NED University of Engineering and Technology

CYBER SECURITY LAWS AND REGULATIONS (CT-374)

Advanced Email OSINT Machine

PROJECT REPORT

Submitted By:

Muhammad Abdullah Ayub CR-042

Muhammad Hasan Khan CR-047

Muhammad Imaad Tahir CR-014

Table Of Contents

Contents Page

1. Introduction

Open-Source Intelligence (OSINT) refers to the process of gathering, analyzing, and interpreting publicly available information from legitimate sources. This project focuses on **Email OSINT**, a subcategory dedicated to extracting intelligence associated with email addresses. The Advanced Email OSINT Machine automates multiple investigation steps to provide both technical and behavioral insights into the email being analyzed.

2. What is OSINT?

OSINT (Open-Source Intelligence) is one of the most important frameworks used in cybersecurity, intelligence agencies, digital forensics, and investigative journalism. OSINT relies on data that is legally and publicly accessible, such as:

- DNS records
- Search engines
- Public APIs
- Web directories
- Social media public metadata
- Cyber breach databases

OSINT helps identify risks, understand digital footprints, detect fraud, track threat actors, and validate identities.

3. Email-based OSINT Overview

Email-based OSINT primarily focuses on analyzing an email address from both a technical and a behavioral perspective. The goal is to understand the credibility, exposure, and digital presence behind an email. This type of OSINT is frequently used for cybersecurity investigations, phishing detection, fraud prevention, HR background checks, penetration testing, and digital identity verification.

A full email OSINT workflow reveals details such as:

- Whether the email is valid or fabricated
- If it has appeared in data breaches
- The domain's server configuration (MX, SPF, DMARC)
- The trustworthiness of the domain's mail infrastructure
- Publicly available profile photos (via Gravatar)
- Behavioral indicators such as reputation and exposure history

4. Project Objective

The objective of this project is to design and implement a complete Email OSINT tool capable of delivering a professional-grade investigation report.

The system aims to:

- Validate email format
- Extract and analyze the domain
- Perform DNS-OSINT
- Detect if the email is exposed in cyber breaches
- Identify Gravatar public profiles
- Conduct reputation analysis
- Output results in JSON and human-readable summary formats

The tool is designed for academic, ethical hacking, cybersecurity training, and research purposes.

5. System Architecture

The system follows a modular architecture with the following components:

1. Input Handling Module: Accepts email from the user.
2. Email Validation Module: Ensures email follows RFC format.
3. Domain Extraction Module: Retrieves domain from email.
4. DNS Intelligence Module Performs: A/MX/SPF/DMARC lookups.
5. Breach Detection Module: Uses HIBP API to check leaks.
6. Reputation Analysis Module: Evaluates trust indicators.
7. Gravatar Lookup Module: Finds linked public profiles.
8. Output Formatting Module: Generates detailed reports.

6. Explanation of Functions and Modules

6.1 validate_email_format()

Ensures the email is syntactically correct using regex. Prevents invalid inputs from reaching deeper OSINT modules.

6.2 get_domain()

Extracts domain part of the email. Critical for DNS checks and domain-based intelligence.

6.3 reputation_emailrep()

Fallback reputation system analyzing:

- DNS functionality
- MX validity
- Blacklist presence

Produces an 'overall' trust score.

6.4 hibp_breach_lookup()

Integrates HaveIBeenPwned data breach intelligence. Determines if the target email has been leaked on the dark web or public breaches.

6.5 gravatar_lookup()

Queries Gravatar’s MD5-hashed email database to find associated public display names or profile photos.

6.6 dns_osint()

Performs DNS-level investigation including:

- IP mapping
- Mail server discovery
- Security policy checks (SPF, DMARC)

7. API Integrations Theory

HaveIBeenPwned (HIBP):

- World's largest breach intelligence database.
- Identifies leaked passwords, emails, and compromised accounts.

EmailRep:

- Provides reputation analysis using machine learning.
- Since our tool is academic, a fallback engine is used.

8. DNS Intelligence Theory

DNS (Domain Name System) is the backbone of email communication. DNS-based OSINT exposes:

- Technical misconfigurations
- Spoofing vulnerabilities
- Server infrastructure SPF prevents unauthorized senders.

DMARC ensures domain-level authentication policies.

9. Gravatar Intelligence Theory

Gravatar links email addresses to globally recognized avatars. If a user has ever created a Gravatar account, digital identity clues can be extracted from public metadata.

10. Reputation Analysis Theory

Reputation analysis evaluates multiple indicators:

- Technical legitimacy
- DNS security
- Breach exposure
- Blacklist presence
- Domain trust score

The combination creates a multi-layer threat score used in cybersecurity screening.

