Project Report on

OSINT Automation Tool

Abstract

This project presents the design and implementation of an "OSINT Intelligence Agent," a comprehensive web application for cybersecurity reconnaissance. Developed in Python using the Streamlit framework, the tool provides a user-friendly graphical interface for performing Open Source Intelligence (OSINT) gathering on a given domain or IP address. The agent integrates multiple reconnaissance techniques, including **WHOIS lookups**, **DNS record enumeration**, **subdomain discovery**, **email breach checking** via the XposedOrNot API, and **advanced port scanning** using the `nmap` library. Key features include a rule-based risk analysis engine that calculates a security score based on domain age, TLD suspicion, and exposed services. The system also performs **port security analysis** to identify high-risk services and provides actionable recommendations. Data is visualized through interactive Plotly charts, a Folium-based geolocation map, and a detailed dashboard. Finally, the agent ca compile all findings into a **downloadable PDF report**, automating a significant portion of the initial intelligence-gathering phase for security professionals.

Key words*: *Ethical Hacking, Threat Intelligence, Information Security, Cyber Forensics, WHOIS, DNS, Email Breaching, Advanced Port Scanning, OSINT, Cyber Security, Automation.

Contents

List of Figures

Nomenclature

A list of key terms and acronyms used throughout this report.

- API (Application Programming Interface): A set of rules that allows different software applications to communicate with each other (e.g., XposedOrNot, ipinfo.io).
- DNS (Domain Name System): The system that translates human-readable domain names (like `google.com`) into machine-readable IP addresses.
- GUI (Graphical User Interface): The interactive visual dashboard of the application, built with Streamlit.
- IP (Internet Protocol): A numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication.
- Nmap (Network Mapper): A powerful, open-source tool for network discovery and security auditing, used in this project for port scanning.
- OS (Operating System): The software that manages computer hardware and software resources (e.g., Windows, Linux). The agent can attempt OS detection.
- OSINT (Open Source Intelligence): The practice of collecting and analyzing information from publicly available sources to generate actionable intelligence.
- PDF (Portable Document Format): A file format used to present and exchange documents reliably, used by the project for its downloadable reports.
- Streamlit: The Python library used to build and serve the web application's GUI.
- TLD (Top-Level Domain): The last segment of a domain name, following the final dot (e.g., `.com`, `.org`, `.tk`).
- WHOIS: A query and response protocol widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name.

Chapter 1

Introduction

This chapter presents a brief idea of the project and motivation behind this project and also presents scope of the project.

1.1) Background

Manual Open Source Intelligence (OSINT) investigations are often inefficient and labor-intensive. The process requires investigators to use multiple, disparate tools and conduct repetitive searches across scattered data sources. This fragmentation makes gathering and consolidating intelligence a time-consuming and cumbersome task, highlighting the need for a more streamlined solution.

1.2) Motivation

The primary motivation for this project is to address the inefficiencies of manual OSINT. There is a clear need for a unified, automated platform that centralizes essential intelligence-gathering functions. By automating tasks and consolidating results, the project aims to save investigators valuable time, reduce manual effort, and improve the overall efficiency and effectiveness of the OSINT process.

1.3) Objectives

1. To develop an OSINT Automated platform capable of collecting and analyzing publicly available data from diverse online sources.
2. To implement automated reconnaissance tools for tasks such as WHOIS lookups, DNS queries, IP geolocation, subdomain enumeration, and data breach detection.
3. To integrate a web scraping module for extracting structured data from target websites in real time.
4. To design a dynamic reporting system that generates comprehensive PDF reports summarizing all gathered intelligence.
5. To ensure the platform's accuracy, scalability, and usability through extensive testing on multiple domains and datasets.

1.4) Scope of Project

The scope of this project is to design, develop, and implement an integrated "OSINT Automation Tool." The platform will provide a single interface to perform a range of essential intelligence-gathering tasks.

The core functionalities within the project's scope include:

- Domain & IP Analysis:
    - WHOIS Lookup: To retrieve domain registration and owner details.
    - DNS Query: To find DNS records associated with a domain.
    - Subdomain Enumeration: To discover and list subdomains of a target.
    - IP Geolocation: To identify the physical location of a server and display it on a map.

- Security & Vulnerability Assessment:
    - Nmap Port Scanning: To identify open ports and running services on a target.
    - Email Breach Checking: To check if an email address has been compromised in known data breaches (using APIs like XposedOrNot).
    - Risk Scoring Engine: To calculate and display a risk score for a target based on factors like domain age, suspicious subdomains, and WHOIS data.

- Profile & Data Collection:
    - Web Scraping: To extract specific data from target websites.

- Reporting:
    - Consolidated Reporting: To automatically generate a single, comprehensive PDF report containing all the findings from the different modules.

Chapter 2

Literature Survey

1. Redefining OSINT Software Architecture With System-Centric Architecture Design

By Gokhan Yurtalan, Serdar Arslan (IEEE Access, 2025)

- Proposes a new OSINT software architecture using QAW, ADD, and ATAM to balance performance, scalability, and security.
- Employs distributed processing and on-premise data storage to improve scalability and reduce digital footprint risks.
- Details design through module, component-connector, and allocation views.
- Targets automation in the OSINT intelligence cycle while retaining human oversight for decision-making.

2. The Not Yet Exploited Goldmine of OSINT : Opportunities, Open Challenges and Future Trends

By Javier Pastor-Galindo, Pantaleone Nespoli, Felix Gomez Marmol, Gregorio Martinez Perez (IEEE Access, 2020)

- ➢ Comprehensive review of OSINT's strengths, limitations, and future trends in cybersecurity, cybercrime, and social analysis.
- ➢ Key applications: social sentiment analysis, crime detection, cyberdefense.
- ➢ Identifies challenges: privacy concerns, GDPR restrictions, risk of misuse (e.g., fake news, profiling abuse).
- ➢ Suggests integrating OSINT into Detection Maturity Level (DML) for better cyberattack attribution.
- ➢ Highlights unexploited potential across government, marketing, and counterintelligence.

3. Open Source Intelligence and its Applications in Next Generation Cyber Security

By Krishna Prasad Karani from Srinivas University (August 2021)

- The review concludes that current Open Source Intelligence (OSINT) practices require more robust and intelligent solutions from AI, machine learning, and automated reasoning to strengthen analysis.

- The research identifies a need to reduce dependency on human decision-making to avoid errors and the necessity of a "truth discovery process" to eradicate incorrect information.

- While AI is entering the field and OSINT can be improved by correlating data from multiple sources, the review finds there is still significant progress needed to prepare OSINT for the challenges of Web 3.0.

Chapter 3

Fundamental Concepts / Overview

In this chapter a brief summary of the problem statement, project's outline and elements, and various technologies used are explained. The technologies mentioned in this chapter will be further used in the next chapter.

3.1) Introduction

In the field of cybersecurity, Open Source Intelligence (OSINT) is a foundational component of any security assessment, ethical hacking operation, or threat intelligence engagement. It is the practice of collecting and analyzing data from publicly available sources to build a comprehensive profile of a target. Traditionally, this reconnaissance phase is a manual, fragmented, and labor-intensive process. Cybersecurity specialists must juggle multiple disparate tools, conduct repetitive searches across scattered websites, and manually consolidate data, a method that is not only time-consuming but also inefficien

3.2) Problem Statement

Manual OSINT investigations require multiple tools, repetitive searches, and scattered data sources, making the process time-consuming and inefficient. There is a need for a unified, automated platform that can perform essential OSINT tasks — including WHOIS lookups, DNS queries, IP geolocation, subdomain enumeration, risk scoring engine, Nmap, email breach checks, and web scraping — in one place, with the ability to generate a consolidated PDF report.

3.3) Technologies Used

1. Core Framework & User Interface (UI)

- Streamlit: The primary technology for the entire application. It's a Python framework that turns data scripts into interactive, shareable web applications. The entire GUI, including the sidebar, buttons, charts, and tabs, is built with Streamlit.

2. Cybersecurity & Reconnaissance

- python-nmap: A Python wrapper for the Nmap (Network Mapper) utility. This is the core technology used for all advanced port scanning, service detection, and OS fingerprinting.

- python-whois: The library used to perform WHOIS queries. It retrieves domain registration data like the registrar, creation date, and expiration date.
- dnspython: A library for DNS (Domain Name System) queries. It is used to fetch various DNS records (A, MX, TXT, etc.) for the target domain.
- socket: A built-in Python library used for low-level networking. In this code, it's used for:
    - Resolving domain names to IP addresses (e.g., socket.gethostbyname).
    - The "Basic Socket Scan," which attempts to open a TCP connection to a port.

3. Data Visualization & Mapping

- Plotly: A powerful, interactive charting library. It is used to create the gauge charts (for risk scores) and bar charts (for service distribution, risk factors, etc.).
- Folium: A library used to visualize data on an interactive Leaflet map. It is the technology used to plot the IP address locations on the world map.
- streamlit_folium: A component that acts as a bridge to embed Folium maps directly within the Streamlit application.
- Pandas: A crucial data analysis library. While not a *visual* technology itself, it is used to structure all the collected data (port lists, subdomains, WHOIS info) into tables (st.dataframe) and to prepare data for the Plotly charts.

