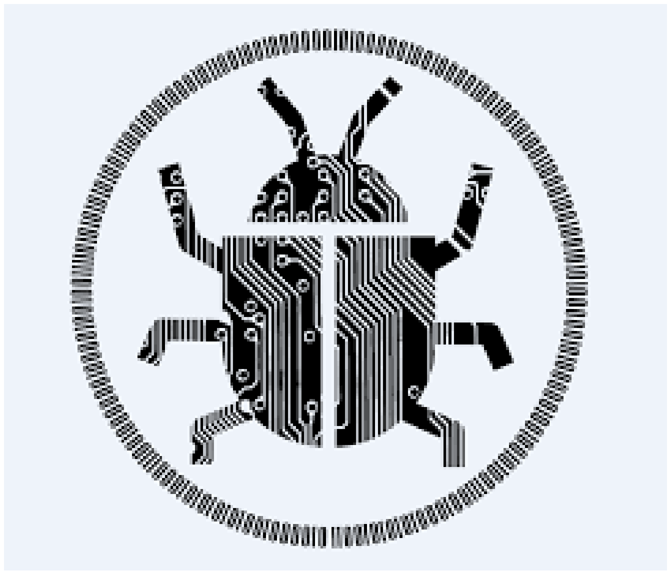
Bug Bounty journal



W T L D TISSERA

Web security

Table of content

1. Introduction

This report provides an overview of my bug bounty journey, detailing the selection process of the program, methodologies employed to identify vulnerabilities, and the tools utilized throughout the endeavor. It encompasses insights into OWASP Top 10 vulnerabilities and their significance in the context of my findings. Additionally, it features daily logs, documenting the challenges encountered and the actions taken to overcome them. Moreover, the report includes proof of concept demonstrations, insights into TryHackMe room creation, and the process of submitting reports to the platform.

2.Bug Bounty program

I used hacker1 platform to choose domain as my bug bounty program. At first, I selected different domains, they were sheer.com and trip.com but those bug bounty programs scope was limited then I found a bug bounty program www.boozt.com , comparing to previous two bug bounty programs it has a wider scope. The reason to select it because the wider the scope there is more possibility to find more vulnerabilities.

In the boozt fashion AB bug bounty program there were 6 main assets in scope they are,

- Com.booztlet
- Com.boozt
- Com.boozt.booztlet
- *.booztlet.com
- *.boozt.com
- Com.boozt.app

From these assets I mainly focused on *.boozt.com and *.booztlet.com

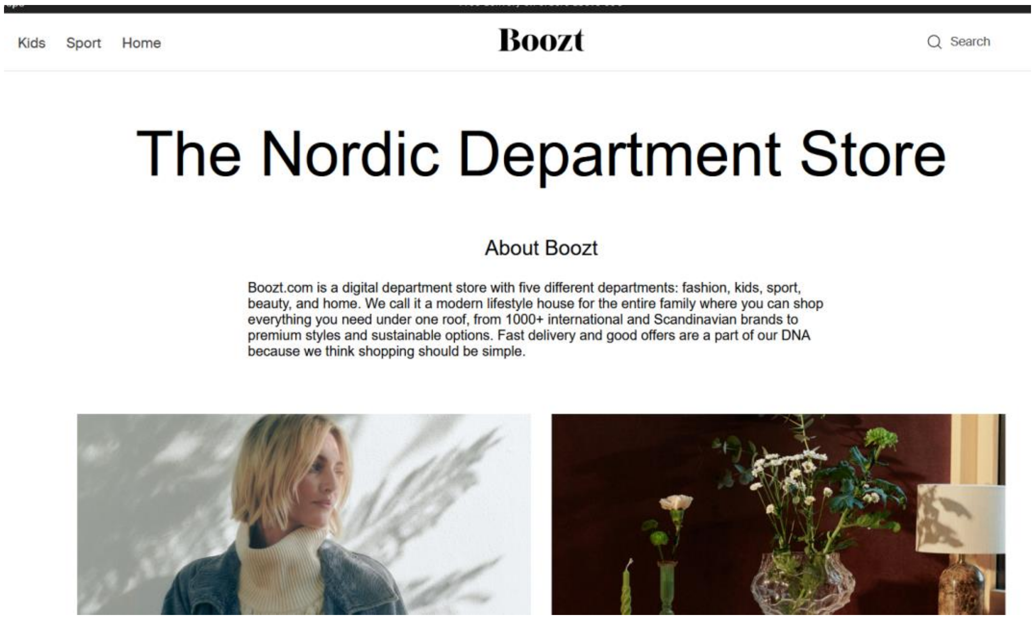
This bug bounty program was published in hacker1 by the company Boozt Fashion.

About the company

Boozt fashion is a Nordic technology company selling fashion and lifestyle online through our multi-brand webstore Boozt.com and outlet Booztlet.com.

Boozt.com is a multi-brand website for online shopping and booztlet.com is the Nordic fashion outlet.

About boozt.com

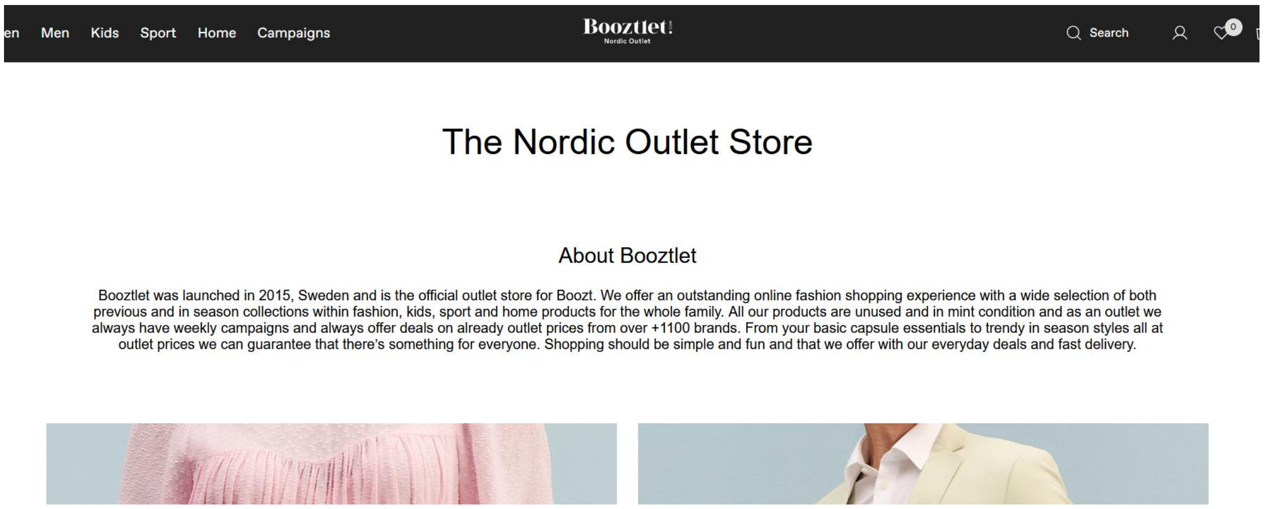


The Nordic Department Store

About Boozt

Boozt.com is a digital department store with five different departments: fashion, kids, sport, beauty, and home. We call it a modern lifestyle house for the entire family where you can shop everything you need under one roof, from 1000+ international and Scandinavian brands to premium styles and sustainable options. Fast delivery and good offers are a part of our DNA because we think shopping should be simple.

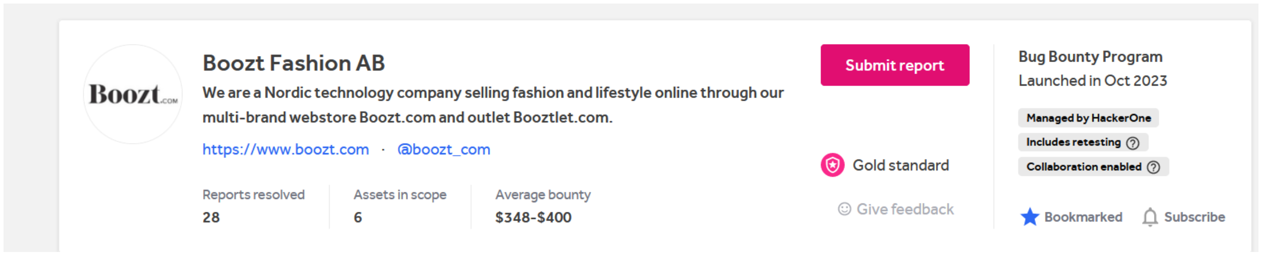
About booztlet.com



The Nordic Outlet Store

About Booztlet

Booztlet was launched in 2015, Sweden and is the official outlet store for Boozt. We offer an outstanding online fashion shopping experience with a wide selection of both previous and in season collections within fashion, kids, sport and home products for the whole family. All our products are unused and in mint condition and as an outlet we always have weekly campaigns and always offer deals on already outlet prices from over +1100 brands. From your basic capsule essentials to trendy in season styles all at outlet prices we can guarantee that there's something for everyone. Shopping should be simple and fun and that we offer with our everyday deals and fast delivery.



Boozt Fashion AB

We are a Nordic technology company selling fashion and lifestyle online through our multi-brand webstore Boozt.com and outlet Booztlet.com.

https://www.boozt.com · @boozt_com

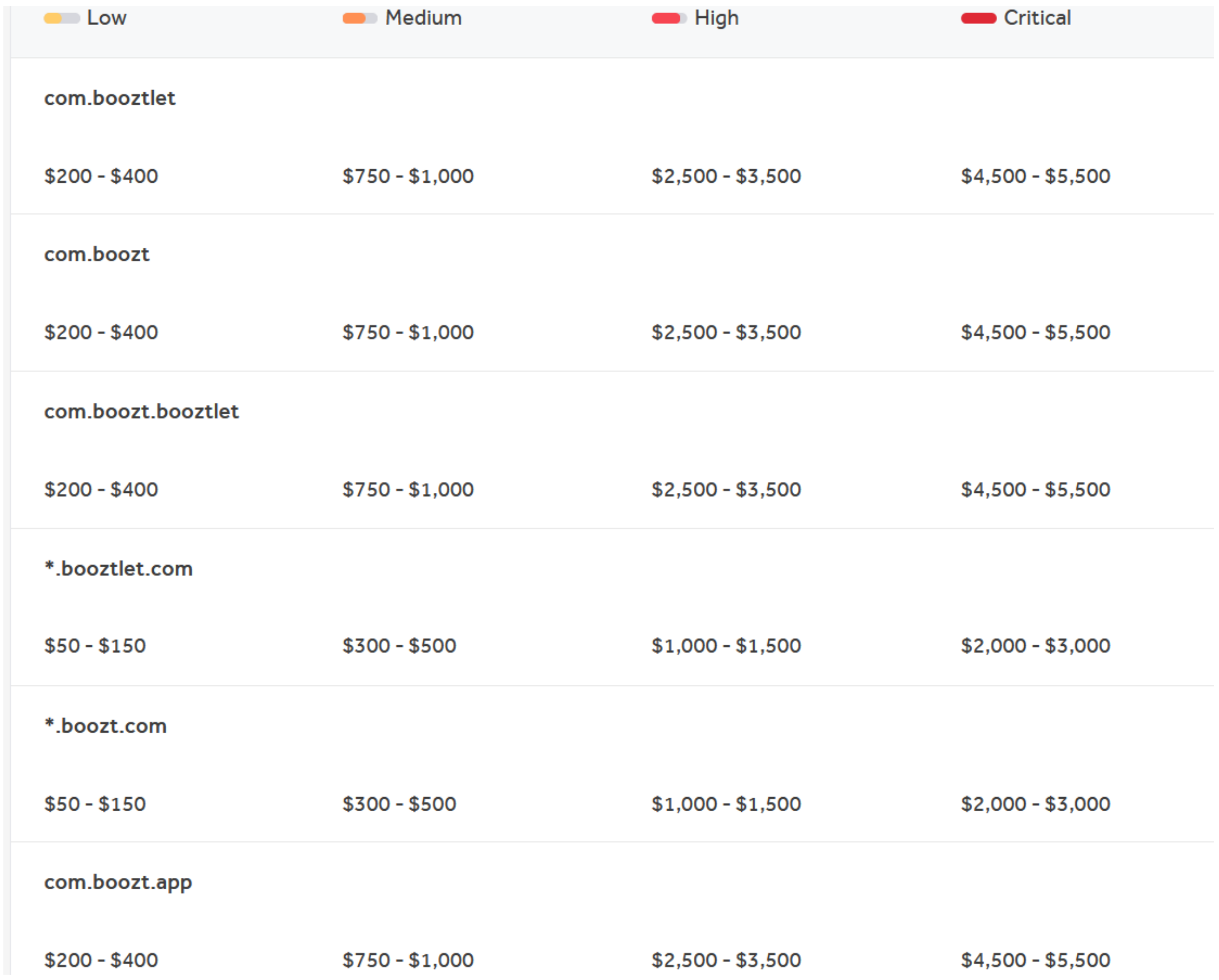
Reports resolved	Assets in scope	Average bounty
28	6	$348-$400

Bug Bounty Program
Launched in Oct 2023

The main domain - https://www.boozt.com/

Following are the list of assets which are in scope of the program and the bounty scheme depending of the severity level of the vulnerability

Reward schema



	Low	Medium	High	Critical
com.booztlet	$200 - $400	$750 - $1,000	$2,500 - $3,500	$4,500 - $5,500
com.boozt	$200 - $400	$750 - $1,000	$2,500 - $3,500	$4,500 - $5,500
com.boozt.booztlet	$200 - $400	$750 - $1,000	$2,500 - $3,500	$4,500 - $5,500
*.booztlet.com	$50 - $150	$300 - $500	$1,000 - $1,500	$2,000 - $3,000
*.boozt.com	$50 - $150	$300 - $500	$1,000 - $1,500	$2,000 - $3,000
com.boozt.app	$200 - $400	$750 - $1,000	$2,500 - $3,500	$4,500 - $5,500

In this bug bounty program, the minimum reward is $50 and the maximum is the $5000.

1. Program Rules

Please provide detailed reports with reproducible steps. If the report is not detailed enough to reproduce the issue, the issue will not be eligible for a reward.

- Submit one vulnerability per report, unless you need to chain vulnerabilities to provide impact.
- When reporting vulnerabilities, please consider attack scenario, exploitability, and security impact of the bug on our customers.
- When duplicates occur, we only award the first fully reproducible report that we receive.
- Multiple vulnerabilities caused by one underlying issue will be awarded one bounty. This also applies in the case of the vulnerability affecting both Boozt and Booztlet domains.
- Social engineering (e.g. phishing, vishing, smishing) is prohibited.
- If you require an account with our services to showcase a vulnerability, please use your @wearehackerone.com email when registering.
- Make a good faith effort to avoid privacy violations, destruction of data, and interruption or degradation of our service. Only interact with accounts you own or with the explicit permission of the account holder.

2. How we determine severity of a report

- After a report has been triaged by HackerOne, we will conduct an internal investigation to understand the impact of the vulnerability on our customers; the final severity will always factor in an assessment of how the issue affects our customers. Issues that can only be leveraged to attack one's own account will have their severity score or applicability reflect this.
- Reports for security issues that aren't vulnerabilities in our systems might receive a bounty based on our assessment of the impact of the finding.

3. Out of Scope

The following issues are considered out of scope:

- Best practice concerns: evidence of a security issue is required.

- Issues that can only be leveraged to attack one's own account.
- Clickjacking on pages with no sensitive actions.
- Attacks requiring MITM or physical access to a customer's device.
- Use of known vulnerable libraries without a working proof of concept showcasing leverage of that vulnerability.
- CSV injection without demonstrating a vulnerability.
- Any vulnerability affecting the availability of Boozt systems (e.g. denial of service vulnerabilities).
- Missing HttpOnly or Secure flags on cookies.
- Vulnerabilities only affecting users of outdated or unpatched browsers (less than two stable versions behind the latest released stable version).
- Software version disclosure.
- Public 0-day vulnerabilities that have had an official patch for less than one month will be awarded on a case-by-case basis.
- Tabnabbing.
- Open redirect - unless an additional security impact can be demonstrated.
- Sessions being hijacked because of insecure protocol use.
- Reports from automated tools or scans.
- Spam techniques.
- Code obfuscation in mobile applications.
- Issues relating to password policies.
- Race conditions that don't compromise the security of Boozt or our customers.
- Issues that require unlikely customer interaction.
- Issues related to hardcoded vendor tokens on mobile applications which don't compromise the security of Boozt or our customers.
- User enumeration vulnerabilities.

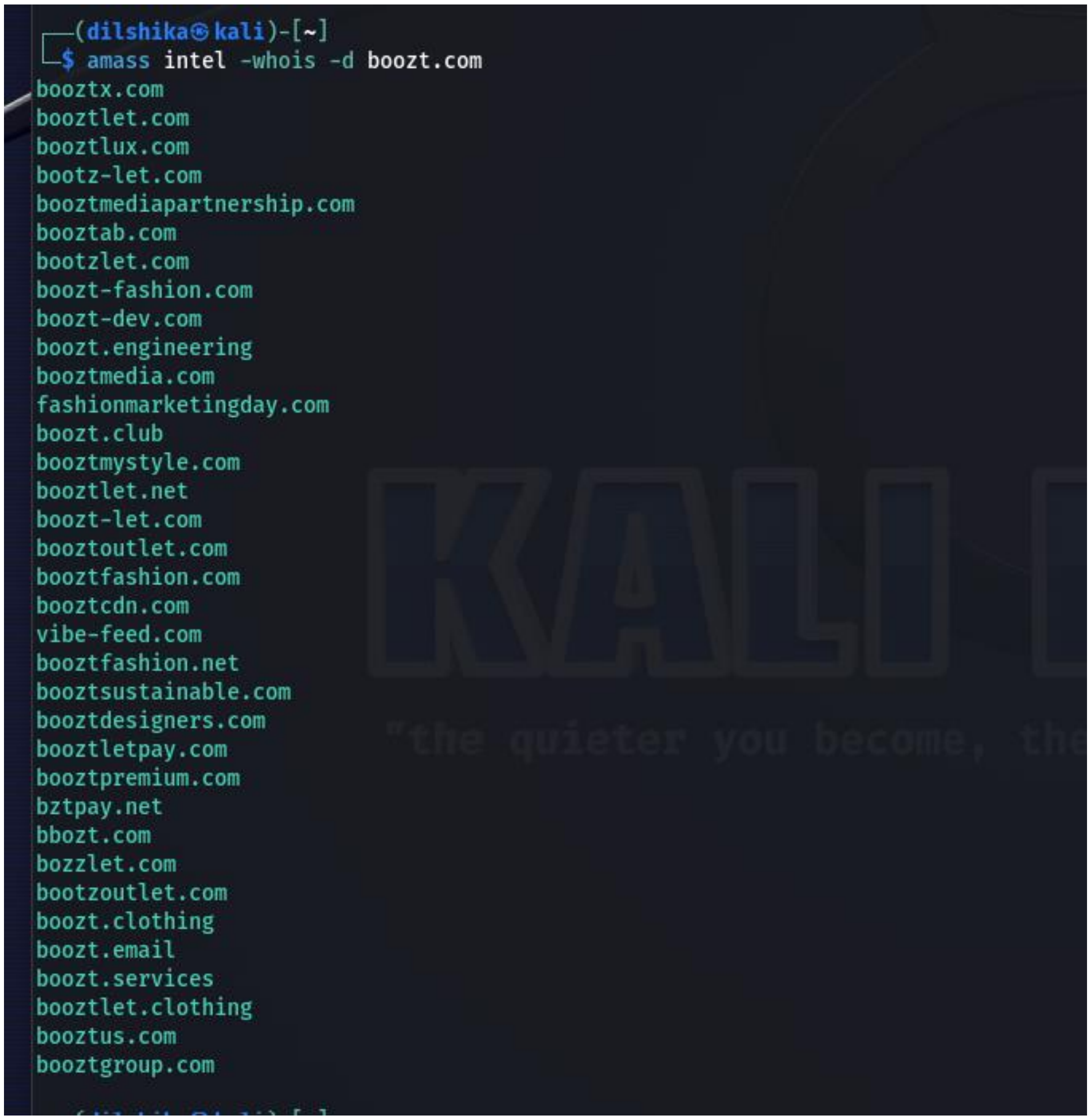
4. Known issues

The following issues are known to us and are being actively worked on. We consider them to be out of scope while remediation is in progress.

- Cross-Site Request Forgery (CSRF) vulnerabilities on the /api/me/favorites endpoint.
- Issues related to mobile application authentication tokens.

- Brute-force issues, in particular with authentication endpoints.

5. Sub domains of the main domain



```
┌──(dilshika㉿kali)-[~]
└─$ amass intel -whois -d boozt.com
booztx.com
booztlet.com
booztlux.com
bootz-let.com
booztmediapartnership.com
booztab.com
bootzlet.com
boozt-fashion.com
boozt-dev.com
boozt.engineering
booztmedia.com
fashionmarketingday.com
boozt.club
booztmystyle.com
booztlet.net
boozt-let.com
booztoutlet.com
booztfashion.com
booztcdn.com
vibe-feed.com
booztfashion.net
booztsustainable.com
booztdesigners.com
booztletpay.com
booztpremium.com
bztpay.net
bbozt.com
bozzlet.com
bootzoutlet.com
boozt.clothing
boozt.email
boozt.services
booztlet.clothing
booztus.com
booztgroup.com
```

3.<u>Owasp top 10</u>

The OWASP Top 10 is a widely recognized and respected document that lists the top ten most critical security risks facing web applications. It's created and regularly updated by the Open Web

Application Security Project (OWASP), a non-profit organization focused on improving software security.

1. **Injection:** Injection vulnerabilities occur when attackers insert untrusted data into an input or command, which is then interpreted by a system component. For instance, SQL injection involves inserting SQL commands into input fields, potentially allowing attackers to manipulate the database or access unauthorized data. Other types include OS injection and LDAP injection. Preventive measures include using parameterized queries, input validation, and escaping user input to mitigate these risks.

2. **Broken Authentication:** Broken authentication refers to weaknesses in the mechanisms used for user authentication in web applications. Attackers may exploit these weaknesses to impersonate users, steal session tokens, or bypass authentication altogether. Common vulnerabilities include weak passwords, session fixation, and improper session management. Mitigation strategies involve implementing secure authentication mechanisms, enforcing strong password policies, and employing multi-factor authentication.

3. **Sensitive Data Exposure:** Sensitive data exposure occurs when web applications fail to adequately protect confidential information, such as personal or financial data. Attackers can intercept, modify, or steal this data, leading to identity theft, financial fraud, or privacy violations. To mitigate this risk, developers should encrypt sensitive data, use secure transmission protocols (e.g., HTTPS), and implement access controls to limit data exposure.

4. **XML External Entities (XXE):** XXE vulnerabilities arise when applications parse XML input from untrusted sources without proper validation. Attackers can exploit these vulnerabilities to access sensitive data or execute remote code. Mitigation techniques include disabling XML external entity processing, validating XML input, and using secure XML parsers that prevent entity expansion attacks.

5. **Broken Access Control:** Broken access control occurs when web applications fail to enforce proper access controls, allowing attackers to perform unauthorized actions. This can include modifying or deleting data, accessing restricted resources, or escalating privileges. To

mitigate this risk, developers should implement access controls at both the application and database levels, validate user permissions, and enforce least privilege principles.

6. **Security Misconfiguration:** Security misconfiguration refers to insecure configurations, default settings, or incomplete setups in web applications. Attackers can exploit these misconfigurations to gain unauthorized access, expose sensitive data, or compromise the application's security. Preventive measures include regularly updating software, disabling unnecessary features, and properly configuring security settings, including those for cloud environments.

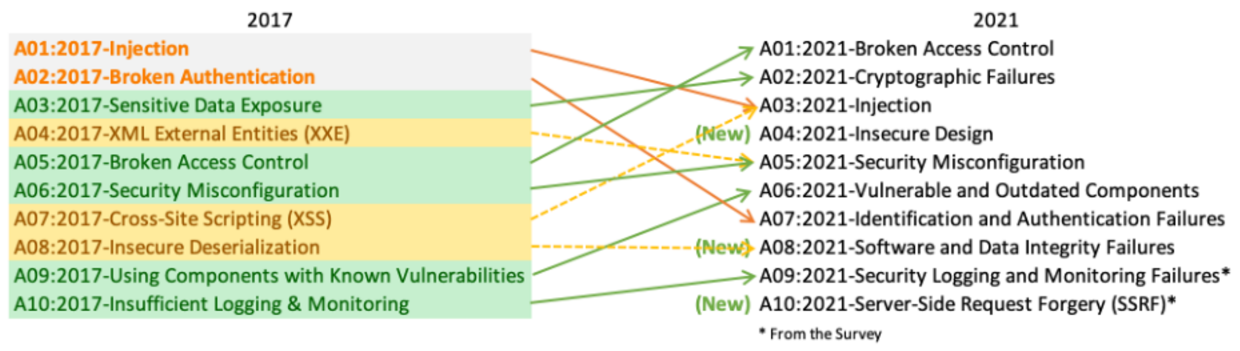
7. **Cross-Site Scripting (XSS):** XSS vulnerabilities occur when attackers inject malicious scripts into web pages viewed by other users. These scripts can hijack user sessions, steal cookies, or deface websites. Mitigation techniques involve input validation, output encoding, and implementing Content Security Policy (CSP) to prevent unauthorized script execution.

8. **Insecure Deserialization:** Insecure deserialization vulnerabilities arise when applications deserialize untrusted data without proper validation. Attackers can exploit these vulnerabilities to execute arbitrary code, perform remote code execution, or escalate privileges. Mitigation strategies include input validation, integrity checks, and using secure deserialization libraries.

9. **Using Components with Known Vulnerabilities:** This risk involves using third-party libraries or software components with known security flaws. Attackers can exploit these vulnerabilities to compromise the application's security, execute arbitrary code, or steal sensitive data. To mitigate this risk, developers should regularly update dependencies, monitor for security advisories, and use vulnerability scanners to identify and remediate vulnerabilities.

10. **Insufficient Logging and Monitoring:** Insufficient logging and monitoring occur when web applications lack adequate logging and monitoring systems. This can enable attackers to maintain persistence, evade detection, or escalate attacks unnoticed. Mitigation

measures include implementing comprehensive logging, setting up intrusion detection systems, and establishing incident response procedures to detect and respond to security incidents effectively.


2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

4.Tools used

There are plenty of tools that a bug bounter can use for programs there can be automated tools as well as manual methods for reconnaissance , find vulnerabilities, exploit.

Following are the tools that I utilized during the bug bounty hunting

- Owasp zap
- Burp suite
- XSSer
- Nikto
- Sql map
- Wapiti
- Amass
- Nmap
- Nslookup up

1. OWASP ZAP

Owasp zap also called as zed attack proxy is a free and open-source web application scanner that is widely used in the world.

This tool mainly has two type of scans namely automated scan and the manual scan.

How you can use it

• Three interfaces

– Desktop

– API

– Heads Up Display (HUD - new)

Automated scans

1. Launch ZAP and select the Workspace Window's Quick Start tab.

2. Press the large "Automated Scan" icon.

3. Type the entire URL of the website you wish to target into the URL to attack text field.

4. Select Attack

ZAP will then use its spider to progressively crawl the web application, passively scanning every page it comes across.

ZAP will then attack all of the found pages, functionality, and parameters using the active scanner.

Manual Exploration

1. Launch ZAP and select the Workspace Window's Quick Start tab.

2. Press the large Manual

Explore button.

3. Type the complete URL of the web application you wish to explore into the URL to explore text box.

4. Decide which browser you want to use.

5. Select "Launch Browser."

2. Burp suite

Burp Suite which is an integrated platform s used to test web applications for security. Its many tools function in together to facilitate every step of the testing process, from the first mapping and analysis of the attack surface of an application to the identification and exploitation of security flaws.

With Burp, you have complete control over combining cutting-edge automation with sophisticated manual techniques to increase productivity, enjoyment, and speed of work.

3. Sql map

sqlmap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

I used sql map to check whether the website is vulnerable to any sql injection

4. XSSer

A web-based application's Cross Site "Scripter" (also known as XSSer) is an automatic framework for identifying, taking advantage of, and reporting XSS vulnerabilities.

It offers multiple ways to attempt evading specific filters as well as a number of unique code injection techniques.

5. Wapiti

Wapiti is a powerful tool used to audit the security of web applications. It conducts "black-box" scans, meaning it examines the web pages of deployed applications without analyzing the source code directly. Instead, it looks for vulnerabilities by injecting data into scripts and forms and observing the application's response.

Wapiti can detect the following vulnerabilities:

- Database Injection (PHP/ASP/JSP SQL Injections and XPath Injections)
- Cross Site Scripting (XSS) reflected and permanent
- File disclosure detection (local and remote include, require, fopen, readfile…)
- Command Execution detection (eval(), system(), passtru()…)
- XXE (Xml external Entity) injection

- CRLF Injection
- Search for potentially dangerous files on the server (thank to the Nikto db)
- Bypass of weak htaccess configurations
- Search for copies (backup) of scripts on the server
- Shellshock
- DirBuster like
- Server-Side Request Forgery (through use of an external Wapiti website)

6. Amass

Amass is a powerful open-source reconnaissance tool designed for network mapping and information gathering. Researchers and security experts use it frequently to map out external network space and find assets that are owned by a target organization.

