Penetration Test Report

PREPARED FOR

Chaintrack Company

Simon Fraser University, MPCS Security

December 2022

PREPARED BY

Alia Bohsain Navarro

Allen Shaolun Liu

Jason Chen

Kaiyu Dong

Parsa Azimi Bajestani

Sami Salamati

Table of Contents

Executive Summary

1. Project Scope

Generate a report with a risk analysis based on the vulnerabilities found during the recognition and penetration phase to provide recommendations to improve the security in Chaintrack.

The report will be based on the Chaintrack web application only used by the dispatchers and excluding the mobile application that runs on iOS used by the drivers.

The results will be categorized according to their risk levels.

These measurements will help identify which vulnerabilities need to be addressed faster than others.

2. Project Objectives

There are several tasks that we will accomplish in this project. First, each team member is responsible for researching and implementing the penetration test and writing the report regarding the part of their assigned tasks.

Objectives	Team member
Gathering information	Allen
Configuration & deployment management	Kaiyu
Identity management	Jason
Authentication	Jason
Authorization	Sami
Session management	Sami
Input validation	Alia
Error handling test	Allen
Weak cryptography	Kaiyu

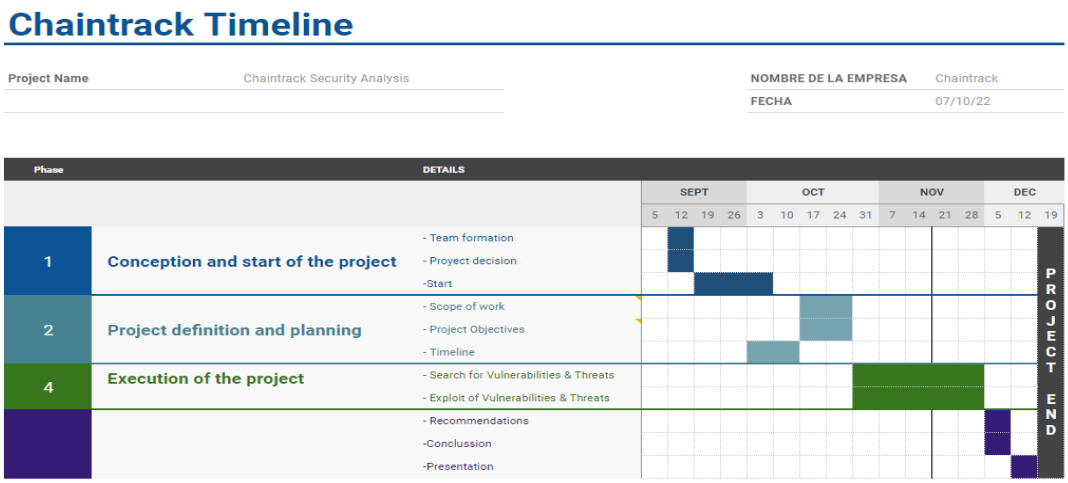
Business logic test	Parsa
Client side test	Alia

The table above indicates the main topic the penetration test covers with corresponding team members. Each member must choose several sub-topics under each main topic suitable to the chaintrack application.

More specifically, information gathering is essential because it helps penetration testers to understand the target system and identify potential vulnerabilities. Configuration and deployment management help to ensure the system is configured correctly and deployed securely. Proper configuration and deployment management can reduce the risk of vulnerabilities and attacks on the system. Identity management ensures that only authorized users have access to systems and data. Authentication and Authorization also help in preventing unauthorized access to sensitive information and reduce the risk of attacks on systems. Session management ensures the entrance to the methods and data is appropriately controlled and authorized. Input validation helps ensure the data input into the system is valid and legitimate. Error handling tests can reveal information about the application tested. Weakness of the cryptography indicates the possibility of the application being exploited by an attacker to gain unauthorized access to sensitive information. Business logic testing involves testing the logic and processes of the application. Client-side testing consists in testing the security of the client-side component.

3. Timeline

The proposed timeline to complete this project was planted as follows: each cell represents a week since the project was initiated. The execution of the project phase is related to the table of responsibilities previously provided.



4. Summary of Findings

Below is a list of the vulnerabilities we found. For more details, please check the Detailed Findings section.

(1) The remote web server is not enforcing HSTS, which allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

(2) According to the server response header, the installed version of nginx is prior to 1.17.7. It is, therefore, affected by an information disclosure vulnerability.

(3) The browser sends requests with 'Remembered Password Cookie' everytime.

(4) No account lock or IP lock after too many failed log in attempts.

(5) Cookie HttpOnly, Secure, SameSite attributes are not set.

(6) Users can export other teams' orders.

(7) Users can get a list of vehicles of other teams.

(8) Users can get, create, and delete other teams orders.

5. Summary of Recommendations

Below is a list of the recommendations to remediate the vulnerabilities mentioned above. For more details, please check the Detailed Findings section.

(1) Set cookie HttpOnly, Secure, SameSite attributes.

(2) Configure the remote web server to use HSTS.

(3) Upgrade to nginx version 1.17.7 or later.

(4) Set the path of the remembered credential cookie in the response to the login page.

(5) Add account lock and IP lock after too many failed log in attempts.

(6) Control the access of the users over other teams in different APIs.

Methodology

Tests are splitted using OWASP testing guide categories.

Tools used:

1. ZAP

OWASP ZAP (short for Zed Attack Proxy) is an open-source web application security scanner. It is intended to be used by both those new to application security as well as professional penetration testers.

It is one of the most active Open Web Application Security Project (OWASP) projects and has been given Flagship status.

When used as a proxy server it allows the user to manipulate all of the traffic that passes through it, including traffic using HTTPS.

It can also run in a daemon mode which is then controlled via a REST API.

2. Burp Suite

Burp Suite is an integrated platform/graphical tool for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities. Burp Suite is installed by default in Kali Linux.

3. Maltego

Maltego is software used for open-source intelligence and forensics, developed by Paterva from Pretoria, South Africa. Maltego focuses on providing a library of transforms for discovery of data from open sources, and visualizing that information in a graph format, suitable for link analysis and data mining.

4. Nessus

Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer you have connected to a network. It does this by running over 1200 checks on a

given computer, testing to see if any of these attacks could be used to break into the computer or otherwise harm it.

5. Bash

Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell.

6. Arachni

Arachni is a feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of modern web applications.

7. sqlmap

Sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Detailed Findings

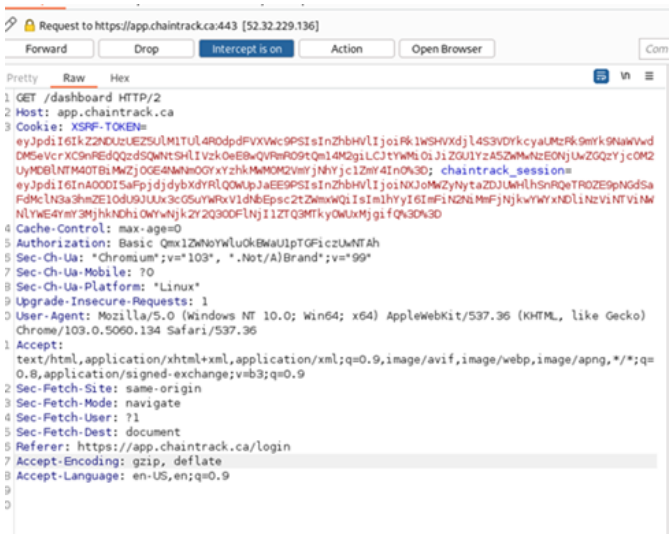
1 Information Gathering

Fingerprint Web Server

In the information-gathering section, our group decided to obtain the web server fingerprint with banner grabbing using tools such as Burp Suite and Nmap, etc. The following image indicates the result we get from Nmap functions to identify the type and version of the server on which the target application is running. Automated testing tools can easily capture such information. Also, they need to replace the path of the "target" with their target path. This step sets the target of the web application that users want to scan.

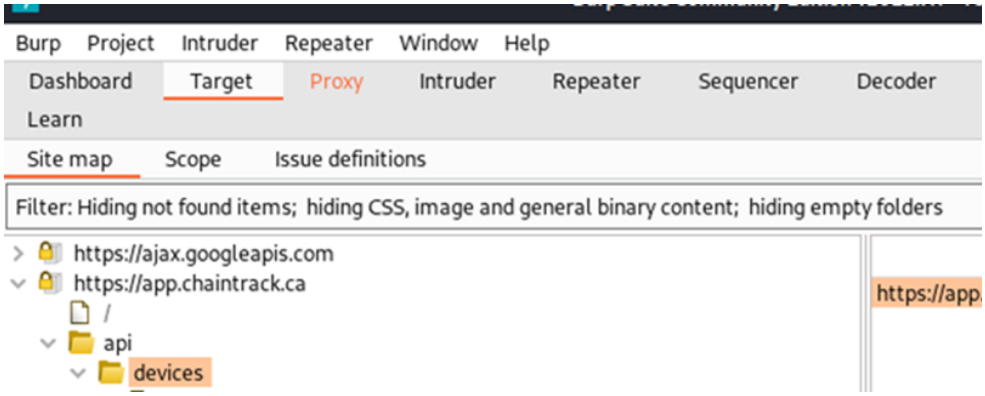
```
8008/tcp open   http
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8015/nice%20ports%2C/Tri%6Eity.txt%2ebak
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors 'self'
|   GenericLines, HTTPOptions, RTSPRequest, SIPOptions:
|     HTTP/1.1 302 Found
|     Location: https://:8015
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors 'self'
|   GetRequest:
|     HTTP/1.1 302 Found
|     Location: https://:8015/
|     Connection: close
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|_    Content-Security-Policy: frame-ancestors 'self'
8010/tcp open   ssl/xmpp?
| fingerprint-strings:
|   GenericLines, GetRequest:
|     HTTP/1.1 200 OK
|     Content-Length: 4492
|     Connection: close
|     Cache-Control: no-cache
|     Content-Type: text/html; charset=utf-8
|     X-Frame-Options: SAMEORIGIN
|     X-XSS-Protection: 1; mode=block
|     X-Content-Type-Options: nosniff
|     Content-Security-Policy: frame-ancestors 'self'
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta http-equiv="X-UA-Compatible" content="IE=8; IE=EDGE">
|     <meta name="viewport" content="width=device-width, initial-scale=1">
|     <style type="text/css">
|     body {
|     height: 100%;
|     font-family: Helvetica, Arial, sans-serif;
|     color: #6a6a6a;
|     margin: 0;
|     display: flex;
|     align-items: center;
|     justify-content: center;
|_    input[type=date], input[type=email], input[type=number], input[type=password]
```

Banner grabbing is performed by sending an HTTP request to the web server and examining its response headers. It can also execute by using the Burp Suite application.

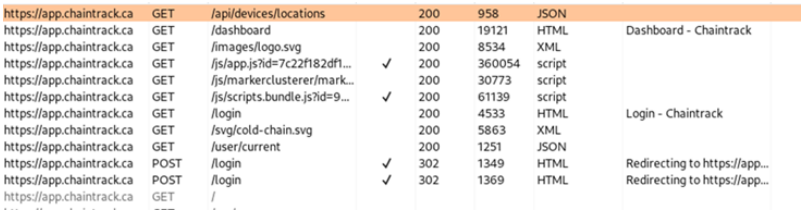


Application Entry Points

An application entry point is a point in a web application at which the execution of the program begins. It is the first code executed when a web app starts running. The entry point is crucial because it determines where the app starts executing and sets the context for the rest of the program. In this project, we decide to use Burp Suite to scan for all URLs in the application and identify all hidden parameters and all pages containing GET and POST requests.



We can go to Target and Site map tab to view the scan. Meanwhile, we need to open the application in the browser. In this case, we can see all the URLs that contain requests and corresponding parameters as well.



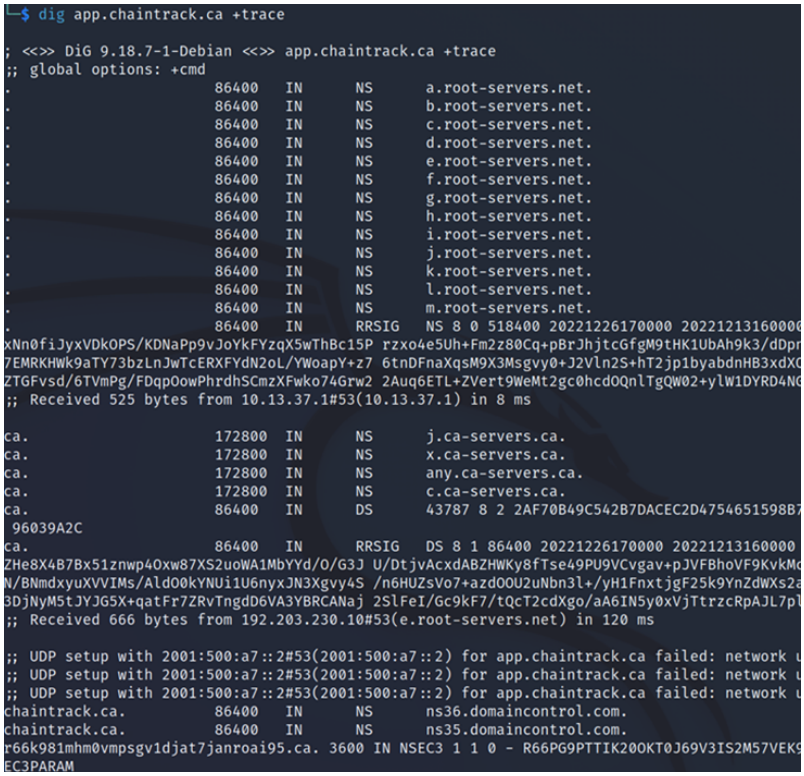
https://app.chaintrack.ca	GET	/api/devices/locations		200	958	JSON	
https://app.chaintrack.ca	GET	/dashboard		200	19121	HTML	Dashboard - Chaintrack
https://app.chaintrack.ca	GET	/images/logo.svg		200	8534	XML	
https://app.chaintrack.ca	GET	/js/app.js?id=7c22f182df1...	✓	200	360054	script	
https://app.chaintrack.ca	GET	/js/markerclusterer/mark...		200	30773	script	
https://app.chaintrack.ca	GET	/js/scripts.bundle.js?id=9...	✓	200	61139	script	
https://app.chaintrack.ca	GET	/login		200	4533	HTML	Login - Chaintrack
https://app.chaintrack.ca	GET	/svg/cold-chain.svg		200	5863	XML	
https://app.chaintrack.ca	GET	/user/current		200	1251	JSON	
https://app.chaintrack.ca	POST	/login	✓	302	1349	HTML	Redirecting to https://app...
https://app.chaintrack.ca	POST	/login	✓	302	1369	HTML	Redirecting to https://app...
https://app.chaintrack.ca	GET	/					

In Burp Suite, we are able to get all parameters that existing in the request on every page. For instance, in the login page, we can find username, password and session token. If users going through the list on Burp Suite, they can find all parameters they need for each page that contained by the web application.

DNS Trace Process

Domain Name System trace is a process involving following DNS query path from the client computer to the DNS server. The process is helpful in trouble identifying and

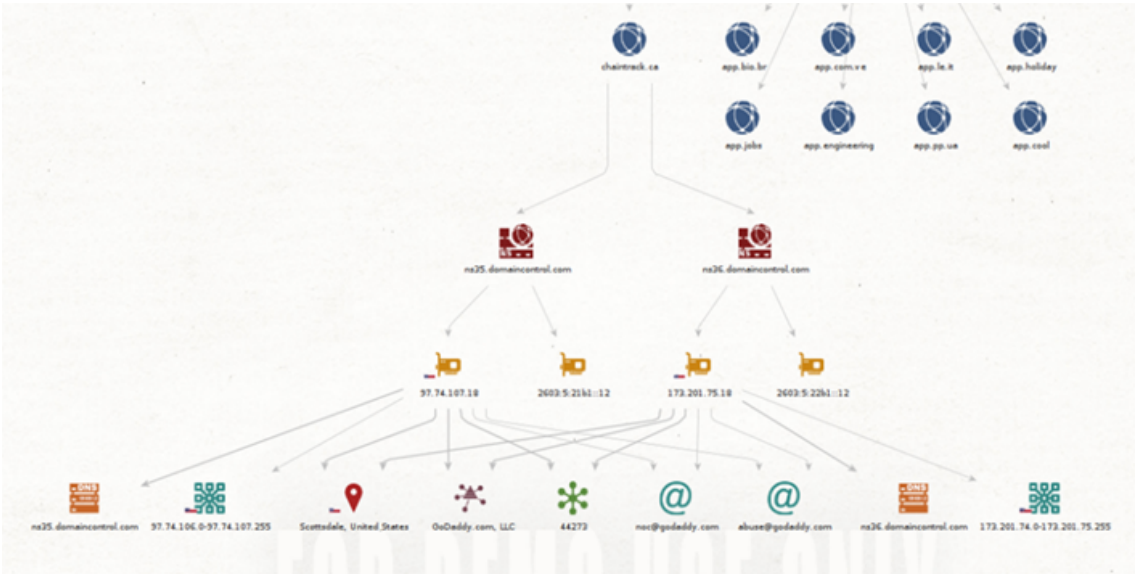
troubleshooting issues, also helps in for understanding how DNS queries are handled on the network.



The trace can help identify issues such as misconfigured nameservers or any network problem between the client and server. There are a number of tools that can help to implement such functions, such as Nmap, Dig, Nslookup, etc. in this case, we use the dig function to trace the DNS, which shows all the steps required to resolve the domain string from the root name server.

Maltego

Maltego is a powerful visualization tool for analyzing web applications, such as intelligence gathering, data mining and link analysis. It is good at demonstrating the relationship between entities such as domains, name servers, IP addresses, etc.



In this case, we can quickly gather information such as any related domains, IP addresses, locations, and email addresses from Maltego. For example, from the image above, we can tell that there are two name servers: ns35.chaintrack.ca and ns36.chaintrack.ca. Furthermore, two IP addresses are assigned to each of them.