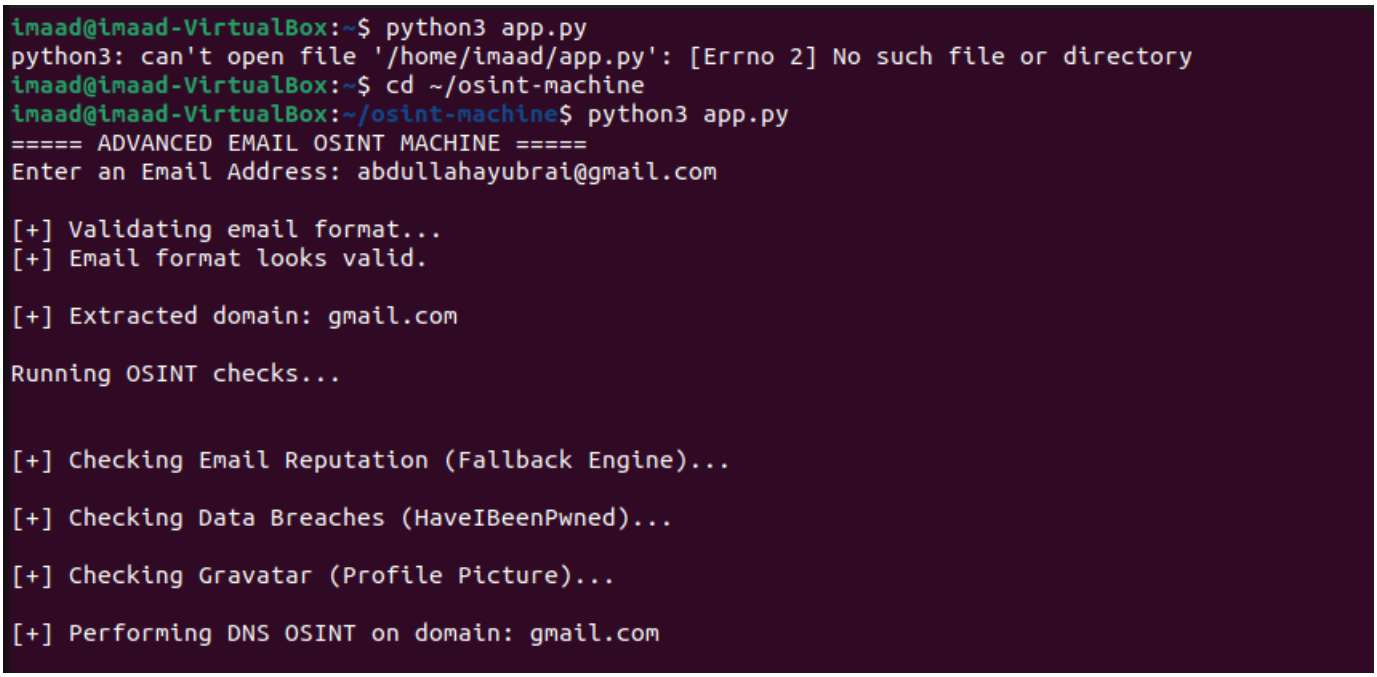
11. Workflow of the Program

The complete workflow:

1. User enters email
2. Email validated
3. Domain extracted
4. DNS OSINT executed
5. Breach lookup performed
6. Gravatar lookup executed
7. Reputation score calculated
8. JSON output built
9. Human-friendly report generated

