Phishing Email Forensics & Analysis Project

Table of Contents

Project Overview

Phishing Email Forensics is a comprehensive cybersecurity project demonstrating evidence preservation techniques, malicious header analysis, domain verification, and threat detection methodologies. This project showcases real-world email forensic investigation using Kali Linux in a virtual environment.

Key Components

- **Evidence Preservation**: Proper handling and documentation of email artifacts
- **Header Analysis**: Deep inspection of SPF, DKIM, and DMARC records
- **Domain Intelligence**: Investigation of sender domains and IPs
- **Threat Detection**: Identifying malicious links, attachments, and indicators
- **Chain of Custody**: Maintaining forensic integrity throughout investigation

Learning Objectives

By completing this project, you will:

✅ Understand email authentication protocols (SPF, DKIM, DMARC)
✅ Master evidence preservation techniques
✅ Analyze email headers for malicious indicators
✅ Identify phishing attempts through header analysis
✅ Document findings using forensic standards
✅ Apply NIST CSF and ISO 27001 controls
✅ Create comprehensive incident reports
✅ Understand the full attack chain of phishing campaigns

Prerequisites

Required Knowledge

- Basic Linux command-line operations
- Understanding of email systems (SMTP, MX records)
- Fundamental networking concepts (DNS, IP addressing)
- Basic security concepts

Required Software

- **Operating System**: Kali Linux (2023.x or later)
- **Virtualization**: Oracle VirtualBox or VMware
- **Email Client**: Thunderbird (pre-installed on Kali)
- **Analysis Tools**:
    - VirusTotal CLI
    - Maltego (optional)
    - TheHarvester
    - whois
    - nslookup/dig
    - curl/wget

Hardware Requirements

- **RAM**: Minimum 4GB (8GB recommended)
- **Storage**: 30GB free space
- **Processor**: 64-bit processor with virtualization support

Email Authentication Protocols

1. SPF (Sender Policy Framework)

Definition: SPF is an email authentication method that specifies which mail servers are authorized to send emails on behalf of a domain.

Purpose: Prevents email spoofing by verifying sender IP addresses

How it Works:

1. Domain owner publishes SPF record in DNS

2. Receiving server checks sender's IP against SPF record

3. Result: Pass, Fail, SoftFail, Neutral, or None

SPF Record Example:

v=spf1 ip4:192.0.2.0/24 include:_spf.google.com ~all

Key Components:

- v=spf1: SPF version
- ip4: Authorized IPv4 address range
- include: Include another domain's SPF record
- ~all: SoftFail (suspicious but not rejected)
- -all: Fail (reject)
- +all: Pass (not recommended)

✅ **Pass Results**: spf=pass
❌ **Fail Results**: spf=fail, spf=softfail

2. DKIM (DomainKeys Identified Mail)

Definition: DKIM uses cryptographic signatures to verify that an email hasn't been tampered with during transit and confirms the sender's domain.

Purpose: Ensures email integrity and authenticity through digital signatures

How it Works:

1. Sending server signs email with private key

2. Public key published in DNS

3. Receiving server verifies signature with public key

4. Result: Pass or Fail

DKIM Header Example:

DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;

d=example.com; s=selector1;

h=from:to:subject:date;

bh=BASE64_BODY_HASH;

b=BASE64_SIGNATURE

Key Components:

- v=1: DKIM version
- a=rsa-sha256: Algorithm used
- d=: Signing domain
- s=: Selector (DNS record identifier)
- bh=: Body hash
- b=: Signature

✅ **Pass Results**: dkim=pass
❌ **Fail Results**: dkim=fail, dkim=temperror, dkim=permerror

3. DMARC (Domain-based Message Authentication, Reporting & Conformance)

Definition: DMARC builds on SPF and DKIM to provide a policy framework for email authentication, reporting, and enforcement.