2 Configuration and Deployment Management

Test Network Infrastructure Configuration

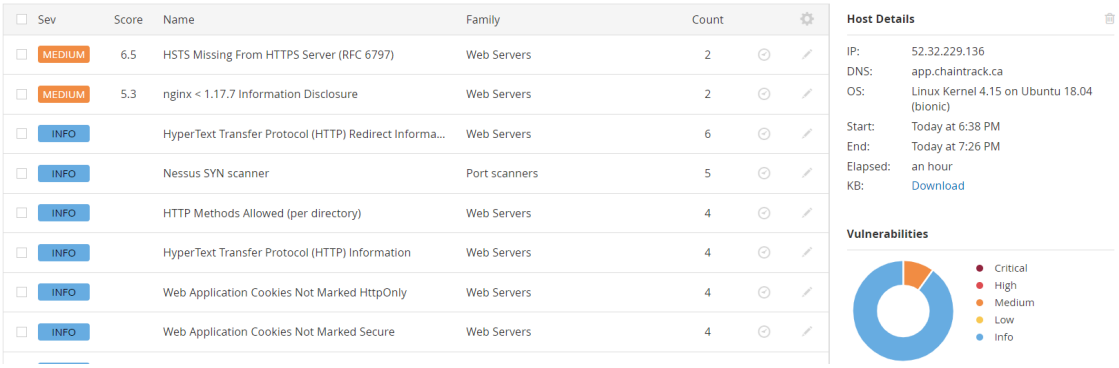
Test Objectives

- Review the applications' configurations set across the network and validate that they are not vulnerable.
- Validate that used frameworks and systems are secure and not susceptible to known vulnerabilities due to unmaintained software or default settings and credentials.

Steps and results

The tester should be provided with internal information of the software used, including versions and releases used and patches applied to the software.

We can partially do this by using Nessus:



Sev	Score	Name	Family	Count
MEDIUM	6.5	HSTS Missing From HTTPS Server (RFC 6797)	Web Servers	2
MEDIUM	5.3	nginx < 1.17.7 Information Disclosure	Web Servers	2
INFO		HyperText Transfer Protocol (HTTP) Redirect Informa...	Web Servers	6
INFO		Nessus SYN scanner	Port scanners	5
INFO		HTTP Methods Allowed (per directory)	Web Servers	4
INFO		HyperText Transfer Protocol (HTTP) Information	Web Servers	4
INFO		Web Application Cookies Not Marked HttpOnly	Web Servers	4
INFO		Web Application Cookies Not Marked Secure	Web Servers	4

According to its Server response header, the installed version of nginx is prior to 1.17.7. It is, therefore, affected by an information disclosure vulnerability.

Medium

MEDIUM nginx < 1.17.7 Information Disclosure

Description

According to its Server response header, the installed version of nginx is prior to 1.17.7. It is, therefore, affected by an information disclosure vulnerability.

Solution

Upgrade to nginx version 1.17.7 or later.

See Also

http://www.nessus.org/u?fd026623

Output

```
URL              : http://app.chaintrack.ca/
Installed version : 1.17.3
Fixed version     : 1.17.7
```

NGINX before 1.17.7, with certain error_page configurations, allows HTTP request smuggling, as demonstrated by the ability of an attacker to read unauthorized web pages in environments where NGINX is being fronted by a load balancer.

Recommendation

Upgrade to nginx version 1.17.7 or later.

Test Application Platform Configuration

Test Objectives

- Ensure that defaults and known files have been removed.
- Validate that no debugging code or extensions are left in the production environments.
- Review the logging mechanisms set in place for the application.

Steps and results

No results found. CGI scanning will be disabled for this host because the host responds to requests for non-existent URLs with HTTP code 301 rather than 404.

Test File Extensions Handling for Sensitive Information

Test Objectives

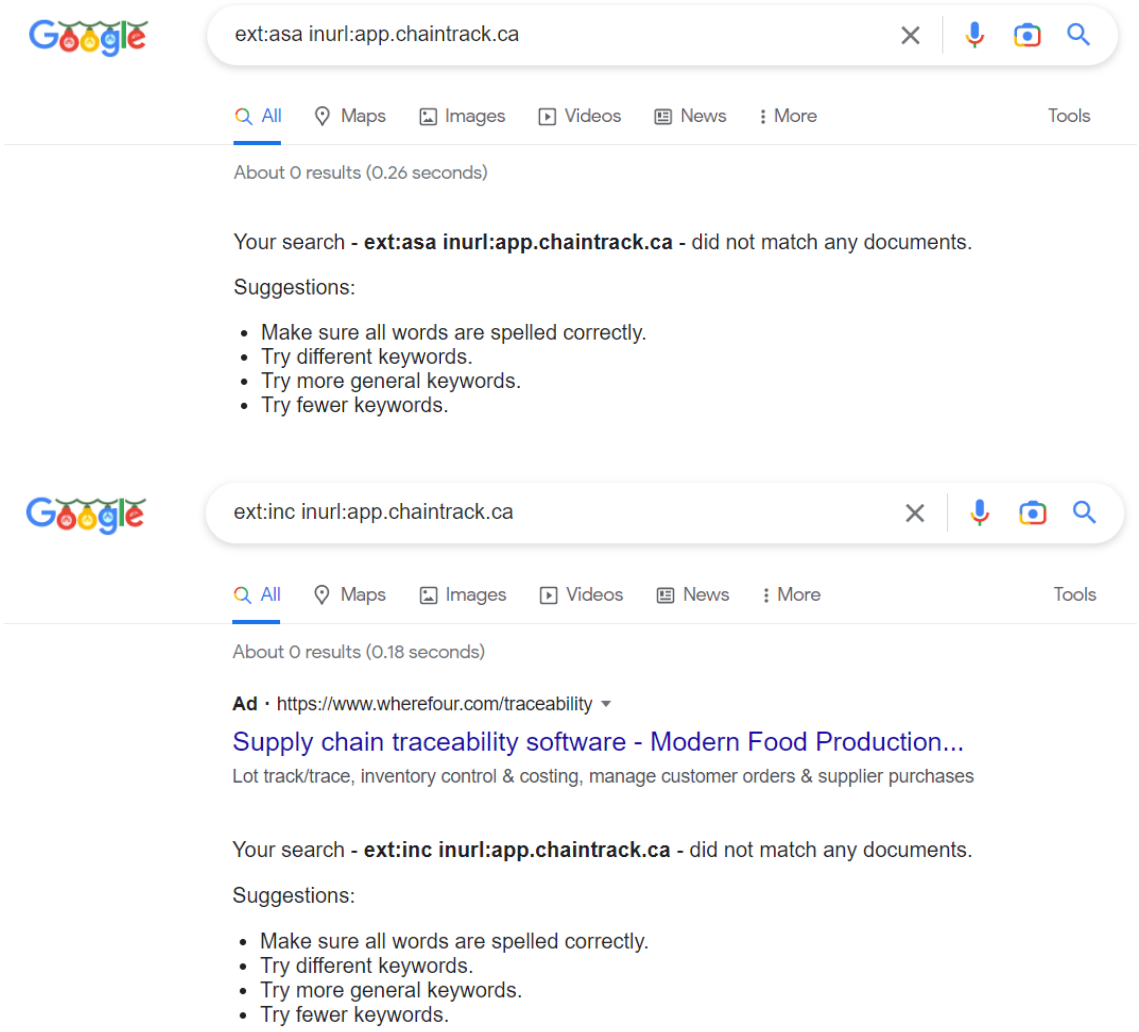
- Dirbust sensitive file extensions, or extensions that might contain raw data (e.g. scripts, raw data, credentials, etc.).
- Validate that no system framework bypasses exist on the rules set.

Steps and results

The following file extensions should NEVER be returned by a web server, since they are related to files which may contain sensitive information, or to files for which there is no reason to be served.

- .asa
- .inc

Black box testing using google hack:



No results found.

Enumerate Infrastructure and Application Admin Interfaces

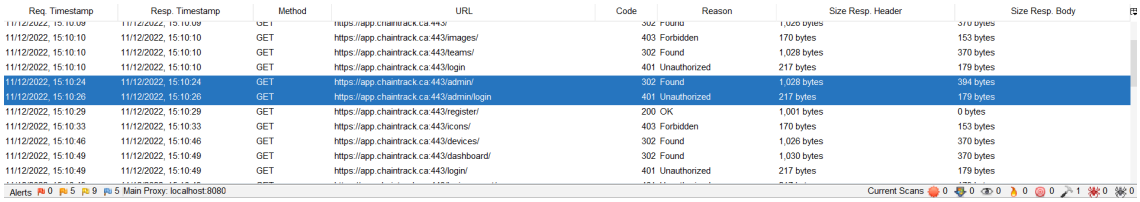
Test Objectives

- Identify hidden administrator interfaces and functionality.

Steps and results

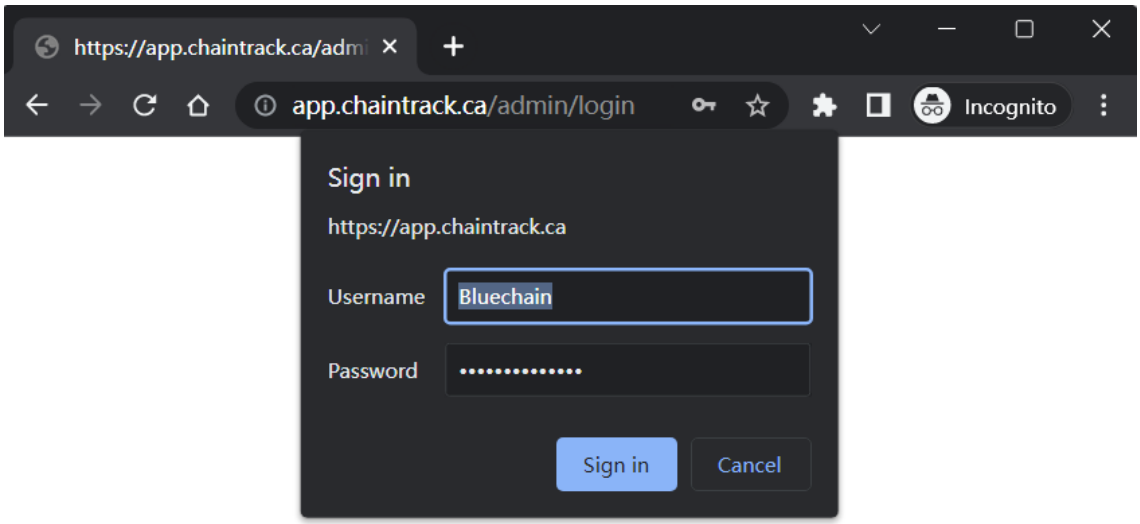
Black box testing using OWASP ZAP:

Forced Browse



Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	Size Resp. Header	Size Resp. Body
11/12/2022, 15:10:10	11/12/2022, 15:10:10	GET	https://app.chaintrack.ca:443/images/	403	Forbidden	170 bytes	153 bytes
11/12/2022, 15:10:10	11/12/2022, 15:10:10	GET	https://app.chaintrack.ca:443/teams/	302	Found	1,028 bytes	370 bytes
11/12/2022, 15:10:10	11/12/2022, 15:10:10	GET	https://app.chaintrack.ca:443/login	401	Unauthorized	217 bytes	179 bytes
11/12/2022, 15:10:24	11/12/2022, 15:10:24	GET	https://app.chaintrack.ca:443/admin/	302	Found	1,028 bytes	394 bytes
11/12/2022, 15:10:26	11/12/2022, 15:10:26	GET	https://app.chaintrack.ca:443/admin/login	401	Unauthorized	217 bytes	179 bytes
11/12/2022, 15:10:29	11/12/2022, 15:10:29	GET	https://app.chaintrack.ca:443/register/	200	OK	1,001 bytes	0 bytes
11/12/2022, 15:10:33	11/12/2022, 15:10:33	GET	https://app.chaintrack.ca:443/icons/	403	Forbidden	170 bytes	153 bytes
11/12/2022, 15:10:46	11/12/2022, 15:10:46	GET	https://app.chaintrack.ca:443/devices/	302	Found	1,026 bytes	370 bytes
11/12/2022, 15:10:49	11/12/2022, 15:10:49	GET	https://app.chaintrack.ca:443/dashboard/	302	Found	1,030 bytes	370 bytes
11/12/2022, 15:10:49	11/12/2022, 15:10:49	GET	https://app.chaintrack.ca:443/login/	401	Unauthorized	217 bytes	179 bytes

There is an admin page app.chaintrack.ca/admin/login which can be accessed from the public Internet:



Recommendation

Administration interfaces are an attack surface. Only legitimate administrators should be able to communicate with them. You should isolate these interfaces using architectural controls and constrain who can connect to the system and from where. This will help to protect against attacks such as brute forcing administrator login, or using an exploit to gain access.

There are a number of ways that you can reduce the exposure of your management interfaces. For example:

- You could create a dedicated management network that only authorized administrators have physical access to.
- You could place your administration interfaces behind a VPN that only authenticated administrators and devices can use.
- You could implement an IP allow list to restrict the devices or networks that can access the administration interface.

Test HTTP Strict Transport Security

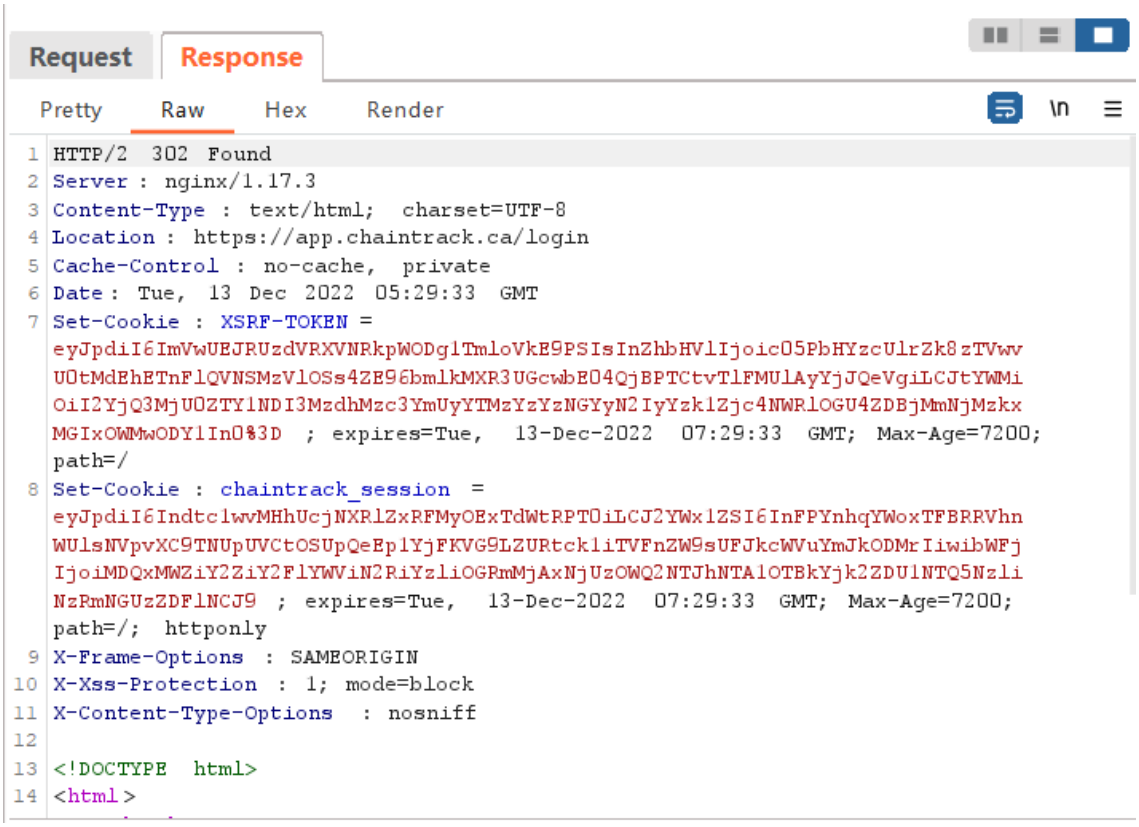
Test Objectives

- Review the HSTS header and its validity.