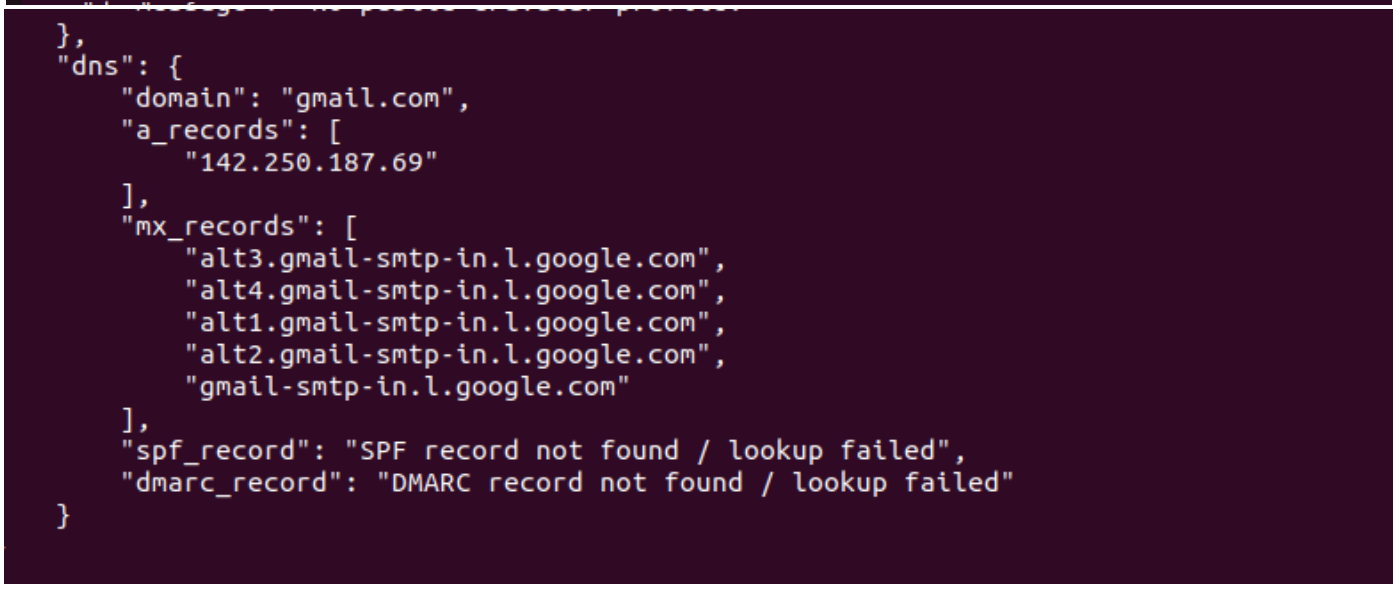
12. Screenshots

Tool Running:



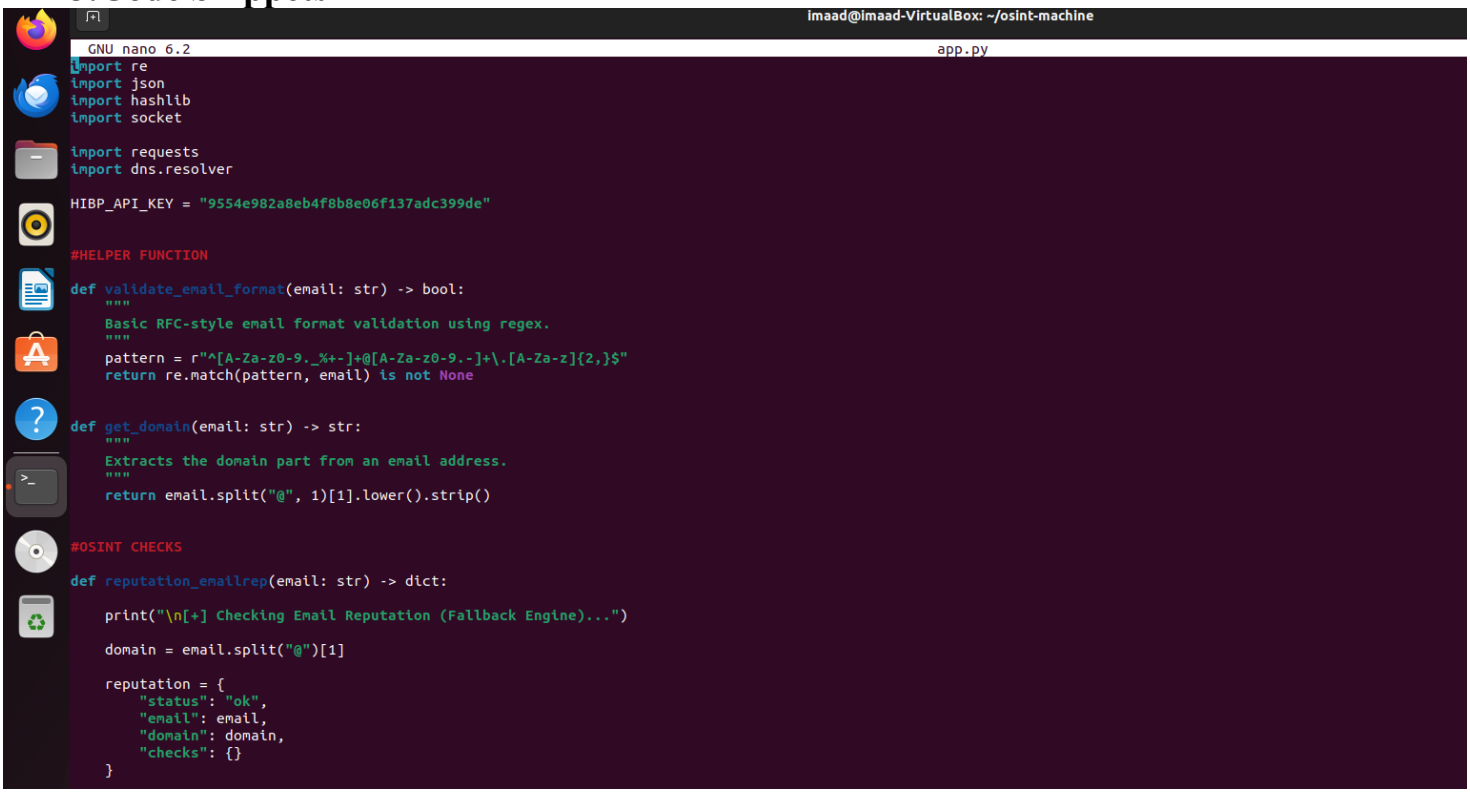
JSON Output:

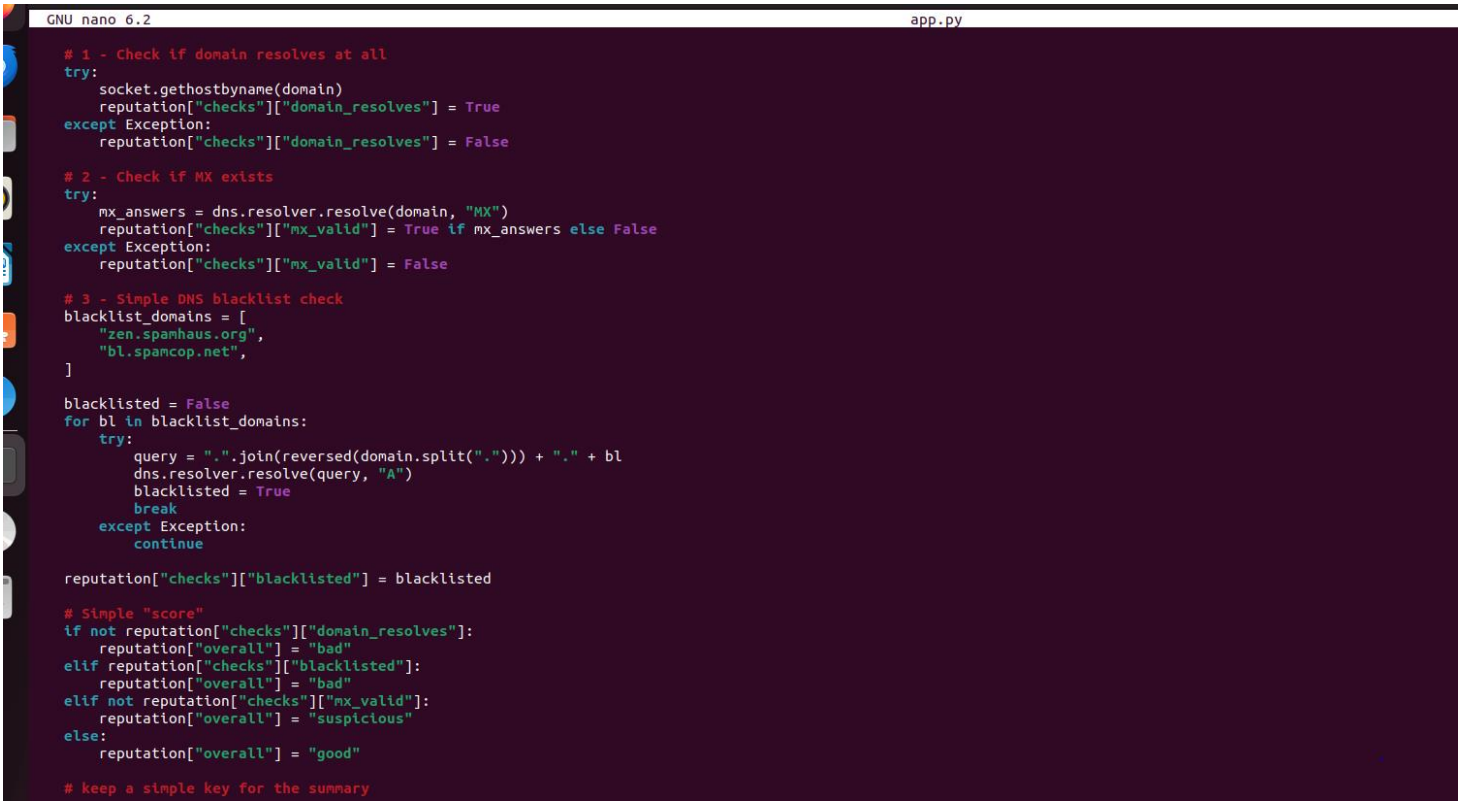

========== RESULTS ==========
{
"email": "abdullahayubrai@gmail.com",
"domain": "gmail.com",
"format_valid": true,
"reputation_emailrep": {
"status": "ok",
"email": "abdullahayubrai@gmail.com",
"domain": "gmail.com",
"checks": {
"domain_resolves": true,
"mx_valid": true,
"blacklisted": false
},
"overall": "good",
"reputation": "good"
},
"breach_info_hibp": {
"status": "breached",
"details": [
{
"Name": "MemeChat"
},
{
"Name": "OperationEndgame2"
}
]
},
"gravatar": {
"status": "not_found",
"message": "No public Gravatar profile."
},

