Project Report: Python-Based Phishing Link Scanner

Introduction

Phishing is one of the most prevalent forms of cyberattacks that exploit human behavior to trick users into revealing sensitive information such as credentials, personal data, and financial details. As part of a cybersecurity internship project at **Future Intern**, I developed a command-line phishing link scanner using Python. The goal was to understand how phishing works, identify red flags in URLs, and enhance my technical and analytical cybersecurity skills.

This report comprehensively details the design, development, testing, and output of the Python-based phishing link scanner, including environment setup, libraries used, features implemented, and results obtained.

Objective

The main objective of this project was to develop a Python tool that detects potentially malicious or suspicious URLs by examining:

- Structural elements of the URL (e.g., presence of IP address, special characters, hyphens)
- Certificate and domain information (SSL and domain age)
- Reputation from a third-party service (VirusTotal API)
- Use of obfuscation techniques (e.g., URL shortening)

Tools and Technologies Used

- **Python 3.10+** – Core programming language
- **Libraries**:
  - `requests` – For sending HTTP requests
  - `beautifulsoup4` – HTML parsing (future extensibility)
  - `tldextract` – For extracting subdomain, domain, and suffix
- **VirusTotal API** – For checking URL/domain reputation
- **Windows 10 (Command Prompt)** – Primary development and execution environment

Setting up the Environment

Before starting development, I created a dedicated environment to isolate the project dependencies. The process included:

1. **Creating Project Directory**

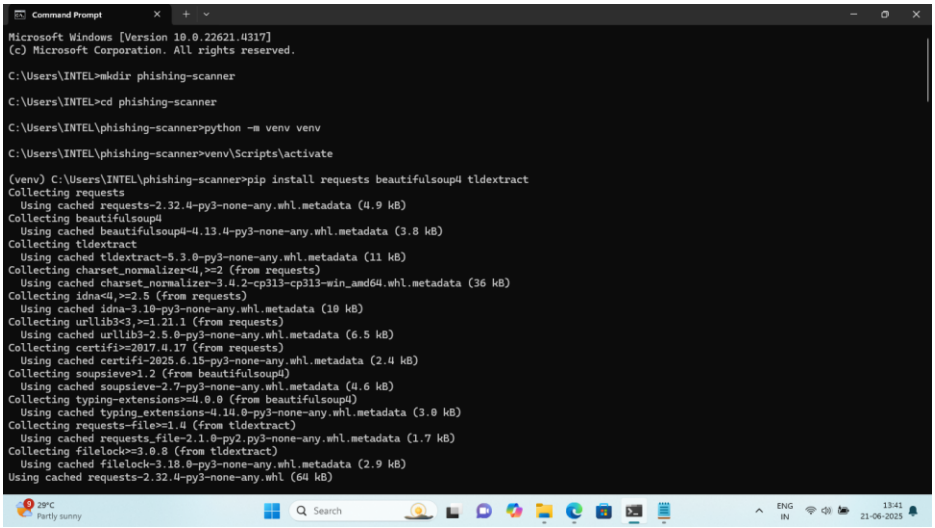
```
mkdir phishing-scanner
cd phishing-scanner
```

2. **Creating a Virtual Environment**

```
python -m venv venv
venv\Scripts\activate
```

3. **Installing Required Libraries**

```
pip install requests beautifulsoup4 tldextract
```



What I Did

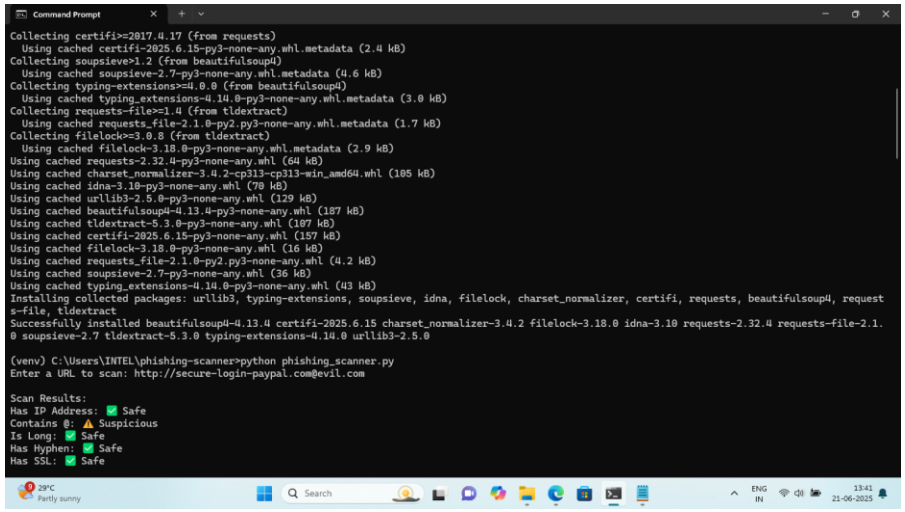
1. Developed the Scanner Script

I wrote a Python script (`phishing_scanner.py`) that prompts the user to input a URL, then runs a series of checks on it.

