**BUB BOUNTY**

A blue and orange logo

Description automatically generated

**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**

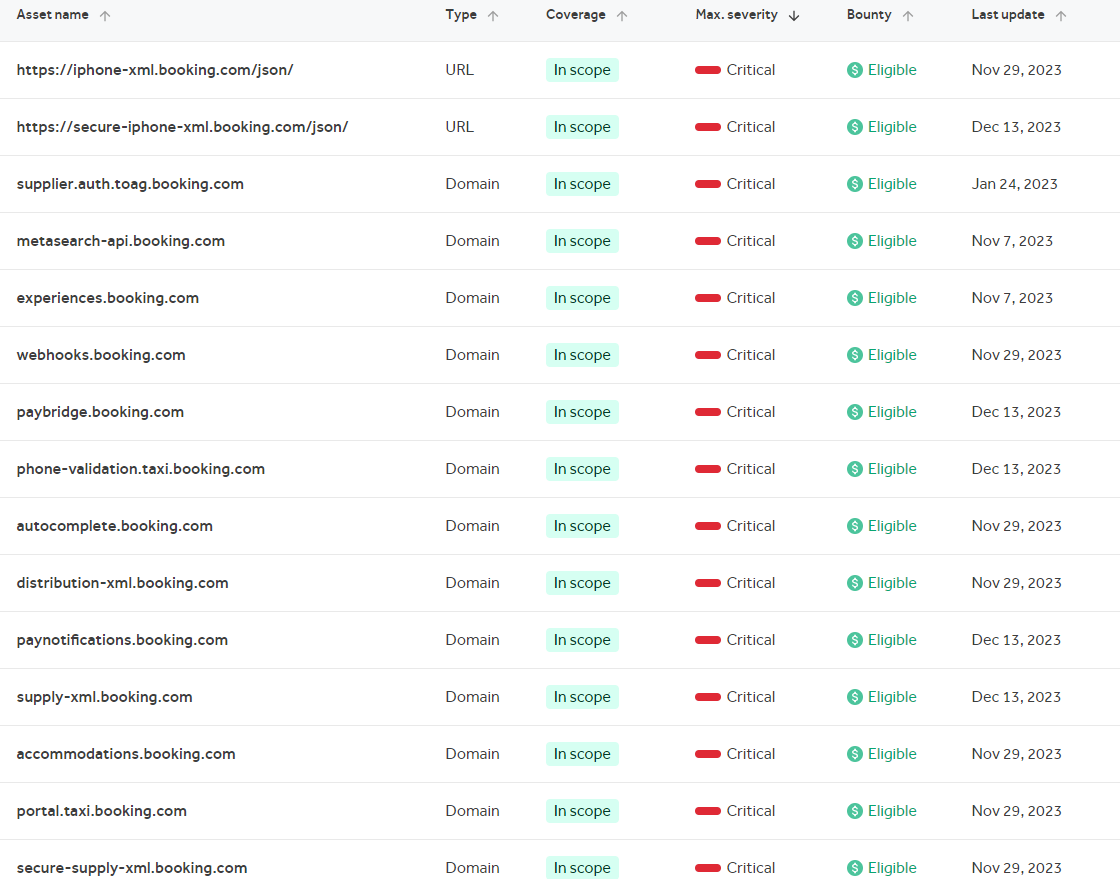
**Accepted, in-scope vulnerabilities include, but are not limited to:**

* 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.

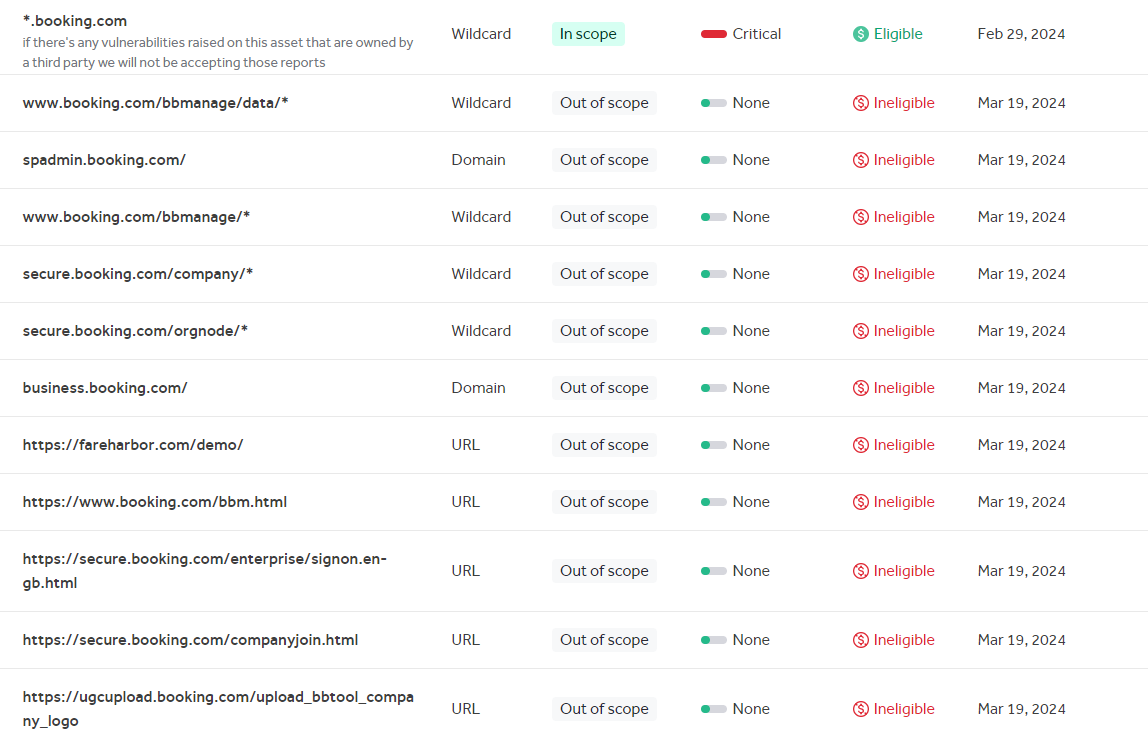
**The following issues are outside the scope of our vulnerability rewards program:**

* 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.)



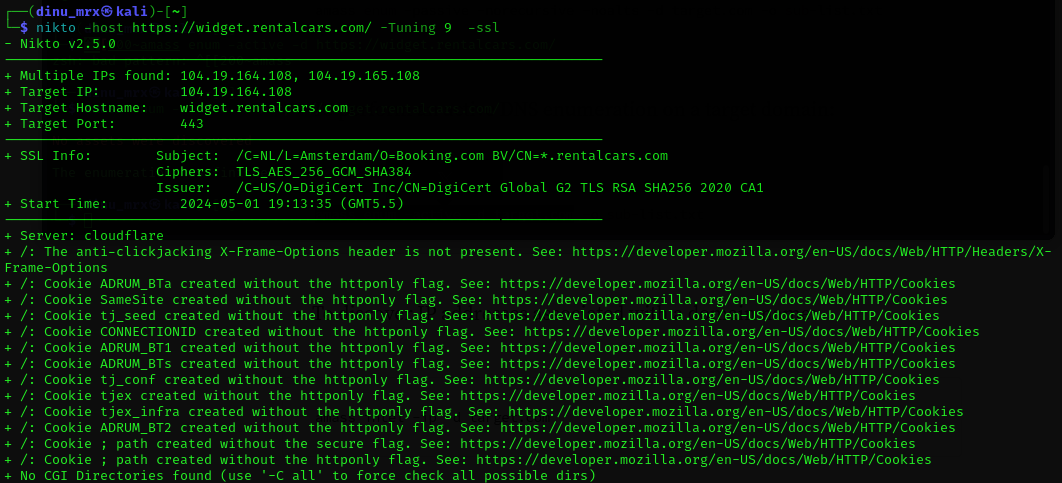
A screenshot of a computer

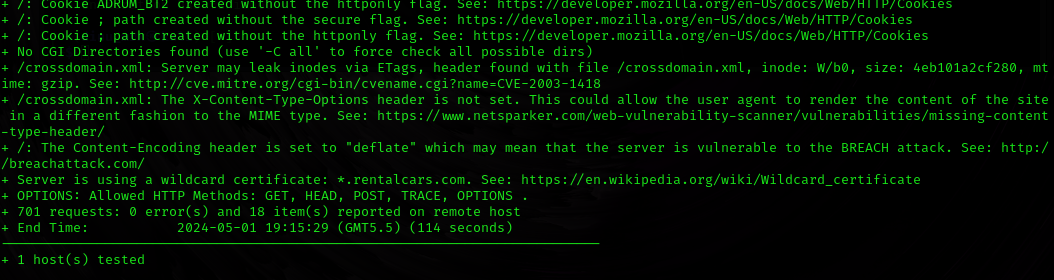
Description automatically generated



<https://widget.rentalcars.com/>

result obtain from nikto





According to the nikto The content-Encoding header is set to “deflate” where it may mean the server is vulnerable to breach attacks.

