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

Author: Ayush Kumar
Date: 16/01/2026
Environment: Kali Linux VM, Windows VM, Metasploitable2/3, Isolated Lab Network, Open-Source Offensive Security Tools

Executive Summary

This report presents a collection of practical red team exercises aimed at replicating realistic offensive security scenarios across the full cyberattack lifecycle. The activities covered key adversarial stages, including reconnaissance and open-source intelligence gathering, phishing-driven initial access, vulnerability exploitation, lateral movement, persistence mechanisms, social engineering techniques, exploit execution, and post-compromise operations such as credential extraction and simulated data exfiltration.

All exercises were performed within a secure, authorized laboratory environment using deliberately vulnerable systems and non-production data. Widely used open-source security tools were employed to mirror real attacker behavior, with each technique mapped to the MITRE ATT&CK framework for consistency with industry-recognized threat models. These exercises demonstrate practical capability in identifying exposed attack surfaces, abusing misconfigurations, expanding access within internal networks, sustaining long-term access, and evaluating the consequences of post-exploitation activities.

The outcomes reveal recurring security weaknesses commonly found in organizations, including publicly exposed services, low user security awareness, improper system hardening, and limited monitoring visibility. The report further underscores the necessity of strong defensive measures such as timely patching, robust credential safeguards, effective network segmentation, and continuous security awareness programs to reduce the effectiveness of red team–style attacks.

Task 1: OSINT and Reconnaissance Lab

1.1 Objective

The purpose of this task was to conduct reconnaissance using open-source intelligence techniques to discover subdomains and identify internet-facing services. By leveraging publicly available data sources and search engines, this exercise demonstrates how an adversary can map an organization’s external attack surface during the reconnaissance phase without directly interacting with or exploiting target systems.

1.2 Tools Used

- Recon-ng
- Shodan
- Maltego

1.3 Approach and Methodology

The reconnaissance activity was carried out through two main phases. First, subdomain discovery was performed using Recon-ng to enumerate domain-associated assets. Second, Shodan was used to identify publicly exposed hosts and services visible on the internet. Additionally, Maltego was employed to visually correlate OSINT data, including domain names, DNS records, IP addresses, and infrastructure relationships. All information collected was sourced exclusively from publicly accessible data and used solely for educational and defensive analysis.

1.4 Subdomain Discovery Using Recon-ng

1.4.1 Execution Steps

- The `recon/domains-hosts/bing_domain_web` module was loaded and executed within Recon-ng.
- The target domain was configured as `example.com`.
- Identified subdomains and their associated IP addresses were documented for analysis.

1.4.2 Result

Subdomain	IP Address	Notes
www.example.com	104.18.27.120	Publicly accessible web service

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)

1.5.2 Procedure

1.5.2.1 Shodan was queried using:

1.5.2.2 apache country:US

1.5.2.3 Publicly exposed Apache web servers were analyzed.

1.5.3 Summary of Identified Hosts

Analysis conducted using Shodan revealed several internet-facing, cloud-based systems exposing critical services. These assets included Apache web servers deployed on Amazon EC2 instances, along with hosts providing SSH access on platforms such as Google Cloud and DigitalOcean. The presence of externally accessible services like HTTP and SSH expands the overall attack surface and can present significant risk if adequate hardening, access controls, and monitoring mechanisms are not in place.