Purpose: Provides clear policy on how to handle authentication failures

How it Works:

1. Domain owner publishes DMARC policy in DNS

2. Receiving server checks SPF and DKIM alignment

3. Applies policy based on authentication results

4. Sends reports back to domain owner

DMARC Record Example:

Admin123v=DMARC1; p=reject; rua=mailto:dmarc@example.com; pct=100; adkim=s; aspf=s

Key Components:

- v=DMARC1: DMARC version
- p=: Policy (none, quarantine, reject)
- rua=: Aggregate report email
- pct=: Percentage of emails to filter
- adkim=: DKIM alignment mode (s=strict, r=relaxed)
- aspf=: SPF alignment mode

Policies:

- p=none: Monitor only (no action)
- p=quarantine: Mark as spam
- p=reject: Block completely

✅ **Pass Results**: dmarc=pass
❌ **Fail Results**: dmarc=fail

Authentication Protocol Summary Table

Protocol	What It Validates	Result Location	Pass Indicator	Fail Indicator
SPF	Sender IP address	Received-SPF header	spf=pass	spf=fail / spf=softfail
DKIM	Email signature/integrity	Authentication-Results	dkim=pass	dkim=fail
DMARC	SPF+DKIM alignment	Authentication-Results	dmarc=pass	dmarc=fail

Lab Environment Setup

Step 1: Install Kali Linux on VirtualBox

bash

Download Kali Linux ISO from official website

URL: https://www.kali.org/get-kali/#kali-virtual-machines

Create new VM in VirtualBox:

Name: Kali-Forensics-Lab

Type: Linux

Version: Debian (64-bit)

RAM: 4096 MB (minimum)

