Penetration Test Report - ChainTrack

December 16th, 2021

Mufeng Huang, Jiazhou Liu, and Jiacheng Wu

Simon Fraser University, Burnaby, BC, Canada

Table of Contents

Executive Summary

Overview

ChainTrack is a business that focuses on providing a delivery tracking system for dispatchers in Vancouver. Dispatchers are provided access to the website where they can create driver accounts and initiate the delivery process and monitor its progress. The product owner wants our team to analyze the application security of the web application. This is a report of our findings.

Objectives

Our deliverable will be a security report which includes detailed information about vulnerabilities we found, as well as their potential impact from both technical and business aspects. In addition, we will also try to provide solutions to eliminate those vulnerabilities inside the system. In general, our report focuses on analyzing the vulnerabilities and their influences, which can help the product owner to better understand the system and make decisions from a business point of view.

Scope And Limitations

We begin with defining a scope that can be agreed upon between the product owner and the team. The scope includes the entire chaintrack.ca domain. Analysis of the mobile app and IoT devices are optional. The scope may be limited in what is allowed to be analyzed: certain penetration activity such as running/testing an exploit may be limited as it can have devastating impact on the production environment.

One limitation is that we are unable to perform the IoT API endpoints due to it being in maintenance for an unspecified length of time. We have brought this to the product owner's attention but due to other factors, the product owner was unable to help us resolve this particular issue.

Summary Of Results

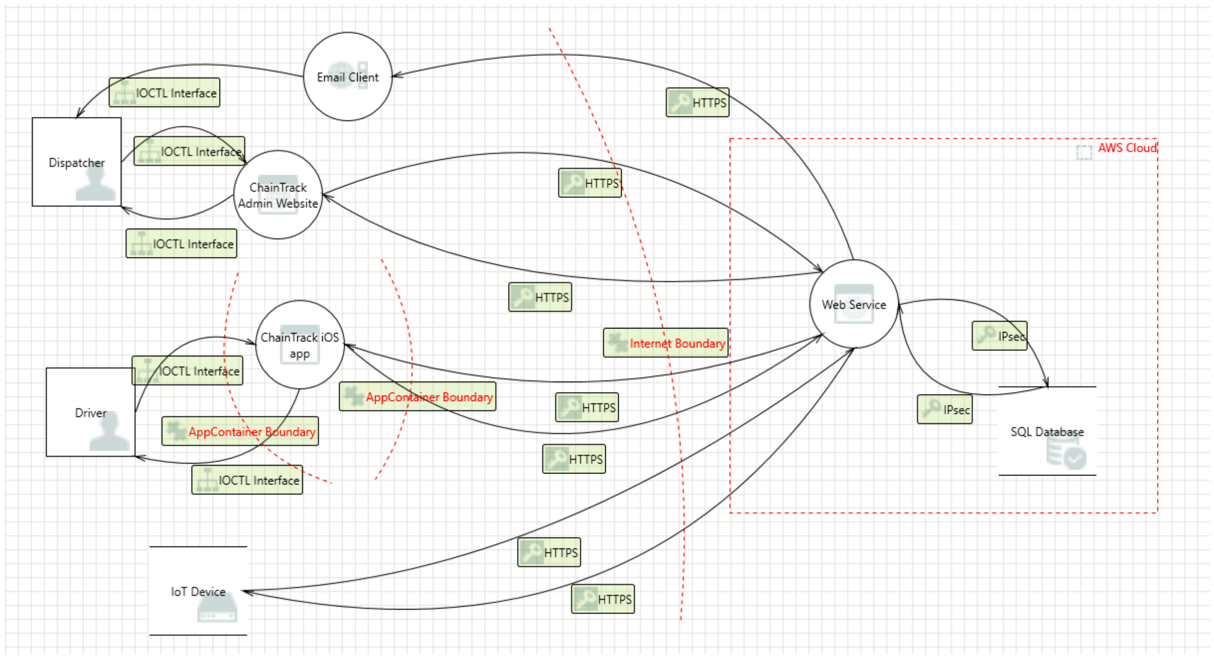
Using the STRIDE threat model and other best practice methodologies, we found vulnerabilities ranging from low to high. We also have recommendations and solutions to address these vulnerabilities.

Methodology

We began our analysis using OSINT tools. We use the domain that was provided to us as the starting point. The information gathered helped us create an appropriate threat model.

Threat Modelling

We believe that using a threat model is the ideal way to make an impactful analysis without wasting effort in areas that may not be relevant to the project. We use the STRIDE model developed by Microsoft (https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats). The acronym stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. Using the Microsoft Threat Model Tool that is made by the same vendor, we are able to construct a high-level model and generate threats based on our input. Details of our threat analysis are found in Appendix D.



Risk Rating Model

We are using a risk model proposed by OWASP (https://owasp.org/www-community/OWASP_Risk_Rating_Methodology). The method of calculating the risk involves a matrix comparing the likelihood of the risk with the impact of the risk. Both the likelihood and impact contain 8 criteria. The ranking of each criterion ranges from 1 to 9 where 1 is a low risk and 9 is the highest risk. Then the score of the likelihood and the impact is the average of all criteria. The risk rating is reduced to five possible results based on the matrix shown below. All calculations of the vulnerabilities are found in Appendix C.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Detail Findings

In this section, we highlight some of the most important findings. Some are done using tools which we add details of their usage in Appendix A and other findings are done through manual testing and analysis. For findings of vulnerabilities, we added details in Appendix B. Below we present them in no particular order except that the categories are sorted by the STRIDE acronym. Not all threats have vulnerabilities; this shows that some security standards are being adhered.

Each section may contain two sub sections. Testing is for testings that passed security checks and vulnerabilities found is for testing where we found a vulnerability.

