BUB BOUNTY

IT NUMBER: IT22345332

NAME: G.P DINUJAYA THAMARA

WEEKEND BATCH

MALABE CAMPUS

Bug Bounty Platform – Hacker One

Bug Bounty Program - Booking.com

Scope

In Scope Assets

For in Scope Assets please refer to the Scope tab

Out-Of-Scope Applications Any application whether owned by Booking.com or third-party vendor **not included as an in-scope asset** will be mentioned on the scope tab as out of scope.

For Out Of Scope Assets please refer to the Scope tab

In-scope Vulnerabilities

Accepted, in-scope vulnerabilities include, but are not limited to:

- Disclosure of sensitive or personally identifiable information
- Cross-Site Scripting (XSS) - Please note, for XSS if the same issue is reported for the different subdomains but with the same root cause, it will be considered duplicate
- Cross-Site Request Forgery (CSRF) for sensitive functions in a privileged context
- Remote code execution (RCE)
- Authentication or authorization flaws, including insecure direct object references and authentication bypass
- Injection vulnerabilities, including SQL and XML injection
- Directory traversal
- Significant security misconfiguration with a verifiable vulnerability
- Account takeover by exploiting a vulnerability

- SSRF
- XXE
- Subdomain takeover in *.booking.com domains

Out-Of-Scope Vulnerabilities Depending on their impact, not all reported issues may qualify for a monetary reward. However, all reports are reviewed on a case-by-case basis and any report that results in a change being made will at a minimum receive recognition. Please note that our **program terms and rules of engagement** still apply.

The following issues are outside the scope of our vulnerability rewards program:

- Any vulnerability which requires access to a compromised email account or Booking.com account for successful exploitation
- Vulnerabilities on Third Party Products
- Attacks requiring physical access to a user's device or network.
- Forms missing CSRF tokens (we require evidence of actual CSRF vulnerability)
- Login/Logout CSRF
- Missing security headers which do not lead directly to a vulnerability
- Use of a known-vulnerable library (without evidence of exploitability)
- Reports from automated tools or scans
- Social engineering of Booking staff or contractors
- Denial of Service attacks and/or reports on rate limiting issues
- Not enforcing certificate pinning
- Any issues that require a rooted or jailbroken device or a compromised device
- Clickjacking
- Improper session invalidation
- User enumeration
- Host header injections without a specific, demonstrable impact
- Self-XSS, which includes any payload entered by the victim

- Any vulnerabilities requiring significant and unlikely interaction by the victim, such as disabling browser controls
- Content spoofing without embedded HTML or JavaScript
- Hypothetical issues that do not have any practical impact
- Infrastructure vulnerabilities, including:
- Issues related to SSL certificates
- DNS configuration issues
- Server configuration issues (e.g. open ports, TLS versions, etc.)

Asset name	Type	Coverage	Max. severity	Bounty	Last update
https://iphone-xml.booking.com/json/	URL	In scope	Critical	Eligible	Nov 29, 2023
https://secure-iphone-xml.booking.com/json/	URL	In scope	Critical	Eligible	Dec 13, 2023
supplier.auth.toag.booking.com	Domain	In scope	Critical	Eligible	Jan 24, 2023
metasearch-api.booking.com	Domain	In scope	Critical	Eligible	Nov 7, 2023
experiences.booking.com	Domain	In scope	Critical	Eligible	Nov 7, 2023
webhooks.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023
paybridge.booking.com	Domain	In scope	Critical	Eligible	Dec 13, 2023
phone-validation.taxi.booking.com	Domain	In scope	Critical	Eligible	Dec 13, 2023
autocomplete.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023
distribution-xml.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023
paynotifications.booking.com	Domain	In scope	Critical	Eligible	Dec 13, 2023
supply-xml.booking.com	Domain	In scope	Critical	Eligible	Dec 13, 2023
accommodations.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023
portal.taxi.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023
secure-supply-xml.booking.com	Domain	In scope	Critical	Eligible	Nov 29, 2023