Storage: 30 GB dynamic VDI

Step 2: Initial System Configuration

bash

Update system

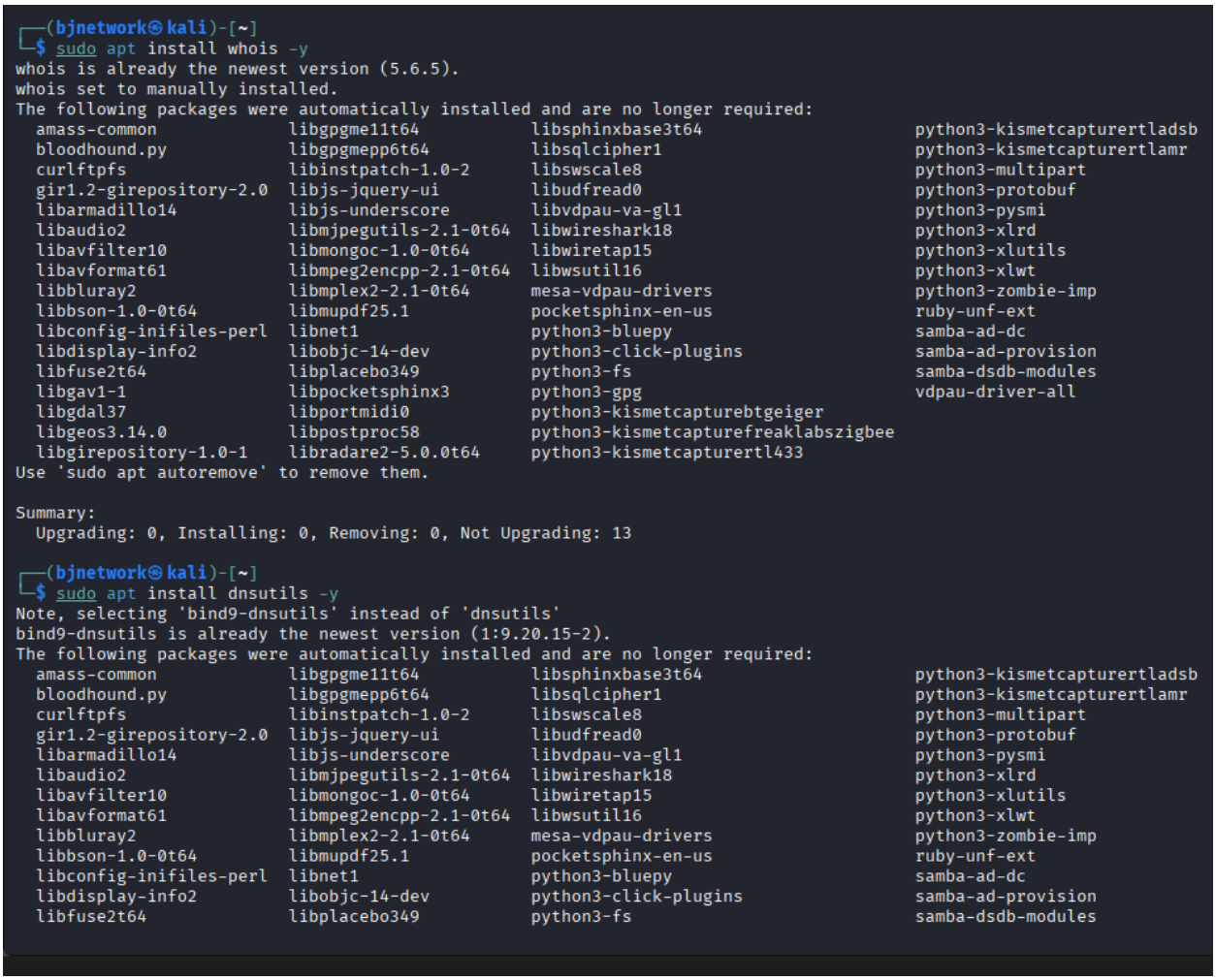
sudo apt update && sudo apt upgrade -y

Install required tools

```
sudo apt install -y \
    thunderbird \
    curl \
    wget \
    whois \
    dnsutils \
    net-tools \
    wireshark \
    tcpdump \
    forensics-all \
    python3-pip
```

Install VirusTotal CLI (optional)

```
pip3 install vt-py
```



Step 3: Create Forensic Working Directory

bash

Create structured directory for case management

mkdir -p ~/Forensics/Phishing_Cases/{evidence,reports,screenshots,logs}

cd ~/Forensics/Phishing_Cases

Create case documentation template

touch case_notes.txt chain_of_custody.txt findings.txt

Directory Structure:

~/Forensics/Phishing_Cases/

├── evidence/ # Original email files (.eml, .msg)

├── reports/ # Investigation reports

├── screenshots/ # Visual documentation

├── logs/ # Analysis logs and command outputs

├── case_notes.txt # Investigator notes

├── chain_of_custody.txt # Evidence tracking

└── findings.txt # Summary of findings

Step-by-Step Forensic Process

Phase 1: Evidence Acquisition

Step 1: Download and Preserve Email

Using Thunderbird:

1. Open Thunderbird email client
2. Locate the suspicious email
3. Right-click → "Save As..." → Save as .eml format

Command-Line Method:

bash

```
# Navigate to evidence folder
cd ~/Forensics/Phishing_Cases/evidence/

# Create timestamped filename
DATE=$(date +%Y%m%d_%H%M%S)
CASE_ID="PHISH_001"
EVIDENCE_FILE="${CASE_ID}_${DATE}.eml"

# If downloading from email server (IMAP example)
# curl --url "imaps://mail.example.com/INBOX;UID=12345" \
#      --user "user@example.com:password" \
#      --output "$EVIDENCE_FILE"
```

Step 2: Generate File Hash (Evidence Integrity)

CRITICAL FOR FORENSICS: Always calculate cryptographic hashes immediately after acquisition

bash

 # Generate MD5 hash

md5sum "$EVIDENCE_FILE" > "${EVIDENCE_FILE}.md5"

 # Generate SHA256 hash (preferred)

sha256sum "$EVIDENCE_FILE" > "${EVIDENCE_FILE}.sha256"

 # Display both hashes

echo "=== Evidence Hashes ==="

cat "${EVIDENCE_FILE}.md5"

cat "${EVIDENCE_FILE}.sha256"

Example Output:

=== Evidence Hashes ===

5d41402abc4b2a76b9719d911017c592 PHISH_001_20260128_143022.eml

e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 PHISH_001_20260128_143022.eml

Step 3: Document Chain of Custody

bash

 # Create chain of custody entry

cat << EOF >> ../chain_of_custody.txt

Case ID: ${CASE_ID}

Evidence File: ${EVIDENCE_FILE}

```
Acquisition Date: $(date)
Acquired By: $(whoami)
Source: Phishing report - user inbox
MD5 Hash: $(cat ${EVIDENCE_FILE}.md5 | cut -d' ' -f1)
SHA256 Hash: $(cat ${EVIDENCE_FILE}.sha256 | cut -d' ' -f1)
Storage Location: $(pwd)

EOF

# View chain of custody
cat ../chain_of_custody.txt
```

