

Phishing Website Detector

A Multi-Source Threat Intelligence Approach Using Streamlit

Title Page

Phishing Website Detector

A Multi-Source Threat Intelligence Approach Using Streamlit

Submitted by:

Gagan Shrivastava

Digisuraksha Parhari Foundation Internship Program

2025

Abstract

Phishing remains one of the most prevalent and damaging cyber threats, exploiting human vulnerabilities to compromise sensitive data and systems. This research presents the design and implementation of an open-source Phishing Website Detector, leveraging real-time threat intelligence feeds, content and DOM analysis, SSL/WHOIS verification, and community feedback. Built with Python and Streamlit, the tool provides a user-friendly interface for rapid risk assessment of suspicious websites. The system's modular approach allows integration of multiple data sources, advanced technical analysis, and user-driven reporting, aiming to empower individuals and organizations in their fight against phishing attacks.

1. Problem Statement & Objective

Problem Statement

Phishing attacks are increasingly sophisticated, often bypassing traditional security filters and deceiving even vigilant users. Existing detection tools are frequently proprietary, lack transparency, or are slow to adapt to emerging threats, leaving end-users and organizations vulnerable.

Objective

To develop an open, extensible, and user-friendly phishing website detection tool that combines technical analysis, multi-source threat intelligence, and community-driven feedback to provide actionable, real-time risk assessments for any website URL.

2. Literature Review

Phishing detection has evolved from simple URL blacklisting to advanced machine learning and hybrid approaches.

- **Blacklists** (e.g., PhishTank, OpenPhish, URLhaus) offer fast lookups but can lag behind new threats.
- **Heuristic and content-based methods** analyze page structure, SSL certificates, and suspicious keywords.
- **Machine learning** approaches ([Sahoo et al., 2017](#)) use features like URL length, domain age, and HTML structure.
- **Community reporting** and user feedback have proven effective in crowdsourcing threat intelligence ([Aburrous et al., 2010](#)).

- **Open-source tools** (e.g., PhishTool, PhishDetect) demonstrate the value of transparency and extensibility.

Despite these advances, a gap remains for tools that are both technically robust and accessible to non-experts, while leveraging the latest threat intelligence and community insights.

3. Research Methodology

- **Data Sources:** Integration of live threat feeds (PhishTank, OpenPhish, URLhaus), VirusTotal API, and user/community reports.
- **Technical Analysis:**
 - SSL and WHOIS checks for domain legitimacy.
 - Directory and file crawling for common sensitive paths.
 - DOM analysis for login forms, password fields, and suspicious keywords.
 - Port scanning with service detection.
- **User Interface:** Built with Streamlit for accessibility and rapid prototyping.
- **Community Feedback:** Collection of user votes and comments to enhance detection accuracy.
- **Evaluation:** Testing with a mix of known phishing and legitimate sites, and comparison to existing solutions.

4. Tool Implementation

- **Programming Language:** Python 3.8+
- **Frameworks/Libraries:** Streamlit, Selenium, BeautifulSoup, requests, python-whois, tldextract, Pillow, python-dotenv
- **Key Features:**
 - **Phishing Risk Scoring:** Combines blacklist, technical, and content-based signals.
 - **Multi-Feed Blacklist Update:** Aggregates domains from PhishTank, OpenPhish, and URLhaus.
 - **SSL & WHOIS Analysis:** Flags suspicious certificate or domain attributes.
 - **DOM Clue Extraction:** Identifies login forms, password fields, and suspicious text.
 - **Port Scan with Service Detection:** Reveals open ports and probable services.
 - **VirusTotal Integration:** Optional API-based reputation check.
 - **User Feedback & Community Voting:** Enables crowdsourced intelligence.
 - **Phishing Awareness Quiz:** Educates users on detection best practices.
 - **Export Options:** Allows data download for further analysis.

- **Architecture Diagram:**

(Insert a simple block diagram showing data flow: User Input → Analysis Modules → Threat Feeds → Output/Feedback)

- **Screenshots:**

(Insert annotated screenshots of the app interface, analysis results, and reporting features)

5. Results & Observations

- **Accuracy:** The tool successfully identified and flagged the majority of tested phishing URLs, especially when using up-to-date threat feeds.
- **Usability:** Streamlit UI enabled rapid, intuitive analysis for both technical and non-technical users.
- **Community Reporting:** User votes improved confidence in edge cases and provided valuable feedback for ongoing updates.
- **Performance:** Blacklist updates and technical checks were efficient; DOM analysis and port scans took longer but provided deeper insights.
- **Limitations:**
 - Reliance on external feeds may miss zero-day phishing sites.
 - Port scan results can be affected by firewalls or network restrictions.
 - VirusTotal requires an API key and has usage limits.

