Title: Practical Application of Red Team Operations and Offensive Security Techniques

Author: Utkarsh Kumar
Date: 07/01/2026
Environment: Kali Linux VM, Windows VM, Metasploitable2/3, Isolated Lab Network, Open-Source Offensive Security Tools

Executive Summary

This report documents a series of hands-on red team exercises designed to simulate real-world offensive security operations across the cyber attack lifecycle. The practical tasks focused on reconnaissance and OSINT, phishing-based initial access, vulnerability exploitation, lateral movement and persistence, social engineering, exploit development, and post-exploitation activities including credential harvesting and data exfiltration.

All tasks were conducted in a controlled and authorized lab environment using intentionally vulnerable systems and test data. Open-source tools were leveraged to emulate realistic attacker techniques aligned with the MITRE ATT&CK framework. The exercises demonstrate the ability to identify attack surfaces, exploit security weaknesses, move laterally within a network, maintain persistence, and assess the impact of post-exploitation actions.

The findings highlight common organizational security gaps such as exposed services, weak user awareness, insecure configurations, and insufficient monitoring. This report also emphasizes the importance of defensive controls including patch management, credential protection, network segmentation, and security awareness training to mitigate red team–style attacks.

Task 1: OSINT and Reconnaissance Lab

1.1 Objective

The objective of this task was to perform **OSINT-based reconnaissance** to enumerate subdomains and identify **publicly exposed services** using internet-wide search engines and OSINT tools. This task demonstrates how attackers can identify potential attack surfaces during the reconnaissance phase without actively exploiting target systems.

1.2 Tools Used

- Recon-ng
- Shodan
- Maltego

1.3 Methodology

The reconnaissance process involved two primary activities:

1. **Subdomain enumeration** using Recon-ng to identify domain-related assets.
2. **Exposed service identification** using Shodan to analyze publicly accessible hosts and services.

Maltego was used to visually correlate OSINT findings such as domains, DNS records, and IP addresses. All activities were performed using **publicly available data** for educational purposes only.

1.4 Subdomain Enumeration (Recon-ng)

1.4.1 Procedure

- The recon/domains-hosts/bing_domain_web module was executed in Recon-ng.
- The target domain was set to example.com.
- Discovered subdomains and IP addresses were recorded.

1.4.2 Findings

Subdomain	IP Address	Notes
www.example.com	104.18.27.120	Hosts web server

1.4.3 Observation

The enumeration revealed a publicly accessible web subdomain, which could be further analyzed in later phases for service fingerprinting or vulnerability scanning.

Recon-ng output showing subdomain enumeration.

1.5 Exposed Services Identification (Shodan)

1.5.1 Procedure

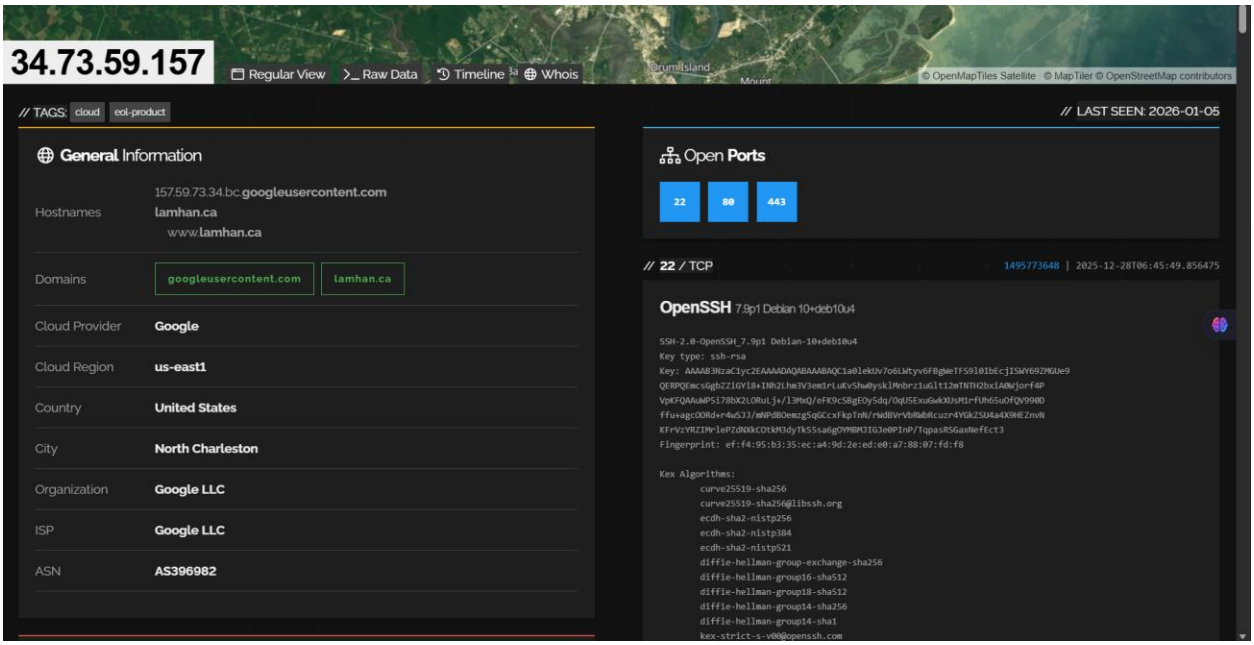
- Shodan was queried using:
- apache country:US
- Publicly exposed Apache web servers were analyzed.

