Incident Response Handbook:

Project: Phishing → Ransomware Simulation (MailHog + GoPhish)

Author:
-Abdulrahman Al-Sayed Abdulaziz
-Khaled Mohamed Mohamed
-Kareem Ibrahim Mahmoud Omar
-Abdul Rahman Ali Abu Al-Maati
-Mohammed Ahmed Issa
-Mazen Mohamed Fathi

Table of Contents

---

1. Project Overview

Objective: This project demonstrates a controlled phishing campaign that delivers a benign, mock ransomware payload. The educational goals are:

- Show how phishing can deliver malicious payloads.
- Demonstrate how an unsuspecting user can be tricked into executing a payload.
- Gather campaign metrics (open, click rates) using GoPhish.
- Teach detection, mitigation, and incident response best practices.

Scope: Local/lab environment using MailHog as a local SMTP capture server and GoPhish as the phishing framework. The `ticket.exe` payload is a safe, non-destructive simulation that shows ransom-note behavior without encrypting real user data.

---

3. Environment & System Requirements

Minimum recommended: - CPU: 2 cores - RAM: 4 GB (2 GB may suffice for very small tests) - Disk: 10 GB free - OS: Windows 10/11, Ubuntu 20.04+, or macOS (latest) - Network: Localhost / isolated lab network

Software: - GoPhish (latest release) - MailHog (latest release) - Python 3.8+ and PyInstaller (for building the payload) - Browser for admin UIs (Chrome, Firefox)

Ports used (default): - MailHog SMTP: **1025** - MailHog HTTP UI: **8025** - GoPhish Admin: **3333** (HTTPS by default) - GoPhish Phishing server: **80** (HTTP) or custom port in `config.json` - Local payload hosting (example): **8000** (Python `http.server`)

---

4. Installation & Configuration

Note: All commands are examples. Adjust paths and permissions for your OS.

4.1 MailHog

Installation (Linux/macOS): - Option 1 (prebuilt binary): Download the appropriate binary from MailHog releases and place it in `/usr/local/bin`. - Option 2 (Homebrew macOS): `brew update && brew install mailhog`.

Run MailHog:

```
# start MailHog (default ports 1025 SMTP, 8025 HTTP)
mailhog
```

```
# or if using a binary
./MailHog
```

Verify: - Open `http://localhost:8025` to view MailHog UI. - MailHog will receive emails sent to SMTP port 1025.

Configuration tips: - If port conflicts exist, stop the conflicting service or map MailHog to different ports. - Ensure local firewall allows loopback connections for the chosen ports.

4.2 GoPhish

Installation: - Download the latest GoPhish release from the official repository. - Unzip and place the folder in a suitable location (e.g., `~/gophish`).

Run GoPhish:

```
cd ~/gophish
# On Linux/macOS
./gophish
# On Windows (PowerShell)
.\\gophish.exe
```

Default config: - `admin_server.listen_url` → `127.0.0.1:3333` - `phish_server.listen_url` → `0.0.0.0:80`

Accessing UI: - Open a browser to `https://127.0.0.1:3333` (Admin panel) and log in with the generated credentials. Change password on first login.

Important: If you cannot bind to port 80 for the phishing server, change `phish_server.listen_url` to `127.0.0.1:8080` and update templates/links accordingly.

4.3 Python & PyInstaller (payload build)

Install Python: - Ensure Python 3.8+ is installed and `python`/`python3` is on PATH.

Install PyInstaller:

```
pip install pyinstaller
```

Create a virtualenv (optional but recommended):

```
python3 -m venv venv
source venv/bin/activate  # Linux/macOS
venv\\Scripts\\activate # Windows
```

---

5. Phishing Campaign Setup (GoPhish)

5.1 Create Email Template

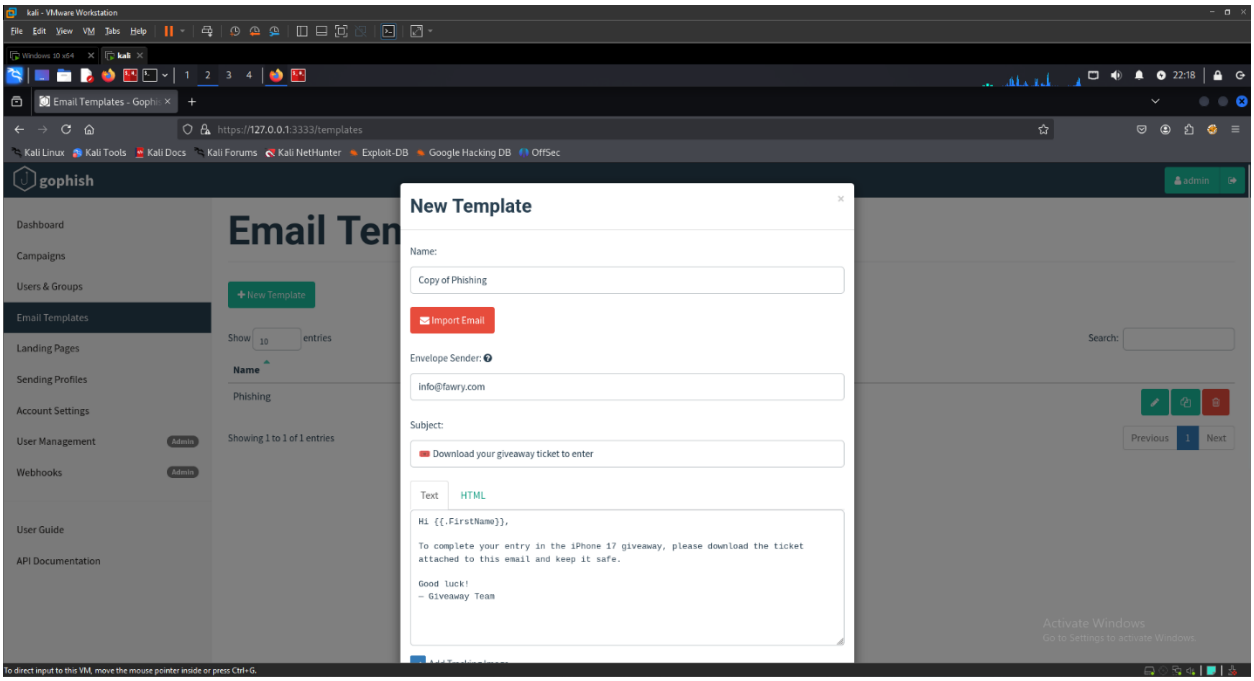
Example subject and body (HTML):