Steps and results

Using Burpsuite:

Medium



The response shows that the remote web server is not enforcing HSTS, which means the application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also

perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Recommendation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

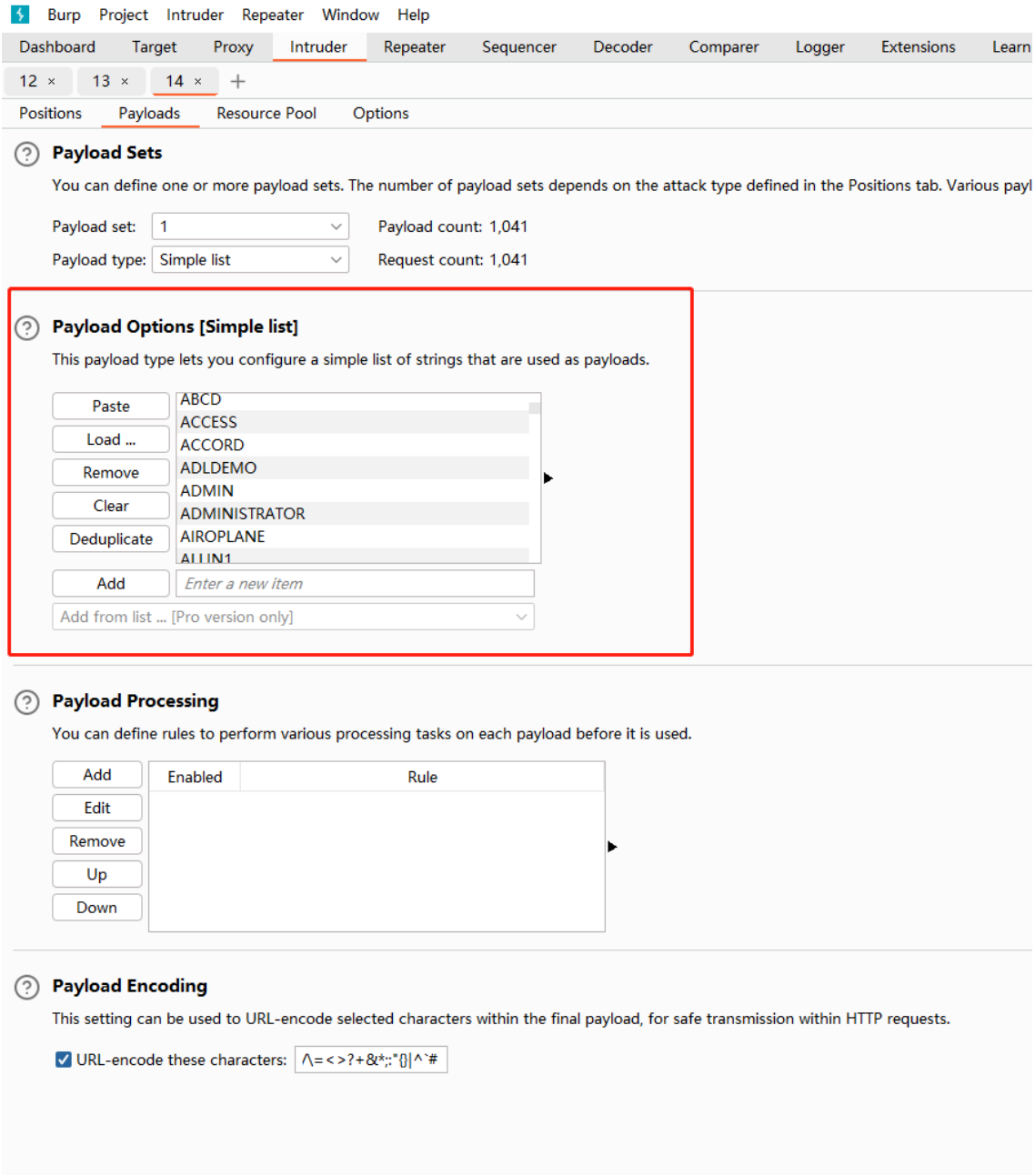
Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

3 Identity Management

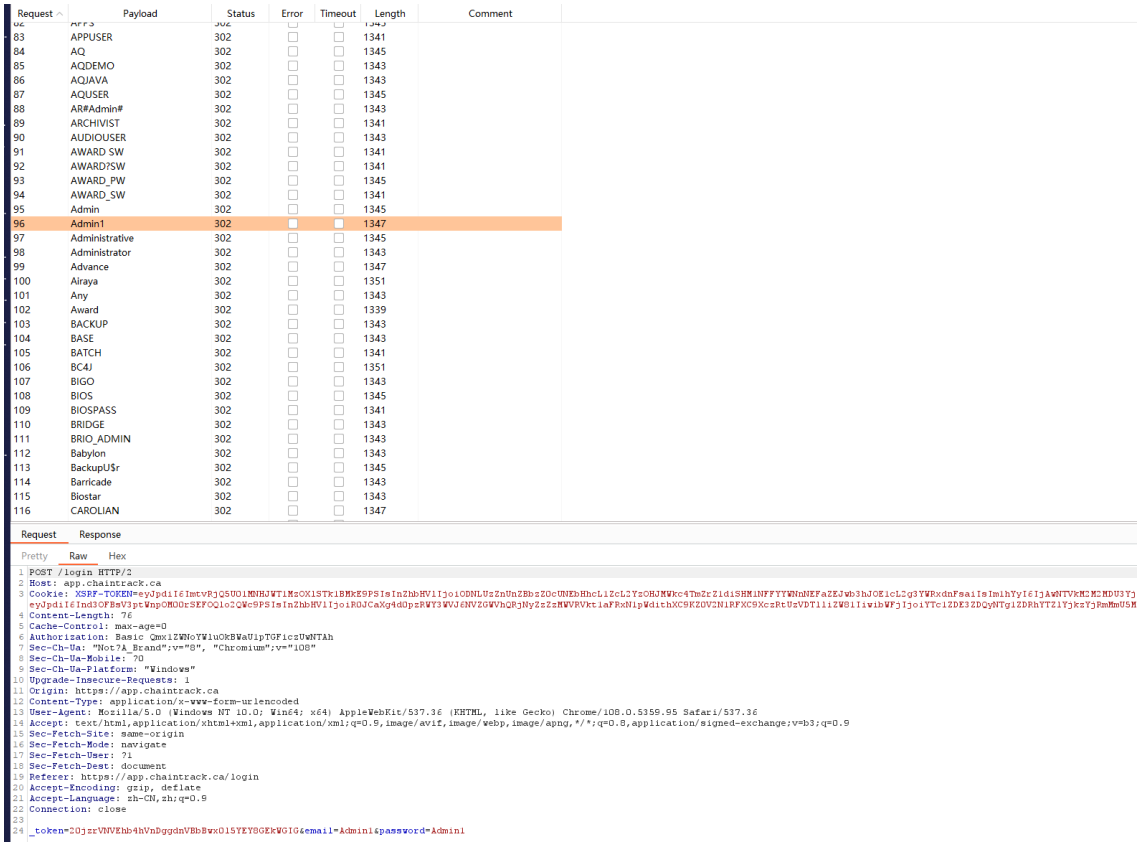
Testing for default credentials

Test Objectives:

- Try default usernames such as admin, administrator, root, system, guest, operator, and superuser. Use burp intruder to send automated requests with default credentials, brute forcing the login API.



No successful login:



Request	Payload	Status	Error	Timeout	Length	Comment
83	APPUSER	302			1341	
84	AQ	302			1345	
85	AQDEMO	302			1343	
86	AQJAVA	302			1343	
87	AQUSER	302			1345	
88	AR#Admin#	302			1343	
89	ARCHIVIST	302			1341	
90	AUDIOUSER	302			1343	
91	AWARD SW	302			1341	
92	AWARD?SW	302			1341	
93	AWARD_PW	302			1345	
94	AWARD_SW	302			1341	
95	Admin	302			1345	
96	Admin1	302			1347	
97	Administrative	302			1345	
98	Administrator	302			1343	
99	Advance	302			1347	
100	Airaya	302			1351	
101	Any	302			1343	
102	Award	302			1339	
103	BACKUP	302			1343	
104	BASE	302			1343	
105	BATCH	302			1341	
106	BC4J	302			1351	
107	BIGO	302			1343	
108	BIOS	302			1345	
109	BIOSPASS	302			1341	
110	BRIDGE	302			1343	
111	BRIO_ADMIN	302			1343	
112	Babylon	302			1343	
113	BackupU$r	302			1345	
114	Barricade	302			1343	
115	Biostar	302			1343	
116	CAROLIAN	302			1347	

Request Response

Pretty Raw Hex

```
1 POST /login HTTP/2
2 Host: app.chaintrack.ca
3 Cookie: XSRF-TOKEN=eyJpdiI6ImtvRjQSUOlMNHJWTlMzOX1STkiBMkE9PSIsInZhbHVlIjoiODNLUzZnUnZBbzZOcUNEbHhcL1ZcL2YzOHJMWkc4TmZrZ1diSHMiNFFYYWNnNEFaZEJwb3hJOE1cL2g3YWRxdnFsaiIsImihYyI6IjAwNTVkM2M2MDU3Yj
  eyJpdiI6Ind3OFBmV1ptWnpOMUUrSEFCQlo2QWc9PSIsInZhbHVlIjoiR0JCaXg4dUpzRWY3WVJ4NVZGWVhQRjNyZzZzMWVPVktlaFRxNlpWdithXC9KZOVZNiRFXC9XczRtUsVDTllZW8liwibWFjIjoiYTc1ZDE3ZDQyNTg1ZDRhYTZlYjkzYjRmMmU5M
4 Content-Length: 76
5 Cache-Control: max-age=0
6 Authorization: Basic QmxlZWNoYWiluOkBWaUipTGFicsUwNTAh
7 Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
8 Sec-Ch-Ua-Mobile: ?0
9 Sec-Ch-Ua-Platform: "Windows"
10 Upgrade-Insecure-Requests: 1
11 Origin: https://app.chaintrack.ca
12 Content-Type: application/x-www-form-urlencoded
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://app.chaintrack.ca/login
20 Accept-Encoding: gzip, deflate
21 Accept-Language: zh-CN,zh;q=0.9
22 Connection: close
23
24 _token=20jzrVNVEhb4hVnDggdnVBbBwxO15YEY8GEkWGIG&email=Admin1&password=Admin1
```

Low

Result:

The app passed the test, but no lockout, and no CAPTCHA challenge.

Recommendation:

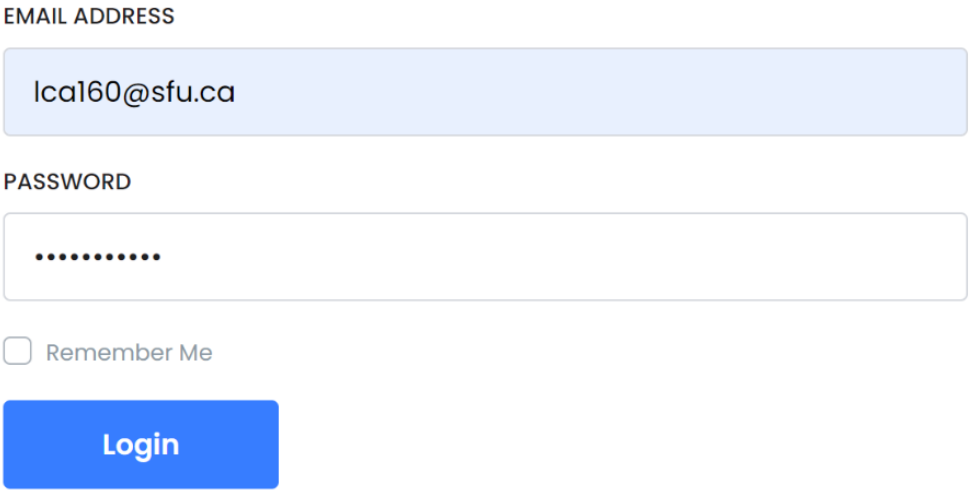
Add lockout IP mechanism and CAPTCHA challenges after a certain number of failed login attempts.

4 Authorization

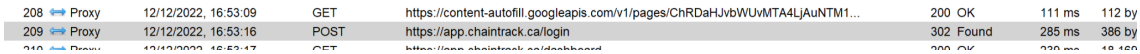
Encrypted Channel

Test Objectives:

- Black Box Testing
- To test if the App uses HTTPS to send login parameters, such as the user name, password, and challenge answers.



Check the login request using HTTPS and POST-method.



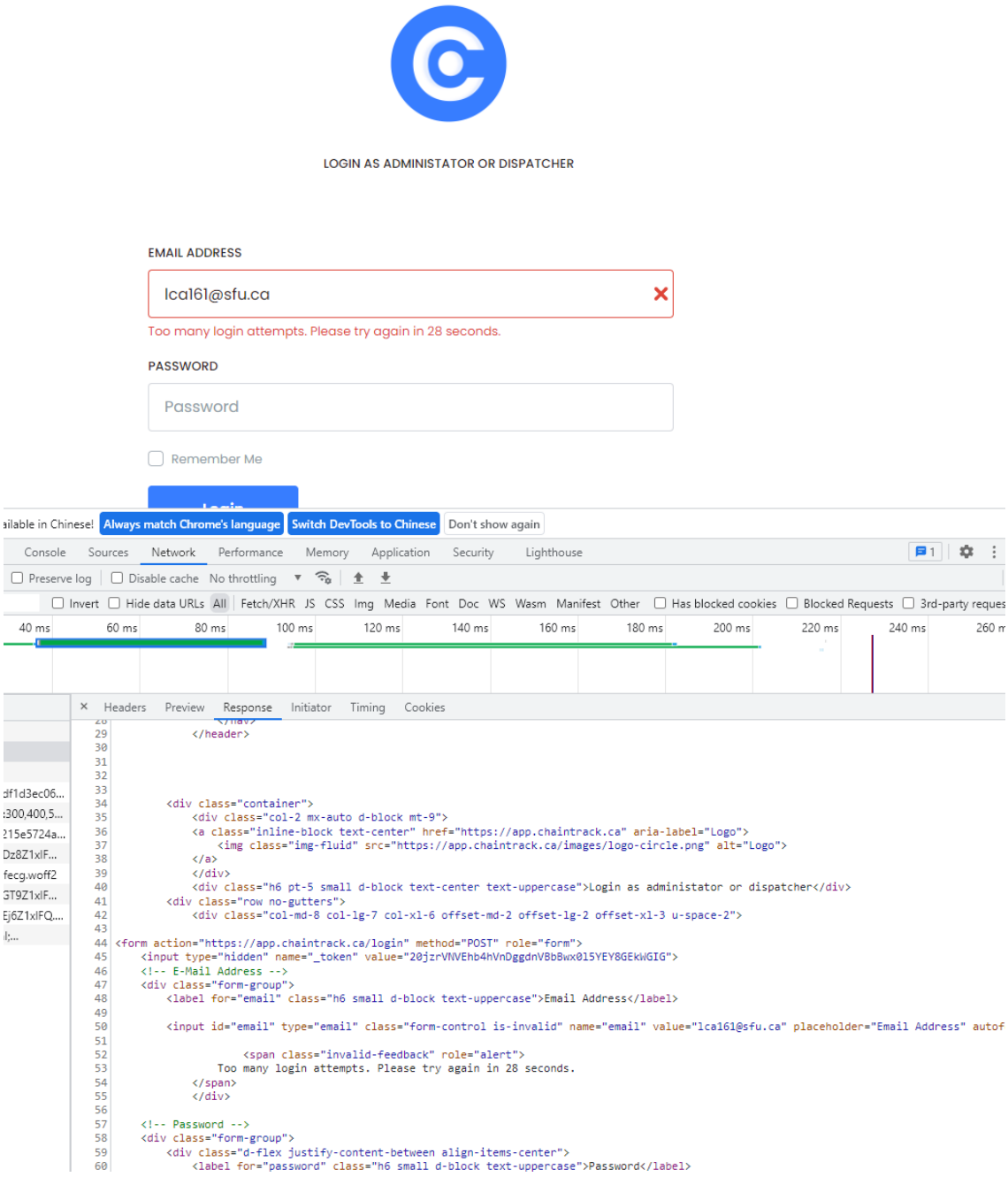
Result:

We can see that the app use HTTPS send the login parameters indeed. And the referrer is also HTTPS.

Testing for Weak lockout mechanism

Test Objectives:

- The objective of testing for weak lockout mechanisms is to ensure that the system being tested has adequate security measures in place to prevent unauthorized access and protect against brute force attacks.
- A weak lockout mechanism is a security feature that is designed to block access to a system or account after a certain number of failed login attempts. This is intended to prevent malicious actors from guessing passwords or using automated tools to try multiple combinations in order to gain access to a system.
- However, if the lockout mechanism is weak, it may not be effective at preventing unauthorized access, as it may allow too many failed login attempts before triggering a lockout. This can make the system vulnerable to brute force attacks, where an attacker can try a large number of different passwords in a short period of time in order to gain access.
- To test for weak lockout mechanisms, a tester may attempt to access a system or account using multiple incorrect passwords in rapid succession, and observe whether the lockout mechanism is triggered and how long it takes to do so. The tester may also attempt to bypass the lockout mechanism by using different IP addresses or other techniques.



Low

Result:

Locked out 60s after 5 failed attempts, but no lock.

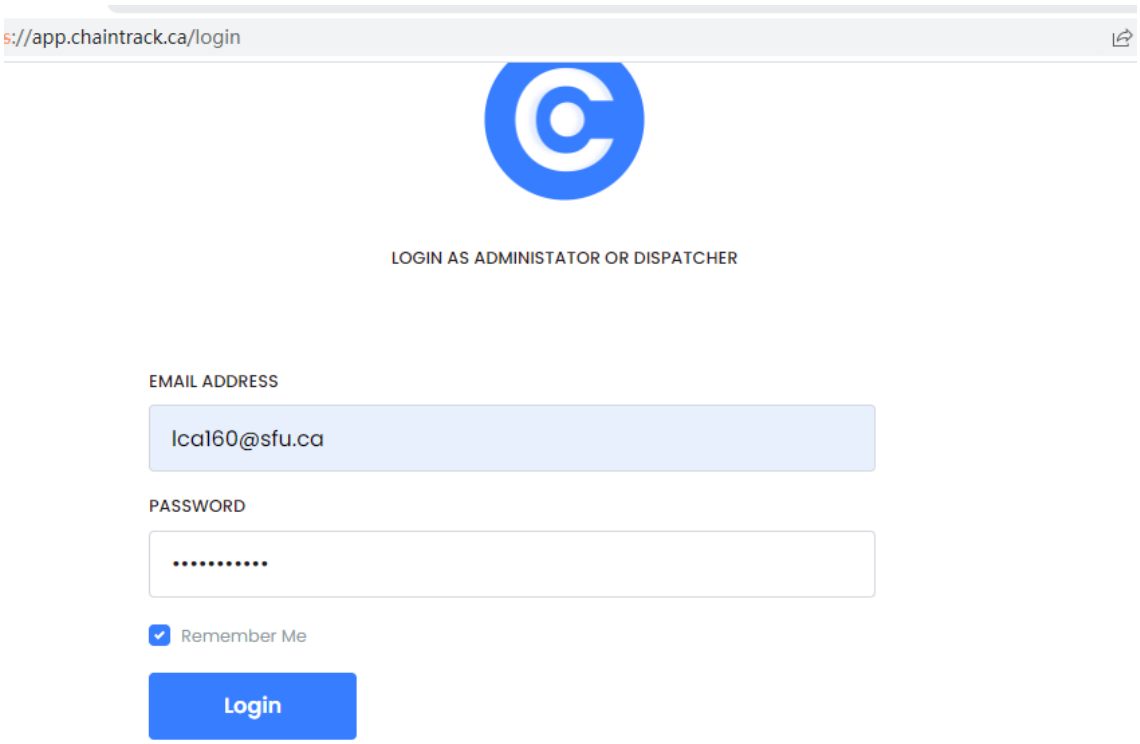
There is no CAPTCHA challenge.

