successfully forward traffic to the https:// scheme. However, it was determined that services were not configured properly to forward requests that specified the port explicitly in the url. URL requests that specified port 80, 8080, and 8334 all failed to be upgraded to HTTPS connections. Screenshots of the attempts to connect to the website on those specific available ports are shown below in Figures 1-3.

[IMAGES REDACTED]

The error reported by the attempts to connect via browser to port 80 and 8080 resulted in the error **SSL_ERROR_RX_RECORD_TOO_LONG** to be reported.

Using the Chrome browser, requests using an **http://** scheme were allowed to establish HTTP connections, view, and interact with the website. They were not forwarded and upgraded to **https://** schemes or use HTTPS connections. This vulnerability allows a MITM attacker placed anywhere on the route to see all communications with the website in plaintext unencrypted form. Sensitive information such as user credentials or other data contained on or submitted to the website can be collected by an attacker.

Furthermore, when specifying the port explicitly in a URL request using the Chrome browser, port 80 was able to show the page successfully, although the connection was not upgraded from HTTP to HTTPS. Explicitly specifying port 8080 resulted in an error page being displayed, which can be seen in Figure 4, and similarly requests that explicitly specified port 8443 resulted in a 400 Bad Request error and error page being displayed, which can be seen in Figure 5.

[IMAGES REDACTED]

Using the Safari browser, on MacOS resulted in identical results as using Chrome browser on MacOS.

HTTP Connections Are Allowed

It was determined that Chrome allows HTTP connections. Mitigation of this vulnerability is critical. The target web-server should be correctly configured to forward all HTTP requests reliably to an HTTPS connection. Failure to mitigate this vulnerability can result in MITM attackers being able to steal sensitive information such as credentials and all other connection data. The target domain is vulnerable to a downgrade attack that would allow connections to be downgraded by a MITM.

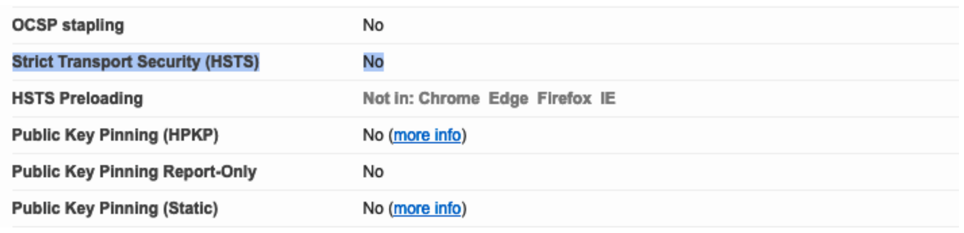
URLs With Explicit Port Declarations Resolve to Error Pages

It was also determined that Firefox and Chrome failed to forward URL requests that specified ports 80, 8080, and 8334 correctly. The server configuration should be updated to allow the correct forwarding when specifying the port explicitly.

HSTS Not Enabled

The status of the HSTS security header was examined and it was found that the server is not using an HSTS security header. HTTP Strict Transport Security (HSTS) is a policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks[1] and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should automatically interact with it using only HTTPS connections, which provide Transport Layer Security (TLS/SSL), unlike the insecure HTTP used alone. HSTS is an IETF standard track protocol and is specified in RFC 6797. Multiple methods were used to verify that the HSTS header was missing including using **curl** to check for the header specifically, checking the Qualys SSL Labs report, and using Chrome Dev Tools. The **curl** command shown below was used to check for the HSTS header and no output was returned. Also, an image of the Qualys SSL Labs result is shown below.

```
$ curl -s -D- https://[www.example.com] | grep -i Strict
```



OCSP stapling	No
Strict Transport Security (HSTS)	No
HSTS Preloading	Not in: Chrome Edge Firefox IE
Public Key Pinning (HPKP)	No (more info)
Public Key Pinning Report-Only	No
Public Key Pinning (Static)	No (more info)

Figure 6: Qualys SSL Labs Report showing HSTS not configured

Testing Server Configuration Post Mitigation

After the server configuration vulnerability has been mitigated all OS and browsers, and scripting language libraries that allow connections such as Python requests, and urllib2 and others should be tested to ensure proper and successful forwarding of all HTTP requests to HTTPS connections.

[3.4.3 OpenVAS Scan

The scan was done with Greenbone Security Assistant Version 21.4.3 on Kali Linux which uses the OpenVAS vulnerability feed. The report was identical for all IPv4 IP addresses found during the information gathering stage. A sample report is available in the file *'report-cc3aa081-53ce-447d-b1d9-4277d384b894.pdf'*. A summary of the report findings is shown below.

Although the OpenVAS scan did not uncover any vulnerabilities, there were several notes logged by the scan that should be described. Also, several additional open ports on the server were discovered by the scan. Those open ports include **2086, 2082, 2083, 2087, 2052, 2053, 2095, and 2096**.