BREACH is a category of vulnerabilities and not a specific instance affecting a specific piece of software. To be vulnerable, a web application must:

* Be served from a server that uses HTTP-level compression
* Reflect user-input in HTTP response bodies
* Reflect a secret (such as a CSRF token) in HTTP response bodies.

Additionally, while not strictly a requirement, the attack is helped greatly by responses that remain mostly the same (modulo the attacker's guess). This is because the difference in size of the responses measured by the attacker can be quite small. Any noise in the side-channel makes the attack more difficult (though not impossible).

It is important to note that the attack is agnostic to the version of TLS/SSL, and does not require TLS-layer compression. Additionally, the attack works against any cipher suite. Against a stream cipher, the attack is simpler; the difference in sizes across response bodies is much more granular in this case. If a block cipher is used, additional work must be done to align the output to the cipher text blocks.

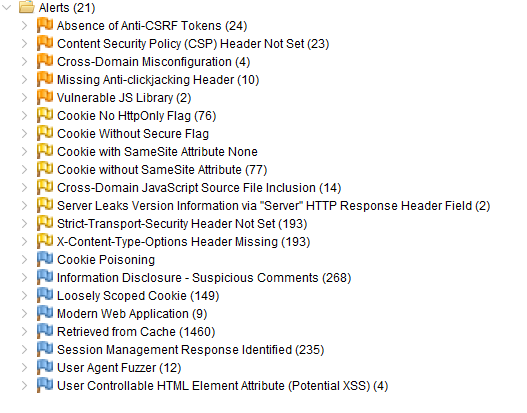
<https://www.breachattack.com/>

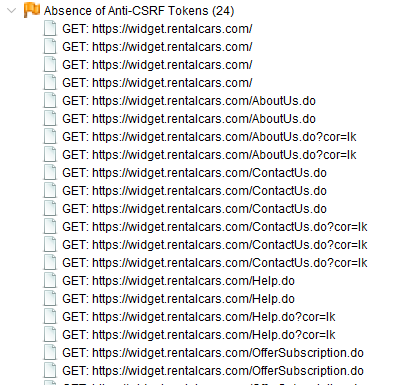
And also it shows that the server may leak inodes via ETags, header found with file /crossdomain.xml, inode: W/b0, size: 4eb101a2cf280, mtime: gzip.

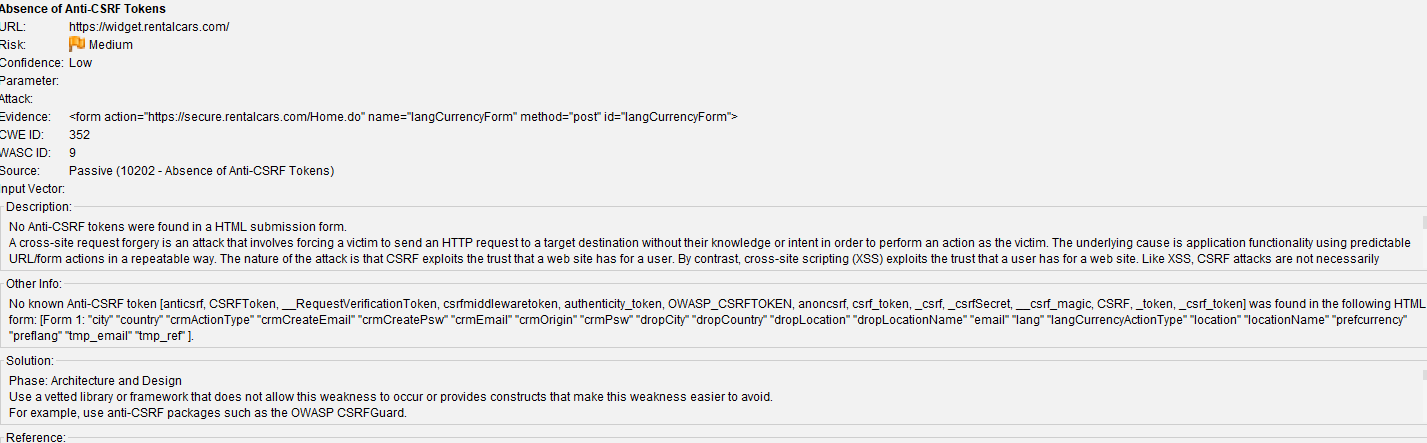
Apache HTTP Server 1.3.22 through 1.3.27 on OpenBSD allows remote attackers to obtain sensitive information via (1) the ETag header, which reveals the inode number, or (2) multipart MIME boundary, which reveals child process IDs (PID).

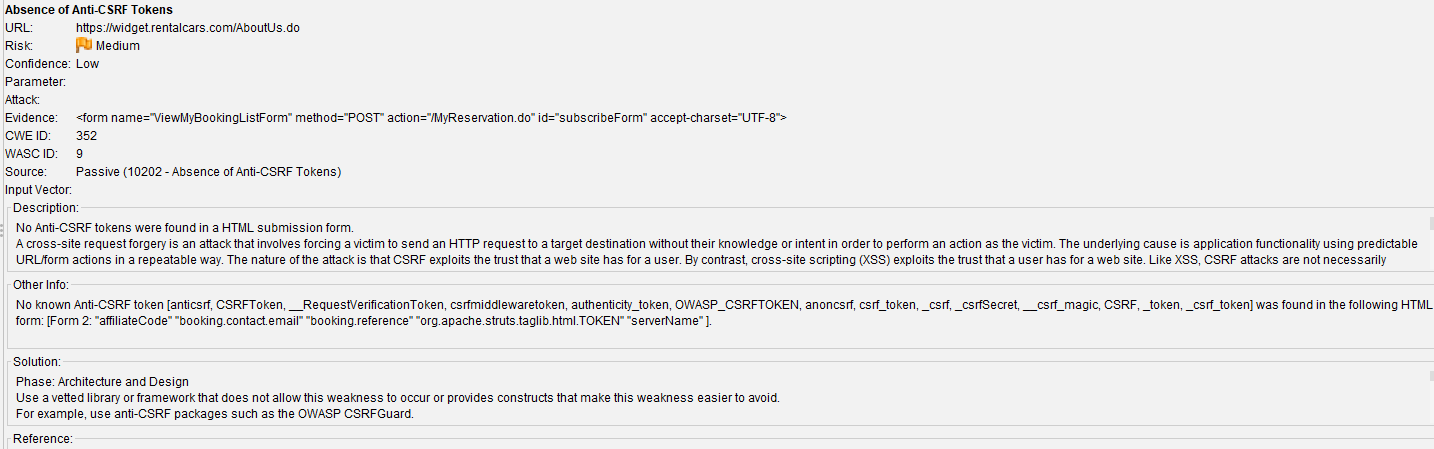
<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418>