The following features were analyzed:

- **IP Address in URL** – Typically used to mask malicious destinations
- “Symbol”– Used to obscure true URL destination
- **URL Length** – Very long URLs can signal obfuscation
- **Hyphen in Domain** – Often used to mimic legitimate domains
- **SSL Certificate Presence** – Checked by verifying “https”
- **Domain Age** – Newly created domains are often used in phishing

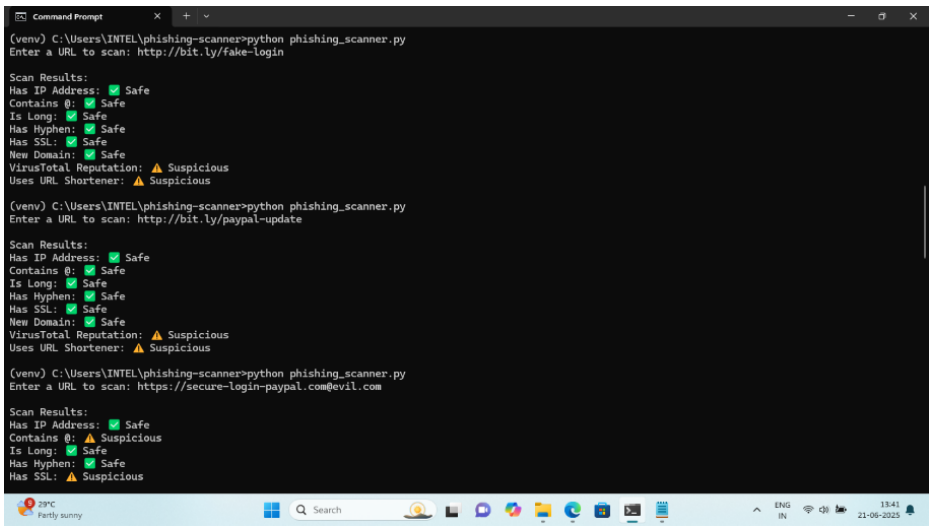
- **URL Shortener** – Services like `bit.ly` or `tinyurl` may conceal the true destination
- **VirusTotal Reputation** – Used public API to gather threat intelligence



2. Ran Tests on Multiple URLs

To validate my scanner, I tested it with a variety of real-world suspicious URLs:

- `http://bit.ly/paypal-update` (Shortened URL)
- `https://secure-login-paypal.com@evil.com` (Use of @ symbol, fake login)
- `https://tinyurl.com/free-gift-card` (URL shortener and likely phishing bait)



The script provided a clear terminal output showing a ✅ for “Safe” or ⚠️ /❌ for “Suspicious” indicators based on the analysis.

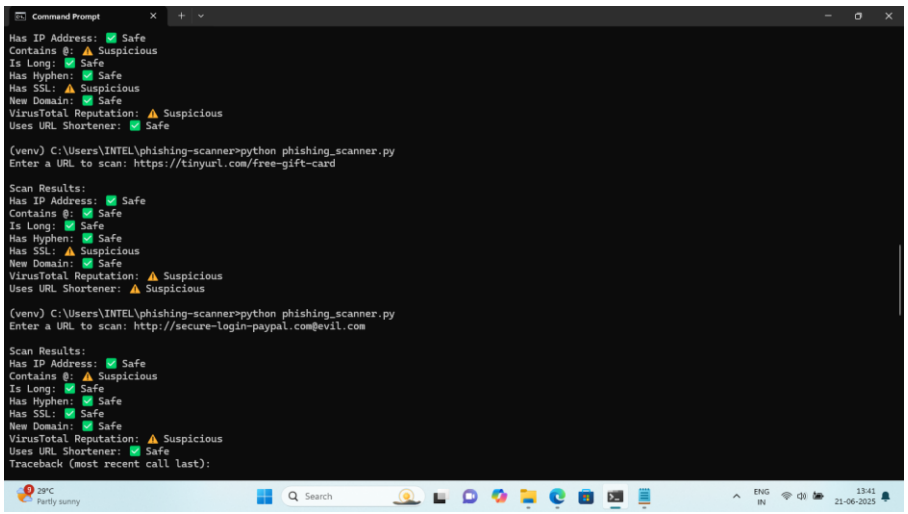
3. Exported Results to Report

Results from each scan were also saved in a human-readable text file (`report.txt`). To handle emojis and non-ASCII characters, I encoded the file in UTF-8.

Example format:

```
Scan Results:
Has IP Address: ✅Safe
Contains @: ❌Suspicious
Is Long: ✅Safe
...
```

This makes it suitable for reviewing results or including in reports.



Outcome

- A fully functional phishing scanner script was created.
- Capable of identifying 8+ suspicious features within a URL.
- Integrated real-time threat intelligence via VirusTotal.
- Demonstrated robustness against multiple real phishing test cases.
- Output successfully exported for reporting.

What I Learned

- Deep understanding of phishing tactics and techniques
- Hands-on experience using Python for cybersecurity applications
- Practical use of third-party APIs (VirusTotal)
- Writing modular, readable, and testable Python code
- Output formatting and file I/O best practices

- Debugging encoding-related issues and understanding Unicode handling in Python

Project Structure

```
phishing-scanner/
├── phishing_scanner.py        # Main script
├── report.txt                 # Scan results output
├── venv/                      # Python virtual environment
├── screenshots/               # Screenshots from the task
```

Final Thoughts

This phishing scanner is a strong entry-level tool for analyzing the risk level of URLs. It can be enhanced further by:

- Incorporating machine learning classifiers
- Expanding the detection logic to include WHOIS records and DNS lookup
- Providing a GUI or web interface

It serves as a valuable asset for security awareness and educational training, especially for recognizing suspicious links before clicking.