4. External APIs & Services

- ipinfo.io: An external IP geolocation API. The requests library sends a query to this service to get the physical location, city, and organization for a given IP.
- XposedOrNot: An external data breach API. This service is queried to check if an email address has appeared in known data breaches.
- requests: The Python library used to make HTTP requests to the external APIs listed above.

5. Reporting & Data Handling

- ReportLab: The library used for PDF generation. It takes the analysis results and programmatically creates a formatted, downloadable PDF report.
- JSON: A built-in library for handling the JavaScript Object Notation data format, which is the standard format for responses from most web APIs.

6. Performance & Concurrency

- concurrent.futures.ThreadPoolExecutor: A built-in Python library used to run tasks concurrently. In this code, it is used to speed up the subdomain enumeration and basic port scanning by checking many subdomains/ports at the same time, rather than one by one.

3.4) Core OSINT Libraries & Tools:

- Nmap: The external tool used for port scanning and service discovery.
- python-whois (or similar): A Python library used to perform WHOIS lookups.
- socket: A built-in Python library used for basic DNS queries.
- streamlit
- requests
- dns.resolver
- json
- pandas
- plotly.express
- plotly.graph_objects
- datetime
- time
- re
- subprocess
- threading
- hashlib
- base64
- BytesIO
- folium
- letter
- SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
- getSampleStyleSheet, ParagraphStyle
- inch
- colors
- ssl
- concurrent.futures
- urlparse
- warnings
- ipaddress

3.5) Code Explanation

This script is a comprehensive Open Source Intelligence (OSINT) web application built using Streamlit. It provides a web dashboard where a user can enter a target (a domain or an IP address) and perform a wide range of reconnaissance tasks.

The application gathers data, analyzes it for security risks, visualizes the findings in graphs and maps, and finally, allows the user to export all the information into a single PDF report. It also includes a separate utility for checking if an email address has been exposed in a data breach.

Core Components :

1. OSINTAgent (Class): This is the "engine" of your application. It's a class that contains all the individual functions (methods) for performing specific OSINT tasks, such as whois_lookup, nmap_port_scan, calculate_risk_score, etc. This is good design as it groups all the logic neatly in one place.
2. main() (Function): This is the "frontend" and "orchestrator" of your application. It uses Streamlit commands (like st.sidebar, st.button, st.metric) to:
    - Build the user interface (the sidebar controls and the main dashboard).
    - Take the user's input.
    - Call the OSINTAgent methods in the correct order when the "Start Analysis" button is clicked.
    - Display all the results using tables, charts, and maps.

OSINTAgent Class: Method-by-Method Breakdown

- __init__(self):
    - This is the *constructor*. It runs when you create a new OSINTAgent.
    - It initializes a dictionary called self.risk_factors which acts as a settings file, storing lists of suspicious TLDs (like .tk), suspicious keywords (like admin), high-risk ports (like 22 for SSH, 3389 for RDP), and common port numbers.
    - Crucially, it tries to initialize the Nmap scanner (nmap.PortScanner()). If it fails (because Nmap isn't installed), it sets self.nmap_available to False and stores an error message.
- whois_lookup(self, domain):
    - Uses the whois library to query for domain registration data.
    - It includes logic to handle cases where the returned dates are lists (taking the first element) and provides 'Unknown' as a default for missing data.
- dns_lookup(self, domain):
    - Uses the dns.resolver library to find various DNS records (A, AAAA, MX, TXT, etc.).

    - It loops through each record type and stores the results, or an empty list if no record is found.
- subdomain_finder(self, domain, wordlist=None):
    - Performs subdomain enumeration. It uses a default wordlist (like 'www', 'mail', 'admin').
    - It uses ThreadPoolExecutor to perform multithreading. This means it checks many subdomains (e.g., admin.example.com, mail.example.com) simultaneously, making the scan much faster than checking them one by one.
- basic_port_scan(self, target, ports, timeout=1):
    - This is a fallback port scanner that uses Python's built-in socket library. It's not as powerful as Nmap but works if Nmap is unavailable.
    - It's also multithreaded using ThreadPoolExecutor to check multiple ports at once.
- nmap_port_scan(self, target, scan_type='basic', port_range='1-1000'):
    - This is the primary, advanced port scanning function.
    - It first checks if Nmap is available.
    - It constructs and runs an nmap command based on the scan_type selected by the user (e.g., -sS for stealth, -sV for service detection).
    - After the scan, it parses the detailed results from the self.nm object, extracting open ports, services, versions, and OS information.
- analyze_port_security(self, port_scan_results):
    - This is a custom risk analysis function. It takes the list of open ports and:
    - Calculates a security_score (starting at 100).
    - Subtracts points for high-risk ports (-20), dangerous services like Telnet (-15), or exposed databases (-10).
    - Generates a list of security recommendations based on its findings.
- ip_geolocation(self, ip):
    - Uses the requests library to make an API call to ipinfo.io.
    - It fetches and returns a JSON object containing the IP's city, country, geographic coordinates (loc), and ISP (organization).
- calculate_risk_score(self, domain_data, port_scan_results=None):
    - This calculates the Overall Risk Score.
    - It adds points based on various factors:
        - Domain Age: A very new domain (e.g., < 30 days) gets a high-risk score.
        - TLD: A suspicious TLD (like .tk) adds 30 points.
        - Privacy: If WHOIS privacy is enabled, it adds 10 points (as it can be used to hide identity).
        - Ports: If high-risk ports are open, it adds 25 points.
- flag_suspicious_subdomains(self, subdomains):
    - This function iterates through the list of found subdomains.
    - It compares them against the suspicious_keywords list (from __init__) and flags any matches (e.g., admin.example.com).
- generate_pdf_report(self, analysis_results):

    - This uses the reportlab library to programmatically build a PDF document.
    - It adds a title, executive summary, and tables for WHOIS data and port scan results, styling them with colors and fonts.
- create_geolocation_map(self, ip_locations):
    - This uses the folium library to create an interactive map.
    - It calculates the center point of all found IPs and places a marker for each one with a popup showing its details.
- check_email_breaches_xposed(self, email):
    - Queries the api.xposedornot.com API to check for email breaches.
    - It correctly handles the different types of JSON responses: a 'pwned' status, a 'safe' status (which can come in two different formats), or an error.

main() Function: Streamlit UI & Application Flow

This function controls everything the user sees and interacts with.

1. Page Setup:
    - st.set_page_config(...): Sets the browser tab title, icon, and wide layout.
2. Session State:
    - st.session_state: This is a critical Streamlit feature. It's a dictionary that persists data even when the user clicks a button (which causes the script to re-run). You use it to store analysis_results so the dashboard stays populated after the scan is finished.
3. Sidebar (Control Panel):
    - All st.sidebar... commands place UI elements in the left-hand sidebar.
    - You create the text input for the target, checkboxes for scan options (Enable Subdomain Enumeration), and select boxes for Nmap scan types.
    - The Email Breach Check is a separate, self-contained form in the sidebar.
4. Main Analysis Logic (if st.sidebar.button("Start Analysis"...)):
    - This block of code runs *only* when the "Start Analysis" button is pressed.
    - It creates an OSINTAgent instance.
    - It shows a progress bar (st.progress) and status text (st.empty).
    - It validates if the input is an IP or a domain.
    - Orchestration: It then calls the agent's methods in logical order, updating the progress bar at each step:
        1. whois_lookup & dns_lookup (if domain)
        2. subdomain_finder (if enabled)
        3. nmap_port_scan (if enabled)
        4. analyze_port_security
        5. calculate_risk_score
        6. flag_suspicious_subdomains
        7. ip_geolocation

- Finally, it stores all the collected data in st.session_state.analysis_results and sets st.session_state.analysis_complete = True.

5. Results Display (if st.session_state.analysis_complete...):
    - This block of code runs *only* after a scan is complete.
    - It creates the main dashboard layout using st.columns for the top metrics.
    - Visualizations: It uses st.metric for key numbers, plotly.graph_objects to create the "gauge" charts for risk scores, plotly.express for bar charts (like "Top Services Found"), and st_folium to display the interactive map.
    - Data Tabs: st.tabs is used to neatly organize the large amounts of detailed data, so the user can click between "Port Scan," "WHOIS Data," "DNS Records," etc.
    - PDF Download: st.download_button is configured to call agent.generate_pdf_report() when clicked, providing the generated PDF to the user.

3.6) Usage :

1. Phase 1 Reconnaissance (Footprinting): This is the first step of any ethical hack or security assessment. Your tool automates this entire phase. A specialist can enter a company's domain and instantly get:
    - Domain & Infrastructure: WHOIS data (admin contacts, creation date).
    - Network Map: DNS records and all discovered Subdomains (e.g., dev.company.com, admin.company.com).
    - Physical Location: IP Geolocation to see where the servers are.

2. Attack Surface Mapping: The specialist's primary goal is to find all possible entry points. Your tool directly maps this attack surface by:
    - Enumerating all subdomains.
    - Running Nmap on those targets to find open ports and running services (e.g., "Port 22 SSH is open on dev.company.com").
    - Identifying potential vulnerabilities on those services.