Phase 2: Email Header Extraction

Step 1: Extract Full Headers from Email

bash

 # Method 1: Using grep to extract headers (most reliable)

grep -E "^[A-Za-z-]+:" "$EVIDENCE_FILE" > ../logs/headers_${CASE_ID}.txt

 # Method 2: Extract everything before first blank line

sed '/^$/q' "$EVIDENCE_FILE" > ../logs/headers_full_${CASE_ID}.txt

 # Method 3: Using formail (if available)

formail -X "" < "$EVIDENCE_FILE" > ../logs/headers_formail_${CASE_ID}.txt

Step 2: Parse Critical Header Fields

bash

 # Create header analysis script

cat << 'EOF' > ../analyze_headers.sh

#!/bin/bash

HEADER_FILE="$1"

echo "=================================="

echo "EMAIL HEADER ANALYSIS REPORT"

echo "=================================="

echo ""

echo "--- SENDER INFORMATION ---"

grep -i "^From:" "$HEADER_FILE"

```
grep -i "^Return-Path:" "$HEADER_FILE"
grep -i "^Reply-To:" "$HEADER_FILE"
echo ""

echo "--- RECIPIENT INFORMATION ---"
grep -i "^To:" "$HEADER_FILE"
grep -i "^Cc:" "$HEADER_FILE"
echo ""

echo "--- SUBJECT & DATE ---"
grep -i "^Subject:" "$HEADER_FILE"
grep -i "^Date:" "$HEADER_FILE"
echo ""

echo "--- AUTHENTICATION RESULTS ---"
grep -i "^Authentication-Results:" "$HEADER_FILE"
grep -i "^Received-SPF:" "$HEADER_FILE"
grep -i "^DKIM-Signature:" "$HEADER_FILE"
echo ""

echo "--- MESSAGE ID & TRACKING ---"
grep -i "^Message-ID:" "$HEADER_FILE"
grep -i "^X-Mailer:" "$HEADER_FILE"
grep -i "^User-Agent:" "$HEADER_FILE"
echo ""

echo "--- RECEIVED PATH (chronological order) ---"
```

```
grep -i "^Received:" "$HEADER_FILE" | tac
echo ""

echo "====================================="
EOF

# Make script executable
chmod +x ../analyze_headers.sh

# Run analysis
../analyze_headers.sh ../logs/headers_${CASE_ID}.txt | tee ../logs/analysis_${CASE_ID}.txt
```

Phase 3: SPF/DKIM/DMARC Verification

Step 1: Extract Authentication Headers

bash

 # Extract authentication results

```
echo "=== SPF CHECK ===" > ../logs/auth_results_${CASE_ID}.txt

grep -i "Received-SPF:" ../logs/headers_${CASE_ID}.txt >>
../logs/auth_results_${CASE_ID}.txt

echo "" >> ../logs/auth_results_${CASE_ID}.txt


echo "=== DKIM CHECK ===" >> ../logs/auth_results_${CASE_ID}.txt

grep -i "DKIM-Signature:" ../logs/headers_${CASE_ID}.txt >>
../logs/auth_results_${CASE_ID}.txt

echo "" >> ../logs/auth_results_${CASE_ID}.txt


echo "=== DMARC CHECK ===" >> ../logs/auth_results_${CASE_ID}.txt

grep -i "Authentication-Results:" ../logs/headers_${CASE_ID}.txt >>
../logs/auth_results_${CASE_ID}.txt


# Display results

cat ../logs/auth_results_${CASE_ID}.txt
```

Step 2: Identify Key Indicators

🚩 **Red Flags to Look For**:

Indicator	What to Check	Suspicious Pattern
SPF	Received-SPF	fail, softfail, none
DKIM	dkim= in Authentication-Results	fail, none, temperror
DMARC	dmarc= in Authentication-Results	fail, none
From Address	Match display name vs actual	Mismatch indicates spoofing
Return-Path	Compare to From address	Different domain = suspicious
Reply-To	Different from sender	Often used in phishing

Example Analysis:

bash

```
# Check for SPF failure
if grep -qi "spf=fail\|spf=softfail" ../logs/auth_results_${CASE_ID}.txt; then
    echo "[ALERT] SPF validation failed - sender may be spoofed"
fi

# Check for DKIM failure
if grep -qi "dkim=fail\|dkim=none" ../logs/auth_results_${CASE_ID}.txt; then
    echo "[ALERT] DKIM validation failed - email may be forged"
fi
# Check for DMARC failure
if grep -qi "dmarc=fail\|dmarc=none" ../logs/auth_results_${CASE_ID}.txt; then
    echo "[ALERT] DMARC validation failed - high phishing probability"
Fi
```

Phase 4: Domain & IP Intelligence Gathering

Step 1: Extract Sender Domain and IP

bash

 # Extract sender domain

```
SENDER_DOMAIN=$(grep -i "^From:" ../logs/headers_${CASE_ID}.txt | grep -oE '@[^>]+' | tr -d '@' | head -1)

echo "Sender Domain: $SENDER_DOMAIN"


# Extract originating IP from last Received header

ORIGIN_IP=$(grep -i "^Received:" ../logs/headers_${CASE_ID}.txt | tail -1 | grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' | head -1)

echo "Origin IP: $ORIGIN_IP"


# Save to investigation log

echo "Sender Domain: $SENDER_DOMAIN" >> ../logs/investigation_${CASE_ID}.txt

echo "Origin IP: $ORIGIN_IP" >> ../logs/investigation_${CASE_ID}.txt
```