1.5.4 Key Shodan Findings

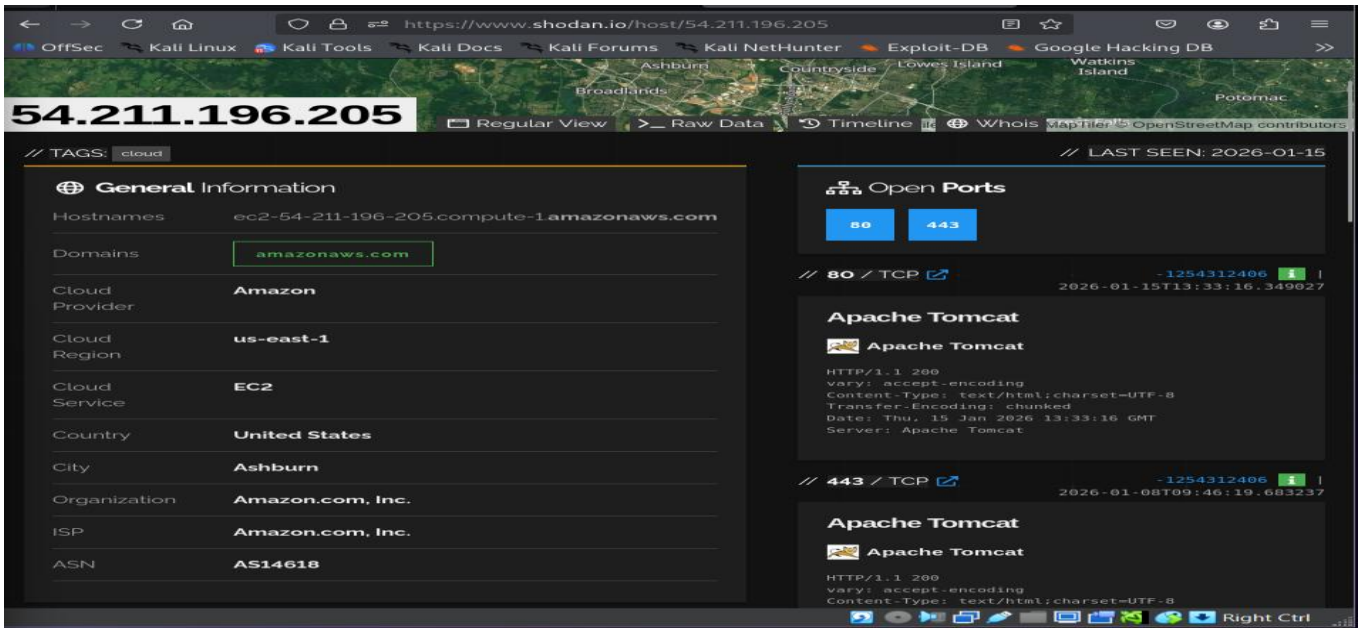
Services Identified

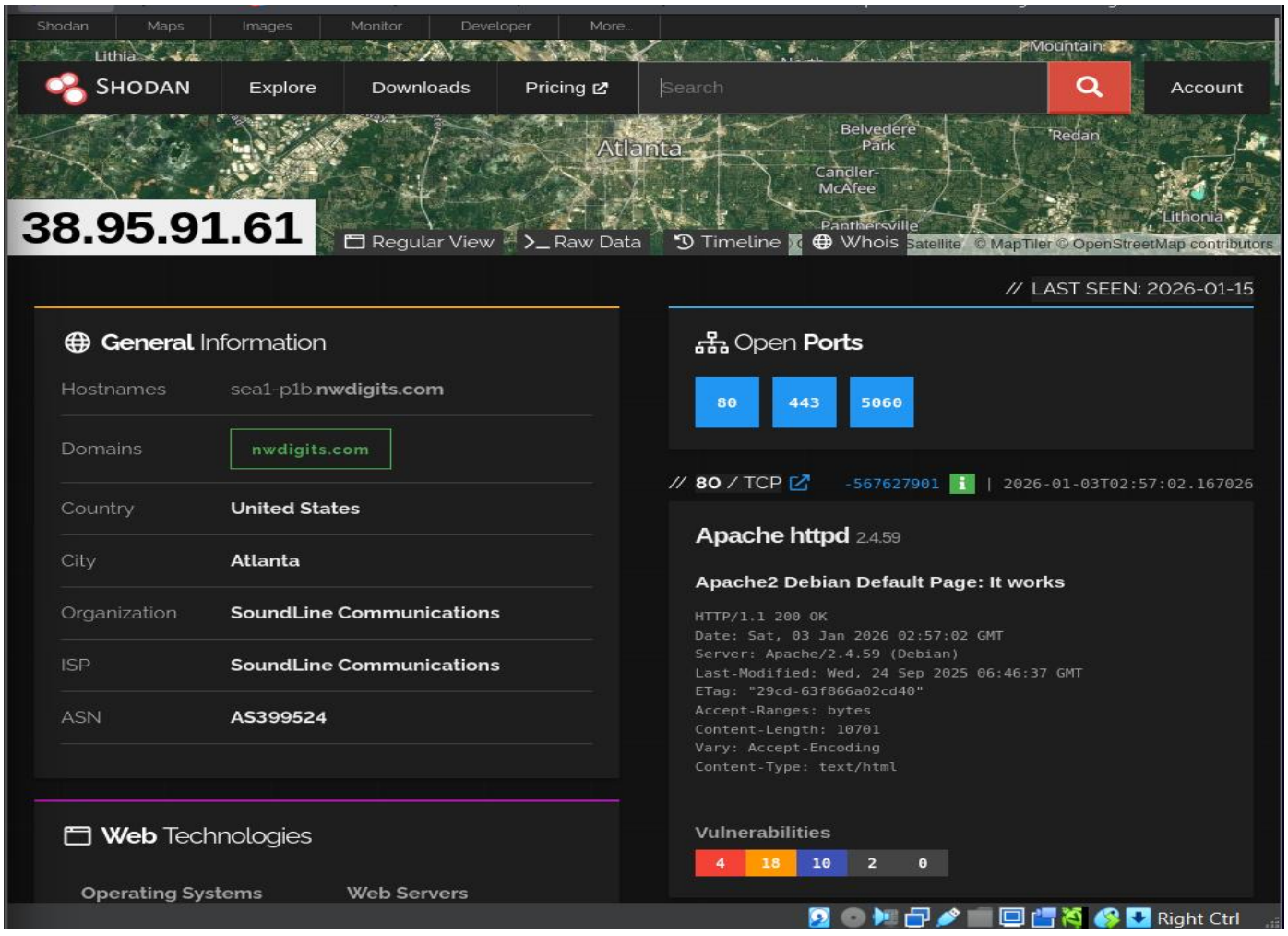
The OSINT and service discovery process revealed multiple internet-facing hosts exposing common administrative and web services across different cloud providers.