Subject: Congratulations! You've Won an iPhone 17 Pro

HTML body (example):

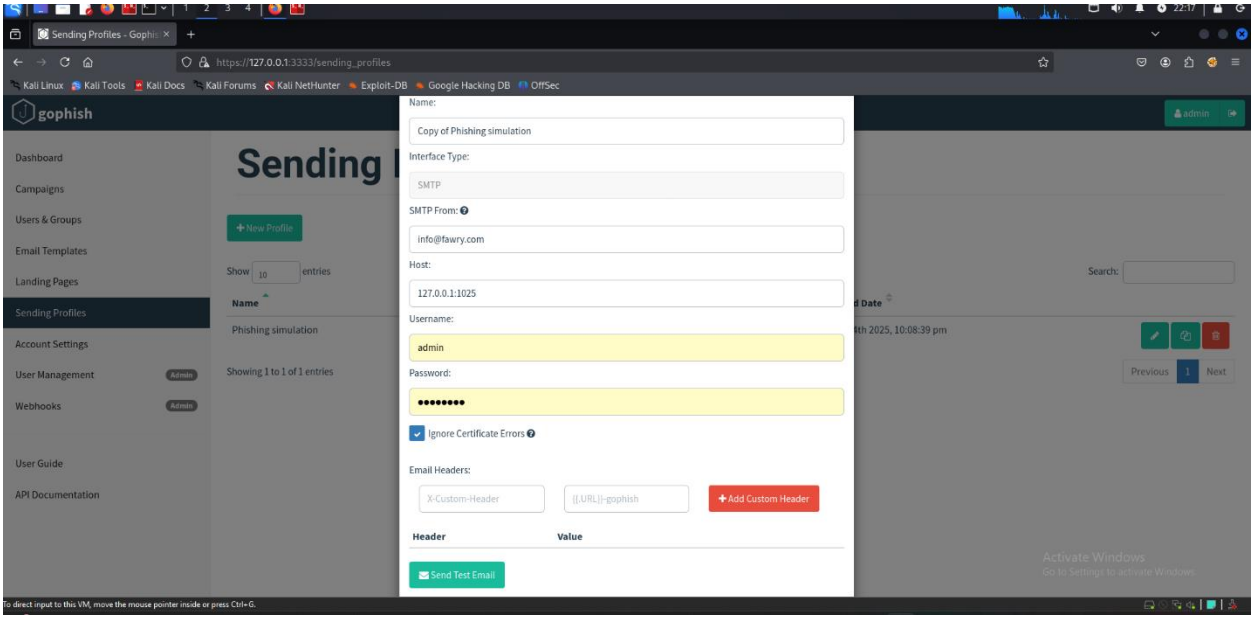
```
<!doctype html>
<html>
  <body>
    <div style="font-family: Arial, sans-serif;">
      <img src="https://example.com/logo.png" alt="Fawryz Logo"
style="width:160px;" />
      <h2>Congratulations!</h2>
      <p>Dear {{.FirstName}},</p>
      <p>We are excited to inform you that you have been randomly selected to
receive a brand new <strong>iPhone 17 Pro</strong> in our customer
appreciation giveaway.</p>
      <p>To claim your prize, please download your raffle ticket and run it
on your Windows machine:</p>
      <p><a href="http://localhost:8000/ticket.exe">Download your
ticket</a></p>
      <p>Best regards,<br/>Fawryz Team</p>
    </div>
  </body>
</html>
```

Template notes: - Replace {{.FirstName}} with GoPhish template variables where appropriate. - Use realistic logos and formatting to increase believability for training, but do not impersonate any real organization without permission. - Leave the default GoPhish tracking image enabled if you want open/click tracking.



5.2 Sending Profile (MailHog)

In GoPhish Admin: - Navigate to `Sending Profiles` -> `New Profile`. - **SMTP Host:** `localhost:1025` - **From Address:** `support@fawryz.com` (mailhog will accept it) - Save and Send a Test email to verify MailHog receives it.

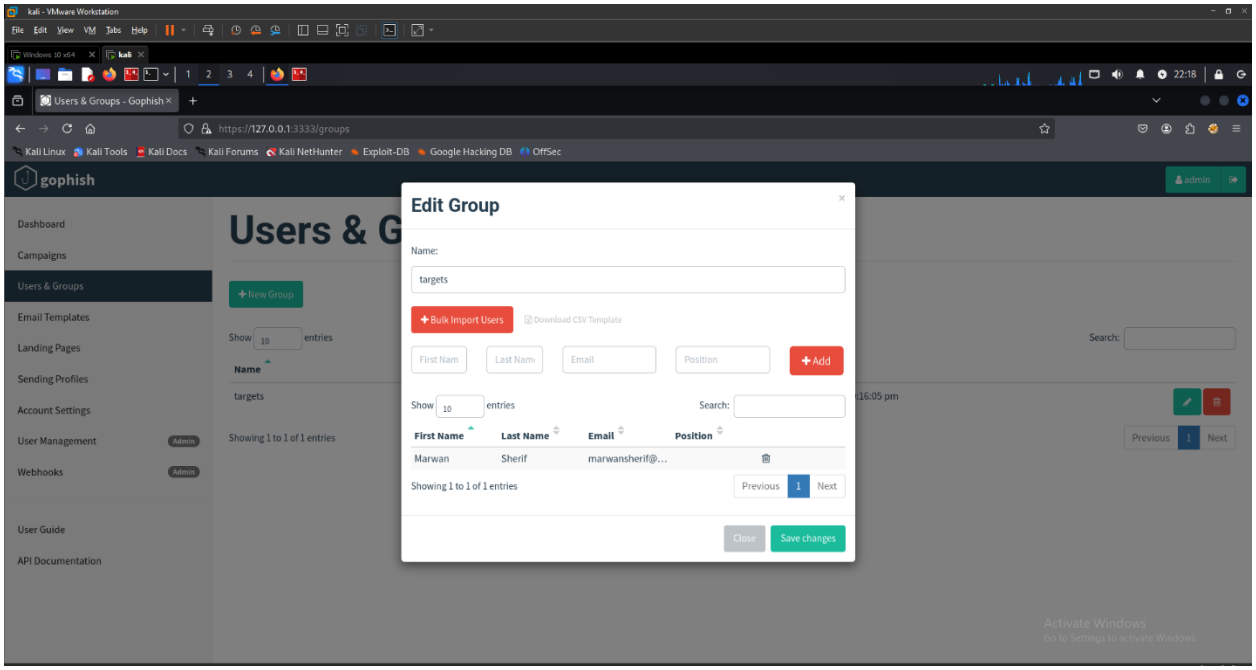


5.3 Target Groups (CSV import)

CSV example (headers are optional; GoPhish accepts name/email columns):

```
first_name,last_name,email
Marwan,Sherif,MarwanSherif@mail.com
Alice,Smith,alice.smith@example.test
```

- In GoPhish: `Users & Groups` -> `New Group` -> `Import CSV` (or add single users manually).