Spoofing

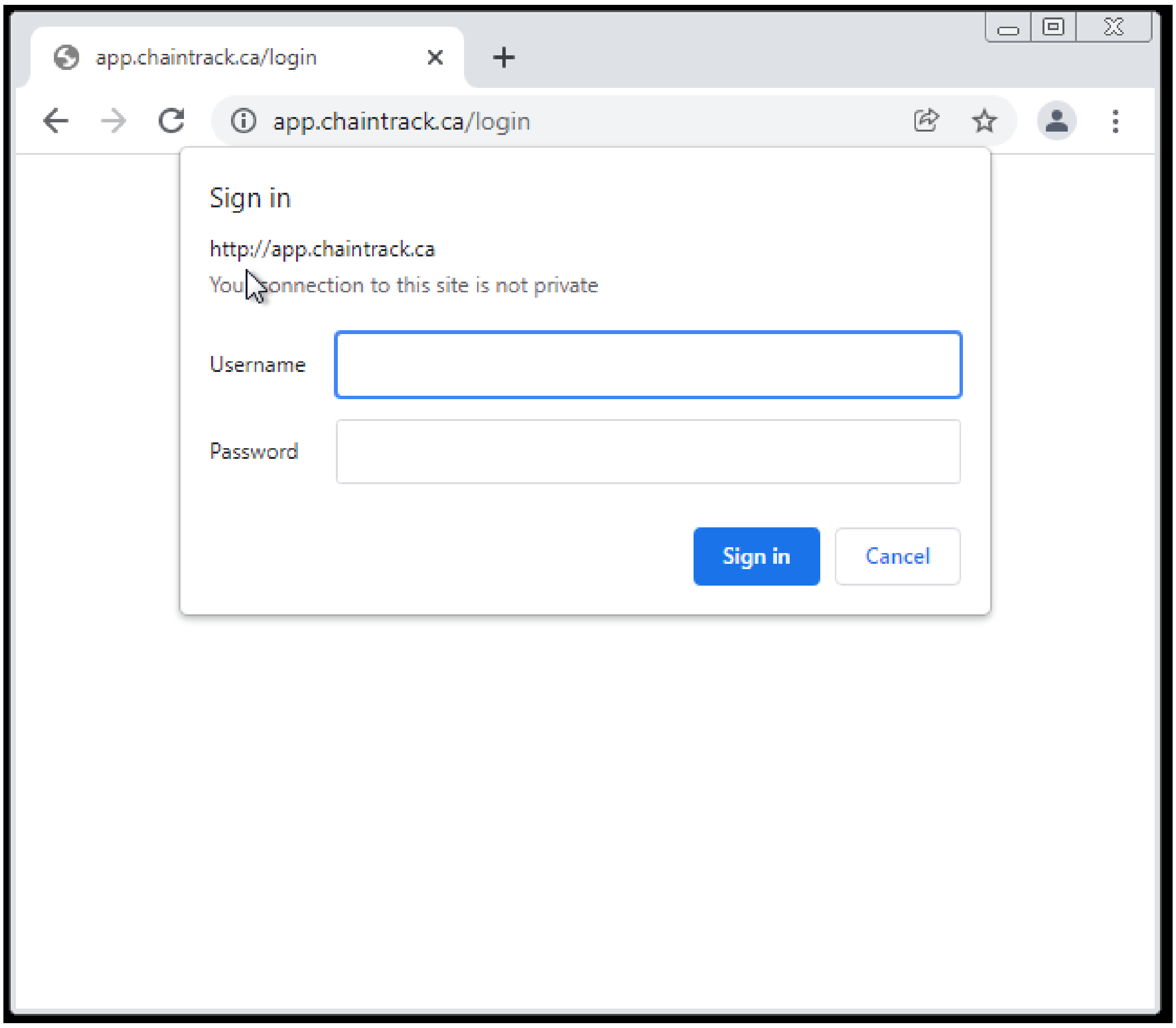
Spoofing the Website and ChainTrack iOS App

The ChainTrack Website and ChainTrack iOS app may be spoofed by an attacker and this may lead to unauthorised access to ChainTrack Website and its web service.

.

Vulnerabilities Found

Man-in-the-middle - The lack of HSTS header allows the man-in-the-middle attack to spoof the HTTP version of the website. More details of this are discussed in the information disclosure section. Below is the HTTP version of the website after the spoofing. One note is that the HTTP website does not work properly after the login. However, since the credentials are sent in plaintext to the server, attackers can replay it in the real https website.


app.chaintrack.ca/login

Sign in

http://app.chaintrack.ca

Your connection to this site is not private

Username

Password

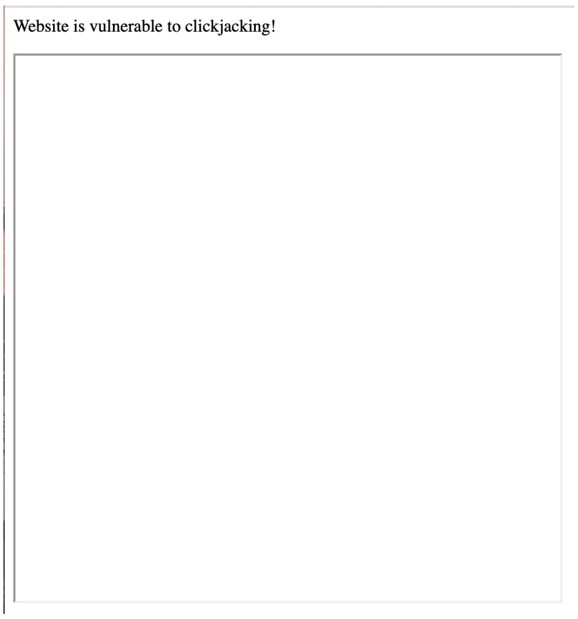
Weak password policy - We observed that the password reset page allows entry of password without any special criteria. The only requirement is a password length of 8 characters. In the worst case scenario, the user can select a password that is found in popular dictionaries used in brute forcing exploits. We also observed that the same weakness is found for the driver creation page.

Clickjacking

Clickjacking is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page.

Vulnerabilities Found

Missing X-Frame-Options - The ChainTrack website doesn’t return an X-frame-Options header, which means this website could be at risk of a clickjacking attack.


Website is vulnerable to clickjacking!

We create a sample html file where the Chaintrack website is put inside an iframe. Although the website doesn't show up in this case, it may still be vulnerable to more sophisticated attacks. We suggest that the X-frame-Options header should be set just in case.

Spoofing the email

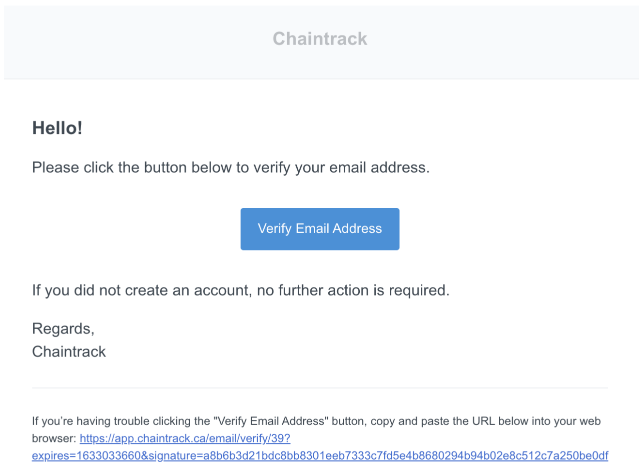
An outside attacker may impersonate the official website and send emails soliciting for client details.

Scenario

A forged email is sent to our dispatcher account email, with the source code presented as below:

```
<p style="color:#3d4852; line-height:1.5em; margin-top:0; text-align:left; font-size:12px">
If you're having trouble clicking the "Verify Email Address" button, copy and paste the URL below into your web browser:
<a href="https://this.fake.com" style="color:#3869d4">
https://app.chaintrack.ca/email/verify/39?expires=1633033660&amp;signature=a8b6b3d21bdc8bb8301eeb7333c7fd5e4b8680294b94b02e8c512c7a250be0df</a></p>
</td>
```

The email appears as below, but both link actually points to a fake website:


Chaintrack

Hello!

Please click the button below to verify your email address.

Verify Email Address

If you did not create an account, no further action is required.

Regards,
Chaintrack

If you're having trouble clicking the "Verify Email Address" button, copy and paste the URL below into your web browser: https://app.chaintrack.ca/email/verify/39?expires=1633033660&signature=a8b6b3d21bdc8bb8301eeb7333c7fd5e4b8680294b94b02e8c512c7a250be0df

Tampering

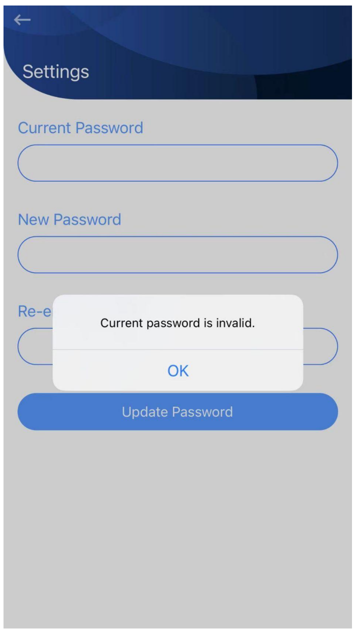
Potential SQL injection on ChainTrack Web Service and IOS

SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

Testing

For the ChainTrack Web Service, we used sqlmap to test SQL injection attacks, and the result shows no vulnerabilities.