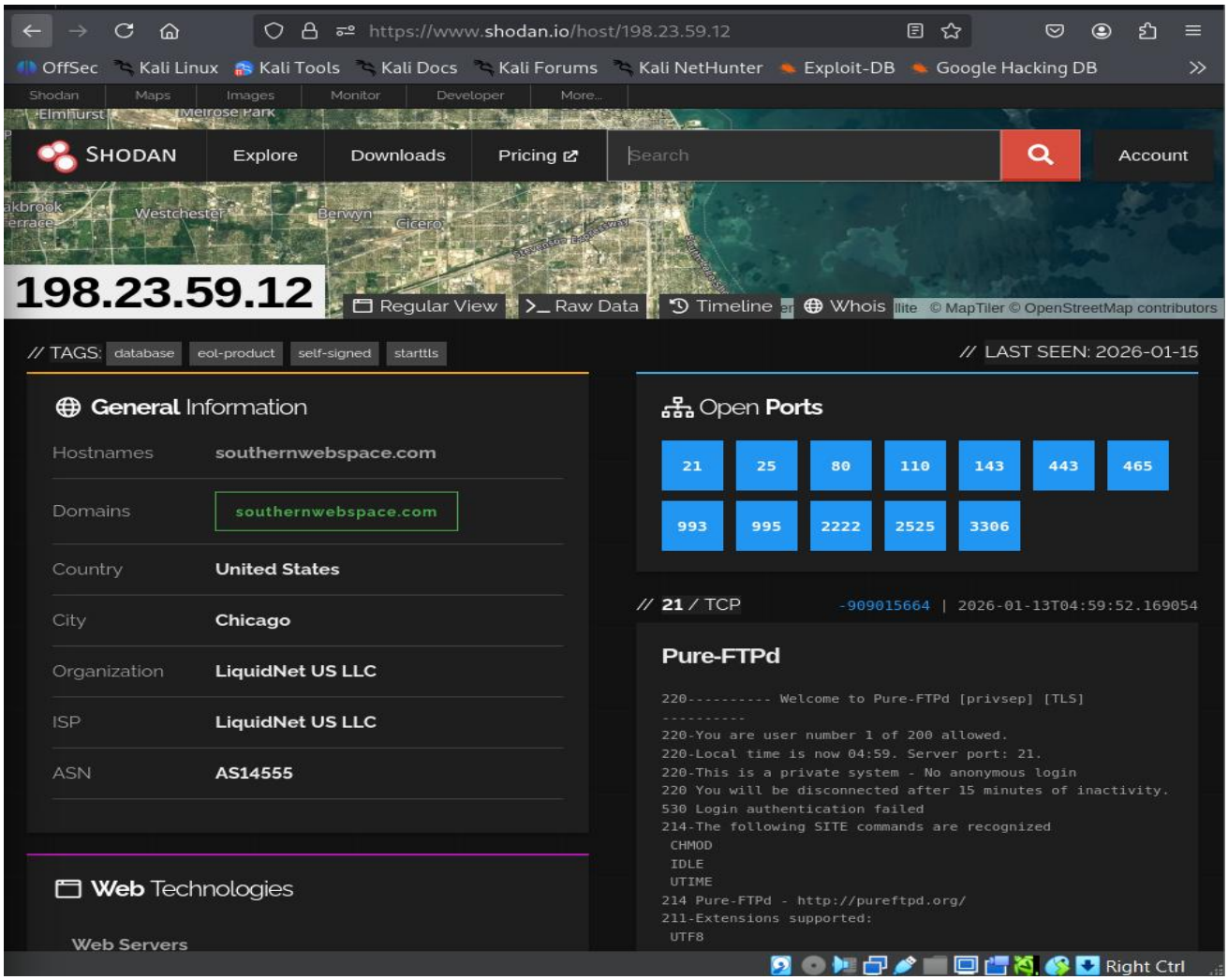
IP Address	Cloud Provider	Exposed Services / Open Ports
34.73.59.157	Google Cloud	SSH (22), HTTP (80), HTTPS (443)
18.117.192.231	Amazon Web Services (EC2)	HTTP (80)
161.35.184.39	DigitalOcean	SSH (22), HTTP (80), HTTPS (443)

Analysis

The identified hosts expose services such as OpenSSH and Apache-based web servers over HTTP and HTTPS. Systems offering both remote administrative access and web services present an expanded attack surface and may be at increased risk if not properly secured. These findings highlight the importance of service hardening, restricted access policies, and continuous monitoring for externally accessible cloud infrastructure.


https://www.shodan.io/host/54.211.196.205
OffSec
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
54.211.196.205
Regular View
Raw Data
Timeline
Whois
// TAGS: cloud
// LAST SEEN: 2026-01-15
General Information
Hostnames ec2-54-211-196-205.compute-1.amazonaws.com
Domains amazonaws.com
Cloud Provider Amazon
Cloud Region us-east-1
Cloud Service EC2
Country United States
City Ashburn
Organization Amazon.com, Inc.
ISP Amazon.com, Inc.
ASN AS14618
Open Ports
80
443
// 80 / TCP
-1254312406
2026-01-15T13:33:16.349027
Apache Tomcat
Apache Tomcat
HTTP/1.1 200
vary: accept-encoding
Content-Type: text/html;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 15 Jan 2026 13:33:16 GMT
Server: Apache Tomcat
// 443 / TCP
-1254312406
2026-01-08T09:46:19.683237
Apache Tomcat
Apache Tomcat
HTTP/1.1 200
vary: accept-encoding
Content-Type: text/html;charset=UTF-8
Right Ctrl