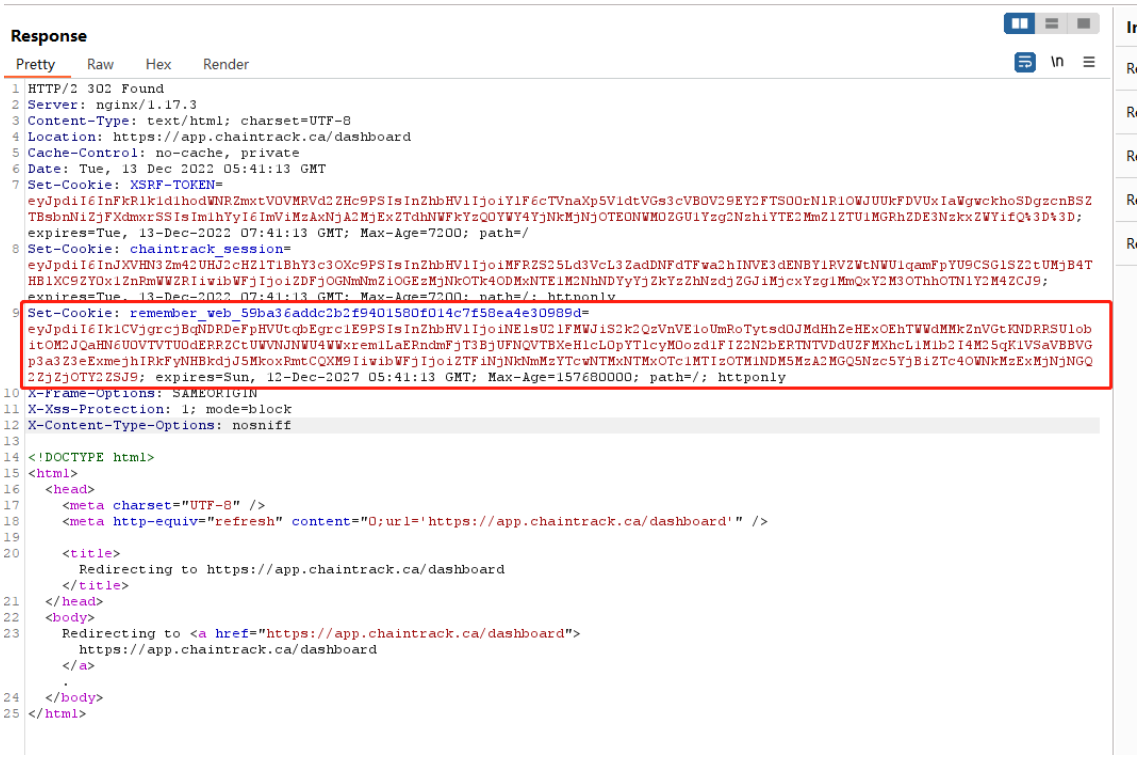
Recommendation:

Add a permanent lockout mechanism and CAPTCHA challenges after a certain number of failed login attempts.

Test remember password functionality

Look for passwords being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hashed.

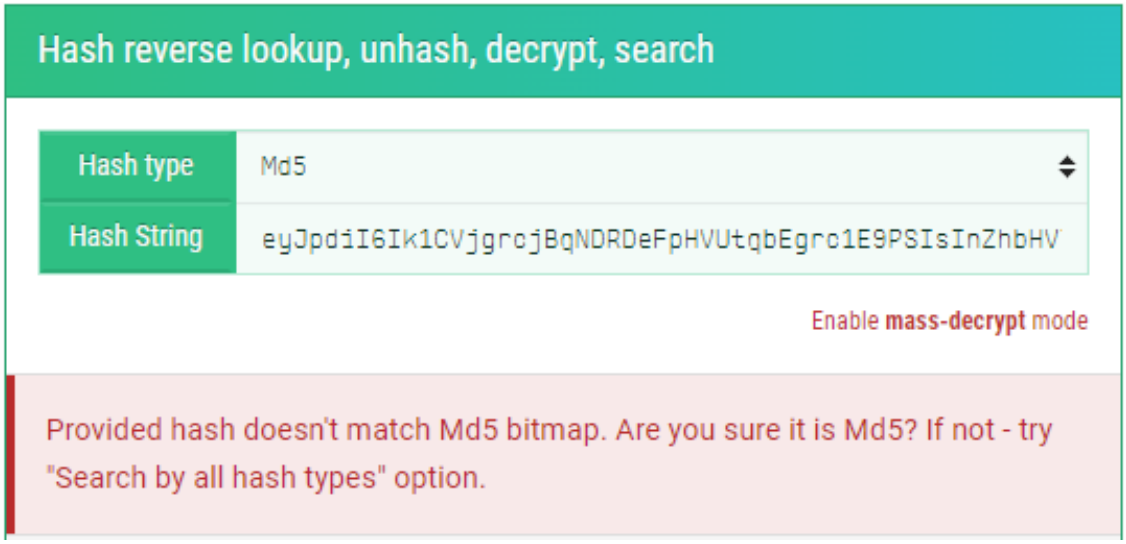


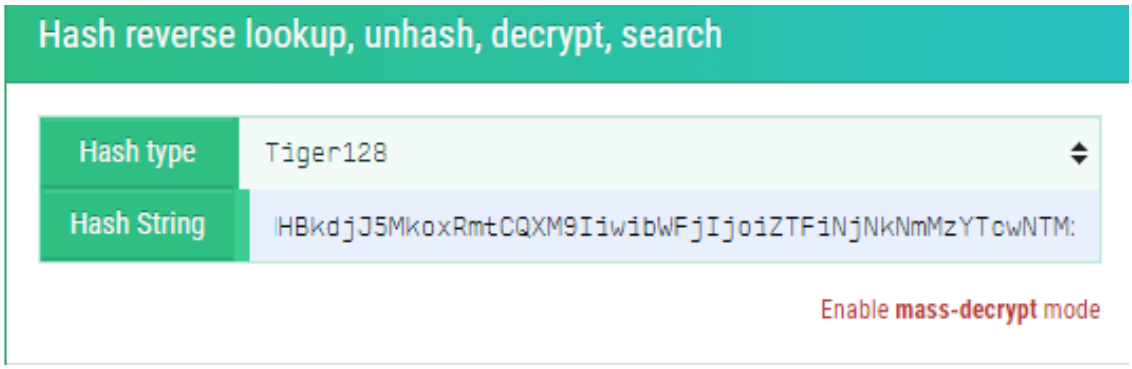


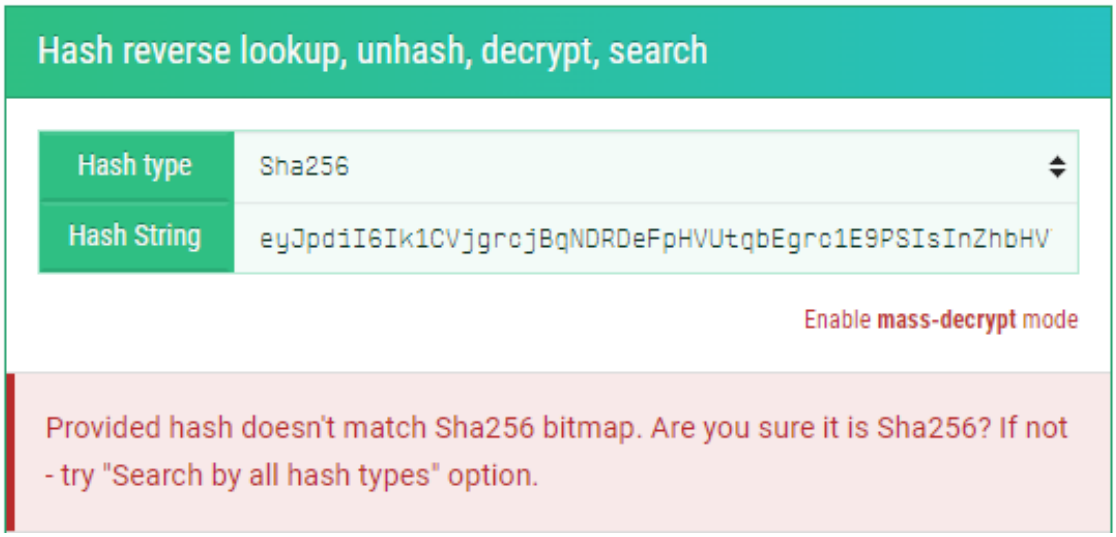
Result: The password is stored in cookies and hashed.

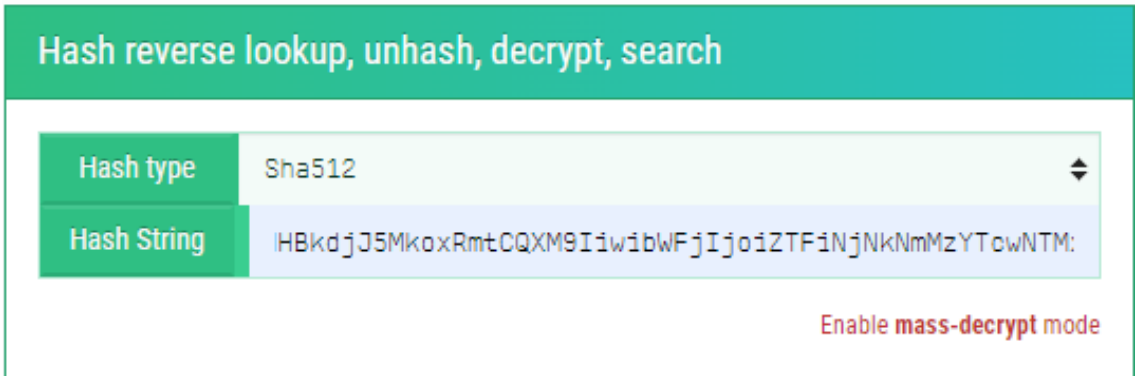
Examine the hashing mechanism

Examine the hashing mechanism: if it is a common, well-known algorithm, check for its strength, and its homegrown hash functions, and attempt several usernames to check whether the hash function is easily guessable.


Hash reverse lookup, unhash, decrypt, search
Hash type
Md5
Hash String
eyJpdiI6Ik1CVjgrojBqNDRDeFpHVUtqbEgro1E9PSIsInZhbHV
Enable mass-decrypt mode
Provided hash doesn't match Md5 bitmap. Are you sure it is Md5? If not - try "Search by all hash types" option.


Hash reverse lookup, unhash, decrypt, search
Hash type
Tiger128
Hash String
HBkdjJ5MkoxRmtCQXM9IiwibWFjIjoiZTFiNjNkNmMzYTcwNTM:
Enable mass-decrypt mode


Hash reverse lookup, unhash, decrypt, search
Hash type
Sha256
Hash String
eyJpdiI6Ik1CVjgrojBqNDRDeFpHVUtqbEgro1E9PSIsInZhbHV
Enable mass-decrypt mode
Provided hash doesn't match Sha256 bitmap. Are you sure it is Sha256? If not - try "Search by all hash types" option.



Result:

The hash is not breakable by simple methods, which means it is not vulnerable to hash collision attacks:

Verify that the credentials are only sent during the log-in phase and not sent together with every request to the application.

396	https://app.chaintrack.ca	GET	/devices
401	https://app.chaintrack.ca	GET	/user/current
402	https://app.chaintrack.ca	GET	/api/devices?page=1
403	https://use.fontawesome.com	GET	/releases/v5.0.13/webfonts/fa-solid-900.woff2

Request

Pretty Raw Hex

```
GET /devices HTTP/2
Host: app.chaintrack.ca
Cookie:
remember_web_59ba36addc2b2f9401580f014c7f58ea4e30989d=
eyJpdiI6Ik9BTmVDaSs3bjZ5ZVY2U3JUa1lDNnc9PSIsInZhbHVlIjoiYW9wZzVyS2RpKzdpVERORWHlLWGFWbGEraz I3OWtcLzYwZzY1UjBcLOozQ3NKTjV6eDBFQ1RTUk5nVHpDRkJnTDhRNXBrdFNueFZkSnZQWnUybWh3aWlZcTN2MmdkTzRIdEVQS1wvNDBoQ1wvTVhhNWkQ3dmdWRTTmN1UkxMcEY4T2hzMitLcXpPSXhUem93WWNWeFYlbzI2SUR1dVMzcE5jMEJUZGpVVFZwOXBcL1U9IiwibWFjIjoiN2RhOGQzM2JkMDE5YWRiNTNkYmY5MjdjMjU2NjIyMDM4MzcOMWIwZWZhYTU5NGMyYzNiNWQ4ZTJmMmJjZWQwZSJ9; XSRF-TOKEN=eyJpdiI6Ilwvako2VWdtZ3hDVmRCUzBldW9EMGtBPTOiLCJ2YWx1ZSI6IlZYMDBmRng3WDBPWm5kUmJYMTBzUWJoWUxOSO4rS1lEeVwvZDZtWXVXQjdFVXFcL1BROXpwa2U4ajNRQ3ZkelU3NyIsImlhYyI6IjBkNjdiZTdiNzdkZDAOZTNhNDBiNmQ2NzUwYWZkZTAzZmE2ZDEOZWViNTI3MmI4OTdjNzIyNTBjNWFjNzgxZmQifQ%3D%3D; chaintrack_session=eyJpdiI6ImJHdWxzWHRFREVIRWNScEYOZ1dqeHc9PSIsInZhbHVlIjoiZkoyYXMzTVRJcnhucDUOeWp6djBUdVNnT2ZpQ1NmVnN4aEdMTEJ6R3VuakwOODR1M1I1YmhOcDFDenhFckUyNyIsIm1hYyI6ImVlMDQ2MGYOZDQ2NTR1NjZkM2VkNjYxZDA3ZTVmYmJmMmVlZDU4OTYzNGQzMzkONWI1MzNjNTkwZWYyNWM1MzQifQ%3D%3D
Authorization: Basic Qmx1ZWNoYWluOkBWaU1pTGFiczUwNTAh
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://app.chaintrack.ca/dashboard
Accept-Encoding: gzip, deflate
```