For the ios app, the only input place is the password resetting for the driver's account. Due to a limitation of the testing environment and tools, we manually tried some of the trivial scripts for SQL, Javascript, Node.js injections and the password reset function works correctly.



Current password is invalid.

OK

Cross Site Scripting

The web server 'ChainTrack Website' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.

Testing

We tried the input <script>alert(1)</script> in random inputs inside the website and they are being properly parsed to prevent any XSS. We also noted that some cookies have the HTTP only flag. When we run the console.log(document.cookie), only the XSRF cookie shows. With the proper output parsing, XSS is not possible.

Potential Lack of Input Validation

Data flowing across the website, and iOS app may be tampered with by an attacker. This may lead to a denial of service attack against ChainTrack iOS app or an elevation of privilege attack against ChainTrack iOS app or an information disclosure by ChainTrack iOS app. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.

Testing

We tried the login using manual SQL injection as well as for the iOS app and was unable to compromise the login. Other input validations we noticed are that expected email formats will not accept any other value except valid emails. This was tested on the screen for email invitations.

Repudiation

Potential Data Repudiation by ChainTrack Website and ChainTrack iOS app

ChainTrack Website, or ChainTrack iOS app claim that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.

Testing
We manually reviewed the page for logs and found logs related to dispatching activities.

Vulnerabilities Found

No login activity history feature - We are unable to locate a page in the website that indicates a log of our login history. This is a concern as the users do not know if someone else has logged into their account.

Information Disclosure

Weak Access Control for a Resource

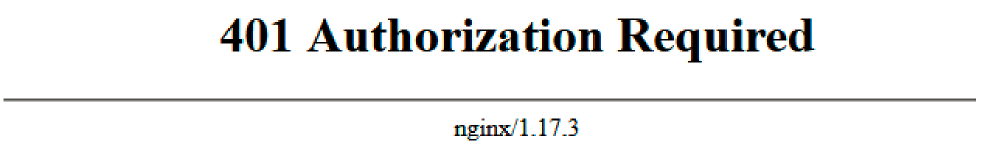
Improper data protection can cause information to be leaked via output of errors such as SQL injection, bad configuration, input validation, etc.

Testing

We tried online tools such as wappalyzer to determine the web technologies used but did not find any interesting results. Manual inspection of the view-source feature in the browser also did not bring anything interesting. There is no mention of any framework and the javascript comments did not reveal any sensitive information, or names.

Vulnerabilities Found

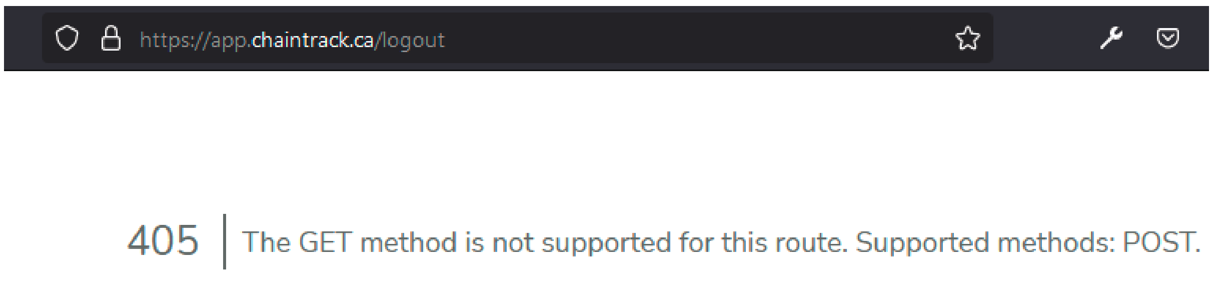
Web server information is disclosed - On the login page, providing the wrong credentials on https://app.chaintrack.ca/login caused the default error page of Nginx. We noticed that the Nginx version is also outdated by more than 24 months (1.17.3 was released on 13 Aug 2019). We are also able to detect the web server using OSINT tools.



401 Authorization Required

nginx/1.17.3

Framework potentially disclosed - When navigating to the logout page manually, we observed the following error. Clues on this page suggest that the framework used is Laravel, a PHP framework.



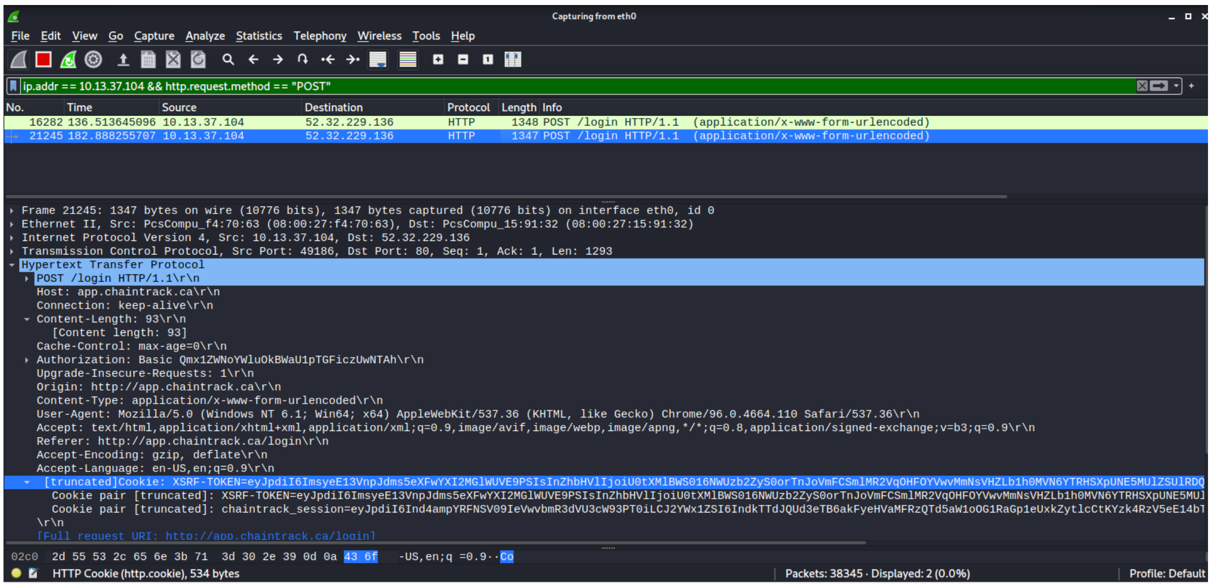
405 | The GET method is not supported for this route. Supported methods: POST.

Data Flow Sniffing

Data flowing across between the IoT, web browser, and mobile app to the web server may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations.