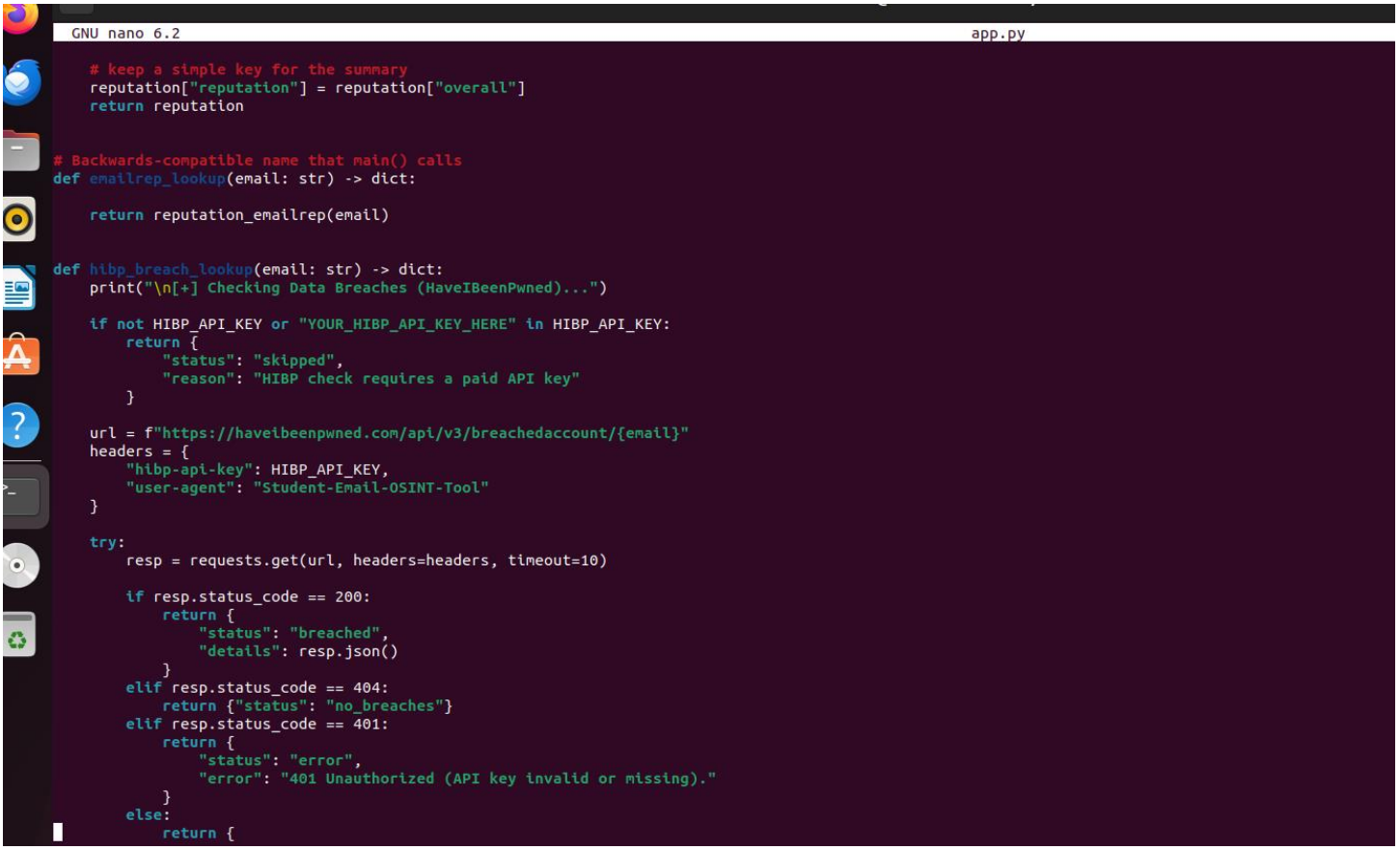

},
"dns": {
"domain": "gmail.com",
"a_records": [
"142.250.187.69"
],
"mx_records": [
"alt3.gmail-smtp-in.l.google.com",
"alt4.gmail-smtp-in.l.google.com",
"alt1.gmail-smtp-in.l.google.com",
"alt2.gmail-smtp-in.l.google.com",
"gmail-smtp-in.l.google.com"
],
"spf_record": "SPF record not found / lookup failed",
"dmarc_record": "DMARC record not found / lookup failed"
}


========== SUMMARY ==========
[+] Email reputation (fallback): good
[!] HIBP: Account appears in known data breaches.
[-] Gravatar: No public profile.
[+] A records: ['142.250.187.69']
[+] MX records: ['alt3.gmail-smtp-in.l.google.com', 'alt4.gmail-smtp-in.l.google.com', 'alt1.gmail-smtp-in.l.google.com', 'alt2.gmail-smtp-in.l.google.com', 'gmail-smtp-in.l.google.com']
[+] SPF: SPF record not found / lookup failed
[+] DMARC: DMARC record not found / lookup failed
Activate Windows
Go to Settings to activate Win
===== END OF REPORT =====
imaad@imaad-VirtualBox:~/osint-machine$