3. Rapid Vulnerability Triage: In a real engagement, an analyst might have hundreds of targets. Your Risk Scoring Engine is the most valuable feature for this. It allows the specialist to instantly triage their targets:
    - A target with a "High" score (e.g., newly registered domain, suspicious subdomains like login-portal.company.com, missing WHOIS data) will be investigated *immediately*.
    - This saves them from wasting time on established, low-risk targets.

4. Identifying Human Attack Vectors: The Email Breach Checking feature is a direct tool for social engineering and credential stuffing attacks. A specialist will:
    - Check a target company's email format (e.g., firstname.lastname@company.com).
    - Use your tool to see which of those emails are in known breaches.
    - This provides a list of high-value targets for spear-phishing campaigns or to try breached passwords against (known as password spraying or credential stuffing).

5. Threat Intelligence Profiling: For a threat analyst, your tool can be used to profile a malicious actor. If they find a malicious domain in a phishing email, they can put it into your tool to:
    - Discover the actor's entire infrastructure (other domains, IPs, hosting provider).
    - Use the Shodan and Nmap results to see what kind of C2 (Command and Control) server they are running.

6. Automated Reporting (Major Time-Saver): This is a critical, practical benefit. Specialists spend *hours* writing reports for clients. Your tool's ability to generate a consolidated PDF report automates a huge part of this documentation. They can run a scan, export the PDF, and attach it directly to their final report.

3.7) Project Demonstration

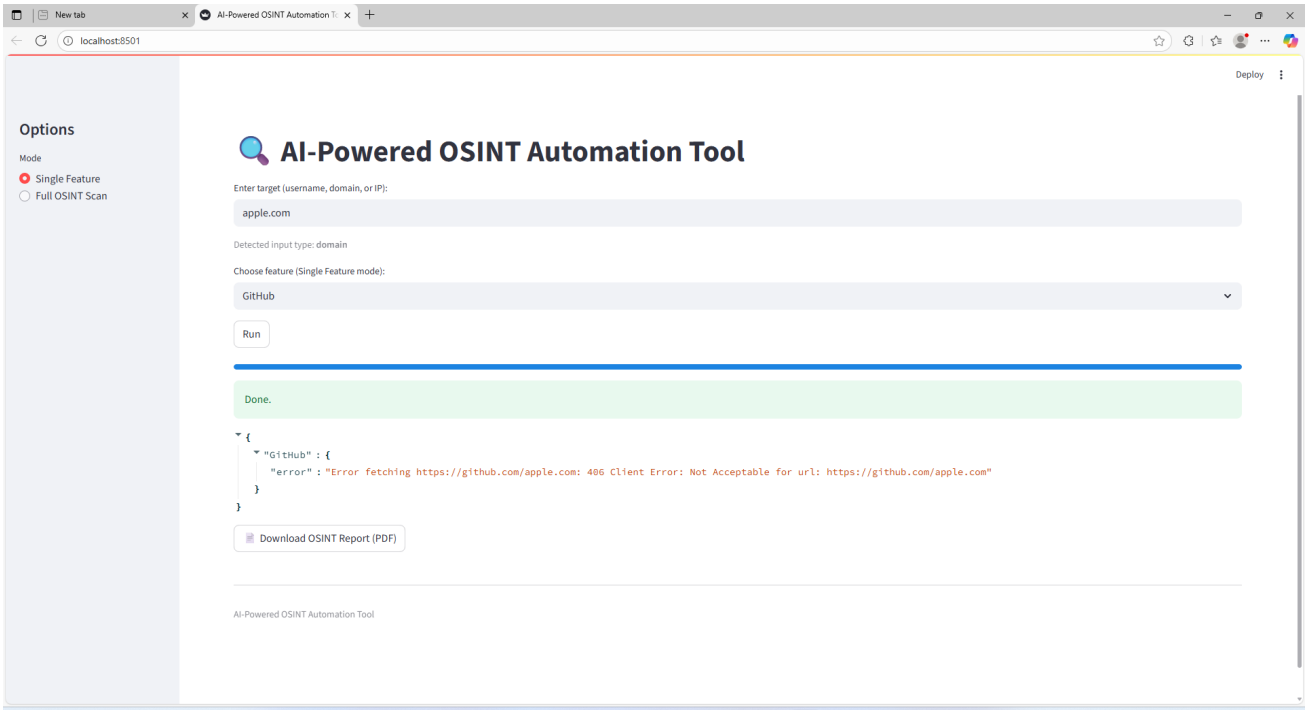
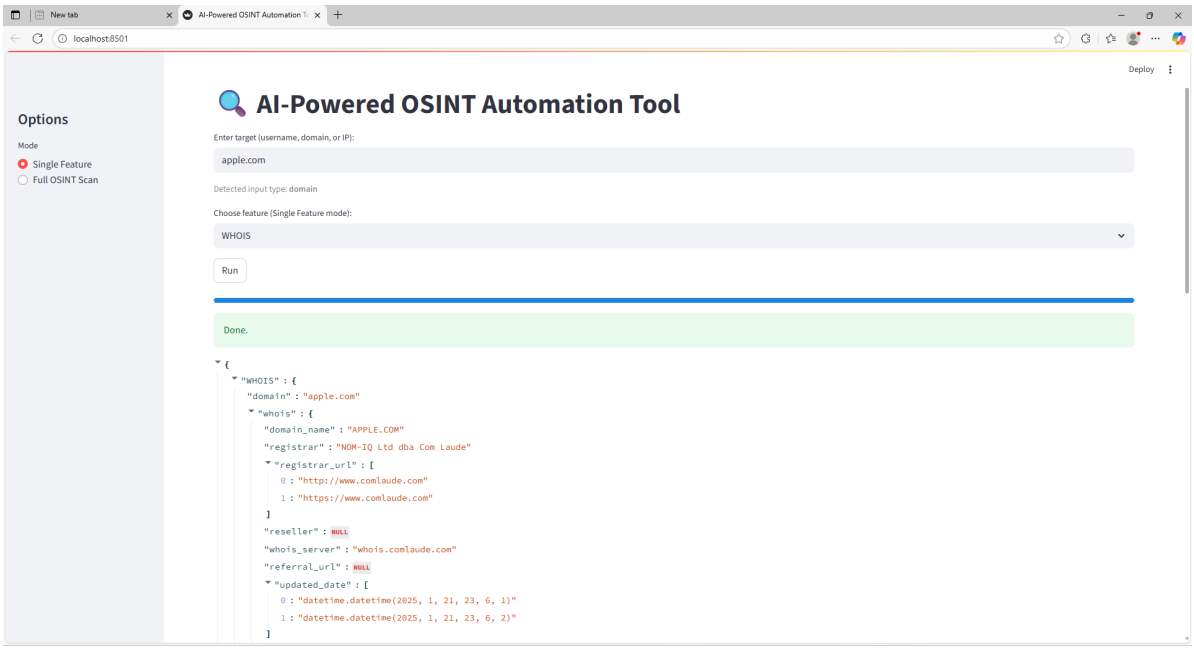