Vulnerabilities Found

Susceptible to SSL stripping - When we perform a man-in-the-middle attack, we are able to perform the ssl stripping on the website. This means we can force all traffic to HTTP. This also allows us to use wireshark and sniff the traffic and obtain the login information.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 10.13.37.104 && http.request.method == "POST"

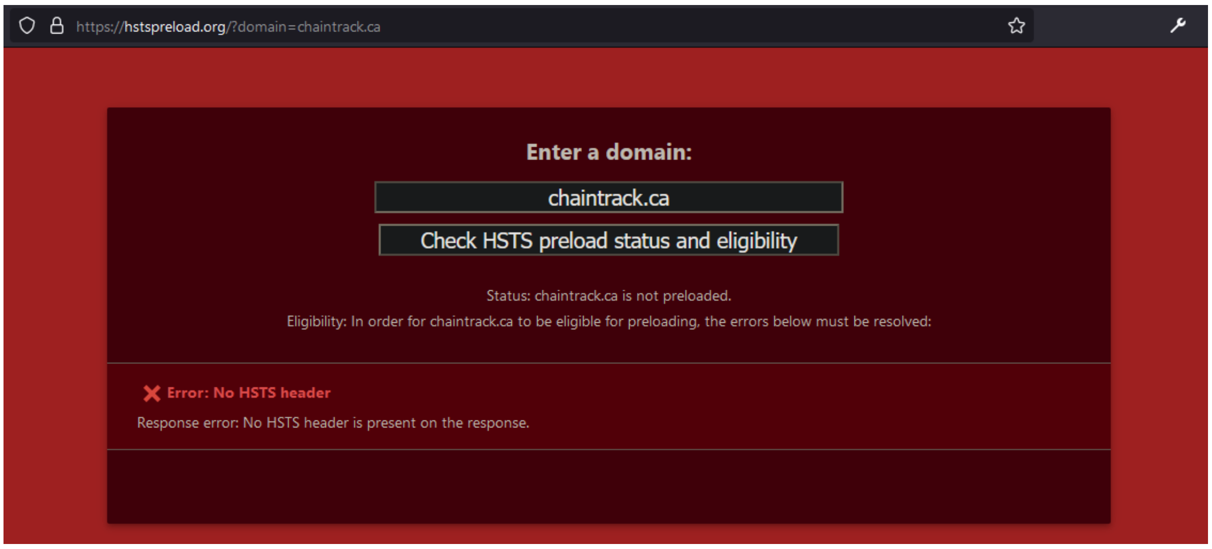
```
No.   Time           Source         Destination    Protocol Length Info
16282 136.513645096 10.13.37.104   52.32.229.136  HTTP     1348 POST /login HTTP/1.1  (application/x-www-form-urlencoded)
21245 182.888255707 10.13.37.104   52.32.229.136  HTTP     1347 POST /login HTTP/1.1  (application/x-www-form-urlencoded)
```

```
Frame 21245: 1347 bytes on wire (10776 bits), 1347 bytes captured (10776 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_f4:70:63 (08:00:27:f4:70:63), Dst: PcsCompu_15:91:32 (08:00:27:15:91:32)
Internet Protocol Version 4, Src: 10.13.37.104, Dst: 52.32.229.136
Transmission Control Protocol, Src Port: 49186, Dst Port: 80, Seq: 1, Ack: 1, Len: 1293
Hypertext Transfer Protocol
  POST /login HTTP/1.1\r\n
  Host: app.chaintrack.ca\r\n
  Connection: keep-alive\r\n
  Content-Length: 93\r\n
    [Content length: 93]
  Cache-Control: max-age=0\r\n
  Authorization: Basic Qmx1ZWNoYWluOkBWaU1pTGFiczUwNTAh\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Origin: http://app.chaintrack.ca\r\n
  Content-Type: application/x-www-form-urlencoded\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36\r\n
  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\r\n
  Referer: http://app.chaintrack.ca/login\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  [truncated]Cookie: XSRF-TOKEN=eyJpdiI6ImsyeE13VnpJdms5eXFwYXI2MGlWUVE9PSIsInZhbHVlIjoiU0tXMlBWS016NWUzb2ZyS0orTnJoVmFCSmlMR2VqOHFOYVwvMmNsVHZLb1h0MVN6YTRHSXpUNE5MUlZSUlRDQ
    Cookie pair [truncated]: XSRF-TOKEN=eyJpdiI6ImsyeE13VnpJdms5eXFwYXI2MGlWUVE9PSIsInZhbHVlIjoiU0tXMlBWS016NWUzb2ZyS0orTnJoVmFCSmlMR2VqOHFOYVwvMmNsVHZLb1h0MVN6YTRHSXpUNE5MUl
    Cookie pair [truncated]: chaintrack_session=eyJpdiI6Ind4ampYRFNSV09IeVwvbmR3dVU3cW93PT0iLCJ2YWx1ZSI6IndkTTdJQUd3eTB6akFyeHVaMFRzQTd5aW1oOG1RaGp1eUxkZytlcCtKYzk4RzV5eE14bT
  \r\n
  [Full request URI: http://app.chaintrack.ca/login]
```

```
02c0  2d 55 53 2c 65 6e 3b 71  3d 30 2e 39 0d 0a 43 6f   -US,en;q =0.9··Co
```

HTTP Cookie (http.cookie), 534 bytes | Packets: 38345 · Displayed: 2 (0.0%) | Profile: Default

The MiTM is successful due to the missing HSTS ([HTTP Strict Transport Security](https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security)) header.



https://hstspreload.org/?domain=chaintrack.ca

Enter a domain:

chaintrack.ca

Check HSTS preload status and eligibility

Status: chaintrack.ca is not preloaded.

Eligibility: In order for chaintrack.ca to be eligible for preloading, the errors below must be resolved:

Error: No HSTS header

Response error: No HSTS header is present on the response.

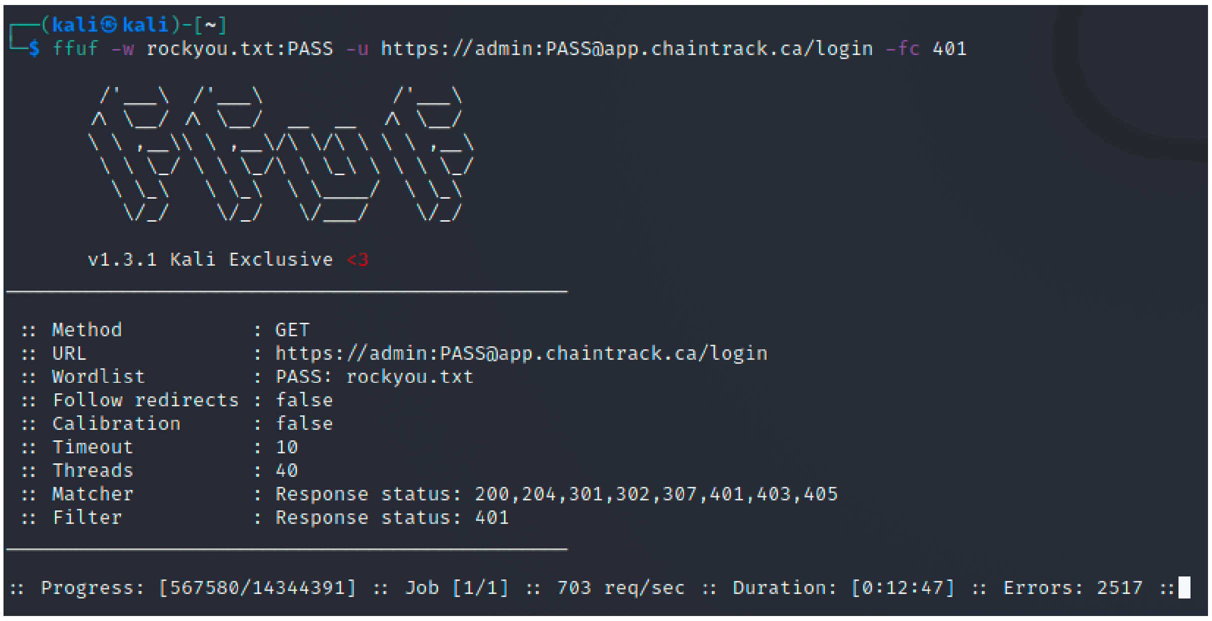
Denial Of Service

Potential Process Crash or Stop for Web Service and ChainTrack iOS app

Web Service and/or ChainTrack iOS crash, halt, stop or run slowly; in all cases violating an availability metric.