[3.5 Application Scan

[3.5.1 OWASP ZAP

A scan of the target application from a logged in state was conducted using OWASP ZAP HUD. The identified alerts are listed below and explanations are expanded in the table below.

Ajax Spider

- Medium critical alert: Missing Anti-clickjacking Header (1)
- Low critical alert: Cross-Domain JavaScript Source File Inclusion (2)
- Low critical alert: X-Content-Type-Options Header Missing (1)
- Info critical alert: Information Disclosure - Suspicious Comments (1)
- Info critical alert: Re-examine Cache-control Directives (1)

Active Scan

- INFO: Suspicious comments on 17 files
- INFO Cross-Domain JavaScript Source File Inclusion (6)
- INFO Timestamp Disclosure - Unix (475)
- INFO X-Content-Type-Options Header Missing (28)

Missing Anti-clickjacking Header	
Cross-Domain JavaScript Source File Inclusion	
X-Content-Type-Options Header Missing	
Information Disclosure - Suspicious Comments	
Re-examine Cache-control Directives	
Suspicious comments on 17 files	
Cross-Domain JavaScript Source File Inclusion	
X-Content-Type-Options Header Missing	

[3.5.2 Access Security

An examination of the policy requirements for passwords on the target application were assessed and showed that the existing password policy requires a password of 6 or more characters, with one capital letter, one lowercase letter and one number. The result of this policy is **insufficient keyspace**. An explanation of the results are shown below.

- 6 characters not long enough (min of 8 better to have 10 or 12)
- Users can create weak password such as the ones below
    - Password1
    - Abc123
    - Pass11

During testing, it was also observed that the registration system is able to lock out an email address. By submitting an email address and password, the email address was effectively blocked from attempting to register before submitting a phone number for verification. This allows an attacker to attack the website registration by submitting a list of email addresses one at a time to the registration page, and effectively block those email addresses from being able to register. Depending on the duration of time that the email address is reserved and the target application releases the email address this is a low priority vulnerability.

However, the recommended remediation for this vulnerability is to not reserve the email address for an account until after a phone number has been submitted by the user and verified with an SMS message. Also, it would be advisable to still include a button or link that allows the registering user to request another verification email. This feature would prevent an attacker from being able to block an email address from registration.

In the case of attempting to brute force a user password, the target application fails to prevent this continuous high-rate submissions to the login page. This is a high priority vulnerability. Thus, in order to prevent robots from being able to submit forms, some form of robot detection could be used.

[3.5.3 Static HTML Code Analysis

The HTML code served by the target application was analyzed and several interesting things were found. The most critical of those findings was that JavaScript for a paywall service was included in the HTML page served at a not-logged-in state. In combination with other vulnerabilities such as XSS or XSRF, would allow an attacker to attempt to process payments from the website at a not-logged-in state. An image of the source code with the highlighted library is shown in the image below.

[IMAGE REDACTED]

The recommended mitigation for this vulnerability is to not include unrequired JavaScript libraries. The inclusion of JavaScript libraries should be configured to only include the libraries that are required for each page to function normally.

[3.5.4 Application Stack Libraries

The **Angular Inspector** tool was used to identify the software libraries used in the development of the target application. The tool was able to identify a list of JS libraries including their version. The libraries and their versions are shown below in Table 3.

JQuery 3.6.0	Current stable version
Modernizer 3.3.1	Modernizer is an old package. Angular Inspector reports that the site is using 3.3.1, However, that is 2 major versions behind. The last release was 3.12 but that was still in 2017. Maybe false positive. Reported as vulnerable: https://www.tenable.com/plugins/was/112381
Angular 8.2.14	Angular 8 was released on May 28, 2019. Current stable release version is 13.1.1. There are only 3 reported vulnerabilities which are all for version 1. No vulnerabilities disclosed for version 8. https://www.cvedetails.com/vendor/18512/Angularjs.html
https://js.stripe.com/v3/	This is current stable release version: https://stripe.com/docs/libraries

Table 3: JS Libraries detected on the homepage

[3.5.5 Burp Suite Proxy Fuzzing

The information revealed by the Burp Proxy is shown below.

- The API endpoint was revealed
    - **Host: api.example.com**

[3.5.6 SQL Injection

The tool sqlmap was used to conduct a SQL injection on the target application login page using POST request. The test included level=5 and risk=3. The test was unable to execute an SQL injection attack against the server.

[3.5.7 OWASP Top Ten

The OWASP Top Ten website vulnerabilities is the de facto list of items to check for in a web application penetration test. The OWASP Top Ten is updated every few years and has evolved significantly over the past decade. The most recent version was released in 2021. All items are shown below and vulnerabilities discovered in the target application have been added to each appropriate category.

