Phishing URL Detection: Feature Extraction and Dataset Creation

1. Introduction

This Python script is designed to create a dataset for phishing URL detection. It extracts various features from both legitimate and phishing URLs, which can be used to train machine learning models for phishing detection. The script utilizes multiple data sources and implements parallel processing for efficient feature extraction.

2. Libraries and Dependencies

The script uses several Python libraries:

- pandas: For data manipulation and CSV file operations
- requests: For making HTTP requests to fetch web content
- zipfile and gzip: For handling compressed files
- io and csv: For file I/O operations
- urllib.parse: For URL parsing
- BeautifulSoup: For HTML parsing
- re: For regular expression operations
- time and random: For implementing delays and randomization
- tqdm: For progress bars
- concurrent.futures: For parallel processing
- multiprocessing: For utilizing multiple CPU cores
- whois: For domain WHOIS lookups
- socket: For IP address operations
- datetime: For date and time operations

3. Data Sources

The script uses three main data sources:

1. **Tranco List**: A list of popular domains, used to determine website popularity.
2. **Majestic Million**: Another list of popular domains, used in conjunction with Tranco.
3. **PhishTank**: A source of known phishing URLs.

4. Main Functions

4.1 Data Retrieval Functions

get_tranco_data()

- Downloads the Tranco list
- Processes the ZIP file
- Returns a dictionary of domains and their ranks

get_majestic_data()

- Downloads the Majestic Million CSV
- Processes the CSV data
- Returns a dictionary of domains and their ranks

get_phishtank_data()

- Attempts to download PhishTank data
- Handles rate limiting by creating a dummy dataset if necessary
- Processes the gzipped CSV data
- Returns a DataFrame of phishing URLs

4.2 Feature Extraction

feature_extraction(url, label, tranco_dict, majestic_dict)

This is the core function that extracts features from a given URL. It extracts the following features:

1. **Domain of the URL**
   Here, the the domain present in the URLs are extracted. This feature is not that significant might be dropped in the training.
2. **IP Address in the URL**
   here we are checking for the presence of IP address in the URL. URLs may have IP address instead of domain name. If an IP address is used as an alternative of the domain name in the URL, we can be sure that someone is trying to steal personal information with this URL.
   If the domain part of URL has IP address, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
3. **"@" Symbol in URL**
   Checks for the presence of '@' symbol in the URL. Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.
   If the URL has '@' symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).
4. **URL Length**
   Computes the length of the URL. Phishers often use long URLs to obscure the suspicious part in the address bar. In this project, if the URL length is 54 characters or more, it is classified as phishing; otherwise, it is classified as legitimate.

If the URL length is >= 54 characters, this feature is assigned a value of 1 (phishing); otherwise, it is 0 (legitimate).

5. **URL Depth**

Calculates the depth of the URL, which is the number of sub-pages based on the number of '/' characters.
This feature is represented numerically based on the URL.

6. **Presence of "http/https" in Domain**

Checks for the presence of "http/https" in the domain part of the URL.
Phishers may add the "HTTPS" token to the domain to deceive users.
If "http/https" is found in the domain part, this feature is assigned a value of 1 (phishing); otherwise, it is 0 (legitimate).

7. **URL Shortening Services**

URL shortening is a method where a URL is made significantly shorter while still leading to the required webpage, achieved through an "HTTP Redirect" on a short domain name.
If the URL uses shortening services, this feature is assigned a value of 1 (phishing); otherwise, it is 0 (legitimate).

8. **Presence of "-" in Domain**

Checks for the presence of a dash ("-") in the domain part of the URL.
Legitimate URLs rarely use the dash symbol. Phishers often add prefixes or suffixes separated by a dash to make the domain name appear legitimate.
If the domain part of the URL contains a "-", this feature is assigned a value of 1 (phishing); otherwise, it is 0 (legitimate).

9. **DNS Record**

Phishing websites often have unrecognized identities in the WHOIS database or no records for the hostname. If the DNS record is empty or not found, this feature is assigned a value of 1 (phishing); otherwise, it is assigned a value of 0 (legitimate).

10. **Web Traffic**

This feature assesses the popularity of a website by measuring the number of visitors and the pages they visit. Phishing websites, due to their short lifespan, may not be recognized by the Majestic Million and Tranco Database (For more reliability, I have combined data from multiple sources.). In our dataset, legitimate websites typically rank among the top 100,000. If a domain has no traffic or is not recognized by the Majestic Million and Tranco Database, it is classified as “Phishing”.
If the domain's rank is less than 100,000, the value of this feature is 1 (phishing); otherwise, it is 0 (legitimate).

11. **End Period of Domain**

This feature is also derived from the WHOIS database. It calculates the remaining domain time by finding the difference between the expiration time and the current time. For this project, a legitimate domain is considered to have an end period of 6 months or less.

If the end period of the domain is greater than 6 months, the value of this feature is 1 (phishing); otherwise, it is 0 (legitimate).

12. **IFrame Redirection**

    An IFrame is an HTML tag used to display an additional webpage within the current one. Phishers often use the “iframe” tag invisibly by setting the “frameBorder” attribute to 0, making it visually undetectable.
    If the iframe is empty or a response is not found, the value assigned to this feature is 1 (phishing); otherwise, it is 0 (legitimate).

13. **Status Bar Customization**

    Phishers may use JavaScript to display a fake URL in the status bar to deceive users. This feature is extracted by examining the webpage's source code, particularly looking for the “onMouseOver” event to see if it alters the status bar.
    If the response is empty or if “onMouseOver” is found, the value assigned to this feature is 1 (phishing); otherwise, it is 0 (legitimate).

14. **Disabling Right Click**

    Phishers often use JavaScript to disable the right-click function, preventing users from viewing and saving the webpage source code. This feature is similar to the use of “onMouseOver” to hide links. For this feature, the source code is checked for the event “event.button==2” to determine if right-click is disabled.
    If the response is empty or if “event.button==2” is found, the value assigned to this feature is 1 (phishing); otherwise, it is 0 (legitimate).

15. **Website Forwarding**

    A key distinction between phishing and legitimate websites is the number of times a site is redirected. In our dataset, legitimate websites are redirected a maximum of once, while phishing websites typically have at least four redirects.

4.3 Helper Functions

feature_extraction_with_retry()

- Implements a retry mechanism for feature extraction to handle temporary failures.

process_url()

- Wrapper function for feature extraction to be used in parallel processing.

parallel_feature_extraction()

- Implements parallel processing of URLs for faster feature extraction.

5. Main Execution Flow

1. Download and process data from Tranco and Majestic Million.
2. Attempt to download PhishTank data, use a dummy dataset if rate-limited.
3. Combine domains from Tranco and Majestic Million.
4. Randomly select 5000 legitimate domains and 5000 phishing URLs.
5. Extract features from legitimate URLs in parallel.
6. Extract features from phishing URLs in parallel.
7. Combine the extracted features into a single DataFrame.
8. Save the resulting dataset as a CSV file.

6. Output

The script produces a CSV file named 'phishing_detection_dataset.csv' containing the extracted features for both legitimate and phishing URLs. Each row represents a URL, and columns represent different features and the label (0 for legitimate, 1 for phishing).

7. Conclusion

This script provides a comprehensive solution for creating a dataset for phishing URL detection. It extracts a wide range of features that capture various aspects of URLs and websites, making it suitable for training machine learning models for phishing detection. The use of parallel processing and multiple data sources enhances its efficiency and the quality of the resulting dataset.