*.booking.com if there's any vulnerabilities raised on this asset that are owned by a third party we will not be accepting those reports	Wildcard	In scope	Critical	Eligible	Feb 29, 2024
www.booking.com/bbmanage/data/*	Wildcard	Out of scope	None	Ineligible	Mar 19, 2024
spadmin.booking.com/	Domain	Out of scope	None	Ineligible	Mar 19, 2024
www.booking.com/bbmanage/*	Wildcard	Out of scope	None	Ineligible	Mar 19, 2024
secure.booking.com/company/*	Wildcard	Out of scope	None	Ineligible	Mar 19, 2024
secure.booking.com/orgnode/*	Wildcard	Out of scope	None	Ineligible	Mar 19, 2024
business.booking.com/	Domain	Out of scope	None	Ineligible	Mar 19, 2024
https://fareharbor.com/demo/	URL	Out of scope	None	Ineligible	Mar 19, 2024
https://www.booking.com/bbm.html	URL	Out of scope	None	Ineligible	Mar 19, 2024

https://cars.booking.com

- Alerts (16)
  - Absence of Anti-CSRF Tokens
  - Content Security Policy (CSP) Header Not Set
  - Missing Anti-clickjacking Header
  - Cookie No HttpOnly Flag (5)
  - Cookie Without Secure Flag (6)
  - Cookie with SameSite Attribute None
  - Cookie without SameSite Attribute (5)
  - Cross-Domain JavaScript Source File Inclusion (21)
  - Timestamp Disclosure - Unix (2)
  - Content Security Policy (CSP) Report-Only Header Found
  - Information Disclosure - Suspicious Comments (3)
  - Loosely Scoped Cookie (2)
  - Modern Web Application
  - Re-examine Cache-control Directives
  - Session Management Response Identified (2)
  - User Agent Fuzzer (24)

Absence of Anti-CSRF Tokens

URL: https://cars.booking.com
Risk: Medium
Confidence: Low
Parameter:
Attack:
Evidence:
```
<form action="https:&#47;&#47;www.booking.com&#47;newslettersubscribe.html"
method="post"
name="newsletterform"
id="emk-footer"
class="footerForm emk-subscription-entry-point "
data-component="emk/subscription-entry-point emk/subscription-entry-point-feedback-msg"
data-emk-entry-point-label="footer"
data-ga4-track="newsletter_sign_up"
>
```
CWE ID: 352
WASC ID: 9
Source: Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:

No Anti-CSRF tokens were found in a HTML submission form. A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.

CSRF attacks are effective in a number of situations, including: * The victim has an active session on the target site. * The victim is authenticated via

HTTP auth on the target site. * The victim is on the same local network as the target site.

CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy.

Solution

Phase: Architecture and Design Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. Phase: Implementation Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script. Phase: Architecture and Design Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). Note that this can be bypassed using XSS. Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. Note that this can be bypassed using XSS. Use the ESAPI Session Management control. This control includes a component for CSRF. Do not use the GET method for any request that triggers a state change. Phase: Implementation Check the HTTP Referer header to see if the request originated from an expected page. This could

break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Normally Anti-CSRF tokens are used to protect against cross-site request forgery attacks. This post explains the basics of CSRF tokens and shows how to use them to protect the users of your websites and applications against CSRF.

Anti-CSRF token basics

The idea behind anti-CSRF tokens (aka synchronizer token patterns or simply CSRF tokens) is give the user's browser a piece of information (a token) that the browser then has to send back. The token must be unique and impossible to guess by a third party, and the application must only process HTTP requests once the token has been verified. This ensures that only the original user can send requests within an authenticated session.

For a basic example without CSRF protection, say you run a web application on *www.example.com*. To publish a message on their profile in the app, a user completes an HTML form and clicks the *Submit* button:

```
<form action="/action.php" method="post">
  Subject: <input type="text" name="subject"/><br/>
  Content: <input type="text" name="content"/><br/>
  <input type="submit" value="Submit"/>
</form>
```

The submit action causes the web browser to send a POST request to the server, with whatever data the user entered being sent as parameters.

```
POST /post.php HTTP/1.1
Host: example.com

subject=I am feeling well&content=I just ate a cookie and it was delicious
```

If the user is logged in and the attacker knows the request syntax, it may be possible to use a CSRF attack to publish an ad on that user's profile:

```
<form action="/action.php" method="post">
  Subject: <input type="text" name="subject" value="Buy my product!"/>
  Content: <input type="text" name="content" value="To buy my product, visit this site: example.biz."/>
  <input type="submit" value="Submit"/>
</form>
<script>
  document.forms[0].submit();
</script>
```

As a result, the web browser sends the following POST request:

```
POST /post.php HTTP/1.1
Host: example.com

subject=Buy my product!&content=To buy my product, visit this site: example.biz.
```

On an unprotected page, this could achieve CSRF if the server treats the forged request as coming from an authenticated user.