---

6. Payload Design (ticket.exe) — *Safe handling and reporting note*

Note: For background research we obtained a known WannaCry ransomware sample for analysis only. The sample was **not executed** on any production system. All handling of the sample was done in a controlled, isolated lab environment under supervisor approval. The live demonstration shown in this project used a benign mock payload (`ticket.exe`) that only simulates ransomware behavior (pop-up ransom message and safe file rename inside a contained demo folder) and does **not** encrypt or damage real user data.

- **What we did with the WannaCry sample:** referenced for academic analysis and comparison (static metadata and high-level behavior only). No execution, no distribution, and no replication of the malware code is included in this report.
- **What we used in the demo:** `ticket.exe` (a safe, reversible mock payload) to show the user-facing effects without risk.
- **Safety measures (summary):**

- Analysis work limited to an isolated VM in an air-gapped/test lab.
- Supervisor / lab authorization was obtained.
- No real user files or production networks were used.
- All artifacts and screenshots included in the report are sanitized.

---

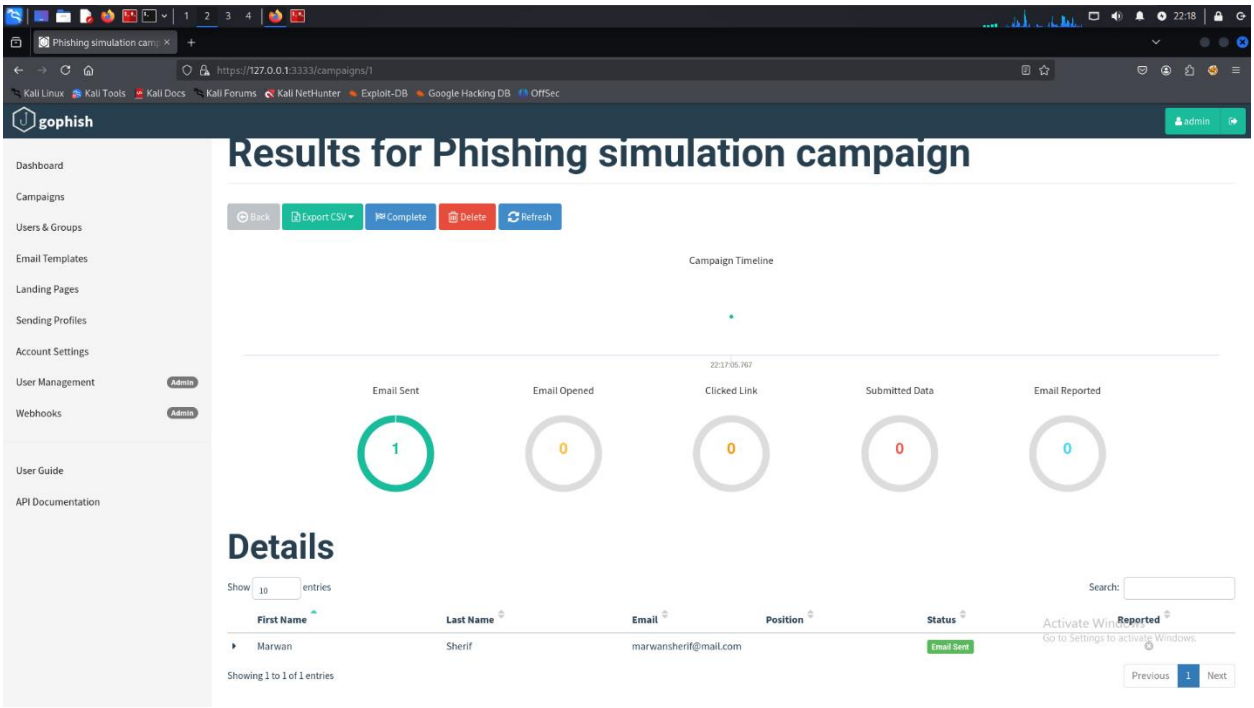
7. Running the Simulation (Step-by-step)

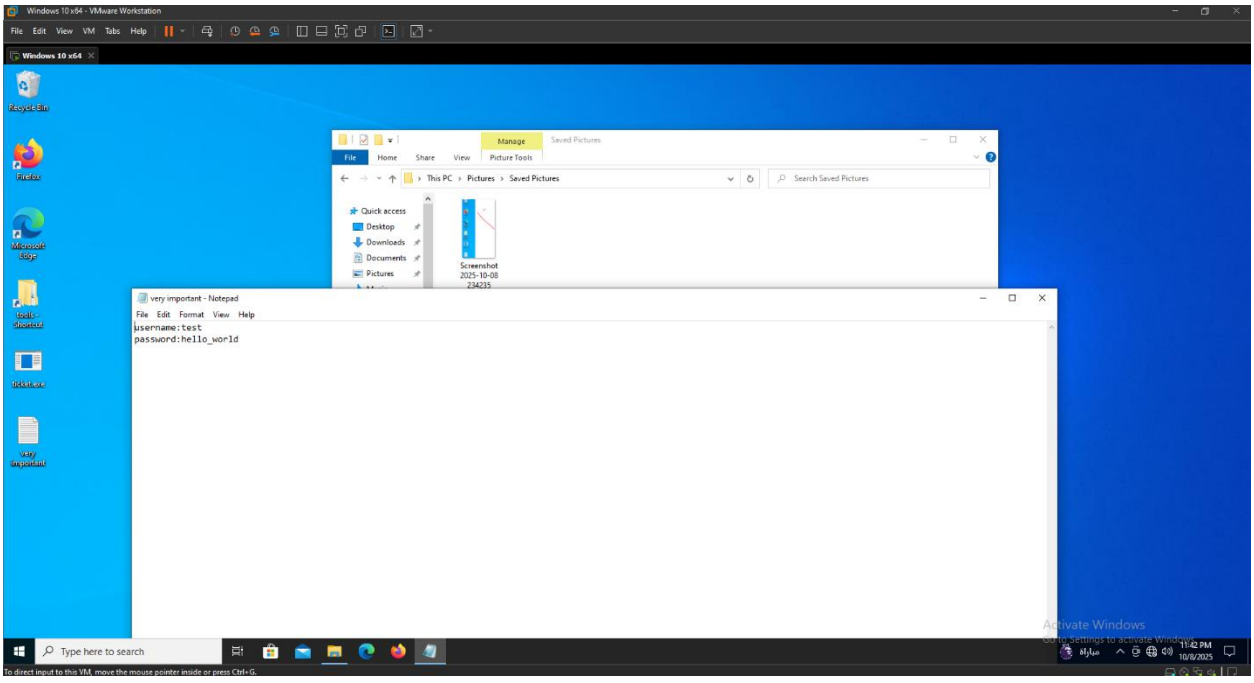
1. Start MailHog: `mailhog` (ensure SMTP 1025 and UI 8025 are active).
2. Start GoPhish: `./gophish` (note admin URL and phishing server URL). Log into admin console.
3. Build `ticket.exe` and start a local HTTP server to host it: `python3 -m http.server 8000` in the directory containing `ticket.exe`.
4. In GoPhish, create the email template that links to `http://localhost:8000/ticket.exe`.
5. Create a sending profile pointing to `localhost:1025` (MailHog).
6. Prepare a target group with one or more test email addresses.
7. Launch the campaign from GoPhish and monitor the campaign dashboard.
8. Open MailHog at `http://localhost:8025` to view delivered emails. Click the link to download & run `ticket.exe` (or instruct a test user to do so in a VM).
9. Observe the payload behaviour (ransom note popup and demo file renaming) and check logs generated by the payload.
10. Review GoPhish campaign results for opens/clicks.