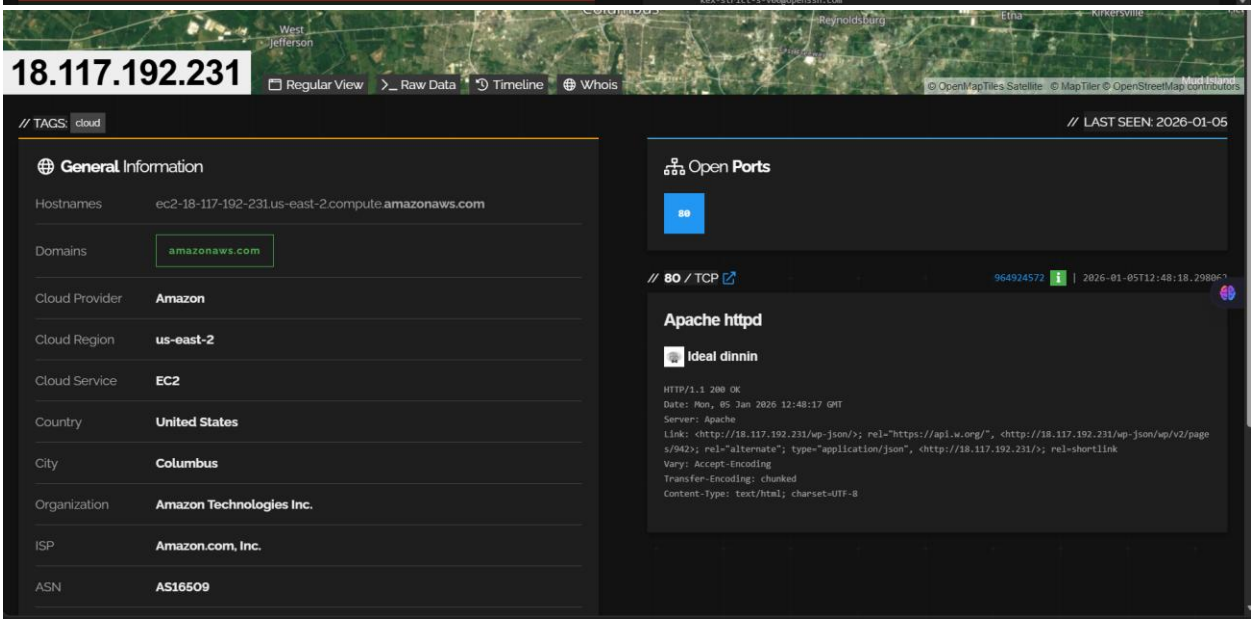
1.5.2 Summary of Identified Hosts

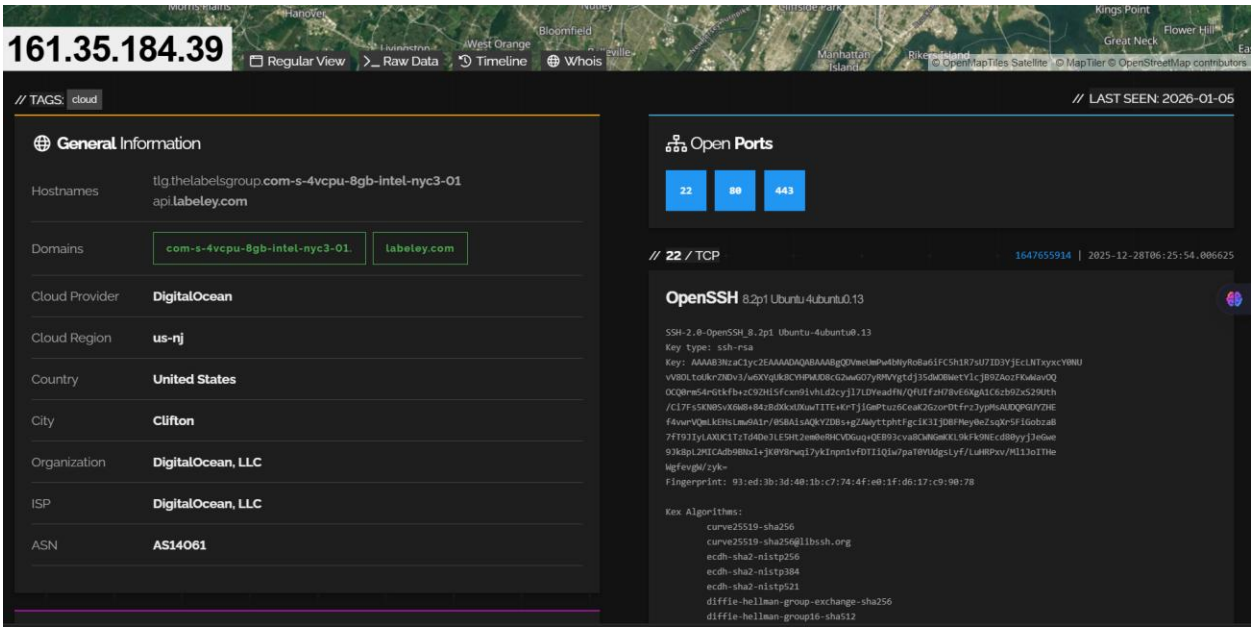
Shodan analysis identified multiple publicly exposed cloud-hosted servers running web and SSH services. The findings included Apache web servers on Amazon EC2 and SSH-enabled systems hosted on Google Cloud and DigitalOcean. Exposed services such as HTTP and SSH increase attack surface and may be targeted if improperly secured.

1.5.3 Key Shodan Findings

IP Address	Cloud Provider	Open Ports	Services Identified
34.73.59.157	Google Cloud	22, 80, 443	OpenSSH, HTTP/HTTPS
18.117.192.231	AWS (EC2)	80	Apache HTTP Server
161.35.184.39	DigitalOcean	22, 80, 443	OpenSSH, Web Services