But now let's say your site uses simple token-based CSRF mitigation, and your web server sets the token in a session cookie sent to the browser right after login. All the form submissions then include a hidden field containing the token. Assuming proper token validation, this completely eliminates the CSRF vulnerability:

```
<form>
  Subject: <input type="text" name="subject"/><br/>
  Content: <input type="text" name="content"/><br/>
  <input type="submit" value="Submit"/>
  <input type="hidden" name="token" value="R6B7hoBQd0wfG5Y6qOXHPNm4b9WKsTq6Vy6Jssxb"/>
</form>
```

The server should then only accept POST requests from the same user that include this exact token value, for example:

```
POST /post.php HTTP/1.1
Host: example.com

subject=I am feeling well&content=I just ate a cookie and it was
delicious.&token=R6B7hoBQd0wfG5Y6qOXHPNm4b9WKsTq6Vy6Jssxb
```

With this protection in place, an attacker who tries to perform CSRF using a malicious site cannot fake HTTP requests without knowing the current token set in the valid user's cookie. Because your server rejects all requests without this token, any attack attempts will fail.

https://www.invicti.com/blog/web-security/protecting-website-using-anti-csrf-token/

Content Security Policy (CSP) Header Not Set

URL: https://cars.booking.com
Risk: Medium
Confidence: High
Parameter:
Attack:
Evidence:
CWE ID: 693
WASC ID: 15
Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference: 10038-1
Input Vector:

Description:

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Other Info:

Content Security Policy (CSP) is a security feature that helps prevent code injection attacks by defining and enforcing a whitelist of approved content sources. It does this by defining a policy. If the CSP header is not set

correctly, attackers can inject malicious scripts into your web application, leading to potential data theft, or unauthorized access.

https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html this is a cheat sheet for content security policy.

We can prevent this by defining the CSP correctly.

Missing Anti-clickjacking Header

URL:	https://cars.booking.com
Risk:	Medium
Confidence:	Medium
Parameter:	x-frame-options
Attack:	
Evidence:	
CWE ID:	1021
WASC ID:	15
Source:	Passive (10020 - Anti-clickjacking Header)
Alert Reference:	10020-1
Input Vector:	

Description:

The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Cookie No HttpOnly Flag

URL:	https://cars.booking.com/
Risk:	Low
Confidence:	Medium
Parameter:	tj_seed
Attack:	
Evidence:	Set-Cookie: tj_seed
CWE ID:	1004
WASC ID:	13
Source:	Passive (10010 - Cookie No HttpOnly Flag)
Input Vector:	

Description:

A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Cookie Without Secure Flag

URL: https://cars.booking.com/
Risk: Low
Confidence: Medium
Parameter: tj_seed
Attack:
Evidence: Set-Cookie: tj_seed
CWE ID: 614
WASC ID: 13
Source: Passive (10011 - Cookie Without Secure Flag)
Input Vector:

Description:

A cookie has been set without the secure flag, which means that the cookie can be accessed via unencrypted connections.

Cookie with SameSite Attribute None

URL: https://cars.booking.com
Risk: Low
Confidence: Medium
Parameter: bkng
Attack:
Evidence: set-cookie: bkng
CWE ID: 1275
WASC ID: 13
Source: Passive (10054 - Cookie without SameSite Attribute)
Alert Reference: 10054-2
Input Vector:

Description:

A cookie has been set with its SameSite attribute set to "none", which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Cookie without SameSite Attribute

URL: https://cars.booking.com/
Risk: Low
Confidence: Medium
Parameter: tj_seed
Attack:
Evidence: Set-Cookie: tj_seed
CWE ID: 1275
WASC ID: 13
Source: Passive (10054 - Cookie without SameSite Attribute)
Alert Reference: 10054-1
Input Vector:

Description:

A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.

Cross-Domain JavaScript Source File Inclusion

URL: https://cars.booking.com
Risk: Low
Confidence: Medium
Parameter: https://cf.bstatic.com/libs/privacy-consent/releases/2.1.55/customer/cookie-banner.min.js
Attack:
Evidence: <script type="text/javascript" nonce="cZyt71Prpngo0eY" src="https://cf.bstatic.com/libs/privacy-consent/releases/2.1.55/customer/cookie-banner.min.js" data-domain-script="3ea94870-d4b1-483a-b1d2-faf1d982bb31"></script>
CWE ID: 829
WASC ID: 15
Source: Passive (10017 - Cross-Domain JavaScript Source File Inclusion)
Input Vector:

Description:

The page includes one or more script files from a third-party domain.