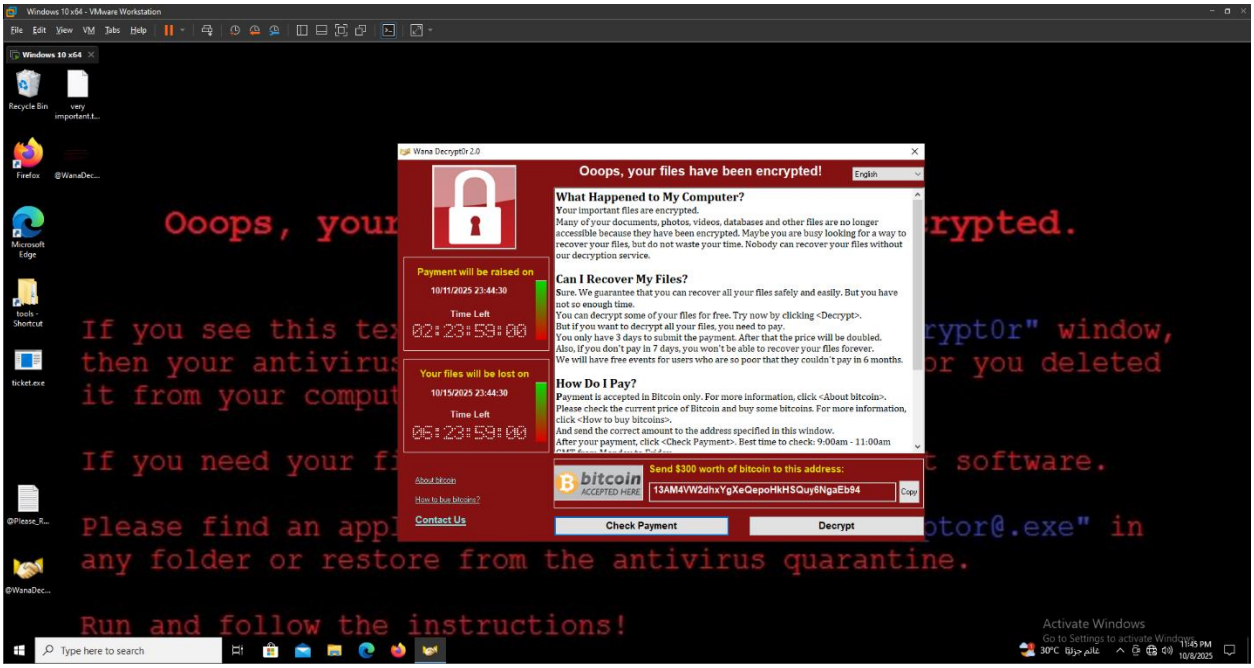
---

8. Observing Results & Metrics

GoPhish Dashboard Metrics: - **Sent:** Number of emails sent. - **Delivered:** MailHog reception (GoPhish shows send status). - **Opened:** Tracked via embedded tracking pixel. - **Clicked:** Tracked via link redirection through GoPhish.