The tool collects data about the target network using a variety of methods, such as:

- DNS enumeration: To learn more about the target's domain name system, Amass sends queries to DNS servers.
- Data scraping from search engines: The program searches for the target's online presence (websites, social media profiles, and other online platforms) and gathers information about it.
- Web crawling: Amass searches the target's webpages for possible points of attack, like weak web applications.
- Reverse IP lookups: The tool checks which other domains are hosted on the same IP address as the target, potentially uncovering additional attack vectors.

5.vulnerabilities I found

- PII disclosure
- Sql Injection
- Vulnerable js library
- Cross domain misconfiguration
- CSP : wildcard

- Absence of anti-csrf token
- Hash disclosure
- Apart from these I tried to perform xss on the website

1. PII disclosure

Owasp top 10 – sensitive data exposure (A03:2017)/ Cryptographic failures(A02:2021)

PII (Personally Identifiable Information) disclosure vulnerability refers to a flaw or weakness in a system or application that could potentially expose sensitive personal information about individuals, this information can be used to identify or contact a person uniquely or can be traced back to a specific individual PII includes data such as names, addresses, social security numbers, credit card number, phone numbers, email addresses, financial information, and more

2. sql injection

Owasp top 10 – Injection (rank 3)

SQL injection, or SQLI for short, is a well-known attack vector that leverages malicious SQL code to manipulate backend databases and retrieve data that wasn't meant to be shown. Any number of things, such as user lists, private customer information, or sensitive company data, may be included in this information.

In my bug bounty program, I found 3 vulnerabilities related to sql injection. They are,

- SQL Injection hypersonic SQL
- SQL Injection – SQLite
- SQL Injection – Oracle

3. Vulnerable js library

Owasp top 10 - Using Components with Known Vulnerabilities

vulnerable JS library is a security vulnerability that occurs due to the use of outdated or unpatched js libraries. Cybercriminals often target web applications that utilize these vulnerable libraries, exploiting them for malicious purposes.

4. Cross domain misconfiguration

Owasp top 10 - Security Misconfiguration(rank 5)

Cross-domain misconfiguration, also known as cross-origin misconfiguration, refers to a security vulnerability that occurs when a web application improperly allows interactions between different domains or origins. Web browsers have a security feature called the Same-Origin Policy (SOP), which prevents scripts on one domain from accessing resources on another domain, to mitigate various types of attacks such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF)

5. CSP : wildcard

Owasp top 10 – Insecure design (rank 4)

The "Content Security Policy" (CSP) is a security standard that enables web developers specify which content sources are acceptable for their online application, therefore assisting in the prevention of many sorts of attacks, including data injection and Cross-Site Scripting (XSS). A component of the Content Security Policy is the CSP Wildcard Directive, sometimes referred to as the default-src directive.

6. Absence of anti-csrf token

Owasp top 10- Insecure design(rank 4)

The absence of anti csrf token leaves a web application vulnerable to csrf attacks. CSRF tokens are unique , random values generated by the server and embedded in web forms. They help to prevent unauthorized commands executed on behalf of user . without these tokens, attackers can forge request , leading to unauthorized actions behalf of the user

7. Hash disclosure

Owasp top 10 – cryptographic failures(rank 2)

An application that inadvertently divulges a hashed version of a password is known as a hash disclosure vulnerability. Although this might seem safe because the actual password isn't exposed, it can still be dangerous.

Hash disclosure - SHA-256 crypt refers to a vulnerability where the web server inadvertently exposes hashed passwords that have been hashed using the SHA-256 crypt algorithm on a system.

6.Daily logs

- **2024.3.22**

Started the assignment. First, I wanted to select a domain as my bug bounty program. I have not done bug bounty previously so I searched how to selected a domain.

For that I referred some YouTube videos. Following are those video links,

https://www.youtube.com/watch?v=vbXpRHcKIr0

https://www.youtube.com/watch?v=F9dV5lH8nvo&list=RDCMUCCZDt7MuC3Hzs6IH4xODLBw&start_radio=1&rv=F9dV5lH8nvo&t=10

✓ **Takeaways**

we need to check scope, rewards in the program. And choosing

✓ **Challenges**

there are so many videos in you tube about how to select a domain, most of them said

- **2024.4.4**

As a beginner for bug bounty, I did not know much about tools, methodologies , platforms for bug bounty.

Then I found an amazing article,

https://infosecwriteups.com/bug-hunting-methodology-for-beginners-20b56f5e7d19 this article covered wide range of basic information we need to know as a bug bounty programs.

It mentioned about bug bounty platforms. I only knew about hacker one, bug crowd and integrity but mentioned about some other bug bounty platforms as well.

Most important thing to me was it mentioned about useful YouTube channels for learning

I referred some videos

- ✓ **Takeaways**

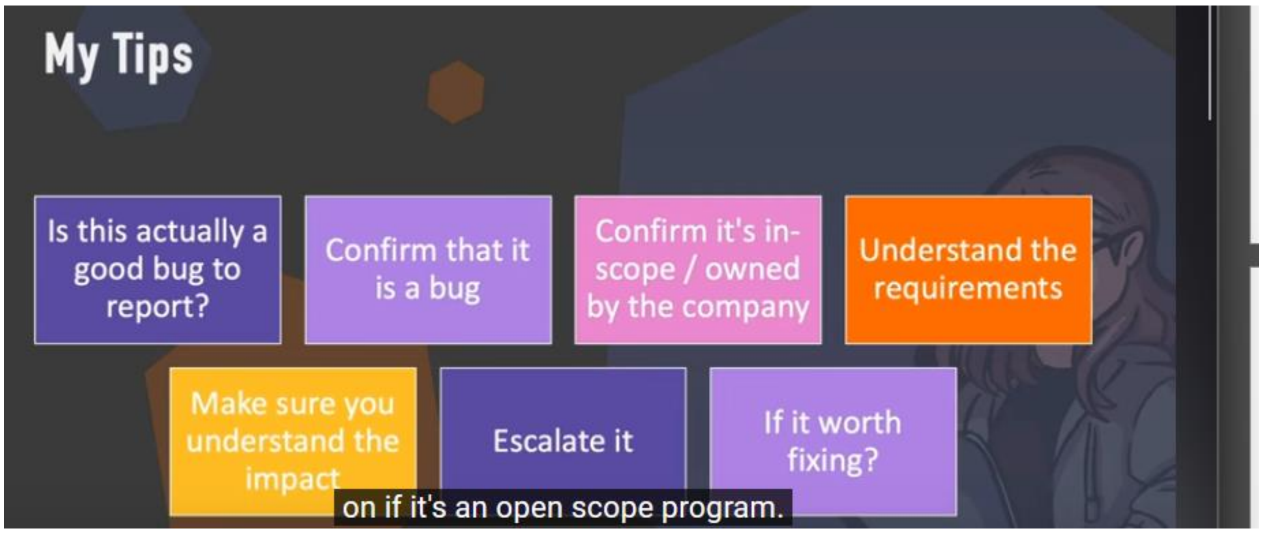
https://www.youtube.com/watch?v=qlzbzfNAXXE

when to report a bug

how to find sub domains – it mentioned some tools such as subfinder, Amass, sublist3r

Different tools we can use – burp suite, xsser, google dorking, shodan, Metasploit, sql map , nmap

take aways from the videos -



My Tips

- Is this actually a good bug to report?
- Confirm that it is a bug
- Confirm it's in-scope / owned by the company
- Understand the requirements
- Make sure you understand the impact
- Escalate it
- If it worth fixing?

on if it's an open scope program.

https://www.youtube.com/watch?v=OsIXSuVO8ig – Bug bounty fundamentals

- ✓ **Takeaways**

scope : why there is out of scope in programs

what happen if we find a bug in out of scope – most probably we will not get a reward

top 5 bug bounty tips - https://www.google.com/search?client=firefox-b-d&sca_esv=88efb82a0fd29766&sxsrf=ACQVn0_FEQxGgHHDYA7jPkPXhK6zJ3grZg:1714628535164&q=bug+bounty+tips&tbm=vid&source=lnms&prmd=visnmbz&sa=X&ved=2ahUKEwj

qioqGoe6FAxULbmwGHaINBTQQ0pQJegQIFBAB&biw=766&bih=730&dpr=1.25#fpstate=ive&vld=cid:75167493,vid:QLYVXIme10I,st:0

- ✓ **Takeaways**

spend time looking for target

use your target until you are bored

test wisely

test every parameter for reflection

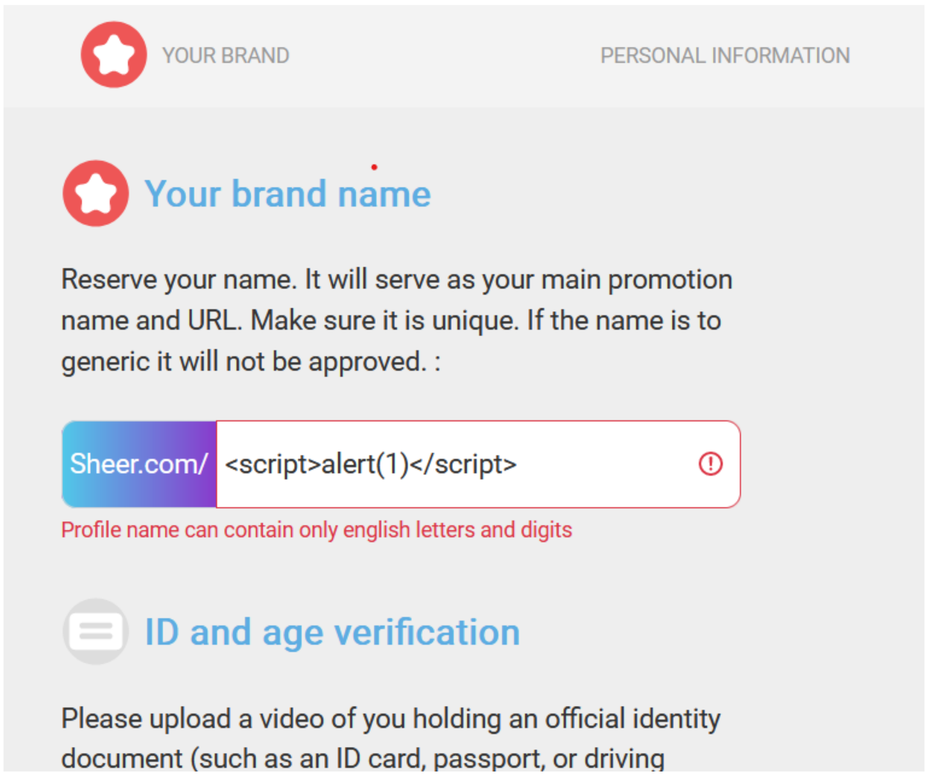
pick your targets wisely

- **2024.4.7**

I choose a domain; The platform is hacker one. The bug bounty program is sheer.com.

This website is a content creator website, u can either register as a content creator or a fan

There was field to enter brand name, the brand name will save as a URL. So, I tried to enter most basic xss payload. But it says it only accept letters and numbers.



Your brand name

Reserve your name. It will serve as your main promotion name and URL. Make sure it is unique. If the name is to generic it will not be approved. :

Sheer.com/ <script>alert(1)</script>

Profile name can contain only english letters and digits

ID and age verification

Please upload a video of you holding an official identity document (such as an ID card, passport, or driving

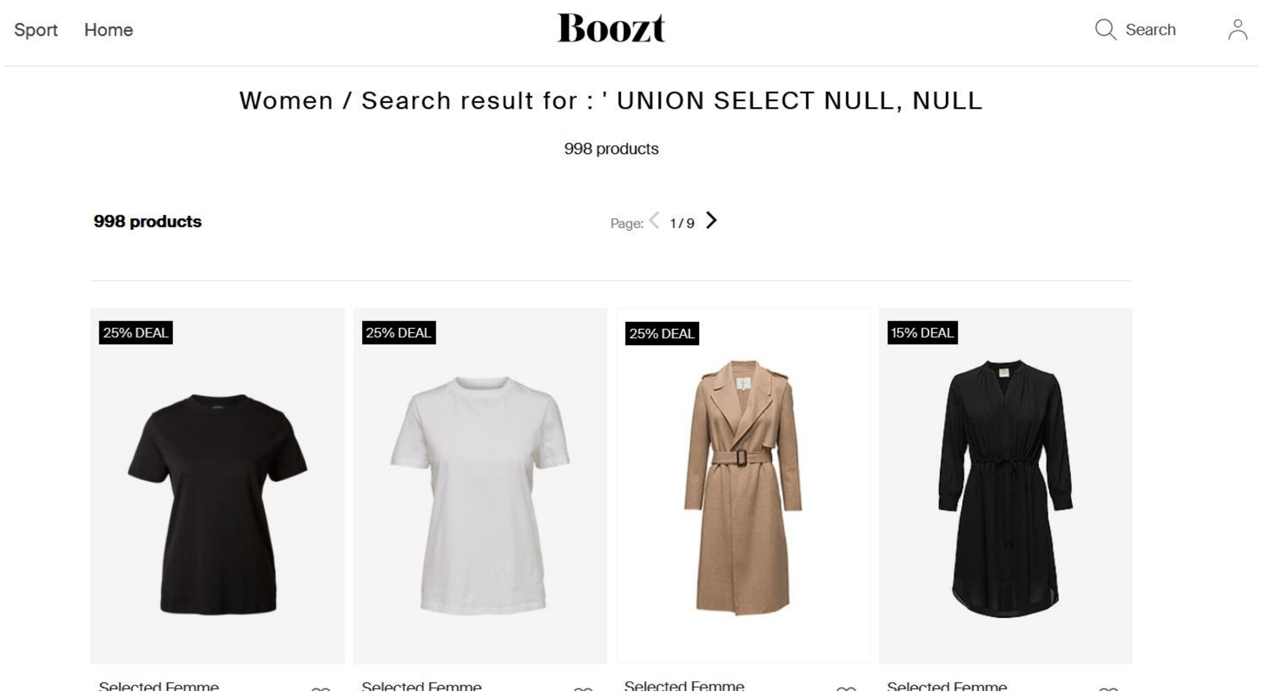
I had some issues in creating an account in sheer website.

So, I had to look for another domain.

Then I found a fashion clothing website called boozt.com

It is a Nordic online fashion store, it has mainly two sites namely, boozt.com and booztlet.com .

First, I did some basic injections like sql injection and xss .



Women / Search result for : ' UNION SELECT NULL, NULL

998 products

998 products

Page: 1 / 9

- ✓ **Challenges**

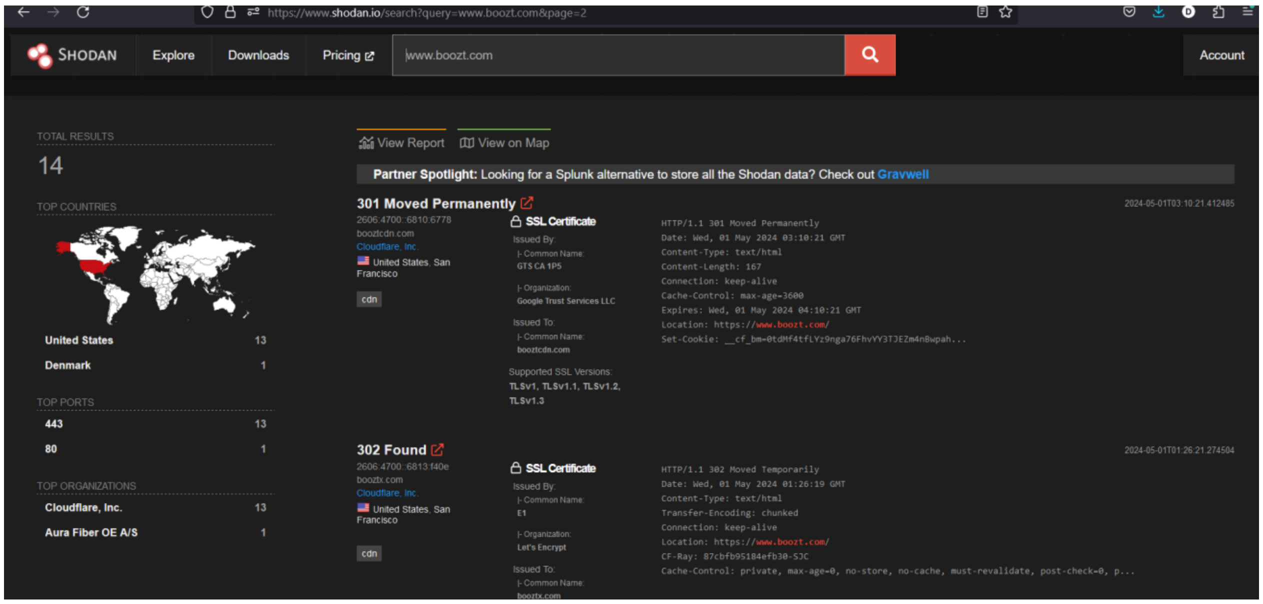
I got tried some sql injections but none of them worked . I got over whelmed as I spent so much time but yet I did not get any results. But I did not give up.

So far, I have found vulnerabilities in portswigger labs, dvwa, pico ctf , try hack me rooms. Nut these are to enhance our skills so they are built to be vulnerable. But bug bounty programs are not like that most of them already has robust security measurements . It is hard to hack then or find bug using basic payloads.

- **<u>2024.4.8</u>**

I searched about tools mentioned in the assignment.

Shodan - Shodan is a search engine for finding specific devices, and device types, that exist online. The most popular searches are for things like webcam, linksys, cisco, netgear, SCADA, etc. shodan can be also used to find sub domains.



TOTAL RESULTS
14

TOP COUNTRIES

United States	13
Denmark	1

TOP PORTS

443	13
80	1

TOP ORGANIZATIONS

Cloudflare, Inc.	13
Aura Fiber OE A/S	1

Partner Spotlight: Looking for a Splunk alternative to store all the Shodan data? Check out Gravwell

301 Moved Permanently
2606:4700::6810:6778
booztcdn.com
Cloudflare, Inc.
United States, San Francisco
cdn

SSL Certificate
Issued By:
|- Common Name:
GTS CA 1P5
|- Organization:
Google Trust Services LLC
Issued To:
|- Common Name:
booztcdn.com
Supported SSL Versions:
TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

2024-05-01T03:10:21.412485

```
HTTP/1.1 301 Moved Permanently
Date: Wed, 01 May 2024 03:10:21 GMT
Content-Type: text/html
Content-Length: 167
Connection: keep-alive
Cache-Control: max-age=3600
Expires: Wed, 01 May 2024 04:10:21 GMT
Location: https://www.boozt.com/
Set-Cookie: __cf_bm=0tdMf4tfLYz9nga76FhvYY3TJEZm4nBwpah...
```

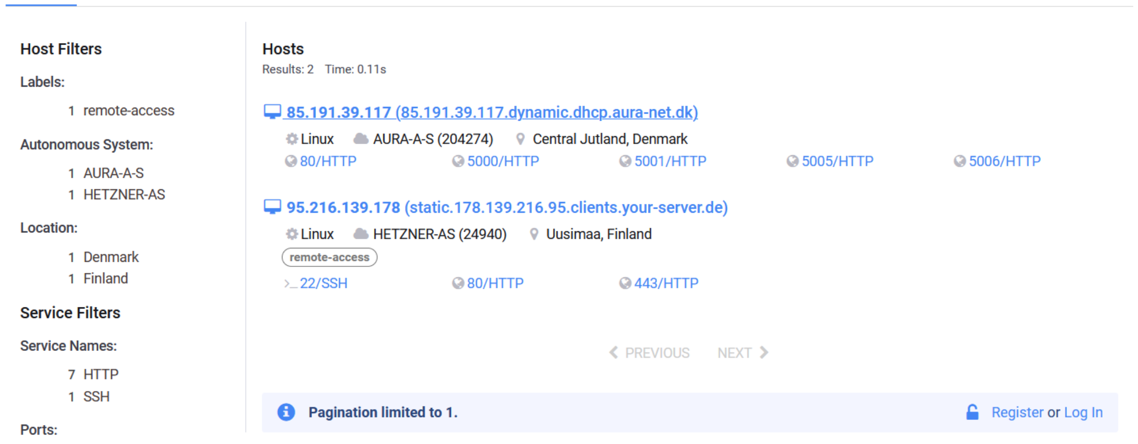
302 Found
2606:4700::6813:f40e
booztx.com
Cloudflare, Inc.
United States, San Francisco
cdn

SSL Certificate
Issued By:
|- Common Name:
E1
|- Organization:
Let's Encrypt
Issued To:
|- Common Name:
booztx.com

2024-05-01T01:26:21.274504

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 01 May 2024 01:26:19 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Location: https://www.boozt.com/
CF-Ray: 87cbfb95184efb30-SJC
Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, p...
```

Cencys – information about a domain when we give ip address or domain name. information like host, ports, status of connection, location, redirection locations, technologies used.



Host Filters

Labels:
1 remote-access

Autonomous System:
1 AURA-A-S
1 HETZNER-AS

Location:
1 Denmark
1 Finland

Service Filters

Service Names:
7 HTTP
1 SSH

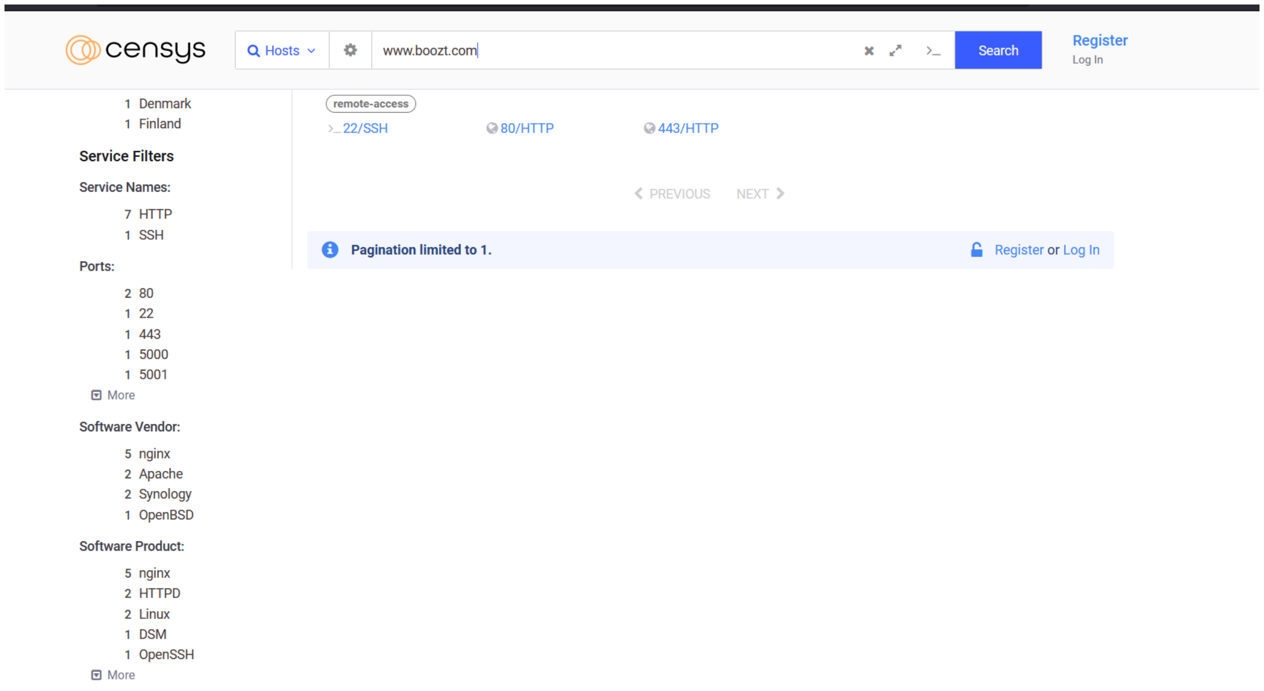
Ports:

Hosts
Results: 2 Time: 0.11s

85.191.39.117 (85.191.39.117.dynamic.dhcp.aura-net.dk)
Linux AURA-A-S (204274) Central Jutland, Denmark
80/HTTP 5000/HTTP 5001/HTTP 5005/HTTP 5006/HTTP

95.216.139.178 (static.178.139.216.95.clients.your-server.de)
Linux HETZNER-AS (24940) Uusimaa, Finland
remote-access
22/SSH 80/HTTP 443/HTTP

Pagination limited to 1.


Hosts
www.boozt.com
Search
Register
Log In

1 Denmark
1 Finland

Service Filters

Service Names:

7 HTTP
1 SSH

Ports:

2 80
1 22
1 443
1 5000
1 5001
More

Software Vendor:

5 nginx
2 Apache
2 Synology
1 OpenBSD

Software Product:

5 nginx
2 HTTPD
2 Linux
1 DSM
1 OpenSSH
More

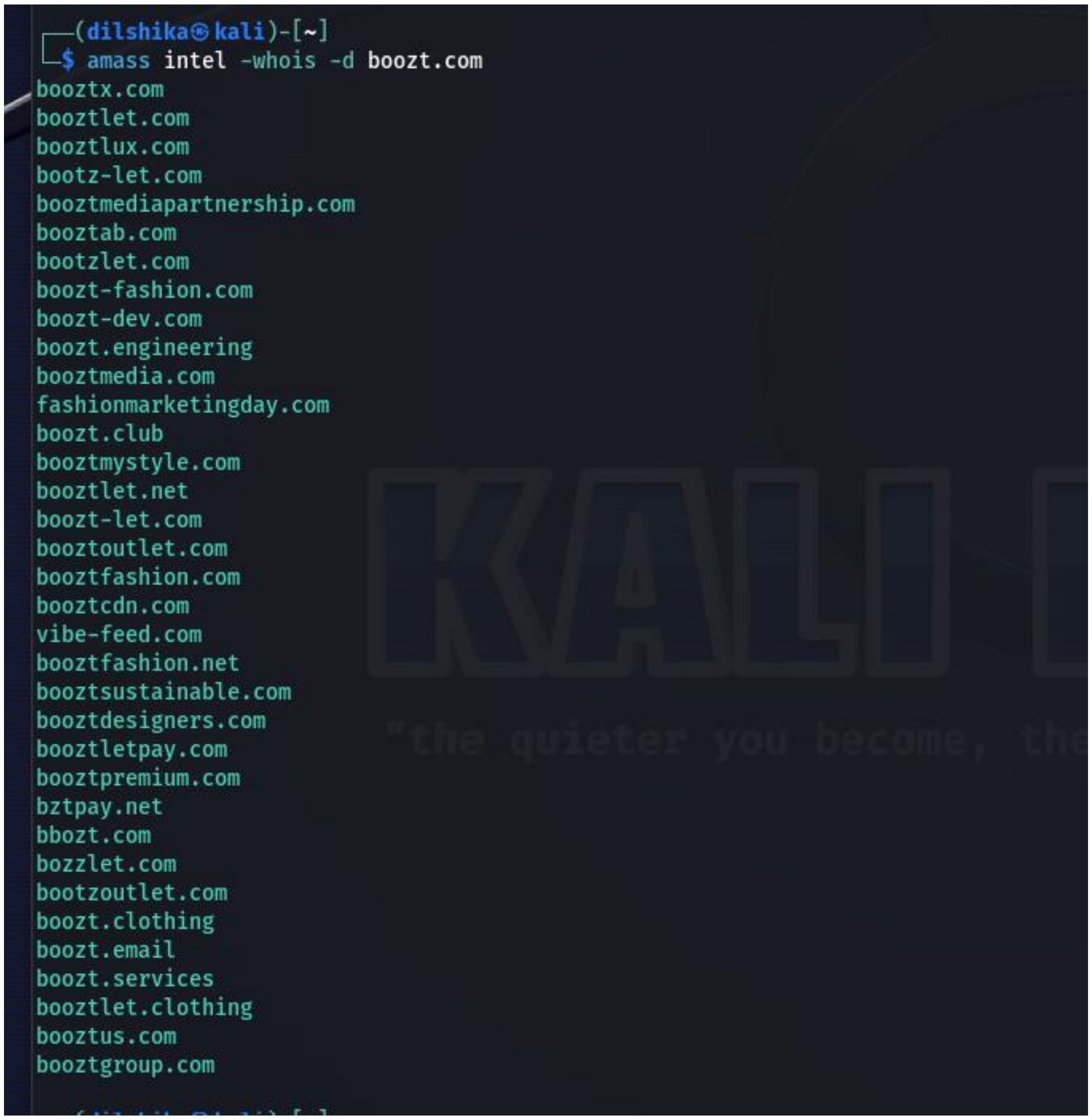
remote-access
22/SSH 80/HTTP 443/HTTP

PREVIOUS NEXT

Pagination limited to 1. Register or Log In

Amass – enumeration tool

I used amass tool and find out the related subdomains of bootz.com and booztlet.com