Timestamp Disclosure - Unix

URL: https://cars.booking.com
Risk: Low
Confidence: Low
Parameter:
Attack:
Evidence: 1714331758
CWE ID: 200
WASC ID: 13
Source: Passive (10096 - Timestamp Disclosure)
Input Vector:

Description:

A timestamp was disclosed by the application/web server - Unix

Results that were obtained when scanned through nikto

```
┌──(dinu_mrx㉿kali)-[~]
└─$ nikto -h cars.booking.com

- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 104.19.164.108, 104.19.165.108
+ Target IP:          104.19.164.108
+ Target Hostname:    cars.booking.com
+ Target Port:        80
+ Start Time:         2024-04-29 02:44:57 (GMT5.5)
---------------------------------------------------------------------------
+ Server: cloudflare
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://devel
oper.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent t
o render the content of the site in a different fashion to the MIME type. See: https
://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type
-header/
+ Root page / redirects to: https://cars.booking.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /cdn-cgi/trace: Retrieved access-control-allow-origin header: *.
+ /cdn-cgi/trace: Cloudflare trace CGI found, which may leak some system information.
+ 8046 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time:           2024-04-29 02:48:16 (GMT5.5) (199 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

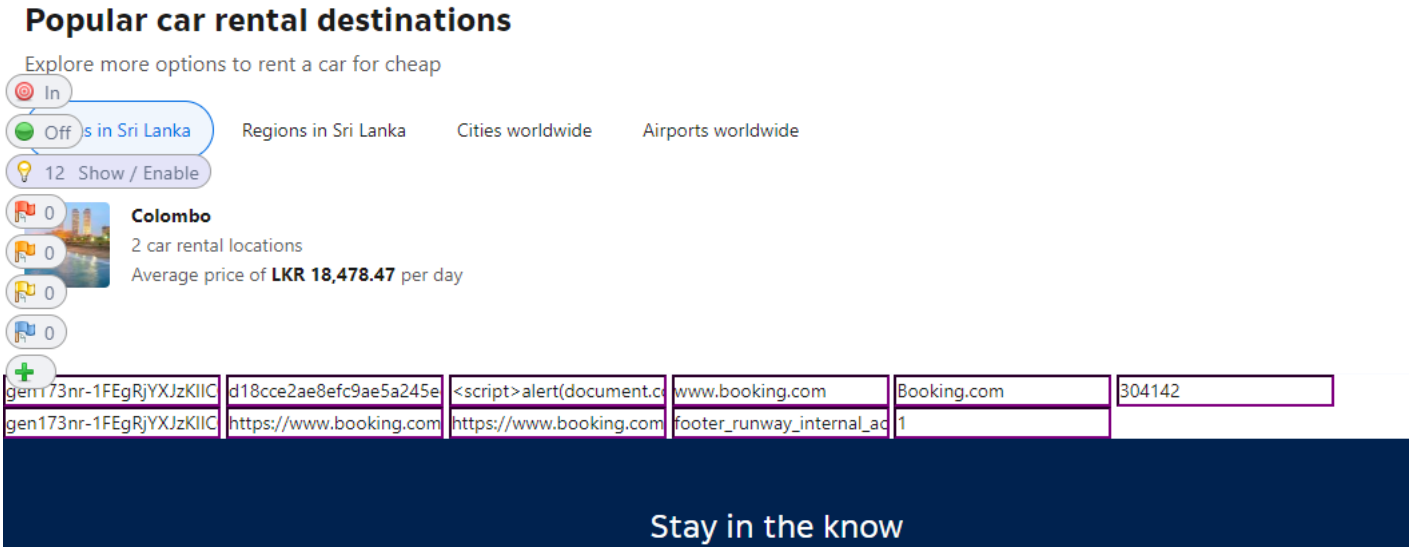
```
┌──(dinu_mrx㉿kali)-[~]
└─$ nikto -h cars.booking.com -dbcheck -evasion+ -config+ -RSAcert+ -Userbds