Step 2: WHOIS Lookup

bash

 # Domain WHOIS lookup

```
echo "=== DOMAIN WHOIS LOOKUP ===" | tee -a ../logs/investigation_${CASE_ID}.txt

whois "$SENDER_DOMAIN" | tee -a ../logs/investigation_${CASE_ID}.txt

echo "" | tee -a ../logs/investigation_${CASE_ID}.txt


# IP WHOIS lookup

echo "=== IP WHOIS LOOKUP ===" | tee -a ../logs/investigation_${CASE_ID}.txt

whois "$ORIGIN_IP" | tee -a ../logs/investigation_${CASE_ID}.txt
```

Key Information to Extract:

- **Registrar**: Who registered the domain
- **Registration Date**: Recently registered domains are suspicious
- **Registrant Organization**: Verify legitimacy
- **Country**: Does it match expected location?
- **Name Servers**: Hosting provider information

Step 3: DNS Record Investigation

bash

 # Check SPF record

echo "=== SPF RECORD ===" | tee -a ../logs/investigation_${CASE_ID}.txt

dig TXT "$SENDER_DOMAIN" +short | grep "v=spf1" | tee -a ../logs/investigation_${CASE_ID}.txt

echo "" | tee -a ../logs/investigation_${CASE_ID}.txt

 # Check DMARC record

echo "=== DMARC RECORD ===" | tee -a ../logs/investigation_${CASE_ID}.txt

dig TXT "_dmarc.$SENDER_DOMAIN" +short | tee -a ../logs/investigation_${CASE_ID}.txt

echo "" | tee -a ../logs/investigation_${CASE_ID}.txt

 # Check MX records

echo "=== MX RECORDS ===" | tee -a ../logs/investigation_${CASE_ID}.txt

dig MX "$SENDER_DOMAIN" +short | tee -a ../logs/investigation_${CASE_ID}.txt

echo "" | tee -a ../logs/investigation_${CASE_ID}.txt

 # Check A record

echo "=== A RECORD ===" | tee -a ../logs/investigation_${CASE_ID}.txt

```
dig A "$SENDER_DOMAIN" +short | tee -a ../logs/investigation_${CASE_ID}.txt
```

Step 4: Reverse DNS Lookup

```
bash
# Reverse DNS to verify IP reputation
echo "=== REVERSE DNS LOOKUP ===" | tee -a ../logs/investigation_${CASE_ID}.txt
dig -x "$ORIGIN_IP" +short | tee -a ../logs/investigation_${CASE_ID}.txt

# Check if reverse DNS matches domain
REVERSE_DNS=$(dig -x "$ORIGIN_IP" +short | head -1)
echo "Reverse DNS: $REVERSE_DNS"

if [[ "$REVERSE_DNS" == *"$SENDER_DOMAIN"* ]]; then
   echo "[PASS] Reverse DNS matches sender domain" | tee -a
../logs/investigation_${CASE_ID}.txt
else
   echo "[ALERT] Reverse DNS does NOT match sender domain" | tee -a
../logs/investigation_${CASE_ID}.txt
fi
```

Phase 5: Link & Attachment Analysis

Step 1: Extract All URLs from Email

```
bash
# Extract all URLs using regex
echo "=== EXTRACTED URLS ===" | tee ../logs/urls_${CASE_ID}.txt
grep -Eo 'https?://[^"<> ]+' "$EVIDENCE_FILE" | sort -u | tee -a ../logs/urls_${CASE_ID}.txt
```