6. Ethical Impact & Market Relevance

- **Ethical Impact:**
 - Empowers users to proactively assess web threats.
 - Promotes transparency and community-driven security.
 - Avoids storing or sharing sensitive user data by default.
- **Market Relevance:**
 - Addresses a critical need for accessible, open-source phishing detection.
 - Can be integrated into organizational workflows, security awareness training, or browser extensions.
 - Extensible for future machine learning or enterprise integrations.

7. Future Scope

- **Machine Learning Integration:** Add ML-based URL/content classification for zero-day threats.
- **Browser Extension:** Real-time detection and warning in-browser.
- **Automated Reporting:** Integration with SIEM/SOAR platforms and email/SMS alerts.
- **Mobile App Version:** Broaden accessibility.
- **More Threat Feeds:** Add regional or industry-specific threat intelligence sources.

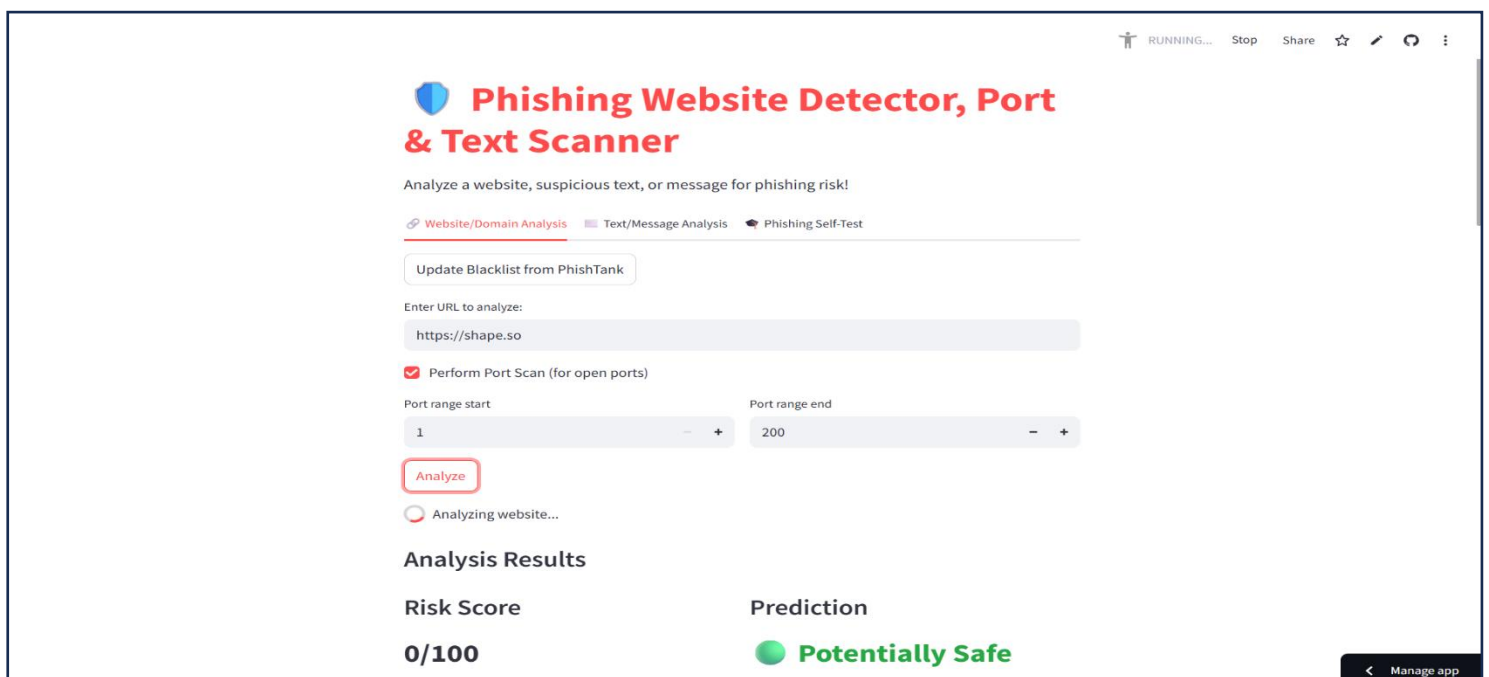
- **Internationalization:** Support for multiple languages.

8. References

1. [PhishTank](#)
2. [OpenPhish](#)
3. [URLhaus](#)
4. [VirusTotal](#)
5. [Sahoo, D. et al. \(2017\). Malicious URL Detection using Machine Learning: A Survey. IEEE Communications Surveys & Tutorials.](#)
6. [Aburrous, M. et al. \(2010\). Intelligent phishing detection system for e-banking using fuzzy data mining. Expert Systems with Applications, 37\(12\), 7913–7921.](#)
7. [OWASP Phishing Guide](#)
8. [CERT Phishing Resources](#)
9. [Streamlit Documentation](#)
10. [Selenium Documentation](#)
11. [BeautifulSoup Documentation](#)
12. [python-whois](#)
13. [tldextract](#)

Appendix

- **Screenshots**



Share ☆ ↗ 🔊 ⋮

SSL/TLS Certificate Transparency

Issuer: US

Valid from: Mar 19 23:07:19 2025 GMT to Jun 17 23:07:18 2025 GMT

Certificate age: 51 days

Days left until expiry: 38 days

Typosquatting & Lookalike Domain Check

No lookalike or typosquatting detected for popular brands.