Syntax Check: /var/lib/nikto/databases/db_favicon
        361 entries
Syntax Check: /var/lib/nikto/databases/db_dictionary
        1825 entries
Syntax Check: /var/lib/nikto/databases/db_404_strings
        39 entries
Syntax Check: /var/lib/nikto/databases/db_outdated
        1256 entries
Syntax Check: /var/lib/nikto/databases/db_variables
        38 entries
Syntax Check: /var/lib/nikto/databases/db_tests

        6954 entries
Syntax Check: /var/lib/nikto/databases/db_realms
        170 entries
Syntax Check: /var/lib/nikto/databases/db_parked_strings
        8 entries
Syntax Check: /var/lib/nikto/databases/db_embedded
        16 entries
Syntax Check: /var/lib/nikto/databases/db_headers
        118 entries
Syntax Check: /var/lib/nikto/databases/db_server_msgs
        259 entries
Syntax Check: /var/lib/nikto/databases/db_domino
        274 entries
Syntax Check: /var/lib/nikto/databases/db_httpoptions
```

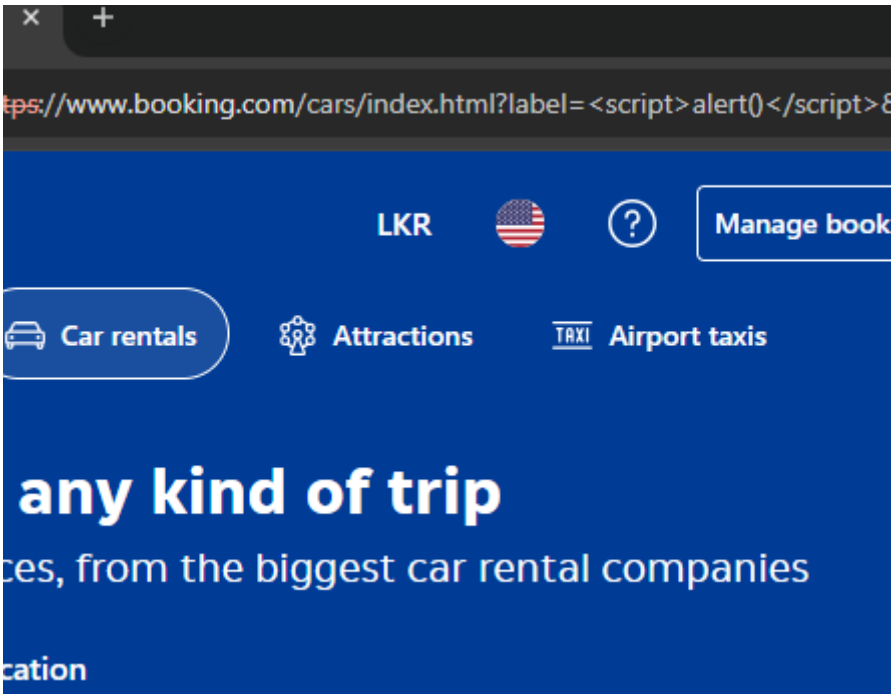