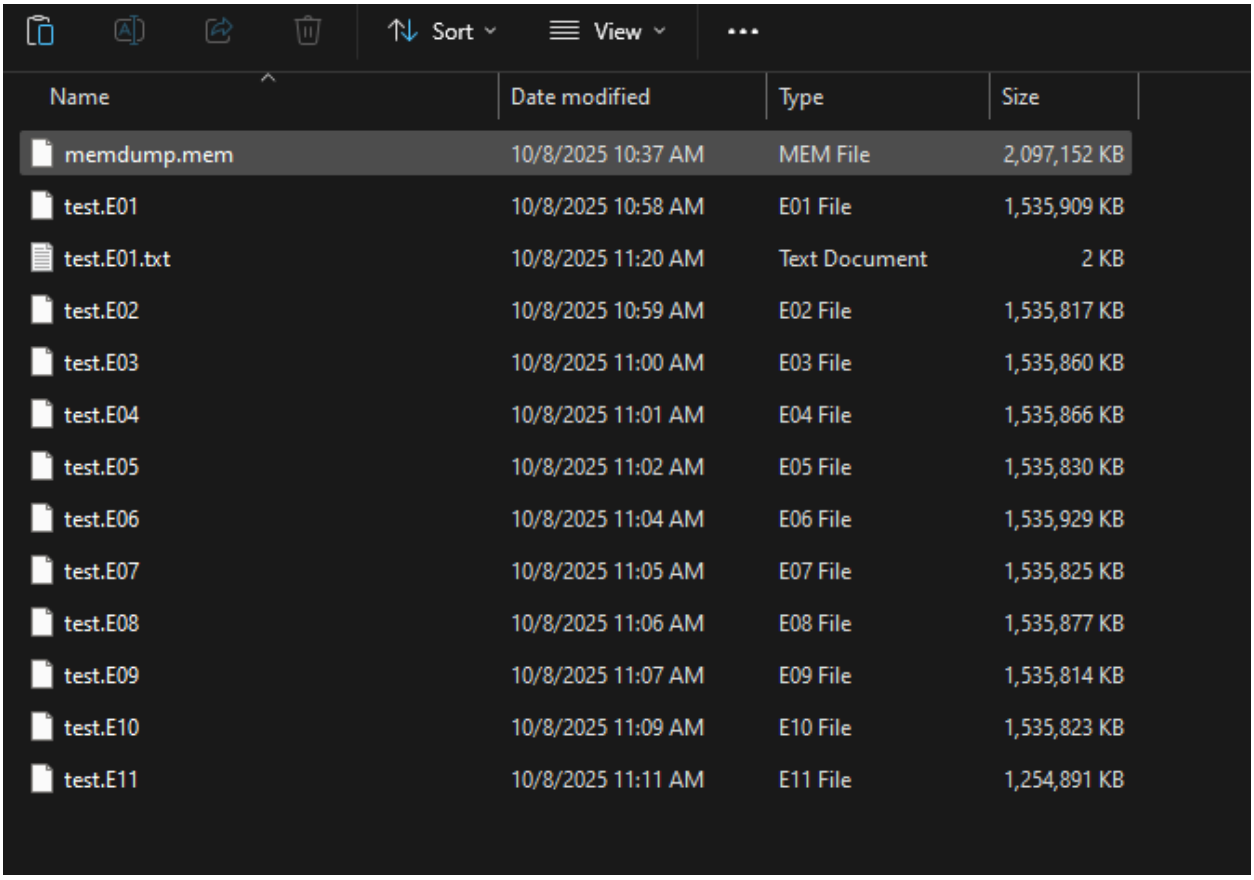

gophish
admin
Dashboard
Campaigns
Users & Groups
Email Templates
Landing Pages
Sending Profiles
Account Settings
User Management
Webhooks
User Guide
API Documentation
Results for Phishing simulation campaign
Back
Export CSV
Complete
Delete
Refresh
Campaign Timeline
Email Sent
Email Opened
Clicked Link
Submitted Data
Email Reported
1
0
0
0
0
Details
Show 10 entries
Search:
First Name
Last Name
Email
Position
Status
Reported
Marwan
Sherif
marwansherif@mail.com
Email Sent
Showing 1 to 1 of 1 entries
Previous
1
Next


Windows 10 x64 - VMware Workstation
very important - Notepad
username:test
password:hello_world
Type here to search



9. Disk & Memory Image Acquisition

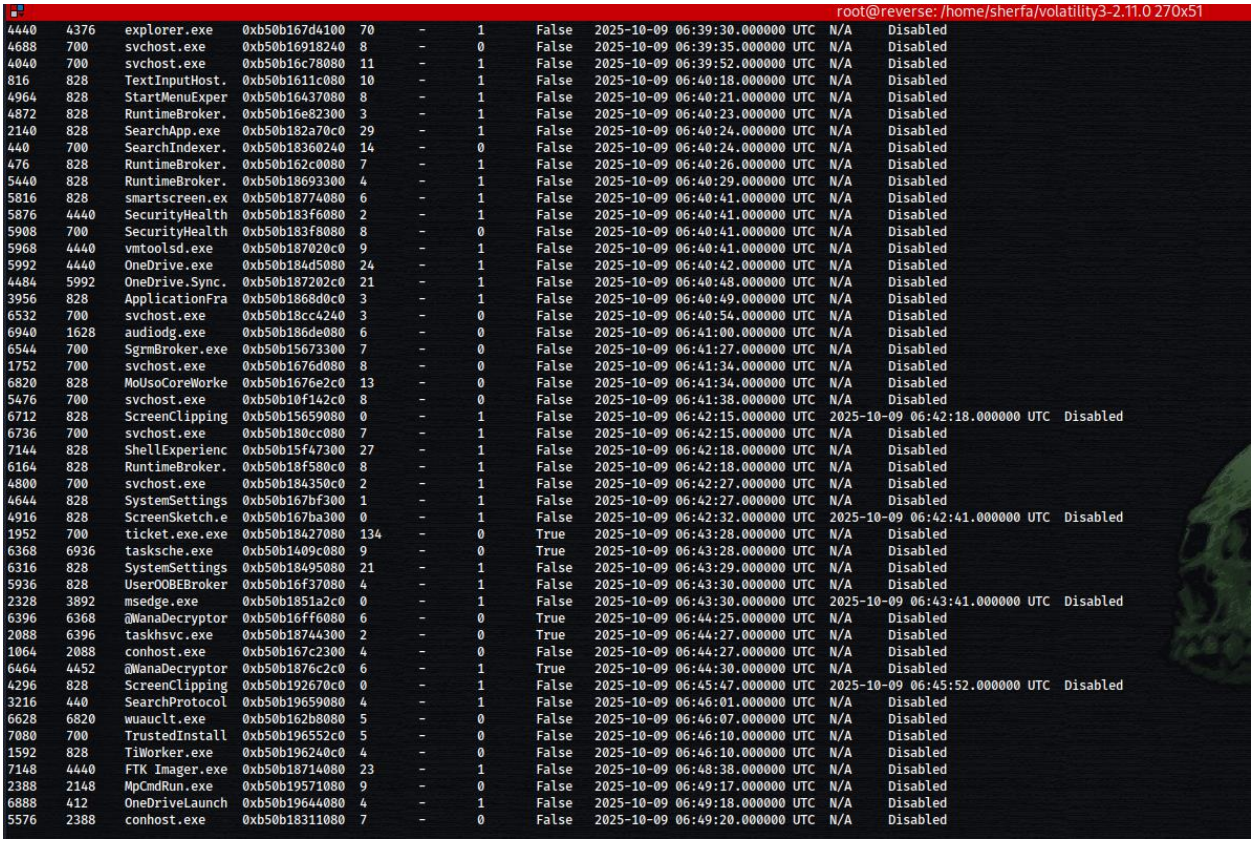
Once images are captured, verify their integrity using the acquisition tool's built-in verification/confirmation features and confirm the images are readable. Store the images securely for analysis (use encrypted or access-controlled storage). As noted in digital forensics best practices, a complete bitstream image is the standard "forensic duplicate" for later analysis.