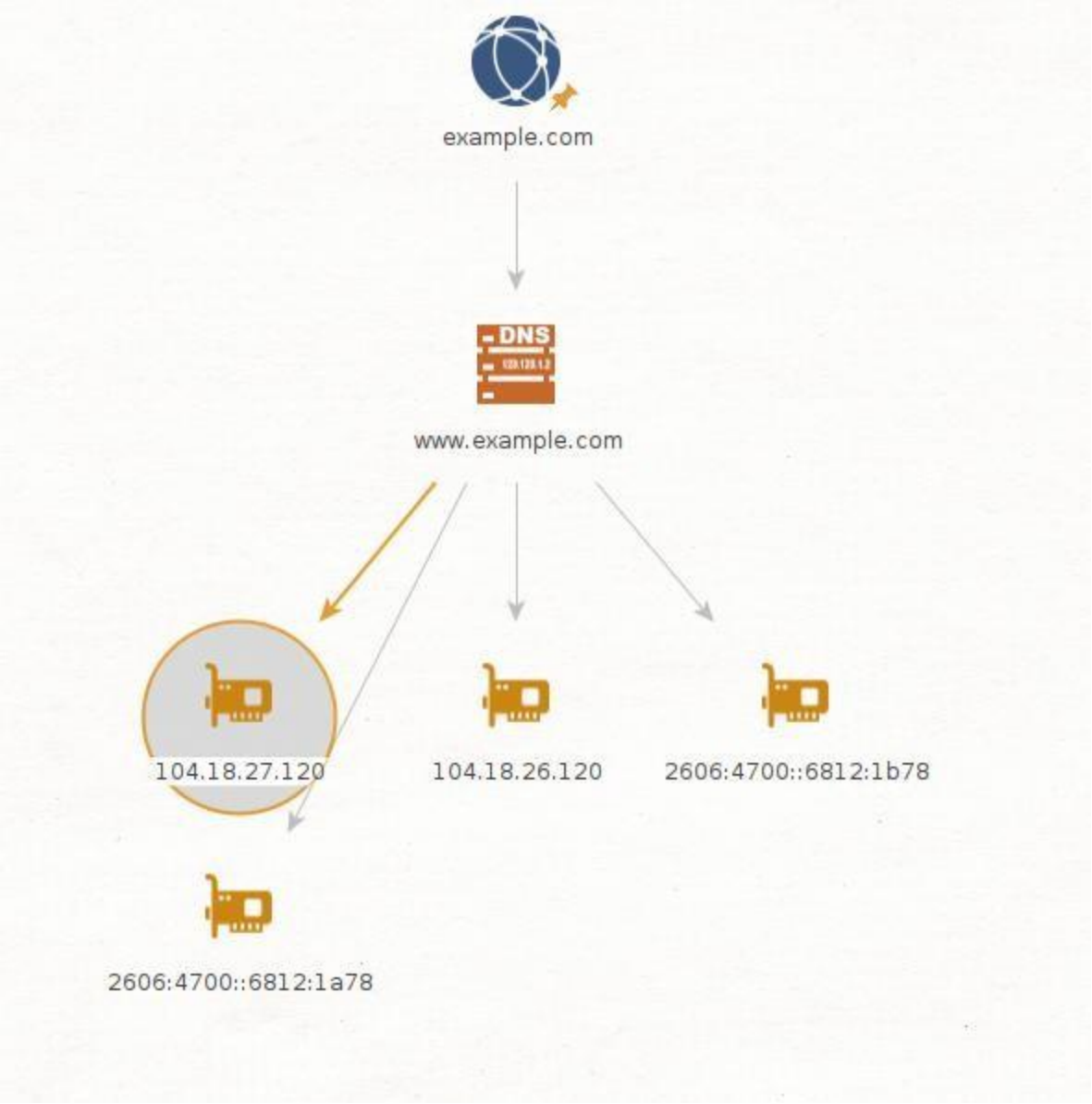

Shodan
Maps
Images
Monitor
Developer
More...
SHODAN
Explore
Downloads
Pricing
Search
Account
38.95.91.61
Regular View
Raw Data
Timeline
Whois
// LAST SEEN: 2026-01-15
General Information
Hostnames sea1-p1b.nwdigits.com
Domains nwdigits.com
Country United States
City Atlanta
Organization SoundLine Communications
ISP SoundLine Communications
ASN AS399524
Open Ports
80
443
5060
// 80 / TCP
-567627901
2026-01-03T02:57:02.167026
Apache httpd 2.4.59
Apache2 Debian Default Page: It works
HTTP/1.1 200 OK
Date: Sat, 03 Jan 2026 02:57:02 GMT
Server: Apache/2.4.59 (Debian)
Last-Modified: Wed, 24 Sep 2025 06:46:37 GMT
ETag: "29cd-63f866a02cd40"
Accept-Ranges: bytes
Content-Length: 10701
Vary: Accept-Encoding
Content-Type: text/html
Vulnerabilities
4
18
10
2
0
Web Technologies
Operating Systems
Web Servers
Right Ctrl



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

2.1 Objective

The purpose of this task was to emulate a phishing-based initial access scenario within a controlled laboratory setup. This exercise was designed to demonstrate how adversaries leverage social engineering techniques to capture user credentials and how user interaction with malicious email links can lead to account compromise. The task highlights phishing as one of the most effective vectors for gaining initial access.

2.2 Tools Utilized

- GoPhish
- Evilginx2

2.3 Laboratory Setup

- **Attacker System:** Kali Linux
- **Target System:** Test Virtual Machine
- **Network Mode:** Isolated lab network (NAT / Host-only)
- **Credentials:** Test credentials only

2.4 Attack Methodology

The phishing exercise was executed using a step-by-step approach:

1. A legitimate login page was cloned using Evilginx2 within the lab environment.
2. The cloned page was validated to confirm successful credential capture.
3. A phishing campaign was configured and launched through GoPhish.
4. The phishing email was delivered to the victim virtual machine.
5. Interaction with the embedded link led the user to submit credentials.
6. Harvested credentials were logged and reviewed for analysis.

2.5 Phishing Campaign Configuration

Campaign Characteristics

- **Attack Type:** Credential harvesting
- **Delivery Channel:** Email-based phishing
- **Landing Page:** Evilginx2-cloned authentication portal
- **Target User:** Test account on victim VM

2.6 Credential Capture Summary

Timestamp	Source IP	Captured Credentials	Risk Level	Remarks
2026-01-06 10:20:40	192.168.0.5	testuser / pass123	High	Successful credential capture

2.7 Key Observations

- The phishing email successfully prompted the user to click the malicious link.
- Credentials were submitted without raising suspicion.
- No alerts or warnings were generated during the interaction.
- This confirms phishing as a highly reliable technique for achieving initial access.