34.73.59.157
Regular View
Raw Data
Timeline
Whois
© OpenMapTiles Satellite © MapTiler © OpenStreetMap contributors
// TAGS: cloud eol-product
// LAST SEEN: 2026-01-05
General Information
Hostnames 157.59.73.34.bc.googleusercontent.com
lamhan.ca
www.lamhan.ca
Domains googleusercontent.com lamhan.ca
Cloud Provider Google
Cloud Region us-east1
Country United States
City North Charleston
Organization Google LLC
ISP Google LLC
ASN AS396982
Open Ports
22 80 443
// 22 / TCP
1495773648 | 2025-12-28T06:45:49.856475
OpenSSH 7.9p1 Debian 10+deb10u4
SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u4
Key type: ssh-rsa
Key: AAAAB3NzaC1yc2EAAAADAQABAAABAQC1a0lekUv7o6LWtyv6FBgWeTF5910IbEcjISWY69ZMGUe9
QERPQEmcsGgbZZlGYl8+INh2LhmJV3emtruKvShw0ysklMnbrz1uGlt12mTNTH2hxiA0WjorF4P
VpKFQAAuWP5i78bX2LORuIj+/13MxQ/eFK9cSBgEOy5dq/OqU5ExuGwkXUsM1rfUh65uOfQV990D
ffu+agcOORd+r4u5JJ/mNPdBOemzg5qGCcxfkpTnN/rWdBVrVbRabRcuzr4YGkZSU4a4X9HEZnvN
KFrVzYRZIMrlePZdNXkCOtkM3dyTk5Ssa6gOYMBMJIGJe0PInP/TqpasRSGaxNefEct3
Fingerprint: ef:f4:95:b3:35:ec:a4:9d:2e:ed:e0:a7:88:07:fd:f8
Kex Algorithms:
curve25519-sha256
curve25519-sha256@libssh.org
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521
diffie-hellman-group-exchange-sha256
diffie-hellman-group16-sha512
diffie-hellman-group18-sha512
diffie-hellman-group14-sha256
diffie-hellman-group14-sha1
kex-strict-s-v00@openssh.com

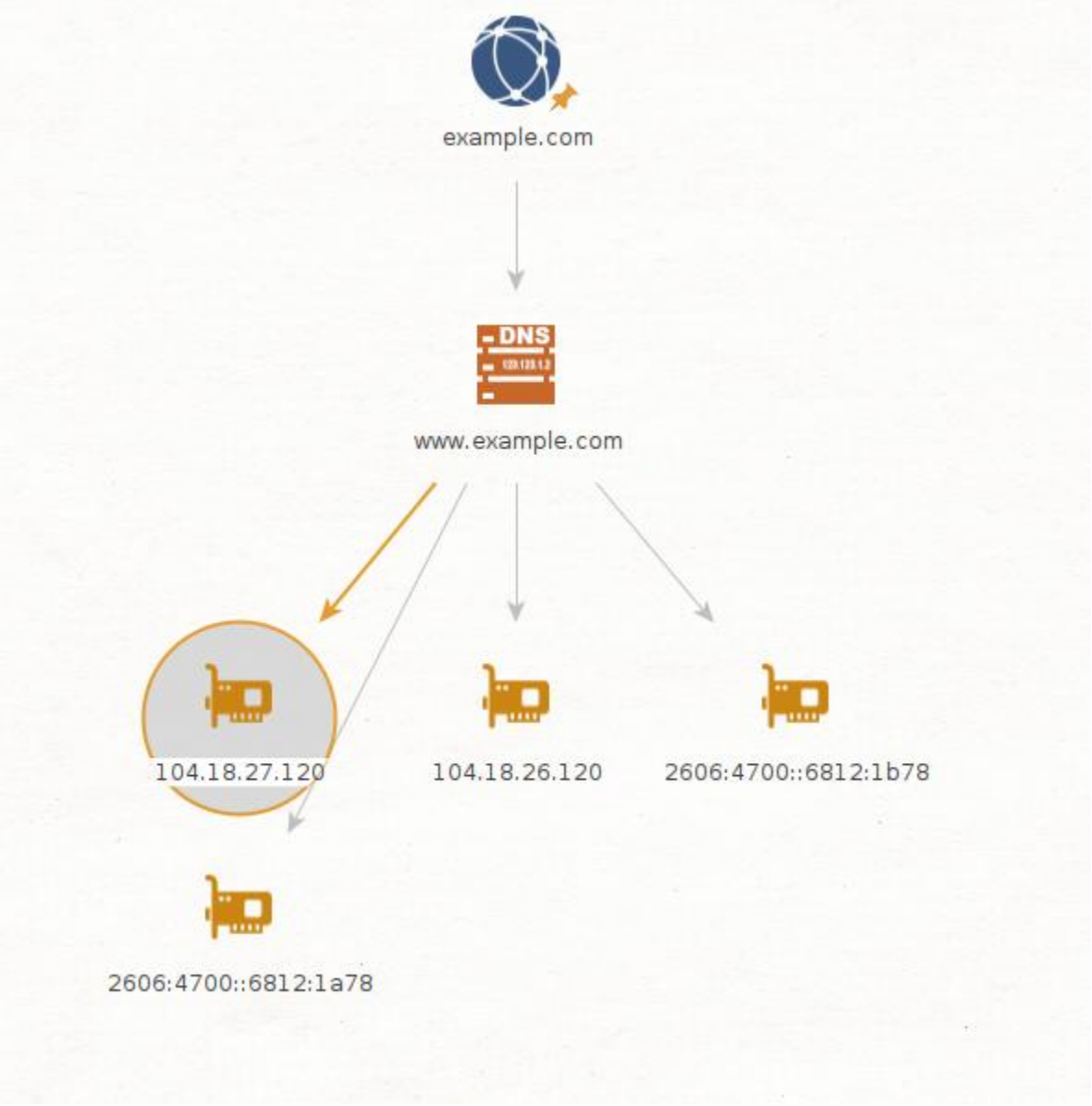

18.117.192.231
Regular View
Raw Data
Timeline
Whois
© OpenMapTiles Satellite © MapTiler © OpenStreetMap contributors
// TAGS: cloud
// LAST SEEN: 2026-01-05
General Information
Hostnames ec2-18-117-192-231.us-east-2.compute.amazonaws.com
Domains amazonaws.com
Cloud Provider Amazon
Cloud Region us-east-2
Cloud Service EC2
Country United States
City Columbus
Organization Amazon Technologies Inc.
ISP Amazon.com, Inc.
ASN AS16509
Open Ports
80
// 80 / TCP
964924572 | 2026-01-05T12:48:18.298062
Apache httpd
Ideal dinnin
HTTP/1.1 200 OK
Date: Mon, 05 Jan 2026 12:48:17 GMT
Server: Apache
Link: <http://18.117.192.231/wp-json/>; rel="https://api.w.org/", <http://18.117.192.231/wp-json/wp/v2/pages/942>; rel="alternate"; type="application/json", <http://18.117.192.231/>; rel=shortlink
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8



Shodan host details showing open ports, services, and cloud provider metadata.