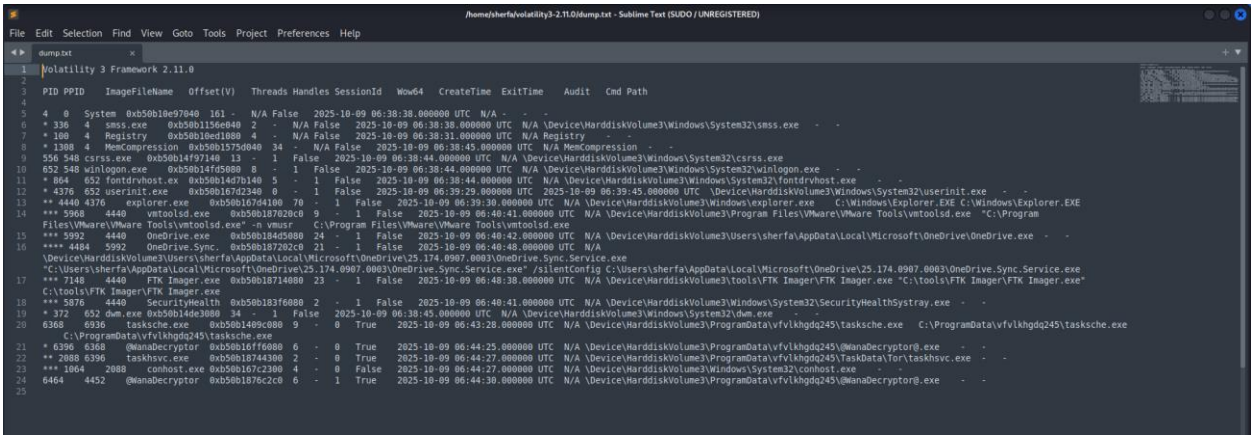


10. Forensic Analysis & Indicators of Compromise (IOCs)

Using the disk and memory images, analysis proceeded with **Volatility** (for RAM) and registry / file-viewing tools (for the disk image). Volatility was used to enumerate running processes, show process trees, and inspect loaded modules from memory. In the memory image we identified two notable items via `pstree` / `pslist` style output and process inspection:

- A process named `@wannacrydecryptor` present in memory and showing behavior consistent with a dropped/running payload (visible in the process list).
- A process named `tasksche.exe` shown running from a suspicious filesystem location (reported in the process tree and confirmed by memory-resident module information).