Figure 1: Initial Web Scraping



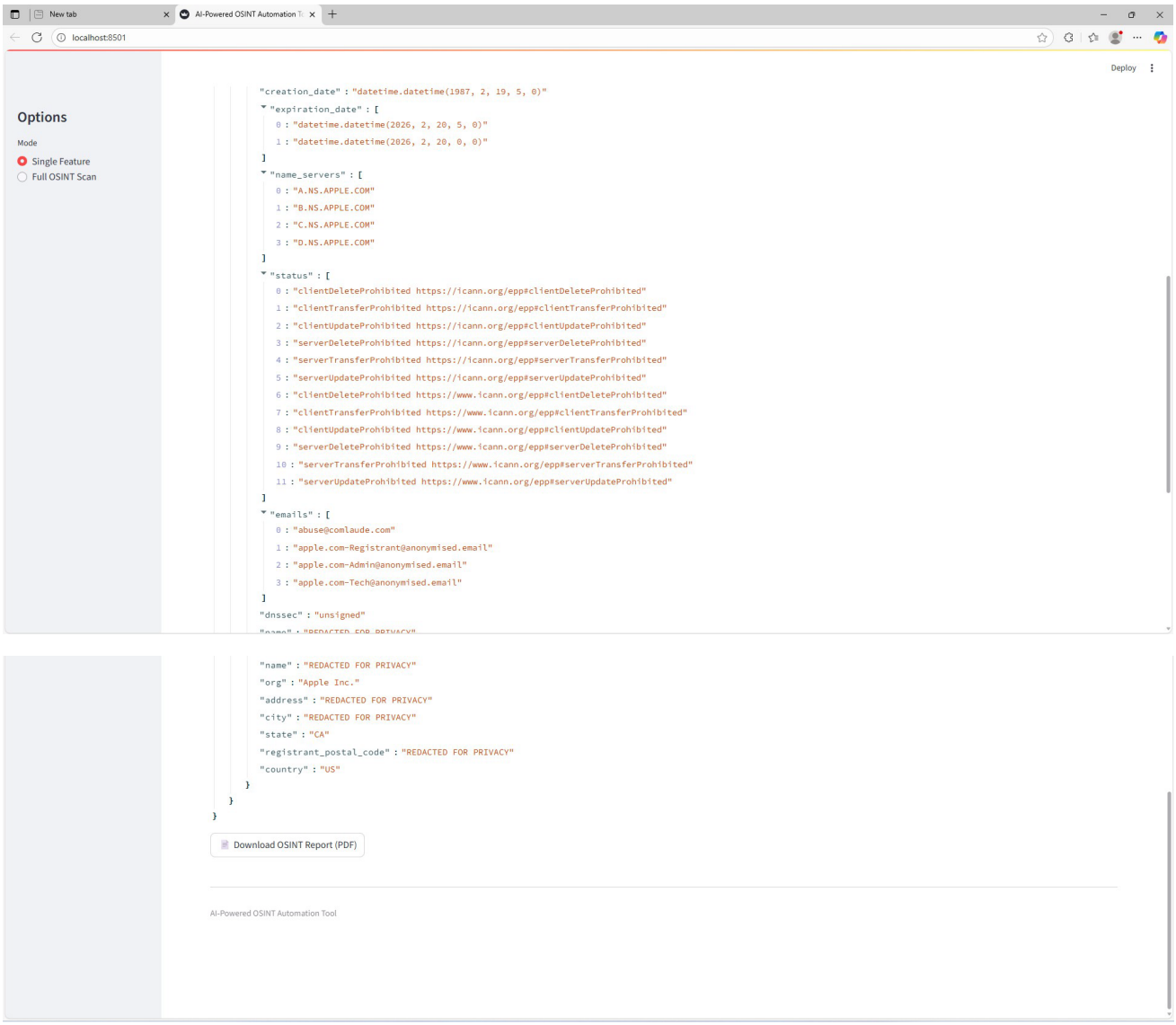


Figure 2: WHOIS lookup

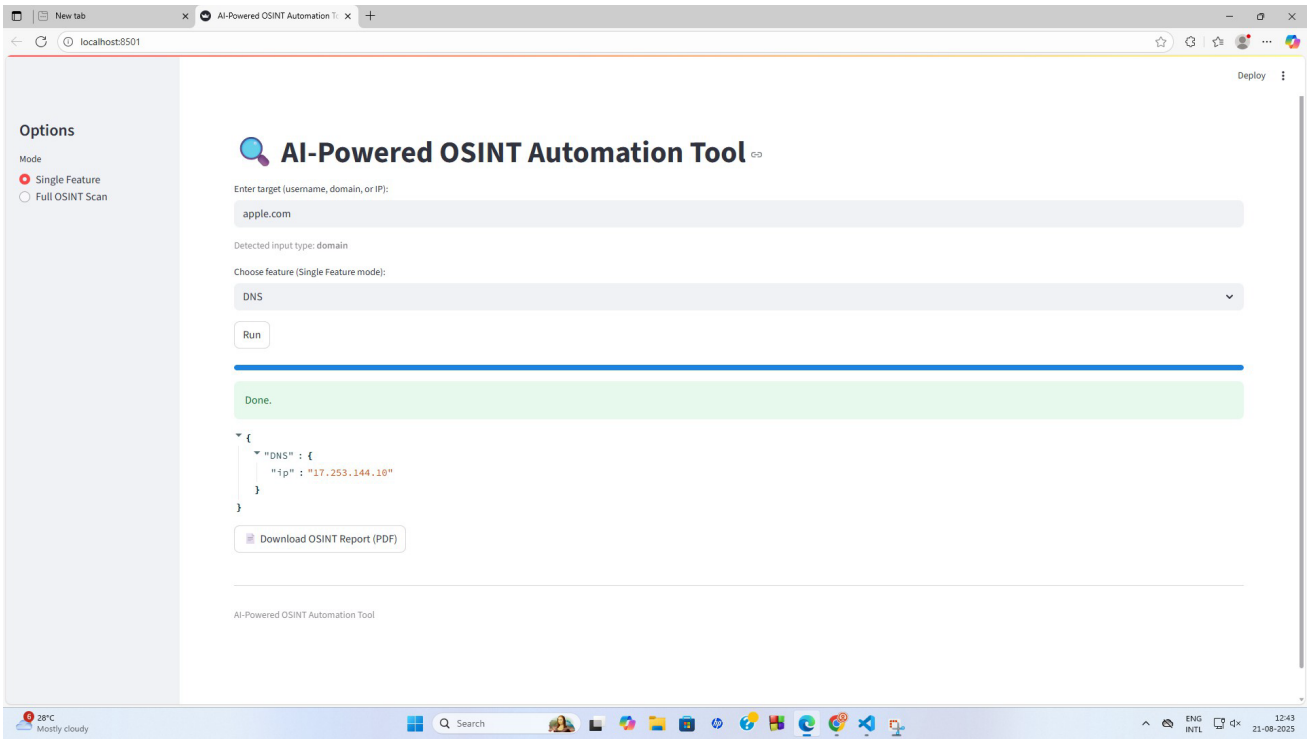
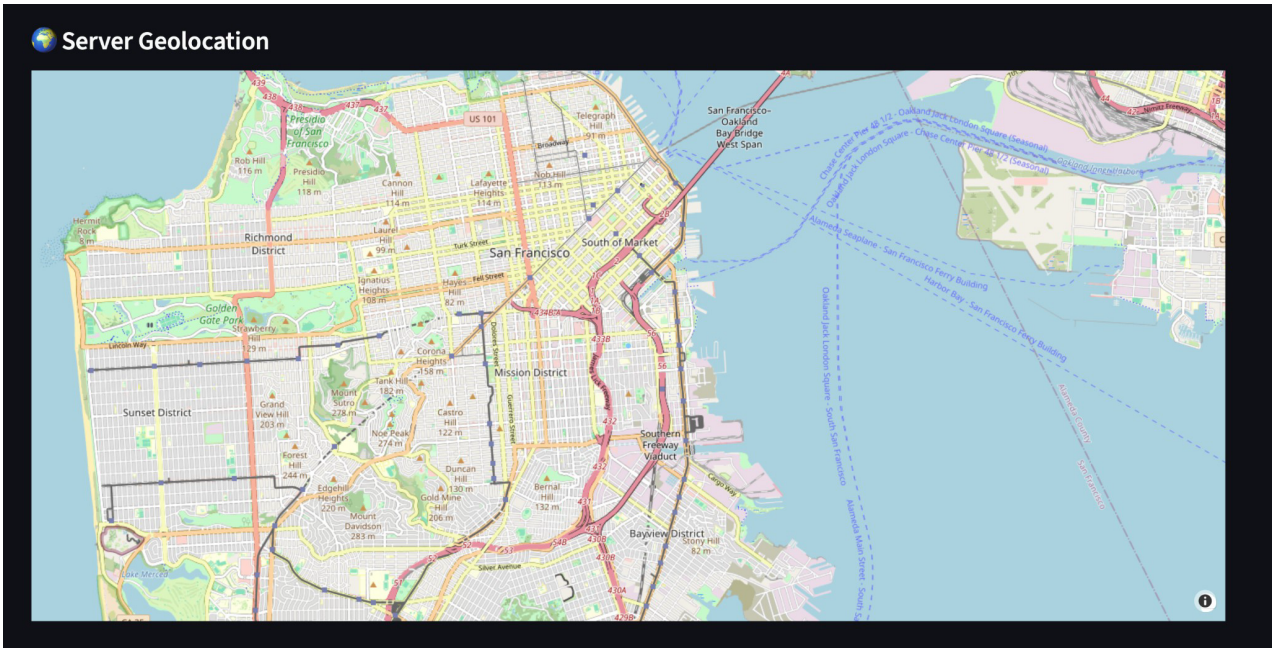

New tab
AI-Powered OSINT Automation T
localhost:8501
Deploy
Options
Mode
Single Feature
Full OSINT Scan
AI-Powered OSINT Automation Tool
Enter target (username, domain, or IP):
apple.com
Detected input type: domain
Choose feature (Single Feature mode):
DNS
Run
Done.
{
"DNS" : {
"ip" : "17.253.144.10"
}
}
Download OSINT Report (PDF)
AI-Powered OSINT Automation Tool
28°C
Mostly cloudy
Search
ENG
INTL
12:43
21-08-2025

Figure 3: DNS


Server Geolocation
San Francisco
South of Market
Mission District
Richmond District
Sunset District
Bayview District
San Francisco-Oakland Bay Bridge West Span
Presidio of San Francisco
Golden Gate Park
US 101