Search... 0 matches

Response

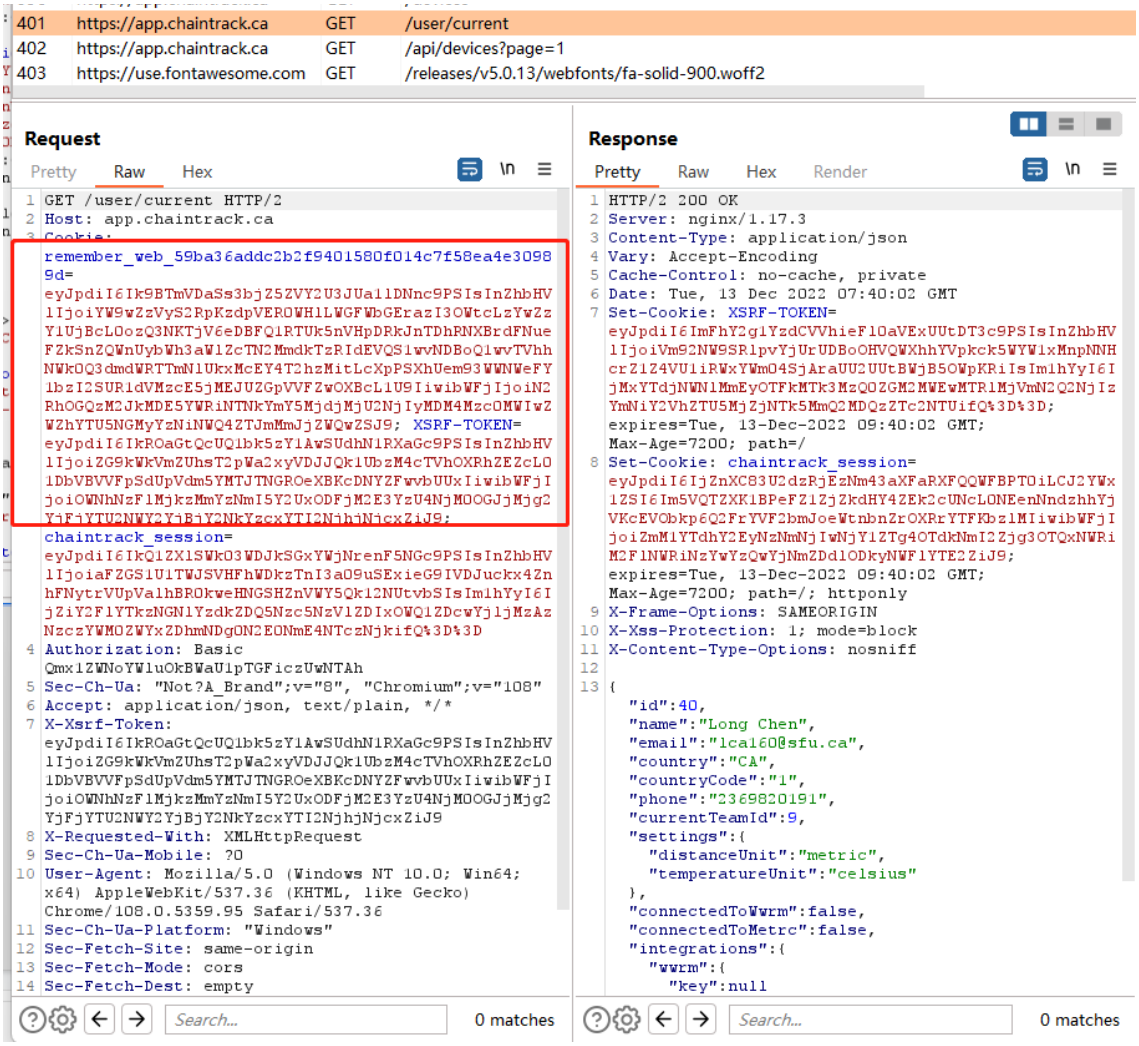
Pretty Raw Hex Render

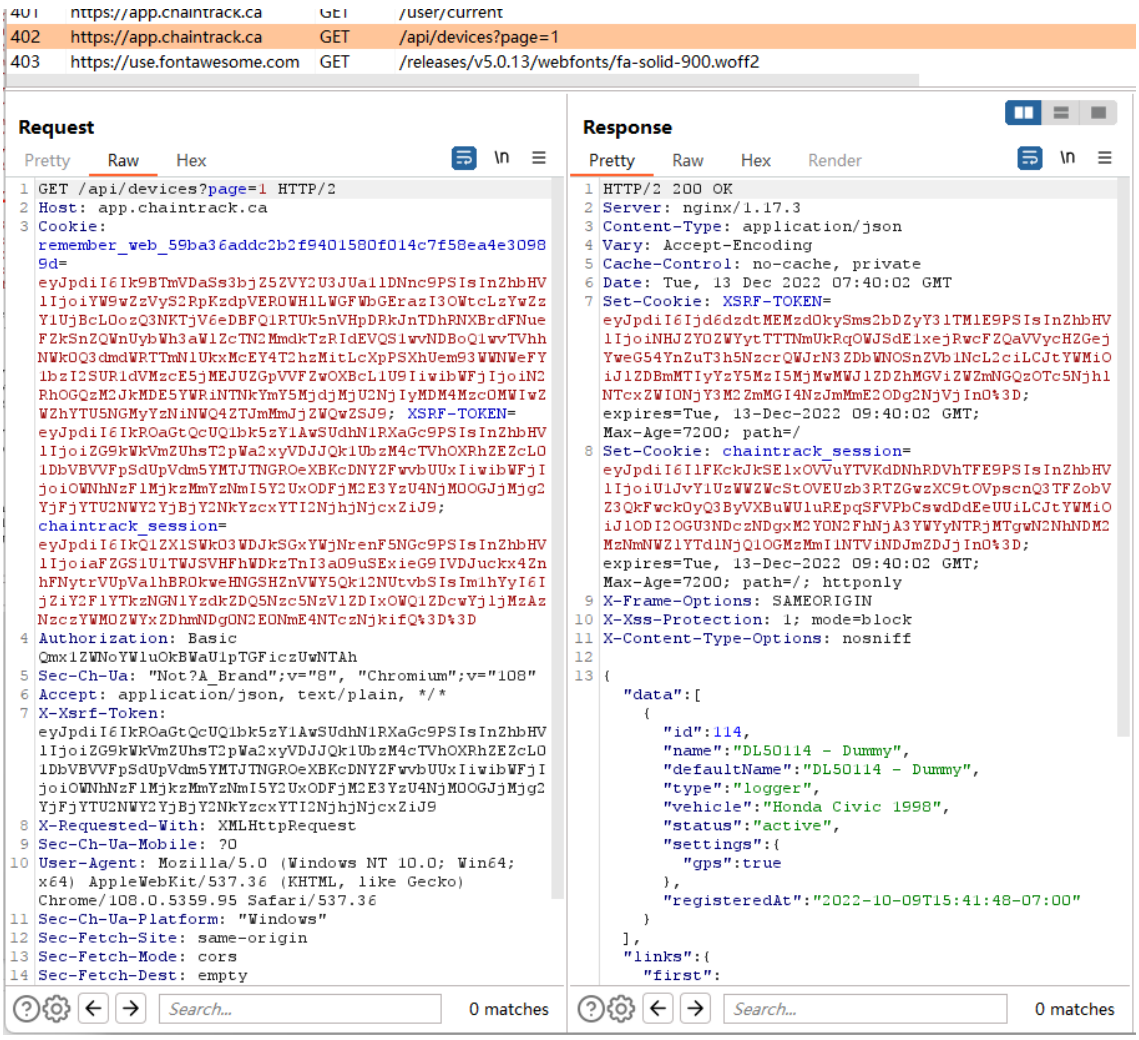
```
HTTP/2 200 OK
Server: nginx/1.17.3
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Cache-Control: no-cache, private
Date: Tue, 13 Dec 2022 07:40:02 GMT
Set-Cookie: XSRF-TOKEN=eyJpdiI6IkROaGtQcUQ1bk5zY1AwSUdhN1RXaGc9PSIsInZhbHVlIjoiZG9kWkVmZUhsT2pWa2xyVDJJQk1UbzM4cTVhOXRhZEZcLO1DbVBVVFpSdUpVdm5YMTJTNGROeXBKcDNYZFwvbUUxIiwibWFjIjoiOWNhNzFlMjkzMmYzNmI5Y2UxODFjM2E3YzU4NjM0OGJjMjg2YjFjYTU2NWY2YjBjY2NkYzcxYTI2NjhjNjcxZiJ9; expires=Tue, 13-Dec-2022 09:40:02 GMT; Max-Age=7200; path=/
Set-Cookie: chaintrack_session=eyJpdiI6IkQ1ZX1SWkO3WDJkSGxYWjNrenF5NGc9PSIsInZhbHVlIjoiaFZGS1U1TWJSVHFhWDkzTnI3aO9uSExieG9IVDJuckx4ZnhFNytrVUpValhBROkweHNGSHZnVWY5Qk12NUtvbSIsImlhYyI6IjZiY2F1YTkzNGN1YzdkZDQ5Nzc5NzVlZDIxOWQ1ZDcwYjljMzAzNzczYWMOZWYxZDhmNDgON2EONmE4NTczNjkifQ%3D%3D; expires=Tue, 13-Dec-2022 09:40:02 GMT; Max-Age=7200; path=/; httponly
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, shrink-to-fit=no">

    <title>
      IoT Devices - Chaintrack
    </title>
    <meta name="description" content="Get real-time monitoring for your temperature sensitive products.">
    <meta name="author" content="Chaintrack">
```

Search... 0 matches


401 https://app.chaintrack.ca GET /user/current
402 https://app.chaintrack.ca GET /api/devices?page=1
403 https://use.fontawesome.com GET /releases/v5.0.13/webfonts/fa-solid-900.woff2
Request
Pretty Raw Hex
GET /user/current HTTP/2
Host: app.chaintrack.ca
Cookie:
remember_web_59ba36addc2b2f9401580f014c7f58ea4e30989d=
eyJpdiI6Ik9BTmVDaSs3bjZ5ZVY2U3JUa1lDNnc9PSIsInZhbHVlIjoiYW9wZzVyS2RpKzdpVEROWHlLWGFWbGErazI3OWtcLzYwZzY1UjBcLOozQ3NKTjV6eDBFQ1RTUk5nVHpDRkJnTDhRNXBrdFNueFZkSnZQWnUybWh3aWlZcTN2MmdkTzRIdEVQS1wvNDBoQ1wvTVhhNWkQ3dmdWRTTmN1UkxMcEY4T2hzMitLcXpPSXhUem93WWNWeFY1bzI2SURldVMzcE5jMEJUZGpVVFZwOXBcL1U9IiwibWFjIjoiN2RhOGQzM2JkMDE5YWRiNTNkYmY5MjdjMjU2NjIyMDM4MzcOMWIwZWZhYTU5NGMyYzNiNWQ4ZTJmMmJjZWQwZSJ9; XSRF-TOKEN=eyJpdiI6IkROaGtQcUQ1bk5zY1AwSUdhN1RXaGc9PSIsInZhbHVlIjoiZG9kWkVmZUhsT2pWa2xyVDJJQk1UbzM4cTVhOXRhZEZcLO1DbVBVVFpSdUpVdm5YMTJTNGROeXBKcDNYZFwvbUUxIiwibWFjIjoiOWNhNzF1MjkzMmYzNmI5Y2UxODFjM2E3YzU4NjMOOGJjMjg2YjFjYTU2NWY2YjBjY2NkYzcxYTI2NjhjNjcxZiJ9;
chaintrack_session=eyJpdiI6IkQ1ZXlSWkO3WDJkSGxYWjNrenF5NGc9PSIsInZhbHVlIjoiaFZGS1U1TWJSVHFhWDkzTnI3a09uSExieG9IVDJuckx4ZnhFNytrVUpValhBROkweHNGSHZnVWY5Qk12NUtvbSIsIm1hYyI6IjZiY2F1YTkzNGN1YzdkZDQ5Nzc5NzVlZDIxOWQ1ZDcwYjljMzAzNzczYWM0ZWYxZDhmNDgON2EONmE4NTczNjkifQ%3D%3D
Authorization: Basic Qmx1ZWNoYWluOkBWaU1pTGFiczUwNTAh
Sec-Ch-Ua: "Not?A_Brand";v="8", "Chromium";v="108"
Accept: application/json, text/plain, */*
X-Xsrf-Token: eyJpdiI6IkROaGtQcUQ1bk5zY1AwSUdhN1RXaGc9PSIsInZhbHVlIjoiZG9kWkVmZUhsT2pWa2xyVDJJQk1UbzM4cTVhOXRhZEZcLO1DbVBVVFpSdUpVdm5YMTJTNGROeXBKcDNYZFwvbUUxIiwibWFjIjoiOWNhNzF1MjkzMmYzNmI5Y2UxODFjM2E3YzU4NjMOOGJjMjg2YjFjYTU2NWY2YjBjY2NkYzcxYTI2NjhjNjcxZiJ9
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Search...
0 matches
Response
Pretty Raw Hex Render
HTTP/2 200 OK
Server: nginx/1.17.3
Content-Type: application/json
Vary: Accept-Encoding
Cache-Control: no-cache, private
Date: Tue, 13 Dec 2022 07:40:02 GMT
Set-Cookie: XSRF-TOKEN=eyJpdiI6ImFhY2g1YzdCVVhieFlOaVExUUtDT3c9PSIsInZhbHVlIjoiVm92NW9SRlpvYjUrUDBoOHVQWXhhYVpkck5WYW1xMnpNNHcrZ1Z4VU1iRWxYWmO4SjAraUU2UUtBWjB5OWpKRiIsIm1hYyI6IjMxYTdjNWNlMmEyOTFkMTk3MzQ0ZGM2MWEwMTRlMjVmN2Q2NjIzYmNiY2VhZTU5MjZjNTk5MmQ2MDQzZTc2NTUifQ%3D%3D; expires=Tue, 13-Dec-2022 09:40:02 GMT; Max-Age=7200; path=/
Set-Cookie: chaintrack_session=eyJpdiI6IjZnXC83U2dzRjEzNm43aXFaRXFQQWFBPTOiLCJ2YWx1ZSI6Im5VQTZXK1BPeFZ1ZjZkdHY4ZEk2cUNcLONEenNndzhhYjVKcEVObkp6Q2FrYVF2bmJoeWtnbnZrOXRrYTFKbzlMIiwibWFjIjoiZmM1YTdhY2EyNzNmNjIwNjY1ZTg4OTdkNmI2Zjg3OTQxNWRiM2F1NWRiNzYwYzQwYjNmZDdlODkyNWF1YTE2ZiJ9; expires=Tue, 13-Dec-2022 09:40:02 GMT; Max-Age=7200; path=/; httponly
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
{
"id":40,
"name":"Long Chen",
"email":"lca160@sfu.ca",
"country":"CA",
"countryCode":"1",
"phone":"2369820191",
"currentTeamId":9,
"settings":{
"distanceUnit":"metric",
"temperatureUnit":"celsius"
},
"connectedToWwrm":false,
"connectedToMetrc":false,
"integrations":{
"wwrm":{
"key":null
Search...
0 matches



Low

Result:

The browser sends every request with the remembered credential.

Recommendation: set the path of the remembered credential cookie in the response to the login page.

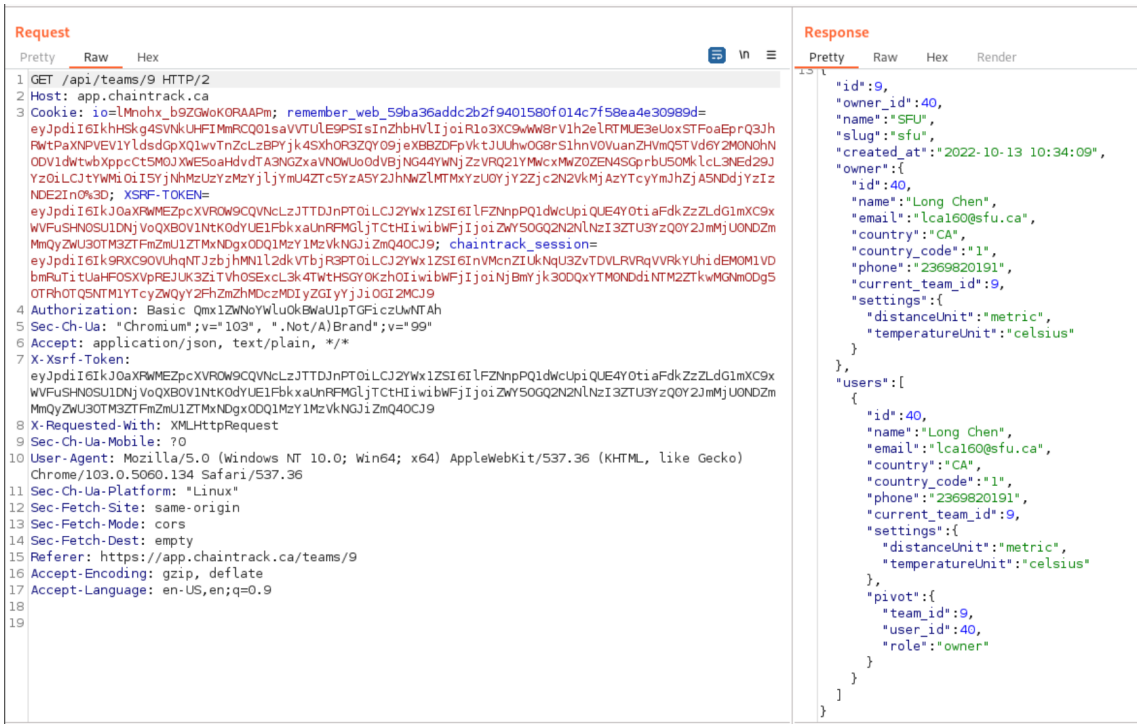
Insecure Direct Object References

Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files.

Methodology

To test for this vulnerability we first need to map out all locations in the application where user input is used to reference objects directly. Sometimes the object reference may be split between more than one parameter, and testing should be adjusted accordingly.

Structure of APIs



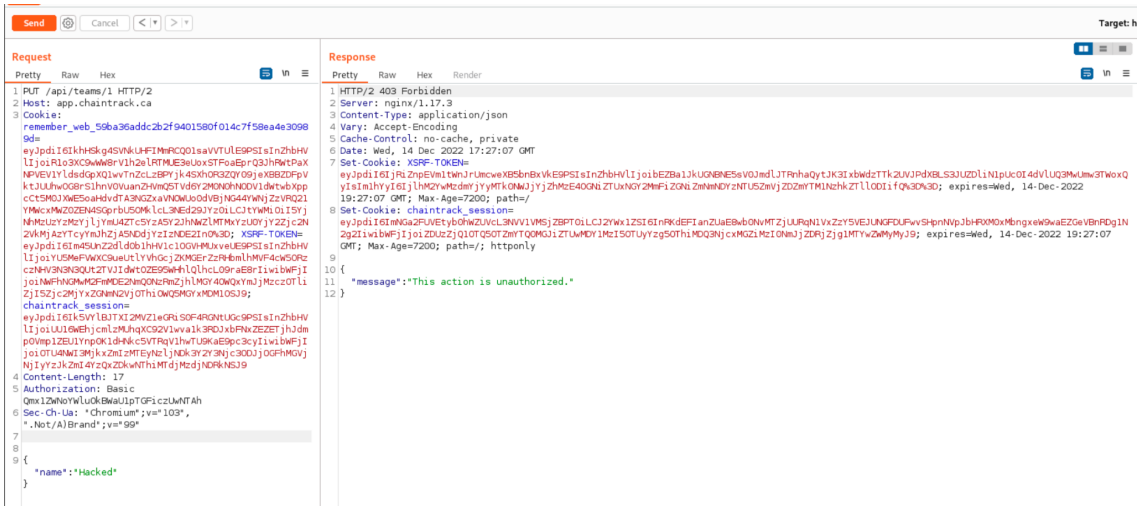
Request

```
GET /api/teams/9 HTTP/2
Host: app.chaintrack.ca
Cookie: io=lMnohx_b9ZGWoKORAAPm; remember_web_59ba36addc2b2f9401580f014c7f58ea4e30989d=
eyJpdiI6IkhHSkg4SVNkUHFIMmRCQO1saVVTUlE9PSIsInZhbHVlIjoiR1o3XC9wWW8rV1h2elRTMUE3eUoxSTFoaEprQ3Jh
RWtPaXNPVEV1YldsdGpXQ1wvTnZcLzBPYjk4SXhOR3ZQY09jeXBBZDFpVktJUUhwOG8rS1hnVOVuanZHVmQ5TVd6Y2M0NOhN
ODV1dWtwbXppcCt5M0JXWE5oaHdvdTA3NGZxaVNOWUo0dVBjNG44YWNjZzVRQ21YMWcxMWZ0ZEN4SGprbU5OMklcL3NEd29J
YzOiLCJtYWMiOiI5YjNhMzUzYzMzYjljYmU4ZTc5YzA5Y2JhNWZlMTMxYzU0YjY2Zjc2N2VkMjAzYTcyYmJhZjA5NDdjYzIz
NDE2InO%3D; XSRF-TOKEN=
eyJpdiI6IkJOaXRWMEZpcXVROW9CQVNcLzJTTDJnPTOiLCJ2YWx1ZSI6IlFZNnpPQ1dWcUpiQUE4YOtiaFdkZzZLdG1mXC9x
WVFuSHNOSU1DNjVoQXBOV1NtK0dYUE1FbkxaUnRFMGljTCtHIiwibWFjIjoiZWY5OGQ2N2NlNzI3ZTU3YzQ0Y2JmMjU0NDZm
MmQyZWU3OTM3ZTFmZmU1ZTMxNDgxODQ1MzY1MzVkNGJiZmQ4OCJ9; chaintrack_session=
eyJpdiI6Ik9RXC9OVUhqNTJzbjhMNll2dkVTbjR3PTOiLCJ2YWx1ZSI6InVMcnZIUkNqU3ZvTDVLRVRqVVRkYUhidEM0M1VD
bmRuTitUaHFOSXVpREJUK3ZiTVhOSExcL3k4TWtHSGY0KzhOIiwibWFjIjoiNjBmYjk3ODQxYTM0NDdiNTM2ZTkwMGNmODg5
OTRhOTQ5NTM1YTcyZWQyY2FhZmZhMDczMDIyZGIyYjJiOGI2MCJ9
Authorization: Basic Qmx1ZWNoYWluOkBWaU1pTGFiczUwNTAh
Sec-Ch-Ua: "Chromium";v="103", ".Not/A)Brand";v="99"
Accept: application/json, text/plain, */*
X-Xsrf-Token:
eyJpdiI6IkJOaXRWMEZpcXVROW9CQVNcLzJTTDJnPTOiLCJ2YWx1ZSI6IlFZNnpPQ1dWcUpiQUE4YOtiaFdkZzZLdG1mXC9x
WVFuSHNOSU1DNjVoQXBOV1NtK0dYUE1FbkxaUnRFMGljTCtHIiwibWFjIjoiZWY5OGQ2N2NlNzI3ZTU3YzQ0Y2JmMjU0NDZm
MmQyZWU3OTM3ZTFmZmU1ZTMxNDgxODQ1MzY1MzVkNGJiZmQ4OCJ9
X-Requested-With: XMLHttpRequest
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/103.0.5060.134 Safari/537.36
Sec-Ch-Ua-Platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://app.chaintrack.ca/teams/9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Response