Step 2: Analyze URLs for Malicious Indicators

```
bash
# Create URL analysis script
cat << 'EOF' > ../analyze_urls.sh
#!/bin/bash

URL_FILE="$1"

echo "======================================"
echo "URL ANALYSIS REPORT"
echo "======================================"
echo ""

while IFS= read -r url; do
    echo "Analyzing: $url"

    # Extract domain from URL
    domain=$(echo "$url" | sed -E 's|^https?://([^/]+).*|\1|')

    # Check if it's an IP address (suspicious)
    if [[ $domain =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
        echo "  [ALERT] URL uses IP address instead of domain"
    fi

    # Check for suspicious TLDs
    if [[ $domain =~ \.(tk|ml|ga|cf|gq|xyz|top)$ ]]; then
        echo "  [WARNING] Suspicious TLD detected"
    fi
```

```
    # Check for URL shorteners
    if [[ $domain =~ (bit\.ly|tinyurl|goo\.gl|t\.co|ow\.ly) ]]; then
        echo "  [INFO] URL shortener detected - investigate redirect"
    fi

    # Check domain length
    length=${#domain}
    if [ $length -gt 50 ]; then
        echo "  [WARNING] Unusually long domain name ($length chars)"
    fi

    # Check for typosquatting patterns
    if [[ $domain =~ (paypal|amazon|microsoft|google|apple|bank) ]]; then
        echo "  [WARNING] Contains brand name - verify legitimacy"
    fi

    echo ""
done < "$URL_FILE"
EOF

chmod +x ../analyze_urls.sh
../analyze_urls.sh ../logs/urls_${CASE_ID}.txt | tee ../logs/url_analysis_${CASE_ID}.txt
```

Step 3: VirusTotal Scanning (Online Analysis)

Note: Only submit hashes or URLs, never sensitive email content

bash

```
# Using VirusTotal CLI (requires API key)
# Get free API key from: https://www.virustotal.com/

# Check URL reputation
for url in $(cat ../logs/urls_${CASE_ID}.txt); do
    echo "Checking: $url"
    # curl -X POST 'https://www.virustotal.com/api/v3/urls' \
    #     -H 'x-apikey: YOUR_API_KEY' \
    #     --form "url=$url"
    # Manual alternative: Visit https://www.virustotal.com and paste URL
done
```

Step 4: Check URLs Against Threat Intelligence

bash

```
# Check against Cisco Talos reputation
# Visit: https://talosintelligence.com/reputation_center

# Check against URLhaus
echo "=== URLHAUS CHECK ===" | tee -a ../logs/investigation_${CASE_ID}.txt
for url in $(cat ../logs/urls_${CASE_ID}.txt); do
    curl -s "https://urlhaus-api.abuse.ch/v1/url/" \
        --data "url=$url" | jq '.' >> ../logs/investigation_${CASE_ID}.txt
done
```

Phase 6: Attachment Forensics

Step 1: Extract Attachments

bash

```
# Using munpack to extract attachments
mkdir -p ../evidence/attachments_${CASE_ID}
cd ../evidence/attachments_${CASE_ID}

munpack ../../evidence/"$EVIDENCE_FILE"

# List extracted files
echo "=== EXTRACTED ATTACHMENTS ===" | tee ../../logs/attachments_${CASE_ID}.txt
ls -lah | tee -a ../../logs/attachments_${CASE_ID}.txt
```

Step 2: Calculate Attachment Hashes

bash

```
# Generate hashes for all attachments
for file in *; do
    if [ -f "$file" ]; then
        echo "File: $file" | tee -a ../../logs/attachment_hashes_${CASE_ID}.txt
        md5sum "$file" | tee -a ../../logs/attachment_hashes_${CASE_ID}.txt
        sha256sum "$file" | tee -a ../../logs/attachment_hashes_${CASE_ID}.txt
        echo "" | tee -a ../../logs/attachment_hashes_${CASE_ID}.txt
    fi
done
```

Step 3: File Type Verification

bash

```
# Verify actual file type (detect extension spoofing)
echo "=== FILE TYPE ANALYSIS ===" | tee ../../logs/file_types_${CASE_ID}.txt
for file in *; do
    if [ -f "$file" ]; then
        echo "File: $file" | tee -a ../../logs/file_types_${CASE_ID}.txt
        file "$file" | tee -a ../../logs/file_types_${CASE_ID}.txt

        # Check for double extensions (e.g., document.pdf.exe)
        if [[ "$file" =~ \.[^.]+\.[^.]+$ ]]; then
            echo "  [ALERT] Double extension detected!" | tee -a
../../logs/file_types_${CASE_ID}.txt
        fi

        echo "" | tee -a ../../logs/file_types_${CASE_ID}.txt
    fi
done
```