The automated scan through the OWSAP ZAP gives the following results

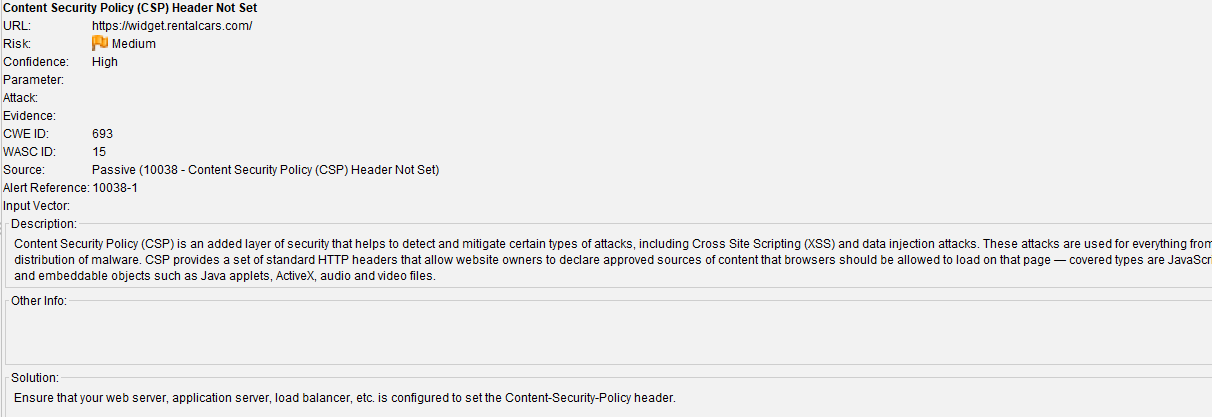


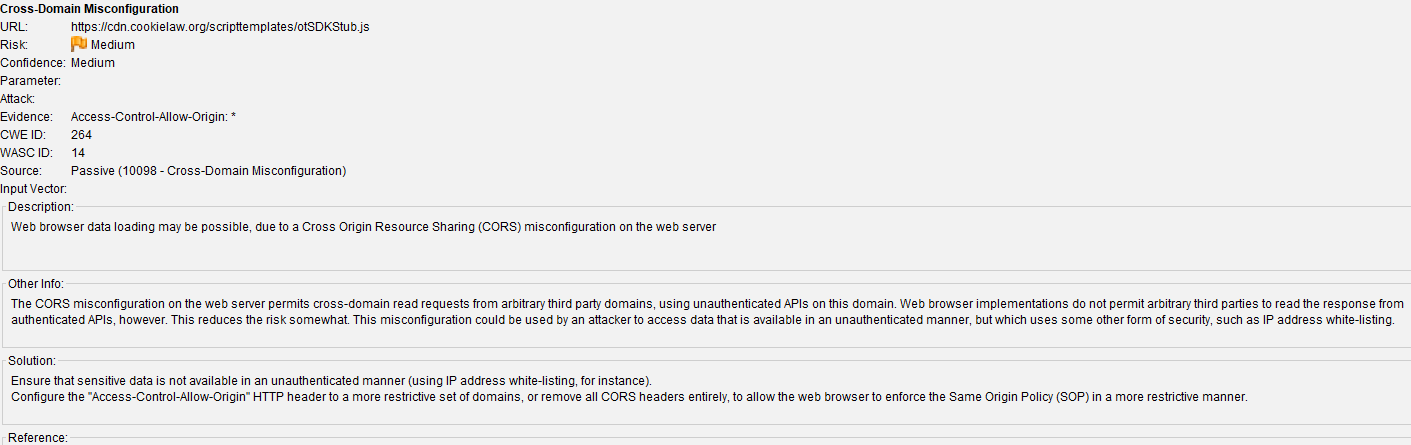


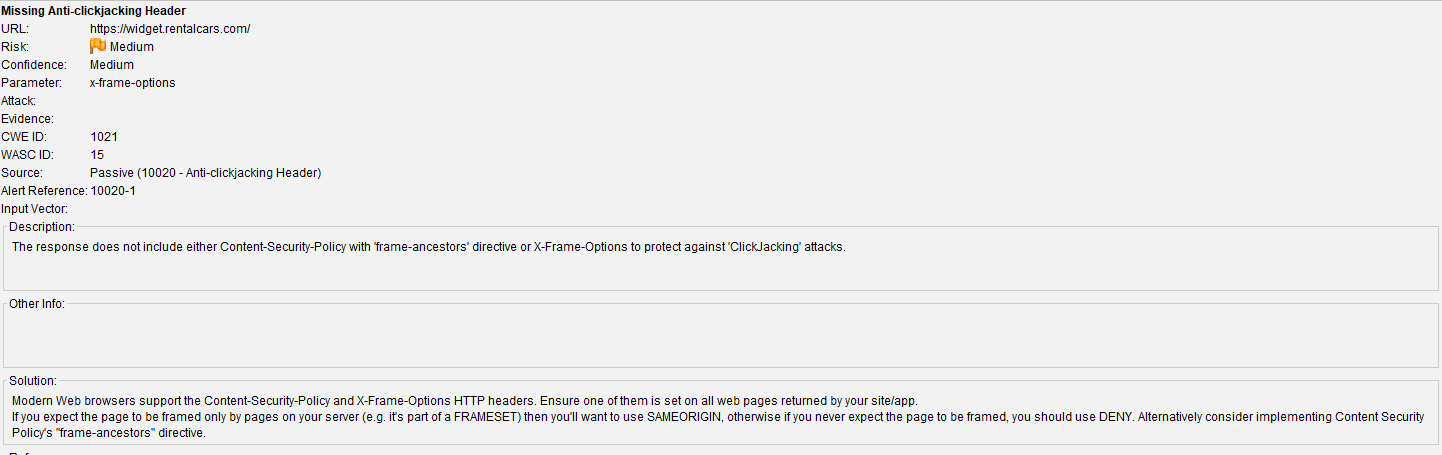


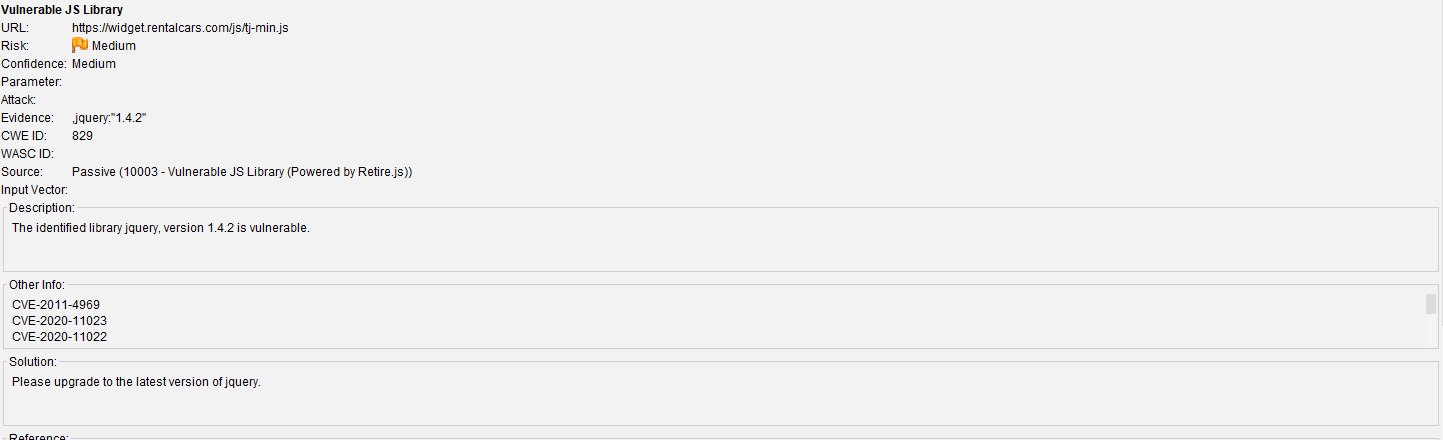










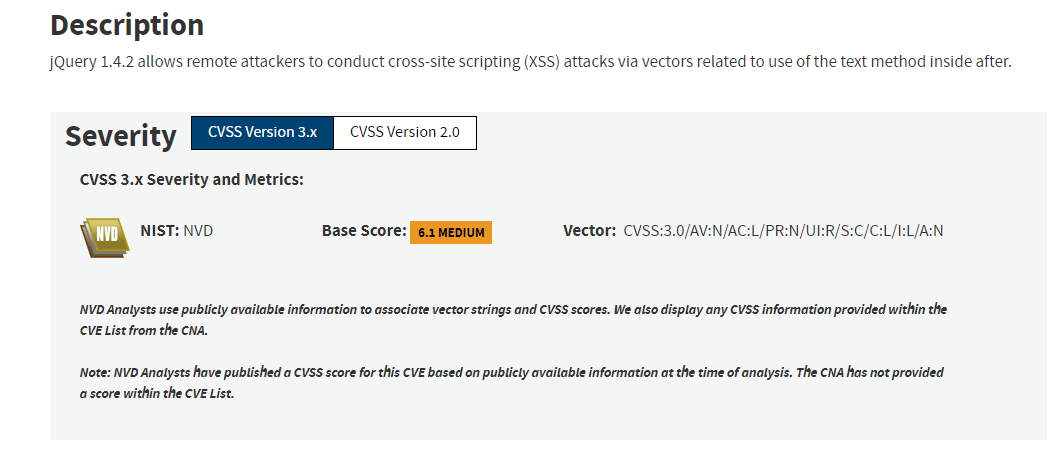




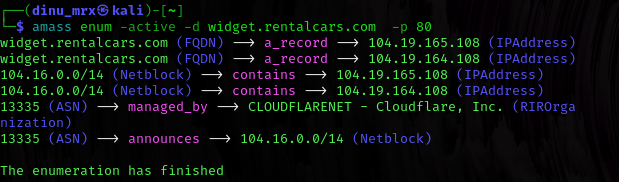
The js library they are using is jquery 1.4.2

jQuery 1.4.2 allows remote attackers to conduct cross-site scripting (XSS) attacks via vectors related to use of the text method inside after.

**CVE-2014-6071 Detail**



The results given from amass enum scan

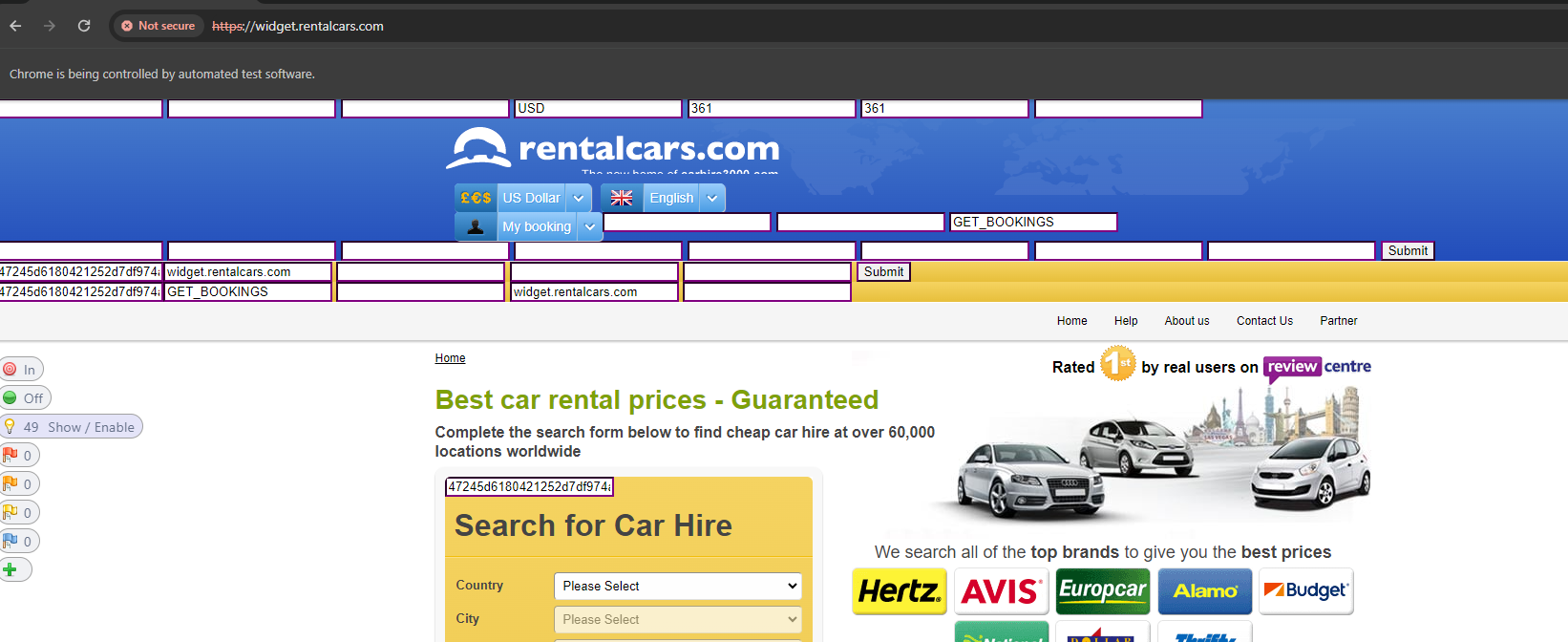


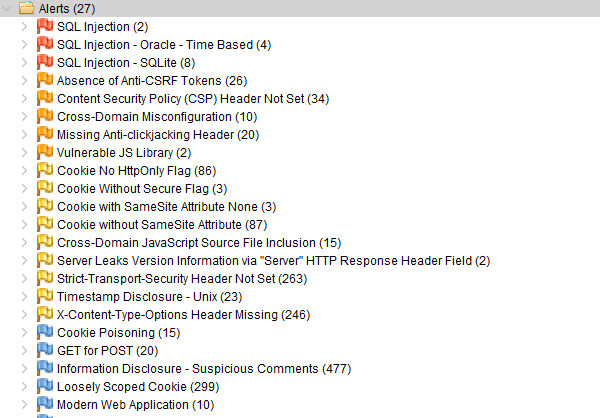
From the above amass **enum** command output for **widget.rentalcars.com**, here's what we can derive:

1. **Domain Name and IP Addresses**:
   * The domain **widget.rentalcars.com** resolves to two IP addresses:
     + **104.19.165.108**
     + **104.19.164.108**
2. **Netblock Information**:
   * Both IP addresses (**104.19.165.108** and **104.19.164.108**) belong to the netblock **104.16.0.0/14**, which is managed by Cloudflare (**13335 ASN**).
3. **Autonomous System Number (ASN)**:
   * The Autonomous System Number (**13335**) is managed by Cloudflare (**CLOUDFLARENET - Cloudflare, Inc.**).
   * Cloudflare announces the IP range **104.16.0.0/14**.
4. **Interpretation**:
   * The domain **widget.rentalcars.com** is hosted on Cloudflare's infrastructure (**104.16.0.0/14** netblock).
   * Cloudflare manages the DNS resolution and serves as a proxy for incoming traffic to the domain.
5. **Additional Context**:
   * The presence of multiple IP addresses (**104.19.165.108** and **104.19.164.108**) might indicate load balancing or redundancy for the domain.
   * Cloudflare's management of the domain suggests that security features like DDoS protection, SSL termination, and content caching are likely employed.

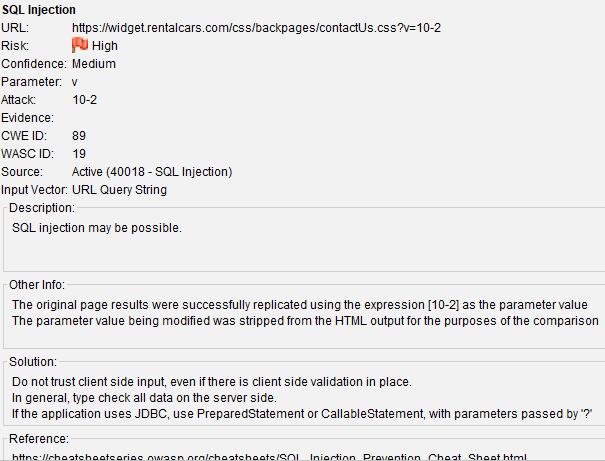
Results which were obtain by manual exploration of the site.