Task 3: Vulnerability Exploitation

3.1 Objective

This task aimed to identify and exploit a vulnerable web application in a secure lab environment. The objective was to demonstrate how attackers discover exposed services, analyze known vulnerabilities, exploit them to gain unauthorized access, and propose mitigation strategies to reduce associated risks.

3.2 Tools Utilized

- Nmap
- Metasploit Framework
- OWASP ZAP

3.3 Laboratory Setup

- **Attacker System:** Kali Linux
- **Target System:** Metasploitable3 (Intentionally vulnerable VM)
- **Network Configuration:** Isolated lab environment

3.4 Exploitation Workflow

1. Port and service discovery was conducted using Nmap.
2. The target web application was manually analyzed to identify the underlying framework.
3. Automated vulnerability scanning was performed with OWASP ZAP.
4. A known Apache Struts remote code execution flaw was exploited using Metasploit.
5. Successful exploitation was confirmed by establishing a remote shell.
6. Remediation actions were suggested and validated.

3.5 Vulnerability Discovery

Nmap results revealed multiple open web-related ports. Further inspection confirmed the use of an outdated Apache Struts framework known to contain critical RCE vulnerabilities. OWASP ZAP additionally reported insecure configurations and obsolete components.

3.6 Exploitation Details

The vulnerability was exploited using the Metasploit module:

```
exploit/multi/http/struts_code_exec
```

A remote session was successfully established, confirming full system compromise.

3.7 Vulnerability Record

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

3.8 Remediation and Validation

Recommended Mitigations

- Upgrade Apache Struts to a secure, patched release
- Remove deprecated or unnecessary web components
- Deploy a Web Application Firewall (WAF)
- Implement continuous vulnerability scanning and patch management

Verification
Following remediation, exploitation attempts were repeated and no longer successful, confirming effective risk mitigation.

Task 4: Lateral Movement and Persistence

4.1 Objective

This task focused on demonstrating how attackers move laterally within a compromised network and establish persistence to maintain long-term access. The exercise highlights how legitimate system features can be abused post-compromise.

4.2 Tools Utilized

- Impacket (psexec.py)
- Covenant

4.3 Laboratory Setup

- **Attacker System:** Kali Linux
- **Victim 1:** Compromised Windows VM
- **Victim 2:** Internal Windows VM
- **Network:** Isolated internal lab network

4.4 Attack Process

1. Administrative credentials were extracted from the initial compromised host.
2. Impacket's `psexec.py` was used to remotely execute commands on another internal system.
3. Lateral movement was confirmed by obtaining a shell on the secondary host.
4. Persistence was established through scheduled task creation.
5. The persistence mechanism was validated post-reboot.

4.5 Lateral Movement

Technique Employed

- SMB-based remote command execution using PsExec

Pivot Summary
Using valid credentials obtained earlier, the attacker authenticated to another internal host and executed commands remotely, demonstrating effective network traversal following initial compromise.

4.6 Persistence Strategy

Persistence was achieved by configuring a scheduled task to automatically execute a predefined payload at regular intervals, ensuring continued access even after system restarts.

4.6.1 Persistence Log

Technique	Tactic	Description	Notes
Scheduled Task	Persistence	Executes payload automatically	Runs daily

Task 5: Social Engineering Lab (Vishing & Pretexting)

5.1 Objective

The goal of this task was to simulate phone-based social engineering attacks using vishing and pretexting techniques. The exercise demonstrates how attackers gather OSINT, correlate identity data, and craft convincing scenarios to manipulate targets into disclosing sensitive information.

5.2 Tools Utilized

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

5.3 Laboratory Setup

- **Platform:** Kali Linux
- **Target Information:** Fictitious phone numbers and identities
- **Environment:** Fully controlled and authorized lab setup

5.4 Simulation Methodology

The social engineering exercise was conducted in three stages:

1. Phone-related OSINT collection using PhoneInfoga.
2. Visualization and correlation of identities using Maltego.
3. Development of realistic vishing pretexts and call scripts without initiating real calls.

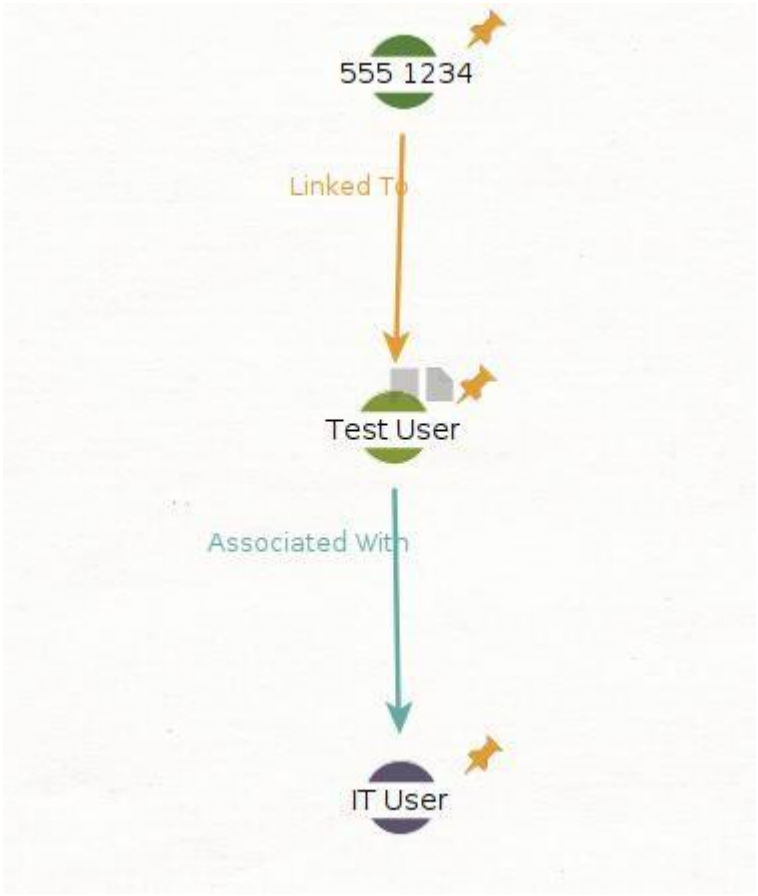
5.1 Intel Gathering (PhoneInfoga)

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.