```
┌──(dilshika㉿kali)-[~]
└─$ amass intel -whois -d boozt.com
booztx.com
booztlet.com
booztlux.com
bootz-let.com
booztmediapartnership.com
booztab.com
bootzlet.com
boozt-fashion.com
boozt-dev.com
boozt.engineering
booztmedia.com
fashionmarketingday.com
boozt.club
booztmystyle.com
booztlet.net
boozt-let.com
booztoutlet.com
booztfashion.com
booztcdn.com
vibe-feed.com
booztfashion.net
booztsustainable.com
booztdesigners.com
booztletpay.com
booztpremium.com
bztpay.net
bbozt.com
bozzlet.com
bootzoutlet.com
boozt.clothing
boozt.email
boozt.services
booztlet.clothing
booztus.com
booztgroup.com
```

- **<u>2024.4.9</u>**

I found a tool called owasp zap which can be used to identify vulnerabilities in a website. I searched about the tool and it is a tool that can be useful for bug hunters specially if they are new to bug hunting.

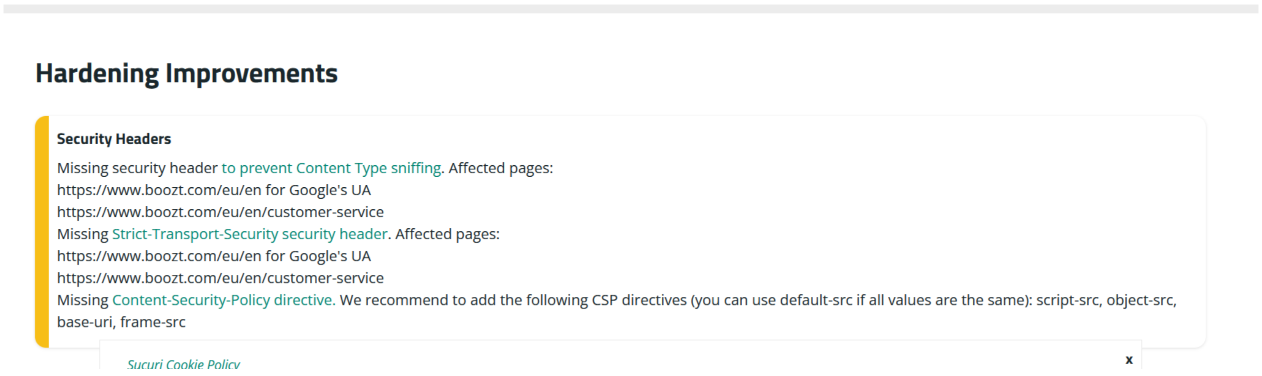
In my kali machine owasp zap was not downloaded, so I had to download it on my machine. Yet I didn't know how to use the tool.

Using owasp zap I scanned subdomains of boozt.com , I found about some csp issues, but still couldn't find any serious vulnerabilities of the website.

I also found a site for sitechecking

https://sitecheck.sucuri.net this site check whether a given domain has malware, whether it is blacklisted, and other security issues.

But it suggested some heading improvements in the domain



Hardening Improvements

Security Headers

Missing security header to prevent Content Type sniffing. Affected pages:
https://www.boozt.com/eu/en for Google's UA
https://www.boozt.com/eu/en/customer-service
Missing Strict-Transport-Security security header. Affected pages:
https://www.boozt.com/eu/en for Google's UA
https://www.boozt.com/eu/en/customer-service
Missing Content-Security-Policy directive. We recommend to add the following CSP directives (you can use default-src if all values are the same): script-src, object-src, base-uri, frame-src

Sucuri Cookie Policy

- **Challenges**

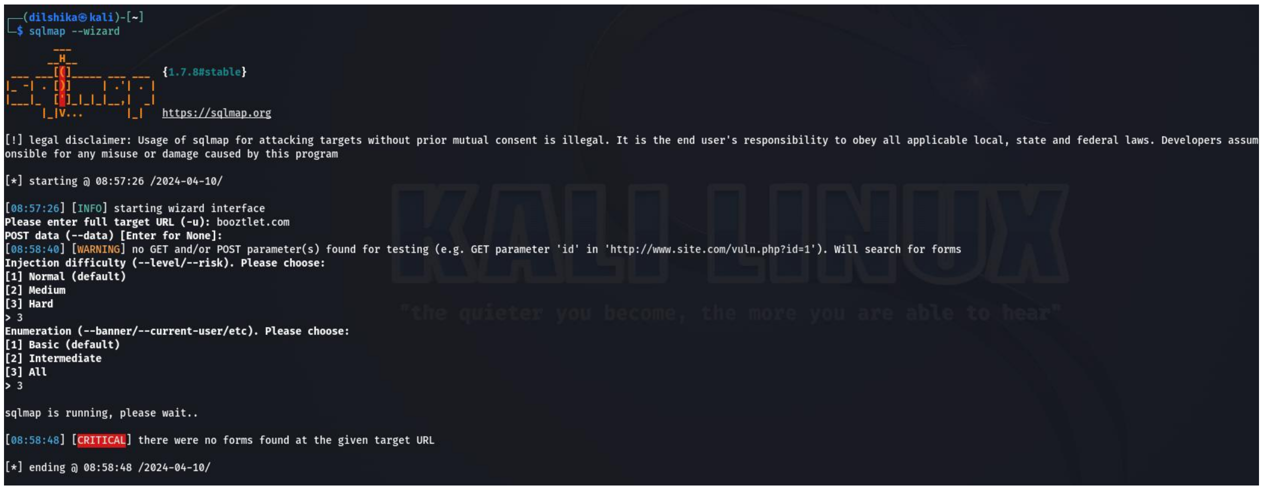
Some of the tools were completely new to me, so I had to learn how to use them.

- <u>**2024.4.10**</u>

Today I started doing sql injection on the website. For that I used sql map tool which is a popular tool for finding sql injection vulnerabilities

First, I started on kali virtual machine and accessed to SQL map tool. In sql map wizard you only have to give the url and then wizard will ask to choose some options like difficulty of attack, enumeration.

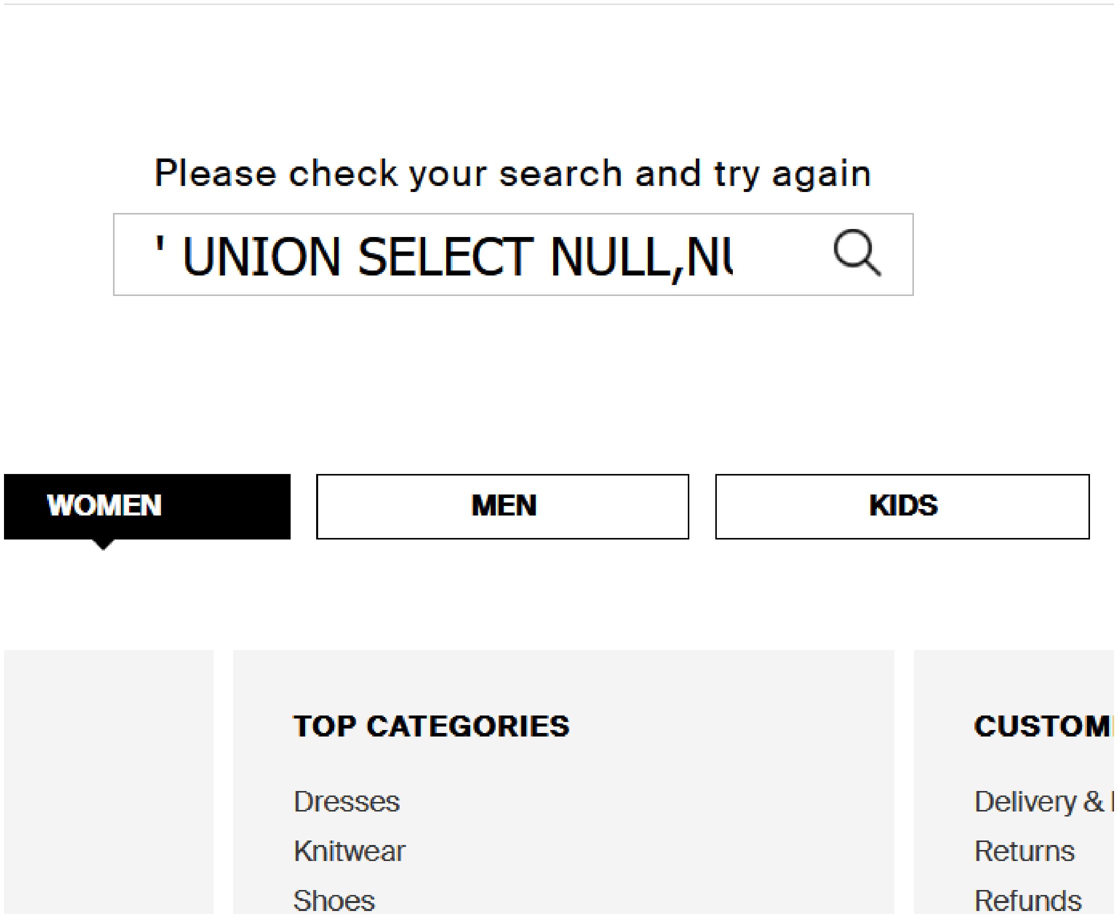
First, I used sql map on www.boozt.com , but it did not give any output it displayed that there were no forms at given target url.



```
┌──(dilshika㉿kali)-[~]
└─$ sqlmap --wizard
                                 {1.7.8#stable}

                                 https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assum
onsible for any misuse or damage caused by this program

[*] starting @ 08:57:26 /2024-04-10/

[08:57:26] [INFO] starting wizard interface
Please enter full target URL (-u): booztlet.com
POST data (--data) [Enter for None]:
[08:58:40] [WARNING] no GET and/or POST parameter(s) found for testing (e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will search for forms
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 3
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
> 3

sqlmap is running, please wait..

[08:58:48] [CRITICAL] there were no forms found at the given target URL

[*] ending @ 08:58:48 /2024-04-10/
```

' UNION SELECT NULL,NULL,NULL-- : This payload is used for testing if the application is vulnerable to union-based SQL injection.

I also tried to inject some sql payloads in the search field of the website. Due to input sanitization and validation the payload that are used are not successful.


Please check your search and try again

' UNION SELECT NULL,NU

- **Takeaways**

Learned some new sql payloads

- **<u>2024.4.11</u>**

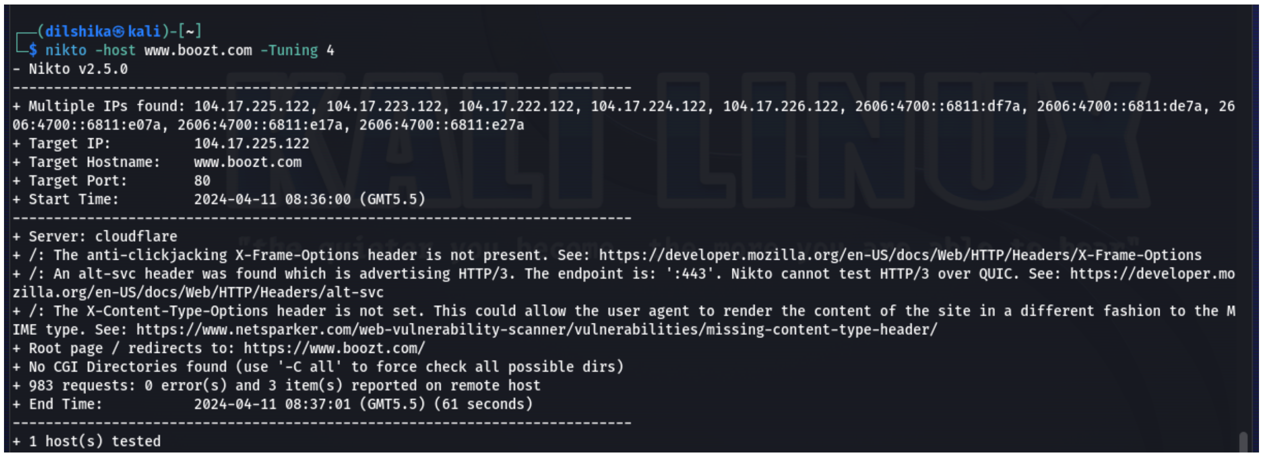
Today I used tool called nikto, nikto is used test a Web Site, Virtual Host and Web Server for known security vulnerabilities and mis-configurations.

Nikto performs over 6000 tests against a website. The large number of tests for both security vulnerabilities and mis-configured web servers

Through the nikto scanning following are what I found

- Anti-hijacking header is not present
- Alt-svc header was found which is advertising HTTP/3 . the end point is : 443
- The x-content-type-option header is not set

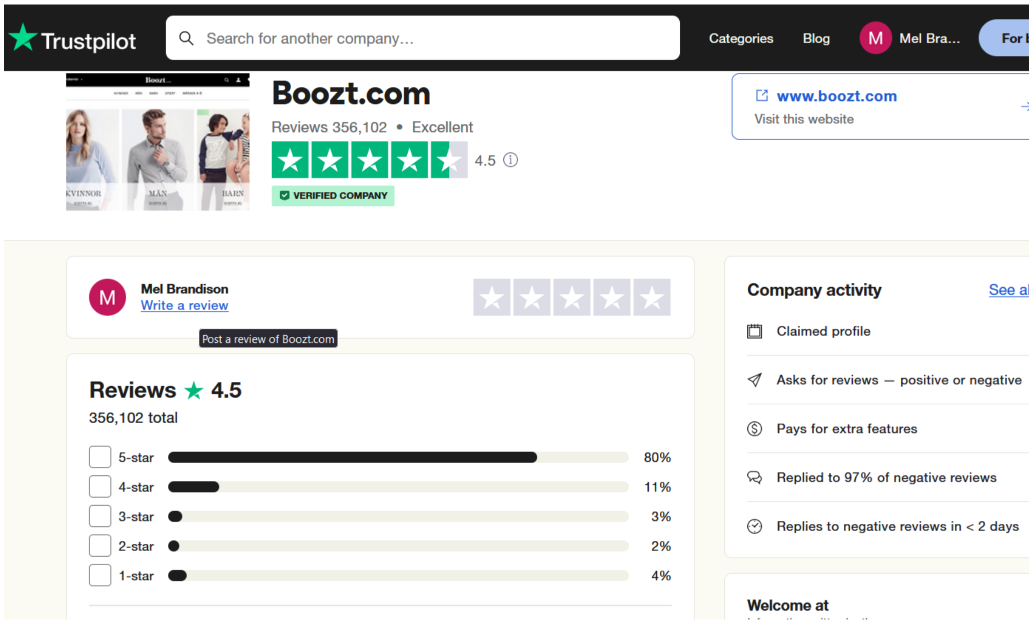
Also, it gave information about the host, the target ip, port using, root page



```
┌──(dilshika㉿kali)-[~]
└─$ nikto -host www.boozt.com -Tuning 4
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Multiple IPs found: 104.17.225.122, 104.17.223.122, 104.17.222.122, 104.17.224.122, 104.17.226.122, 2606:4700::6811:df7a, 2606:4700::6811:de7a, 26
06:4700::6811:e07a, 2606:4700::6811:e17a, 2606:4700::6811:e27a
+ Target IP:          104.17.225.122
+ Target Hostname:    www.boozt.com
+ Target Port:        80
+ Start Time:         2024-04-11 08:36:00 (GMT5.5)
---------------------------------------------------------------------------
+ Server: cloudflare
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: An alt-svc header was found which is advertising HTTP/3. The endpoint is: ':443'. Nikto cannot test HTTP/3 over QUIC. See: https://developer.mo
zilla.org/en-US/docs/Web/HTTP/Headers/alt-svc
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the M
IME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page / redirects to: https://www.boozt.com/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 983 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2024-04-11 08:37:01 (GMT5.5) (61 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Did some cross site scripting attempts

A review aggregation system or third-party review integration where users can submit reviews for products and services on a website, such as www.boozt.com , via a third-party platform like https://uk.trustpilot.com. The 'topic' field allows users to specify the subject of their review, and once submitted, the topic becomes a clickable link, potentially leading to the full review or related content.


Trustpilot | Search for another company… | Categories | Blog | Mel Bra… | For b

Boozt.com
Reviews 356,102 • Excellent
4.5
VERIFIED COMPANY

www.boozt.com
Visit this website

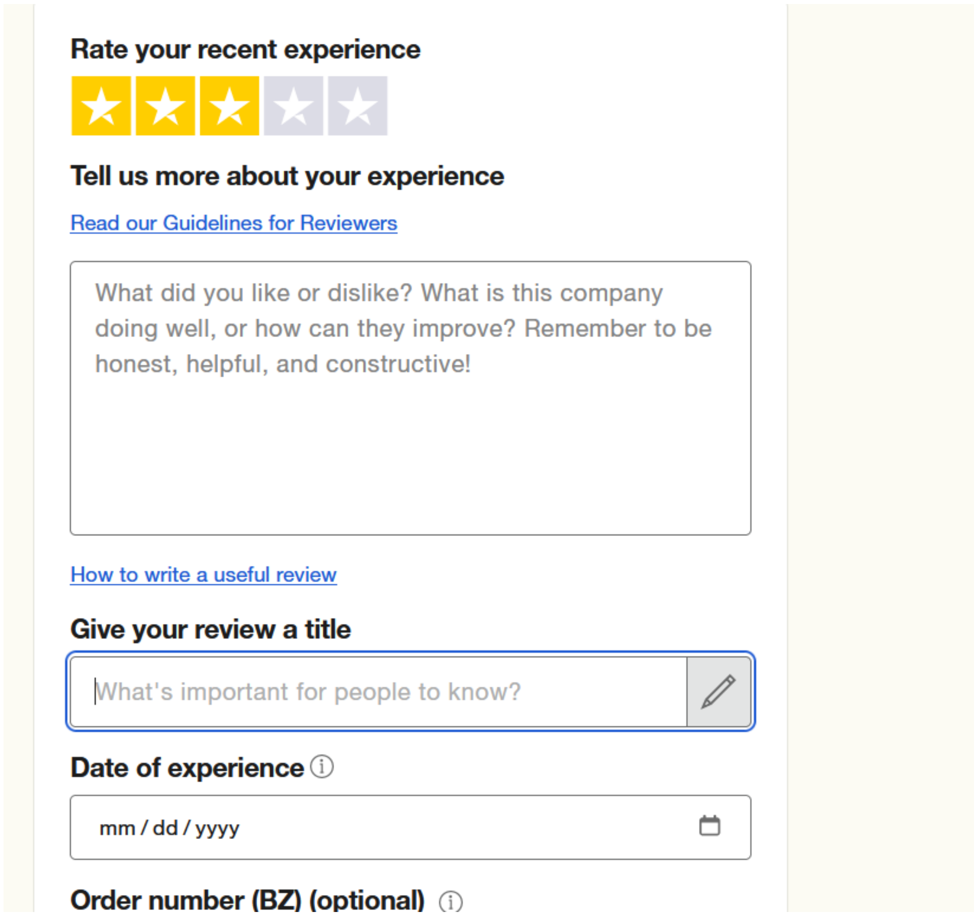
Mel Brandison
Write a review
Post a review of Boozt.com

Reviews ★ 4.5
356,102 total

5-star	80%
4-star	11%
3-star	3%
2-star	2%
1-star	4%

Company activity — See al
- Claimed profile
- Asks for reviews — positive or negative
- Pays for extra features
- Replied to 97% of negative reviews
- Replies to negative reviews in < 2 days

Welcome at


Rate your recent experience

Tell us more about your experience

Read our Guidelines for Reviewers

What did you like or dislike? What is this company doing well, or how can they improve? Remember to be honest, helpful, and constructive!

How to write a useful review

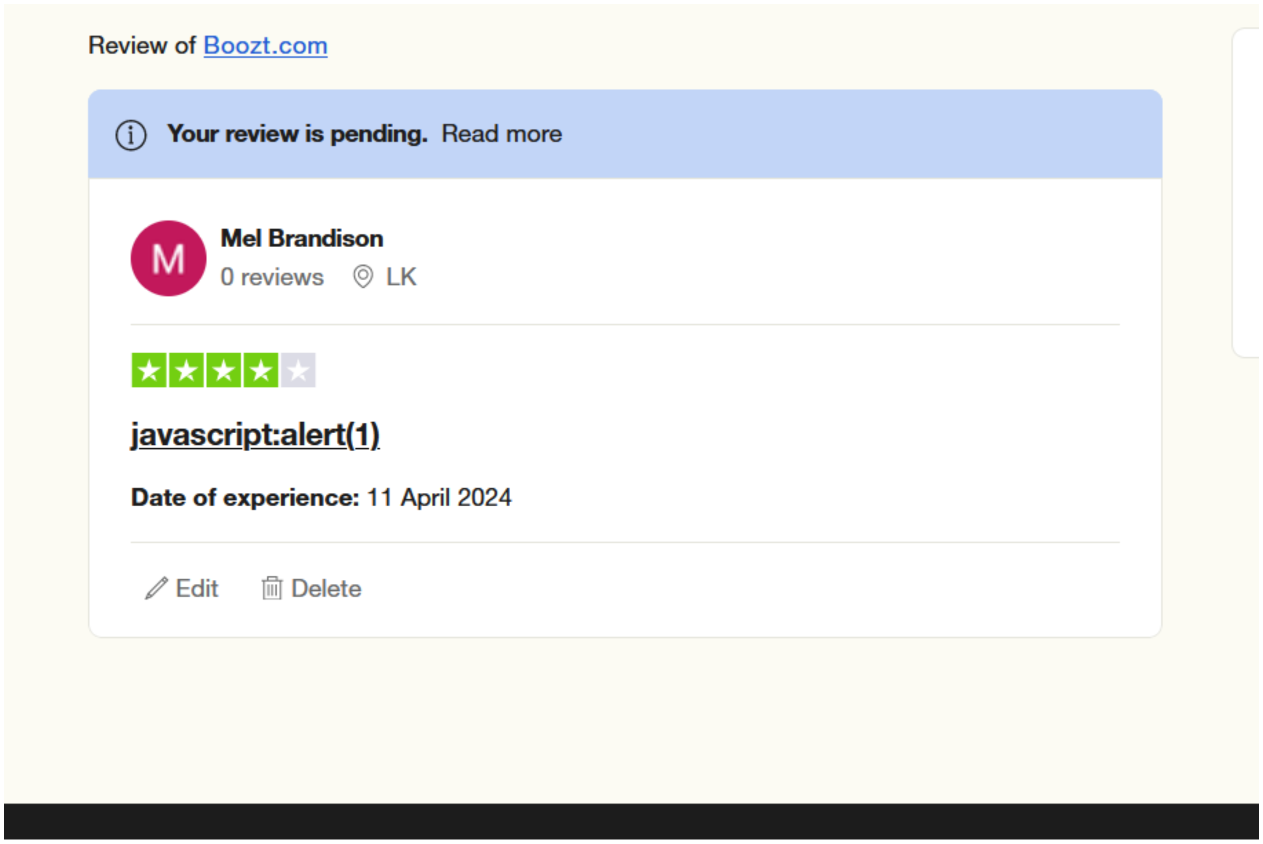
Give your review a title

What's important for people to know?

Date of experience

mm / dd / yyyy

Order number (BZ) (optional)



Review of Boozt.com

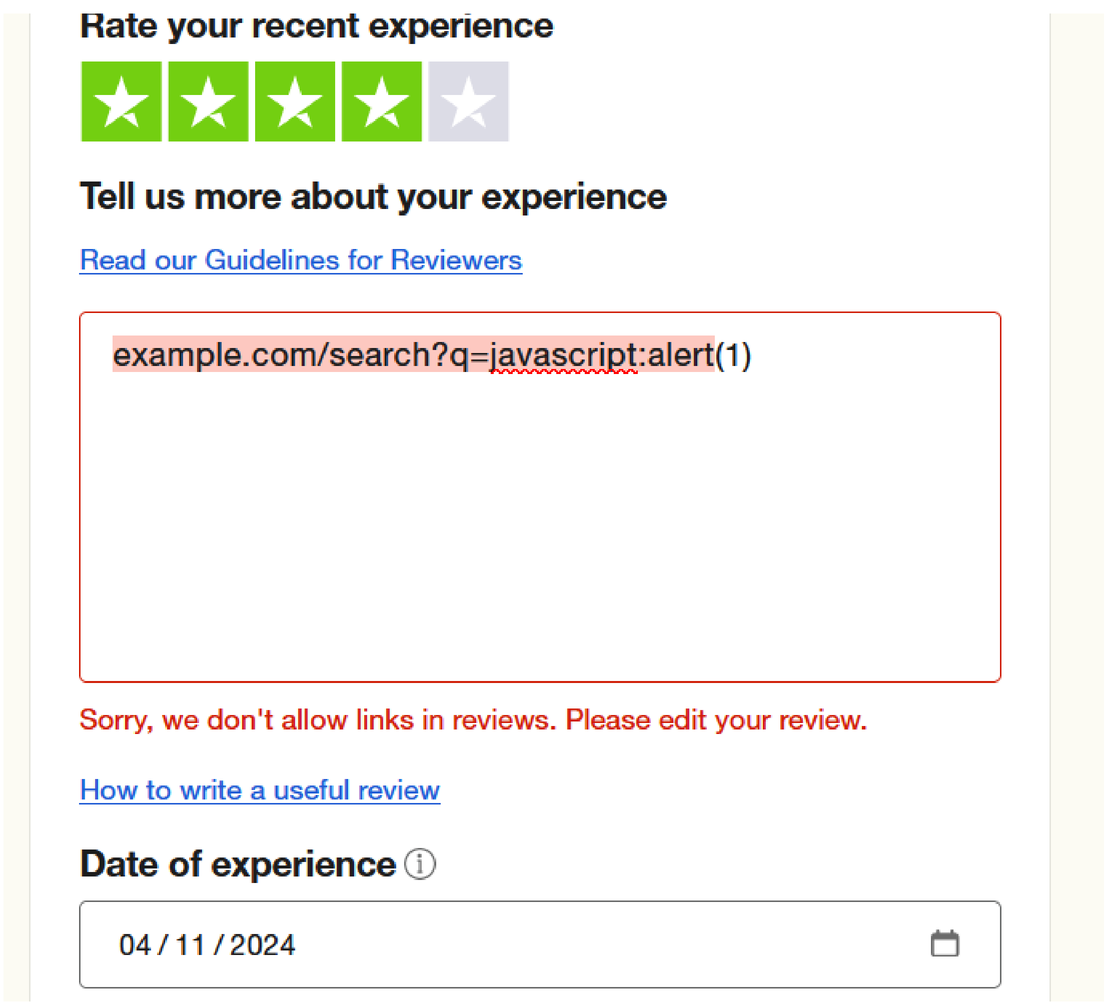
Your review is pending. Read more

Mel Brandison
0 reviews LK

★★★★☆

javascript:alert(1)

Date of experience: 11 April 2024



Rate your recent experience

★★★★☆

Tell us more about your experience

Read our Guidelines for Reviewers

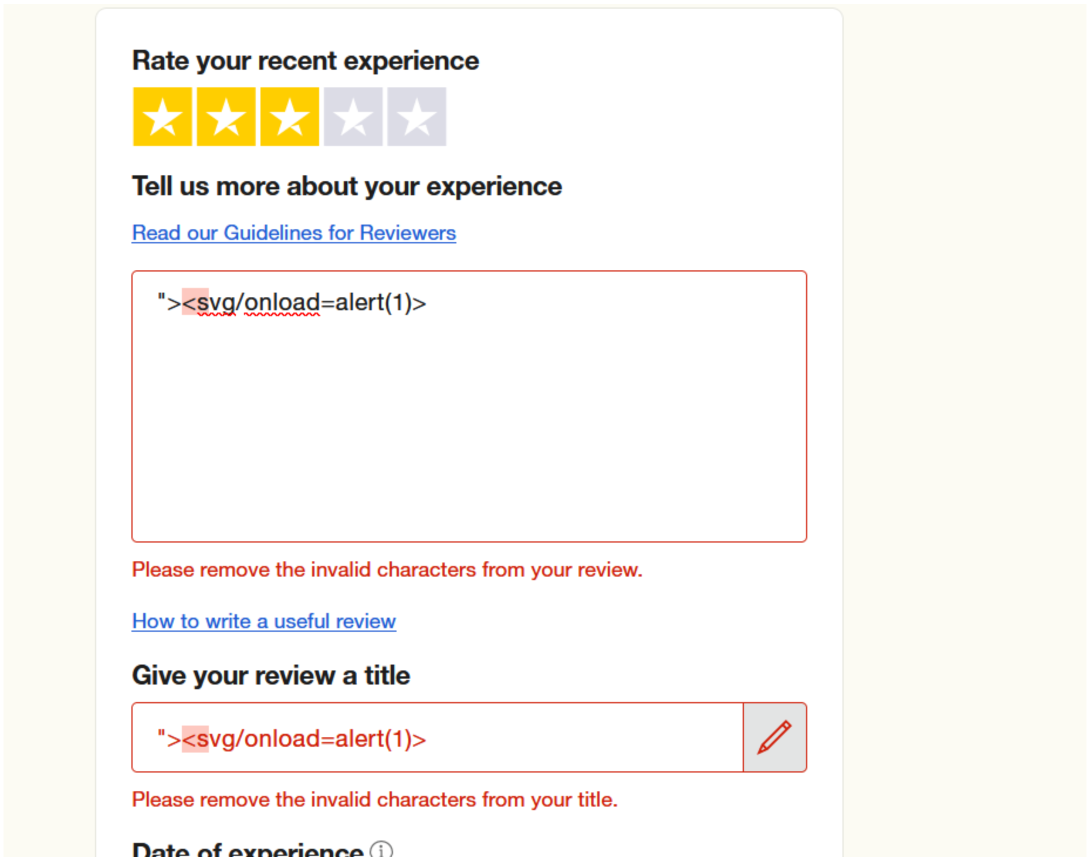
example.com/search?q=javascript:alert(1)

Sorry, we don't allow links in reviews. Please edit your review.