Step 4: Static Malware Analysis

```
bash
# Check for known malware signatures
echo "=== MALWARE SIGNATURE SCAN ===" | tee
../../logs/malware_scan_${CASE_ID}.txt

# Update ClamAV database
sudo freshclam

# Scan attachments
clamscan -r . | tee -a ../../logs/malware_scan_${CASE_ID}.txt
```

```
# Check strings in suspicious files
for file in *.exe *.dll *.scr *.bat *.ps1; do
    if [ -f "$file" ]; then
        echo "Strings from: $file" | tee -a ../../logs/strings_${CASE_ID}.txt
        strings "$file" | head -100 | tee -a ../../logs/strings_${CASE_ID}.txt
        echo "" | tee -a ../../logs/strings_${CASE_ID}.txt
    fi
done 2>/dev/null
```

Phase 7: Email Body Content Analysis

Step 1: Extract Email Body

bash

```
# Extract plain text body
cd ~/Forensics/Phishing_Cases/evidence

# Using formail to extract body
formail -I "" < "$EVIDENCE_FILE" | sed '1,/^$/d' > ../logs/body_${CASE_ID}.txt

# Or use grep
sed -n '/^$/,$p' "$EVIDENCE_FILE" | tail -n +2 > ../logs/body_alt_${CASE_ID}.txt
```

Step 2: Analyze Body for Phishing Indicators

bash

```
# Create phishing indicator checker
cat << 'EOF' > ../check_phishing_indicators.sh
#!/bin/bash
```

```
BODY_FILE="$1"

echo "===================================="
echo "PHISHING INDICATOR ANALYSIS"
echo "===================================="
echo ""

# Check for urgency language
echo "--- URGENCY INDICATORS ---"
grep -i -E "urgent|immediately|action required|verify now|suspended|expires|24 hours|confirm|validate" "$BODY_FILE"
echo ""

# Check for financial threats
echo "--- FINANCIAL THREATS ---"
grep -i -E "account.*locked|account.*suspended|unusual activity|unauthorized|payment.*failed|refund" "$BODY_FILE"
echo ""

# Check for credential requests
echo "--- CREDENTIAL REQUESTS ---"
grep -i -E "password|username|credit card|social security|verify.*identity|confirm.*account" "$BODY_FILE"
echo ""

# Check for call-to-action
echo "--- CALL TO ACTION ---"
```

```
grep -i -E "click here|download|login|verify|update.*information|confirm" "$BODY_FILE"

echo ""


# Check for sender impersonation

echo "--- IMPERSONATION ATTEMPTS ---"

grep -i -E "we are|from.*team|support.*team|security.*team|admin|customer.*service"
"$BODY_FILE"

echo ""


echo "====================================="

EOF


chmod +x ../check_phishing_indicators.sh

../check_phishing_indicators.sh ../logs/body_${CASE_ID}.txt | tee
../logs/phishing_indicators_${CASE_ID}.txt
```

Phase 8: Timeline Reconstruction

Step 1: Build Email Journey Timeline

bash

```
# Extract and parse Received headers in chronological order

cat << 'EOF' > ../build_timeline.sh

#!/bin/bash


HEADER_FILE="$1"


echo "====================================="

echo "EMAIL JOURNEY TIMELINE"

echo "====================================="
```

```
echo ""
echo "Format: [Timestamp] Server -> Next Server (IP)"
echo ""

grep -i "^Received:" "$HEADER_FILE" | tac | nl -v 1 | while read num line; do
    echo "Hop $num:"

    # Extract timestamp
    timestamp=$(echo "$line" | grep -oE '[A-Z][a-z]{2}, [0-9]{1,2} [A-Z][a-z]{2} [0-9]{4} [0-9]{2}:[0-9]{2}:[0-9]{2}.*')
    echo "  Time: $timestamp"

    # Extract from server
    from_server=$(echo "$line" | grep -oE 'from [^ ]+' | head -1)
    echo "  $from_server"

    # Extract IP if present
    ip=$(echo "$line" | grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' | head -1)
    if [ -n "$ip" ]; then
        echo "  IP: $ip"
    fi

    echo ""
done

echo "====================================="
EOF
```

chmod +x ../build_timeline.sh

../build_timeline.sh ../logs/headers_${CASE_ID}.txt | tee ../logs/timeline_${CASE_ID}.txt

Phase 9: Generate Forensic Report

Step 1: Compile All Findings

bash

 # Create comprehensive report

cat << EOF > ../reports/FORENSIC_REPORT_${CASE_ID}.md

📊 PHISHING EMAIL FORENSIC ANALYSIS REPORT

Case ID: ${CASE_ID}

Date of Analysis: $(date)

Analyst: $(whoami)

Status: COMPLETED

📋 EXECUTIVE SUMMARY

This report documents the forensic analysis of a suspected phishing email. The investigation followed industry-standard chain of custody procedures and analyzed email headers, authentication records, sender reputation, URLs, and attachments.