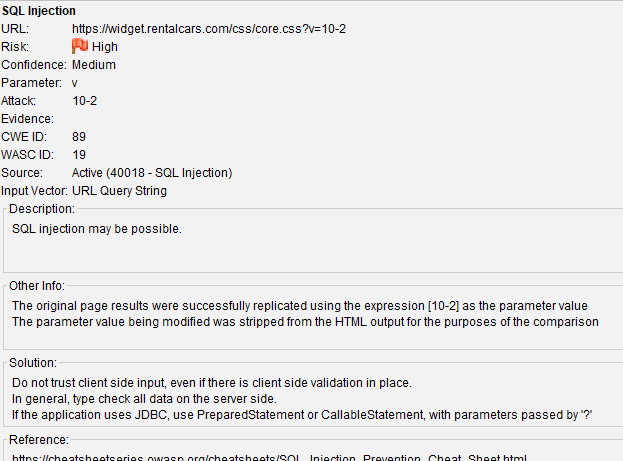
This has 49 hidden fields they have been shown in the screenshot



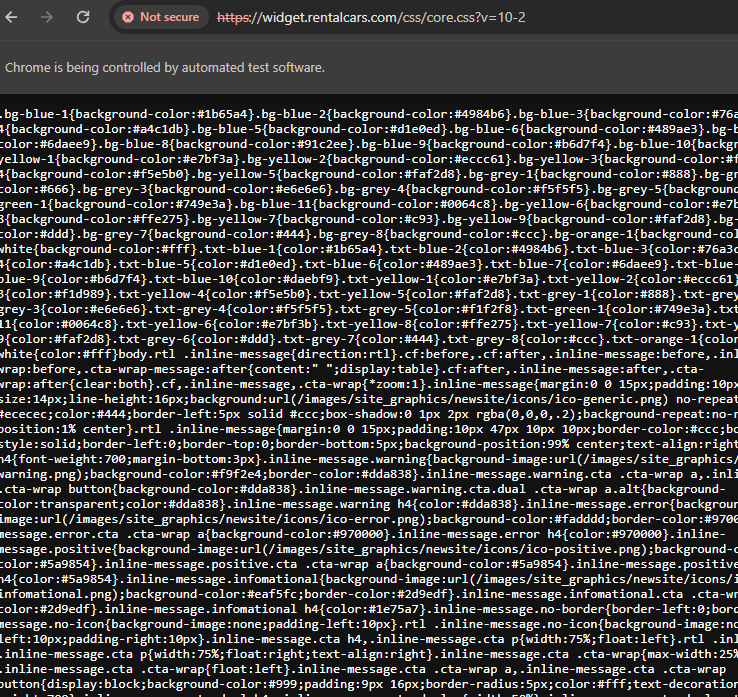






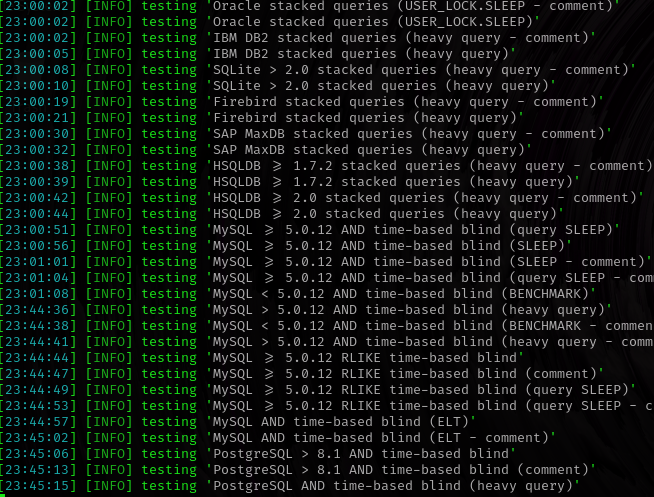


<https://widget.rentalcars.com/css/core.css?v=10-2>



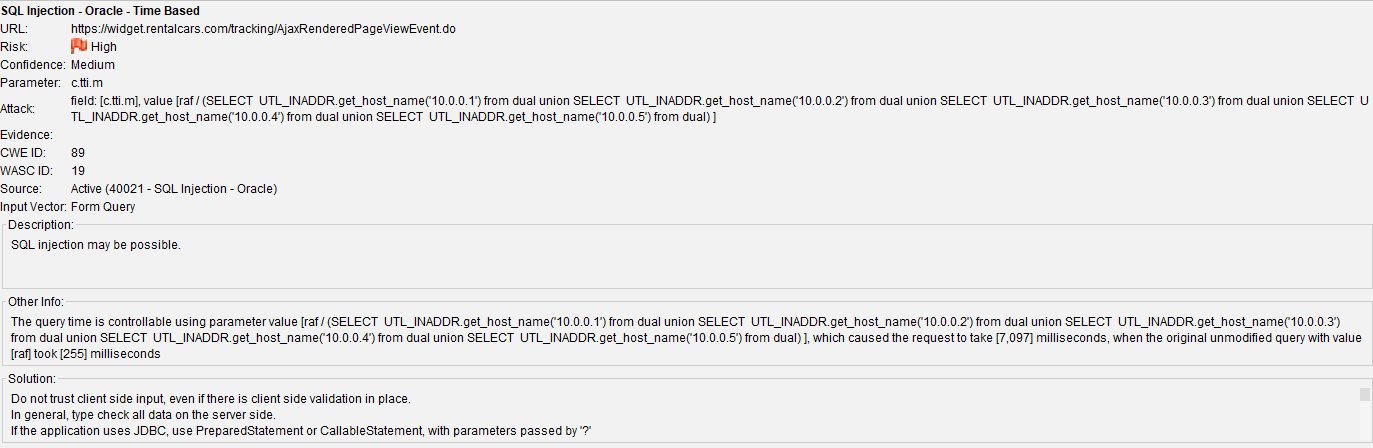
A screenshot of a computer program

Description automatically generated

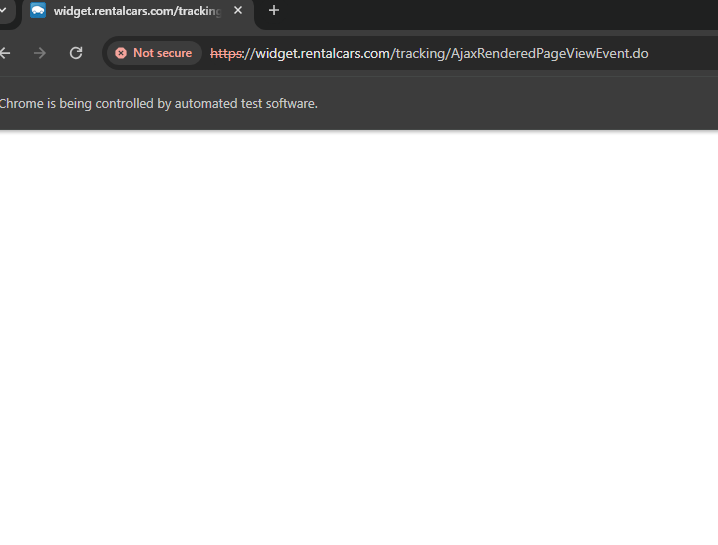




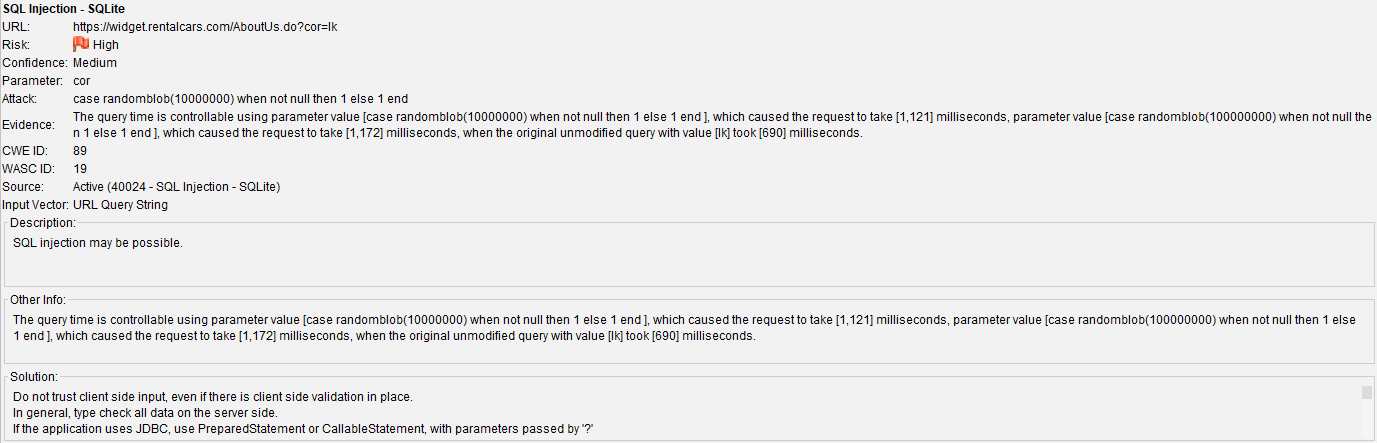
There no any positive result on the vulnerability when scanned with sqlmap.



The above page is in inaccessible the below screenshot gives appearance of the above site <https://widget.rentalcars.com/tracking/AjaxRenderedPageViewEvent.do>