How to write a useful review

Date of experience

04 / 11 / 2024



Rate your recent experience

Tell us more about your experience

Read our Guidelines for Reviewers

```
"><svg/onload=alert(1)>
```

Please remove the invalid characters from your review.

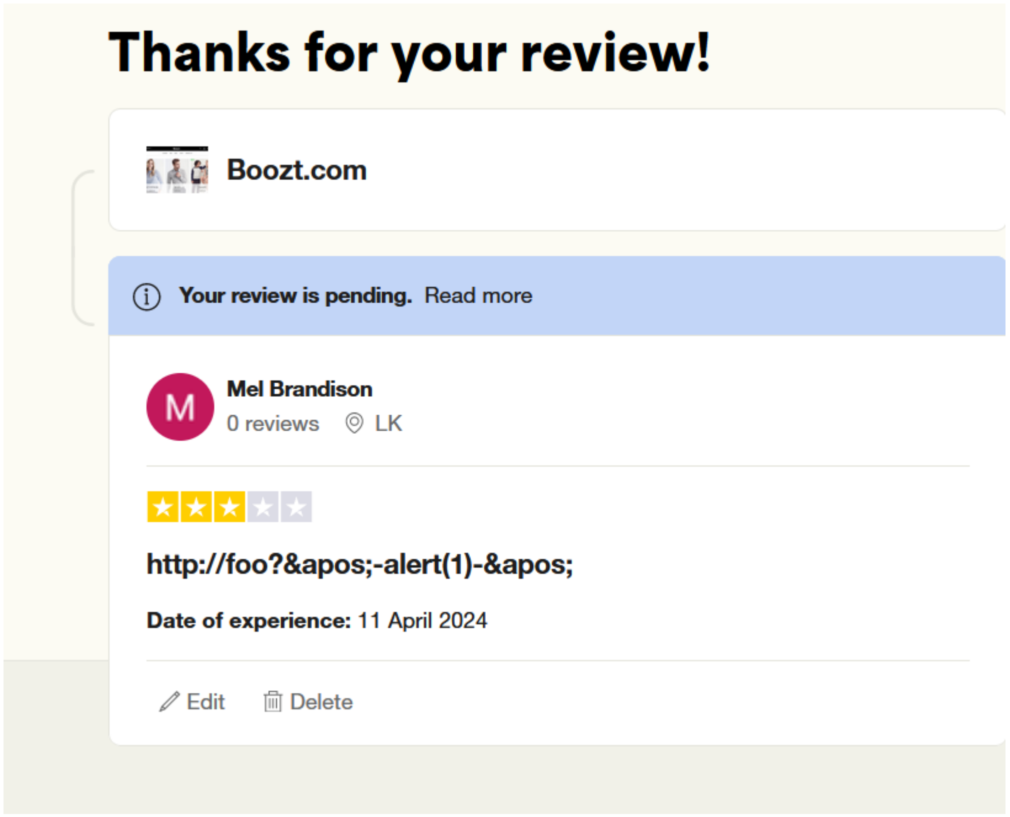
How to write a useful review

Give your review a title

```
"><svg/onload=alert(1)>
```

Please remove the invalid characters from your title.

Date of experience



Thanks for your review!

Boozt.com

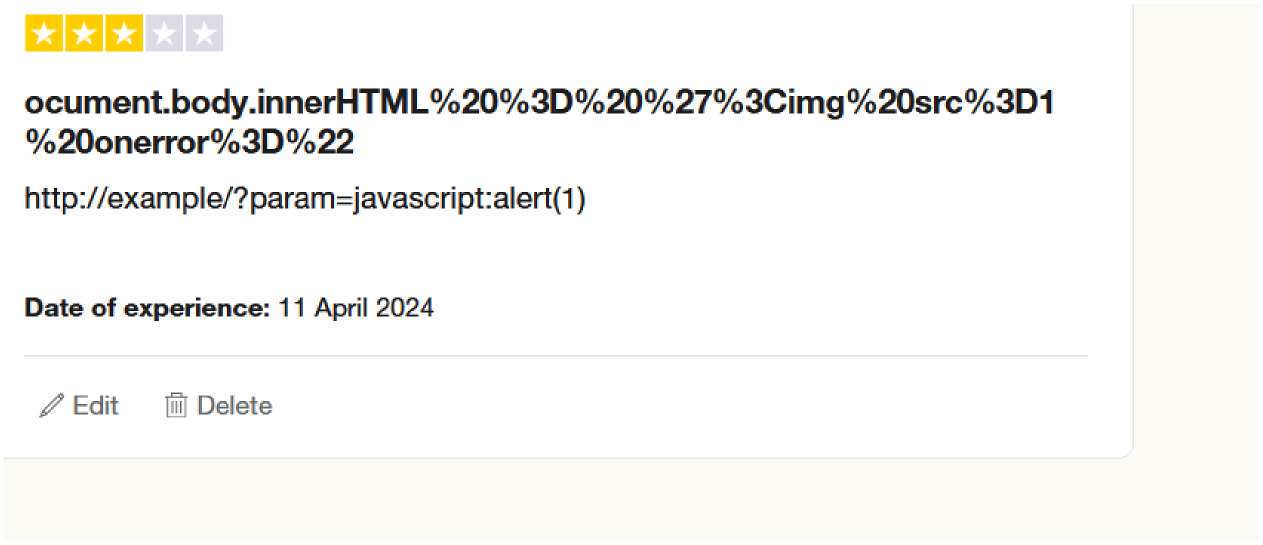
Your review is pending. Read more

Mel Brandison
0 reviews LK

http://foo?&apos;-alert(1)-&apos;

Date of experience: 11 April 2024

Edit Delete

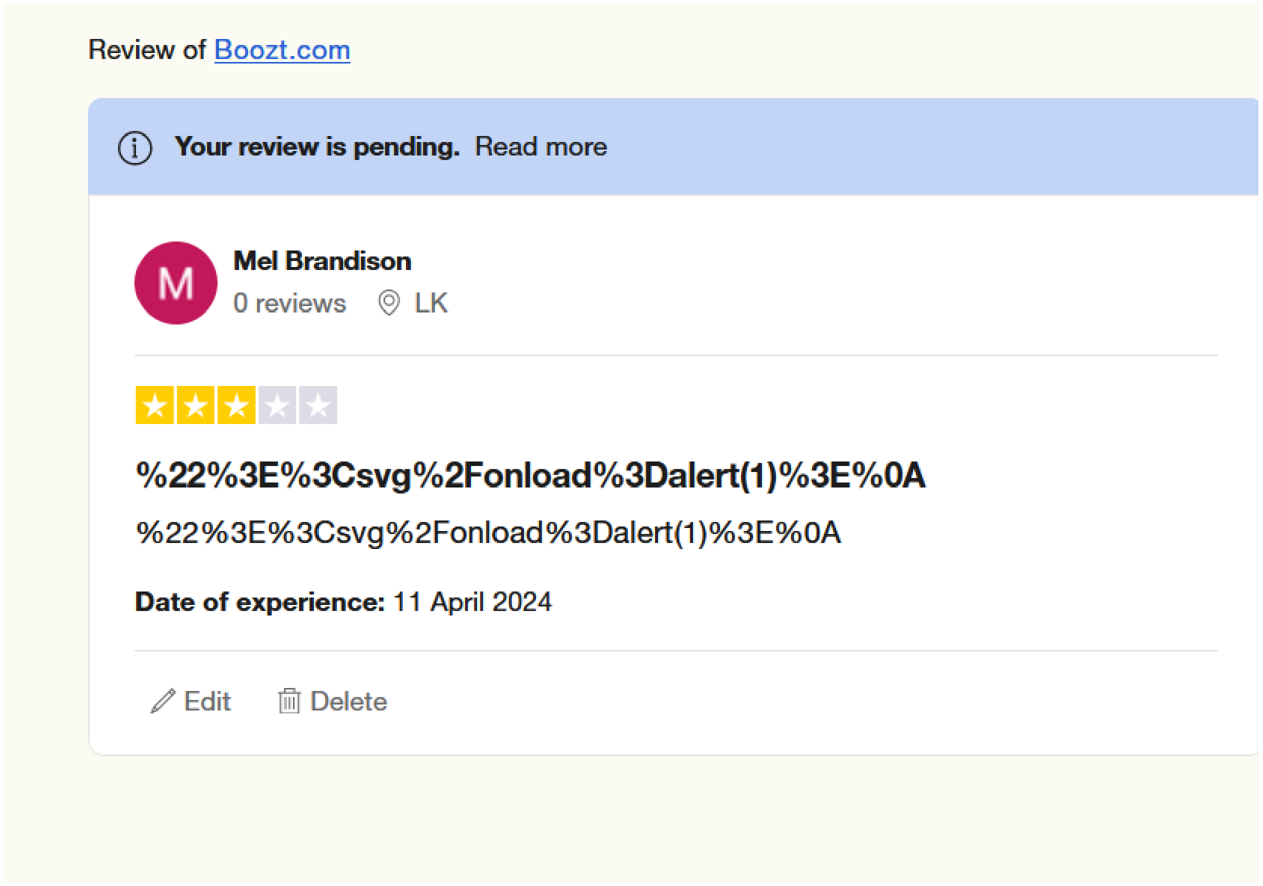


ocument.body.innerHTML%20%3D%20%27%3Cimg%20src%3D1%20onerror%3D%22

http://example/?param=javascript:alert(1)

Date of experience: 11 April 2024

Edit Delete



Review of Boozt.com

Your review is pending. Read more

Mel Brandison
0 reviews LK

%22%3E%3Csvg%2Fonload%3Dalert(1)%3E%0A

%22%3E%3Csvg%2Fonload%3Dalert(1)%3E%0A

Date of experience: 11 April 2024

Edit Delete

✓ **Takeaways**

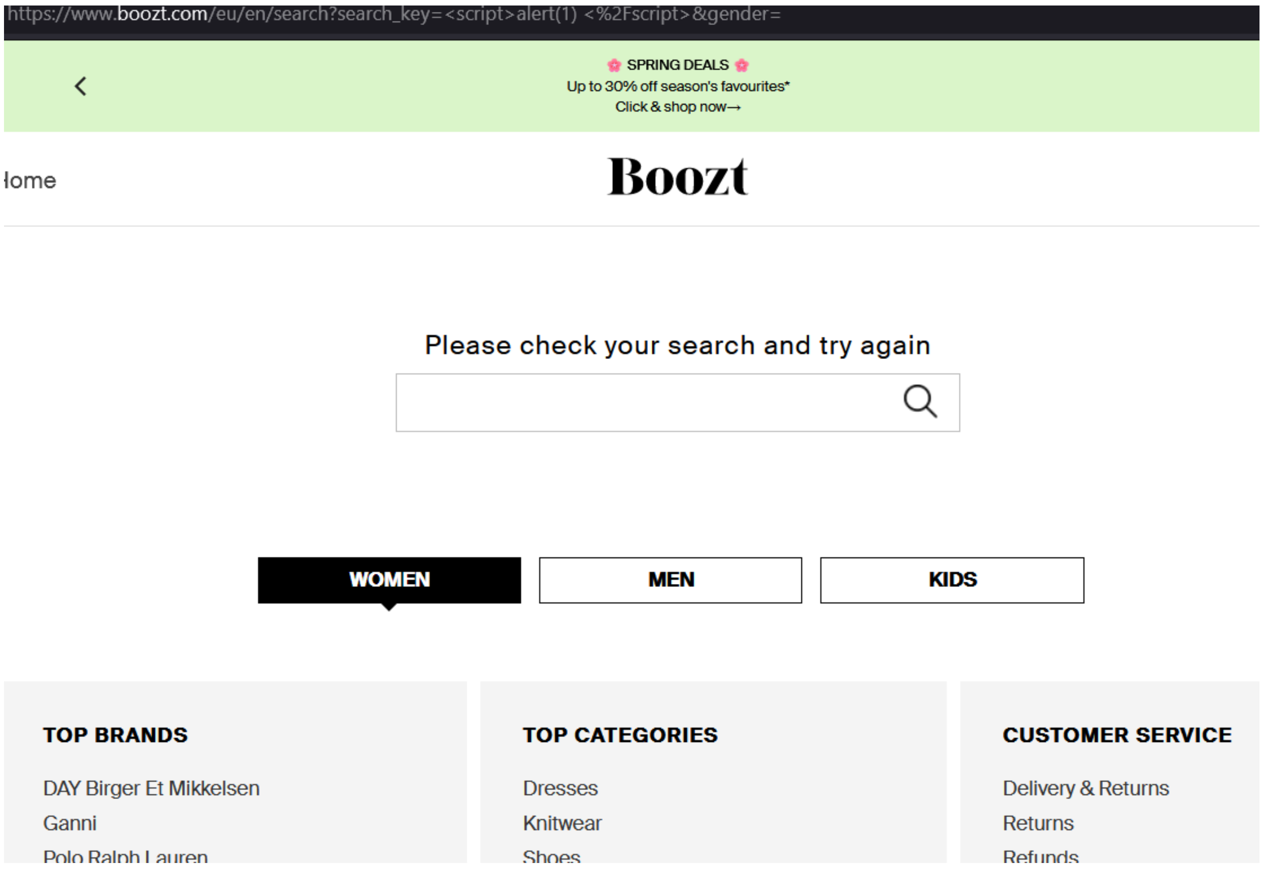
I searched why xss did not work here what I gathered

Why payload did not work?

□ **HTML Encoding**: The application might be encoding the input before displaying it as a link. This means that although you're injecting <script>alert(2)</script>, it's being HTML-encoded before being rendered in the browser. So, it might appear as <script>alert(2)</script> in the link, but it's treated as plain text and not executed as JavaScript.

□ **Input Sanitization**: The application may have input sanitization mechanisms in place that filter out or neutralize potentially dangerous input, including script tags and JavaScript code. This prevents XSS attacks by removing or escaping any HTML or JavaScript code entered into the form field.

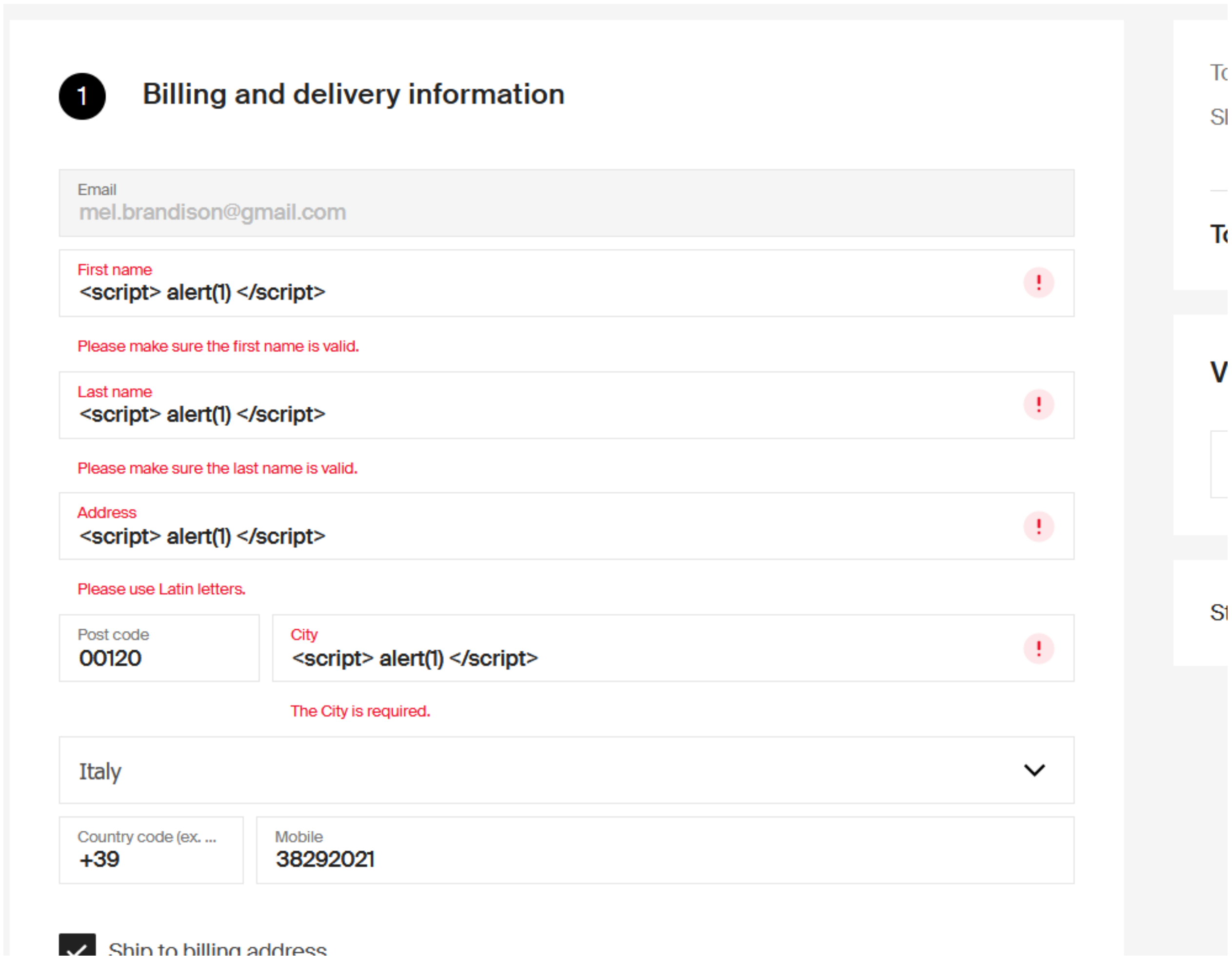
□ **Contextual Filtering**: Even if the input appears as a link, the context in which it's rendered might not allow execution of JavaScript. For example, if the input is rendered within an <a> tag's href attribute, the browser would interpret it as a URL, not as JavaScript code.



https://www.boozt.com/eu/en/search?search_key=<script>alert(1) <%2Fscript>&gender=

Please check your search and try again

Stored xss


Billing and delivery information

Email
mel.brandison@gmail.com

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

Please make sure the first name is valid.

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

Please make sure the last name is valid.

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

Please use Latin letters.

Post code
00120

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

The City is required.

Italy

Country code (ex. ...
+39

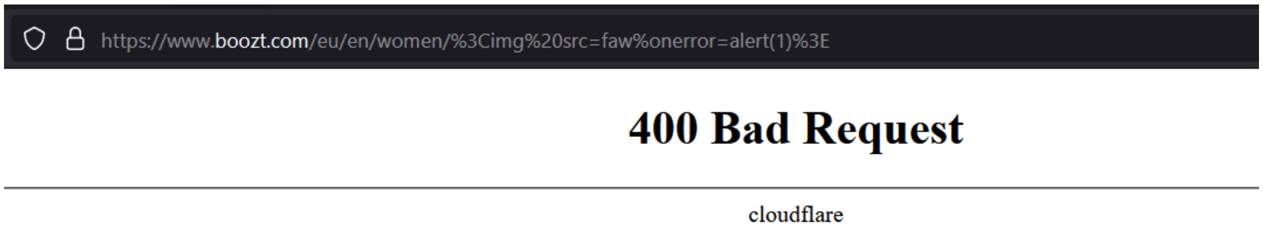
Mobile
38292021

Ship to billing address

- **<u>2024.4.16</u>**

Attempted some cross site scripting.

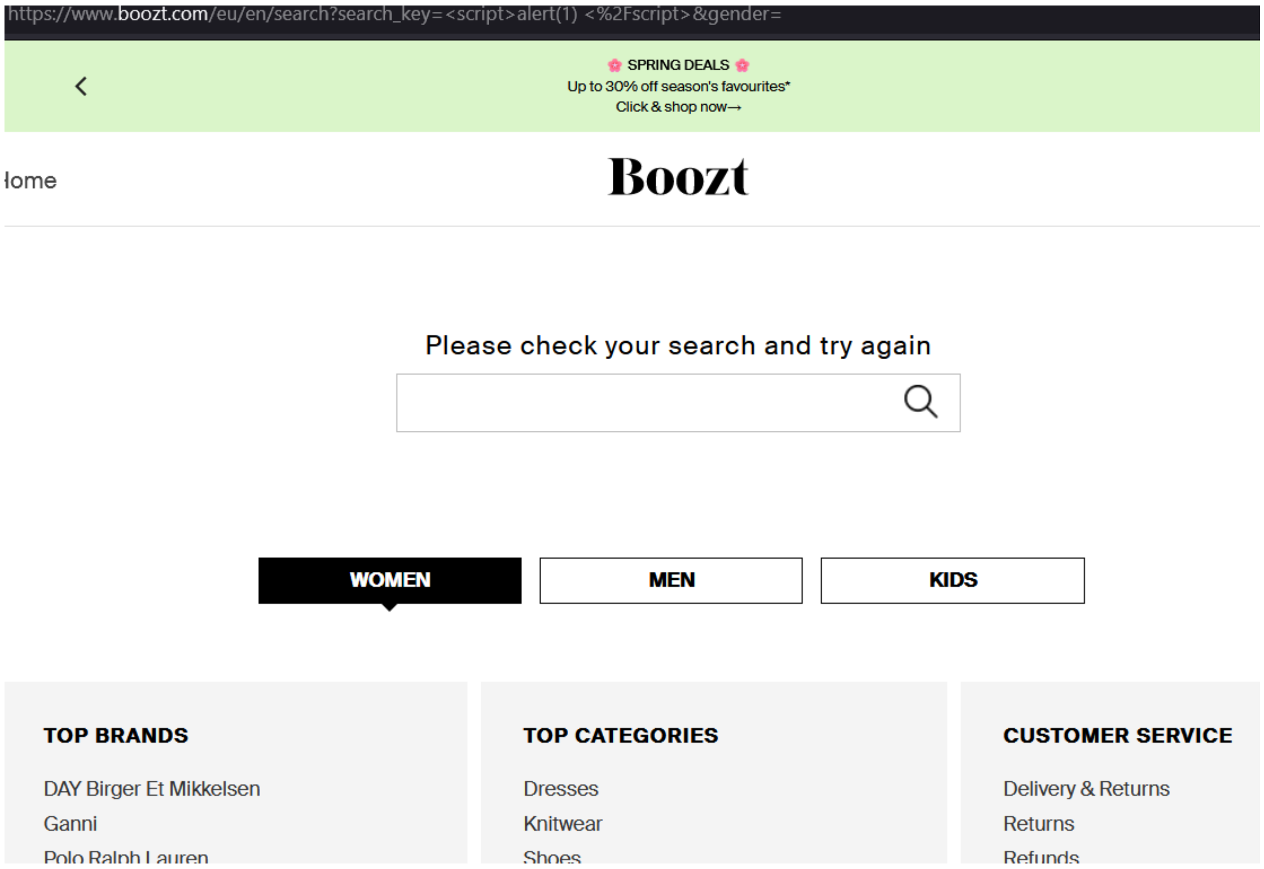
Today I did xss on the url. But for most xss payloads I got a 400-status code. Because this site use url encoding, input sanitation and validation.



https://www.boozt.com/eu/en/women/%3Cimg%20src=faw%onerror=alert(1)%3E

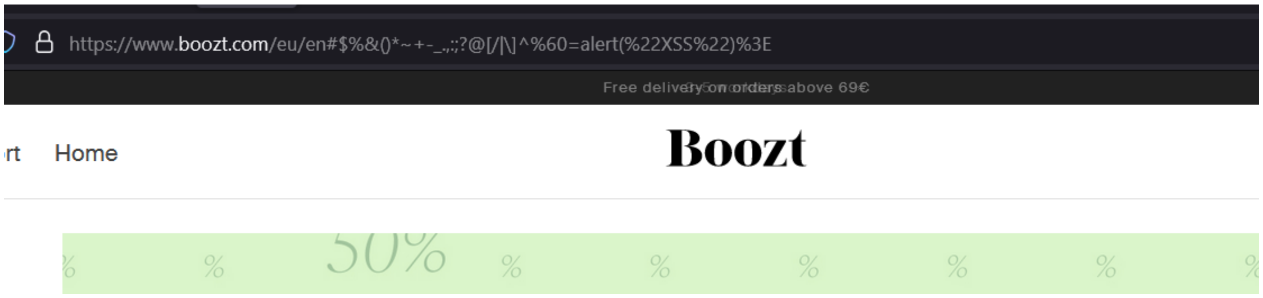
400 Bad Request

cloudflare



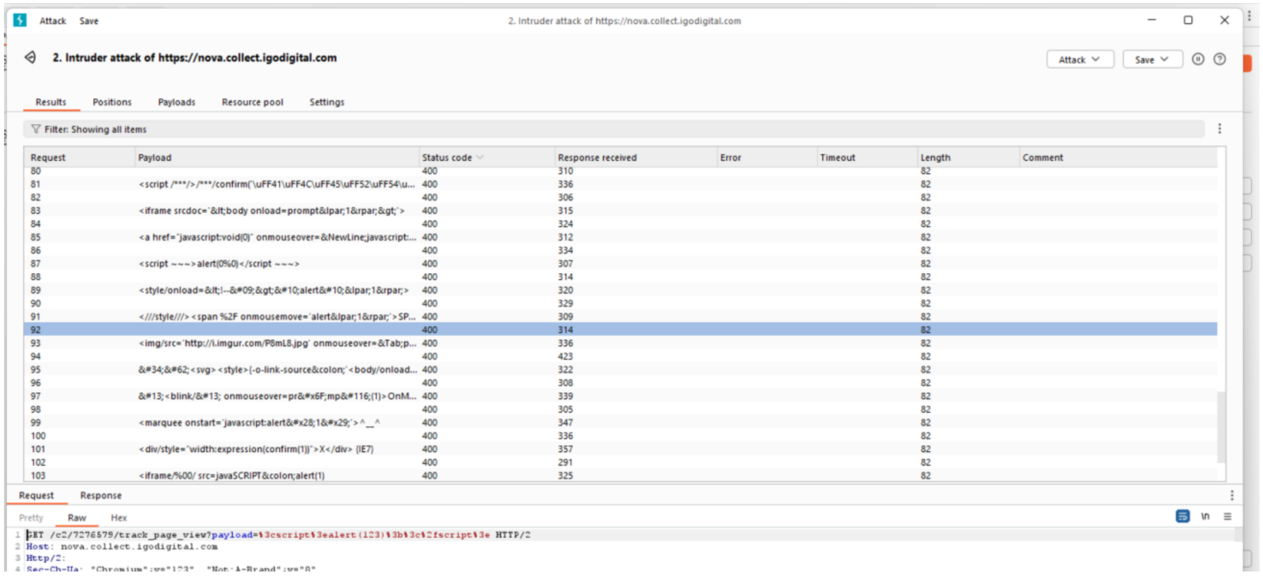
https://www.boozt.com/eu/en/search?search_key=<script>alert(1) <%2Fscript>&gender=

Please check your search and try again



https://www.boozt.com/eu/en#$%&()*~+-_.,:;?@[/|\]^%60=alert(%22XSS%22)%3E

I also use burp suit intruder to perform list of xss payloads on website but any of the attempts were not successful. I used xss payload list in https://github.com/payloadbox/xss-payload-list to attack the website using burp suite intruder.