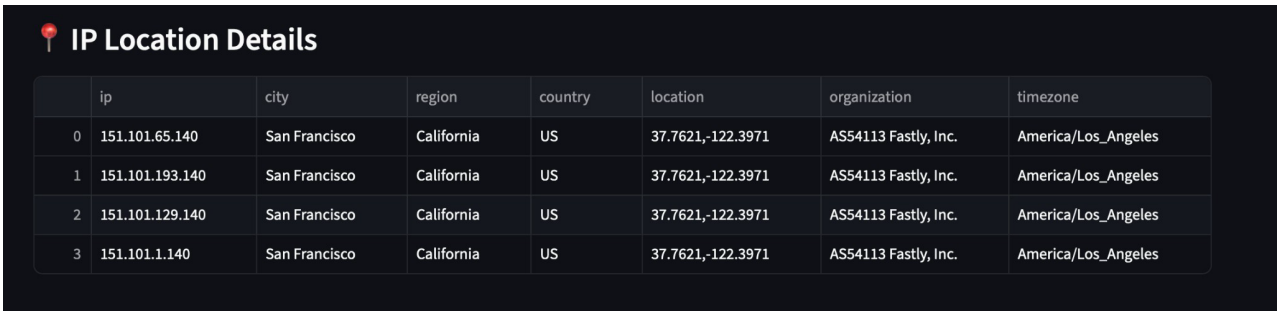


Figure 4: IP Geolocation

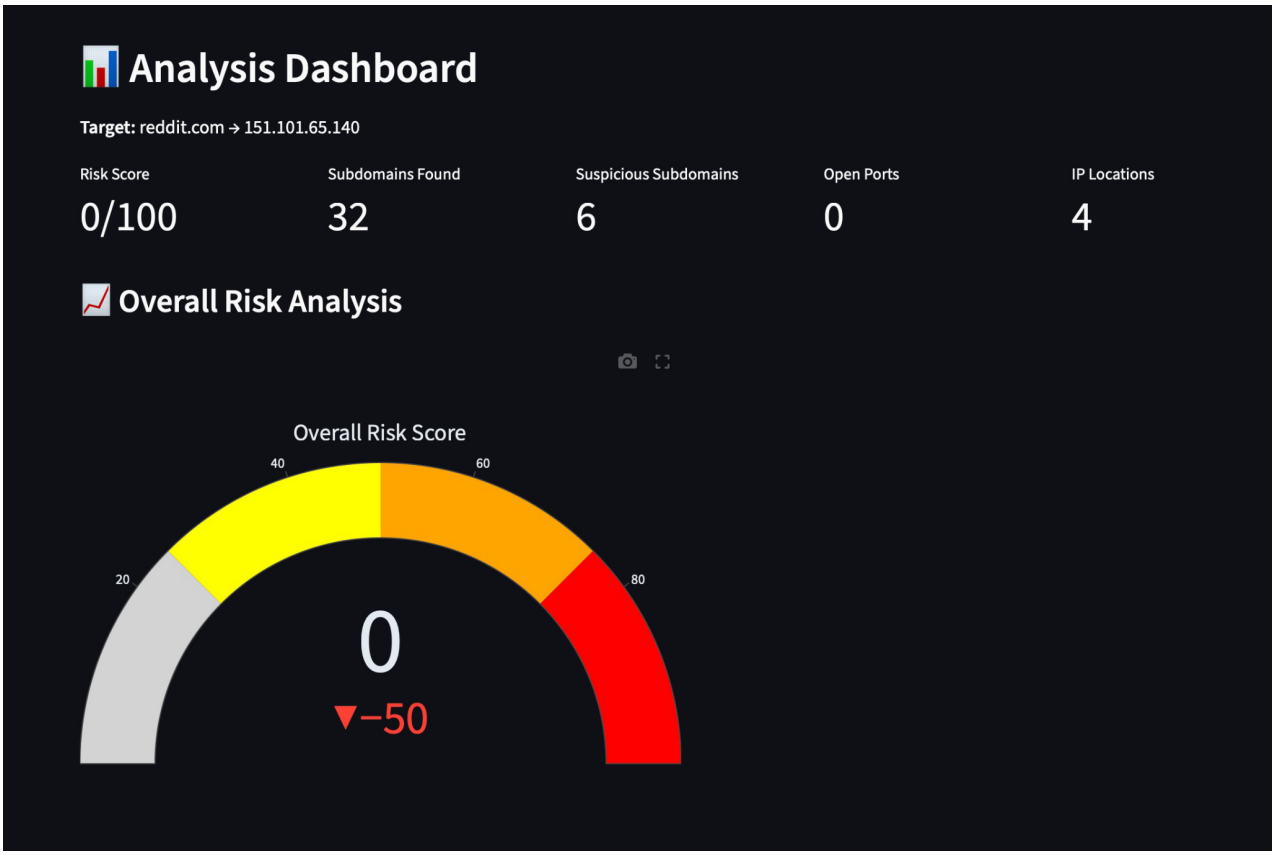


Figure 5: Risk Scoring

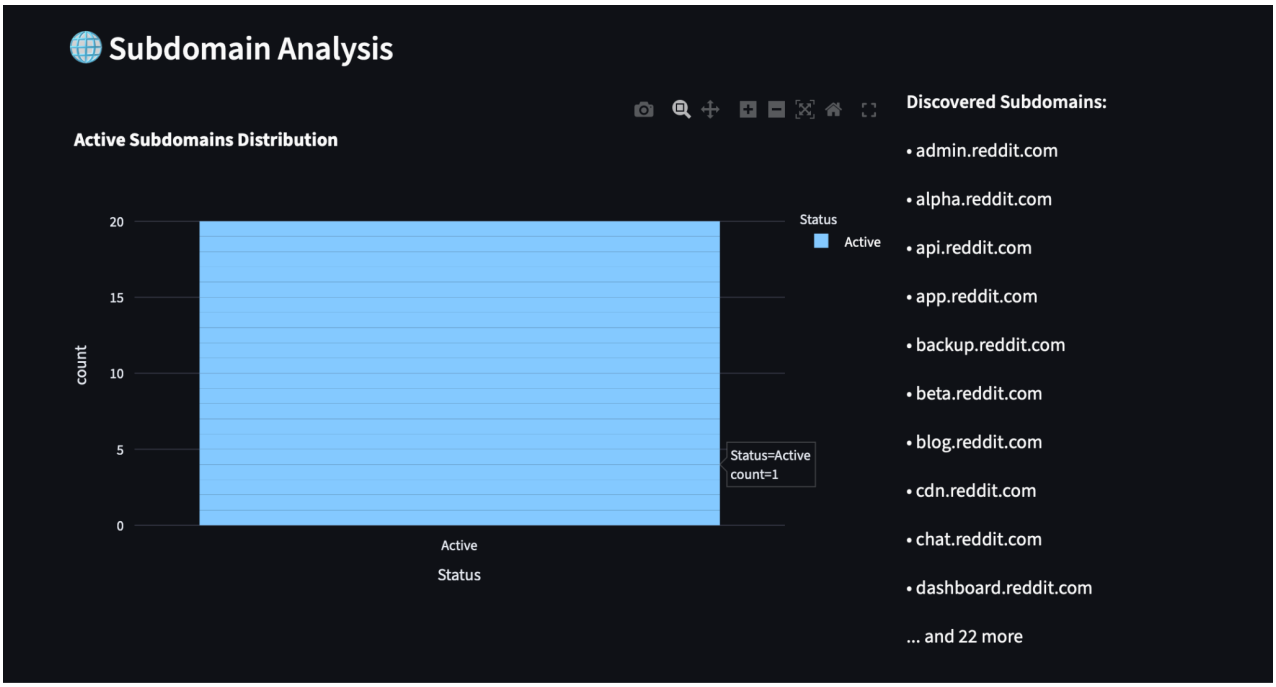
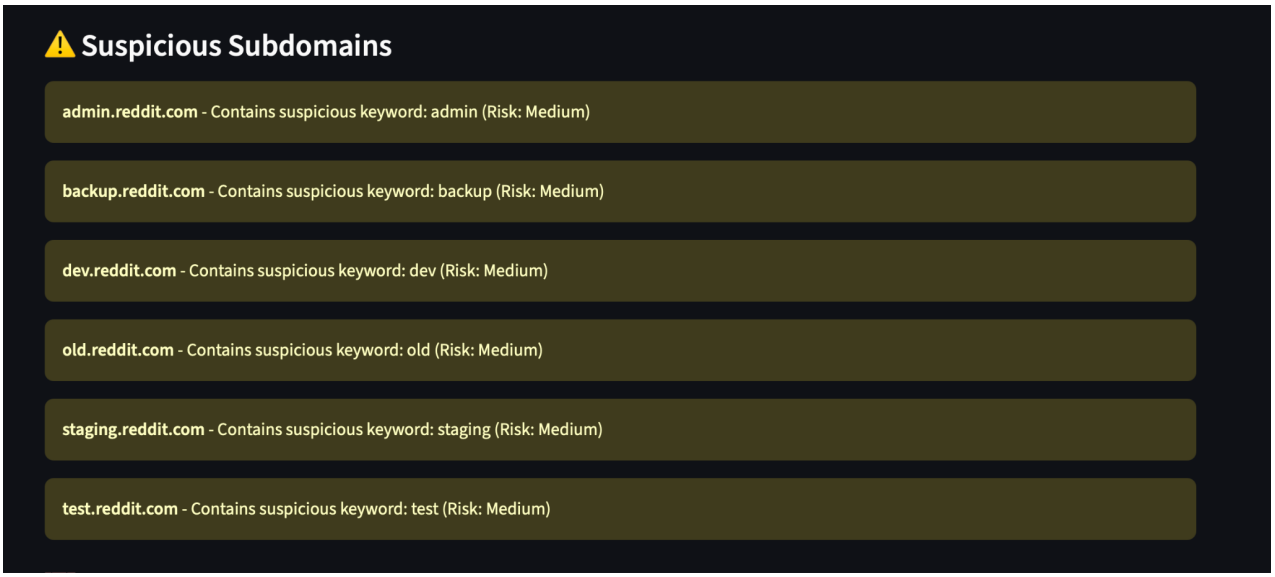