Vulnerabilities Found

Potentially Missing Intrusion Detection - When performing URL enumeration and fuzzing on login endpoints, we do not experience any connection interruption. We are able to perform up to 500 thousand attempts. We stopped the attempt around this range as we do not want to cause damage to the production environment.



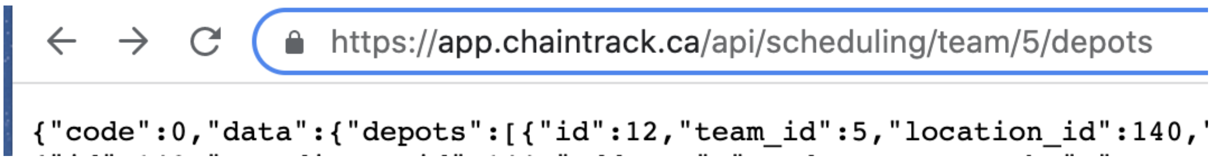
```
┌──(kali㉿kali)-[~]
└─$ ffuf -w rockyou.txt:PASS -u https://admin:PASS@app.chaintrack.ca/login -fc 401

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.3.1 Kali Exclusive <3
________________________________________________

 :: Method           : GET
 :: URL              : https://admin:PASS@app.chaintrack.ca/login
 :: Wordlist         : PASS: rockyou.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
 :: Filter           : Response status: 401
________________________________________________

:: Progress: [567580/14344391] :: Job [1/1] :: 703 req/sec :: Duration: [0:12:47] :: Errors: 2517 ::
```

Elevation Of Privilege

An attacker may pass data into ChainTrack Website or ChainTrack iOS app in order to change the flow of program execution within ChainTrack Website to the attacker's choosing.

Vulnerabilities Found

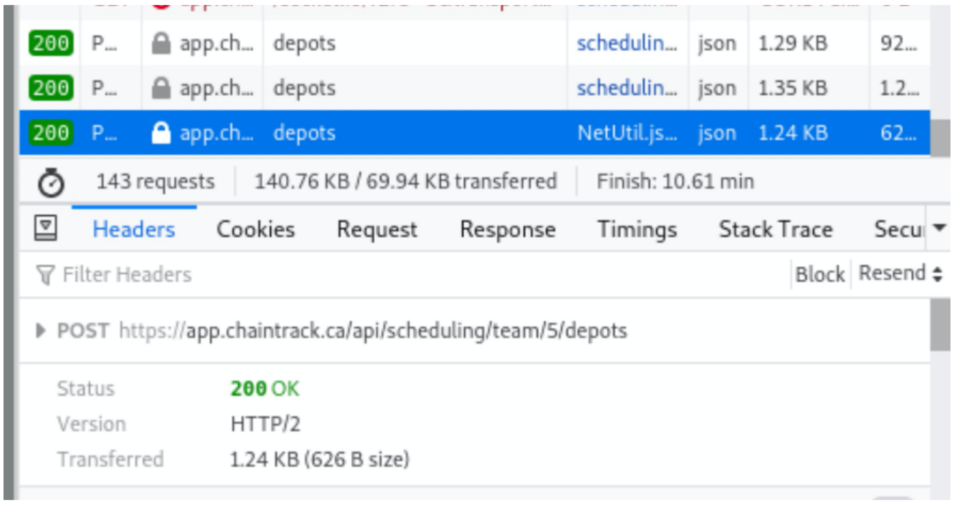
URL Manipulation - We try to pass data into the ChainTrack website, and found that the access control of */api/scheduling/team/#/* is not strictly controlled. The test account is team 7 and so we confirm that we are not allowed to access */teams/5* (return ERROR) or */scheduler/teams/5* (Redirected to team/7). However, under the */api/scheduling/team/#/*, we are able to get/post/delete data from other teams.
To prove that, we access */api/scheduling/team/5/depots* and confirm that, with a session of team 7, we can get data that is controlled by team 5.



https://app.chaintrack.ca/api/scheduling/team/5/depots

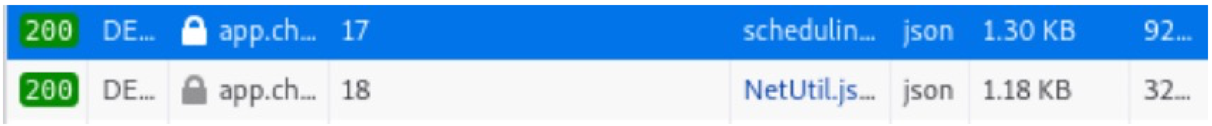
```
{"code":0,"data":{"depots":[{"id":12,"team_id":5,"location_id":140,'
```

we execute the web service ui to create a new depot with the testing account. From the network, we find that a POST request with payload is sent to */api/scheduling/team/7/depots*. We simply modify the request URL and payload before we resend it as POST to */api/scheduling/team/5/.* We get a 200 return and confirm that we can see the data we add in team 5's depots.



```
200  P...  app.ch...  depots    schedulin...  json  1.29 KB  92...
200  P...  app.ch...  depots    schedulin...  json  1.35 KB  1.2...
200  P...  app.ch...  depots    NetUtil.js... json  1.24 KB  62...
143 requests   140.76 KB / 69.94 KB transferred   Finish: 10.61 min
Headers  Cookies  Request  Response  Timings  Stack Trace  Secu
Filter Headers                                   Block  Resend
POST https://app.chaintrack.ca/api/scheduling/team/5/depots
Status       200 OK
Version      HTTP/2
Transferred  1.24 KB (626 B size)
```

Note from the screenshot, the request sent by NetUtil.js is the modified one. The DELETE request works as well.



```
200  DE...  app.ch...  17  schedulin...  json  1.30 KB  92...
200  DE...  app.ch...  18  NetUtil.js... json  1.18 KB  32...
```

We notice from other teams' data that these teams are for demo and testing, therefore, probably this access is allowed only in the same group (for example, with same First-layer credentials). However, this still means that someone with a dispatcher account is able to obtain and modify the information under the control of other dispatchers in the same group.

Conclusion

Recommendations

In general, we found that adding security headers such as Strict-Transport-Security, X-Frame-Options etc. is an important fix for many security problems. In addition, adding intrusion detection is a significant task to prevent exploitations from attackers in time. In terms of security on users' side, a stronger password complexity requirement can prevent most credential guessing from attackers. Finally, we recommend implementing customised error pages to prevent leak of critical system information, which can be gathered for preparing attacks.

Risk Rating

The overall risk rating score is a medium risk based on the vulnerabilities we found. Overall, the ChainTrack has some good security measures implemented to prevent any serious compromise. Full detail of the risk ratings is in Appendix C.

Appendix A: Results From Tools

Tool: Maltego CE 4.2

When using maltego, the URL (https://app.chaintrack.ca/login) is used as the starting point.

- IP address of app.chaintrack.ca is 52.32.229.136
- Name server: ns36.domaincontrol.com
- Searching the domain for owner details does not yield any results (details are redacted).
- Searching the domain for email gives back legal@chaintrack.ca
- Searching the domain for footprint L3 gives a great number of new nodes. The search found another subdomain and DNS Names:
    - www.chaintrack.ca - 52.32.229.136
    - mail.chaintrack.ca - 107.180.2.102
    - webmail.chaintrack.ca - 107.180.2.102

Tool: Spiderfoot 3.0

The command to start the tool: sudo spiderfoot -l xxxxxxxxxxx:80 where xxxxxxxxxxxx is the attacker's ip address.