Above site was just a blank site.

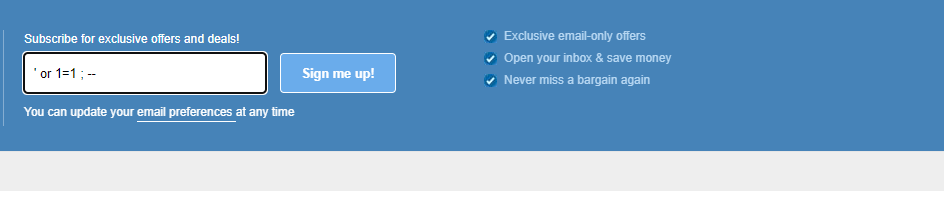


<https://widget.rentalcars.com/AboutUs.do?cor=lk>

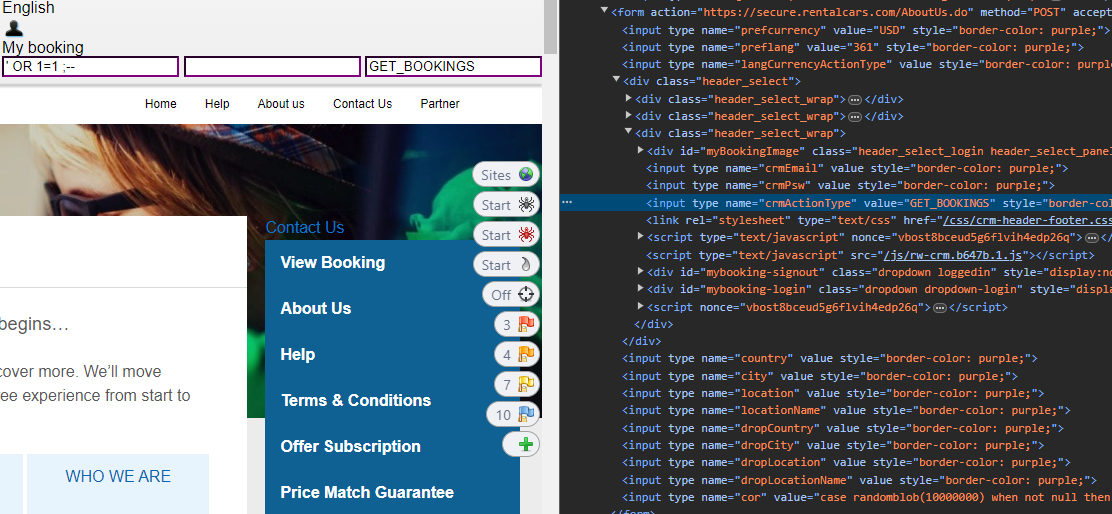
The above site contain 22 hidden fields is shown in the below screenshot

A screenshot of a computer

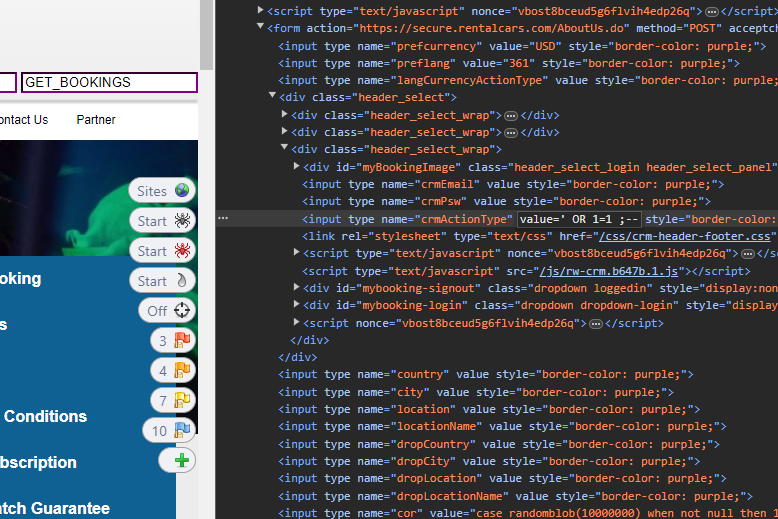
Description automatically generated



We can’t inject here, there the data is properly sanitized.



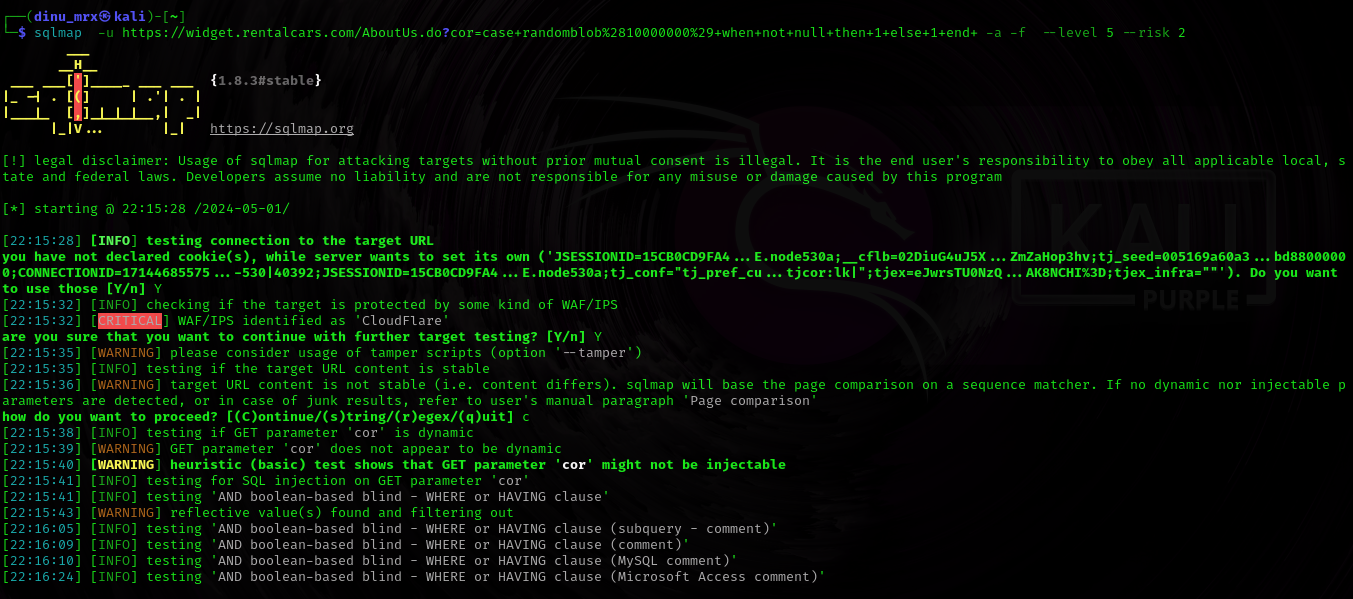
Following inject also didn’t work and I tried to change GET\_BOOKINGS in the hidden field to the payload but we can’t change it once the enter button is pressed it automatically changes.