1.6 OSINT Correlation (Maltego)

Maltego was used to correlate the target domain with DNS records and associated IPv4 and IPv6 addresses. The visual graph demonstrated how a single domain can be expanded into infrastructure-level intelligence using OSINT techniques.



Maltego graph showing domain-to-DNS and IP address relationships.

Task 2: Phishing Simulation

1.1 Objective

The objective of this task was to simulate a **phishing-based initial access attack** in a controlled lab environment to understand how attackers harvest user credentials using social engineering techniques. The task demonstrates the effectiveness of phishing campaigns and highlights the risks associated with user interaction with malicious links.

1.2 Tools Used

- Gophish

- Evilginx2

1.3 Lab Environment

- **Attacker Machine:** Kali Linux
- **Victim Machine:** Test Virtual Machine
- **Network Configuration:** Isolated lab network (NAT / Host-only)
- **Credentials Used:** Test credentials only

1.4 Methodology

The phishing simulation followed a structured approach:

1. A phishing landing page was cloned using Evilginx2 in the lab environment.
2. The cloned login page was tested to ensure proper credential capture.
3. A phishing email campaign was created and launched using Gophish.
4. The phishing email was delivered to the victim VM.
5. User interaction with the phishing link resulted in credential submission.
6. Captured credentials were logged and analyzed.

1.5 Phishing Campaign Setup

Campaign Details

- **Phishing Method:** Credential harvesting via fake login page
- **Delivery Mechanism:** Email-based phishing
- **Landing Page:** Evilginx2 cloned login page
- **Target:** Test user on victim VM

1.6 Credential Harvesting Results

Timestamp	IP Address	Username/Password	Risk	Notes
2026-01-06 10:20:40	192.168.0.5	testuser/pass123	High	Successful capture

1.7 Observations

- The phishing email successfully enticed the user to click the embedded link.
- Credentials were submitted through the cloned login page without user suspicion.

- No security warnings or alerts were triggered during the interaction.
- This demonstrates how phishing remains a highly effective technique for initial access.

Task 3: Vulnerability Exploitation

3.1 Objective

The objective of this task was to identify and exploit a vulnerable web application in a controlled lab environment. The task demonstrates how attackers scan for exposed services, identify known vulnerabilities, exploit them to gain remote access, and recommend remediation measures to mitigate such risks.

3.2 Tools Used

- Nmap
- Metasploit Framework
- OWASP ZAP

3.3 Lab Environment

- **Attacker Machine:** Kali Linux
- **Target Machine:** Metasploitable3 (Intentionally Vulnerable VM)
- **Network Configuration:** Isolated lab network

3.4 Methodology

The vulnerability exploitation process followed these steps:

1. Network and service scanning was performed using Nmap to identify open ports and running services.
2. The web application hosted on the target system was manually inspected to identify framework information.
3. Automated web vulnerability scanning was performed using OWASP ZAP.
4. A known Apache Struts remote code execution vulnerability was exploited using Metasploit.
5. Successful exploitation was validated by gaining remote shell access.
6. Remediation steps were proposed and verified.

3.5 Vulnerability Identification

Nmap scanning revealed multiple open ports associated with web services. Further analysis indicated the presence of an outdated Apache Struts framework, which is known to be vulnerable to remote code execution attacks.

OWASP ZAP highlighted insecure configurations and outdated components related to the web application.

3.6 Exploitation

The Apache Struts vulnerability was exploited using the Metasploit module exploit/multi/http/struts_code_exec. Upon successful exploitation, a remote session was established on the target system, confirming the presence of a critical vulnerability.

3.7 Vulnerability Log

Vulnerability	CVSS Score	Description
Apache Struts RCE	9.8	Remote code execution via vulnerable Struts framework

3.8 Remediation and Verification

Recommended Remediation

- Update Apache Struts to the latest patched version.
- Remove vulnerable or unused web components.
- Implement a Web Application Firewall (WAF).
- Conduct regular vulnerability scanning and patch management.

Verification

After applying remediation measures (patching or disabling the vulnerable application), exploitation attempts were re-tested and found to be unsuccessful.

Task 4: Lateral Movement and Persistence

4.1 Objective

The objective of this task was to demonstrate **lateral movement** within a compromised network and establish **persistence** on a target system. This task highlights how attackers move between

systems after initial compromise and maintain long-term access using legitimate system mechanisms.

4.2 Tools Used

- Impacket (psexec.py)
- Covenant

4.3 Lab Environment

- **Attacker Machine:** Kali Linux
- **Victim 1:** Compromised Windows VM (Initial Access)
- **Victim 2:** Windows VM (Lateral Movement Target)
- **Network Configuration:** Isolated internal network

4.4 Methodology

The lateral movement and persistence process involved the following steps:

1. Valid administrative credentials were obtained from the initially compromised system.
2. Impacket's psexec.py was used to execute commands remotely on another internal host.
3. Successful lateral movement was confirmed by obtaining a remote shell on the target system.
4. Persistence was established using a scheduled task to maintain access after reboot.
5. The persistence mechanism was verified by querying scheduled tasks.

4.5 Lateral Movement (Pivoting)

4.5.1 Technique Used

- Remote command execution via SMB using PsExec

4.5.2 Pivot Summary

Access was initially obtained on the first compromised system. Using valid administrative credentials, Impacket's psexec.py was executed to authenticate and remotely execute commands on a second internal host. This allowed lateral movement within the network and demonstrated how attackers expand access after initial compromise.