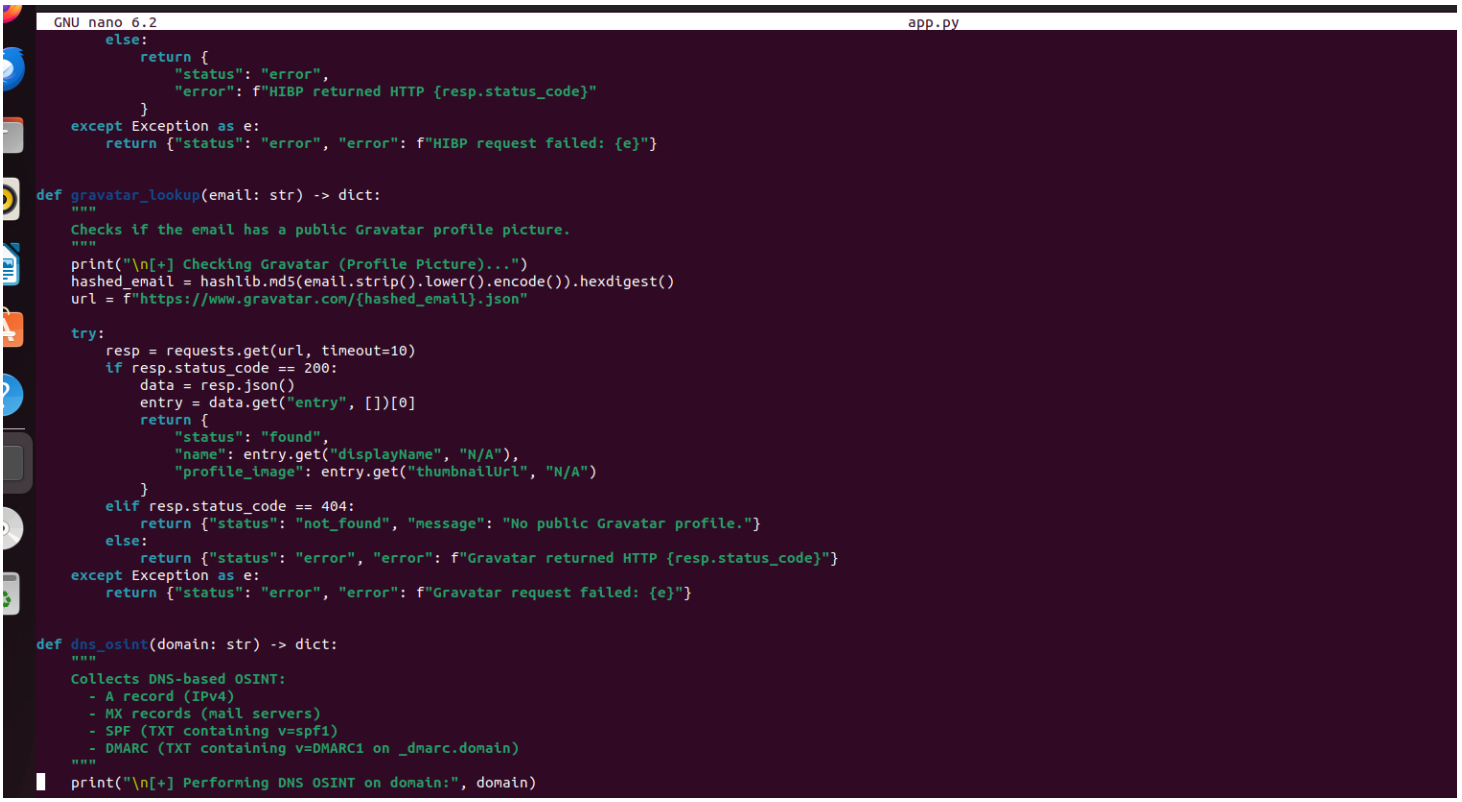
13.Code Snippets





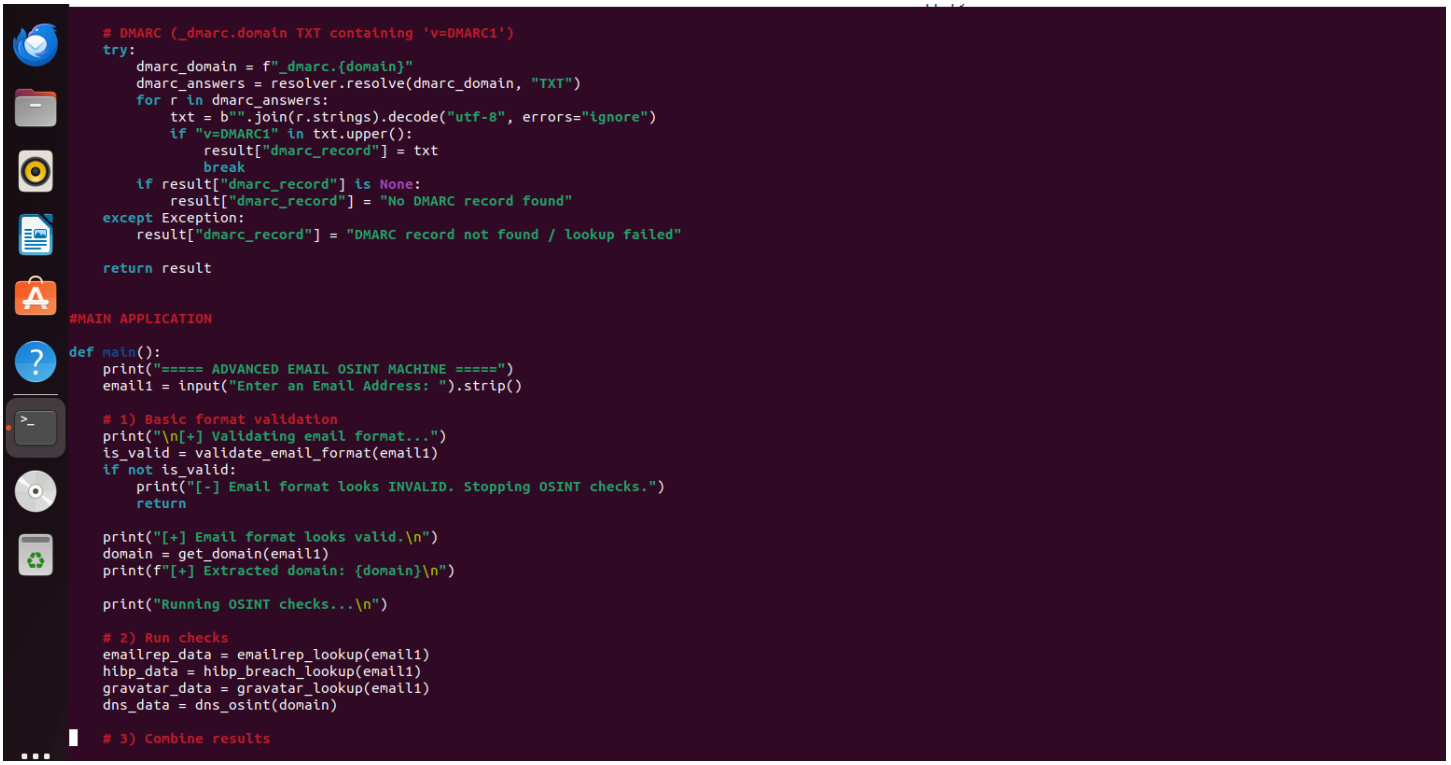


```
GNU nano 6.2                                                    app.py

    # keep a simple key for the summary
    reputation["reputation"] = reputation["overall"]
    return reputation


# Backwards-compatible name that main() calls
def emailrep_lookup(email: str) -> dict:

    return reputation_emailrep(email)


def hibp_breach_lookup(email: str) -> dict:
    print("\n[+] Checking Data Breaches (HaveIBeenPwned)...")

    if not HIBP_API_KEY or "YOUR_HIBP_API_KEY_HERE" in HIBP_API_KEY:
        return {
            "status": "skipped",
            "reason": "HIBP check requires a paid API key"
        }

    url = f"https://haveibeenpwned.com/api/v3/breachedaccount/{email}"
    headers = {
        "hibp-api-key": HIBP_API_KEY,
        "user-agent": "Student-Email-OSINT-Tool"
    }

    try:
        resp = requests.get(url, headers=headers, timeout=10)

        if resp.status_code == 200:
            return {
                "status": "breached",
                "details": resp.json()
            }
        elif resp.status_code == 404:
            return {"status": "no_breaches"}
        elif resp.status_code == 401:
            return {
                "status": "error",
                "error": "401 Unauthorized (API key invalid or missing)."
            }
        else:
            return {
```



```
GNU nano 6.2                                                    app.py

        else:
            return {