Didn’t give any configuration files, client certificate files.

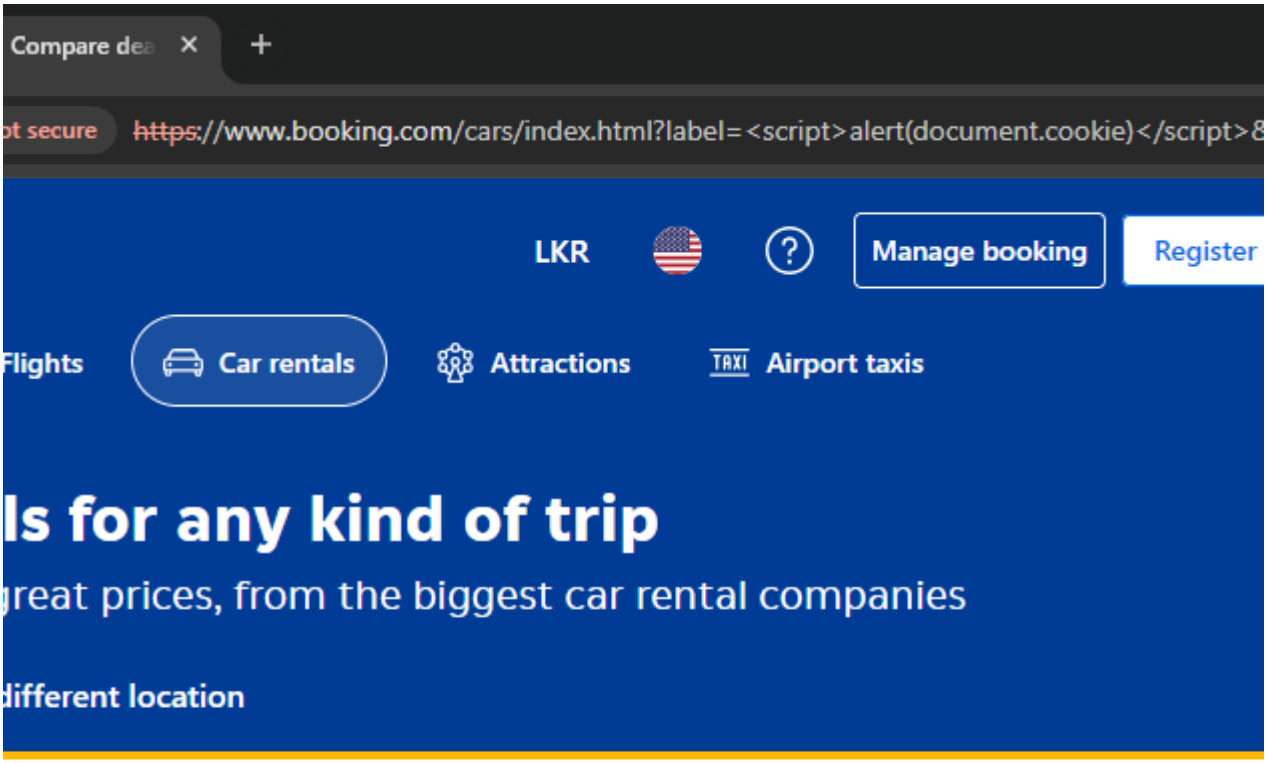
After manually testing the using OWSAP ZAP I found 12 hidden fields, however these hidden fields are sanitized, and they also validate the input



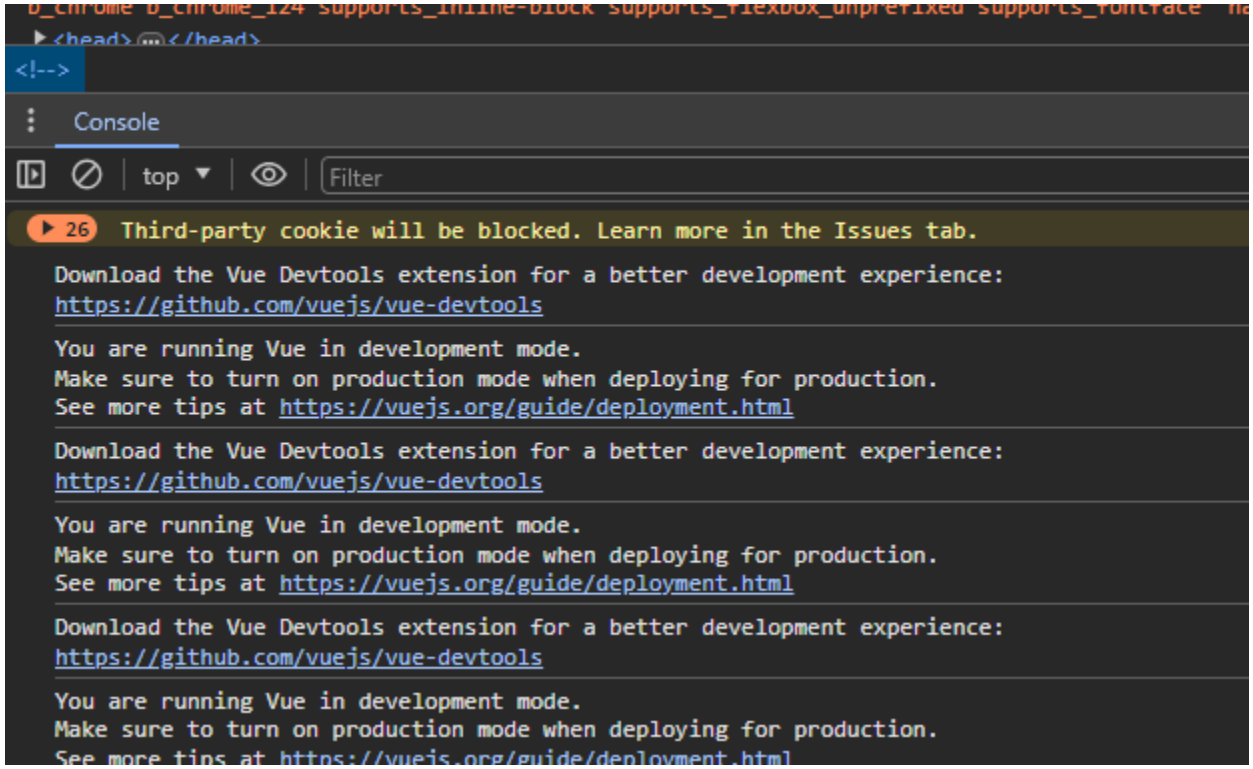
In here it's inputs are sanitized

Results that I obtained when tested with the URL.




Compare dea
ot secure
https://www.booking.com/cars/index.html?label=<script>alert(document.cookie)</script>8
LKR
Manage booking
Register
Flights
Car rentals
Attractions
Airport taxis
ls for any kind of trip
great prices, from the biggest car rental companies
different location


on
Time
10:00 AM



It is blocking the request which means even the URL is properly encoded, with special characters and Js commands.

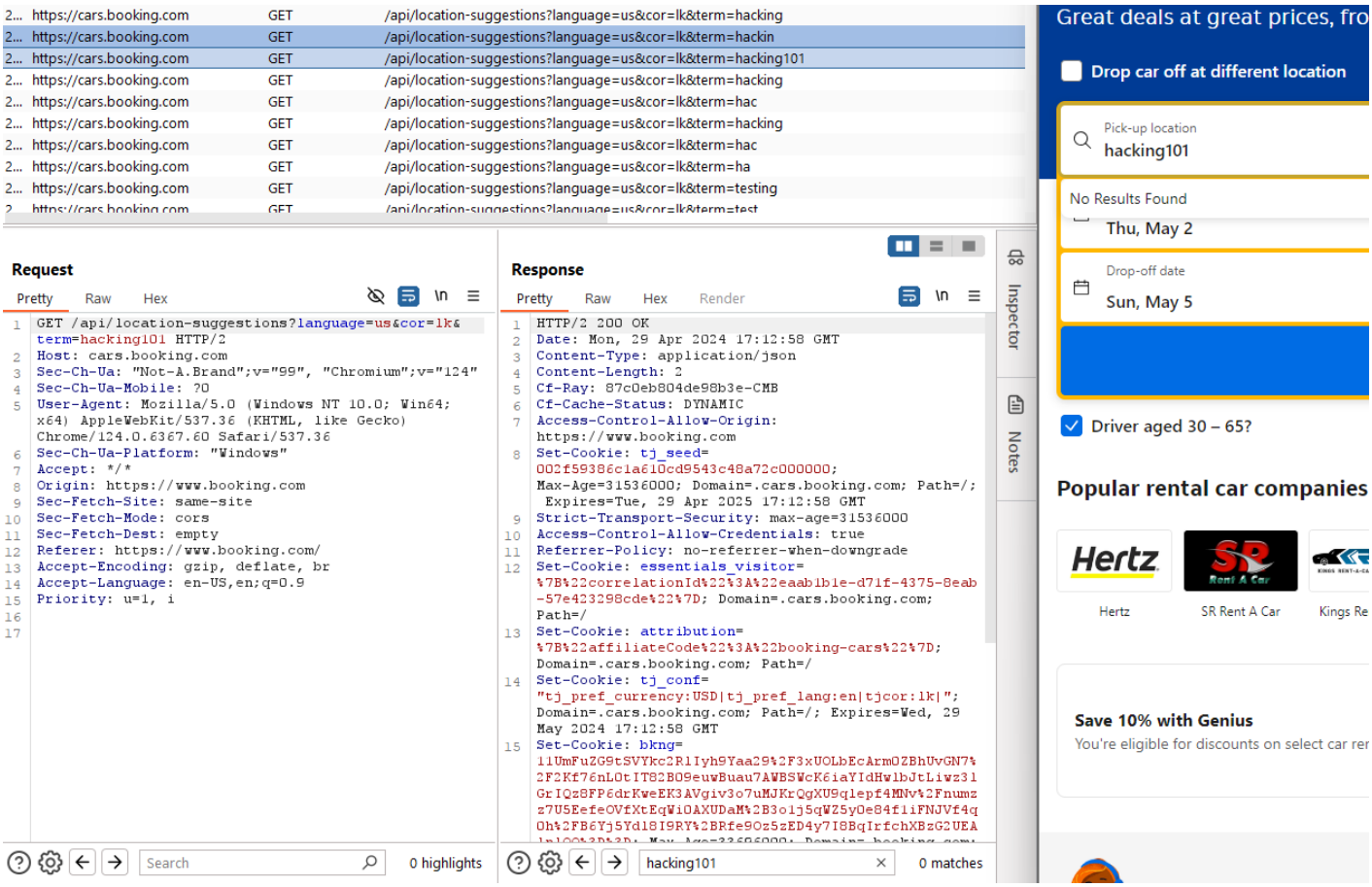
And when we see the cookie value it is also encode

Name	Value	Dom...	Path	Expir...	Size	Http...	Secu...	Sam...	Parti...	Prior...
11_srd	%7B%22features%22%3A%5B%7B%...	.boo...	/	Sessi...	97					Medi...