4.6 Persistence Mechanism

Persistence was achieved by creating a scheduled task on the compromised host. This ensured that a predefined command or payload would execute automatically at scheduled intervals, maintaining attacker access even after system reboot.

4.6.1 Persistence Log

Technique	Tactic	Description	Notes
Scheduled Task	Persistence	Executes payload on a schedule	Runs payload daily

Task 5: Social Engineering Lab (Vishing & Pretexting)

5.1 Objective

The objective of this task was to simulate a **social engineering attack** using vishing (voice phishing) and pretexting techniques in a controlled lab environment. The task demonstrates how attackers gather phone-based OSINT, correlate identity information, and craft believable social-engineering scenarios to manipulate targets into revealing sensitive information.

5.2 Tools Used

- PhoneInfoga
- Maltego Community Edition
- Social-Engineer Toolkit (SET)

5.3 Lab Environment

- **Platform:** Kali Linux
- **Target Data:** Fictitious phone number and user identities
- **Environment:** Controlled and authorized lab setup

5.4 Methodology

The social engineering simulation was conducted in three phases:

1. **Phone-based OSINT collection** using PhoneInfoga.
2. **Relationship mapping** of phone numbers and user identities using Maltego.
3. **Vishing simulation** by crafting a realistic pretext and call script without making real calls.

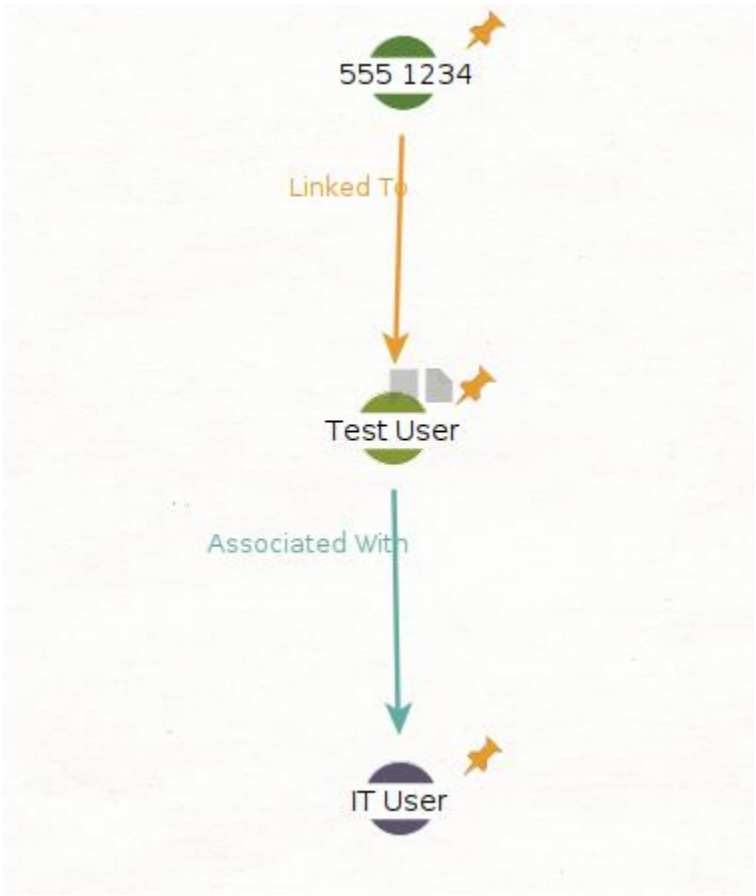
5.5 Intel Gathering (PhoneInfoga)

PhoneInfoga was used to analyze a test phone number to identify basic phone-related OSINT such as number format and potential linkage to a target identity. The collected information was treated as initial reconnaissance data for building a social-engineering scenario.

Screenshot showing PhoneInfoga scan results for the test phone number.

5.6 OSINT Correlation (Maltego)

Maltego was used to visually map relationships between the phone number, a test user identity, and an IT role. The resulting graph demonstrated how minimal OSINT can be expanded into a believable attack narrative suitable for vishing or pretexting.



Screenshot of Maltego graph linking the phone number, test user, and IT user role.

5.7 Intel Log

Target ID	Data Source	Information	Notes
TID001	PhoneInfoga	Phone: 555-1234	Linked to target

5.8 Vishing Simulation

5.8.1 Pretext Used

Impersonation of an internal IT support representative conducting a routine security verification.

5.8.2 Mock Vishing Script

“Hello, this is Alex from the IT support team. We are performing a routine account security verification due to suspicious login attempts. I need to confirm your username to ensure your account is protected.”

No real calls were placed. The script was tested through role-play to evaluate realism and effectiveness.

5.8.3 Vishing Scenario Summary

A vishing attack was simulated by impersonating an IT support representative using phone-based OSINT. By leveraging a believable pretext and role-based trust, the attacker could request sensitive information from the target. This exercise highlights how human trust and urgency are exploited in social engineering attacks.

5.9 Observations

- Phone-based OSINT can significantly improve the credibility of social-engineering attacks.
- Visual relationship mapping helps attackers build convincing pretexts.
- Even limited publicly available information can be weaponized against users.

Task 6: Exploit Development Basics

6.1 Objective

The objective of this task was to understand the fundamentals of **exploit development** by analyzing a vulnerable binary, identifying a **buffer overflow vulnerability**, and crafting a **proof-of-concept (PoC)** exploit in a controlled lab environment. The task focused on binary analysis, debugging, and safe exploitation concepts.