- Target used is 52.32.229.136 with passive mode
    - Found docs.chaintrack.ca
    - Hosting provider: Amazon AWS: http://www.amazon.com/aws/
    - Server detected: nginx/1.17.3
- Target app.chaintrack.ca with passive mode
    - Nothing new
    - Code repositories with the name chaintrack appears to have no relationship

Tool: ffuf

Purpose of this tool is to try to find other user/password combinations for the basic authentication. The command used was

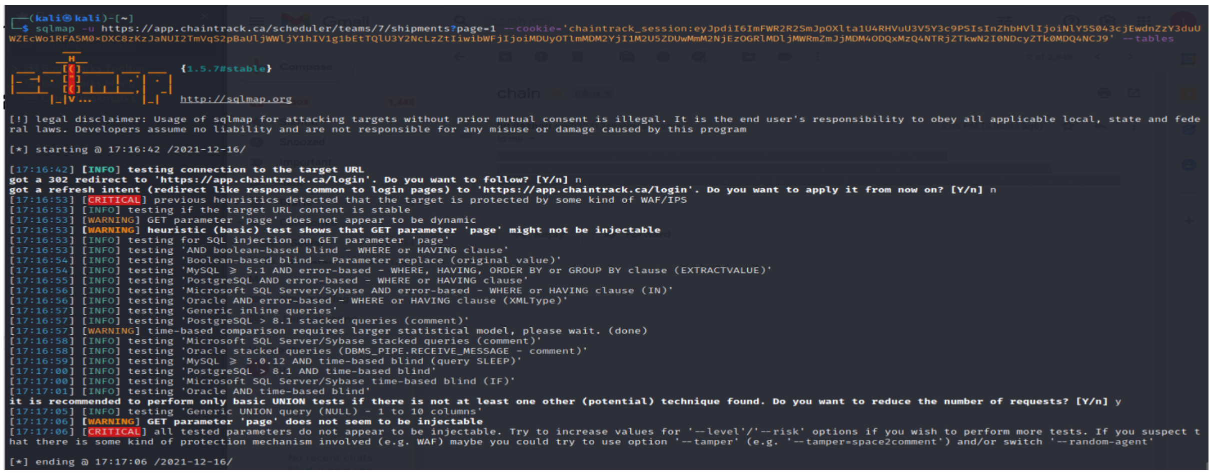
```
ffuf -w /usr/share/wordlists/metasploit/password.lst:PASS -w
/usr/share/wordlists/wfuzz/others/names.txt:USER -u
https://USER:PASS@app.chaintrack.ca/login -fc 401
```

- The basic auth was not practical as this can take ~48 days
- Shorter list was later used: /usr/share/wordlists/wfuzz/others/common_pass.txt but no common passwords or users were found.

Tool: Sqlmap

Purpose of this tool is to find vulnerabilities for potential SQL Injection. The command used was sqlmap -u https://app.chaintrack.ca/scheduler/teams/7/shipments?page=1 --cookie LEAKED_COOKIE –tables

- With leaked cookies, attackers can perform exploration for SQL Injection vulnerabilities.
- According to the result, no SQL Injection vulnerabilities have been found.

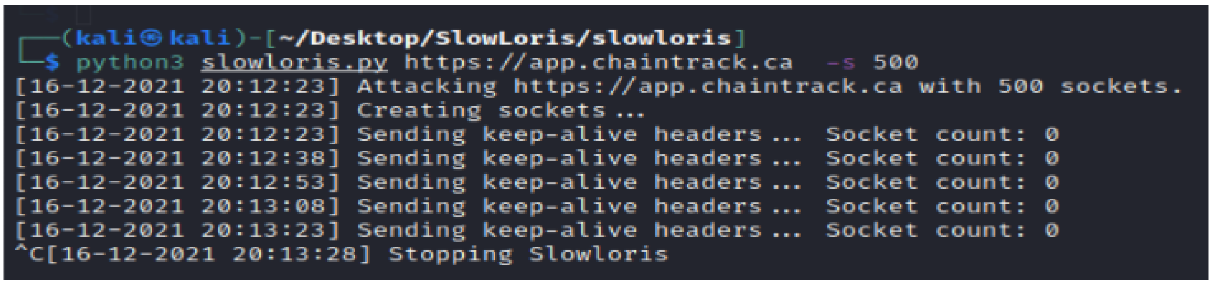


```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u https://app.chaintrack.ca/scheduler/teams/7/shipments?page=1 --cookie='chaintrack_session:eyJpdiI6ImFWR2R2SmJpOXlta1U4RHVuU3V5Y3c9PSIsInZhbHVlIjoiNlY5S043cjEwdnZzY3duU
WZEcWo1RFA5M0×DXC8zKzJaNUI2TmVqS2pBaUljWWljY1hIV1g1bEtTQlU3Y2NcLzZtIiwibWFjIjoiMDUyOTlmMDM2YjI1M2U5ZDUwMmM2NjEzOGRlMDljMWRmZmJjMDM4ODQxMzQ4NTRjZTkwN2I0NDcyZTk0MDQ4NCJ9' --tables
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5.7#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and fede
ral laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:16:42 /2021-12-16/

[17:16:42] [INFO] testing connection to the target URL
got a 302 redirect to 'https://app.chaintrack.ca/login'. Do you want to follow? [Y/n] n
got a refresh intent (redirect like response common to login pages) to 'https://app.chaintrack.ca/login'. Do you want to apply it from now on? [Y/n] n
[17:16:53] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
[17:16:53] [INFO] testing if the target URL content is stable
[17:16:53] [WARNING] GET parameter 'page' does not appear to be dynamic
[17:16:53] [WARNING] heuristic (basic) test shows that GET parameter 'page' might not be injectable
[17:16:53] [INFO] testing for SQL injection on GET parameter 'page'
[17:16:53] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:16:54] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[17:16:54] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[17:16:55] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[17:16:56] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[17:16:56] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[17:16:57] [INFO] testing 'Generic inline queries'
[17:16:57] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[17:16:57] [WARNING] time-based comparison requires larger statistical model, please wait. (done)
[17:16:58] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[17:16:58] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[17:16:59] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[17:17:00] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[17:17:00] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[17:17:01] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[17:17:05] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[17:17:06] [WARNING] GET parameter 'page' does not seem to be injectable
[17:17:06] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect t
hat there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

[*] ending @ 17:17:06 /2021-12-16/
```

Tool: Slowloris

The goal of using this tool is to simulate a DDOS attack on app.chaintrak.ca. It will create many open socket connections to overload the server. The command used was python3 slowloris.py https://app.chaintrack.ca -s 500

- Slowloris failed to create any socket connections, and the reason can be the firewall which blocks the excessive connections to prevent DDOS attack.
- Chaintrack web service is robust against DDOS attacks and no vulnerability has been found.