Attack
Save
2. Intruder attack of https://nova.collect.igodigital.com
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items

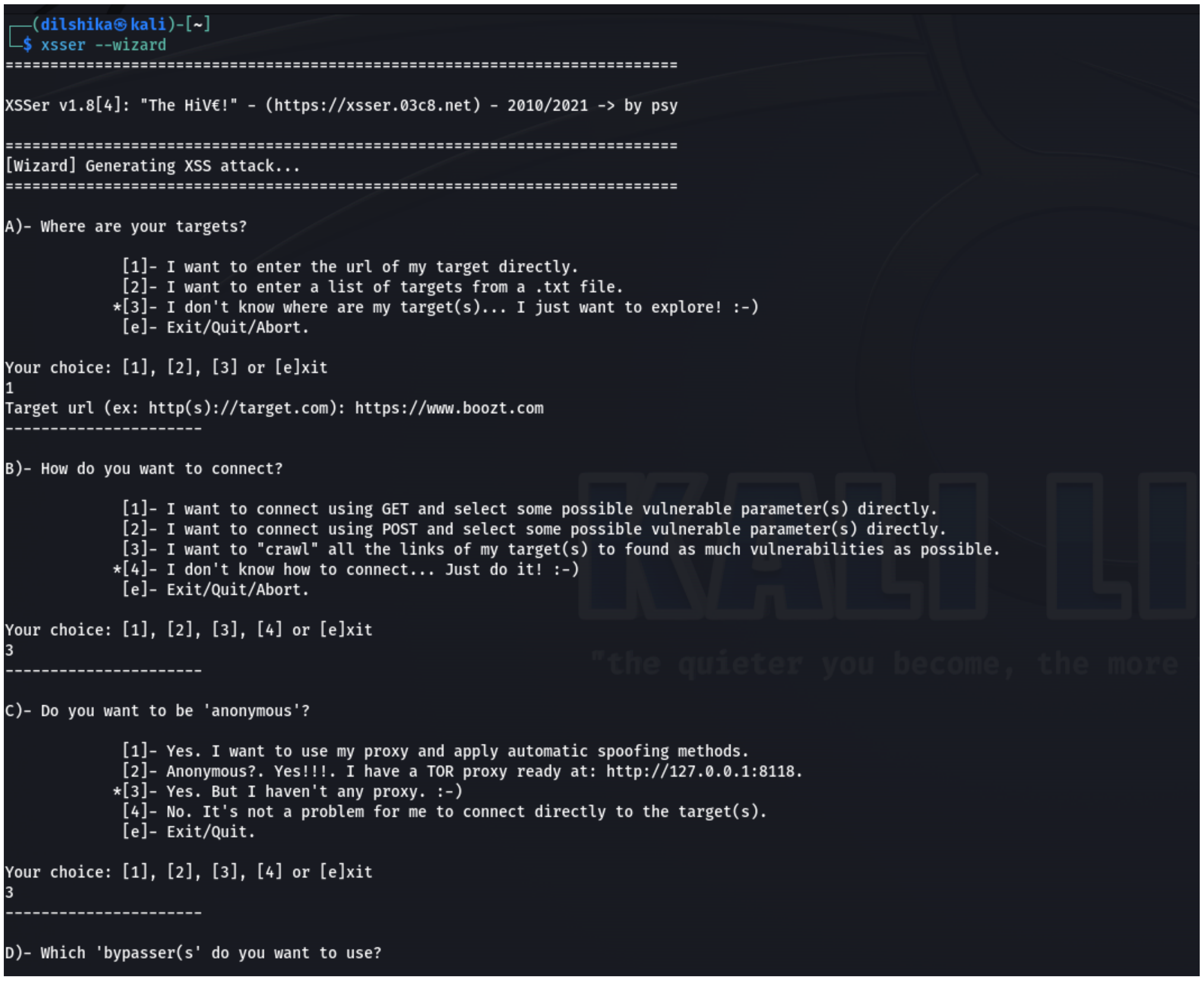
2. Intruder attack of https://nova.collect.igodigital.com

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
80		400	310			82	
81	<script /***/>/***/confirm('\uFF41\uFF4C\uFF45\uFF52\uFF54\u...	400	336			82	
82		400	306			82	
83	<iframe srcdoc='&lt;body onload=prompt&lpar;1&rpar;&gt;'>	400	315			82	
84		400	324			82	
85	<a href="javascript:void(0)" onmouseover=&NewLine;javascript:...	400	312			82	
86		400	334			82	
87	<script ~~~>alert(0%0)</script ~~~>	400	307			82	
88		400	314			82	
89	<style/onload=&lt;!--&#09;&gt;&#10;alert&#10;&lpar;1&rpar;>	400	320			82	
90		400	329			82	
91	<///style///><span %2F onmousemove='alert&lpar;1&rpar;'>SP...	400	309			82	
92		400	314			82	
93	<img/src='http://i.imgur.com/P8mL8.jpg' onmouseover=&Tab;p...	400	336			82	
94		400	423			82	
95	&#34;&#62;<svg><style>{-o-link-source&colon;'<body/onload...	400	322			82	
96		400	308			82	
97	&#13;<blink/&#13; onmouseover=pr&#x6F;mp&#116;(1)>OnM...	400	339			82	
98		400	305			82	
99	<marquee onstart='javascript:alert&#x28;1&#x29;'>^__^	400	347			82	
100		400	336			82	
101	<div/style="width:expression(confirm(1))">X</div> {IE7}	400	357			82	
102		400	291			82	
103	<iframe/%00/ src=javaSCRIPT&colon;alert(1)	400	325			82	

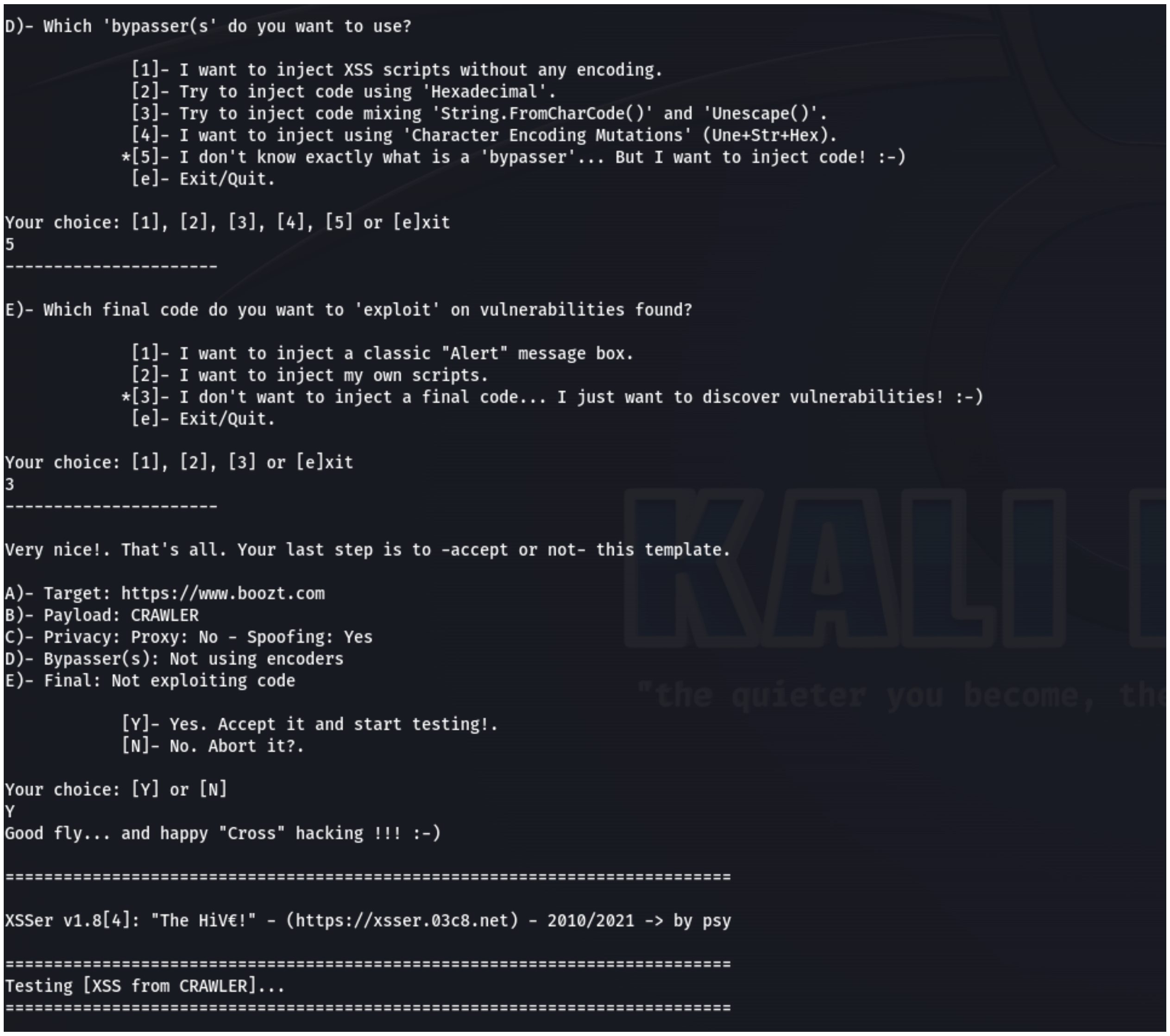
```
GET /c2/7276579/track_page_view?payload=%3cscript%3ealert(123)%3b%3c%2fscript%3e HTTP/2
Host: nova.collect.igodigital.com
Http/2:
Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
```

Using xsser

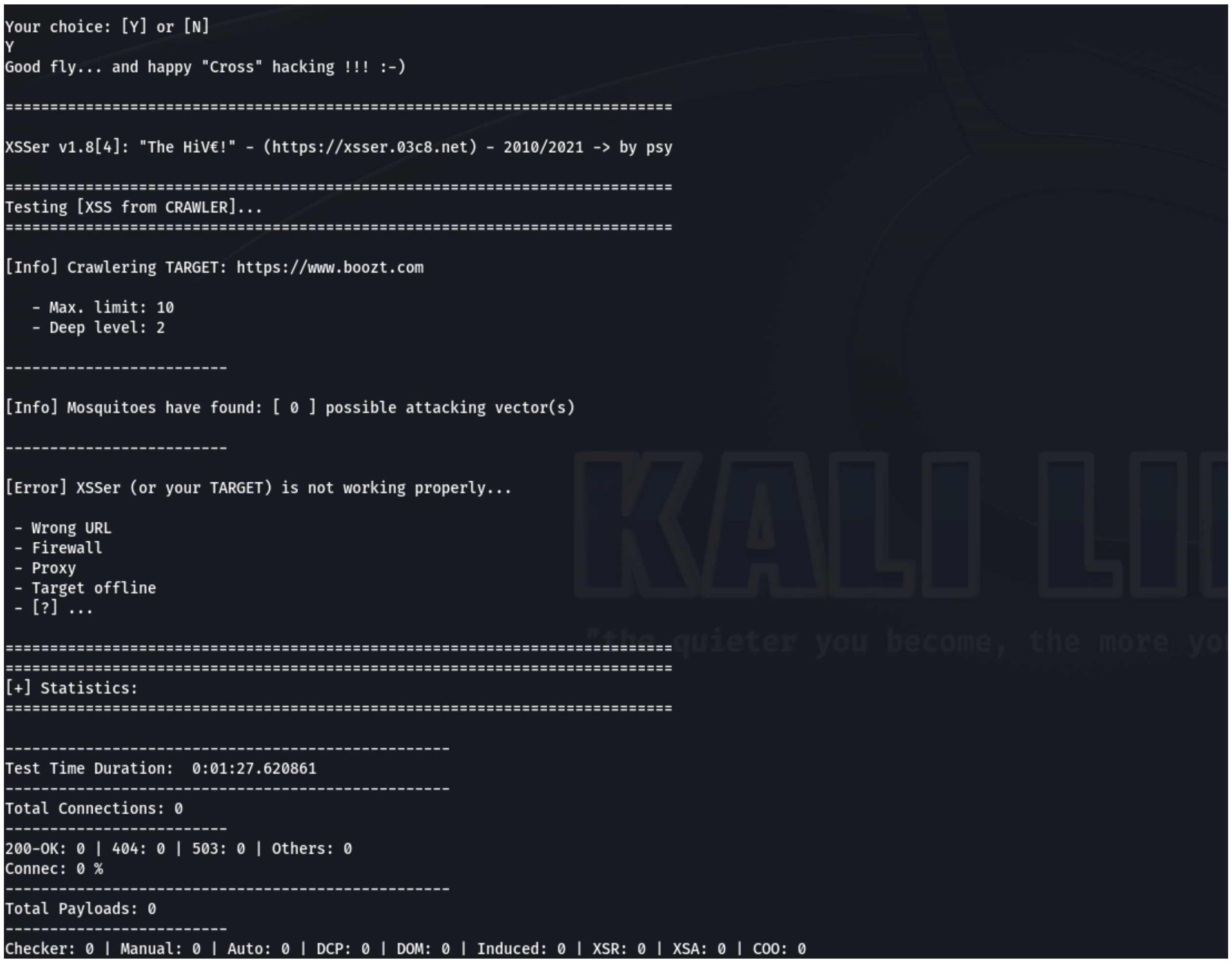
Used xsser tool in kali to perform xss on the boozt.com and booztlet.com. A web-based application's Cross Site "Scripter" (also known as XSSer) is an automatic framework for identifying, exploiting and reporting XSS vulnerabilities.



```
┌──(dilshika㉿kali)-[~]
└─$ xsser --wizard
===========================================================================

XSSer v1.8[4]: "The HiV€!" - (https://xsser.03c8.net) - 2010/2021 -> by psy

===========================================================================
[Wizard] Generating XSS attack...
===========================================================================

A)- Where are your targets?

             [1]- I want to enter the url of my target directly.
             [2]- I want to enter a list of targets from a .txt file.
            *[3]- I don't know where are my target(s)... I just want to explore! :-)
             [e]- Exit/Quit/Abort.

Your choice: [1], [2], [3] or [e]xit
1
Target url (ex: http(s)://target.com): https://www.boozt.com
----------------------

B)- How do you want to connect?

             [1]- I want to connect using GET and select some possible vulnerable parameter(s) directly.
             [2]- I want to connect using POST and select some possible vulnerable parameter(s) directly.
             [3]- I want to "crawl" all the links of my target(s) to found as much vulnerabilities as possible.
            *[4]- I don't know how to connect... Just do it! :-)
             [e]- Exit/Quit/Abort.

Your choice: [1], [2], [3], [4] or [e]xit
3
----------------------

C)- Do you want to be 'anonymous'?

             [1]- Yes. I want to use my proxy and apply automatic spoofing methods.
             [2]- Anonymous?. Yes!!!. I have a TOR proxy ready at: http://127.0.0.1:8118.
            *[3]- Yes. But I haven't any proxy. :-)
             [4]- No. It's not a problem for me to connect directly to the target(s).
             [e]- Exit/Quit.

Your choice: [1], [2], [3], [4] or [e]xit
3
----------------------

D)- Which 'bypasser(s' do you want to use?
```



```
D)- Which 'bypasser(s' do you want to use?

             [1]- I want to inject XSS scripts without any encoding.
             [2]- Try to inject code using 'Hexadecimal'.
             [3]- Try to inject code mixing 'String.FromCharCode()' and 'Unescape()'.
             [4]- I want to inject using 'Character Encoding Mutations' (Une+Str+Hex).
            *[5]- I don't know exactly what is a 'bypasser'... But I want to inject code! :-)
             [e]- Exit/Quit.

Your choice: [1], [2], [3], [4], [5] or [e]xit
5
----------------------

E)- Which final code do you want to 'exploit' on vulnerabilities found?

             [1]- I want to inject a classic "Alert" message box.
             [2]- I want to inject my own scripts.
            *[3]- I don't want to inject a final code... I just want to discover vulnerabilities! :-)
             [e]- Exit/Quit.

Your choice: [1], [2], [3] or [e]xit
3
----------------------

Very nice!. That's all. Your last step is to -accept or not- this template.

A)- Target: https://www.boozt.com
B)- Payload: CRAWLER
C)- Privacy: Proxy: No - Spoofing: Yes
D)- Bypasser(s): Not using encoders
E)- Final: Not exploiting code

            [Y]- Yes. Accept it and start testing!.
            [N]- No. Abort it?.

Your choice: [Y] or [N]
Y
Good fly... and happy "Cross" hacking !!! :-)

===========================================================================

XSSer v1.8[4]: "The HiV€!" - (https://xsser.03c8.net) - 2010/2021 -> by psy

===========================================================================
Testing [XSS from CRAWLER]...
===========================================================================
```



```
Your choice: [Y] or [N]
Y
Good fly... and happy "Cross" hacking !!! :-)

===========================================================================

XSSer v1.8[4]: "The HiV€!" - (https://xsser.03c8.net) - 2010/2021 -> by psy

===========================================================================
Testing [XSS from CRAWLER]...
===========================================================================

[Info] Crawlering TARGET: https://www.boozt.com

   - Max. limit: 10
   - Deep level: 2

-------------------------

[Info] Mosquitoes have found: [ 0 ] possible attacking vector(s)

-------------------------

[Error] XSSer (or your TARGET) is not working properly...

 - Wrong URL
 - Firewall
 - Proxy
 - Target offline
 - [?] ...

===========================================================================
===========================================================================
[+] Statistics:
===========================================================================

--------------------------------------------------
Test Time Duration:  0:01:27.620861
--------------------------------------------------
Total Connections: 0
-------------------------
200-OK: 0 | 404: 0 | 503: 0 | Others: 0
Connec: 0 %
--------------------------------------------------
Total Payloads: 0
-------------------------
Checker: 0 | Manual: 0 | Auto: 0 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA: 0 | COO: 0
```

Why xsser did not work?

In the output it is displayed that

[ERROR] XSSer(or your target) is not properly working

-Wrong url

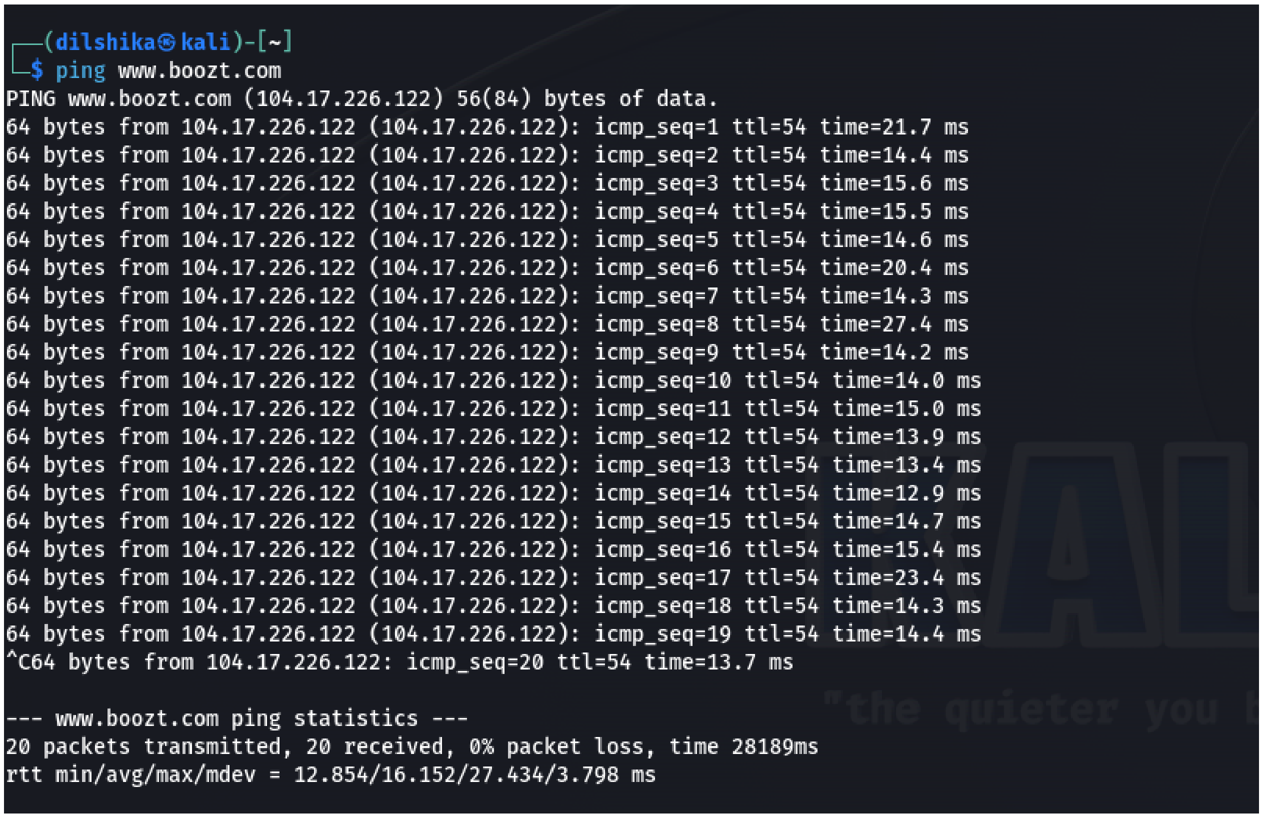
-Firewall

-proxy

-target offline

-[?]

From these options wrong url is not affecting because I double checked the url before entering it. Also, the target is not offline, I did a ping command to the target and as a result I received packets, so that mean our target is online.



```
┌──(dilshika㉿kali)-[~]
└─$ ping www.boozt.com
PING www.boozt.com (104.17.226.122) 56(84) bytes of data.
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=1 ttl=54 time=21.7 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=2 ttl=54 time=14.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=3 ttl=54 time=15.6 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=4 ttl=54 time=15.5 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=5 ttl=54 time=14.6 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=6 ttl=54 time=20.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=7 ttl=54 time=14.3 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=8 ttl=54 time=27.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=9 ttl=54 time=14.2 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=10 ttl=54 time=14.0 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=11 ttl=54 time=15.0 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=12 ttl=54 time=13.9 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=13 ttl=54 time=13.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=14 ttl=54 time=12.9 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=15 ttl=54 time=14.7 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=16 ttl=54 time=15.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=17 ttl=54 time=23.4 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=18 ttl=54 time=14.3 ms
64 bytes from 104.17.226.122 (104.17.226.122): icmp_seq=19 ttl=54 time=14.4 ms
^C64 bytes from 104.17.226.122: icmp_seq=20 ttl=54 time=13.7 ms

--- www.boozt.com ping statistics ---
20 packets transmitted, 20 received, 0% packet loss, time 28189ms
rtt min/avg/max/mdev = 12.854/16.152/27.434/3.798 ms
```

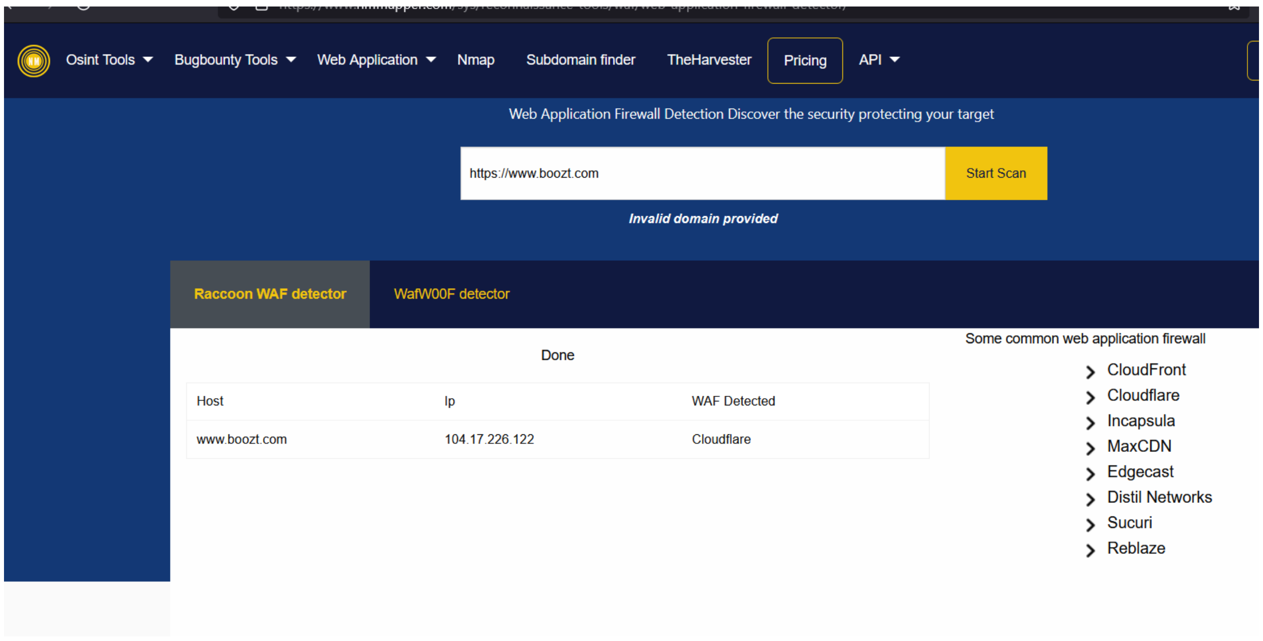
If the xsser did not work it is because of the following reasons,

1. **Proxy :**
   - A proxy server acts as an intermediary between a client and a destination server. It can intercept and inspect traffic passing through it, allowing it to filter out potentially malicious content, including XSS payloads.
   - Proxies can be configured to block requests containing suspicious patterns or known XSS payloads, preventing them from reaching the target application.
   - Some proxy servers offer features like content filtering and web application firewall (WAF) capabilities, which can detect and block XSS attacks in real-time.

2. **Firewall :**
   - Firewalls are network security devices that monitor and control incoming and outgoing network traffic based on predetermined security rules.
   - Web application firewalls (WAFs) specifically focus on filtering and monitoring HTTP traffic to and from web applications.

- WAFs can be configured with rulesets designed to detect and block common XSS attack patterns, such as script injections and suspicious JavaScript payloads.
- They can also enforce policies to block requests containing specific HTTP headers or parameters commonly associated with XSS attacks.

Used firewall



Host	Ip	WAF Detected
www.boozt.com	104.17.226.122	Cloudflare

- **<u>2024.04.17</u>**

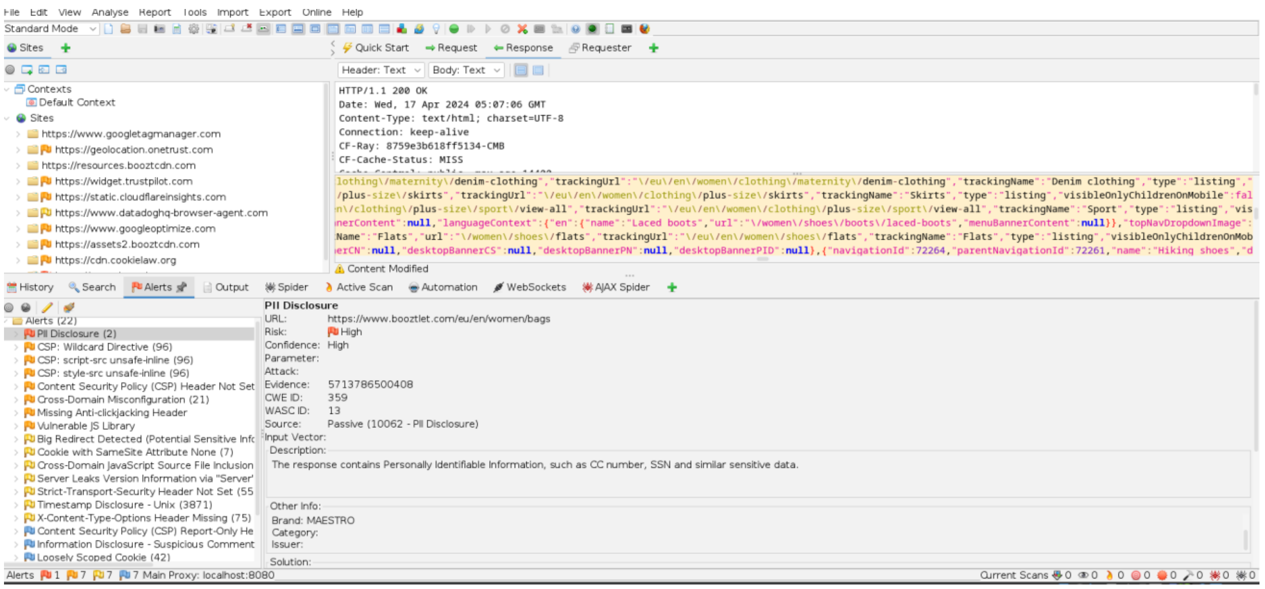
I had downloaded owasp zap, Today I used owasp zap.

I had used this for once but I did not know how to adjust options

First of all, I needed to know how to use this tool . for that I referred following vide. In that video it explains what is automated scan, manual scan and available options in owasp zap tool

https://www.youtube.com/watch?v=wLfRz7rRsH4

through this tool I learned how to adjust and modify default policy. I changed the strength of the attack. After that I scanned the site using automated scan. I found a high risked vulnerability in alert tab. This vulnerability is called PII (personally identifiable information) disclosure.



Contexts
Default Context
Sites
https://www.googletagmanager.com
https://geolocation.onetrust.com
https://resources.booztcdn.com
https://widget.trustpilot.com
https://static.cloudflareinsights.com
https://www.datadoghq-browser-agent.com
https://www.googleoptimize.com
https://assets2.booztcdn.com
https://cdn.cookielaw.org

```
HTTP/1.1 200 OK
Date: Wed, 17 Apr 2024 05:07:06 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
CF-Ray: 8759e3b618ff5134-CMB
CF-Cache-Status: MISS
```

```
lothing\/maternity\/denim-clothing","trackingUrl":"\/eu\/en\/women\/clothing\/maternity\/denim-clothing","trackingName":"Denim clothing","type":"listing","
/plus-size\/skirts","trackingUrl":"\/eu\/en\/women\/clothing\/plus-size\/skirts","trackingName":"Skirts","type":"listing","visibleOnlyChildrenOnMobile":fal
n\/clothing\/plus-size\/sport\/view-all","trackingUrl":"\/eu\/en\/women\/clothing\/plus-size\/sport\/view-all","trackingName":"Sport","type":"listing","vis
nerContent":null,"languageContext":{"en":{"name":"Laced boots","url":"\/women\/shoes\/boots\/laced-boots","menuBannerContent":null}},"topNavDropdownImage":
Name":"Flats","url":"\/women\/shoes\/flats","trackingUrl":"\/eu\/en\/women\/shoes\/flats","trackingName":"Flats","type":"listing","visibleOnlyChildrenOnMob
erCN":null,"desktopBannerCS":null,"desktopBannerPN":null,"desktopBannerPID":null},{"navigationId":72264,"parentNavigationId":72261,"name":"Hiking shoes","d
```

Content Modified

Alerts (22)
- PII Disclosure (2)
- CSP: Wildcard Directive (96)
- CSP: script-src unsafe-inline (96)
- CSP: style-src unsafe-inline (96)
- Content Security Policy (CSP) Header Not Set
- Cross-Domain Misconfiguration (21)
- Missing Anti-clickjacking Header
- Vulnerable JS Library
- Big Redirect Detected (Potential Sensitive Info
- Cookie with SameSite Attribute None (7)
- Cross-Domain JavaScript Source File Inclusion
- Server Leaks Version Information via "Server"
- Strict-Transport-Security Header Not Set (55
- Timestamp Disclosure - Unix (3871)
- X-Content-Type-Options Header Missing (75)
- Content Security Policy (CSP) Report-Only He
- Information Disclosure - Suspicious Comment
- Loosely Scoped Cookie (42)

PII Disclosure

URL:	https://www.booztlet.com/eu/en/women/bags
Risk:	High
Confidence:	High
Parameter:	
Attack:	
Evidence:	5713786500408
CWE ID:	359
WASC ID:	13
Source:	Passive (10062 - PII Disclosure)
Input Vector:	

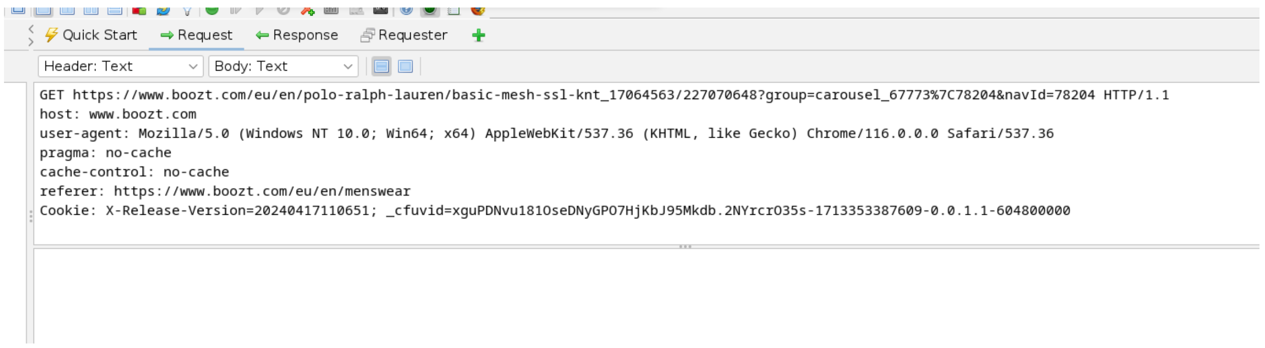
Description:
The response contains Personally Identifiable Information, such as CC number, SSN and similar sensitive data.

