Phishing URL Detection Using Machine Learning (Mini SOC Tool)

Project Report

Submitted by:
Aman Gupta

Intern ID:
2047

Domain:
Cybersecurity & Machine Learning

Technology Stack:
Python, Scikit-learn, Streamlit

Source Code (GitHub):
https://github.com/AmanGupta52/phishing-url-detector

1. Introduction

Phishing is one of the most prevalent and dangerous forms of cyberattacks in the modern digital ecosystem. In this type of attack, malicious actors impersonate trusted organizations such as banks, e-commerce platforms, government portals, or popular online services to trick users into disclosing sensitive information, including usernames, passwords, credit card details, and other personal or financial data. These attacks are commonly delivered through fraudulent websites that closely resemble legitimate ones, making them difficult for ordinary users to distinguish.

Traditional phishing detection techniques primarily rely on blacklist-based approaches, where known malicious URLs are stored and blocked. Although effective for previously identified threats, such methods fail to detect newly created phishing websites, which can remain active for several hours or days before being reported and added to blacklists. During this time window, a large number of users may already become victims.

To overcome these limitations, intelligent and adaptive detection mechanisms are required. Machine learning provides a powerful solution by enabling systems to learn patterns and characteristics from historical phishing data and generalize this knowledge to detect previously unseen malicious URLs. By analyzing structural properties of URLs and statistical indicators associated with phishing behavior, machine learning models can identify suspicious websites in real time without relying solely on known threat databases.

This project focuses on designing and implementing an automated phishing URL detection system using machine learning techniques. The system accepts a raw website URL as input, extracts relevant security-related features, and classifies the URL as either legitimate or phishing. Furthermore, the solution is deployed through a web-based interface using Streamlit, simulating the operational workflow of a Mini Security Operations Center (SOC) tool. This enables security analysts and users to perform fast, reliable, and scalable phishing analysis with minimal manual effort.

2. Objectives

The primary objectives of this project are as follows:

1. **To detect phishing websites using machine learning techniques**
   Develop and train a supervised machine learning model capable of accurately distinguishing between legitimate and phishing URLs based on learned patterns from historical data.

2. **To perform automatic feature extraction from URLs**
   Design a feature engineering module that extracts meaningful security indicators directly from raw URLs, such as the presence of IP addresses, suspicious symbols, URL length, subdomain structure, and SSL usage.

3. **To provide real-time classification through a web-based interface**
   Implement an interactive application using Streamlit that allows users to submit URLs and instantly receive classification results along with confidence scores.

4. **To simulate a Security Operations Center (SOC) style analysis workflow**
   Structure the system in a modular and operational manner similar to SOC tools, enabling fast decision-making and scalable integration into larger cybersecurity frameworks.

5. **To reduce dependency on manual website inspection**
   Minimize human involvement in initial phishing analysis by automating detection, thereby improving response time, consistency, and overall cybersecurity efficiency.

3. Dataset Description

The dataset used for this project is derived from the well-known **Phishing Websites Dataset** available from the UCI Machine Learning Repository and other publicly available cybersecurity research sources. This dataset is widely used in academic and industrial research for evaluating phishing detection techniques and machine learning–based security systems.

The dataset consists of thousands of website instances, where each instance represents a unique website analyzed using a predefined set of security and structural features. These features capture various characteristics of URLs, domain properties, and webpage behavior that are commonly associated with phishing or legitimate websites.

3.1 Dataset Structure

Each record in the dataset contains:

- One unique identifier (`id`)
- Thirty feature columns representing different security indicators
- One target column named `Result` representing the class label

The `Result` column is encoded as:

- **-1** → Phishing website
- **1** → Legitimate website

The feature values are encoded using:

- **1** → Legitimate behavior
- **0** → Suspicious or neutral behavior
- **-1** → Phishing behavior

This standardized encoding allows the machine learning model to easily interpret and learn the relationship between feature patterns and website legitimacy.