🔍 EVIDENCE INFORMATION

Evidence File: \`${EVIDENCE_FILE}\`

File Size: $(stat -f%z "../evidence/${EVIDENCE_FILE}" 2>/dev/null || stat -c%s "../evidence/${EVIDENCE_FILE}")

MD5 Hash: \`$(cat ../evidence/${EVIDENCE_FILE}.md5 | cut -d' ' -f1)\`

SHA256 Hash: \`$(cat ../evidence/${EVIDENCE_FILE}.sha256 | cut -d' ' -f1)\`

📧 HEADER ANALYSIS

Sender Information

\`\`\`

$(grep -i "^From:" ../logs/headers_${CASE_ID}.txt)

$(grep -i "^Return-Path:" ../logs/headers_${CASE_ID}.txt)

$(grep -i "^Reply-To:" ../logs/headers_${CASE_ID}.txt)

\`\`\`

Authentication Results

\`\`\`

$(cat ../logs/auth_results_${CASE_ID}.txt)

\`\`\`

🌐 DOMAIN INTELLIGENCE

Sender Domain: ${SENDER_DOMAIN}

Origin IP: ${ORIGIN_IP}

DNS Records

\`\`\`

$(grep -A 5 "=== SPF RECORD ===" ../logs/investigation_${CASE_ID}.txt | tail -4)

\`\`\`

🔗 URL ANALYSIS

Extracted URLs

\`\`\`

$(cat ../logs/urls_${CASE_ID}.txt)

\`\`\`

Suspicious Indicators

\`\`\`

$(cat ../logs/url_analysis_${CASE_ID}.txt | grep "\[ALERT\]\|\[WARNING\]")

\`\`\`

📎 ATTACHMENT ANALYSIS

$(if [-f ../logs/attachments_${CASE_ID}.txt]; then
    echo "### Extracted Attachments"
    echo "\`\`\`"
    cat ../logs/attachments_${CASE_ID}.txt
    echo "\`\`\`"
else
    echo "**No attachments found**"
fi)

🚨 THREAT INDICATORS (IOCs)

Indicators of Compromise
- **Sender Domain**: \`${SENDER_DOMAIN}\`
- **Origin IP**: \`${ORIGIN_IP}\`
- **URLs**: See URL Analysis section
- **File Hashes**: See Attachment Analysis section

📊 RISK ASSESSMENT

Overall Risk Level: [TO BE DETERMINED]

Category	Finding	Risk Level
SPF Authentication	$(grep -qi "spf=pass" ../logs/auth_results_${CASE_ID}.txt && echo "PASS" || echo "FAIL")	$(grep -qi "spf=pass" ../logs/auth_results_${CASE_ID}.txt && echo "LOW" || echo "HIGH")
DKIM Authentication	$(grep -qi "dkim=pass" ../logs/auth_results_${CASE_ID}.txt && echo "PASS" || echo "FAIL")	$(grep -qi "dkim=pass" ../logs/auth_results_${CASE_ID}.txt && echo "LOW" || echo "HIGH")
DMARC Authentication	$(grep -qi "dmarc=pass" ../logs/auth_results_${CASE_ID}.txt && echo "PASS" || echo "FAIL")	$(grep -qi "dmarc=pass" ../logs/auth_results_${CASE_ID}.txt && echo "LOW" || echo "HIGH")
URLs	$([-f ../logs/urls_${CASE_ID}.txt] && echo "Detected" || echo "None")	$(grep -q "\[ALERT\]" ../logs/url_analysis_${CASE_ID}.txt 2>/dev/null && echo "HIGH" || echo "MEDIUM")
Attachments	$([-f ../logs/attachments_${CASE_ID}.txt] && echo "Detected" || echo "None")	TBD

💡 RECOMMENDATIONS

1. **Block sender domain** at email gateway
2. **Add IP to blocklist**: \`${ORIGIN_IP}\`
3. **Update email filtering rules** to catch similar patterns
4. **User awareness training** on phishing indicators
5. **Implement DMARC policy** if not already configured

📚 REFERENCES

- NIST Cybersecurity Framework
- ISO 27001:2013 Annex A Controls
- RFC 7208 (SPF)
- RFC 6376 (DKIM)
- RFC 7489 (DMARC)