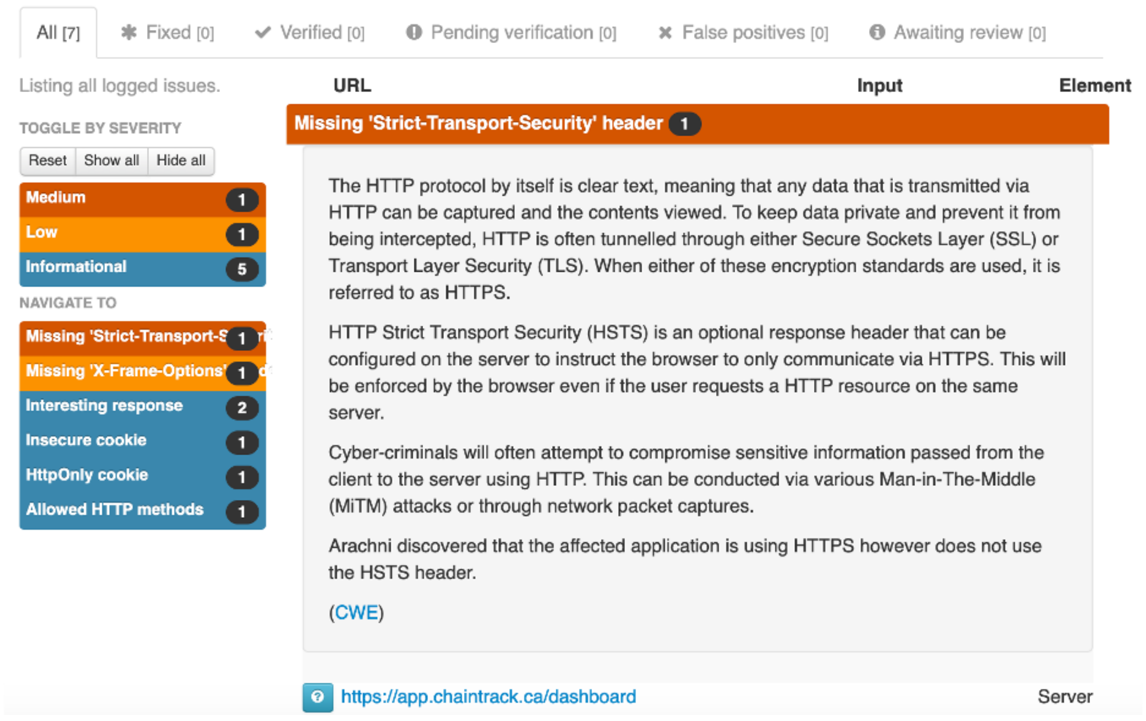
```
┌──(kali㉿kali)-[~/Desktop/SlowLoris/slowloris]
└─$ python3 slowloris.py https://app.chaintrack.ca -s 500
[16-12-2021 20:12:23] Attacking https://app.chaintrack.ca with 500 sockets.
[16-12-2021 20:12:23] Creating sockets...
[16-12-2021 20:12:23] Sending keep-alive headers... Socket count: 0
[16-12-2021 20:12:38] Sending keep-alive headers... Socket count: 0
[16-12-2021 20:12:53] Sending keep-alive headers... Socket count: 0
[16-12-2021 20:13:08] Sending keep-alive headers... Socket count: 0
[16-12-2021 20:13:23] Sending keep-alive headers... Socket count: 0
^C[16-12-2021 20:13:28] Stopping Slowloris
```

Tool: Arachni

Arachni is used to conduct reconnaissance to find vulnerabilities of websites. In this project, we use Arachni to scan through Chaintrack websites and find important vulnerabilities.

- Missing ‘Strict-Transport-Security’ header causes sensitive data leak through HTTP connection.

- Missing 'X-Frame-Options' header can lead to potential clickjacking attacks.



All [7] | Fixed [0] | Verified [0] | Pending verification [0] | False positives [0] | Awaiting review [0]

Listing all logged issues.

TOGGLE BY SEVERITY

Reset | Show all | Hide all

Medium	1
Low	1
Informational	5

NAVIGATE TO

Missing 'Strict-Transport-S	1
Missing 'X-Frame-Options'	1
Interesting response	2
Insecure cookie	1
HttpOnly cookie	1
Allowed HTTP methods	1

URL | Input | Element

Missing 'Strict-Transport-Security' header 1

The HTTP protocol by itself is clear text, meaning that any data that is transmitted via HTTP can be captured and the contents viewed. To keep data private and prevent it from being intercepted, HTTP is often tunnelled through either Secure Sockets Layer (SSL) or Transport Layer Security (TLS). When either of these encryption standards are used, it is referred to as HTTPS.

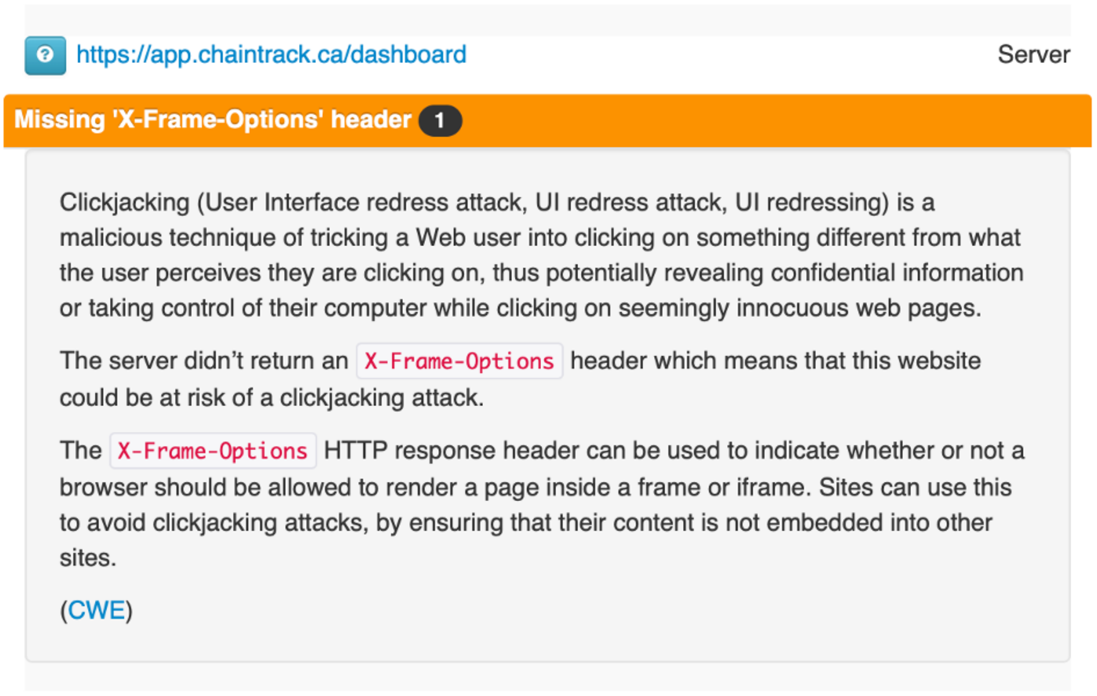
HTTP Strict Transport Security (HSTS) is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. This will be enforced by the browser even if the user requests a HTTP resource on the same server.

Cyber-criminals will often attempt to compromise sensitive information passed from the client to the server using HTTP. This can be conducted via various Man-in-The-Middle (MiTM) attacks or through network packet captures.

Arachni discovered that the affected application is using HTTPS however does not use the HSTS header.

(CWE)

https://app.chaintrack.ca/dashboard Server



https://app.chaintrack.ca/dashboard Server

Missing 'X-Frame-Options' header 1

Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

The server didn't return an `X-Frame-Options` header which means that this website could be at risk of a clickjacking attack.