```
{
  "id":9,
  "owner_id":40,
  "name":"SFU",
  "slug":"sfu",
  "created_at":"2022-10-13 10:34:09",
  "owner":{
    "id":40,
    "name":"Long Chen",
    "email":"lca160@sfu.ca",
    "country":"CA",
    "country_code":"1",
    "phone":"2369820191",
    "current_team_id":9,
    "settings":{
      "distanceUnit":"metric",
      "temperatureUnit":"celsius"
    }
  },
  "users":[
    {
      "id":40,
      "name":"Long Chen",
      "email":"lca160@sfu.ca",
      "country":"CA",
      "country_code":"1",
      "phone":"2369820191",
      "current_team_id":9,
      "settings":{
        "distanceUnit":"metric",
        "temperatureUnit":"celsius"
      },
      "pivot":{
        "team_id":9,
        "user_id":40,
        "role":"owner"
      }
    }
  ]
}
```

Testing some APIs by modifying parameters



Request

```
PUT /api/teams/1 HTTP/2
Host: app.chaintrack.ca
Cookie:
remember_web_59ba36addc2b2f9401580f014c7f58ea4e30989d=
eyJpdiI6IkhHSkg4SVNkUHFIMmRCQO1saVVTUlE9PSIsInZhbHVlIjoiR1o3XC9wWW8rV1h2elRTMUE3eUoxSTFoaEprQ3JhRWtP
aXNPVEV1YldsdGpXQ1wvTnZcLzBPYjk4SXhOR3ZQY09jeXBBZDFpVktJUUhwOG8rS1hnVOVuanZHVmQ5TVd6Y2M0NOhNODV1dW
twbXppcCt5M0JXWE5oaHdvdTA3NGZxaVNOWUo0dVBjNG44YWNjZzVRQ21YMWcxMWZ0ZEN4SGprbU5OMklcL3NEd29JYzOiLCJt
YWMiOiI5YjNhMzUzYzMzYjljYmU4ZTc5YzA5Y2JhNWZlMTMxYzU0YjY2Zjc2N2VkMjAzYTcyYmJhZjA5NDdjYzIzNDE2InO%
3D; XSRF-TOKEN=
eyJpdiI6Im45UnZ2dldObihHV1c1OGVMUxveUE9PSIsInZhbHVlIjoiYU5MeFVWXC9ueUtlYVhGcjZKMGErZzRHbmlhMVF4cW5ORz
czNHV3N3N3QUt2TVJIdWtOZE95WHhlQlhcLO9raE8rIiwibWFjIjoiNWFhNGMwM2FmMDE2NmQONzRmZjhlMGY4OWQxYmJjMzczOTli
ZjI5Zjc2MjYxZGNmN2VjOThiOWQ5MGYxMDM1OSJ9;
chaintrack_session=
eyJpdiI6Ik5vYlBJTXI2MVZleGRiSOF4RGNtUGc9PSIsInZhbHVlIjoiUl6WEhjcmlzMUhqXC92V1wva1k3RDJxbFNxZEZETjhJdm
pOVmp1ZEU1YnpOK1dHNkc5VTRqV1hwTU5KaE9pc3cyIiwibWFjIjoiOTU4NWI3MjkxZmIzMTEyNzljNDk3Y2Y3Njc3ODJjOGFhMGVj
NjIyYzJkZmI4YzQxZDkwNThiMTdjMzdjNDRkNSJ9
Content-Length: 17
Authorization: Basic
Qmx1ZWNoYWluOkBWaU1pTGFiczUwNTAh
Sec-Ch-Ua: "Chromium";v="103",
".Not/A)Brand";v="99"

{
  "name":"Hacked"
}
```

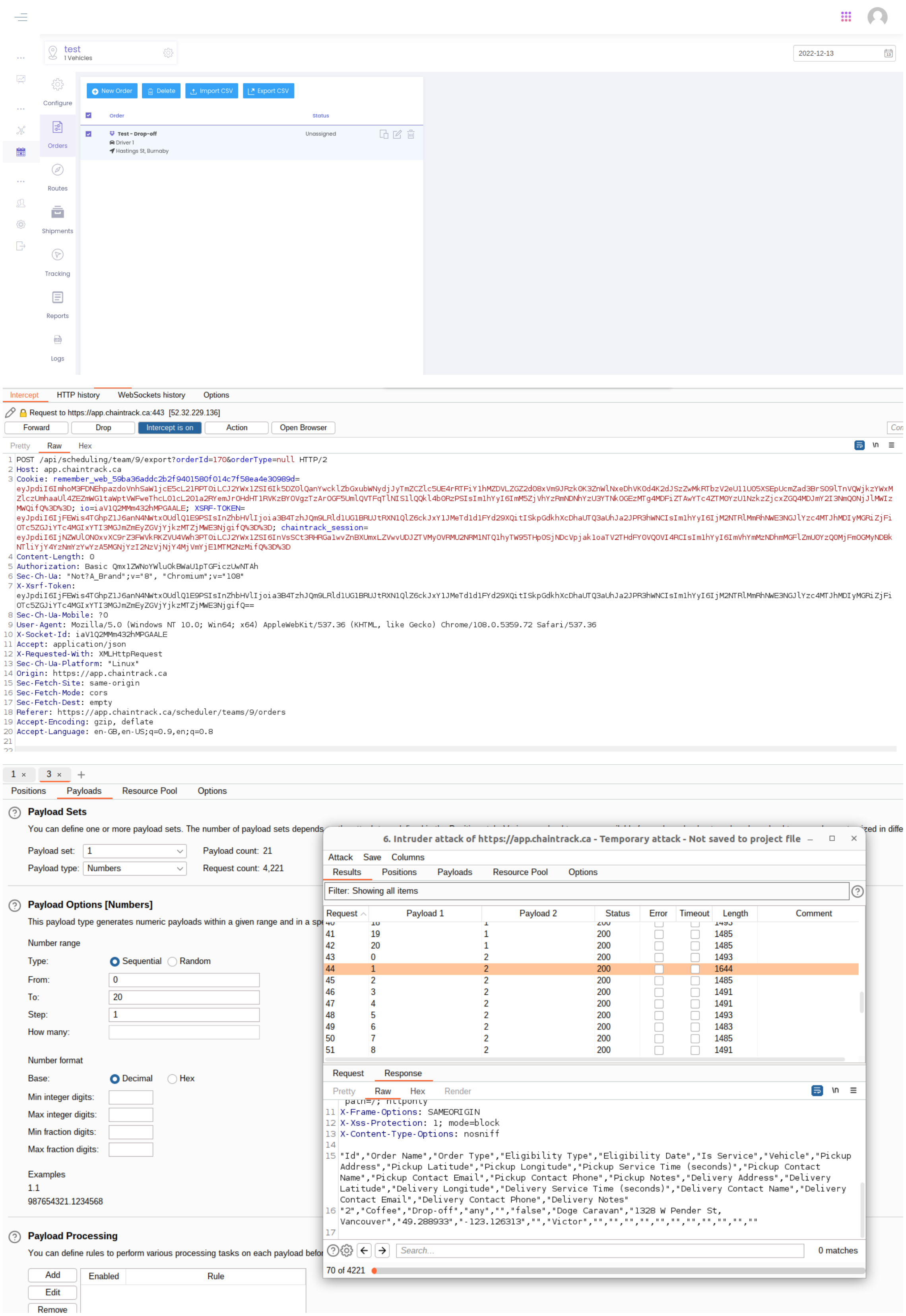
Response

```
HTTP/2 403 Forbidden
Server: nginx/1.17.3
Content-Type: application/json
Vary: Accept-Encoding
Cache-Control: no-cache, private
Date: Wed, 14 Dec 2022 17:27:07 GMT
Set-Cookie: XSRF-TOKEN=
eyJpdiI6IjRiZnpEVmltWnJrUmcweXB5bnBxVkE9PSIsInZhbHVlIjoibEZBa1JkUGNBNE5sVOJmdlJTRnhaQytJK3IxbWdzTTk2UVJPdXBLS3JUZDliN1pUcOI4dVlUQ3MwUmw3TWoxQ
yIsIm1hYyI6IjlhM2YwMzdmYjYyMTk0NWJjYjZhMzE4OGNiZTUxNGY2MmFiZGNiZmNmNDYzNTU5ZmVjZDZmYTM1NzhkZTllODIifQ%3D%3D; expires=Wed, 14-Dec-2022
19:27:07 GMT; Max-Age=7200; path=/
Set-Cookie: chaintrack_session=
eyJpdiI6ImNGa2FUVEtybOhWZUVcL3NVV1VMSjZ8PTOiLCJ2YWx1ZSI6InRKdEFIanZUaE8wb0NvMTZjUURqN1VxZzY5VEJUNGFDUFvvSHpnNVpJbHFXMOxMbngxeW9waEZGeVBnRDg1N
2g2IiwibWFjIjoiZDUzZjQ1OTQ5OTZmYTQ0MGJiZTUwMDY1MzI5OTUyYzg5OThiMDQ3NjcxMGZiMzI0NmJjZDRjZjg1MTYwZWMyMyJ9; expires=Wed, 14-Dec-2022 19:27:07
GMT; Max-Age=7200; path=/; httponly