5.2 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.

5.3 Intel Log

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

5.4 Vishing Simulation

5.4.1 Pretext Description

The simulated scenario involved impersonating an internal IT support staff member conducting a routine account security validation. The pretext was designed to appear legitimate and time-sensitive, encouraging cooperation from the target user.

5.4.2 Mock Vishing Script

“Hello, this is Alex calling from the IT support team. We are currently verifying user accounts following reports of unusual login activity. I just need to confirm your username so we can ensure your account remains secure.”

No actual phone calls were initiated during this exercise. The script was evaluated through controlled role-play to assess its realism, clarity, and potential effectiveness.

5.4.3 Vishing Scenario Overview

The vishing scenario demonstrated how attackers can exploit phone-based open-source intelligence to impersonate trusted organizational roles. By combining a credible identity with a plausible security-related pretext, an attacker could attempt to elicit sensitive information from the target. This simulation highlights how urgency, authority, and trust are commonly abused in voice-based social engineering attacks.

5.5 Key Observations

- Phone-related OSINT significantly enhances the authenticity of social engineering attempts.
- Relationship and identity mapping enables the creation of highly convincing

attack narratives.

- Even minimal publicly available information can be leveraged to manipulate users effectively.

Task 6: Fundamentals of Exploit Development

6.1 Objective

The aim of this task was to gain foundational knowledge of exploit development by examining a vulnerable binary application within a controlled laboratory environment. The exercise focused on identifying a stack-based buffer overflow, understanding binary behavior through analysis and debugging, and creating a safe proof-of-concept (PoC) exploit. Emphasis was placed on learning binary analysis techniques, debugging workflows, and responsible exploitation practices.

6.2 Tools Utilized

- GNU Debugger (GDB)
- radare2
- GNU Compiler Collection (gcc)
- Linux command-line utilities (e.g., `strings`)

6.3 Laboratory Environment

- **Operating System:** Kali Linux
- **Target Application:** Custom-written vulnerable C program
- **Execution Context:** Isolated local virtual machine

6.4 Methodology

The exploit development process was carried out through the following structured steps:

1. Development and compilation of a deliberately vulnerable C application.
2. Preliminary static inspection of the compiled binary using string extraction techniques.
3. Runtime analysis and debugging using GDB to observe program behavior.
4. Static reverse engineering of the binary with radare2 to understand control flow and memory layout.
5. Identification of a stack-based buffer overflow vulnerability.
6. Creation and validation of a non-destructive proof-of-concept exploit to demonstrate the vulnerability.