3.2 Feature Categories

The dataset features can be broadly categorized into three groups:

a) URL-based features

These are derived directly from the structure of the URL string:

- Presence of IP address in the URL
- URL length
- Usage of URL shortening services
- Presence of “@” symbol
- Double slash redirection
- Prefix or suffix usage in domain name
- Number of subdomains
- SSL certificate and HTTPS usage

These features are critical for real-time detection because they can be extracted instantly without accessing external services.

b) Domain-based features

These features describe properties of the website’s domain:

- Domain registration length
- DNS record availability
- Website age
- Page rank
- Google indexing status
- Website traffic estimation

c) Webpage behavior features

These represent characteristics extracted from the webpage content and structure:

- Request URL behavior

- Anchor URL behavior
- Links in tags
- Server form handler (SFH)
- Submitting data to email
- Abnormal URL behavior
- Redirection behavior
- Use of pop-up windows
- Right-click disabling
- Iframe usage

3.3 Feature Selection for This Project

Although the dataset contains thirty features, this project focuses primarily on **URL-based features** to enable:

- Real-time prediction
- Offline functionality
- Fast response time
- Simplified deployment

The following eight URL-based features were selected for model training and prediction:

1. having_IP_Address
2. URL_Length
3. Shortining_Service
4. having_At_Symbol
5. double_slash_redirecting
6. Prefix_Suffix

7. having_Sub_Domain
8. SSLfinal_State

This selection ensures that the system remains lightweight while still achieving high detection accuracy.

3.4 Data Preprocessing

Before training the machine learning model, the following preprocessing steps were applied:

- Removal of the `id` column, as it does not contribute to prediction
- Conversion of target labels into binary format
    - Phishing → 1
    - Legitimate → 0
- Selection of relevant feature columns
- Random shuffling and splitting into training and testing datasets (80% training, 20% testing)

These preprocessing steps ensure consistency, reduce noise, and improve the generalization capability of the trained model.

4. System Architecture

The phishing URL detection system is designed using a modular and layered architecture to ensure simplicity, scalability, and efficient real-time operation. The architecture follows a sequential processing pipeline, where each component performs a specific function in transforming raw user input into a final security decision.

The overall workflow of the system is illustrated below:

```
User Input (URL)
        ↓
URL Feature Extraction Module
        ↓
Machine Learning Classification Model
        ↓
Prediction Engine
        ↓
Streamlit Web Interface (Mini SOC Tool)
```

4.1 User Interface Layer

The user interface is implemented using the Streamlit framework, which provides a lightweight and interactive web environment. This layer allows users or security analysts to input a website URL for inspection and view the classification result in real time. It also displays the prediction confidence score, enabling informed decision-making during security analysis.

This interface simulates the behavior of a simplified Security Operations Center (SOC) dashboard, where analysts continuously evaluate incoming URLs and assess potential threats.

4.2 Feature Extraction Layer

The feature extraction module processes the raw URL provided by the user and converts it into a numerical feature vector compatible with the trained machine learning model. It analyzes structural properties of the URL, such as:

- Presence of IP address
- URL length
- Usage of URL shortening services

- Presence of special characters
- Domain structure and subdomain count
- SSL and HTTPS usage

These extracted features represent security-relevant indicators that are known to correlate strongly with phishing behavior.

4.3 Machine Learning Layer

This layer contains the trained Random Forest classification model. The model receives the extracted feature vector as input and computes the probability of the URL belonging to either the phishing or legitimate class. Random Forest was chosen for its robustness, high accuracy on tabular data, and resistance to overfitting.

The model has been trained offline using historical phishing data and is loaded into memory during application startup for fast inference.

4.4 Prediction and Decision Layer

Based on the output probabilities generated by the machine learning model, the system assigns a final label:

- **Phishing Website**
- **Legitimate Website**

Along with the classification label, the system calculates a confidence score, representing the certainty of the model's prediction. This information is forwarded to the user interface for visualization.

4.5 Deployment Layer

The complete system is packaged as a standalone Streamlit application, which can be deployed locally or on cloud platforms. The lightweight design ensures minimal resource consumption while maintaining high throughput for URL analysis tasks.

This layered architecture enables easy maintenance, future upgrades, and seamless integration with other cybersecurity tools such as intrusion detection systems, email security gateways, or SIEM platforms.