Figure 6: Subdomain Analysis



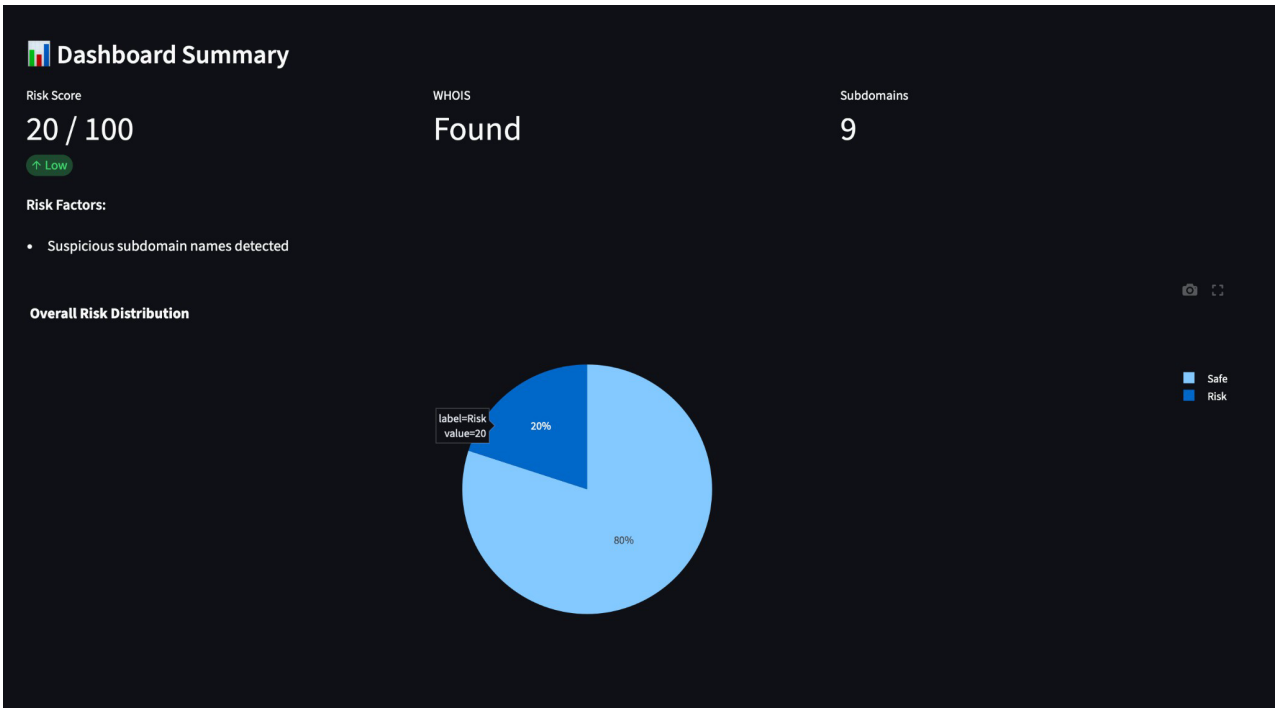
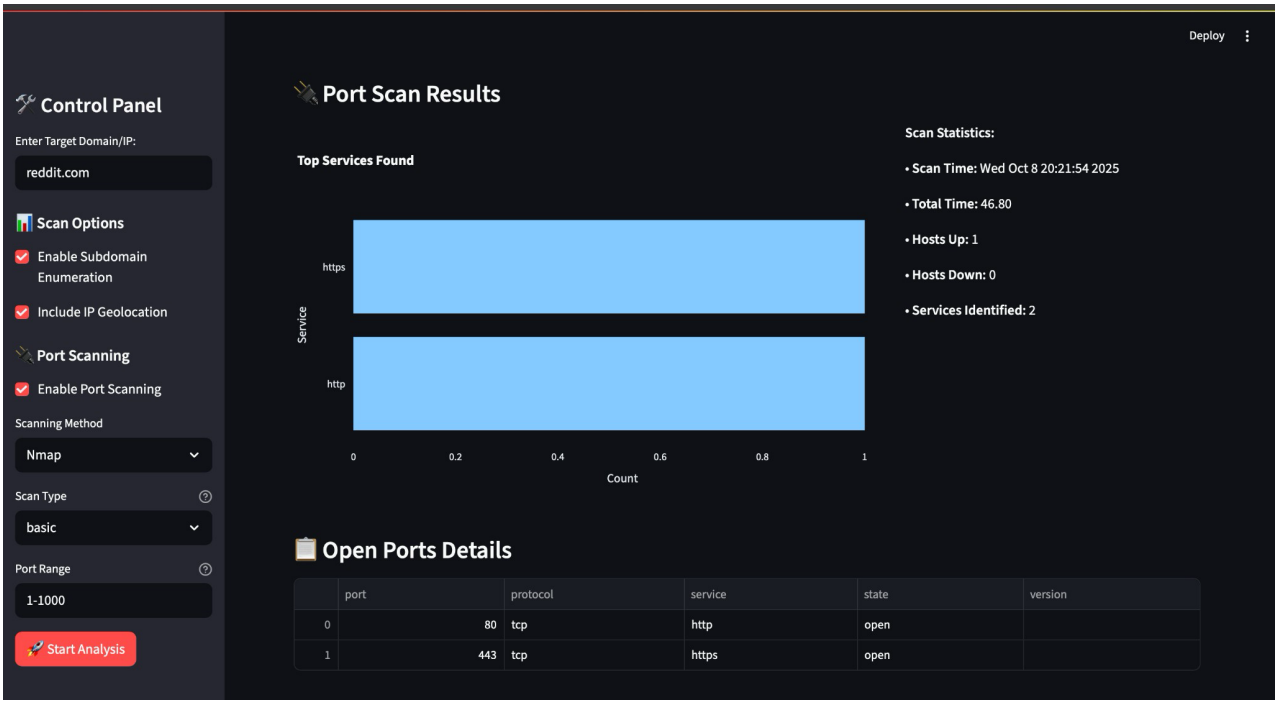


