Email Threat Intelligence & IOC Extraction Tool

Name: Hamid Ali

Project Type: Cybersecurity / SOC Analyst Project

Platform: Windows

Language: Python

1. Project Overview

This project focuses on building a Python-based Email Threat Intelligence and IOC (Indicators of Compromise) Extraction Tool. The tool analyzes .eml email files to extract suspicious artifacts such as IP addresses, URLs, domains, email addresses, cryptographic hashes, and cryptocurrency wallet addresses. The project is designed for SOC Analysts to assist in phishing and malware investigations.

2. Objectives

- Understand email (.eml) structure
- Parse email headers and body
- Extract Indicators of Compromise (IOCs)
- Defang malicious indicators for safe analysis
- Generate hashes for email attachments

3. Tools & Technologies Used

- Operating System: Windows 10
- Programming Language: Python 3.14
- Python Libraries: re, hashlib, ipaddress, email
- Editor: VS Code / Notepad

4. Environment Setup (From Scratch)

Step 1: Install Python using CMD
Command:
winget install Python.Python.3.14

Step 2: Verify Installation
python --version
pip --version

5. Creating Sample Email File

Create a file named sample.eml using Notepad and save it as 'All Files'. This file simulates a phishing email containing malicious indicators.

sample.eml

From: Attacker <attacker@evil-domain.com>

To: Hamid <hamid@example.com>

Subject: Urgent Account Verification Required

Date: Tue, 13 Jan 2026 10:15:00 +0500

Message-ID: <1234567890@evil-domain.com>

Reply-To: support@evil-domain.com

Return-Path: <bounce@evil-domain.com>

Received: from mail.evil-domain.com (185.199.110.153)

Received: from unknown (192.168.1.50)

Hello Hamid,

Your account has been compromised.

Please verify immediately by clicking the link below:

http://malicious-site.com/login.php?user=hamid

If you do not act now, your account will be suspended.

For support contact: support@evil-domain.com

Bitcoin payment address:

1BoatSLRHtKNngkdXEeobR76b53LETtpyT

Malicious IPs observed:

185.199.110.153

8.8.8.8

Sample hash:

d41d8cd98f00b204e9800998ecf8427e

Regards,

Security Team

6. Project Source Code

email_ioc_extractor.py

```
import re
import sys
import hashlib
import ipaddress
from email.parser import BytesParser

def read_file(file_path):
    with open(file_path, 'rb') as file:
        content = file.read()
    parser = BytesParser()
    return parser.parsebytes(content)

def extract_iocs(email_message):
    iocs = {
        "ips": set(),
        "urls": set(),
        "emails": set(),
        "domains": set(),
        "btc_addresses": set(),
        "hashes": set()
    }

    patterns = {
        "ip": r'\b(?:\d{1,3}\.){3}\d{1,3}\b',
        "url": r'https?:\/\/(?:[\w\-]+\.)+[a-z]{2,}(?:\/[\w\-\.\/?%&=]*)?',
        "email": r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}',
        "domain": r'\b(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,}\b',
```

```
        "btc": r'\b[13][a-km-zA-HJ-NP-Z1-9]{25,34}\b',
        "hash": r'\b[a-fA-F0-9]{32,64}\b'
    }

    for header_name, header_value in email_message.items():
        iocs["ips"].update(re.findall(patterns["ip"], header_value))
        iocs["emails"].update(re.findall(patterns["email"], header_value))

    for part in email_message.walk():
        if part.get_content_type() in ["text/plain", "text/html"]:
            payload = part.get_payload(decode=True)
            if isinstance(payload, bytes):
                payload = payload.decode("utf-8", errors="ignore")

            for key, pattern in patterns.items():
                iocs.setdefault(key + "s", set()).update(re.findall(pattern, payload))

    iocs["ips"] = list(set([ip for ip in iocs["ips"] if validate_ip(ip)]))
    return iocs

def validate_ip(ip):
    try:
        return bool(ipaddress.ip_address(ip))
    except ValueError:
        return False

def defang(text):
    return text.replace(".", "[.]").replace("http", "hxxp")
```

```
def extract_headers(email_message):
    headers_to_extract = ["Date", "Subject", "To", "From", "Reply-To", "Return-Path", "Message-ID"]
    return {key: email_message[key] for key in headers_to_extract if key in email_message}

def extract_attachments(email_message):
    attachments = []
    for part in email_message.walk():
        if part.get_content_maintype() == 'multipart' or not part.get_filename():
            continue

        filename = part.get_filename()
        data = part.get_payload(decode=True)
        attachments.append({
            "filename": filename,
            "md5": hashlib.md5(data).hexdigest(),
            "sha1": hashlib.sha1(data).hexdigest(),
            "sha256": hashlib.sha256(data).hexdigest()
        })
    return attachments

def main(file_path):
    email_message = read_file(file_path)
    iocs = extract_iocs(email_message)
    headers = extract_headers(email_message)
    attachments = extract_attachments(email_message)

    print("\nExtracted Headers:")
```

```
    for key, value in headers.items():
        print(f"{key}: {value}")

    print("\nExtracted IOCs:")
    for key, values in iocs.items():
        print(f"{key.capitalize()}: ")
        for value in values:
            print(f"  - {defang(value)}")

    if attachments:
        print("\nExtracted Attachments:")
        for attachment in attachments:
            print(f"Filename: {attachment['filename']}")
            print(f"  MD5: {attachment['md5']}")
            print(f"  SHA1: {attachment['sha1']}")
            print(f"  SHA256: {attachment['sha256']}")
def usage():
    print(f"Usage: python {sys.argv[0]} <file_path>")
    sys.exit(1)

if __name__ == "__main__":
    if len(sys.argv) != 2:
        usage()
    main(sys.argv[1])
```

7. Running the Project

Open Command Prompt and navigate to the project folder:

Run the script:
python email_ioc_extractor.py sample.eml

8. Output Explanation

The script displays extracted email headers and defanged IOCs. These indicators can be used for threat intelligence, SIEM ingestion, or malware analysis.

Output

Extracted Headers:

Date: Tue, 13 Jan 2026 10:15:00 +0500

Subject: Urgent Account Verification Required

To: Hamid <hamid@example.com>

From: Attacker <attacker@evil-domain.com>

Reply-To: support@evil-domain.com

Return-Path: <bounce@evil-domain.com>

Message-ID: <1234567890@evil-domain.com>

Extracted IOCs:

Ips:

- 192[.]168[.]1[.]50
- 185[.]199[.]110[.]153
- 8[.]8[.]8[.]8

Urls:

- hxxp://malicious-site[.]com/login[.]php?user=hamid

Emails:

- attacker@evil-domain[.]com
- hamid@example[.]com
- 1234567890@evil-domain[.]com
- bounce@evil-domain[.]com
- support@evil-domain[.]com

Domains:

- login[.]php
- malicious-site[.]com

- evil-domain[.]com

Btc_addresses:

Hashes:

Btcs:

- 1BoatSLRHtKNngkdXEeobR76b53LETtpyT

Hashs:

- d41d8cd98f00b204e9800998ecf8427e

9. SOC Analyst Use Case

This tool assists SOC analysts in:
- Phishing email investigation
- Malware IOC identification
- Threat intelligence enrichment
- Incident response triage

10. Conclusion

This project demonstrates practical SOC Analyst skills including email analysis, IOC extraction, and secure handling of malicious indicators using Python.