                "status": "error",
                "error": f"HIBP returned HTTP {resp.status_code}"
            }
    except Exception as e:
        return {"status": "error", "error": f"HIBP request failed: {e}"}


def gravatar_lookup(email: str) -> dict:
    """
    Checks if the email has a public Gravatar profile picture.
    """
    print("\n[+] Checking Gravatar (Profile Picture)...")
    hashed_email = hashlib.md5(email.strip().lower().encode()).hexdigest()
    url = f"https://www.gravatar.com/{hashed_email}.json"

    try:
        resp = requests.get(url, timeout=10)
        if resp.status_code == 200:
            data = resp.json()
            entry = data.get("entry", [])[0]
            return {
                "status": "found",
                "name": entry.get("displayName", "N/A"),
                "profile_image": entry.get("thumbnailUrl", "N/A")
            }
        elif resp.status_code == 404:
            return {"status": "not_found", "message": "No public Gravatar profile."}
        else:
            return {"status": "error", "error": f"Gravatar returned HTTP {resp.status_code}"}
    except Exception as e:
        return {"status": "error", "error": f"Gravatar request failed: {e}"}


def dns_osint(domain: str) -> dict:
    """
    Collects DNS-based OSINT:
      - A record (IPv4)
      - MX records (mail servers)
      - SPF (TXT containing v=spf1)
      - DMARC (TXT containing v=DMARC1 on _dmarc.domain)
    """
    print("\n[+] Performing DNS OSINT on domain:", domain)
```