BJS	-	.boo...	/	2024...	4	✓	✓			Medi...
FPID	FPID2.2.J2wlkp4v8G67nDIXK3kqaw0F...	.boo...	/	2025...	69	✓	✓			Medi...
FPLC	g3SvKpyiQeoJhgayGS5JG8%2BC%2F...	.boo...	/	2024...	142		✓			Medi...
IDE	AHWqTUIYHdLf942tty3Wi5GJ32j9PY...	.dou...	/	2025...	67	✓	✓	None		Medi...
MR	0	.bat....	/	2024...	3		✓	None		Medi...
MSPTC	ly6I57tV8M5Ev9cmGjoN0T3puCS07Z...	.bing...	/	2025...	48		✓	None	http...	Medi...
MUID	0EF6A7A1EEBF6E101495B3D1EFE56F...	.bing...	/	2025...	36		✓	None		High
OptanonConsent	implicitConsentCountry=nonGDPR&i...	.ww...	/	2024...	397		✓	None		Medi...
_GRECAPTCHA	09AN_JpP-XbVd1EBXQTGfH-m_cCv6...	www...	/rec...	2024...	100	✓	✓	None		High
_ga	GA1.1.1233558673.1714392451	.boo...	/	2025...	30					Medi...
_ga_A12345	GS1.1.1714395106.2.1.1714395265.0....	.boo...	/	2025...	56					Medi...
_ga_SEJWFCBCVM	GS1.1.1714395106.1.0.1714395261.6...	.boo...	/	2025...	52					Medi...
_gcl_au	1.1.254060765.1714392455	.boo...	/	2024...	31					Medi...