{
  "message":"This action is unauthorized."
}
```

Exporting orders API works for other teams

High



Tools

Burp

Result

We can export orders of other teams.

Test
https://app.chaintrack.ca/scheduler/teams/9/orders

Findings

Orders page is vulnerable to Insecure Direct Object References. By changing the export url, we were able to see other teams orders which include information about their cars, locations, depots, drivers, etc. This is a critical vulnerability.

Recommendations

The API should check if the user is authorized for exporting orders before returning the data.

5 Session Management

Logout Functionality

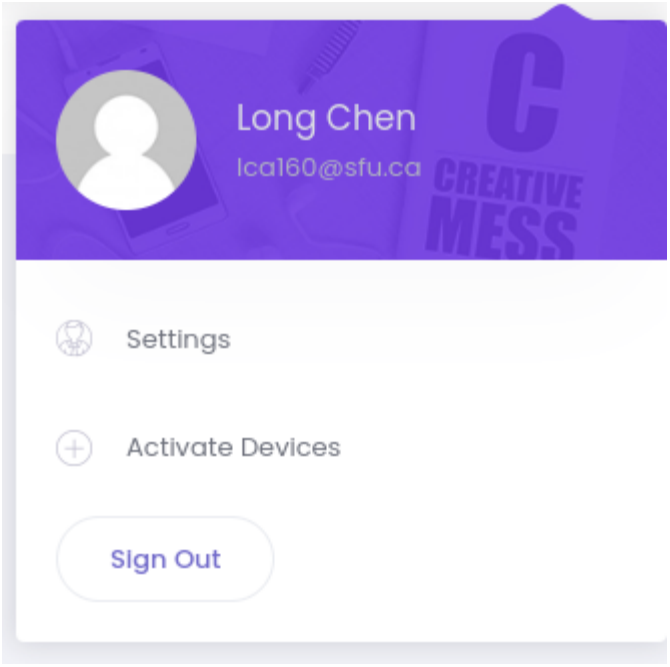
Methodology

There are some properties which indicate a good log out user interface:

1. A logout button is present on all pages of the web application.
2. The log out button should be identified quickly by a user who wants to log out from the web application.
3. After loading a page the logout button should be visible without scrolling.

Ideally the log out button is placed in an area of the page that is fixed in the viewport of the browser and not affected by scrolling of the content.

Chaintrack Logout



Tools

Browser

Test

All pages after login to the system.

Findings

Logout UI and functionality is good.

Recommendations

-

Cross Site Request Forgery (CSRF)

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated.

Methodology

For a black box test we must know URLs in the restricted (authenticated) area. As we have valid credentials, we can find the URLs to be tested just by browsing around the application.

Tools

OWASP ZAP

Test

Scan the app using OWASP ZAP to find CSRF vulnerability.

Findings

The app uses CSRF token and is not vulnerable to CSRF attack.

Recommendations

-

Cookies Attributes

Cookies are often a key attack vector for malicious users (typically targeting other users) and the application should always take due diligence to protect cookies.

Methodology

We should trap all responses where a cookie is set by the application (using the Set-cookie directive) and inspect the cookie for the following attributes:

1. Secure: This attribute tells the browser to only send the cookie if the request is being sent over a secure channel such as HTTPS. This will help protect the cookie from being passed over unencrypted requests. If the application can be accessed over both HTTP and HTTPS, then there is the potential that the cookie can be sent in clear text.
2. HttpOnly: This attribute is used to help prevent attacks such as cross-site scripting, since it does not allow the cookie to be accessed via a client side script such as JavaScript. Note that not all browsers support this functionality.
3. Domain Attribute - Verify that the domain has not been set too loosely. It should only be set for the server that needs to receive the cookie. For example if the application resides on server app.mysite.com, then it should be set to "; domain=app.mysite.com" and NOT "; domain=.mysite.com" as this would allow other potentially vulnerable servers to receive the cookie.

Tools

OWASP ZAP

Findings

Cookie HttpOnly, Secure, SameSite flags are not set.

Recommendations

Mentioned attributes should be set.

6 Input Validation

Reflected Cross Site Scripting

Reflected Cross-site Scripting (XSS) occurs when an attacker injects browser executable code within a single HTTP response. The injected attack is not stored within the application itself; it is non-persistent and only impacts users.

The attack string is included as part of the crafted URI or HTTP parameters, processed by the application, and returned to the victim. Reflected XSS are the most frequent type of XSS attacks.

Methodology

For testing against Reflected XSS we are going to test pages since that requires some user input.
We are going to test in 2 ways a Black box, that consists of sending our own crafted URI or HTTP parameters and checking manually different attacks, and a grey box in which we will use the help of a tool.

Black Box:

Classic attack

<script>alert(Pentest)</ script>
“ onfocus=”alert(document.cookie)

Different syntax or encoding:

“ onfocus=”alert(document.cookie)
“><script>alert(document.cookie)</ script>
“><script>alert(Pentest)</ script>
“%3cscript%3ealert(document.cookie)%3c/script%3e
“%3cscript%3ealert(Pentest)%3c/script%3e

Bypassing non-recursive filtering

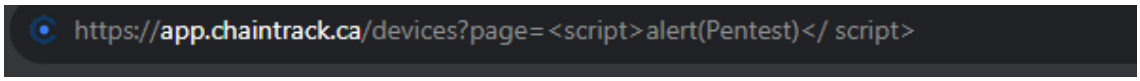
<src<script>ipt>alert(document.cookie)</ script>

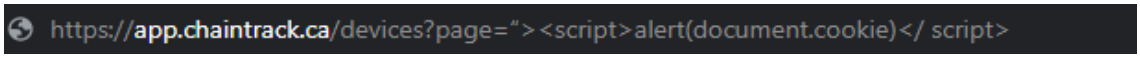
Tools

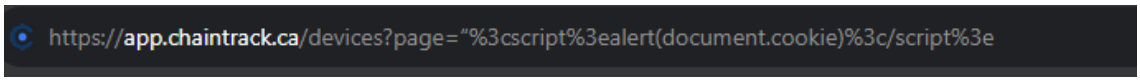
ZAP, Arachni

Test

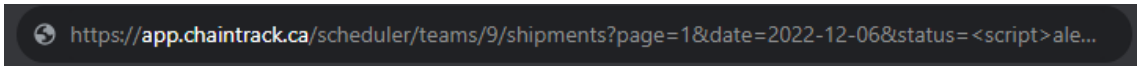
For the first step we have identify the following URLS
https://app.chaintrack.ca/devices?page=1

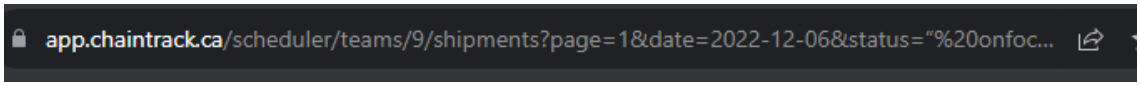


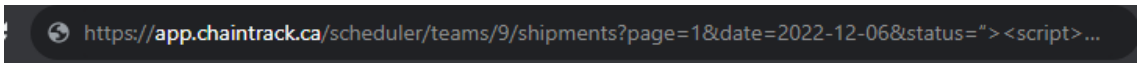


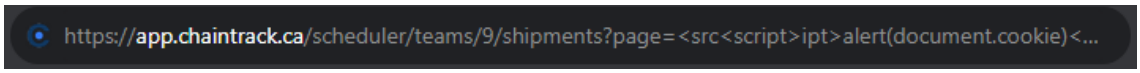


https://app.chaintrack.ca/scheduler/teams/9/shipments?page=1&date=2022-12-06&status=all

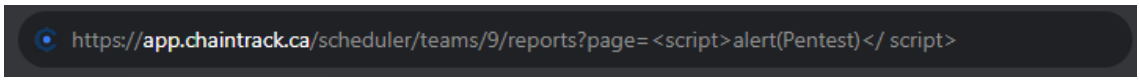


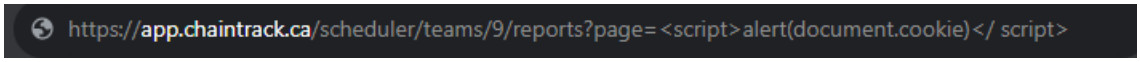


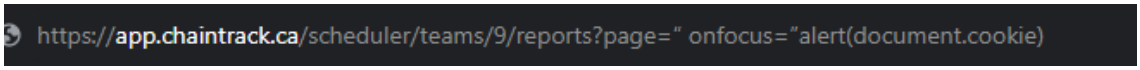


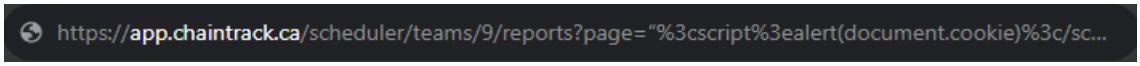


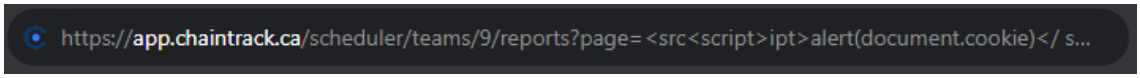
https://app.chaintrack.ca/scheduler/teams/9/reports?page=1&date=2022-12-09

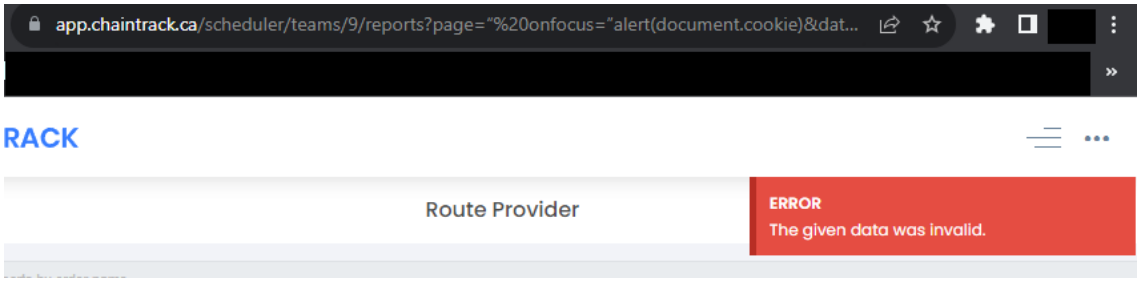












Black Box testing:

Classic attack

Result: Not vulnerable

Different syntax or encoding:

Result: Not vulnerable

Bypassing non-recursive filtering

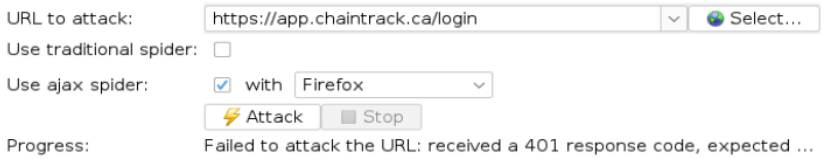
Result: Not vulnerable

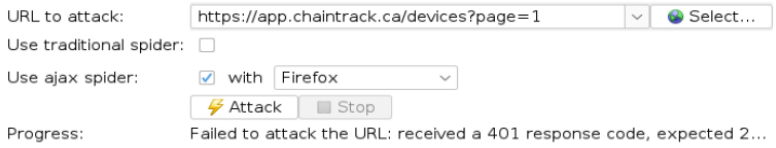
Including external script

Result: Not vulnerable

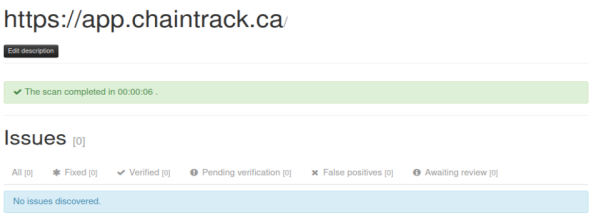
Gray Box testing

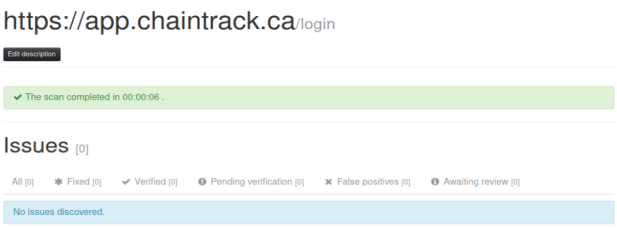
ZAP

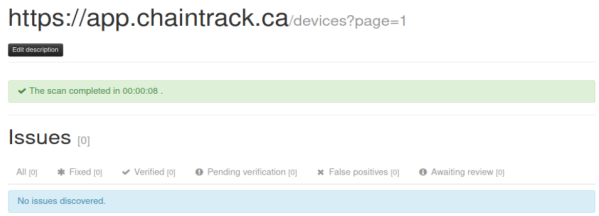




TEST with Arachni







Findings

No vulnerabilities found.

Recommendations: NA

Stored Cross Site Scripting

Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. A successful exploitation occurs when a user visits a page with a stored XSS.

Methodology

We need to identify all points where user input is stored into the back-end and then displayed by the application. Using the same methodology of a black box and grey box.

Black box:

Basic injection

aaa@aa.com”><script>alert(document.cookie)</script>
aaa@aa.com%22%3E%3Cscript%3Ealert(Pentest)%3C%2Fscript%3E

File Upload

If HTML or TXT files are allowed, XSS payload can be injected in the file uploaded. The pen-tester should also verify if the file upload allows setting arbitrary MIME types.

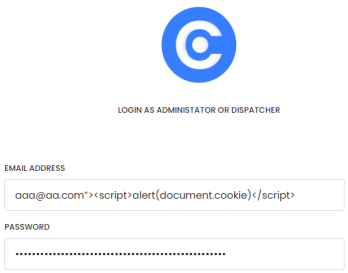
Tools

ZAP

Test

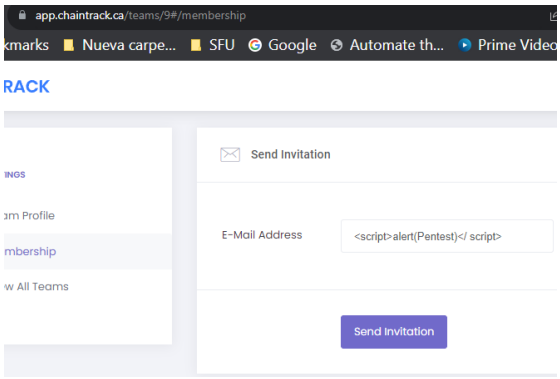
Black box:
Basic injection

URLS tested:
https://app.chaintrack.ca/login

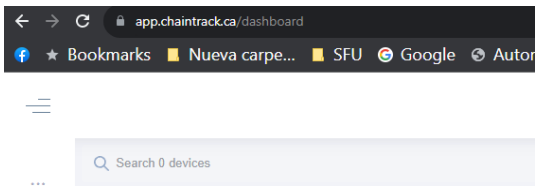


https://app.chaintrack.ca/scheduler/teams/9/orders

https://app.chaintrack.ca/teams/9#/membership



https://app.chaintrack.ca/dashboard

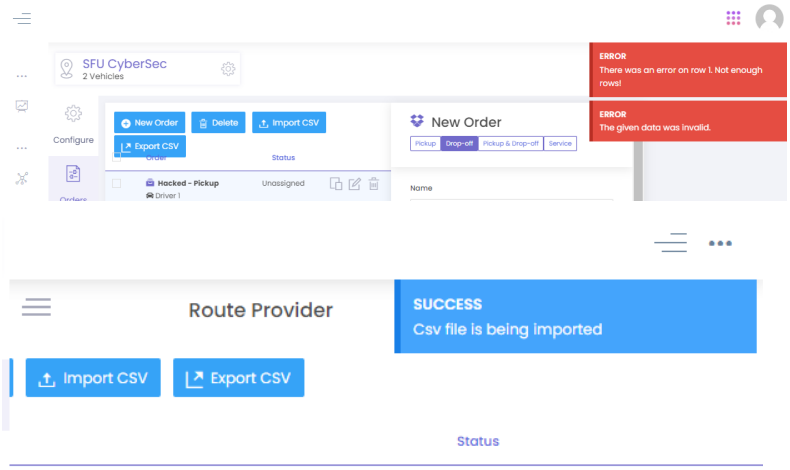


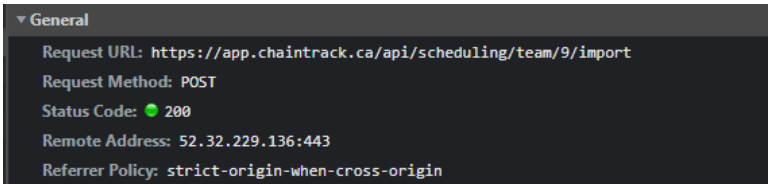
The search bar is greyed out so it is not possible to test.

Upload file.

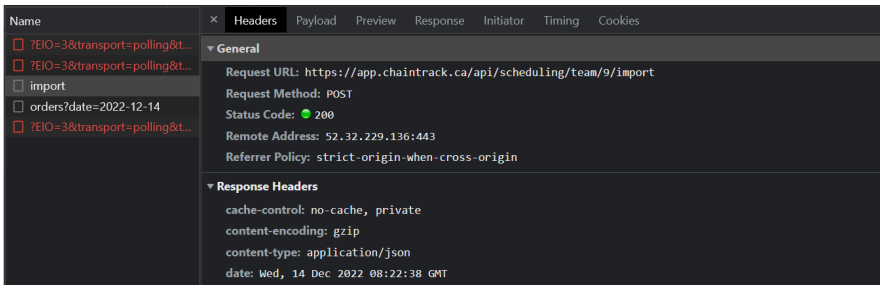
The allowed files to be uploaded are .csv files, so we have testes with different files .txt .py and .js

.txt file:

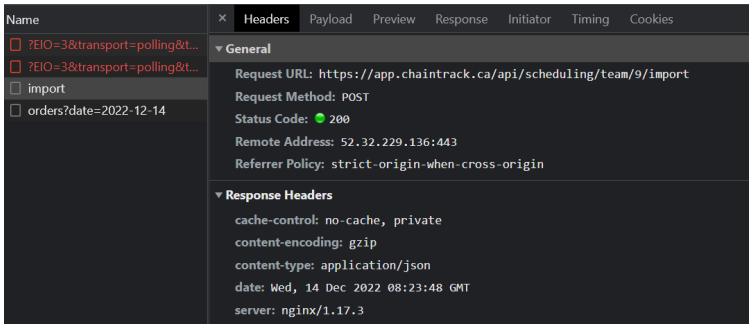




.js file:



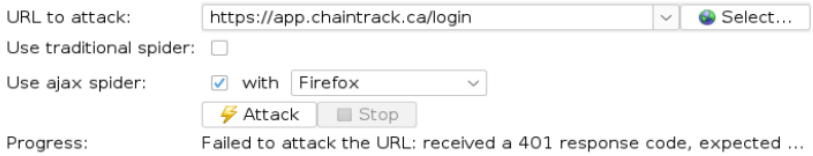
.py file:

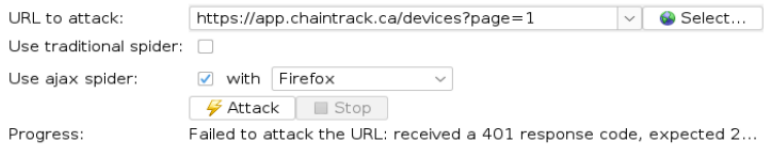


Grey box:

For the grey box testing we will be using the ZAP

ZAP





Findings

He has found that for the file upload the URL being used is https://app.chaintrack.ca/api/scheduling/team/9/import and it accepts multiple file types as long as it has more than one row in it. We also can see that we got a code 200 successful for any type of file, which by standard means that everything worked as planned, the problem here is that the uploaded files do not look to be up loading in the corresponding page of orders, not even legit .csv files containing an order; What made us guess that there is a kind of bug in the import link and then we can not determinate if this can be used for future exploits.

Recommendations

Check on the https://app.chaintrack.ca/api/scheduling/team/9/import to validate that the behaviour found is a bug or an intentional feature. If it is a bug we would recommend checking on the parameters of the accepted file types.

SQL Injection

An SQL injection attack consists of insertion or “injection” of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application.

A successful SQL injection attack can read sensitive data from the database, modify database data, execute administration operations on the db, recover content or write files into the file system, or even issue commands to the operating system.

Methodology

We need to identify when the application interacts with a DB Server in order to access some data. We also Would try to fingerprint the database. Using the same methodology of a black box and grey box.

Black box:

Standard SQL Injection

$username = 1’ or ‘1’ = ‘1
$password = 1’ or ‘1’ = ‘1
' or 1=1;--
$username = 1’ or ‘1’ = ‘1’))/*
?username=1’%20or%20 ’1’%20=%20’1&password=1’%20or%20’1’%20=%20’1
?username=1’%20or%20 ’1’%20=%20’1’))%20LIMIT%201/*&password=foo

SQL command that imposes a condition that the number of the returned results must be one.

$username = 1’ or ‘1’ = ‘1’)) LIMIT 1/*

Simple SELECT statement

in URL:
=10 AND 1=2

Fingerprinting the Database

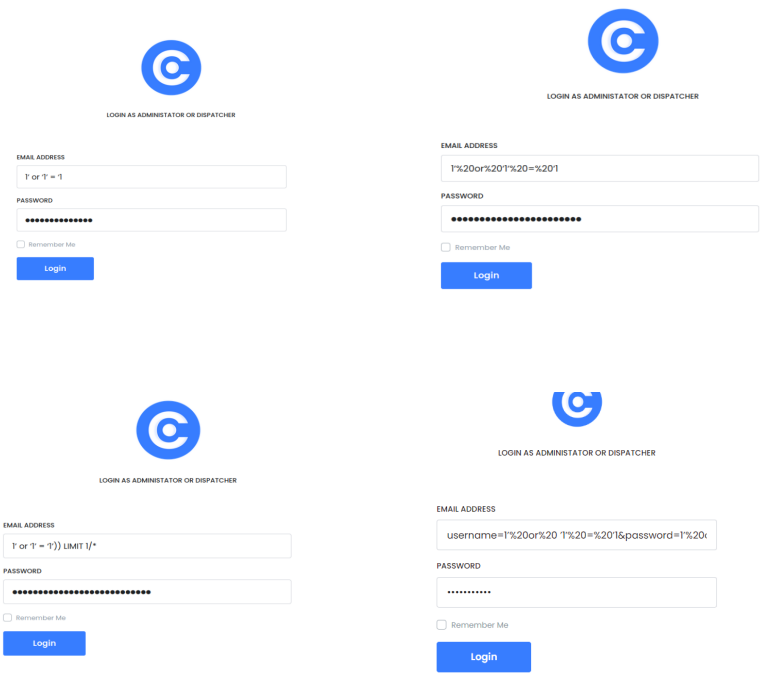
1) The first way to find out what back end database is used is by observing the error returned by the application.
2) If there is no error message or a custom error message, try to inject it into the string field using a concatenation technique.

Tools

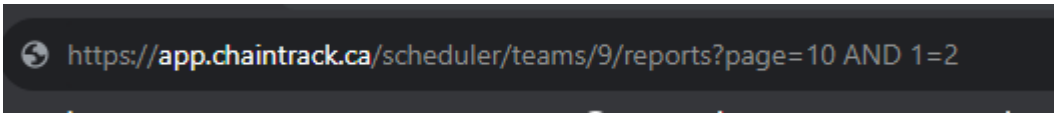
Sqlmap, burp

Test

https://app.chaintrack.ca/login



https://app.chaintrack.ca/scheduler/teams/9/reports?page=1



Fingerprinting the Database

For this task we used burp but we were not able to generate any DB error that gave us a hint on the DB used..

So we did a guess and moved to sqlmap but there is no sitemap.

Findings

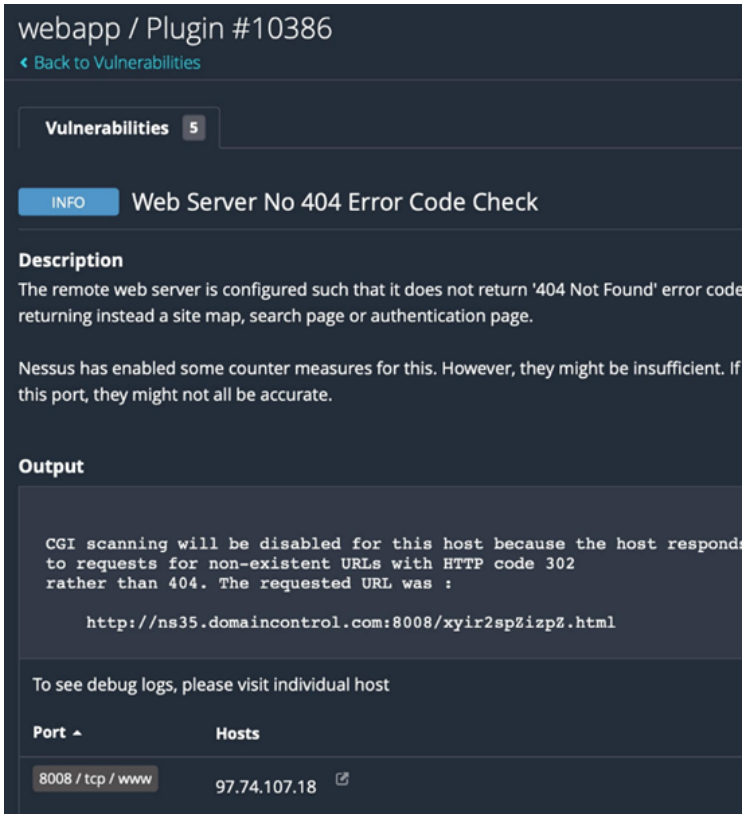
After testing the page against SQL injection we can say that this app is not vulnerable to it

Recommendations

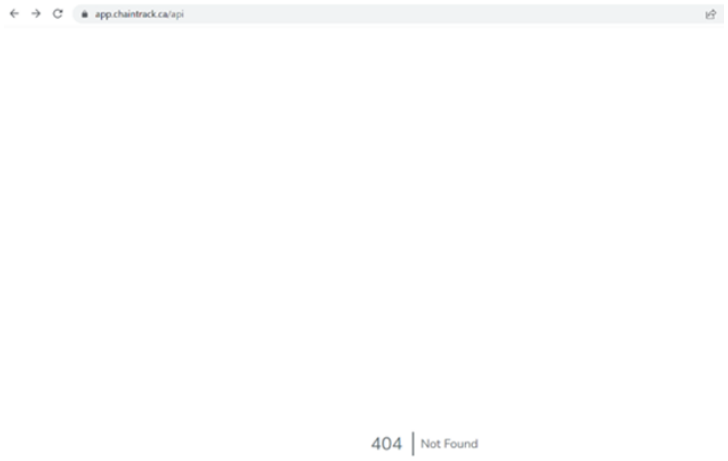
NA

7 Error Handling

Nessus is a widely used tool that helps identify network and web application vulnerabilities and security issues. It uses various techniques, such as port scans, and checks for missing patches and misconfigurations. We use Nessus to scan for any vulnerabilities and security weaknesses in the web application.



One is the 404 error code check, part of the error handling test. The result shows that the application handles the error code appropriately. And we can also use tools such as Telnet and Burp Suite to scan for any URLs that may have potential error-handling issues. In this case, we can also visit from any browser for the URLs we found in information gathering and looking for files or web pages that do not exist.



The images above show the example of our exploration, that code 403 means users do not have permission to visit and code 404 means the page does not exist. Improper error handling can allow attackers to understand the internal use of APIs and gather versions and types of applications.

8 Weak Cryptography

Testing for Weak Transport Layer Security

Test Objectives

- Validate the service configuration.
- Review the digital certificate's cryptographic strength and validity.
- Ensure that the TLS security is not bypassable and is properly implemented across the application.

Steps and Results

Identifying SSL services with Nessus:

Output