6.2 Tools Used

- GDB (GNU Debugger)
- radare2
- gcc
- Linux Terminal Utilities (strings)

6.3 Lab Environment

- **Operating System:** Kali Linux
- **Target:** Self-written vulnerable C program
- **Execution Environment:** Isolated local VM

6.4 Methodology

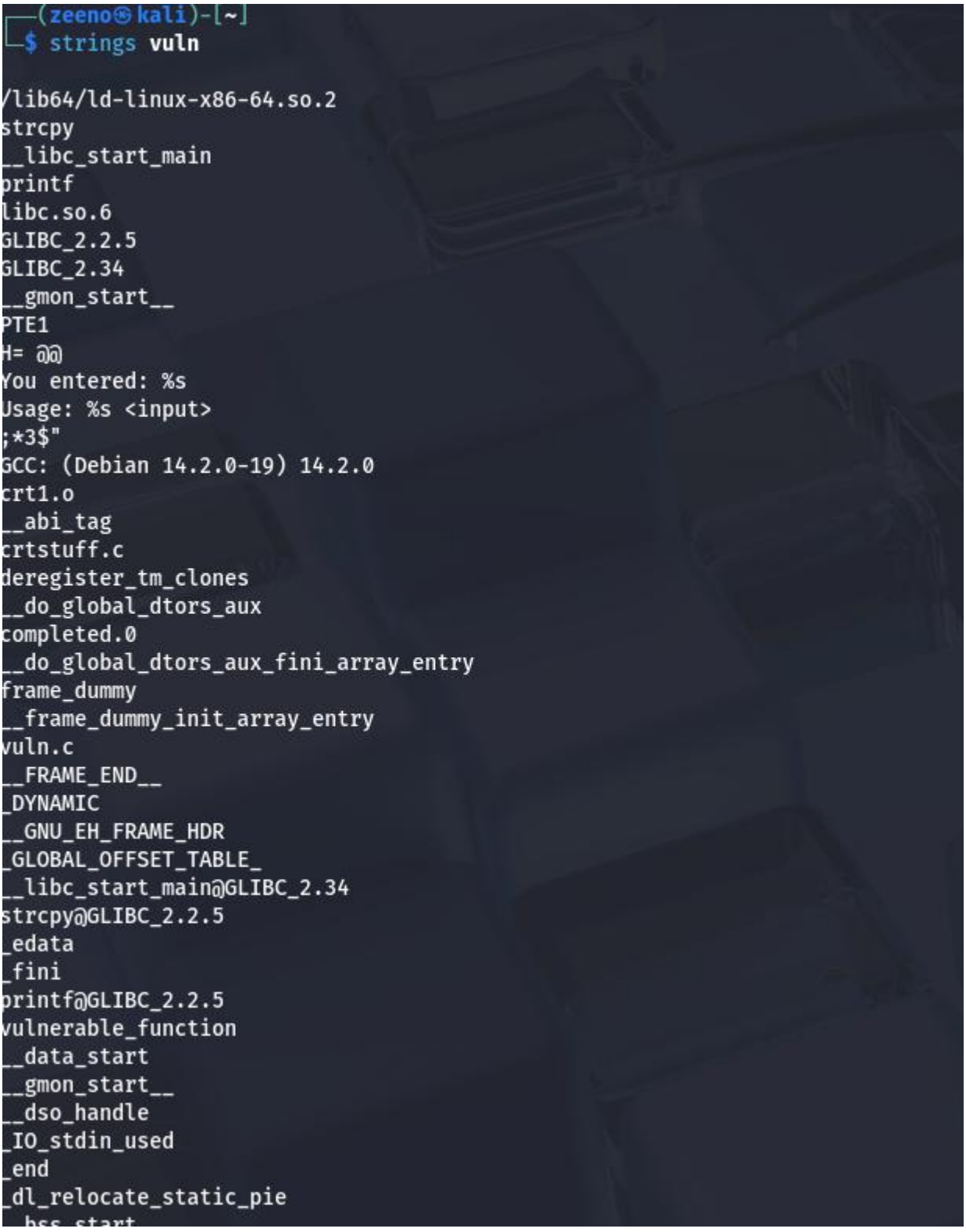
The exploit development exercise was conducted in the following phases:

1. Creation and compilation of a vulnerable C program.
2. Static analysis of the binary using strings.
3. Dynamic analysis and debugging using GDB.
4. Static reverse engineering using radare2.
5. Identification of a stack-based buffer overflow.
6. Development and testing of a safe proof-of-concept exploit.