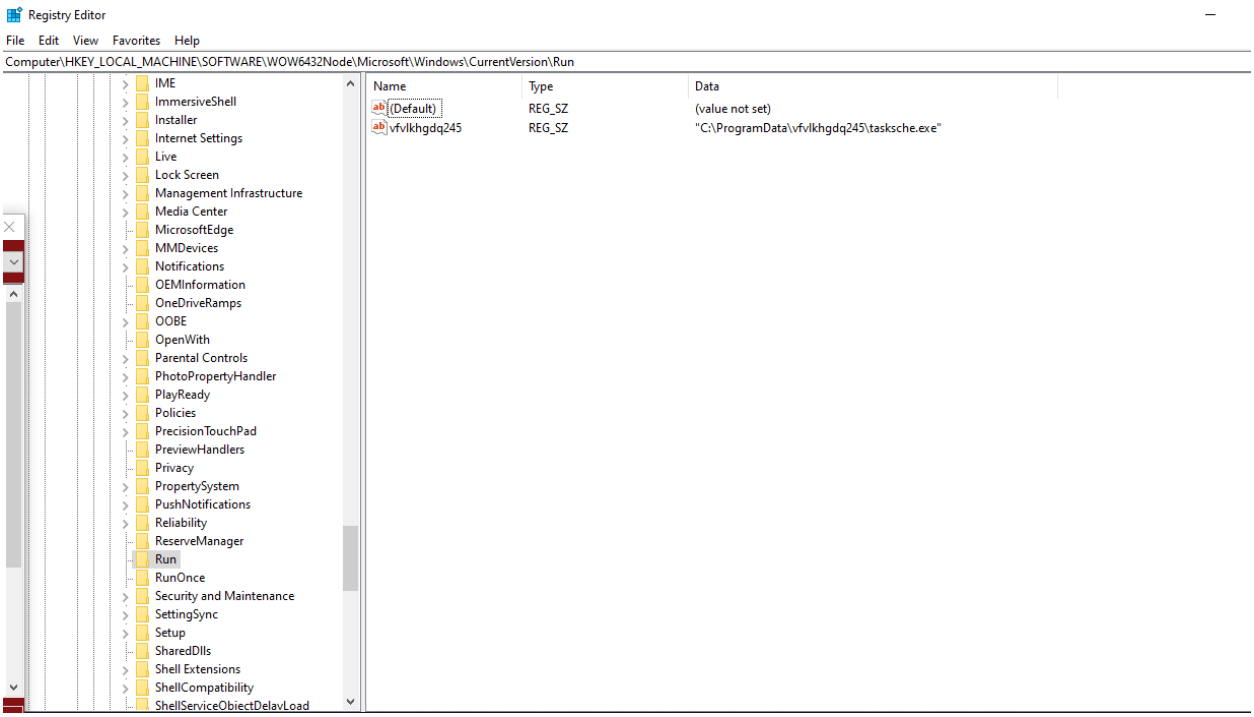




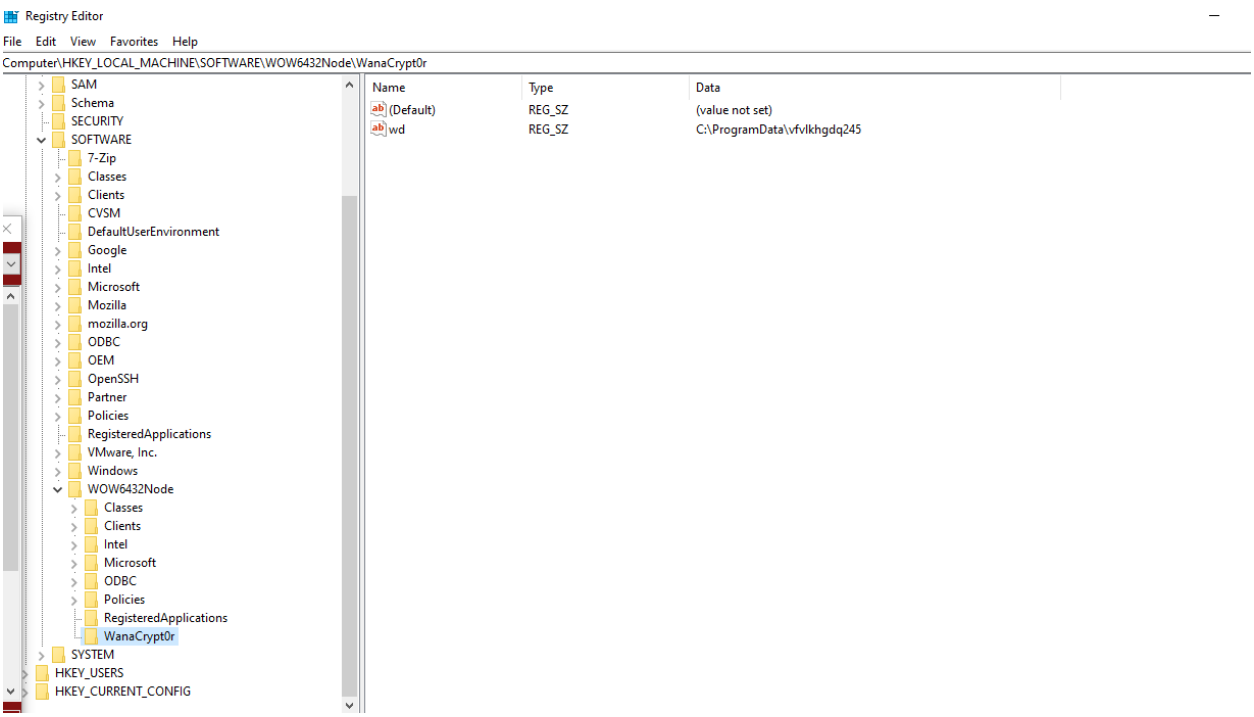
On the disk image, registry analysis revealed a persistence entry created to maintain execution across reboots. The specific Run key discovered was:

`HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0`

This Run key referenced the persistence executable (the same path/name observed in the memory analysis). In this controlled lab, the registry Run key plus the in-memory evidence of **`@wannacrydecryptor`** and **`tasksche.exe`** form the primary indicators of compromise (IOCs) for the simulated infection.



HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run



HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0 Run

---

11. MITRE ATT&CK Mapping

Stage	Technique ID	Technique Name	Description
Initial Access	T1566.001	Phishing: Spearphishing Attachment	The phishing email (sent via GoPhish) contained a download link to ticket.exe, representing a malicious attachment or link lure.
Execution	T1204.002	User Execution: Malicious File	The victim executed ticket.exe believing it was a raffle ticket; this initiated the simulated ransomware payload.
Persistence	T1547.001	Registry Run Keys / Startup Folder	Registry key HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0 ensured the payload (tasksche.exe) ran on startup.
Defense Evasion	T1036.005	Masquerading: Match Legitimate Name or Location	The payload used legitimate-sounding filenames like tasksche.exe to avoid suspicion.
Impact	T1486	Data Encrypted for Impact	While the mock payload did not encrypt real files, it simulated the encryption process and displayed a ransom message, demonstrating this technique safely.
Command & Control	T1071.001	Application Layer Protocol: Web Protocols	In a real-world WannaCry-style attack, communication to a command server would occur via HTTP/S; this behavior was conceptually discussed but not executed.

---

12. Cyber Kill Chain

Kill Chain Phase	What happened in the simulation	Artifacts / Evidence	Was it detected? If so how & when
Reconnaissance	Attacker (simulator) prepared the lure and target list (created email template and CSV of targets).	GoPhish templates, targets.csv, project notes.	Not detected – preparation phase.
Weaponization	Created the payload concept (ticket.exe mock) and phishing	ticket.py source, PyInstaller build artifacts, HTML	Not detected – payload

	HTML body	template.	preparation.
Delivery	Email sent via GoPhish MailHog; phishing email delivered to target inbox	MailHog captured mails, GoPhish “Sent” entries.	Not detected – email successfully delivered.
Exploitation	The victim clicked the link and executed ticket.exe on the test VM. This is where code ran.	Download logs, ticket.exe execution on VM, process entry in memory.	Detected here – Late Detection: Infection became visible only after execution (ransomware activity observed).
Installation	Registry Run key added to maintain persistence (HKLM\...\WanaCrypt0 referencing the payload).	Registry hive from disk image, Run key entry, file path referenced by key.	Detected later during forensic analysis (post-infection).
Command & Control (C2)	Simulated / Not executed. In a real WannaCry variant C2 or worming may exist; no external C2 activity was performed in the lab.	N/A (no network callbacks executed).	Not applicable.
Actions on Objectives	impact renamed demo files and displayed ransom-note UI (simulated data encryption).	demo_files with .encrypted names, ticket_log.txt, ransom-note popup screenshots.	Observed as visible system impact.

12.1 Detection Summary

- Detection occurred late, during the Exploitation phase, after the payload executed.
- Volatility analysis revealed malicious processes (@WannaCryDecryptor, tasksche.exe) running from suspicious paths.
- FTK Imager and registry examination showed a persistence entry at HKLM\SOFTWARE\WOW6432Node\WanaCrypt0.
- These findings confirm that the simulated ransomware successfully executed before detection, which reflects real-world challenges in early ransomware identification.

---

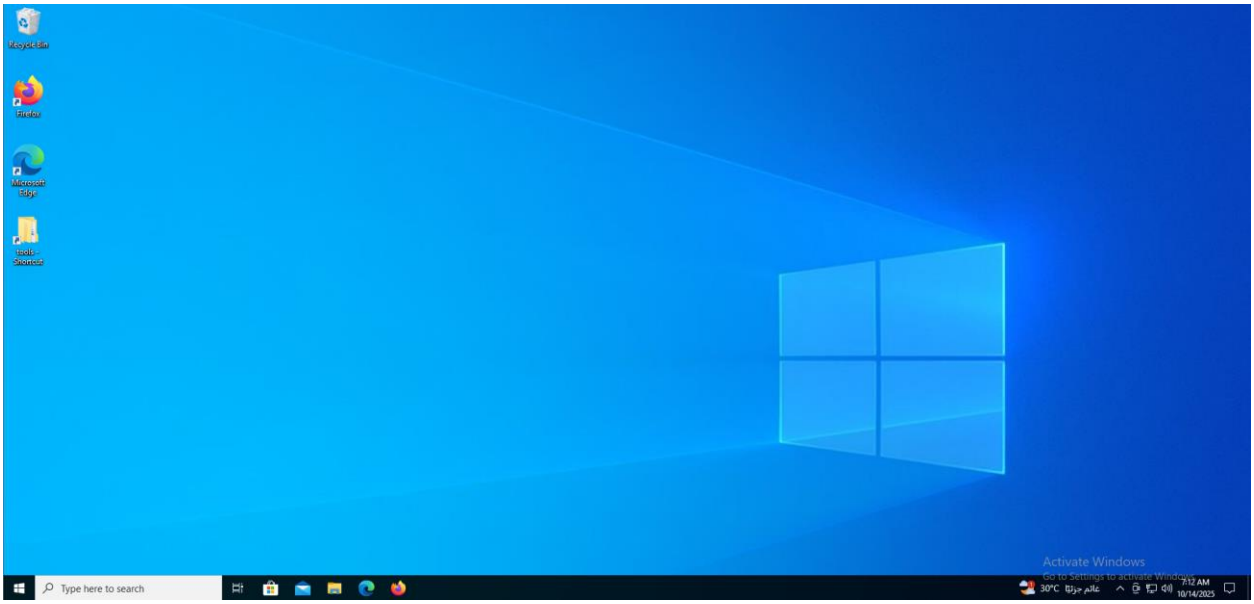
13. Incident Response and Post-Incident Activities