GNU nano 6.2
app.py
"""
print("\n[+] Performing DNS OSINT on domain:", domain)
result = {
"domain": domain,
"a_records": [],
"mx_records": [],
"spf_record": None,
"dmarc_record": None
}
# A record
try:
ip = socket.gethostbyname(domain)
result["a_records"].append(ip)
except Exception:
result["a_records"].append("No A record / DNS lookup failed")
resolver = dns.resolver.Resolver()
# MX records
try:
mx_answers = resolver.resolve(domain, "MX")
result["mx_records"] = [str(r.exchange).rstrip(".") for r in mx_answers]
except Exception:
result["mx_records"].append("No MX records found or lookup failed")
# SPF (TXT containing 'v=spf1')
try:
txt_answers = resolver.resolve(domain, "TXT")
for r in txt_answers:
txt = b"".join(r.strings).decode("utf-8", errors="ignore")
if "v=spf1" in txt.lower():
result["spf_record"] = txt
break
if result["spf_record"] is None:
result["spf_record"] = "SPF record not found / lookup failed"
except Exception:
result["spf_record"] = "SPF record not found / lookup failed"
# DMARC (_dmarc.domain TXT containing 'v=DMARC1')
try:
dmarc_domain = f"_dmarc.{domain}"
dmarc_answers = resolver.resolve(dmarc_domain, "TXT")
for r in dmarc_answers:


# DMARC (_dmarc.domain TXT containing 'v=DMARC1')
try:
dmarc_domain = f"_dmarc.{domain}"
dmarc_answers = resolver.resolve(dmarc_domain, "TXT")
for r in dmarc_answers:
txt = b"".join(r.strings).decode("utf-8", errors="ignore")
if "v=DMARC1" in txt.upper():
result["dmarc_record"] = txt
break
if result["dmarc_record"] is None:
result["dmarc_record"] = "No DMARC record found"
except Exception:
result["dmarc_record"] = "DMARC record not found / lookup failed"
return result
#MAIN APPLICATION
def main():
print("===== ADVANCED EMAIL OSINT MACHINE =====")
email1 = input("Enter an Email Address: ").strip()
# 1) Basic format validation
print("\n[+] Validating email format...")
is_valid = validate_email_format(email1)
if not is_valid:
print("[-] Email format looks INVALID. Stopping OSINT checks.")
return
print("[+] Email format looks valid.\n")
domain = get_domain(email1)
print(f"[+] Extracted domain: {domain}\n")
print("Running OSINT checks...\n")
# 2) Run checks
emailrep_data = emailrep_lookup(email1)
hibp_data = hibp_breach_lookup(email1)
gravatar_data = gravatar_lookup(email1)
dns_data = dns_osint(domain)
# 3) Combine results

```
# 3) Combine results
results = {
    "email": email1,
    "domain": domain,
    "format_valid": is_valid,
    "reputation_emailrep": emailrep_data,
    "breach_info_hibp": hibp_data,
    "gravatar": gravatar_data,
    "dns": dns_data
}

# 4) Pretty JSON output
print("\n========== RESULTS  ==========")
print(json.dumps(results, indent=4))

# 5) Human-readable summary
print("\n==========  SUMMARY ==========")

# Email reputation quick verdict
if isinstance(emailrep_data, dict) and emailrep_data.get("reputation"):
    print(f"[+] Email reputation (fallback): {emailrep_data['reputation']}")
else:
    print("[!] Email reputation: Not available (error or rate limit).")

# HIBP
hibp_status = hibp_data.get("status", "unknown")
if hibp_status == "breached":
    print("[!] HIBP: Account appears in known data breaches.")
elif hibp_status == "no_breaches":
    print("[+] HIBP: No breaches reported.")
elif hibp_status == "skipped":
    print("[!] HIBP: Skipped (no API key configured) - explain this in your report.")
else:
    print(f"[!] HIBP: {hibp_data.get('error', 'Unknown error')}")

# Gravatar
if gravatar_data.get("status") == "found":
    print("[+] Gravatar: Public profile found.")
elif gravatar_data.get("status") == "not_found":
    print("[-] Gravatar: No public profile.")
else:
    print(f"[!] Gravatar: {gravatar_data.get('error', 'Unknown error')}")
```

```
        print("[-] Gravatar: No public profile.")
    else:
        print(f"[!] Gravatar: {gravatar_data.get('error', 'Unknown error')}")

    # DNS summary
    print(f"[+] A records: {dns_data.get('a_records')}")
    print(f"[+] MX records: {dns_data.get('mx_records')}")
    print(f"[+] SPF: {dns_data.get('spf_record')}")
    print(f"[+] DMARC: {dns_data.get('dmarc_record')}")

    print("\n===== END OF REPORT =====")


if __name__ == "__main__":
    main()
```

14. Conclusion

The Advanced Email OSINT Machine demonstrates how OSINT techniques can be applied in cybersecurity investigations. By combining DNS intelligence, breach monitoring, and open-source profile discovery, the system effectively produces a comprehensive digital footprint for any input email. This project showcases the integration of theory, security best practices, and Python automation.