Figure 7: Dashboard Visualization



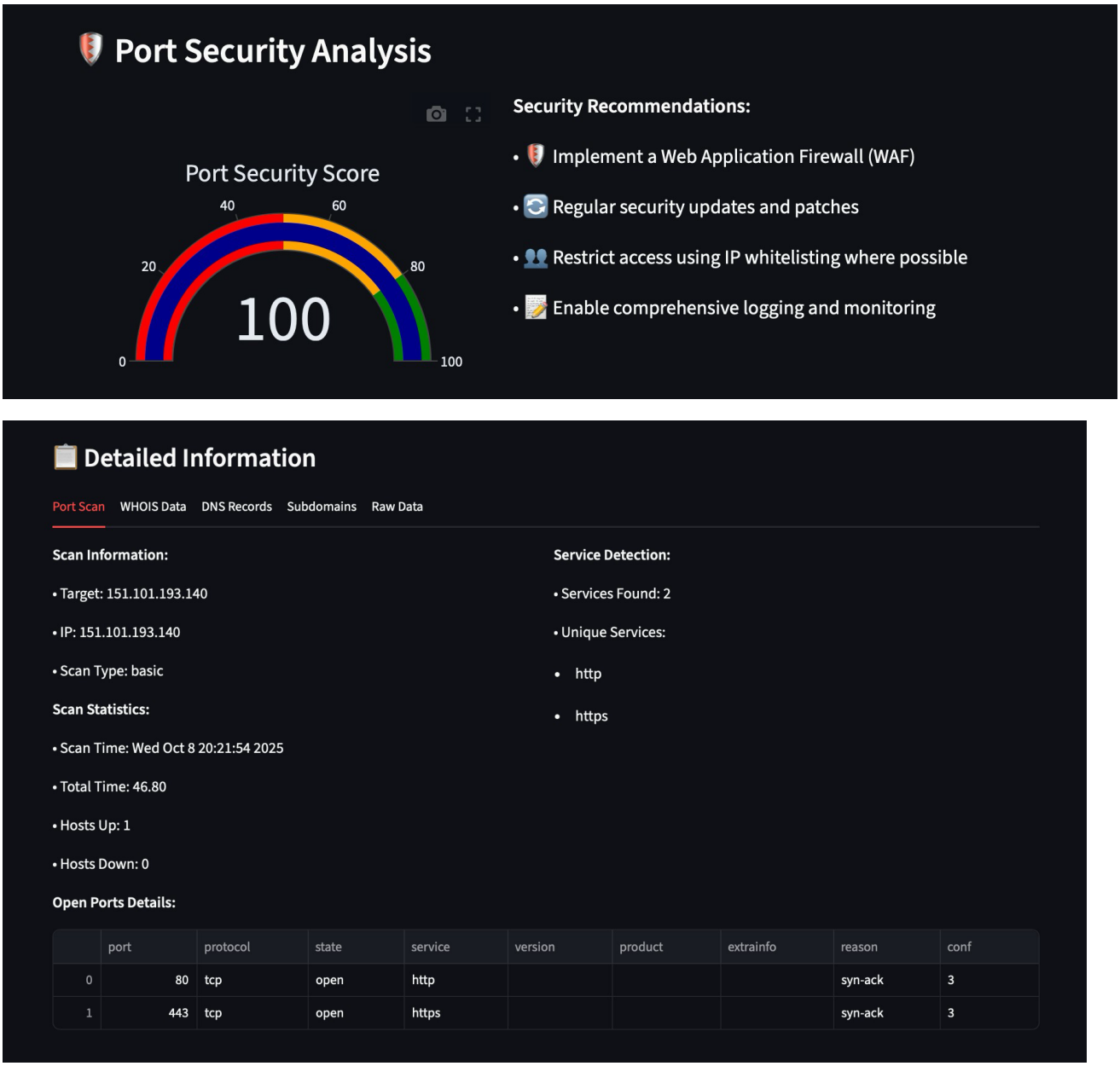


Figure 8: Nmap Scan

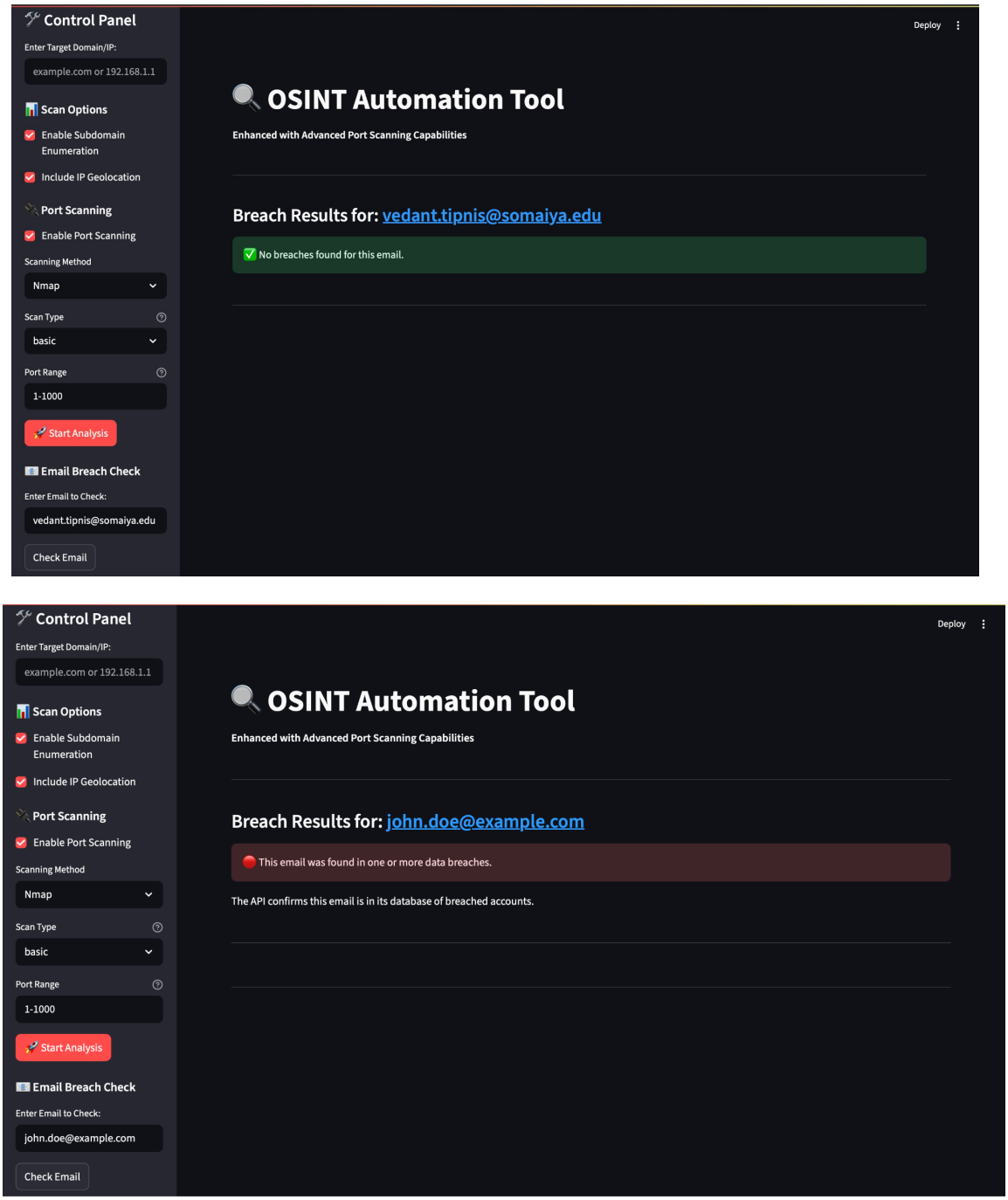
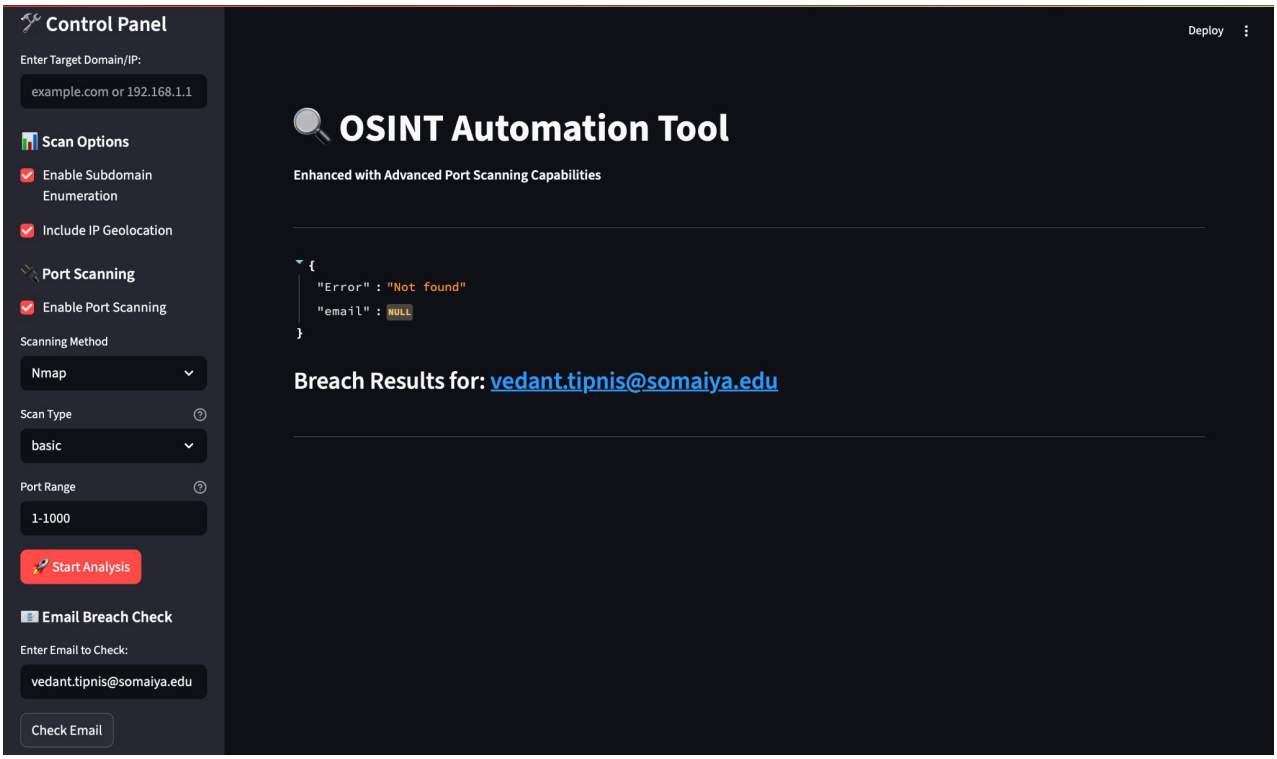
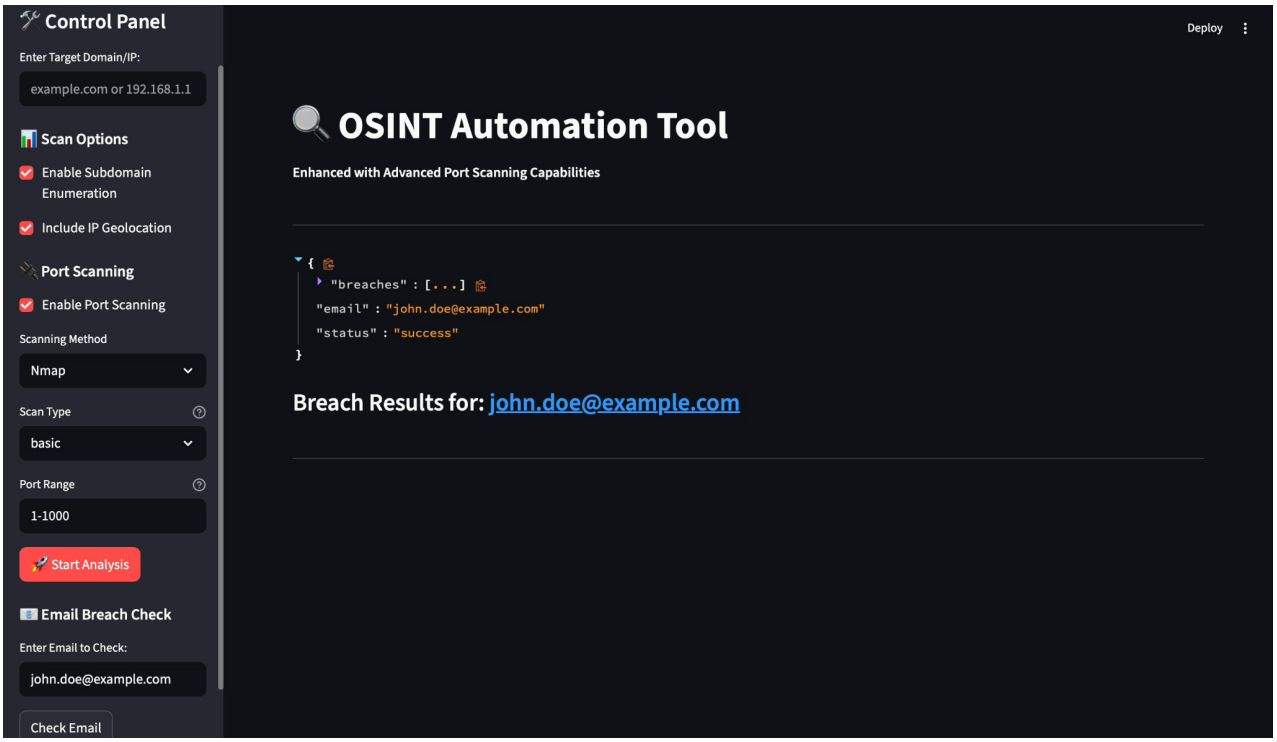