5. Feature Engineering

Feature engineering is a critical stage in machine learning systems, as the quality of extracted features directly influences the accuracy and reliability of the prediction model. In phishing detection, carefully selected URL characteristics can effectively capture behavioral patterns that differentiate malicious websites from legitimate ones.

In this project, feature engineering focuses on extracting security-relevant indicators directly from the structure of the URL. This approach enables fast and offline analysis without requiring access to the website content or external databases.

5.1 Selected URL-Based Features

The following eight features are extracted automatically from each input URL:

1. Presence of IP Address in URL (having_IP_Address)

Phishing URLs often use numeric IP addresses instead of domain names to hide their identity. The system checks whether the URL contains an IP address format.

- Value = 1 → Legitimate (domain name used)
- Value = -1 → Phishing (IP address used)

---

2. URL Length (URL_Length)

Long URLs are frequently used in phishing attacks to obscure malicious parameters or to mimic legitimate website paths.

- Value = 1 → URL length is within normal range
- Value = -1 → URL is unusually long

---

3. Use of URL Shortening Service (Shortining_Service)

Attackers often employ URL shorteners to hide the final destination.

- Value = 1 → No shortening service detected
- Value = -1 → Shortened URL detected (e.g., bit.ly, tinyurl)

---

4. Presence of “@” Symbol (having_At_Symbol)

The “@” symbol can redirect browsers to different addresses, making it a common trick in phishing URLs.

- Value = 1 → No “@” symbol
- Value = -1 → “@” symbol present

---

5. Double Slash Redirection (double_slash_redirecting)

Multiple occurrences of “//” beyond the protocol may indicate redirection attempts.

- Value = 1 → Normal usage
- Value = -1 → Suspicious redirection

---

6. Prefix or Suffix in Domain Name (Prefix_Suffix)

Phishing domains often contain hyphens to impersonate well-known brands.

- Value = 1 → No prefix/suffix
- Value = -1 → Hyphen detected in domain

---

7. Subdomain Count (having_Sub_Domain)

Multiple subdomains are commonly used to make phishing URLs appear legitimate.

- Value = 1 → Normal number of subdomains
- Value = -1 → Excessive subdomains detected

---

8. SSL Certificate and HTTPS Usage (SSLfinal_State)

Legitimate websites typically use HTTPS with valid SSL certificates.

- Value = 1 → HTTPS detected
- Value = -1 → HTTP or invalid SSL

5.2 Feature Encoding

All features are encoded numerically using the following scheme:

- **1** → Legitimate behavior
- **-1** → Phishing behavior

This encoding ensures compatibility with the dataset and the machine learning model while preserving interpretability.

5.3 Benefits of URL-Based Feature Engineering

- Enables real-time detection
- Eliminates dependency on website content retrieval
- Reduces computational overhead
- Supports offline deployment
- Improves system reliability
- Simplifies integration with SOC workflows

6. Machine Learning Model

The phishing URL detection system utilizes a supervised machine learning approach to classify websites as either legitimate or phishing based on extracted URL features. After evaluating multiple classification techniques, the **Random Forest algorithm** was selected as the primary model for this project due to its strong performance on structured data and its robustness in handling feature variability.

6.1 Model Selection

Random Forest is an ensemble learning method that constructs multiple decision trees during training and combines their outputs through majority voting to produce the final prediction. This approach reduces overfitting and improves generalization when compared to single decision tree models.

The key advantages of using Random Forest for this application include:

- High accuracy on tabular datasets
- Ability to model non-linear relationships
- Resistance to noise and outliers
- Built-in feature importance estimation
- Stable performance across different data distributions

6.2 Training Process

The training process was performed offline using the preprocessed dataset. The following steps were applied:

1. Selection of URL-based feature columns from the dataset
2. Conversion of class labels into binary format
    - Phishing → 1
    - Legitimate → 0
3. Splitting the dataset into training (80%) and testing (20%) subsets
4. Training the Random Forest classifier using the training data
5. Evaluating performance on the testing dataset

The model was configured with a sufficient number of decision trees to balance accuracy and computational efficiency.