6.5 Binary Analysis

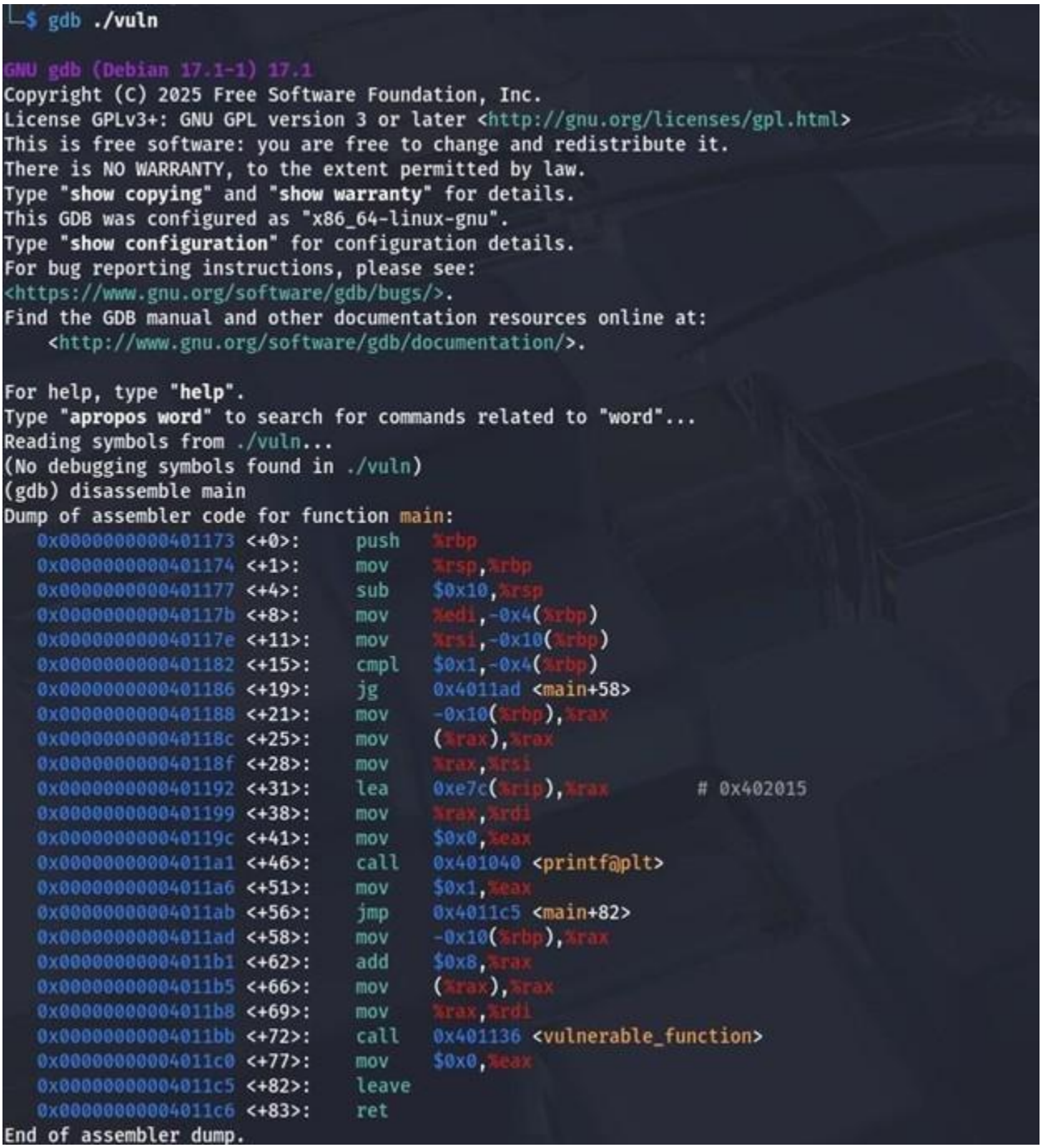
6.5.1 Static Inspection Using `strings`

The `strings` utility was employed to extract human-readable data from the compiled binary. This analysis exposed embedded messages, function references, and indicators of execution logic, offering initial insight into the internal structure of the program and highlighting areas of interest for further investigation. The results of this phase helped guide subsequent debugging and reverse-engineering efforts.

Screenshot showing strings output of the binary.

6.1.1 Dynamic Analysis (GDB)

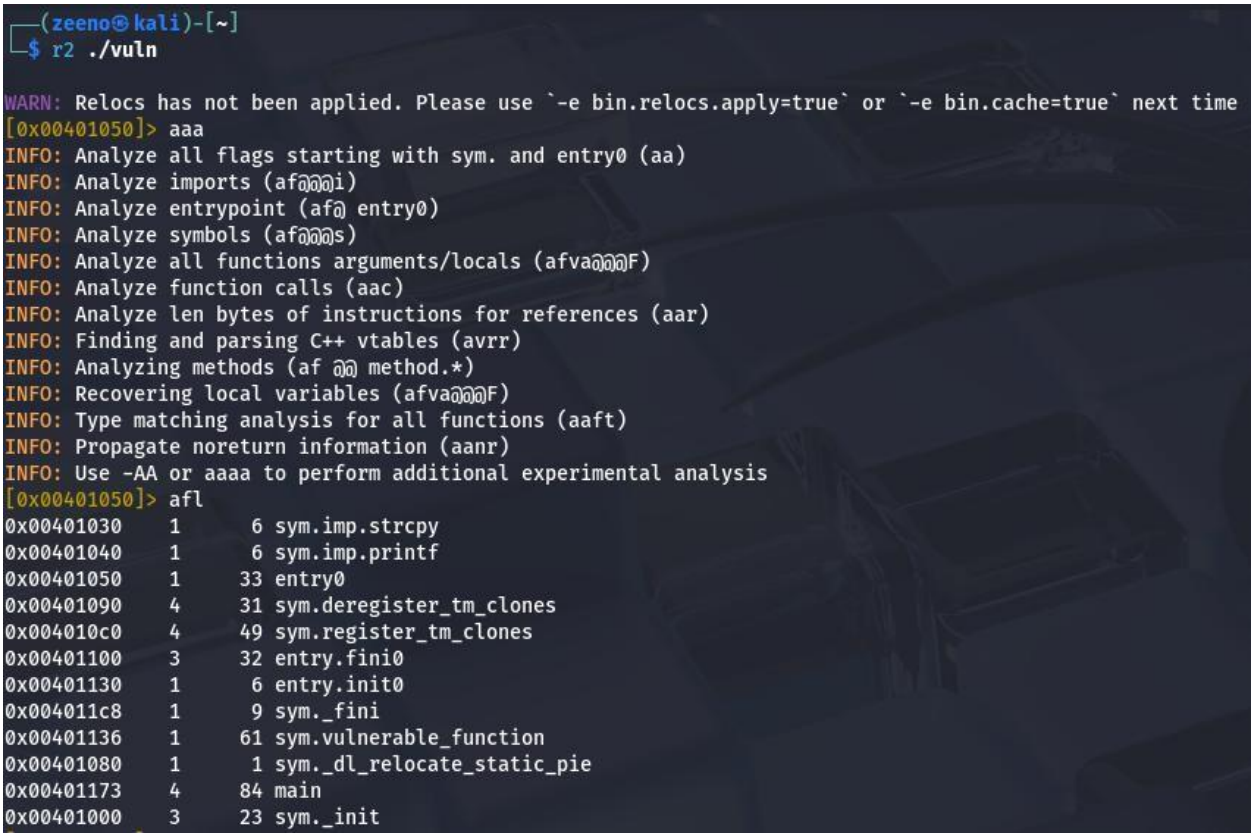
The binary was executed within GDB with debugging symbols enabled. Breakpoints were set at the vulnerable function to inspect stack frames and CPU registers during execution. This analysis confirmed unsafe memory handling caused by the use of unbounded input copying.