6.5 Binary Analysis

6.5.1 Static Analysis (strings)

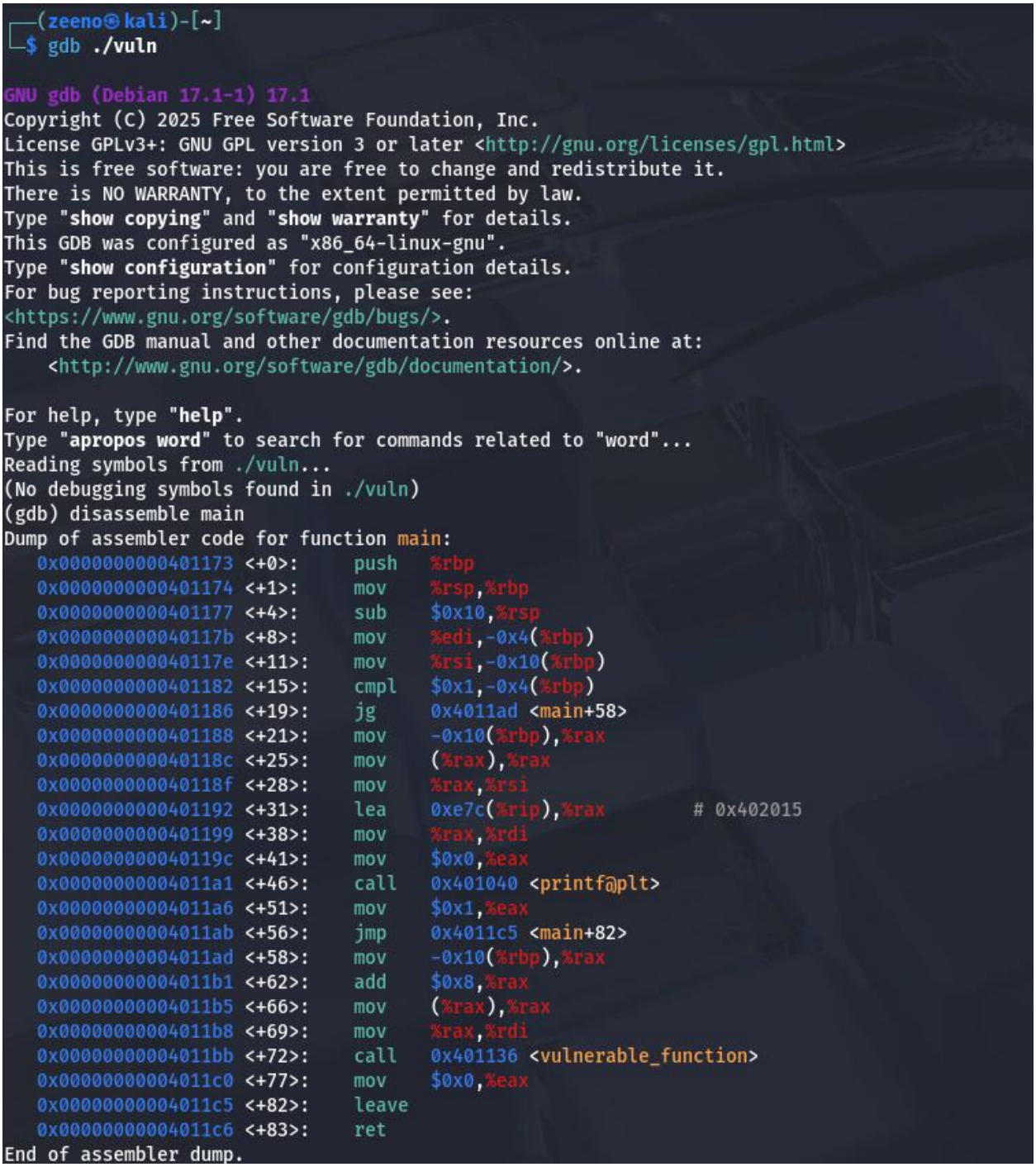
The strings utility was used to extract readable strings from the compiled binary. This revealed function names, program messages, and execution flow hints, providing initial insight into the binary structure and potential attack surface.



Screenshot showing strings output of the binary.

6.5.2 Dynamic Analysis (GDB)

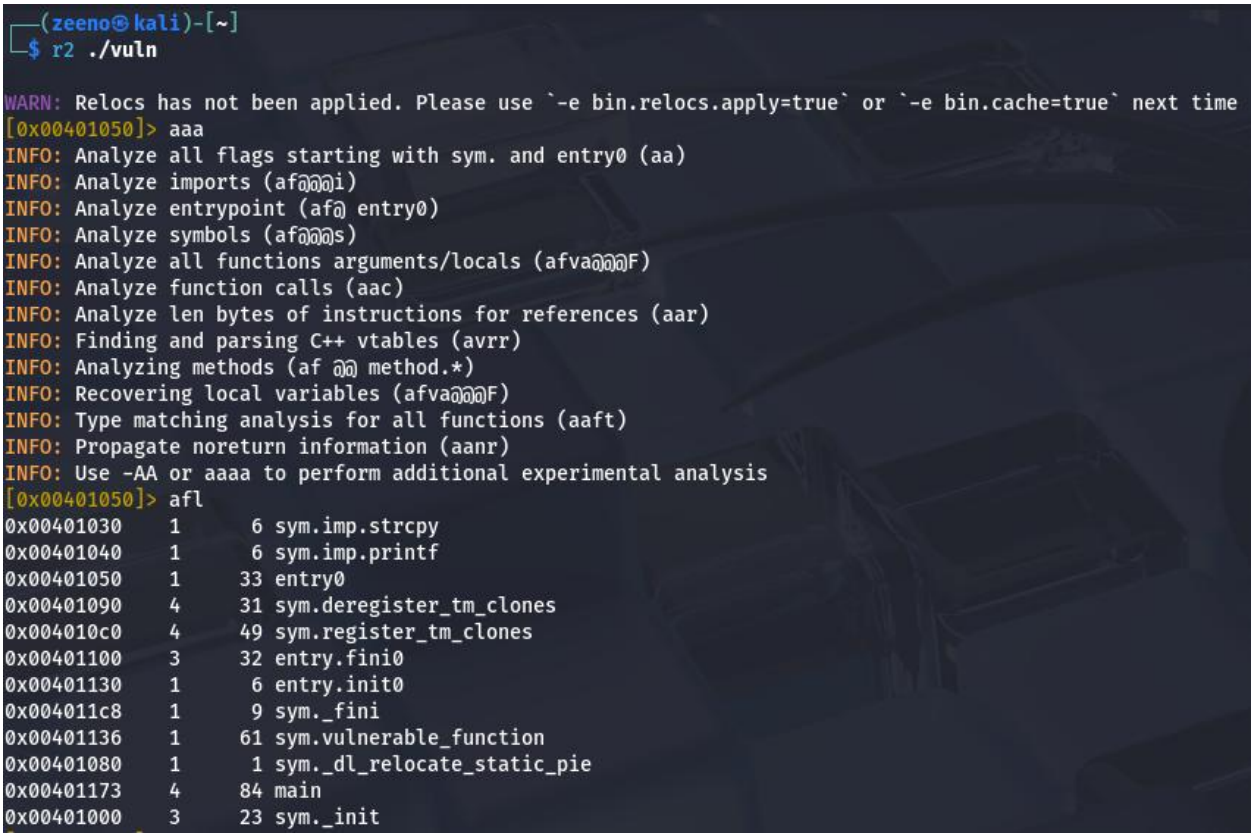
The binary was executed within GDB with debugging symbols enabled. Breakpoints were set at the vulnerable function to inspect stack frames and CPU registers during execution. This analysis confirmed unsafe memory handling caused by the use of unbounded input copying.