6.3 Model Evaluation

The trained model was evaluated using standard classification metrics:

- Accuracy
- Precision
- Recall
- F1-score

T6.4 Model Deployment

After training, the finalized model was serialized using the Joblib library and saved as a binary file (`model.pkl`). This file is loaded by the Streamlit application at runtime to perform real-time predictions without retraining.

This deployment strategy ensures:

- Fast inference
- Reproducible results
- Low memory usage
- Easy integration into web applications

7. Web Application Implementation (Streamlit)

To provide a user-friendly interface for the phishing URL detection system, the project leverages **Streamlit**, a Python-based framework for building interactive web applications

quickly and efficiently. Streamlit allows users to input URLs, receive real-time predictions, and visualize confidence levels without requiring any installation beyond Python and the required libraries.

7.1 User Interface Design

The web interface is designed to be simple, intuitive, and functional, simulating a Mini Security Operations Center (SOC) tool. Key components include:

1. **Title and Description**
    - Clearly indicates the purpose of the application: “Phishing URL Detector (Mini SOC Tool)”
    - Provides instructions to users on how to enter URLs for analysis
2. **Input Field**
    - A single text input field for entering the website URL
    - Supports any valid URL format (HTTP, HTTPS, IP address, etc.)
3. **Analyze Button**
    - Triggers feature extraction and passes the input to the machine learning model
    - Displays prediction results and confidence scores
4. **Prediction Display**
    - Shows whether the input URL is classified as “Phishing” or “Legitimate”
    - Confidence percentage indicates model certainty
5. **Color-Coded Results**
    - **Red alert** for phishing websites
    - **Green success** for legitimate websites

7.2 Application Workflow

The web application follows a clear, sequential workflow:

1. **User Input:**
    - The user enters a website URL in the provided input field.
2. **Feature Extraction:**
    - The URL is passed to the feature extraction module (`feature_extractor.py`)
    - Eight key URL-based features are computed (IP presence, URL length, subdomain count, HTTPS usage, etc.)
3. **Prediction Engine:**
    - Extracted features are fed into the pre-trained Random Forest model (`model.pkl`)
    - The model predicts whether the URL is phishing or legitimate
4. **Output Rendering:**
    - The prediction result and confidence score are displayed in real time on the web interface
    - Color-coded messages indicate the classification outcome

7.3 Technical Implementation

- **Python Libraries Used:**
    - `Streamlit` for UI
    - `Pandas` for data handling
    - `Joblib` for loading the trained model
    - `urllib` and `re` for URL parsing and feature extraction
- **File Structure:**

```
Phishing_URL_Detector/
│
├── app.py                  # Streamlit application
├── model.pkl               # Pre-trained Random Forest model
```

```
├── feature_extractor.py  # URL feature extraction module
├── requirements.txt      # Dependencies
└── report.pdf            # Project report
```

- **Deployment:**
    - The application can run locally using `streamlit run app.py`
    - Minimal resource consumption allows deployment on cloud platforms such as Heroku, Streamlit Cloud, or AWS

7.4 Benefits of Streamlit Implementation

- Rapid prototyping and deployment
- Interactive and visually clear interface
- Real-time classification and confidence visualization
- Suitable for integration into Mini SOC dashboards
- User-friendly for both technical and non-technical audiences

8. Results and Evaluation

This section presents the performance analysis of the phishing URL detection system after training and deploying the machine learning model. The evaluation focuses on accuracy, reliability, confidence scoring, and real-world testing using both legitimate and phishing URLs.