The `X-Frame-Options` HTTP response header can be used to indicate whether or not a browser should be allowed to render a page inside a frame or iframe. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

(CWE)

Appendix B: Vulnerability Detail and Mitigation

Vulnerability:	Web server information is disclosed and outdated
Rating:	**High**
Description:	We found the web server and the version. The version of the server is at least 2 years old.
Impact:	Using the web server information, an attacker can search for known exploits and it takes minimal effort to compromise the server once a critical exploit exists.
Remediation:	Create a custom auth error page that does not reveal any information about the webserver. For more information, see this tutorial on how to customise error page https://www.digitalocean.com/community/tutorials/how-to-configure-nginx-to-use-custom-error-pages-on-ubuntu-14-04 Also check https://www.inmotionhosting.com/support/server/nginx/hide-nginx-server-version/ for a tutorial on removing web server's name from HTTPS response headers.

Vulnerability:	Man-in-the-middle/Data sniffing
Rating:	**Medium**
Description:	Both exploits are caused by the same vulnerability where there is no HSTS header.
Impact:	An attacker that has access to the same network can perform this attack and steal the credentials and other sensitive data.
Remediation:	We recommend the site owner to consider registering the HSTS record to be added to Google Chrome's HSTS list. Other browsers borrow from this list. Also the HSTS header must be added in order to qualify for the listing. Refer to the following link https://hstspreload.org/?domain=chaintrack.ca

Vulnerability:	Potentially Missing Intrusion Detection
Rating:	**High**
Description:	A lack of intrusion detection allows the attacker to make numerous attempts to scan/or use up resources.

Impact:	Failure to block failed logins and other similar behaviours makes it easy for attackers to try other enumerations and fuzzing techniques that will further expose other possible vulnerabilities.
Remediation:	We recommend adding a Fail2ban rule for too many failed login attempts. This can be configured along with Nginx. See the following link: https://www.nginx.com/blog/dynamic-ip-denylisting-with-nginx-plus-and-fail2ban/

Vulnerability:	Weak Password Policy
Rating:	**Medium**
Description:	The password policy only validates that the length is 8 characters.
Impact:	Strong passwords are not enforced. Legitimate users are likely to abuse this policy as even the word “password” or “12345678” is accepted. This leaves accounts potentially vulnerable to simple guesses.
Remediation:	This involves a change of the policy to validate that the password includes at least one special character, one lower case, one upper case character and one number along with the minimum length. Also consider a multi-factor authentication scheme using tokens.

Vulnerability:	No login activity history feature
Rating:	**Notice**
Description:	There is no page in the website to show the login history of the user.
Impact:	Users are unable to tell if their account is compromised if an attacker manages to log into their account. This can lead to users having doubts about the security of the website.
Remediation:	This requires a development on the website to have a page created showing the timestamp of the login and the IP Address. Other useful information can include the user agent of the browser and the geographic location based on the given IP Address.

Vulnerability:	Missing X-frame-Options
Rating:	**Medium**
Description:	X-Frame-Options is missing from header

Impact:	Potential allows javascript injection that is possible due to unrestricted CORS policy
Remediation:	Simply add the header and enforce the same origin policy

Vulnerability:	URL Manipulation
Rating:	**High**
Description:	Changing some values in the URL exposes data not belonging to the user
Impact:	An attacker can use this to gain information and even potential gain privilege as there are not restriction validation
Remediation:	The logic needs to check that the user has permission to the resource and it is the owner of the resource before returning the results.

Appendix C Risk Analysis

Legend

We use the 5 colour schemes to indicate the severity of the risk/risk rating:

- Magenta - Critical
- Red - High
- Orange - Medium
- Yellow - Low
- Blue - Notice

Name of Vulnerability	**Threat Agent Factors**				**Vulnerability Factors**				**Likelihood Score**	**Technical Impact**				**Business Impact**				**Impact Score**	**Final Score**
	Skill level	**Motivation**	**Opportunity**	**Size**	**Ease of Discovery**	**Ease of Exploit**	**Awareness**	**Intrusion Detection**		**Loss of Confidentiality**	**Loss of Integrity**	**Loss of Availability**	**Loss of Accountability**	**Financial damage**	**Reputation damage**	**Non-compliance**	**Privacy violation**		
Web server information is disclosed and outdated	3	4	9	9	9	9	9	9	7.625	2	2	1	9	7	1	2	9	4.125	🟥 HM
Man-in-the-middle/Data	9	9	1	2	3	3	4	8	4.875	6	1	1	9	7	4	2	3	4.125	🟧 MM

sniffing																			🟧
Potentially Missing Intrusion Detection	9	9	9	9	9	9	6	9	8.625	6	1	9	7	7	1	2	9	5.25	HM
Weak Password Policy	1	4	9	9	3	9	1	8	5.5	9	1	1	7	1	1	5	2	3.375	ML
No login activity history feature	1	1	9	2	3	1	1	8	3.25	2	1	1	1	1	1	2	3	1.5	LL
Missing X-frame-Options	4	5	9	6	7	5	4	8	6	6	1	1	9	1	4	2	3	3.375	ML
URL Manipulation	3	9	7	6	3	9	4	3	5.5	9	9	1	7	7	9	9	9	7.5	MH

Appendix D: Threats Analysis

Title	Category	Description
Potential Process Crash or Stop for Web Service and ChainTrack iOS app	Denial Of Service	Web Service and/or ChainTrack iOS crash, halt, stop or run slowly; in all cases violating an availability metric.
Web Service May be Subject to Elevation of Privilege Using Remote Code Execution	Elevation Of Privilege	ChainTrack Website and ChainTrack iOS app may be able to remotely execute code for Web Service.
Elevation Using Impersonation	Elevation Of Privilege	An attacker may impersonate a legitimate user in order to gain additional privilege.
Elevation by Changing the Execution Flow	Elevation Of Privilege	An attacker may pass data into ChainTrack Website or ChainTrack iOS app in order to change the flow of program execution within ChainTrack Website to the attacker's choosing.
Weak Access Control for a Resource	Information Disclosure	Improper data protection can cause information to be leaked via output of errors such as SQL injection, bad configuration, input validation, etc.
Data Flow Sniffing	Information Disclosure	Data flowing across between the IoT, web browser, and mobile app to the web server may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations.
Dispatchers getting phishing emails	Information Disclosure	Attackers may use phishing to obtain information from dispatchers.
Potential Data Repudiation by ChainTrack Website and ChainTrack iOS app	Repudiation	ChainTrack Website, or ChainTrack iOS app claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
External Entity Driver Potentially Denies Receiving Data	Repudiation	Driver claims that it did not receive data from a process on the other side of the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data.
Spoofing the Website and ChainTrack iOS App	Spoofing	The ChainTrack Website and ChainTrack iOS app may be spoofed by an attacker and this may lead to unauthorised access to ChainTrack Website and its web service.
Spoofing of Source Data Store IoT Device	Spoofing	IoT Devices may be spoofed by an attacker and this may lead to incorrect data delivered to Web Service. This may also lead to data being written to the attacker's target instead of IoT Device.
Spoofing the Email	Spoofing	An outside attacker may use phishing to

		impersonate the website and send emails soliciting for client details.
Potential SQL Injection Vulnerability for SQL Database	Tampering	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.
Cross Site Scripting	Tampering	The web server 'ChainTrack Website' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input.
Potential Lack of Input Validation	Tampering	Data flowing across the website, and iOS app may be tampered with by an attacker. This may lead to a denial of service attack against ChainTrack iOS app or an elevation of privilege attack against ChainTrack iOS app or an information disclosure by ChainTrack iOS app. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues. Consider all paths and the way they handle data. Verify that all input is verified for correctness using an approved list input validation approach.