```
TLSv1.2 is enabled and the server supports at least one cipher.
```

Port ▲	Hosts
443 / tcp / www	app.chaintrack.ca

Identifying weak cipher with https://www.ssllabs.com/projects/index.html

Low

Cipher Suites

 # TLS 1.2 (suites in server-preferred order)

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)	ECDH x25519 (eq. 3072 bits RSA)	FS		256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)	ECDH x25519 (eq. 3072 bits RSA)	FS		256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)	ECDH x25519 (eq. 3072 bits RSA)	FS	WEAK	256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)	ECDH x25519 (eq. 3072 bits RSA)	FS	WEAK	256

Identifying SSL Certificate validation with sslscan:

```
  SSL Certificate:
Signature Algorithm: sha256WithRSAEncryption
ECC Curve Name:      secp384r1
ECC Key Strength:    192

Subject:  app.chaintrack.ca
Altnames: DNS:app.chaintrack.ca
Issuer:   R3

Not valid before: Dec 11 14:42:13 2022 GMT
Not valid after:  Mar 11 14:42:12 2023 GMT
```

Recommendations

The server should disable the TLS 1.2 cipher suites TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 and TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, which are considered weak.

9 Business Logic

Testing business logic includes thinking in unconventional methods and checking what happens if the user does not act in the normal, intended process. The hard part of this section is that tools are incapable of detecting logical vulnerabilities. As a result, for this part, we didn't use any tool other than Burp, and it mostly included intercepting requests and changing the destination and body of the request. As a result, the structure of this part is a little bit different, and the most detrimental vulnerability is also found in this part.

This testing section includes tests for Data validation, Unexpected file type upload, Defense against application misuse, and Ability to forge requests. Some of the tests in this section overlap with the previous sections. For example, uploading files is tested in the previous section and the results have been shared there.

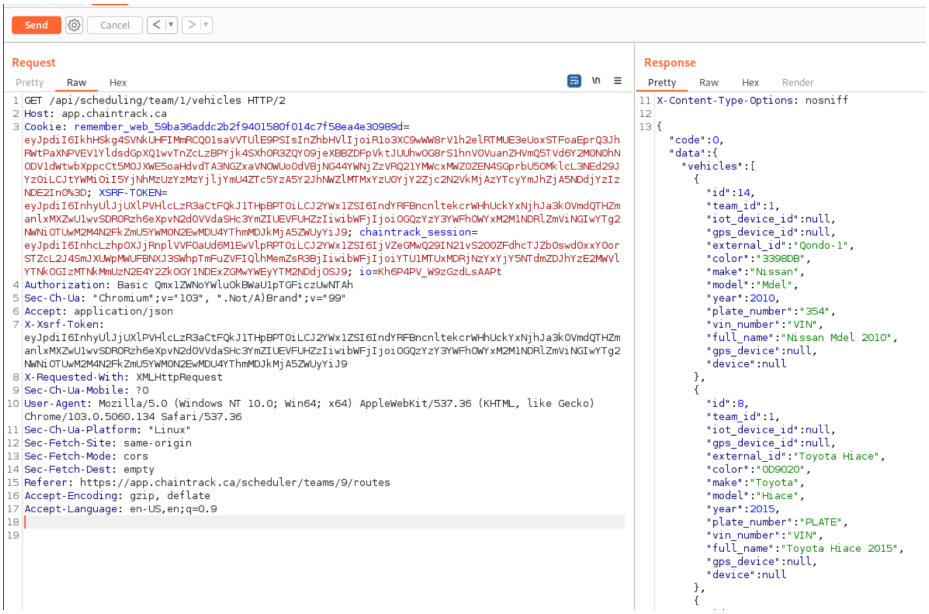
Teams APIs

While testing APIs related to teams, it could be seen that deleting or editing teams is returning a 403 Forbidden status. By forging the requests, we tried different bodies and team numbers but our requests were successfully blocked, which shows correct behavior for these APIs.

Vehicles APIs

Medium

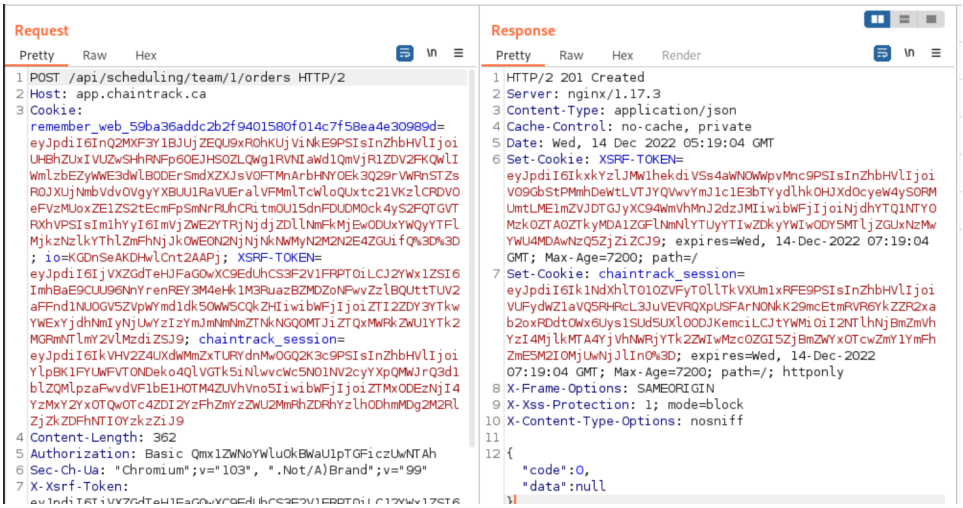
By using get API for vehicles, the list of the other teams' vehicles can be retrieved.



Orders APIs

Critical

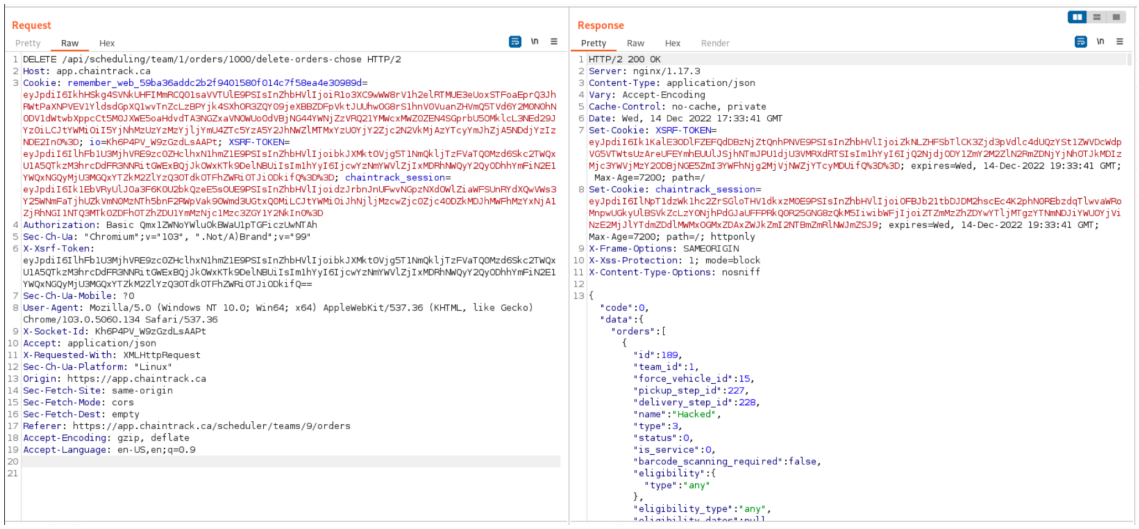
The bigger problem happens in this part. Trying to get the orders for another team, results in a 404 Not Found response. It shows that users' of the teams cannot reach the orders of other teams. But by trying to create an order for another team, you will get a 201 Created response:

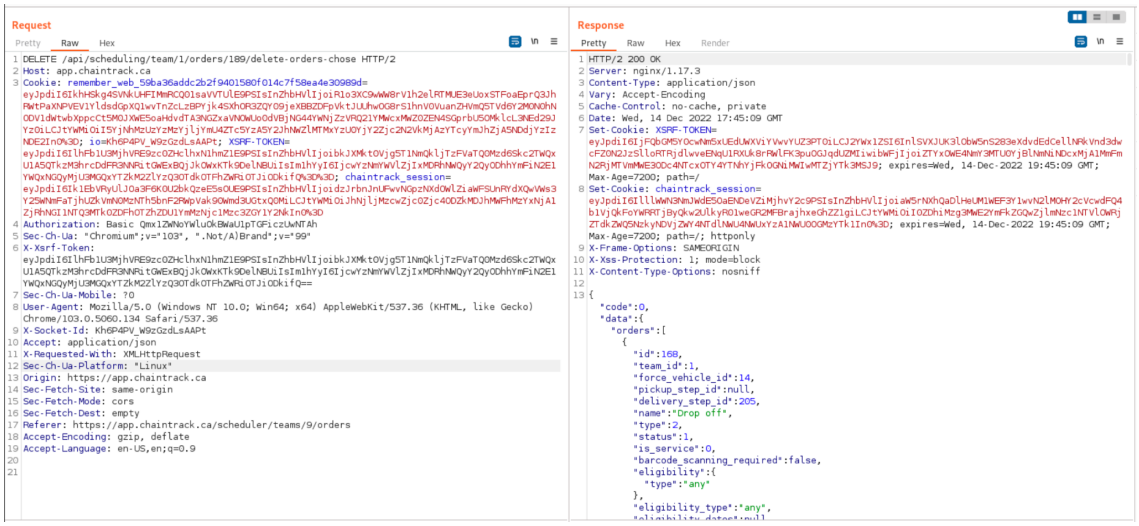


As we couldn't see the orders of other teams at this step, for checking that the order is really created we used 2 approaches: the first one was using the export API vulnerability explained earlier. By using that, it can be seen that the order is successfully created. The other problem is that the order id is incremental among different teams. For example, if the latest order id is 183, by creating an order for another team and then creating an order for our team, the new order id was 185, showing that order creation for the other team was successful. It even proved to us more concrete after finding the vulnerability of delete order API.

The first problem with delete order API is that by calling it even with a non-existing order id, it will return a 200 OK status with a complete list of existing orders. It is a wrong status code as the act of deleting didn't happen here. But the biggest problem is that by calling this API for another team, it returns all the orders of that team. Using this problem, it could be seen clearly that we were able to create an order for another team. The vulnerability doesn't end here, and as it could be expected, deleting orders of other teams is also possible.

By using the mentioned vulnerabilities, a user in team x, can get a complete list of orders of team y, create fake orders for them, and even delete whatever order for team y. This counts as a critical problem for this service.





Recommendations

This problem can be easily fixed by using proper access control for users. It seems that for getting orders everything works fine, and by applying the same policy on creating and deleting orders the problems can be fixed.

10 Client Side

Client Side URL Redirect

Also known as Open Redirection. It is an input validation flaw that exists when an application accepts an user controlled input which specifies a link that leads to an external URL that could be malicious.

A phishing attack example could be the following:
http://www.target.site?#redirect=www.fake-target.site

The victim that visits target.site will be automatically redirected to fake-target.site where an attacker could place a fake page to steal the victim's credentials.

Methodology

Following with previous methodologies we found that Black box testing for Client Side URL Redirect is not usually performed since access to the source code is always available as it needs to be sent to the client to be executed.

We will be using Burp for the grey box testing:

1. Spider target site
2. Filter sitemap by status code such as 3xx [Redirection]
3. Analysis results , modify and scan

Tools

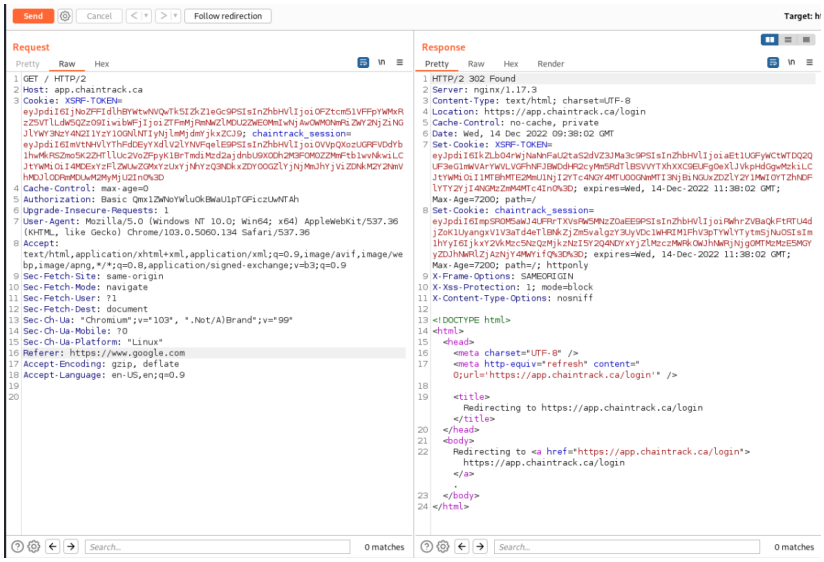
Burp

Test

1. Spider target site
2. Filter sitemap by status code such as 3xx [Redirection]

222	https://app.chaintrack.ca	GET	/api/teams/9/invitations			200	1409	JSON	
223	https://app.chaintrack.ca	GET	/api/teams/roles			200	942	JSON	
1	https://app.chaintrack.ca	GET	/			302	1343	HTML	Redirecting to https://app...
14	https://app.chaintrack.ca	POST	/login	✓		302	1365	HTML	Redirecting to https://app...

3. Analysis results , modify and scan



Findings

We have tried to modify the redirects without a successful result so we can determine that it is not vulnerable to Client Side URL Redirect/Open Redirection.

Recommendations

NA

Clickjacking

Clickjacking is a malicious technique that consists of deceiving a web user into interacting (in most cases by clicking) with something different to what the user believes they are interacting with. The term “Clickjacking” was coined by Jeremiah Grossman and Robert Hansen in 2008.

Methodology

We will use the help of burp and The Burp Clickbandit to test.
From burp we go to the Burp menu and select Burp Clickbandit. Then use the following steps:

1. Click the Copy Clickbandit to clipboard button. This will copy the Clickbandit script to your clipboard.
2. In your browser, visit the web page to test.
3. Open the web developer console.
4. Paste the Clickbandit script into the web developer console, and run it.

Tools

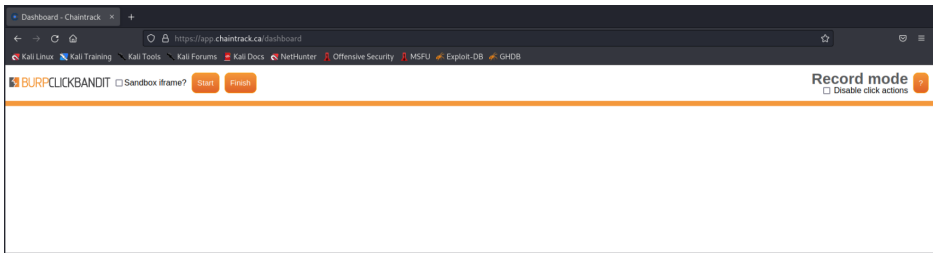
BurpSuite

Test

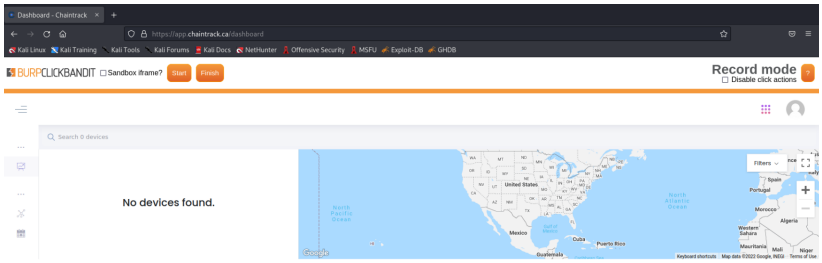
From burp we go to the Burp menu and select Burp Clickbandit. Then use the following steps:

1. Click the Copy Clickbandit to clipboard button. This will copy the Clickbandit script to your clipboard.
2. In your browser, visit the web page to test.
3. Open the web developer console.
4. Paste the Clickbandit script into the web developer console, and run it.

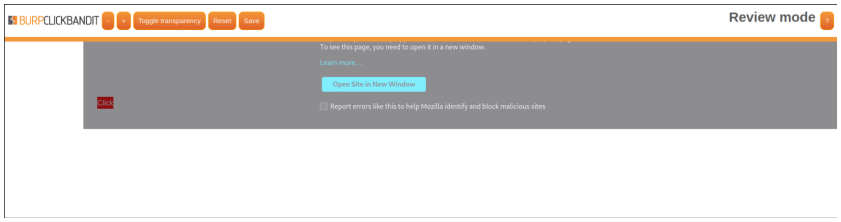
After this The Burp Clickbandit banner will appear at the top of the browser window and the original page will be reloaded within a frame, ready for the attack to be performed.



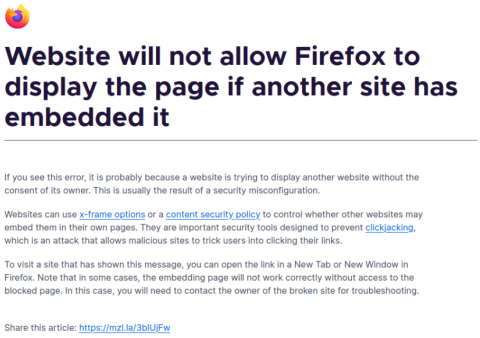
While recording we need to navigate and perform the steps that we want the user to do. Once it is over we click on finish.



Then we should be able to see the page in order to configure where we want to locate our button.



In this case the real page does not load and when we check on the error we got the following message.



Website will not allow Firefox to display the page if another site has embedded it

If you see this error, it is probably because a website is trying to display another website without the consent of its owner. This is usually the result of a security misconfiguration.

Websites can use x-frame options or a content security policy to control whether other websites may embed them in their own pages. They are important security tools designed to prevent clickjacking, which is an attack that allows malicious sites to trick users into clicking their links.

To visit a site that has shown this message, you can open the link in a New Tab or New Window in Firefox. Note that in some cases, the embedding page will not work correctly without access to the blocked page. In this case, you will need to contact the owner of the broken site for troubleshooting.

Share this article: https://mzl.la/3blUjFw

Findings

We can see that this page is not vulnerable to Clickjacking.

Recommendations

NA.