Brainwave

MATRIX SOLUTIONS

منـة الله اشرف على محمد

Task 1

Phishing Link Scanner Report

1. Introduction

Phishing attacks trick users into revealing sensitive information by disguising malicious links as legitimate ones. In this project, I built a Python tool that classifies URLs as either **Phishing** or **Legitimate** based on a set of extracted features.

2. Data Collection

1. Phishing Data

- Source: PhishTank (Online Valid CSV)
- URL: `https://data.phishtank.com/data/online-valid.csv`
- Original size: approximately 600,000 URLs

2. Legitimate Data

- Source: Majestic Million (CSV)
- URL: `https://downloads.majestic.com/majestic_million.csv`
- Key column: `Domain`

3. Preprocessing Steps

- I extracted the `url` column from the PhishTank file and the `Domain` column from the Majestic Million file (renamed to `url`).
- I merged both datasets, assigned `label = 1` for phishing and `label = 0` for legitimate, and shuffled the combined data.
- To avoid long WHOIS lookups and speed up training, I sampled **2,000** URLs at random.

3. Feature Engineering

I wrote a feature extraction function that creates these numerical features for each URL:

Feature	Description
`url_length`	Number of characters in the URL
`num_dots`	Count of "." characters
`num_hyphens`	Count of "-" characters
`has_at`	Presence of "@" symbol (1 if present, 0 otherwise)
`num_digits`	Count of numeric characters
`subdomain_count`	Number of subdomain segments (split by ".")
`suspicious_tld`	Indicator for TLDs often used in phishing (e.g. `.tk, .ga, .cf, .ml, .gq`)
`domain_age_days`	Set to 0 (WHOIS lookups disabled due to timeouts)

Why these features?

- Phishing URLs often include many subdomains, hyphens, or unusual TLDs to obscure their real destination.

- The "@" symbol can redirect browsers to hidden parts of the URL.

- Longer URLs with many parameters may indicate attempts to mask malicious payloads.

4. Model Training

- I used a **Random Forest** classifier with 100 trees (`n_estimators=100`) and a fixed seed (`random_state=42`).

- I split the data into **80% training** (1,600 URLs) and **20% testing** (400 URLs).

- The training script (`src/train.py`) loads data, extracts features, trains the model, evaluates it, and saves it to `models/phish_detector.pkl`.

5. Evaluation Results

- Test set size: 400 URLs
- Confusion Matrix:

```
[[376,   2],   # 376 legitimate URLs correctly classified, 2 false positives
 [  2,  20]]   # 20 phishing URLs correctly classified, 2 false negatives
```

- Accuracy: 0.99
- Phishing class (label=1): Precision = 0.91, Recall = 0.91, F1-score = 0.91
- Legitimate class (label=0): Precision = 0.99, Recall = 0.99, F1-score = 0.99

I am pleased with the high overall accuracy, though the phishing class has room for improvement.

6. CLI Scanner Usage

I created a command-line script (`src/scan.py`) that lets me scan URLs directly:

```bash
# Single URL scan
t> python src/scan.py http://example.com
# Output: http://example.com ➜ Legitimate
# Multiple URLs scan
```

```
t> python src/scan.py https://secure-login.tk/login?user=abc http://sub.test-site.ga/path/page.html

# Output:

# https://secure-login.tk/login?user=abc ➜ Legitimate

# http://sub.test-site.ga/path/page.html ➜ Phishing
```

7. Limitations & Future Work

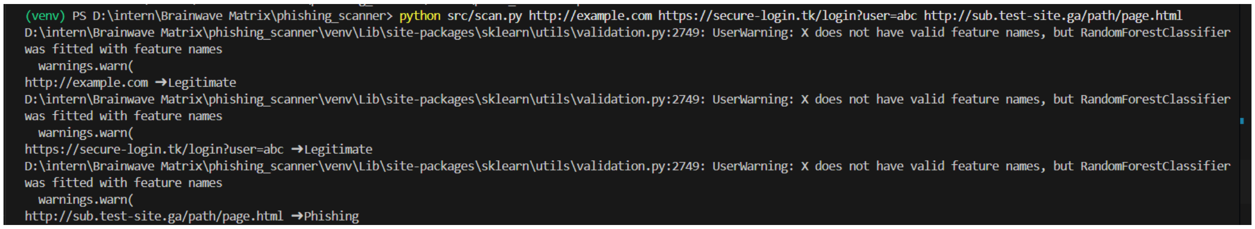
- **WHOIS lookups** are disabled; I can implement caching or use a paid API for domain age features.