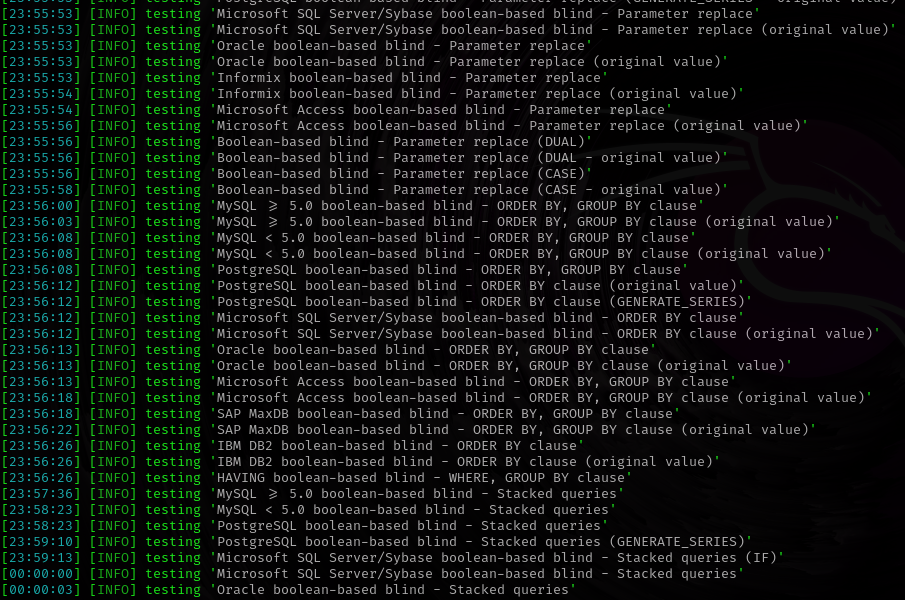


Checking with sqlmap for this particular site

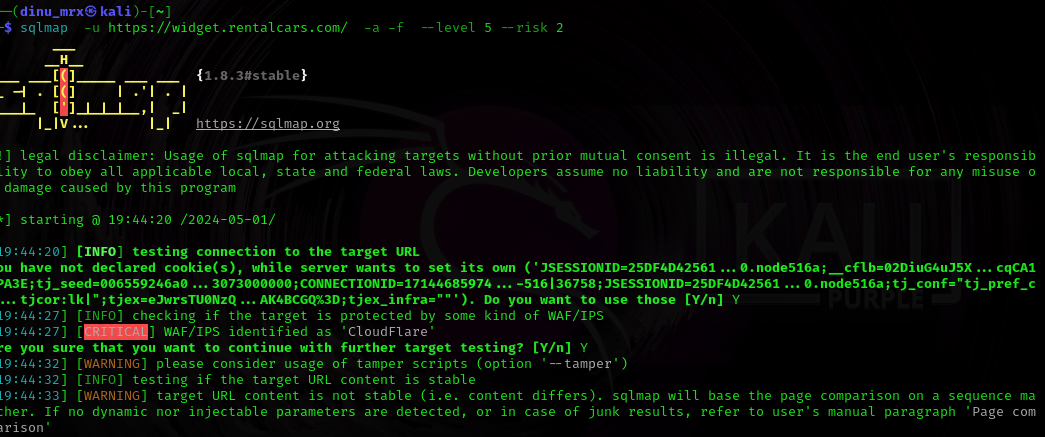
<https://widget.rentalcars.com/AboutUs.do?cor=case+randomblob%2810000000%29+when+not+null+then+1+else+1+end>+

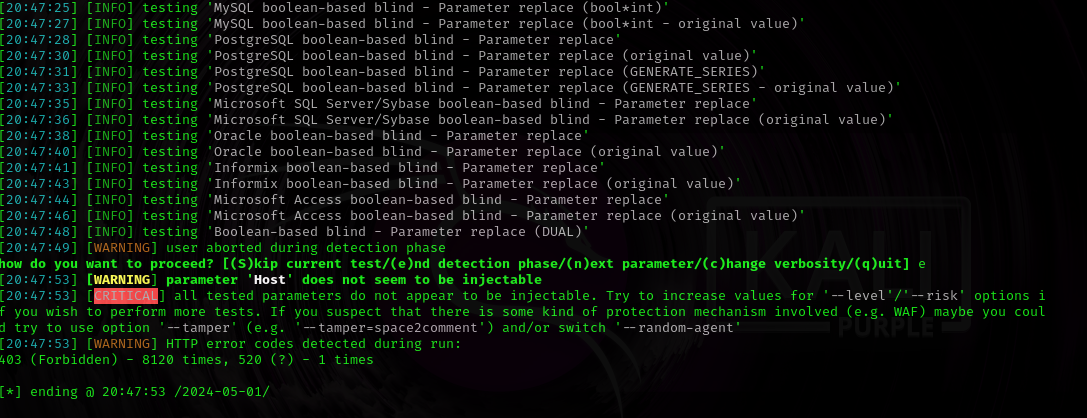
results are as follows.





Results given by SQLmap for <https://widget.rentalcars.com/>





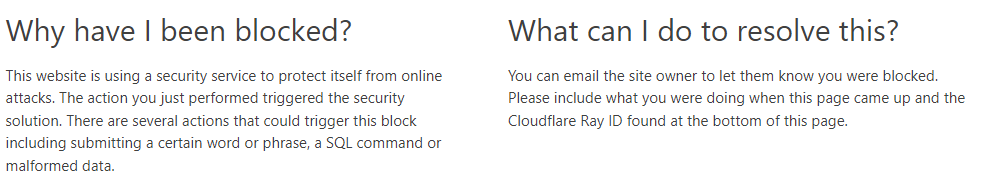
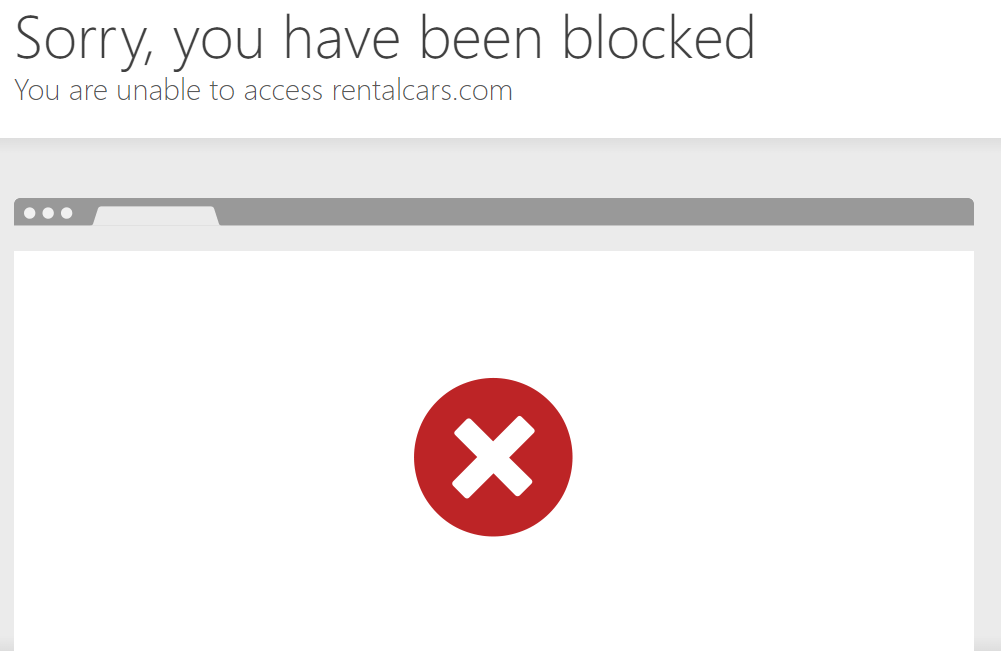
This domain is invulnerable to SQL injection. I tested this manually also here are the results.