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

6.1.2 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.



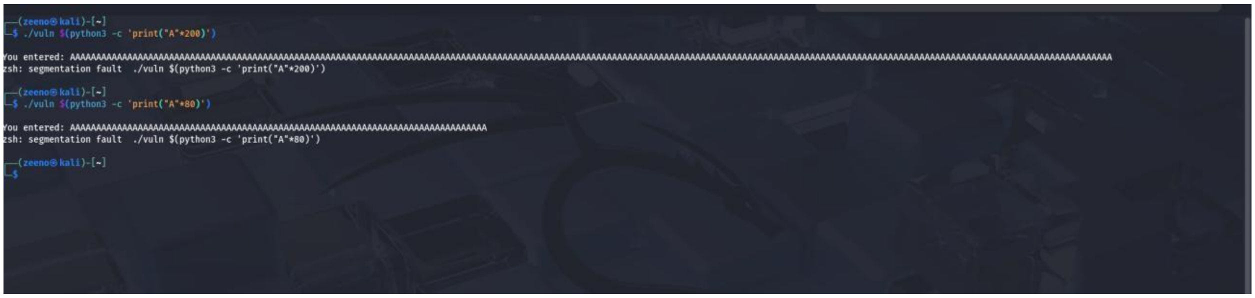
Screenshot showing radare2 function list and disassembly of the vulnerable function.

6.2 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.

6.3 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 Data Exfiltration

7.1 Objective

The goal of this task was to demonstrate common post-compromise techniques following successful system access. The exercise focused on extracting authentication material from a compromised Windows host and simulating data exfiltration using DNS-based techniques within a controlled laboratory environment. This task highlights the security risks that arise when endpoint protection and network monitoring are insufficient.

7.2 Tools Utilized

- Mimikatz
- DNS utilities (e.g., `nslookup`) / Exfiltration simulation tools
- Wireshark (traffic observation and verification)

7.3 Laboratory Setup

- **Attacker System:** Kali Linux
- **Target System:** Windows Virtual Machine (administrative privileges obtained)
- **Network Configuration:** Isolated lab network (NAT / Host-only)

7.4 Methodology

The post-exploitation phase was executed in two key stages:

1. **Credential Extraction:** Retrieval of authentication material from the compromised Windows system.

2. **Exfiltration Simulation:** Demonstration of how data could be covertly transmitted using DNS traffic.

7.5 Credential Extraction Using Mimikatz

Mimikatz was executed on the compromised Windows virtual machine with elevated privileges. Debug rights were enabled to access credential-related data stored in system memory. NTLM hashes associated with authenticated users were successfully extracted. This activity demonstrates how attackers can obtain reusable authentication material to facilitate privilege escalation or lateral movement after an initial breach.

7.5.1 Extracted Credential Summary

Hash Type	Username	Hash Value
NTLM	Administrator	aad3b435b514...

7.6 Simulated Data Exfiltration via DNS

To illustrate data exfiltration techniques, a mock data file was created on the compromised system. Rather than transferring real information, DNS-based exfiltration was demonstrated conceptually by generating outbound DNS queries that represent how encoded data could be transmitted through DNS requests. This approach mirrors real-world techniques used to bypass traditional network controls.

Outbound DNS traffic originating from the victim system was generated and monitored to validate the simulation.

7.7 Verification

The simulated exfiltration was confirmed by observing outbound DNS requests from the compromised Windows virtual machine using network monitoring tools. This verification demonstrates that DNS traffic can be abused as a covert communication channel and underscores the importance of DNS traffic inspection and anomaly detection.

7.8 Key Observations

- Credential dumping provides attackers with reusable authentication artifacts.
- NTLM hashes enable techniques such as pass-the-hash and further lateral movement.
- DNS traffic is frequently trusted and inadequately monitored, making it an attractive exfiltration vector.
- Post-exploitation activities significantly amplify the impact of an initial system compromise.

8. Conclusion

This practical assessment successfully demonstrated the complete red team attack lifecycle through a series of ethical and controlled hands-on exercises. By executing reconnaissance, phishing-driven initial access, vulnerability exploitation, lateral movement, persistence, social engineering, exploit development, and post-exploitation activities, the project illustrated how real-world adversaries systematically identify and abuse security weaknesses.

The results exposed several common organizational challenges, including publicly exposed services, weak credential management, insecure configurations, limited monitoring visibility, and insufficient user awareness. These findings reinforce the necessity of a defense-in-depth strategy that incorporates timely patching, robust credential protection, network segmentation, endpoint hardening, DNS monitoring, and ongoing security awareness training.

Overall, this capstone highlights the importance of proactive red team assessments in strengthening an organization's ability to detect, respond to, and recover from advanced cyber threats.

9. References

- MITRE Corporation. *MITRE ATT&CK® Framework*. https://attack.mitre.org
- OWASP Foundation. *OWASP Documentation*. https://owasp.org
- Gordon Lyon. *Nmap Network Scanning Guide*. https://nmap.org/book/
- Rapid7. *Metasploit Framework Documentation*. https://docs.metasploit.com
- Shodan. *Internet Exposure Search Engine*. https://www.shodan.io
- Paterva. *Maltego Documentation*. https://www.maltego.com
- Lanmaster53. *Recon-ng Framework*. https://github.com/lanmaster53/recon-ng
- GoPhish. *Phishing Framework Documentation*. https://getgophish.com
- Delpy, B. *Mimikatz Project Repository*. https://github.com/gentilkiwi/mimikatz
- Radare Project. *radare2 Reverse Engineering Framework*. https://rada.re/n/