Other Info:
Brand: MAESTRO
Category:
Issuer:

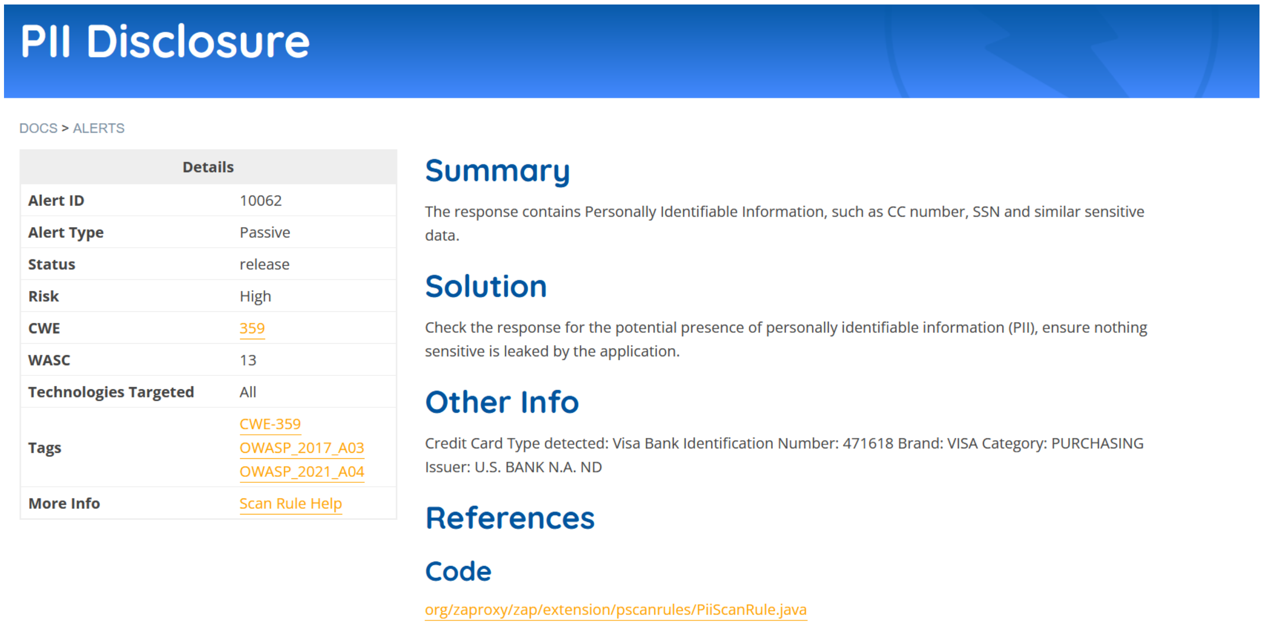
Solution:

Alerts 1 7 7 7 Main Proxy: localhost:8080

Current Scans 0 0 0 0 0 0 0 0



```
GET https://www.boozt.com/eu/en/polo-ralph-lauren/basic-mesh-ssl-knt_17064563/227070648?group=carousel_67773%7C78204&navId=78204 HTTP/1.1
host: www.boozt.com
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: https://www.boozt.com/eu/en/menswear
Cookie: X-Release-Version=20240417110651; _cfuvid=xguPDNvu181OseDNyGPO7HjKbJ95Mkdb.2NYrcrO35s-1713353387609-0.0.1.1-604800000
```



PII Disclosure

DOCS > ALERTS

Details	
Alert ID	10062
Alert Type	Passive
Status	release
Risk	High
CWE	359
WASC	13
Technologies Targeted	All
Tags	CWE-359 OWASP_2017_A03 OWASP_2021_A04
More Info	Scan Rule Help

Summary

The response contains Personally Identifiable Information, such as CC number, SSN and similar sensitive data.

Solution

Check the response for the potential presence of personally identifiable information (PII), ensure nothing sensitive is leaked by the application.

Other Info

Credit Card Type detected: Visa Bank Identification Number: 471618 Brand: VISA Category: PURCHASING Issuer: U.S. BANK N.A. ND

References

Code

org/zaproxy/zap/extension/pscanrules/PiiScanRule.java

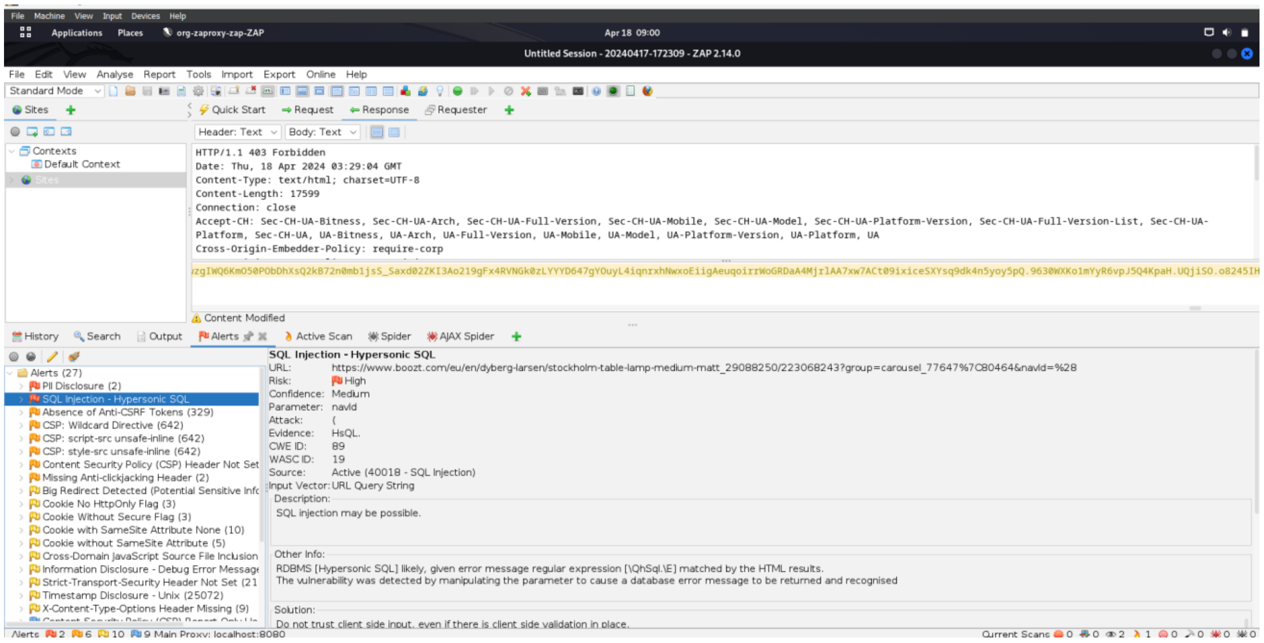
- ✓ **Challenges:**

I had to find about the PII vulnerability, it was a new concept for me.

- **2024.04.18**

Today I used owasp zap to automatically scan www.bootz.com, and I found 5 high-rate alerts. 3 of them are sql vulnerabilities and one of them is a vulnerability regarding hash disclosure.

Sql injection – hypersonic sql



File Machine View Input Devices Help
Applications Places org-zaproxy-zap-ZAP Apr 18 09:00
Untitled Session - 20240417-172309 - ZAP 2.14.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Header: Text Body: Text
Contexts
Default Context
Sites
Content Modified
History Search Output Alerts Active Scan Spider AJAX Spider

```
HTTP/1.1 403 Forbidden
Date: Thu, 18 Apr 2024 03:29:04 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 17599
Connection: close
Accept-CH: Sec-CH-UA-Bitness, Sec-CH-UA-Arch, Sec-CH-UA-Full-Version, Sec-CH-UA-Mobile, Sec-CH-UA-Model, Sec-CH-UA-Platform-Version, Sec-CH-UA-Full-Version-List, Sec-CH-UA-
Platform, Sec-CH-UA, UA-Bitness, UA-Arch, UA-Full-Version, UA-Mobile, UA-Model, UA-Platform-Version, UA-Platform, UA
Cross-Origin-Embedder-Policy: require-corp
```

```
zgIWQ6KmO50PObDhXsQ2kB72n0mb1jsS_Saxd02ZKI3Ao219gFx4RVNGk0zLYYYD647gYOuyL4iqnrxhNwxoEiigAeuqoirrWoGRDaA4MjrlAA7xw7ACt09ixiceSXYsq9dk4n5yoy5pQ.9630WXKo1mYyR6vpJ5Q4KpaH.UQjiSO.o8245IH
```

Alerts (27)
- PII Disclosure (2)
- SQL Injection - Hypersonic SQL
- Absence of Anti-CSRF Tokens (329)
- CSP: Wildcard Directive (642)
- CSP: script-src unsafe-inline (642)
- CSP: style-src unsafe-inline (642)
- Content Security Policy (CSP) Header Not Set
- Missing Anti-clickjacking Header (2)
- Big Redirect Detected (Potential Sensitive Infc
- Cookie No HttpOnly Flag (3)
- Cookie Without Secure Flag (3)
- Cookie with SameSite Attribute None (10)
- Cookie without SameSite Attribute (5)
- Cross-Domain JavaScript Source File Inclusion
- Information Disclosure - Debug Error Message
- Strict-Transport-Security Header Not Set (21
- Timestamp Disclosure - Unix (25072)
- X-Content-Type-Options Header Missing (9)

SQL Injection - Hypersonic SQL

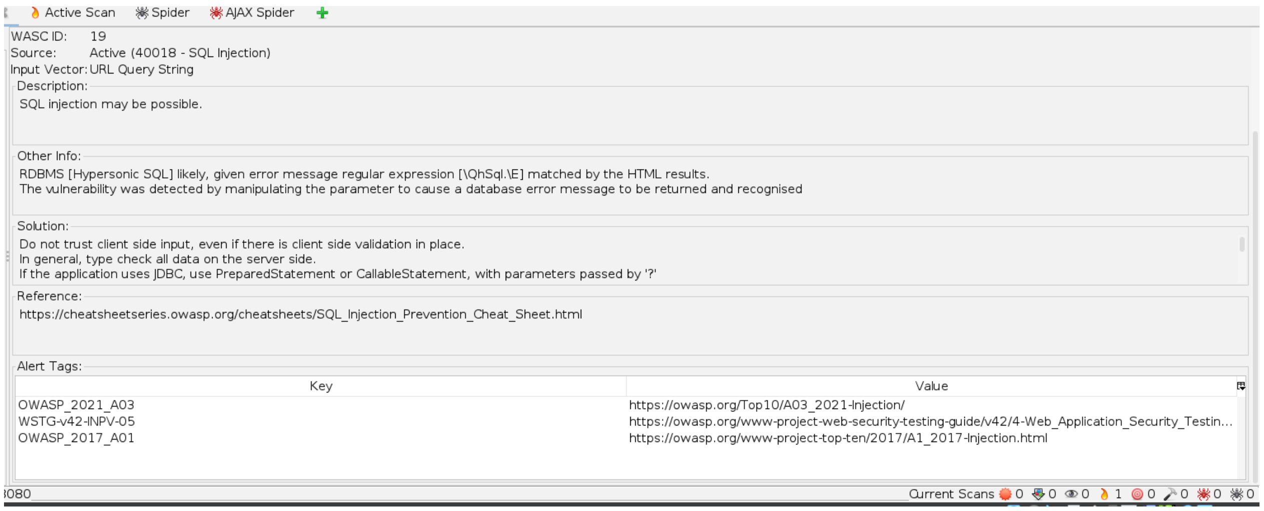
URL:	https://www.boozt.com/eu/en/dyberg-larsen/stockholm-table-lamp-medium-matt_29088250/223068243?group=carousel_77647%7C80464&navId=%28
Risk:	High
Confidence:	Medium
Parameter:	navId
Attack:	(
Evidence:	HsQL.
CWE ID:	89
WASC ID:	19
Source:	Active (40018 - SQL Injection)
Input Vector:	URL Query String

Description:
SQL injection may be possible.

Other Info:
RDBMS [Hypersonic SQL] likely, given error message regular expression [\QhSql.\E] matched by the HTML results.
The vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognised

Solution:
Do not trust client side input, even if there is client side validation in place.

Alerts 2 6 10 9 Main Proxy: localhost:8080 Current Scans 0 0 2 1 0 0 0 0



WASC ID:	19
Source:	Active (40018 - SQL Injection)
Input Vector:	URL Query String

Description:
SQL injection may be possible.

Other Info:
RDBMS [Hypersonic SQL] likely, given error message regular expression [\QhSql.\E] matched by the HTML results.
The vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognised

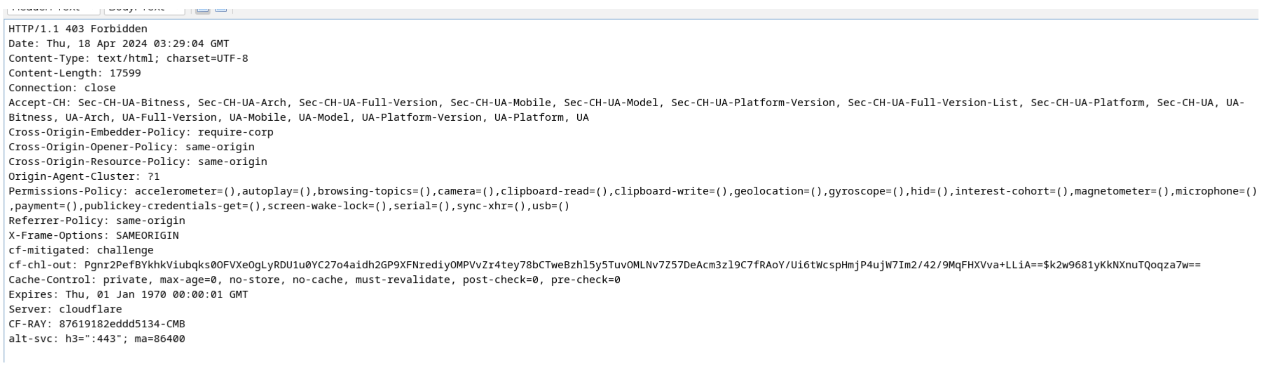
Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alert Tags:

Key	Value
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
WSTG-v42-INPV-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testin...
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html

8080 Current Scans 0 0 0 1 0 0 0 0



```
HTTP/1.1 403 Forbidden
Date: Thu, 18 Apr 2024 03:29:04 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 17599
Connection: close
Accept-CH: Sec-CH-UA-Bitness, Sec-CH-UA-Arch, Sec-CH-UA-Full-Version, Sec-CH-UA-Mobile, Sec-CH-UA-Model, Sec-CH-UA-Platform-Version, Sec-CH-UA-Full-Version-List, Sec-CH-UA-Platform, Sec-CH-UA, UA-
Bitness, UA-Arch, UA-Full-Version, UA-Mobile, UA-Model, UA-Platform-Version, UA-Platform, UA
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: same-origin
Origin-Agent-Cluster: ?1
Permissions-Policy: accelerometer=(),autoplay=(),browsing-topics=(),camera=(),clipboard-read=(),clipboard-write=(),geolocation=(),gyroscope=(),hid=(),interest-cohort=(),magnetometer=(),microphone=()
,payment=(),publickey-credentials-get=(),screen-wake-lock=(),serial=(),sync-xhr=(),usb=()
Referrer-Policy: same-origin
X-Frame-Options: SAMEORIGIN
cf-mitigated: challenge
cf-chl-out: Pgnr2PefBYkhkViubqks0OFVXeOgLyRDU1u0YC27o4aidh2GP9XFNrediyOMPVvZr4tey78bCTweBzhl5y5TuvOMLNv7Z57DeAcm3zl9C7fRAoY/Ui6tWcspHmjP4ujW7Im2/42/9MqFHXVva+LLiA==$k2w9681yKkNXnuTQoqza7w==
Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Expires: Thu, 01 Jan 1970 00:00:01 GMT
Server: cloudflare
CF-RAY: 87619182eddd5134-CMB
alt-svc: h3=":443"; ma=86400
```

Details of vulnerability

RDBMS [Hypersonic SQL] likely, given error message regular expression [\QhSql.\E] matched by the HTML results.

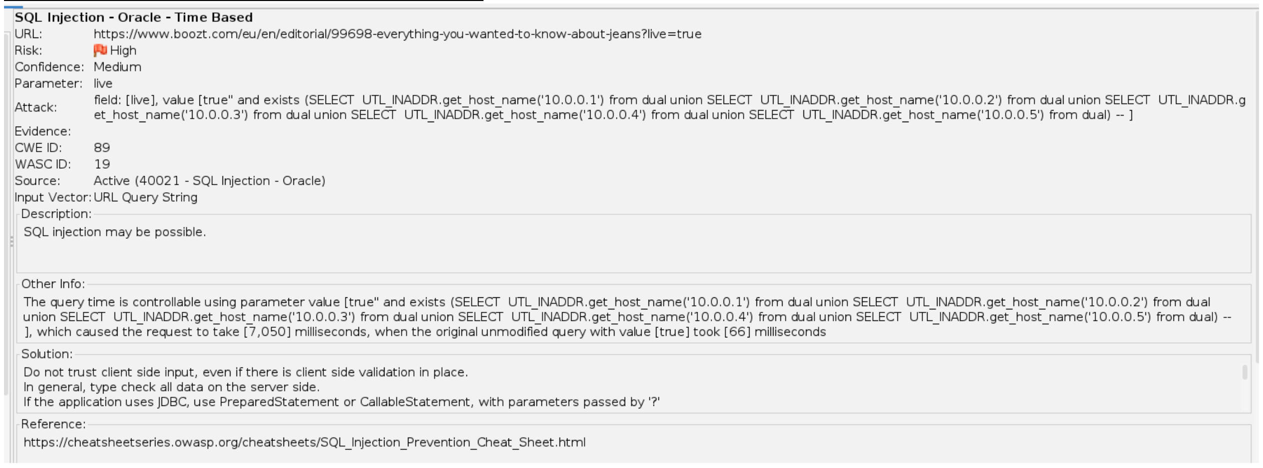
The vulnerability was detected by manipulating the parameter to cause a database error message to be returned and recognized

Solution by owasp zap

- Do not trust client-side input, even if there is client-side validation in place.
- In general, type check all data on the server side.
- If the application uses JDBC, use Prepared Statement or Callable Statement, with parameters passed by '?'
- If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.
- If database Stored Procedures can be used, use them.
- Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!
- Do not create dynamic SQL queries using simple string concatenation.
- Escape all data received from the client.
- Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input.
- Apply the principle of least privilege by using the least privileged database user possible.
- In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.
- Grant the minimum database access that is necessary for the application.

Ref - https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

SQL Injection -Oracle- Time based



SQL Injection - Oracle - Time Based

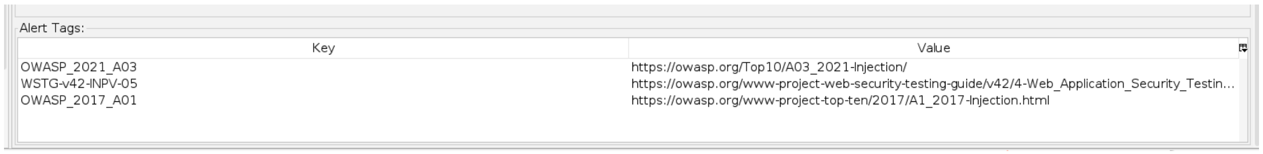
URL:	https://www.boozt.com/eu/en/editorial/99698-everything-you-wanted-to-know-about-jeans?live=true
Risk:	High
Confidence:	Medium
Parameter:	live
Attack:	field: [live], value [true" and exists (SELECT UTL_INADDR.get_host_name('10.0.0.1') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.2') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.3') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.4') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.5') from dual) --]
Evidence:	
CWE ID:	89
WASC ID:	19
Source:	Active (40021 - SQL Injection - Oracle)
Input Vector:	URL Query String

Description:
SQL injection may be possible.

Other Info:
The query time is controllable using parameter value [true" and exists (SELECT UTL_INADDR.get_host_name('10.0.0.1') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.2') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.3') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.4') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.5') from dual) --], which caused the request to take [7,050] milliseconds, when the original unmodified query with value [true] took [66] milliseconds

Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html



Alert Tags:

Key	Value
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
WSTG-v42-INPV-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testin...
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html

Details about vulnerability

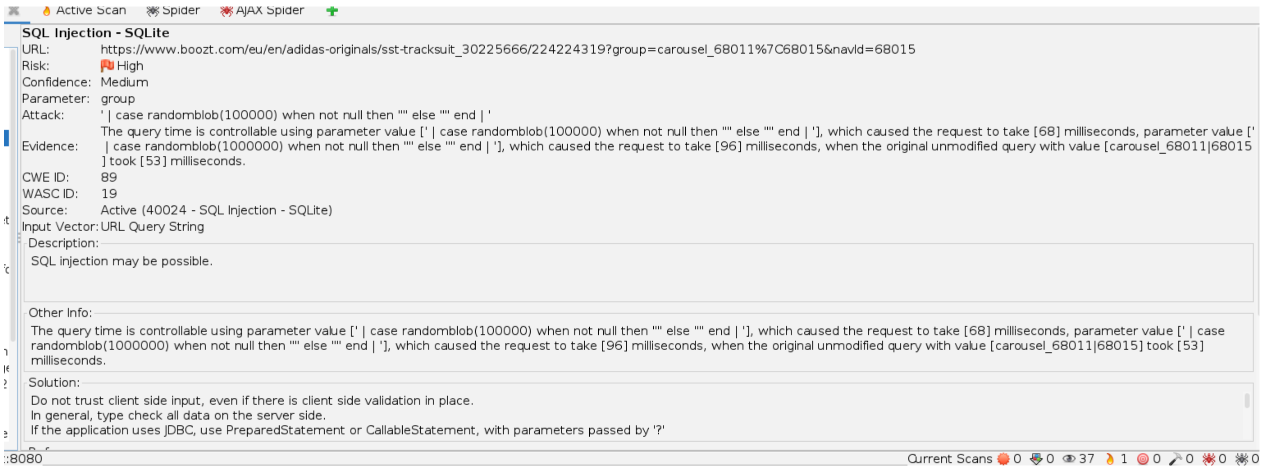
The query time is controllable using parameter value [true" and exists (SELECT UTL_INADDR.get_host_name('10.0.0.1') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.2') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.3') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.4') from dual union SELECT UTL_INADDR.get_host_name('10.0.0.5') from dual) --], which caused the request to take [7,050] milliseconds, when the original unmodified query with value [true] took [66] milliseconds

Solution by owasp zap

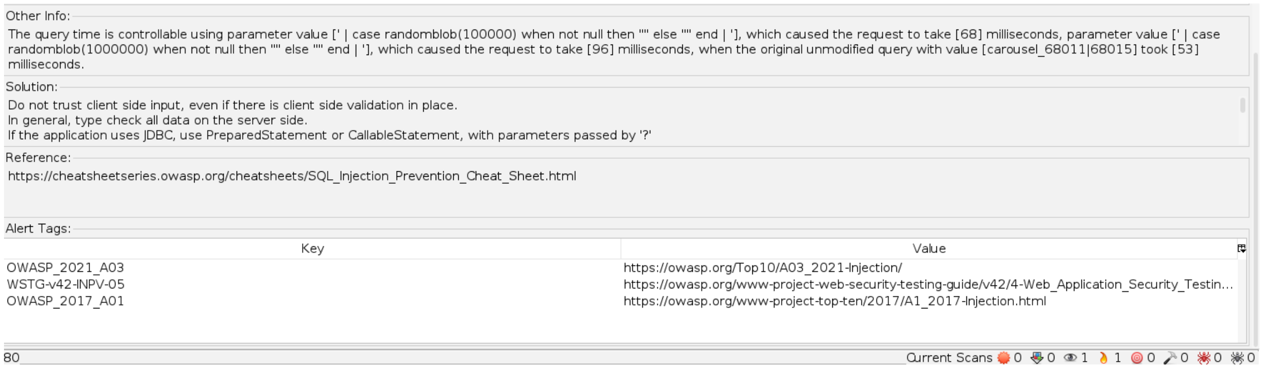
- Do not trust client-side input, even if there is client side validation in place.
- In general, type check all data on the server side.
- If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'
- If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.
- If database Stored Procedures can be used, use them.

- Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!
- Do not create dynamic SQL queries using simple string concatenation.
- Escape all data received from the client.
- Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input.
- Apply the principle of least privilege by using the least privileged database user possible.
- In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.
- Grant the minimum database access that is necessary for the application.

SQL -Injection=Sqlite



```
SQL Injection - SQLite
URL:          https://www.boozt.com/eu/en/adidas-originals/sst-tracksuit_30225666/224224319?group=carousel_68011%7C68015&navId=68015
Risk:         High
Confidence:   Medium
Parameter:    group
Attack:       ' | case randomblob(100000) when not null then "" else "" end | '
Evidence:     The query time is controllable using parameter value [' | case randomblob(100000) when not null then "" else "" end | '], which caused the request to take [68] milliseconds, parameter value [' | case randomblob(1000000) when not null then "" else "" end | '], which caused the request to take [96] milliseconds, when the original unmodified query with value [carousel_68011|68015 ] took [53] milliseconds.
CWE ID:       89
WASC ID:      19
Source:       Active (40024 - SQL Injection - SQLite)
Input Vector: URL Query String
Description:
SQL injection may be possible.
Other Info:
The query time is controllable using parameter value [' | case randomblob(100000) when not null then "" else "" end | '], which caused the request to take [68] milliseconds, parameter value [' | case randomblob(1000000) when not null then "" else "" end | '], which caused the request to take [96] milliseconds, when the original unmodified query with value [carousel_68011|68015] took [53] milliseconds.
Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'
:8080                                                Current Scans 0 0 37 1 0 0 0 0
```



```
Other Info:
The query time is controllable using parameter value [' | case randomblob(100000) when not null then "" else "" end | '], which caused the request to take [68] milliseconds, parameter value [' | case randomblob(1000000) when not null then "" else "" end | '], which caused the request to take [96] milliseconds, when the original unmodified query with value [carousel_68011|68015] took [53] milliseconds.
Solution:
Do not trust client side input, even if there is client side validation in place.
In general, type check all data on the server side.
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'
Reference:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
Alert Tags:
```

Key	Value
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
WSTG-v42-INPV-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testin...
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html

```
80                                                   Current Scans 0 0 1 1 0 0 0 0
```

Details about vulnerability

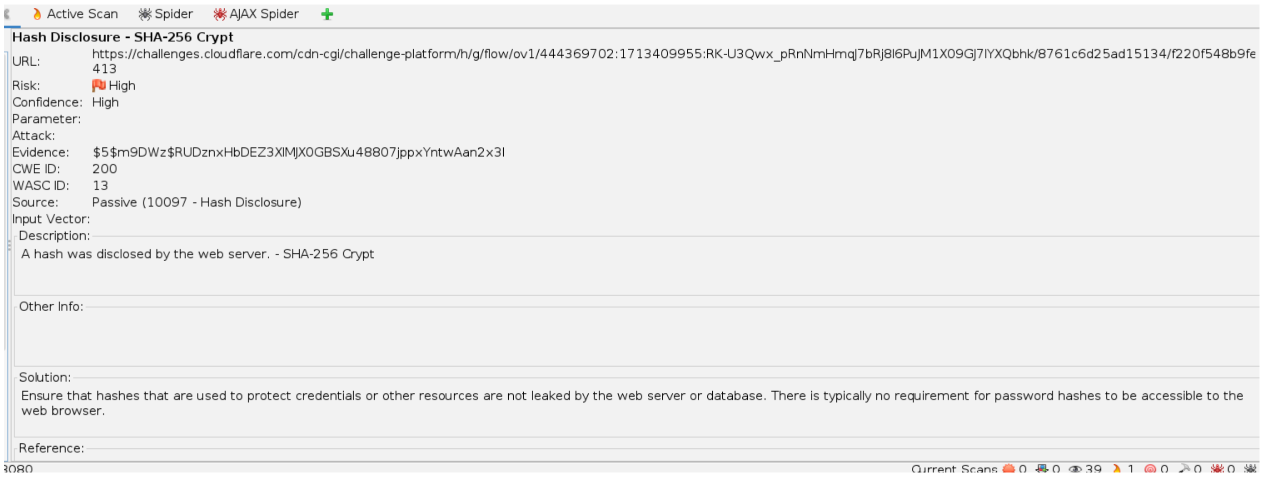
The query time is controllable using parameter value [' | case randomblob(100000) when not null then "" else "" end | '], which caused the request to take [68] milliseconds, parameter value [' | case randomblob(1000000) when not null then "" else "" end | '], which caused the request to take [96] milliseconds, when the original unmodified query with value [carousel_68011|68015] took [53] milliseconds.