A screenshot of a computer

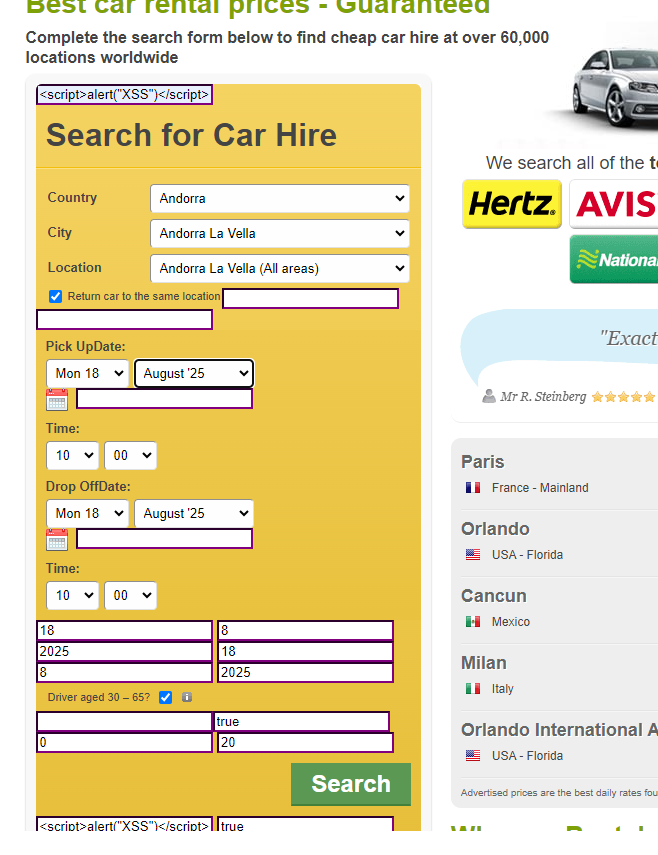
Description automatically generated

A screenshot of a computer

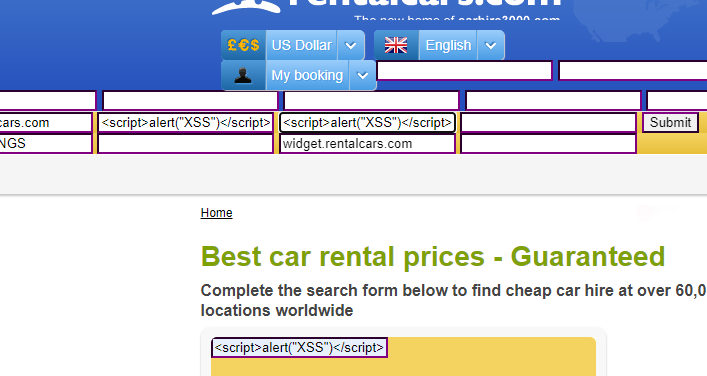
Description automatically generated



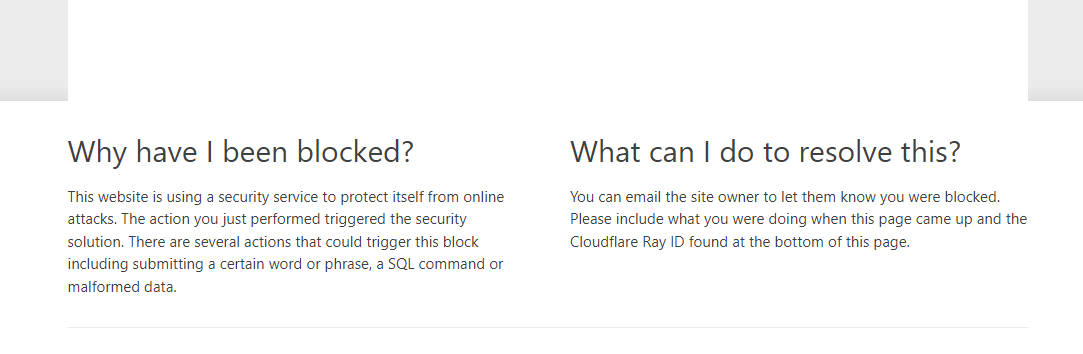
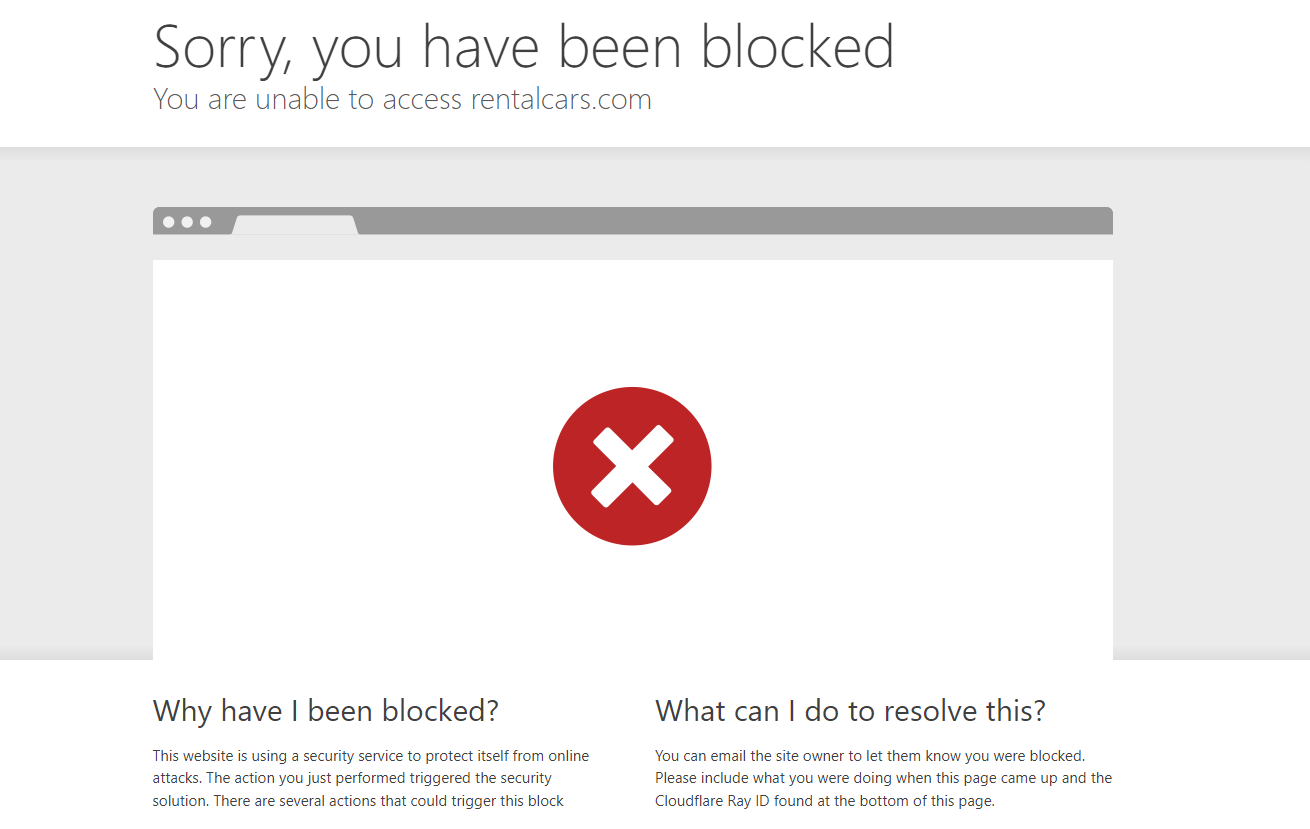
This domain is invulnerable to XSS injection. I test this manually also here are the results.







I enter basic XSS attack to almost all the hidden fields they are also properly sanitized



The site is invulnerable to directory traversal.