_gid	GA1.2.1749576607.1714392451	.boo...	/	2024...	31					Medi...
_uetsid	10ef5a80062111ef87234f123a8d6b1a	.boo...	/	2024...	39					Medi...
_uetvid	10ef7930062111ef98dd4f1988d15f5f	.boo...	/	2025...	39					Medi...
_yjsu_yjad	1714392456.580a670b-6e5d-4a83-b...	.boo...	/	2025...	57					Medi...
aws-waf-token	a21affdc-6c26-43ae-9fcb-dae92078d...	.boo...	/	2024...	295		✓	Lax		Medi...
bkng	11UmFuZG9tSVYkc2RIlyh9Yaa29%2F...	.boo...	/	2025...	226	✓	✓	None		Medi...
bkng_sso_auth	CAIQsOnuTRpmKFU/PgPZOkkKXuL4...	.boo...	/	2025...	161	✓	✓	Lax		Medi...
bkng_sso_ses	e30	.boo...	/	2025...	15	✓	✓			Medi...
bkng_sso_session	e30	.boo...	/	2025...	10	✓	✓			Medi...

Cookie Value ☑ Show URL-decoded

g3SvKpyiQeoJhgayGS5JG8+C/9fUHNLTln9NdIYG0n1G68x1IOI1Gor1iZ6IMMxqbiSZ19iJITp526f1FzcphJvbm7obvRoxFIgH32yxrW7367n2/lemduG1VZDK5g==

When it come to cross site scripting when I enter the location it not reflected in the DOM so it is impossible to inject XSS



No highlights in the response so this site is invulnerable to XSS.