Solution by owasp zap

- Do not trust client side input, even if there is client side validation in place.
- In general, type check all data on the server side.
- If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'
- If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.
- If database Stored Procedures can be used, use them.
- Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!
- Do not create dynamic SQL queries using simple string concatenation.
- Escape all data received from the client.
- Apply an 'allow list' of allowed characters, or a 'deny list' of disallowed characters in user input.
- Apply the principle of least privilege by using the least privileged database user possible.
- In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.
- Grant the minimum database access that is necessary for the application.

Ref -
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Hash disclosure



Active Scan Spider AJAX Spider

Hash Disclosure - SHA-256 Crypt

URL:	https://challenges.cloudflare.com/cdn-cgi/challenge-platform/h/g/flow/ov1/444369702:1713409955:RK-U3Qwx_pRnNmHmqJ7bRj8l6PuJM1X09GJ7lYXQbhk/8761c6d25ad15134/f220f548b9fe 413
Risk:	High
Confidence:	High
Parameter:	
Attack:	
Evidence:	$5$m9DWz$RUDznxHbDEZ3XlMJX0GBSXu48807jppxYntwAan2x3I
CWE ID:	200
WASC ID:	13
Source:	Passive (10097 - Hash Disclosure)
Input Vector:	

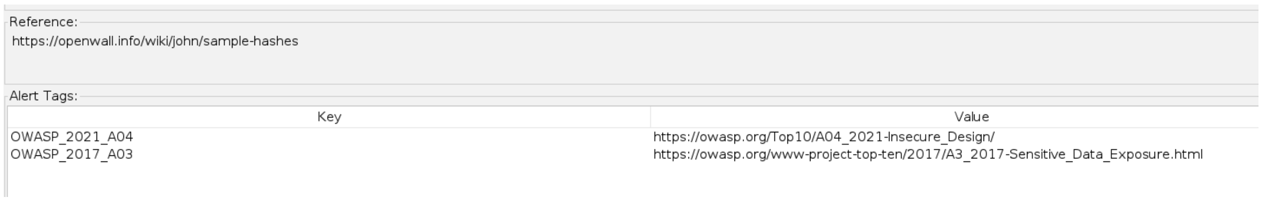
Description:
A hash was disclosed by the web server. - SHA-256 Crypt

Other Info:

Solution:
Ensure that hashes that are used to protect credentials or other resources are not leaked by the web server or database. There is typically no requirement for password hashes to be accessible to the web browser.

Reference:

8080 Current Scans 0 0 39 1 0 0 0



Reference:
https://openwall.info/wiki/john/sample-hashes

Alert Tags:

Key	Value
OWASP_2021_A04	https://owasp.org/Top10/A04_2021-Insecure_Design/
OWASP_2017_A03	https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

- ✓ **Challenges**

The automated scan of owasp zap takes a long time to complete the scan.

I had to assign some time to study about new things I learn. Such as hypersonic sql, sql lite, oracle databases

- **<u>2024.04.19</u>**

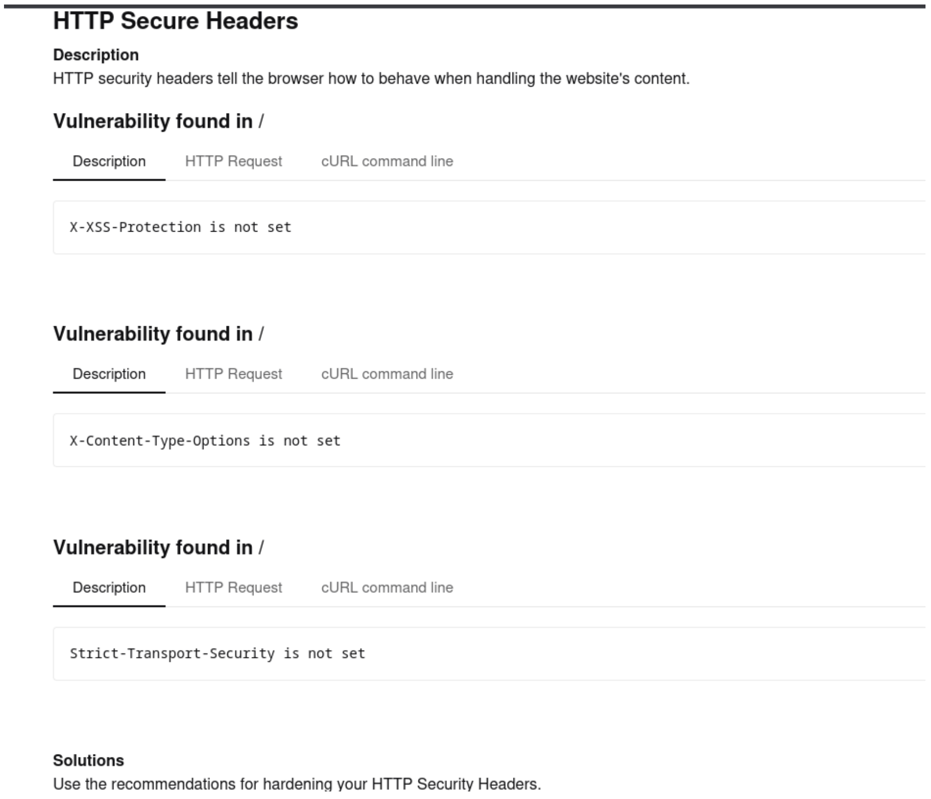
I started making reports for vulnerabilities I found. I made reports for pii disclosure, xss attempts.

Today I used a new tool called wapiti, it comes in kali as a web vulnerability scanner. It scans for vulnerabilities like sqli, xss, csp vulnerabilities and more.

It generates a report after the scan , we can view it in browser. Here Is the report for https://www.boozt.com

Wapiti display the vulnerabilities found also give recommend mitigations. The scan gave alerts for security header issues and csp issues.

Wapiti scan for www.boozt.com



HTTP Secure Headers

Description
HTTP security headers tell the browser how to behave when handling the website's content.

Vulnerability found in /

```
X-XSS-Protection is not set
```

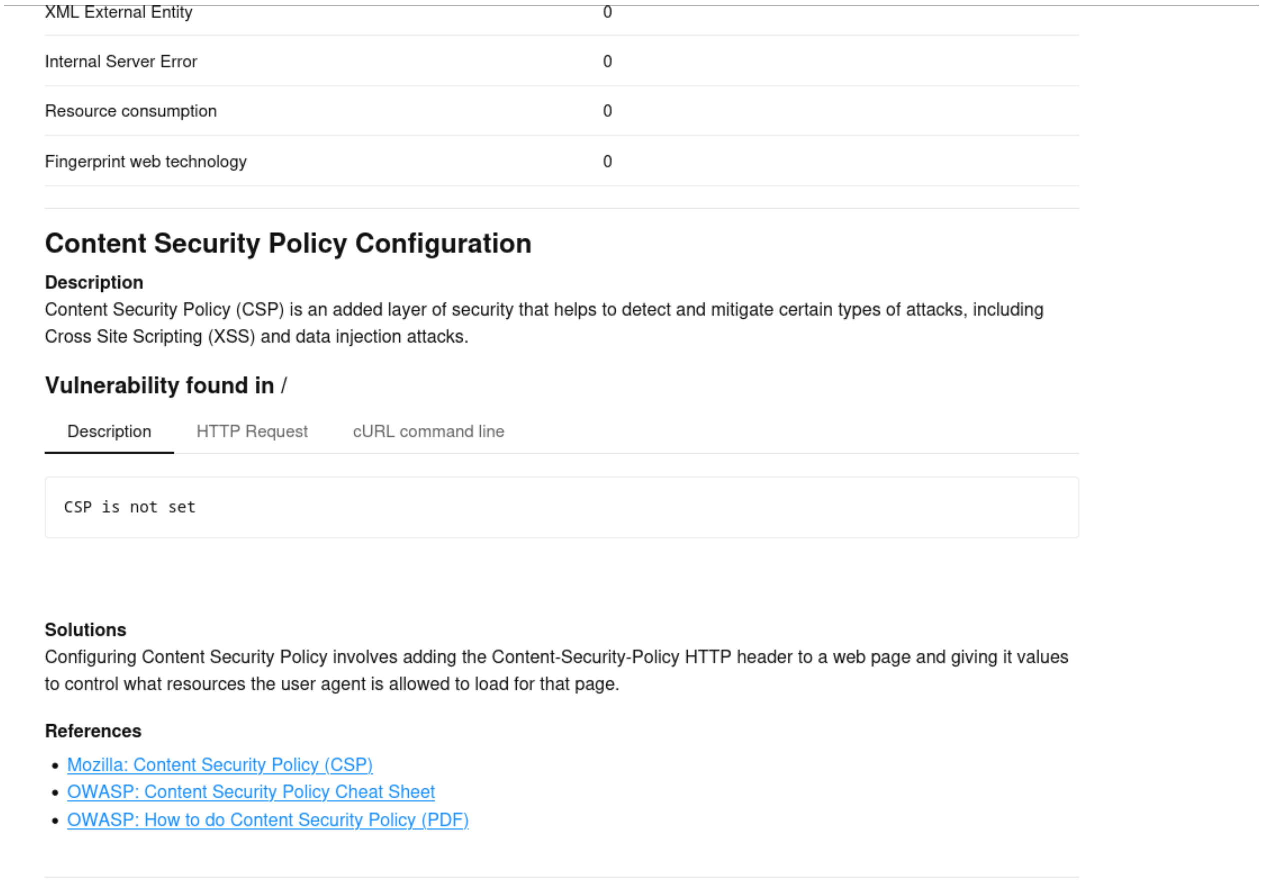
Vulnerability found in /

```
X-Content-Type-Options is not set
```

Vulnerability found in /

```
Strict-Transport-Security is not set
```

Solutions
Use the recommendations for hardening your HTTP Security Headers.



XML External Entity	0
Internal Server Error	0
Resource consumption	0
Fingerprint web technology	0

Content Security Policy Configuration

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.

Vulnerability found in /

```
CSP is not set
```

Solutions
Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

References
- Mozilla: Content Security Policy (CSP)
- OWASP: Content Security Policy Cheat Sheet
- OWASP: How to do Content Security Policy (PDF)

Vulnerability found in /eu/en/women/bags

Description | HTTP Request | cURL command line

```
CSP attribute "script-src" is missing
```

Vulnerability found in /eu/en/women/bags

Description | HTTP Request | cURL command line

```
CSP attribute "object-src" is missing
```

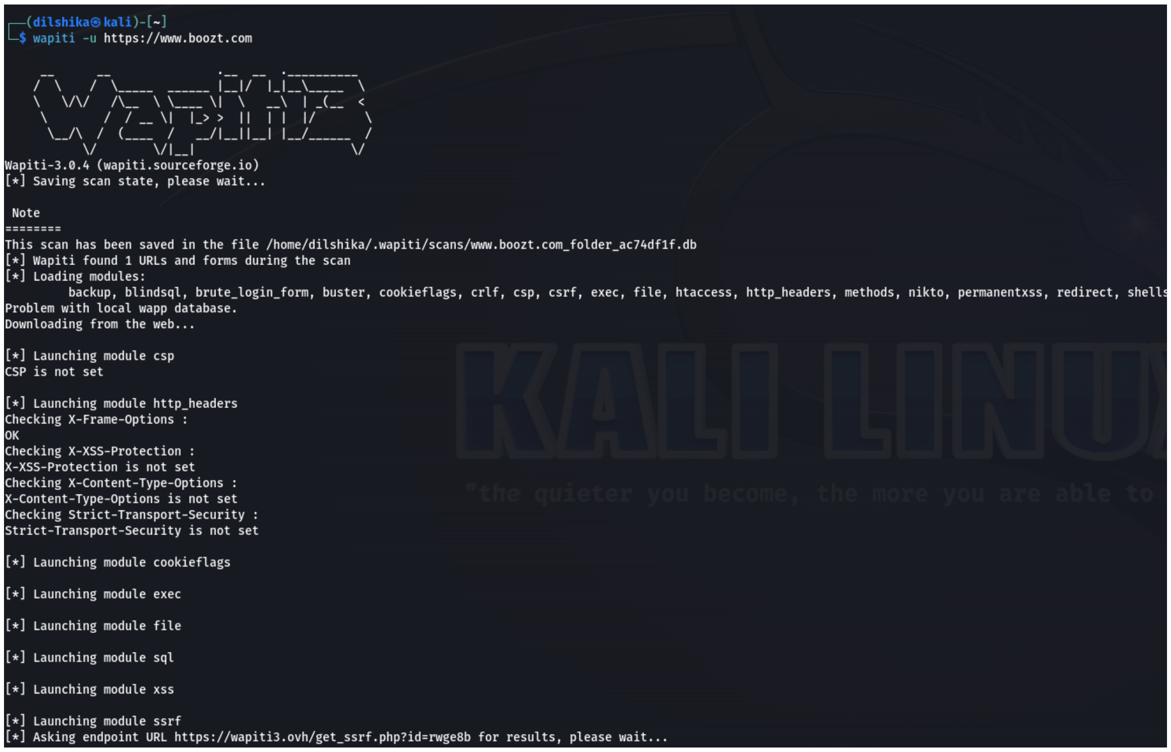
Vulnerability found in /eu/en/women/bags

Description | HTTP Request | cURL command line

```
CSP attribute "base-uri" is missing
```

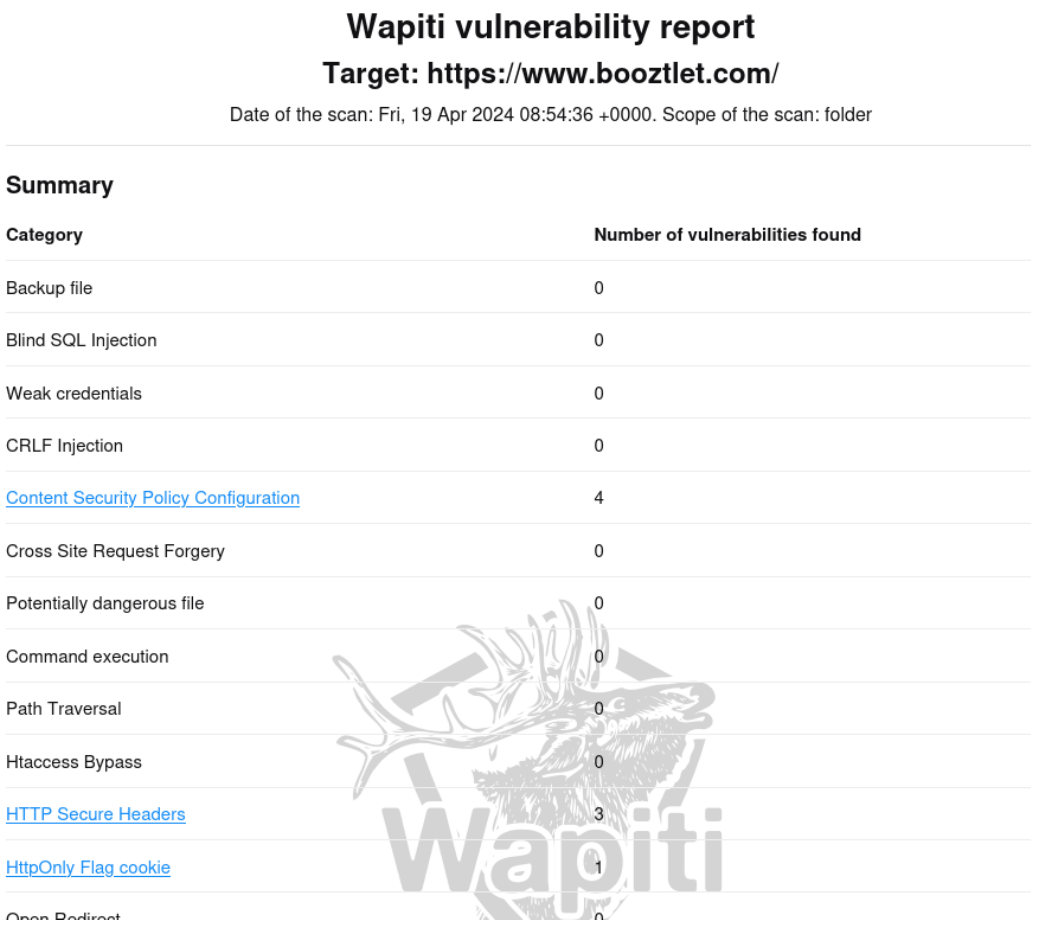
Solutions

Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.



```
┌──(dilshika㉿kali)-[~]
└─$ wapiti -u https://www.boozt.com

 __      __               .__  __  .__________
/  \    /  \_____  ______ |__|/  |_|__\_____  \
\   \/\/   /\__  \ \____ \|  \   __\  | _(__  <
 \        /  / __ \|  |_> >  ||  | |  |/       \
  \__/\  /  (____  /   __/|__||__| |__/______  /
       \/        \/|__|                      \/
Wapiti-3.0.4 (wapiti.sourceforge.io)
[*] Saving scan state, please wait...

 Note
========
This scan has been saved in the file /home/dilshika/.wapiti/scans/www.boozt.com_folder_ac74df1f.db
[*] Wapiti found 1 URLs and forms during the scan
[*] Loading modules:
         backup, blindsql, brute_login_form, buster, cookieflags, crlf, csp, csrf, exec, file, htaccess, http_headers, methods, nikto, permanentxss, redirect, shells
Problem with local wapp database.
Downloading from the web...

[*] Launching module csp
CSP is not set

[*] Launching module http_headers
Checking X-Frame-Options :
OK
Checking X-XSS-Protection :
X-XSS-Protection is not set
Checking X-Content-Type-Options :
X-Content-Type-Options is not set
Checking Strict-Transport-Security :
Strict-Transport-Security is not set

[*] Launching module cookieflags

[*] Launching module exec

[*] Launching module file

[*] Launching module sql

[*] Launching module xss

[*] Launching module ssrf
[*] Asking endpoint URL https://wapiti3.ovh/get_ssrf.php?id=rwge8b for results, please wait...
```

Wapiti scan for www.booztlet.com



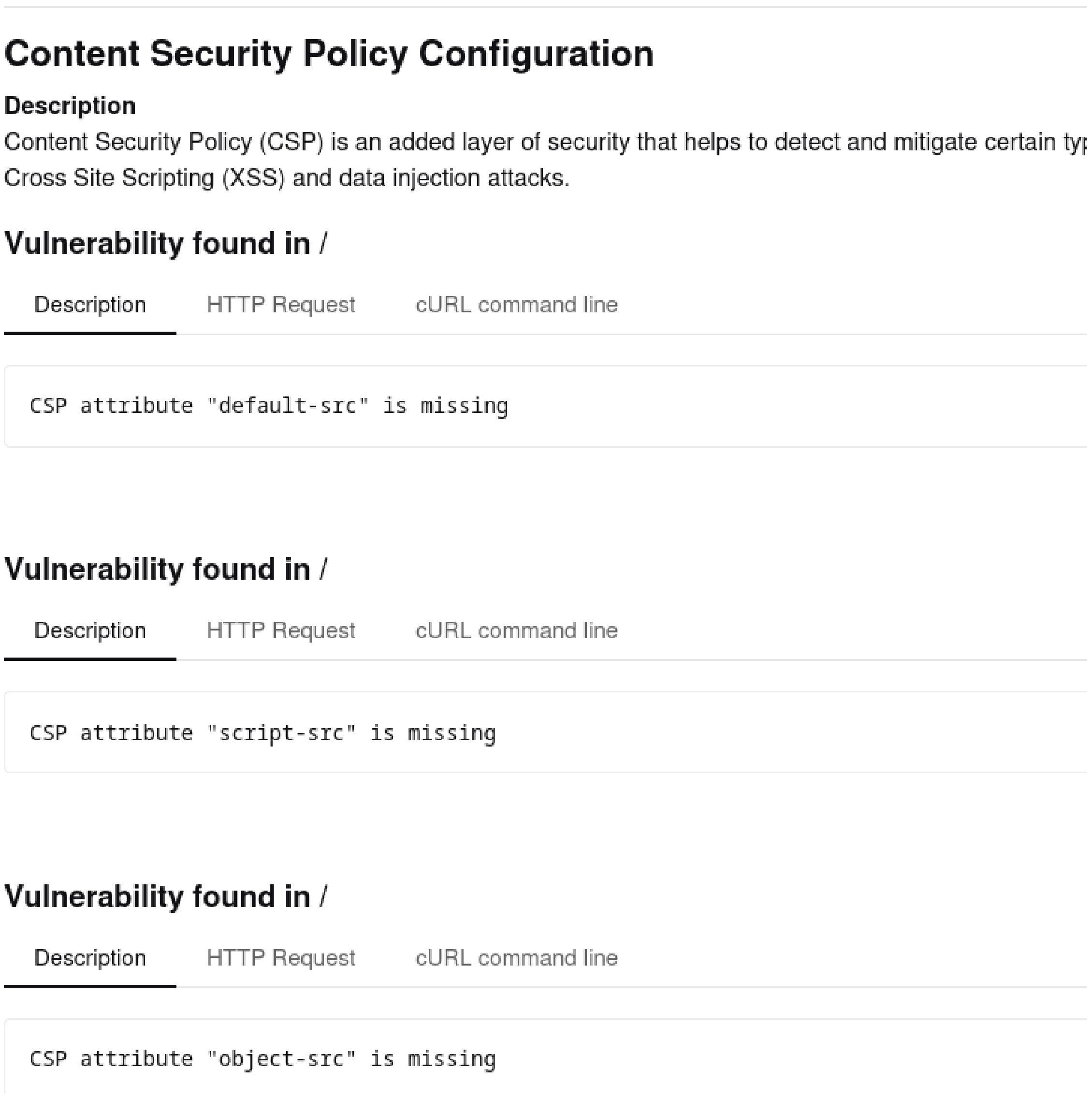
Wapiti vulnerability report

Target: https://www.booztlet.com/

Date of the scan: Fri, 19 Apr 2024 08:54:36 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	0
Blind SQL Injection	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	4
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Htaccess Bypass	0
HTTP Secure Headers	3
HttpOnly Flag cookie	1
Open Redirect	0



Content Security Policy Configuration

Description

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain typ
Cross Site Scripting (XSS) and data injection attacks.

Vulnerability found in /

Description HTTP Request cURL command line

```
CSP attribute "default-src" is missing
```

Vulnerability found in /

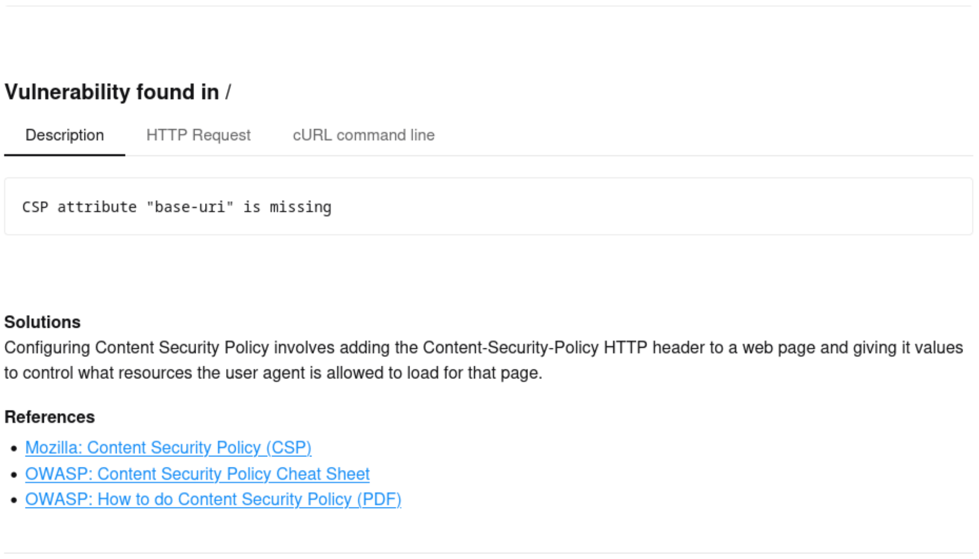
Description HTTP Request cURL command line

```
CSP attribute "script-src" is missing
```

Vulnerability found in /

Description HTTP Request cURL command line

```
CSP attribute "object-src" is missing
```



Vulnerability found in /

Description HTTP Request cURL command line

```
CSP attribute "base-uri" is missing
```

Solutions

Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.

References

- Mozilla: Content Security Policy (CSP)
- OWASP: Content Security Policy Cheat Sheet
- OWASP: How to do Content Security Policy (PDF)

HTTP Secure Headers

Description

HTTP security headers tell the browser how to behave when handling the website's content.

Vulnerability found in /

Description HTTP Request cURL command line

```
X-XSS-Protection is not set
```

Vulnerability found in /

Description HTTP Request cURL command line

```
X-Content-Type-Options is not set
```

Vulnerability found in /

Description HTTP Request cURL command line

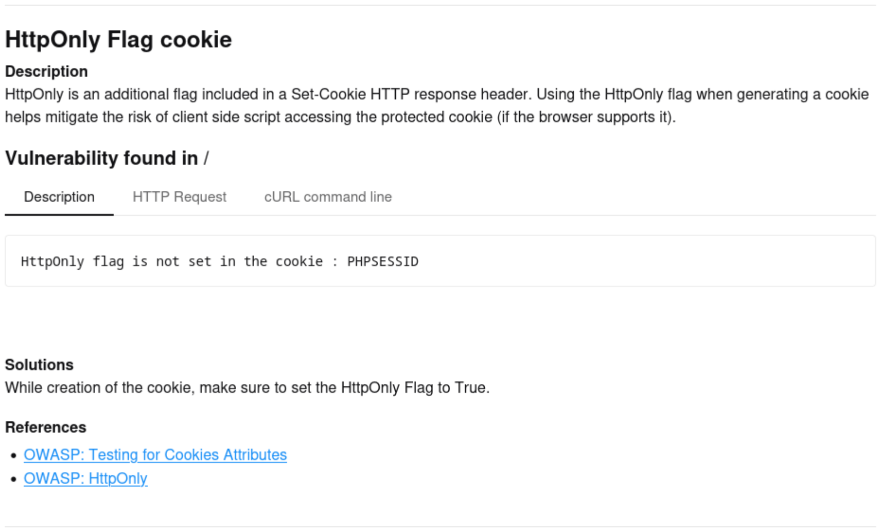
```
Strict-Transport-Security is not set
```

Solutions

Use the recommendations for hardening your HTTP Security Headers.

References

- Netsparker: HTTP Security Headers: An Easy Way to Harden Your Web Applications
- KeyCDN: Hardening Your HTTP Security Headers
- OWASP: HTTP SECURITY HEADERS (Protection For Browsers) (PDF)



HttpOnly Flag cookie

Description

HttpOnly is an additional flag included in a Set-Cookie HTTP response header. Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side script accessing the protected cookie (if the browser supports it).

Vulnerability found in /

```
HttpOnly flag is not set in the cookie : PHPSESSID
```

Solutions

While creation of the cookie, make sure to set the HttpOnly Flag to True.

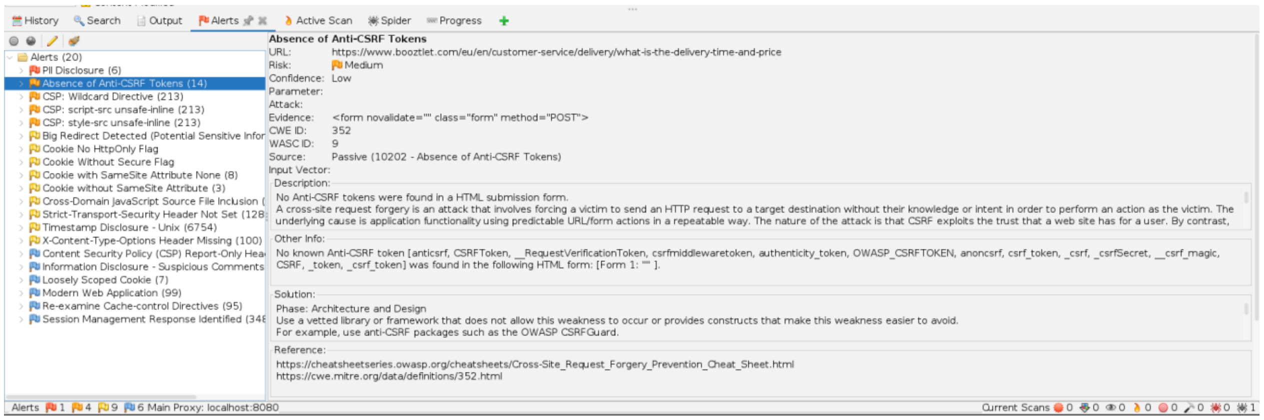
References

- OWASP: Testing for Cookies Attributes
- OWASP: HttpOnly

2024.4.20

I did an automated scan for www.booztlet.com . Apart from PII disclosure I found other medium risked alerts for CSP: wildcard vulnerability, CSP: script-src unsfe-inline, CSP-style-src-unsafe-inline, Absence of anti csrf token.