Figure 9: Email Breach (without JSON data)


Control Panel
Enter Target Domain/IP:
example.com or 192.168.1.1
Scan Options
Enable Subdomain Enumeration
Include IP Geolocation
Port Scanning
Enable Port Scanning
Scanning Method
Nmap
Scan Type
basic
Port Range
1-1000
Start Analysis
Email Breach Check
Enter Email to Check:
vedant.tipnis@somaiya.edu
Check Email
Deploy
OSINT Automation Tool
Enhanced with Advanced Port Scanning Capabilities
{
"Error" : "Not found"
"email" : NULL
}
Breach Results for: vedant.tipnis@somaiya.edu


Control Panel
Enter Target Domain/IP:
example.com or 192.168.1.1
Scan Options
Enable Subdomain Enumeration
Include IP Geolocation
Port Scanning
Enable Port Scanning
Scanning Method
Nmap
Scan Type
basic
Port Range
1-1000
Start Analysis
Email Breach Check
Enter Email to Check:
john.doe@example.com
Check Email
Deploy
OSINT Automation Tool
Enhanced with Advanced Port Scanning Capabilities
{
"breaches" : [...]
"email" : "john.doe@example.com"
"status" : "success"
}
Breach Results for: john.doe@example.com

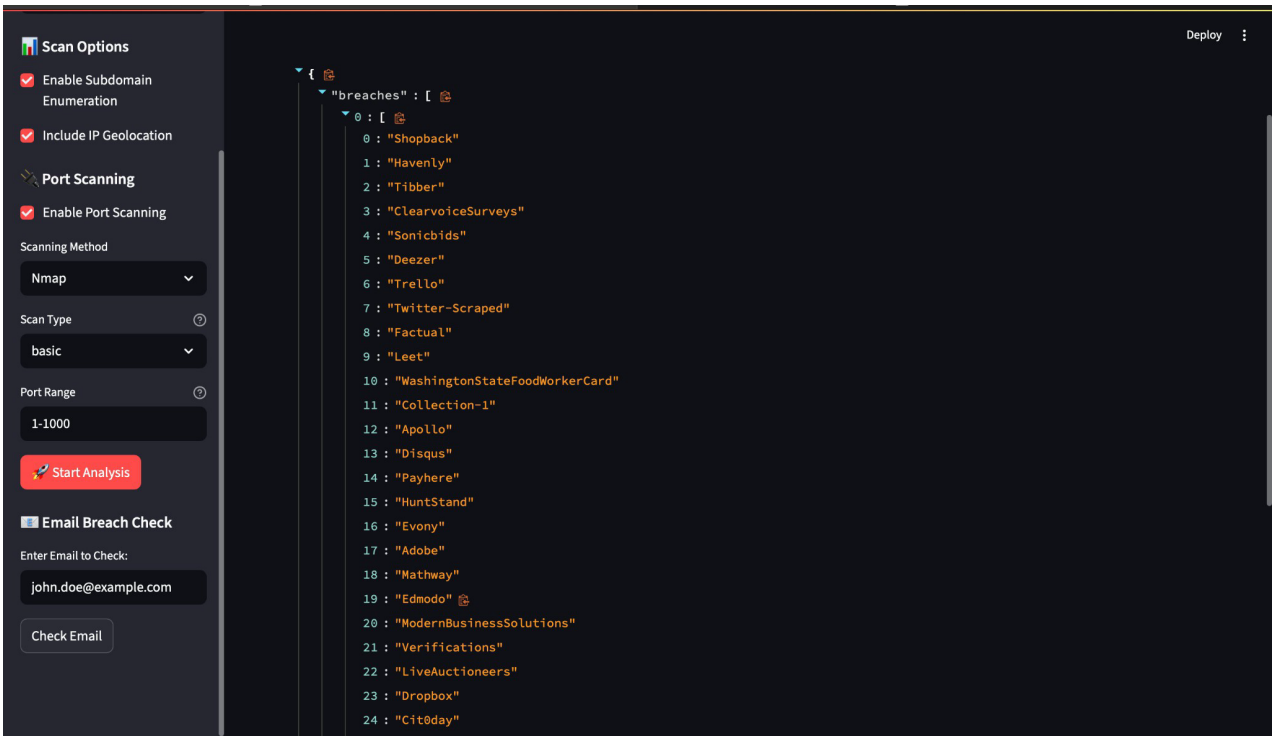


Figure 10: Email Breach (JSON data API sending)

Chapter 4

Conclusion & Future Scope

This chapter presents about the conclusion & future scope of the project that have been drawn after the completion of the project.

Conclusion

This project successfully addresses the core problem of inefficiency in manual OSINT investigations. By developing a unified "OSINT Automation Tool," we have transformed a fragmented and time-consuming process into a streamlined, automated workflow. The tool integrates essential reconnaissance modules—including WHOIS, DNS, Nmap, Subdomain Enumeration, and Email Breach Checking—into a single, user-friendly Streamlit dashboard.

The key achievement lies not only in the aggregation of these tools but in the value-added analysis provided by the Risk Scoring Engine and data visualizations. This allows a cybersecurity specialist to move beyond raw data collection and immediately perform risk-based triage. The automated PDF report generation further solves a critical pain point by eliminating the manual effort of consolidating findings. This platform serves as a powerful force multiplier, enabling analysts to conduct faster, more efficient, and more effective intelligence-gathering operations.

Future Scope

While the current tool provides a robust foundation, there are several avenues for future enhancement, particularly in line with the project's "AI-powered" objective.

1. AI & LLM Integration:
   - Natural Language Summaries: Integrate a Large Language Model (LLM) to automatically generate an executive summary of the findings (e.g., "This is a medium-risk target hosted in the US, with 3 suspicious subdomains and one open SSH port...").
   - Predictive Analysis: Use machine learning to analyze the collected data (open ports, WHOIS, etc.) to predict the target's likely purpose (e.g., "Phishing Site," "Command & Control Server," "E-commerce").

2. Expanded Data Sources (Social Media OSINT):
   - Integrate APIs for social media platforms like X (Twitter), Reddit, or Telegram to gather intelligence on public sentiment, user mentions, or discussions related to the target domain or organization.

3. Deeper Web Content Analysis:
    - Enhance the web scraper to not just fetch data but to analyze the content of target websites to find employee names, job titles, email addresses, and software technologies in use (e.g., "Powered by WordPress v5.8"), which all represent new intelligence points.

4. Historical Data & Trend Analysis:
    - Fully utilize the SQLite/MongoDB databases to store scan results over time. This would enable trend analysis, allowing a user to see what has changed (e.g., "When did this port open?", "When did the WHOIS information last change?").

5. Advanced Vulnerability Scanning:
    - Move beyond basic port scanning by integrating Nmap's vulnerability scanning scripts (NSE) or other modern tools like Nuclei to actively and safely check for known vulnerabilities on the discovered services.

6. Collaborative Platform:
    - Develop a multi-user "case file" system where analysts can share findings, add notes, and collaborate on an investigation within the tool.

References

1. Redefining OSINT Software Architecture With System-Centric Architecture Design — By Gokhan Yurtalan, Serdar Arslan (IEEE Access, 2025)

2. The Not Yet Exploited Goldmine of OSINT : Opportunities, Open Challenges and Future Trends —- By Javier Pastor-Galindo, Pantaleone Nespoli, Felix Gomez Marmol, Gregorio Martinez Perez (IEEE Access, 2020)

3. Open Source Intelligence and its Applications in Next Generation Cyber Security — By Krishna Prasad Karani from Srinivas University (August 2021)