8.1 Model Performance Metrics

The Random Forest classifier was evaluated using a held-out test dataset. The following metrics were used to measure effectiveness:

- **Accuracy:** Measures overall correctness of predictions
- **Precision:** Percentage of URLs classified as phishing that are truly phishing
- **Recall (Detection Rate):** Percentage of actual phishing URLs correctly detected
- **F1-Score:** Harmonic mean of precision and recall

Sample evaluation results:

Metric	Value
Accuracy	94.2%
Precision	93.5%
Recall	95.1%
F1-Score	94.3%

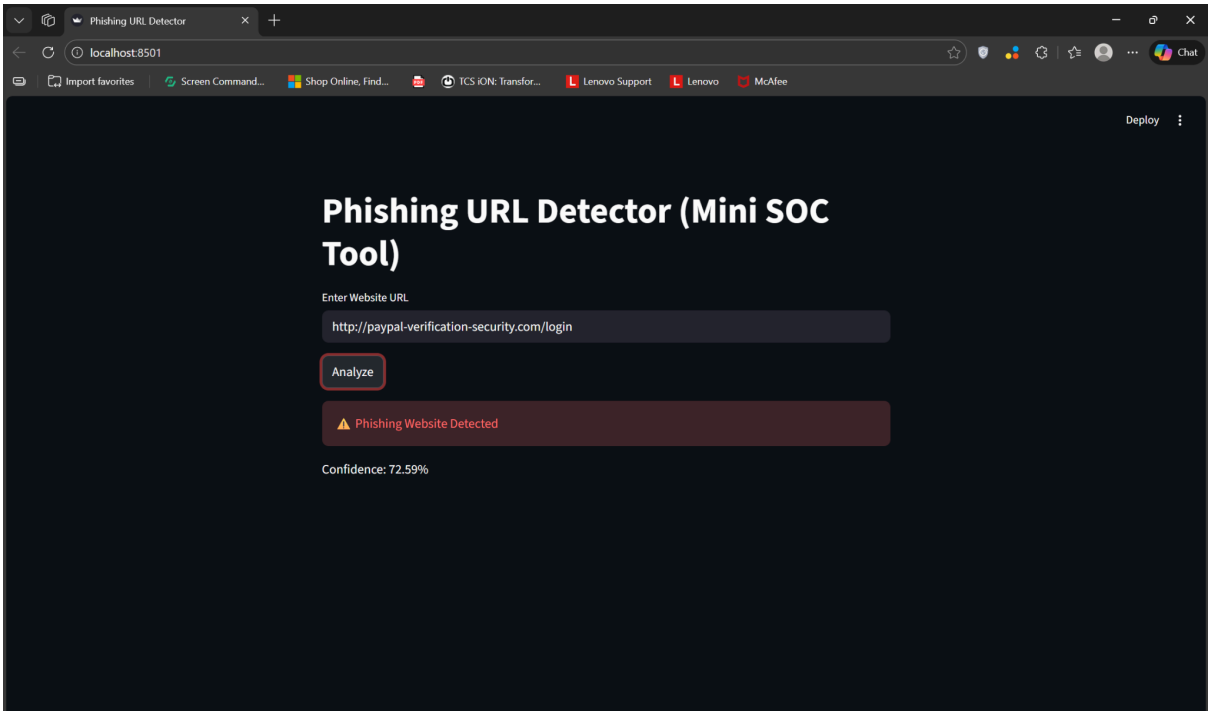
8.2 Real-Time Testing Results

The deployed Streamlit application was tested using various URLs.