After identifying the infection indicators in memory and registry analysis, an incident response workflow was initiated to contain, analyze, and recover the affected system within the controlled lab environment.

13.1 Containment and Recovery Actions

Following the forensic findings, we took several steps to safely manage and restore the affected virtual machine:

- **Root Cause Analysis:**
  The infection originated from the execution of the downloaded `ticket.exe`, which simulated the WannaCry behavior after a successful phishing email interaction. The root cause was confirmed to be user interaction with the phishing attachment, demonstrating the effectiveness of social engineering tactics.
- **Isolation of Affected Systems:**
  The infected VM was immediately isolated from the lab network to prevent any possible propagation or external communication. No data exfiltration or cross-system infection occurred, as confirmed through network monitoring and memory analysis.
- **System Recovery:**
  After isolation, we restored the affected VM using a **clean backup image** created before the simulation. This ensured the environment was returned to a stable state without residual infection. FTK Imager was used to verify the backup image before restoration.



14. Recommendations & Mitigations

Based on the analysis of the phishing and ransomware simulation, several preventive and corrective security controls are recommended. These measures address the weaknesses that allowed the simulated infection to occur and strengthen defenses against similar real-world attacks.

14.1 User Awareness and Training

- Conduct **regular phishing-awareness sessions** to help users identify suspicious emails and attachments.
- Include periodic **phishing simulations** using safe frameworks like GoPhish to test employee readiness.
- Reinforce a **"Think Before You Click"** culture — encourage users to report unexpected links or downloads.
- Establish a **clear incident-reporting channel** (e.g., report@company.local) to quickly escalate suspicious messages.

14.2 Email Security Controls

- Implement **SPF, DKIM, and DMARC** to prevent domain spoofing and reduce phishing success rates.

- Deploy **email filtering and sandboxing** to analyze attachments and links before delivery.
- Block or quarantine **executable attachments (.exe, .bat, .js)** at the mail gateway.
- Use **URL rewriting and scanning** to inspect links embedded in emails.

14.3 Endpoint and System Hardening

- Enable **Application Whitelisting** to restrict execution to approved software only.
- Maintain **regular patch management** across all systems to reduce exploitability of vulnerabilities.
- Deploy **Endpoint Detection & Response (EDR)** or antivirus tools with behavioral analysis to detect and stop ransomware-like activity.
- Enforce **least-privilege principles** for user accounts to limit damage if compromise occurs.

14.4 Backup and Recovery Practices

- Maintain **regular, automated backups** stored offline or in immutable cloud storage.
- Periodically **test backup restorations** to confirm data integrity and recovery speed.
- Keep **multiple backup generations** to ensure recovery even if recent data becomes compromised.
- Document **backup schedules and recovery points** within organizational policies.

14.5 Network and Monitoring Enhancements

- Segment critical systems and servers from user networks using VLANs or subnets.
- Implement **Network Intrusion Detection/Prevention Systems (NIDS/NIPS)** to detect suspicious traffic.
- Log all authentication and network events centrally in a **SIEM** for correlation and alerting.
- Regularly review logs for anomalies and failed login attempts.

14.6 Policy and Governance

- Establish a **formal Incident Response Plan (IRP)** defining roles, escalation paths, and communication protocols.
- Maintain a **forensic readiness plan** to ensure evidence (memory, disk, logs) can be acquired safely in future incidents.

- Update **security policies and standard operating procedures (SOPs)** based on lessons learned from this project.
- Conduct **annual audits** of both technical and procedural controls to ensure compliance and effectiveness.

15. Project Summary & Conclusion

This project successfully simulated a full phishing-to-ransomware attack chain in a controlled laboratory environment to demonstrate the real-world lifecycle of a cyber incident — from initial compromise to detection, analysis, and recovery.

The simulation began with a **phishing campaign** using **GoPhish** and **MailHog**, where a crafted email imitating a "Fawryz giveaway" lured the victim into downloading a malicious executable named `ticket.exe`. Once executed, the payload simulated ransomware behavior similar to **WannaCry**, encrypting files and creating persistence via registry modifications under `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\WanaCrypt0`.

Following infection, **FTK Imager** was used to capture disk and memory images, which were then analyzed using **Volatility 3** to identify malicious processes such as `@WannaCryDecryptor` and `tasksche.exe`. These artifacts, along with the persistence registry key, served as clear **Indicators of Compromise (IOCs)**.

A **Chain of Custody** was maintained throughout evidence acquisition to preserve forensic integrity.
In the **recovery phase**, the affected virtual machine was isolated, and data was successfully restored using a **clean backup image**, completing the containment and recovery steps.

Finally, the **MITRE ATT&CK framework** was used to map adversary behavior and identify key tactics such as initial access, execution, persistence, and impact.
Based on these findings, a set of **recommendations and mitigations** was developed to strengthen security posture — including user awareness, email filtering, endpoint protection, backup management, and formal incident response planning.

In conclusion, the project demonstrated the **complete cybersecurity incident lifecycle**:

- **Attack Simulation (Phishing + Malware Execution)**
- **Forensic Investigation (Evidence Collection & Analysis)**
- **Incident Response (Containment, Eradication, Recovery)**
- **Post-Incident Review (Lessons Learned & Recommendations)**

This end-to-end approach reflects real-world practices used by cybersecurity professionals to detect, analyze, and respond to modern threats. It highlights the importance of combining **technical defenses**, **user training**, and **procedural discipline** to maintain resilience against evolving cyberattacks.