Screenshot showing breakpoint hit at vulnerable_function and register/stack inspection.

6.5.3 Static Reverse Engineering (radare2)

radare2 was used for static analysis of the binary. Automated analysis identified program functions, and the vulnerable function's assembly instructions were inspected. The analysis confirmed fixed-size stack buffer allocation and lack of bounds checking.



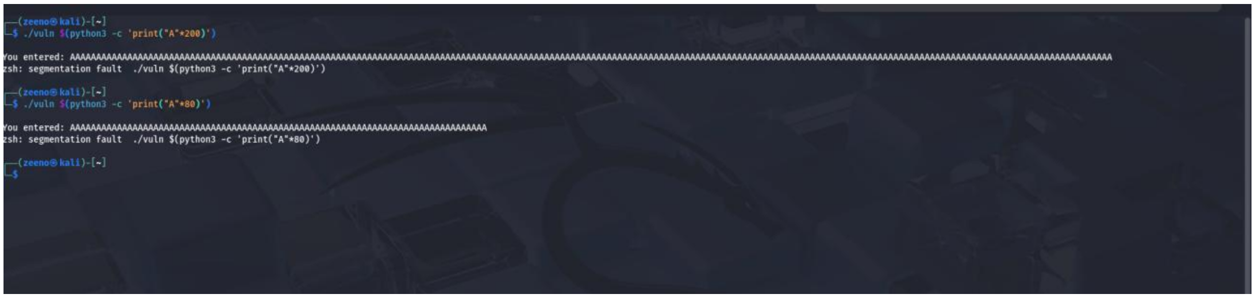
Screenshot showing radare2 function list and disassembly of the vulnerable function.

6.6 Key Findings

Analysis revealed unsafe usage of the strcpy function within a fixed-size stack buffer, allowing user-controlled input to overwrite adjacent memory. The binary lacked common security protections such as stack canaries and position-independent execution, making it vulnerable to stack-based buffer overflow exploitation.

6.7 Exploit Proof-of-Concept (PoC)

A proof-of-concept exploit was developed by supplying oversized input to the program, resulting in a controlled crash (segmentation fault). This demonstrated successful memory corruption without executing any malicious payloads or shellcode.



Screenshot showing program crash caused by buffer overflow input.

Only a proof-of-concept crash was demonstrated; no malicious payloads were executed.

Task 7: Post-Exploitation and Exfiltration

7.1 Objective

The objective of this task was to demonstrate **post-exploitation activities** after a successful compromise. This included extracting credential hashes from a Windows system and simulating data exfiltration using DNS tunneling techniques in a controlled lab environment. The task highlights the risks associated with inadequate endpoint protection and monitoring.

7.2 Tools Used

- Mimikatz
- Exfiltool / DNS utilities (nslookup)
- Wireshark (for verification)

7.3 Lab Environment

- **Attacker Machine:** Kali Linux
- **Victim Machine:** Windows Virtual Machine (Administrator access)
- **Network Configuration:** Isolated lab network (NAT / Host-only)

7.4 Methodology

The post-exploitation exercise was conducted in two main phases:

1. **Credential Dumping** – Extracting authentication hashes from the compromised Windows system.
2. **Data Exfiltration Simulation** – Demonstrating how attackers can exfiltrate data using DNS-based techniques.

7.5 Credential Dumping (Mimikatz)

Mimikatz was executed on the compromised Windows VM with administrative privileges. Debug privileges were enabled to allow access to credential material stored in memory. Credential dumping was performed to extract NTLM hashes of logged-in users.

This demonstrates how attackers can reuse hashes for lateral movement or privilege escalation after initial compromise.

7.5.1 Credential Log

Hash Type	Username	Hash Value
NTLM	Administrator	aad3b435b514...

7.6 Exfiltration Simulation (DNS Tunneling)

To simulate data exfiltration, a mock data file was created on the compromised Windows VM. Instead of real data theft, DNS-based exfiltration was demonstrated conceptually by generating outbound DNS queries that represent how encoded data could be transferred through DNS traffic.

Outbound DNS requests were generated from the victim system and verified through network monitoring.

7.7 Verification

DNS-based exfiltration was verified by observing outbound DNS traffic originating from the compromised Windows VM. This confirms that DNS can be abused as a covert channel for data exfiltration and highlights the need for DNS traffic inspection.

7.8 Observations

- Credential dumping provides attackers with reusable authentication material.
- NTLM hashes can enable pass-the-hash attacks and lateral movement.
- DNS traffic is often trusted and insufficiently monitored, making it an effective exfiltration channel.
- Post-exploitation activities significantly increase the impact of an initial compromise.

8 Conclusion

This practical application successfully demonstrated the complete red team attack lifecycle through a series of controlled and ethical hands-on exercises. By performing reconnaissance, phishing-based initial access, vulnerability exploitation, lateral movement, persistence, social engineering, exploit development, and post-exploitation activities, the report highlighted how real-world attackers identify and exploit security weaknesses.

The exercises revealed common organizational gaps such as exposed services, weak credentials, insecure configurations, insufficient monitoring, and lack of user awareness. The findings emphasize the importance of implementing layered defensive strategies, including timely patch management, strong credential protection, network segmentation, endpoint hardening, DNS traffic inspection, and continuous security awareness training.

Overall, this work reinforces the value of proactive security assessments and red team operations in improving an organization's detection, response, and resilience against advanced cyber threats.

9 References

- MITRE ATT&CK Framework – https://attack.mitre.org
- OWASP Documentation – https://owasp.org
- Nmap Documentation – https://nmap.org/book/
- Metasploit Framework Documentation – https://docs.metasploit.com
- Shodan – https://www.shodan.io
- Maltego Documentation – https://www.maltego.com
- Recon-ng Documentation – https://github.com/lanmaster53/recon-ng
- Gophish Documentation – https://getgophish.com
- Mimikatz Project – https://github.com/gentilkiwi/mimikatz
- radare2 Documentation – https://rada.re/n/