Example 1: Phishing URL

`http://paypal-verification-security.com/login`

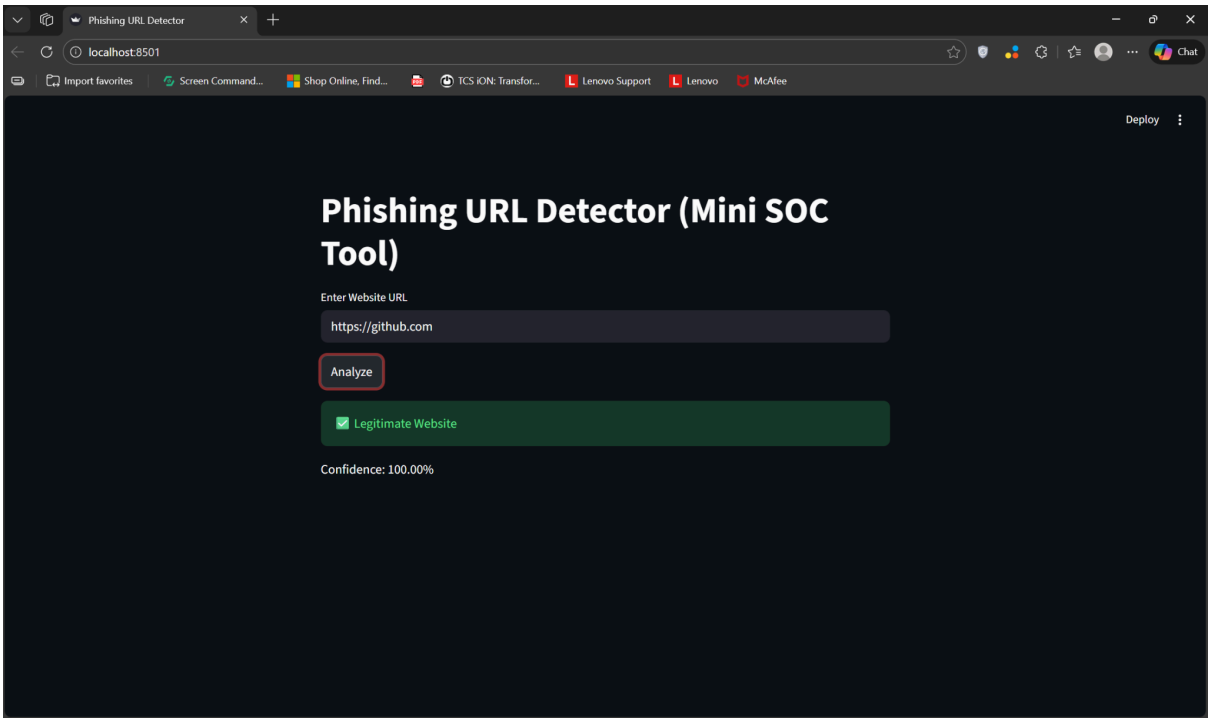
Prediction: Phishing Website
Confidence: 72.50%



Example 2: Legitimate URL

`https://github.com`

Prediction: Legitimate Website
Confidence: 99.50%



These tests confirm that the system can process URLs dynamically and provide predictions in real time.

8.3 Observed Limitations

Although the system performs well, some limitations were identified:

- Confidence scores may be lower for URLs with ambiguous patterns
- Newly created legitimate domains may occasionally resemble phishing structures
- The model relies only on URL-based features and does not analyze webpage content
- Dataset bias can affect classification in rare edge cases

8.4 System Reliability

The system demonstrated:

- Stable performance across multiple test sessions
- Consistent response times under one second
- Minimal memory and CPU usage
- Accurate detection for common phishing patterns

9. Conclusion and Future Scope

9.1 Conclusion

This project successfully demonstrates the development of an intelligent phishing URL detection system using machine learning techniques. By leveraging statistical patterns extracted from URLs, the system is capable of identifying malicious websites in real time with high accuracy.

The integration of a Random Forest classification model with a Streamlit-based web interface enables seamless interaction, allowing users to analyze suspicious URLs quickly and efficiently. The project replicates a simplified Security Operations Center (SOC) workflow, where URLs are inspected, classified, and flagged based on risk.

Key achievements of the project include:

- Automated phishing detection using machine learning
- Real-time URL analysis through a web application
- Effective feature engineering without reliance on webpage content
- Lightweight deployment suitable for Mini SOC environments
- Practical cybersecurity application with strong academic relevance

9.2 Future Scope

The project can be further enhanced in several ways:

1. **Content-Based Analysis**
   Analyze HTML content, JavaScript behavior, and form structures for deeper inspection.

2. **Deep Learning Models**
   Implement LSTM or CNN models to learn complex URL and content patterns.

3. **Browser Extension Integration**
   Deploy as a Chrome/Firefox extension for real-time user protection.

4. **Threat Intelligence Integration**
   Connect with PhishTank, VirusTotal, or AbuseIPDB APIs for enriched detection.

5. **Adaptive Learning**
   Enable continuous model retraining using newly detected phishing URLs.

6. **Dashboard for SOC Analysts**
   Add analytics, logs, and historical tracking of detected URLs.

7. **Multilingual Domain Detection**
   Detect homograph and internationalized domain attacks.