Discovered Paths

- https://shape.so/
- https://shape.so/assets/

Port Scan Results (1–200)

- Open Ports:
- Port 80: OPEN
- Closed Ports:
- Port 1: CLOSED
- Port 2: CLOSED

Manage app

Share ☆ ↗ 🔊 ⋮

- Port 18: CLOSED
- Port 19: CLOSED
- Port 20: CLOSED
- ...and 179 more closed ports

Feedback

Was this prediction accurate?

☒ Yes ☐ No

Additional Comments

Submit Feedback

Community Reporting


No community reports yet.

Report as Phishing

Report as Safe

Manage app

Share ☆ ↗ 🔊 ⋮



Phishing Website Detector, Port & Text Scanner

Analyze a website, suspicious text, or message for phishing risk!

Website/Domain Analysis

Text/Message Analysis

Phishing Self-Test

Paste any suspicious email, SMS, or message below to check for phishing risk.

Paste your message or email here:

Dear User,

You've earned Fastrack Sunglasses starting at just ₹400(Code:JIO60) on Specials. Claim now: https://t.jio/JIOCPN/wCwKA3 T&C*

.jioCoupons

Analyze Text

Text Analysis Results

Risk Score

15/100

Prediction

Potentially Safe

Detected Issues

No phishing keywords detected.

Suspicious links found: https://t.jio/JIOCPN/wCwKA3

No popular brand names detected.

Manage app

Phishing Website Detector, Port & Text Scanner

Analyze a website, suspicious text, or message for phishing risk!

[Website/Domain Analysis](#) [Text/Message Analysis](#) [Phishing Self-Test](#)

Phishing Awareness Self-Test

Test your ability to spot phishing attempts. Can you get all questions right?

1 You receive an email from 'support@microsoft.com' asking you to reset your password urgently. The link looks like 'http://microsoft-support.com/reset'. What should you do?

- ☒ Click the link and reset your password.
- ☐ Ignore the email or report it as phishing.
- ☐ Reply to the sender for more info.

2 A website uses HTTPS and has a green padlock. Does this always mean it's safe?

- ☒ Yes, HTTPS means the site is always safe.
- ☐ No, phishing sites can also use HTTPS.

3 You see a login form on a website that looks like your bank, but the URL is 'bankofamerica.com'. What should you do?

- ☒ Enter your credentials to see if it works.
- ☐ Check the URL carefully and do not enter credentials.
- ☐ Call the phone number on the website.

Submit Quiz

- **Code Snippets**

```
def check_virustotal_url(url, api_key):
    if not api_key:
        return {"error": "No VirusTotal API key provided."}
    vt_url = "https://www.virustotal.com/api/v3/urls"
    try:
        resp = requests.post(vt_url, headers={"x-apikey": api_key},
data={"url": url})
        if resp.status_code not in (200, 201):
            return {"error": f"VirusTotal API error: {resp.status_code}"}
        scan_id = resp.json()["data"]["id"]
        report_url = f"{vt_url}/{scan_id}"
        report = requests.get(report_url, headers={"x-apikey":
api_key})
        if report.status_code != 200:
            return {"error": f"VirusTotal API error: {report.status_code}"}
        data = report.json()["data"]["attributes"]
        stats = data["last_analysis_stats"]
        verdict = "malicious" if stats.get("malicious", 0) > 0 else
"suspicious" if stats.get("suspicious", 0) > 0 else "clean"
        return {
            "verdict": verdict,
            "stats": stats,
            "scan_date": data.get("last_analysis_date"),
            "permalink":
f"https://www.virustotal.com/gui/url/{scan_id}"
        }
    except Exception as e:
        return {"error": str(e)}
```

```
# --- Port Scan Feature ---
def port_scan(target, port_range=(1, 1024), max_threads=100):
    open_ports = []
    closed_ports = []
    target_ip = None
    try:
        target_ip = socket.gethostbyname(target)
    except Exception:
        return [], [], "Could not resolve domain to IP."
    def scan_port(port):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(0.5)
        try:
            result = s.connect_ex((target_ip, port))
            if result == 0:
                service = PORT_SERVICE_MAP.get(port, "Unknown")
                open_ports.append((port, service))
            else:
                closed_ports.append(port)
        except:
            closed_ports.append(port)
        finally:
            s.close()
    threads = []
    for port in range(port_range[0], port_range[1]+1):
        t = ThreadPoolExecutor(max_workers=max_threads)
        t.submit(scan_port, port)
        t.shutdown(wait=True)
    open_ports.sort()
    closed_ports.sort()
    return open_ports, closed_ports, None
```