- **Data size** is small; I plan to train on tens of thousands of URLs for more robust performance.

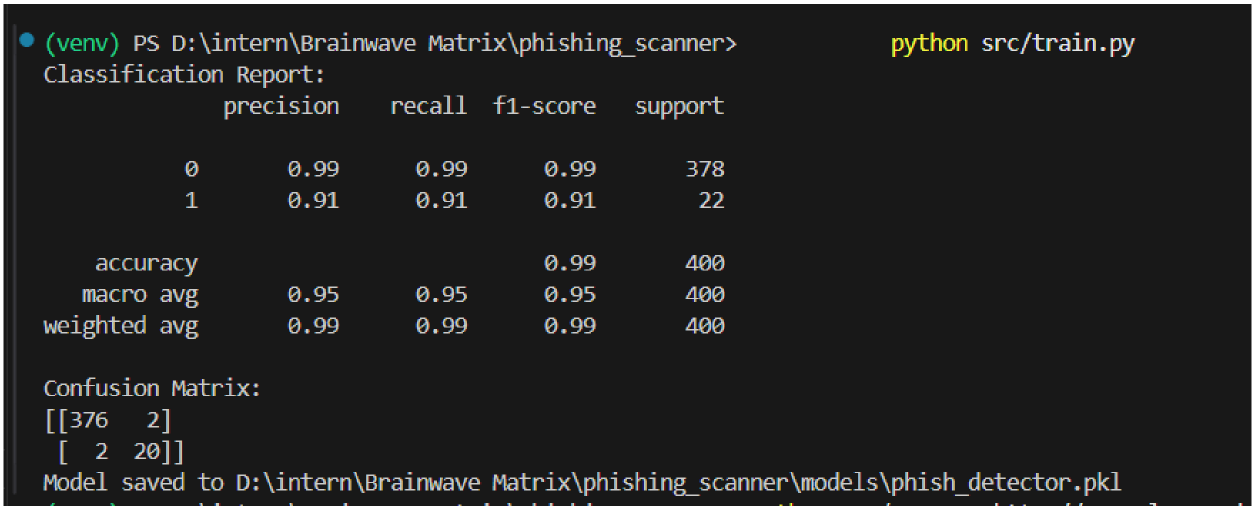
- **Additional features** like SSL certificate checks, content analysis (forms, iframes), and integration with real-time threat feeds could improve detection.

8. Conclusion

I successfully built and evaluated a Python-based phishing link scanner with strong accuracy on a sampled dataset. Future enhancements will focus on expanding the dataset, restoring WHOIS features, and enriching feature extraction for higher reliability.



```
(venv) PS D:\intern\Brainwave Matrix\phishing_scanner> python src/scan.py http://example.com https://secure-login.tk/login?user=abc http://sub.test-site.ga/path/page.html
D:\intern\Brainwave Matrix\phishing_scanner\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but RandomForestClassifier
was fitted with feature names
  warnings.warn(
http://example.com ➜Legitimate
D:\intern\Brainwave Matrix\phishing_scanner\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but RandomForestClassifier
was fitted with feature names
  warnings.warn(
https://secure-login.tk/login?user=abc ➜Legitimate
D:\intern\Brainwave Matrix\phishing_scanner\venv\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but RandomForestClassifier
was fitted with feature names
  warnings.warn(
http://sub.test-site.ga/path/page.html ➜Phishing
```



```
(venv) PS D:\intern\Brainwave Matrix\phishing_scanner>            python src/train.py
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       378
           1       0.91      0.91      0.91        22

    accuracy                           0.99       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.99      0.99      0.99       400

Confusion Matrix:
[[376   2]
 [  2  20]]
Model saved to D:\intern\Brainwave Matrix\phishing_scanner\models\phish_detector.pkl
```