A01:2021-Broken Access Control	
A02:2021-Cryptographic Failures	● Unencrypted connections are possible ● TLS 1.0 and TLS 1.1 are enabled ● Other weak ciphers are available
A03:2021-Injection	● XSS was not successful against the target application ● SQL injection was not tested yet
A04:2021-Insecure Design	● Including unrequired scripts on pages. For example, you have the stripe payment gateway JS script included in the login page ● The target application's login and registration page can be brute forced
A05:2021-Security Misconfiguration	● Chrome and Safari can access HTTP connection, which is unencrypted ● Firefox, Chrome and Safari do not complete requests when a port is explicitly specified ● Using a wildcard certificate means that a stolen certificate has a lot of value. Since the same private key is used across multiple systems, you face increasing the risk of compromise across your organization. I subdomain specific cert requires a MiTM to spoof your domain. ● HSTS security header not set
A06:2021-Vulnerable and Outdated Components	● Angular Inspector browser plugin identified Modernizr Angular package as reported to be 3.6 which was released around 2014 according to the GitHub repository. The current version is 3.12. < 3.4 has reported known vulnerabilities in the Marked package.

A07:2021-Identification and Authentication Failures	● The target application's very weak password keyspace is vulnerable. However, password controls need to be reviewed from the whole flow to properly understand the risks. ● Ability to block email addresses from being able to register by entering email address without any other verification, that address cannot register
A08:2021-Software and Data Integrity Failures	● None
A09:2021-Security Logging and Monitoring Failures	● The target site did not have the ability to detect brute force login attempts, and registration submissions ● The target site does not monitor failed login attempts
A10:2021-Server-Side Request Forgery	● None

Cross-Site Scripting

Attempts to inject JS into the target application were conducted but it is suspected that they were ineffective due to the correct use of **AngularJS** binding of data to output into the HTML DOM. For measure, below is a list of measures for preventing html/script injection in Angular. Reference documents named **angularowasptop10.pdf** and **OWASPLondon20170727_AngularJS.pdf** are included for references to mitigating OWASP Top Ten vulnerabilities in AngularJS.

Preventing html/script injection in Angular

- Use interpolation with {{}} to automatically apply escaping Use safe property binding such as [href], [src], [style.color] Use binding to [innerHTML] to safely insert HTML data
- Do not use bypassSecurityTrust*() on untrusted data

Preventing code injection outside of Angular

- Avoid direct DOM manipulation E.g. through ElementRef or other client-side libraries
- Do not combine Angular with server-side dynamic pages Use Ahead-Of-Time compilation (AOT)

[4. Conclusion

The target domain was reviewed and data was collected in an attempt to find vulnerabilities and misconfigurations in the web server. Secondly, information about the target application was collected and some penetration tests were done to identify any design flaws and misconfigurations.

It was identified that **the target domain is protected by a Cloudflare WAF**. However, there were several configuration errors found. The most critical of these web-server and/or Cloudflare misconfigurations were that **the target domain is accessible via unencrypted HTTP connections**. This allows MITM attackers to read all data passed between the target application and the client host. In a WiFi situation, anyone on the same WiFi network could easily read that data without the need for physical access or exploiting the user client machine with MITM malware.

Secondly, **a wildcard certificate is in use** on the target domain. This greatly increases the potential damage caused by a stolen certificate. The wildcard certificate can be used to falsely verify the authenticity of any subdomain / hostname. Using a subdomain specific certificate would limit the stolen cert to only being able to impersonate the subdomain, which would require a MITM to poison DNS.

Also, **TLS 1.0 and TLS 1.1 should be disabled** due to associated vulnerabilities that can lead to information disclosure. TLS 1.0 and TLS 1.1 would only be required by legacy systems to connect since TLS 1.2 has been supported by all major browsers since 2013. Finally Microsoft and Mozilla have depreciated TLS 1.0 and TLS 1.1 support in 2020.

Another problem with server configuration to provide a robust SSL/TLS connection to clients is that the **HSTS security header is not configured for the site**. This means that clients are not requested to demand an HTTPS (SSL/TLS) connection. This allows an attacker to downgrade the connection, in this case to HTTP unencrypted connection.

The other most critical vulnerabilities are related to the web-application itself. There are several important areas of concern. Firstly, the **site does not prevent brute-force password guessing attempts**. This would allow an attacker unlimited attempts to brute-force the password. Secondly, **the requirements for password keyspace are very low**. The minimum length is 6 characters, and special characters are not required. This keyspace should be strengthened to have a minimum of 8 characters and require at least one special character.

Thirdly, **the Stripe payment gateway API is included on external pages** such as login and registration pages. This is a critical problem since it could potentially allow rouge payments to be processed using another parties API keys and stolen credit card data. It is strongly advised to only include the Stripe JS scripts on a page that accepts user payments. It should not be included on any other pages.

Fourthly, the Modernizr Angular package was reported to be 3.6 which was released around 2014 according to the packages GitHub page. The current version is 3.12. Versions < 3.4 have a known vulnerability. This package should be updated if possible to a more recent version.

Basic SQL injection and XSS attacks were attempted on the target application and it was able to pass those tests successfully.