Absence of anti csrf token



Alerts (20)
- PII Disclosure (6)
- Absence of Anti-CSRF Tokens (14)
- CSP: Wildcard Directive (213)
- CSP: script-src unsafe-inline (213)
- CSP: style-src unsafe-inline (213)
- Big Redirect Detected (Potential Sensitive Infor
- Cookie No HttpOnly Flag
- Cookie Without Secure Flag
- Cookie with SameSite Attribute None (8)
- Cookie without SameSite Attribute (3)
- Cross-Domain JavaScript Source File Inclusion (
- Strict-Transport-Security Header Not Set (128
- Timestamp Disclosure - Unix (6754)
- X-Content-Type-Options Header Missing (100)
- Content Security Policy (CSP) Report-Only Hea
- Information Disclosure - Suspicious Comments
- Loosely Scoped Cookie (7)
- Modern Web Application (99)
- Re-examine Cache-control Directives (95)
- Session Management Response Identified (348

Absence of Anti-CSRF Tokens

```
URL:          https://www.booztlet.com/eu/en/customer-service/delivery/what-is-the-delivery-time-and-price
Risk:         Medium
Confidence:   Low
Parameter:
Attack:
Evidence:     <form novalidate="" class="form" method="POST">
CWE ID:       352
WASC ID:      9
Source:       Passive (10202 - Absence of Anti-CSRF Tokens)
Input Vector:
```

Description:
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,

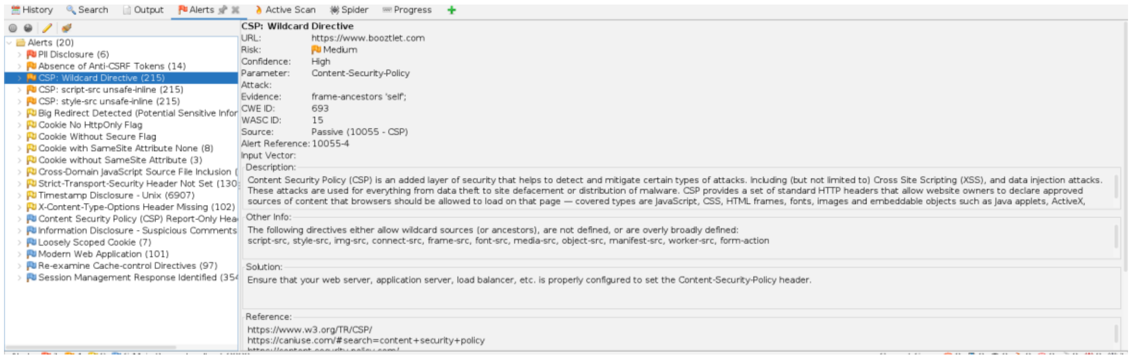
Other Info:
No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: ""].

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.

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
https://cwe.mitre.org/data/definitions/352.html

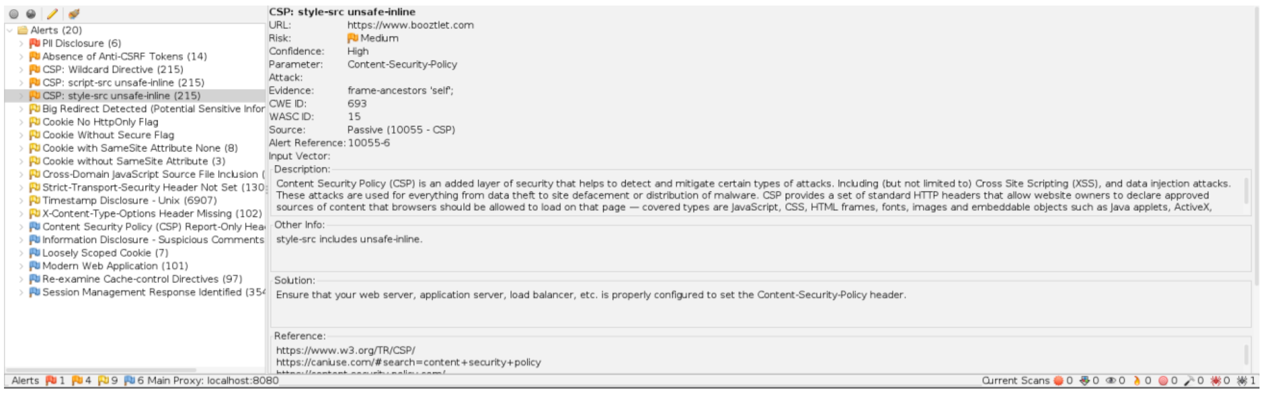
Alerts 1 4 9 6 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 1

CSP- wild card directive



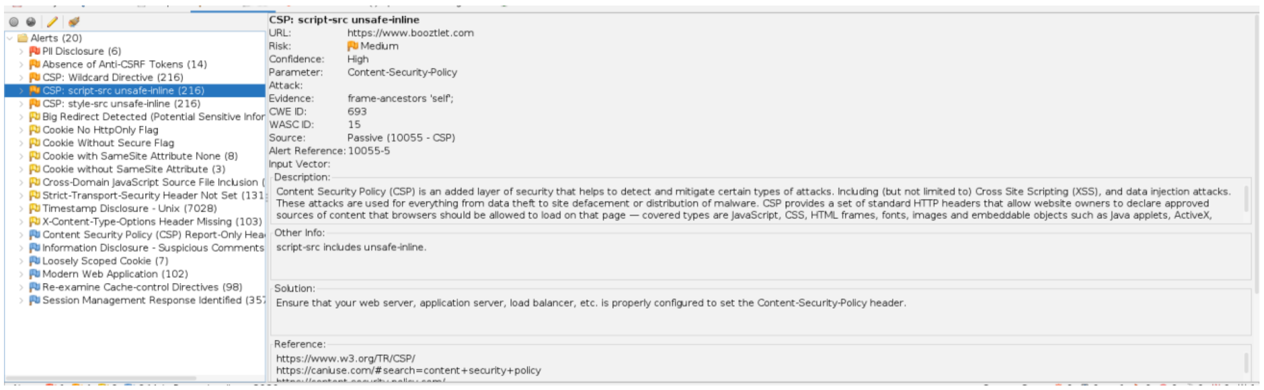
```
History  Search  Output  Alerts  Active Scan  Spider  Progress

Alerts (20)
  PII Disclosure (6)
  Absence of Anti-CSRF Tokens (14)
  CSP: Wildcard Directive (215)
  CSP: script-src unsafe-inline (215)
  CSP: style-src unsafe-inline (215)
  Big Redirect Detected (Potential Sensitive Infor
  Cookie No HttpOnly Flag
  Cookie Without Secure Flag
  Cookie with SameSite Attribute None (8)
  Cookie without SameSite Attribute (3)
  Cross-Domain JavaScript Source File Inclusion (
  Strict-Transport-Security Header Not Set (130
  Timestamp Disclosure - Unix (6907)
  X-Content-Type-Options Header Missing (102)
  Content Security Policy (CSP) Report-Only Hea
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (7)
  Modern Web Application (101)
  Re-examine Cache-control Directives (97)
  Session Management Response Identified (354

CSP: Wildcard Directive
URL:             https://www.booztlet.com
Risk:            Medium
Confidence:      High
Parameter:       Content-Security-Policy
Attack:
Evidence:        frame-ancestors 'self';
CWE ID:          693
WASC ID:         15
Source:          Passive (10055 - CSP)
Alert Reference: 10055-4
Input Vector:
Description:
 Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) 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,
Other Info:
 The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined:
 script-src, style-src, img-src, connect-src, frame-src, font-src, media-src, object-src, manifest-src, worker-src, form-action
Solution:
 Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference:
 https://www.w3.org/TR/CSP/
 https://caniuse.com/#search=content+security+policy
```

CSP – unsafe inline

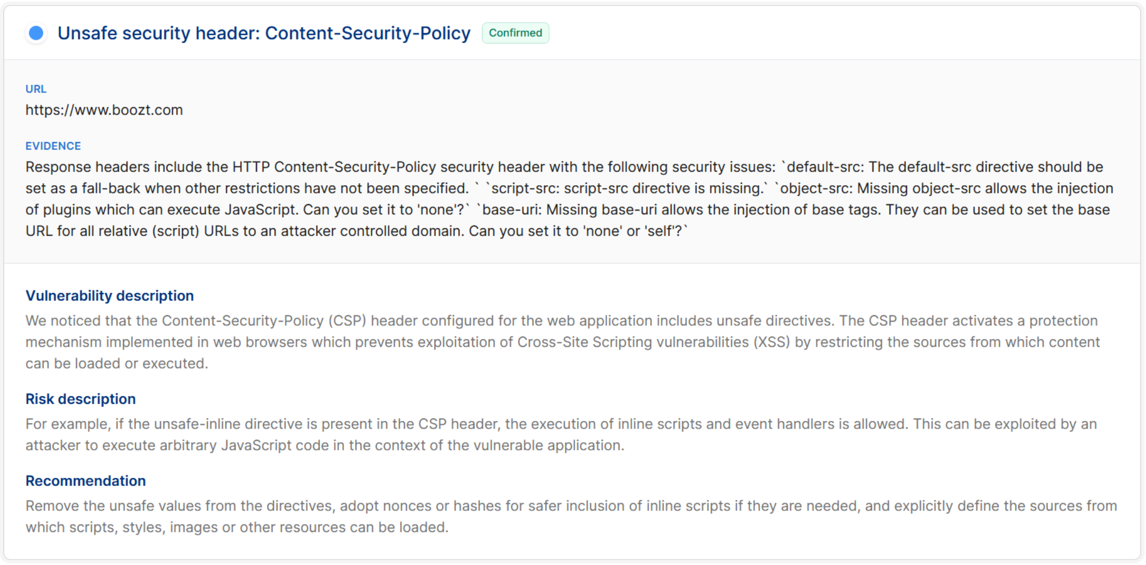


```
Alerts (20)
  PII Disclosure (6)
  Absence of Anti-CSRF Tokens (14)
  CSP: Wildcard Directive (215)
  CSP: script-src unsafe-inline (215)
  CSP: style-src unsafe-inline (215)
  Big Redirect Detected (Potential Sensitive Infor
  Cookie No HttpOnly Flag
  Cookie Without Secure Flag
  Cookie with SameSite Attribute None (8)
  Cookie without SameSite Attribute (3)
  Cross-Domain JavaScript Source File Inclusion (
  Strict-Transport-Security Header Not Set (130
  Timestamp Disclosure - Unix (6907)
  X-Content-Type-Options Header Missing (102)
  Content Security Policy (CSP) Report-Only Hea
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (7)
  Modern Web Application (101)
  Re-examine Cache-control Directives (97)
  Session Management Response Identified (354

CSP: style-src unsafe-inline
URL:             https://www.booztlet.com
Risk:            Medium
Confidence:      High
Parameter:       Content-Security-Policy
Attack:
Evidence:        frame-ancestors 'self';
CWE ID:          693
WASC ID:         15
Source:          Passive (10055 - CSP)
Alert Reference: 10055-6
Input Vector:
Description:
 Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) 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,
Other Info:
 style-src includes unsafe-inline.
Solution:
 Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference:
 https://www.w3.org/TR/CSP/
 https://caniuse.com/#search=content+security+policy

Alerts 1 4 9 6 Main Proxy: localhost:8080                         Current Scans 0 0 0 0 0 0 0 1
```

Csp -unsafe inline script



```
Alerts (20)
  PII Disclosure (6)
  Absence of Anti-CSRF Tokens (14)
  CSP: Wildcard Directive (216)
  CSP: script-src unsafe-inline (216)
  CSP: style-src unsafe-inline (216)
  Big Redirect Detected (Potential Sensitive Infor
  Cookie No HttpOnly Flag
  Cookie Without Secure Flag
  Cookie with SameSite Attribute None (8)
  Cookie without SameSite Attribute (3)
  Cross-Domain JavaScript Source File Inclusion (
  Strict-Transport-Security Header Not Set (131
  Timestamp Disclosure - Unix (7028)
  X-Content-Type-Options Header Missing (103)
  Content Security Policy (CSP) Report-Only Hea
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (7)
  Modern Web Application (102)
  Re-examine Cache-control Directives (98)
  Session Management Response Identified (357

CSP: script-src unsafe-inline
URL:             https://www.booztlet.com
Risk:            Medium
Confidence:      High
Parameter:       Content-Security-Policy
Attack:
Evidence:        frame-ancestors 'self';
CWE ID:          693
WASC ID:         15
Source:          Passive (10055 - CSP)
Alert Reference: 10055-5
Input Vector:
Description:
 Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) 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,
Other Info:
 script-src includes unsafe-inline.
Solution:
 Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference:
 https://www.w3.org/TR/CSP/
 https://caniuse.com/#search=content+security+policy
```



Unsafe security header: Content-Security-Policy Confirmed

URL

https://www.boozt.com

EVIDENCE

Response headers include the HTTP Content-Security-Policy security header with the following security issues: `default-src: The default-src directive should be set as a fall-back when other restrictions have not been specified. ` `script-src: script-src directive is missing.` `object-src: Missing object-src allows the injection of plugins which can execute JavaScript. Can you set it to 'none'?` `base-uri: Missing base-uri allows the injection of base tags. They can be used to set the base URL for all relative (script) URLs to an attacker controlled domain. Can you set it to 'none' or 'self'?`

Vulnerability description

We noticed that the Content-Security-Policy (CSP) header configured for the web application includes unsafe directives. The CSP header activates a protection mechanism implemented in web browsers which prevents exploitation of Cross-Site Scripting vulnerabilities (XSS) by restricting the sources from which content can be loaded or executed.

Risk description

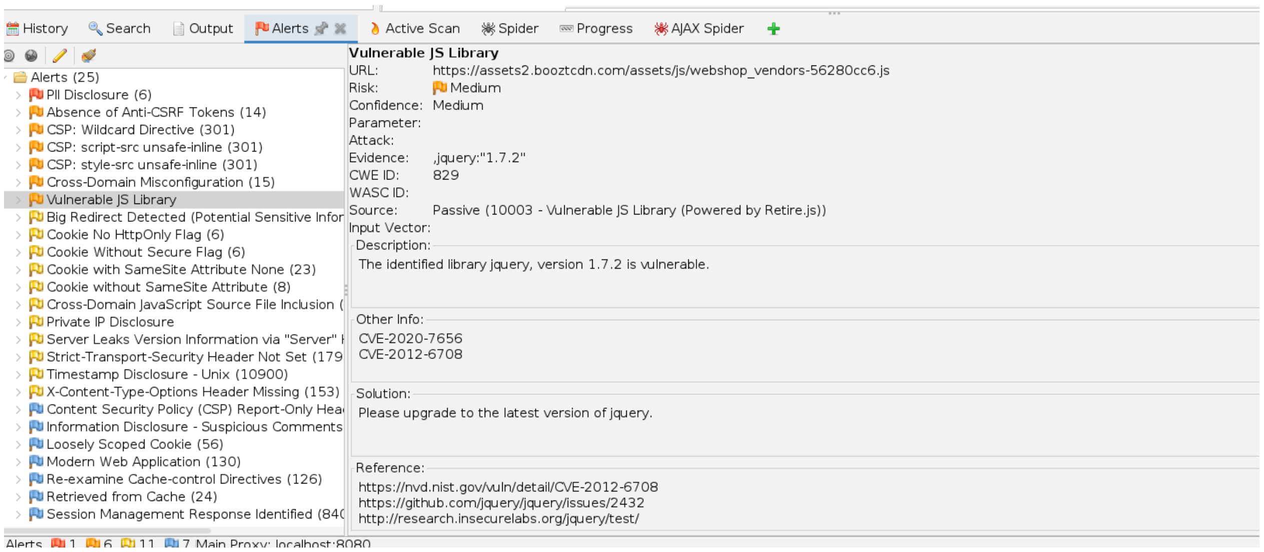
For example, if the unsafe-inline directive is present in the CSP header, the execution of inline scripts and event handlers is allowed. This can be exploited by an attacker to execute arbitrary JavaScript code in the context of the vulnerable application.

Recommendation

Remove the unsafe values from the directives, adopt nonces or hashes for safer inclusion of inline scripts if they are needed, and explicitly define the sources from which scripts, styles, images or other resources can be loaded.

- **<u>2024.04.22</u>**

Found vulnerable javascript library in www.booztcdn.com



```
Alerts (25)
  PII Disclosure (6)
  Absence of Anti-CSRF Tokens (14)
  CSP: Wildcard Directive (301)
  CSP: script-src unsafe-inline (301)
  CSP: style-src unsafe-inline (301)
  Cross-Domain Misconfiguration (15)
  Vulnerable JS Library
  Big Redirect Detected (Potential Sensitive Infor
  Cookie No HttpOnly Flag (6)
  Cookie Without Secure Flag (6)
  Cookie with SameSite Attribute None (23)
  Cookie without SameSite Attribute (8)
  Cross-Domain JavaScript Source File Inclusion (
  Private IP Disclosure
  Server Leaks Version Information via "Server" I
  Strict-Transport-Security Header Not Set (179
  Timestamp Disclosure - Unix (10900)
  X-Content-Type-Options Header Missing (153)
  Content Security Policy (CSP) Report-Only Hea
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (56)
  Modern Web Application (130)
  Re-examine Cache-control Directives (126)
  Retrieved from Cache (24)
  Session Management Response Identified (84(
```

```
Vulnerable JS Library
URL:          https://assets2.booztcdn.com/assets/js/webshop_vendors-56280cc6.js
Risk:         Medium
Confidence:   Medium
Parameter:
Attack:
Evidence:     ,jquery:"1.7.2"
CWE ID:       829
WASC ID:
Source:       Passive (10003 - Vulnerable JS Library (Powered by Retire.js))
Input Vector:
Description:
 The identified library jquery, version 1.7.2 is vulnerable.

Other Info:
 CVE-2020-7656
 CVE-2012-6708

Solution:
 Please upgrade to the latest version of jquery.

Reference:
 https://nvd.nist.gov/vuln/detail/CVE-2012-6708
 https://github.com/jquery/jquery/issues/2432
 http://research.insecurelabs.org/jquery/test/
```

```
Alerts 1 6 11 7 Main Proxy: localhost:8080
```

Vulnerable url – https://assets2.booztcdn.com/assets/js/webshop_vendors-56280cc6.js

References:

https://nvd.nist.gov/vuln/detail/CVE-2012-6708

https://github.com/jquery/jquery/issues/2432

http://research.insecurelabs.org/jquery/test

related cve s

CVE-2020-11023

CVE-2020-11022

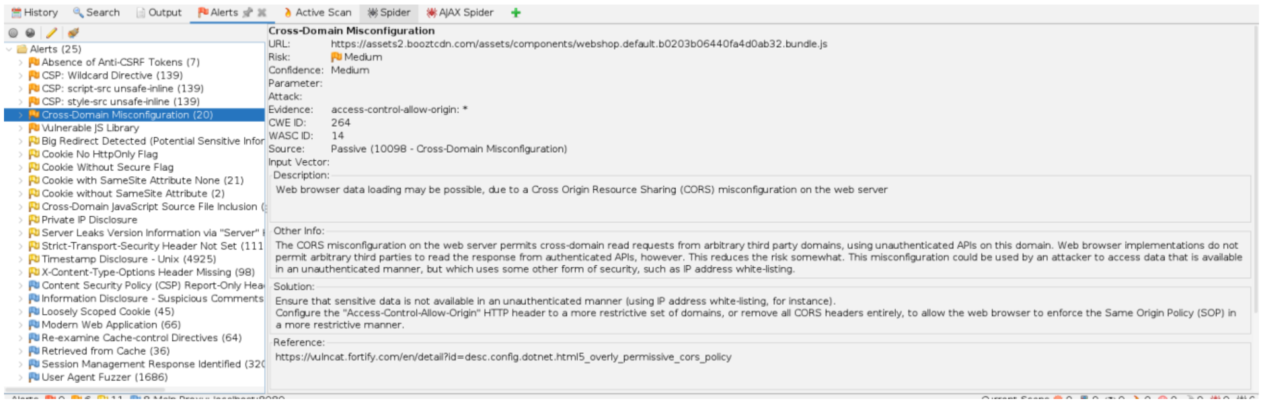
CVE-2015-9251

CVE-2019-11358

CVE-2020-7656

CVE-2012-6708

Found a medium risk vulnerability for cross domain misconfiguration. The vulnerability was found using owasp zap tool



```
Alerts (25)
  Absence of Anti-CSRF Tokens (7)
  CSP: Wildcard Directive (139)
  CSP: script-src unsafe-inline (139)
  CSP: style-src unsafe-inline (139)
  Cross-Domain Misconfiguration (20)
  Vulnerable JS Library
  Big Redirect Detected (Potential Sensitive Infor
  Cookie No HttpOnly Flag
  Cookie Without Secure Flag
  Cookie with SameSite Attribute None (21)
  Cookie without SameSite Attribute (2)
  Cross-Domain JavaScript Source File Inclusion (
  Private IP Disclosure
  Server Leaks Version Information via "Server"
  Strict-Transport-Security Header Not Set (111
  Timestamp Disclosure - Unix (4925)
  X-Content-Type-Options Header Missing (98)
  Content Security Policy (CSP) Report-Only Hea
  Information Disclosure - Suspicious Comments
  Loosely Scoped Cookie (45)
  Modern Web Application (66)
  Re-examine Cache-control Directives (64)
  Retrieved from Cache (36)
  Session Management Response Identified (32
  User Agent Fuzzer (1686)
```

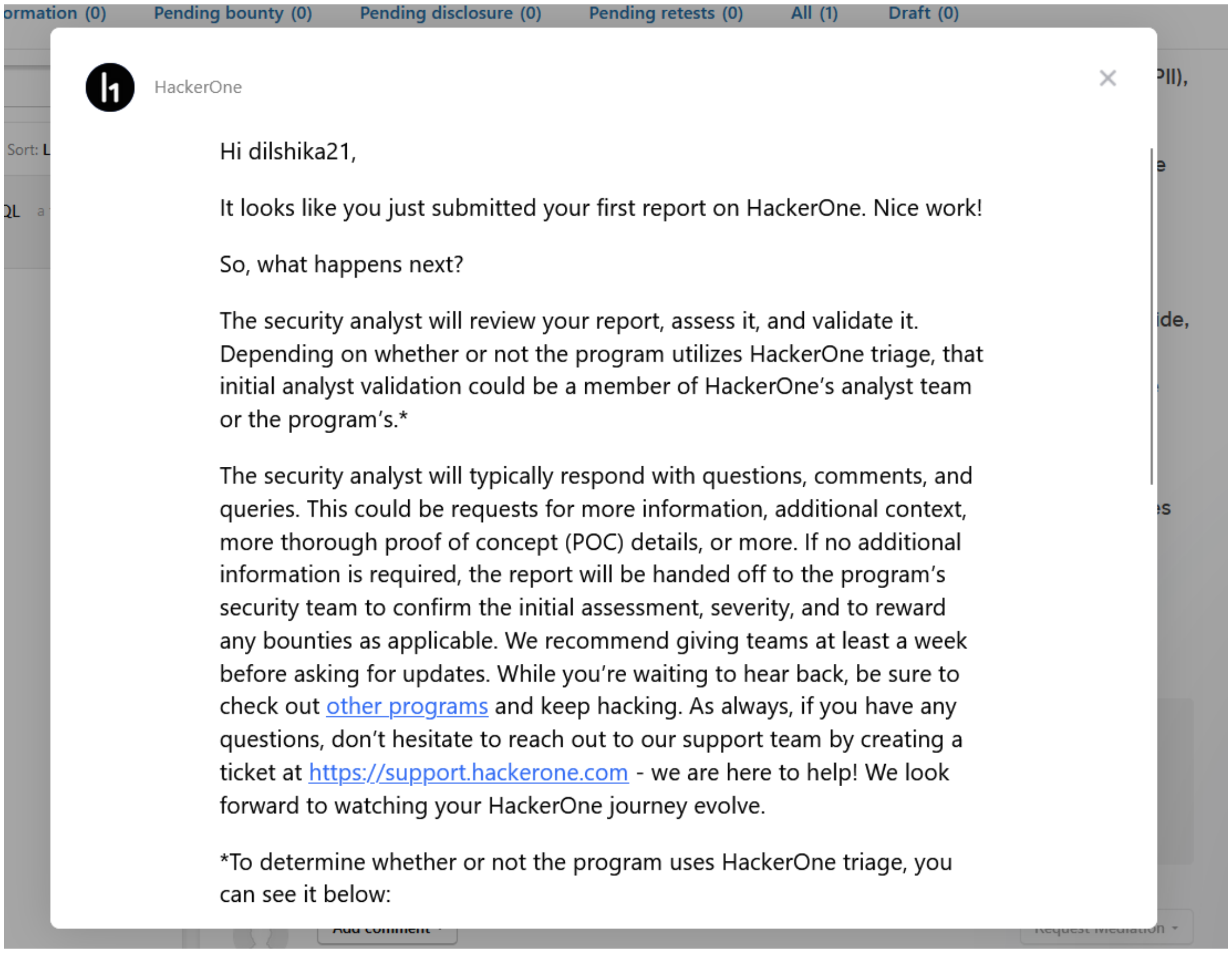
```
Cross-Domain Misconfiguration
URL:         https://assets2.booztcdn.com/assets/components/webshop.default.b0203b06440fa4d0ab32.bundle.js
Risk:        Medium
Confidence:  Medium
Parameter:
Attack:
Evidence:    access-control-allow-origin: *
CWE ID:      264
WASC ID:     14
Source:      Passive (10098 - Cross-Domain Misconfiguration)
Input Vector:
Description:
 Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server
Other Info:
 The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not
 permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available
 in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.
Solution:
 Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).
 Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in
 a more restrictive manner.
Reference:
 https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy
```

- **<u>2024.04.23</u>**

Started to finish some reports .Today started to finish reports for sql vulnerabilities I found, and also cross domain misconfiguration, csp : wildcard issue.

- **<u>2024.04.24</u>**

Today I submitted my first report to my bug bounty program. The platform I used was try hack me room. I created a template for my report and submit the report.



Hi dilshika21,

It looks like you just submitted your first report on HackerOne. Nice work!

So, what happens next?

The security analyst will review your report, assess it, and validate it. Depending on whether or not the program utilizes HackerOne triage, that initial analyst validation could be a member of HackerOne's analyst team or the program's.*

The security analyst will typically respond with questions, comments, and queries. This could be requests for more information, additional context, more thorough proof of concept (POC) details, or more. If no additional information is required, the report will be handed off to the program's security team to confirm the initial assessment, severity, and to reward any bounties as applicable. We recommend giving teams at least a week before asking for updates. While you're waiting to hear back, be sure to check out other programs and keep hacking. As always, if you have any questions, don't hesitate to reach out to our support team by creating a ticket at https://support.hackerone.com - we are here to help! We look forward to watching your HackerOne journey evolve.

*To determine whether or not the program uses HackerOne triage, you can see it below:

- ✓ **Challenges**

I had never uploaded a report in hacker one me before, I did not know the procedure, anyhow I googled and found out how to report a vulnerability in hacker one.

Another challenge I faced was that I can only submit a one report per day, because I was a beginner in hacker one.

- **<u>2024.4.25</u>**

Started making the try hack me room. The try hack me room I created was regarding the hash disclosure vulnerability. It highlights the risk of publicly disclosed hashes, because attackers can get the plain text by using tools and attacks like rainbow attacks.

Here is the try hack me room link.

- ✓ **Challenges**

I had issues with uploading virtual machine to try hack me. It gave me errors repeatedly .

So, I deployed my webpage using Netlify and attached the link in the try hack me room.

2024.4.30

I uploaded copy of this journal to the lean pub.

Following is the link

7. Conclusion

To conclude, this report outlines my bug bounty journey, detailing how I selected programs, found vulnerabilities, and utilized various tools. It emphasizes understanding OWASP Top 10 vulnerabilities and addressing security risks. Daily logs illustrate the challenges faced and the perseverance required. Moreover, it showcases proof of concept demonstrations and insights into report submission, highlighting dedication to thoroughness and professionalism in bug hunting. Overall, this report reflects the ongoing learning and commitment involved in advancing cybersecurity through ethical hacking practices.

8. References

https://www.cloudflare.com/learning/security/threats/owasp-top-10/

https://www.kali.org/tools/burpsuite/

https://www.shodan.io/

https://sqlmap.org/

https://medium.com/@cuncis/amass-an-overview-of-the-network-reconnaissance-tool-79049f34bb46

https://portswigger.net/web-security/sql-injection

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

https://owasp.org/www-project-top-ten/

https://github.com/pgaijin66/XSS-Payloads/blob/master/payload/payload.txt#L18

https://hackerone.com/boozt/bounty_table_versions?type=team&change=2023-10-17T08%3A31%3A22.175Z