Analysis of Certificate Transparency Logs

Project Documentation

Authors:

Oliwia Witkowska
120867

Jakub Sieński
119269

7 June 2025

Contents

1 Introduction

This document provides a technical overview of the real-time phishing detection system that monitors Certificate Transparency logs. The project analyzes newly issued TLS certificates through a local CertStream server, identifying potential phishing threats using multi-layered heuristics.

Justification for Using a Local CertStream Server

The public CertStream endpoint provided by Calidog at `wss://certstream.calidog.io` has been non-functional since March 26, 2025[2]. Although it has not been officially deprecated, the service appears to be abandoned — we have noticed no response from `wss://certstream.calidog.io`.

To ensure reproducibility, reliability, and continued access to Certificate Transparency (CT) data, we have integrated `certstream-server-go` [1], an actively maintained alternative. This implementation faithfully replicates the original CertStream protocol over WebSocket and streams live certificates from trusted CT logs (e.g., Google Argon2023, Nimbus, and others).

Contributions

Jakub Sieński

- Implemented the DNS permutation module (`dns_twister.py`)
- Wrote the statistical analysis script (`stats.py`) and generated visualizations
- Responsible for creating project documentation
- Will present the project during the final presentation

Oliwia Witkowska

- Implemented core logic in `listener.py`, `phishing_detect.py` and `who_is.py`
- Integrated WHOIS lookups and scoring heuristics
- Designed and implemented entropy, keyword, TLD, and issuer-based detection logic
- Organized GitHub repository structure
- Maintained the `README.md`

2 System Architecture

2.1 Overview

The system consists of several coordinated components:

- **CertStream Listener**: Connects to a local CertStream WebSocket server and spawns 10 concurrent workers to handle CT log events in real-time.
- **Phishing Detection Module**: Applies a multi-heuristic scoring system (max 12.5 points) based on entropy, WHOIS data, keyword presence, issuer behavior, TLD reputation, certificate metadata, and brand similarity. It uses DNS permutation analysis (up to 30 variants per domain) and caches WHOIS queries for efficiency.
- **Statistical Analysis Module**: Provides detailed visualizations and summary statistics of observed domains. Handles records deduplication during statistical analysis. Generates histograms, scatter plots, and heatmaps to support threat intelligence analysis.
- **Utility Scripts**: Include `dns_twister.py` for generating domain permutations and `who_is.py` for enriched WHOIS feature extraction.
- **Output Storage**: All suspicious domains are saved with features to `output/suspected_phishing.csv`.

2.2 Directory Structure

```
CT-Logs/
|-- analysis/
    |-- phishing_detect.py
    |-- stats.py
|-- certstream/
    |-- listener.py
|-- data/
    |-- websites.txt
|-- output/
    |-- suspected_phishing.csv
    |-- plots/
        |-- domain_length.png
        |-- registration_age_log.png
        |-- score_distribution.png
        |-- score_vs_age.png
        |-- score_vs_brand_match.png
        |-- score_vs_entropy.png
        |-- score_vs_issuer.png
        |-- score_vs_keyword.png
        |-- tld_vs_issuer.png
        |-- top_tlds.png
|-- utils/
    |-- dns_twister.py
    |-- who_is.py
|-- requirements.txt
|-- README.md
|-- Report.pdf
```

3 Installation and Usage

1. Clone the Repository

```
git clone https://github.com/olivblvck/CT-Logs.git
cd CT-Logs
```

2. Start CertStream Locally via Docker

```
docker pull 0rickyy0/certstream-server-go
docker run -d -p 8080:8080 0rickyy0/certstream-server-go
```

This starts a local WebSocket server at `ws://127.0.0.1:8080` compatible with the CertStream protocol.

3. Set Up Python Environment

```
python3 -m venv .venv
source .venv/bin/activate  # On Windows: .venv\Scripts\activate
pip install -r requirements.txt
```

4. Start Monitoring CT Logs

```
python certstream/listener.py
```

Suspicious domains will be saved to:

- `output/suspected_phishing.csv`

5. Generate Statistics and Visualizations

```
python analysis/stats.py
```

This creates visual summaries in:

- `output/plots/`

4 Data Collection and Processing

4.1 CertStream Listener

The listener module (`certstream/listener.py`) connects to a locally hosted WebSocket endpoint compatible with the CertStream protocol (`ws://127.0.0.1:8080`). Upon receiving a certificate update, the message is parsed from JSON and validated as a `certificate_update` type.

The list of domains is extracted from the `leaf_cert["all_domains"]` field — a union of the Common Name (CN) and Subject Alternative Name (SAN) entries. Each domain in this list is passed to the phishing analysis pipeline along with associated certificate metadata (e.g., timestamp, issuer, validity).

4.2 False Positive Filtering and Domain Whitelisting

A large portion of observed certificates belong to infrastructure services such as Amazon S3 and other AWS endpoints, which often trigger phishing heuristics due to high entropy or keyword matches. To reduce false positives, the system incorporates several filtering layers:

- `FALSE_POSITIVE_PATTERNS`: A set of regular expressions targeting known infrastructure naming schemes.
- `AWS_DOMAINS`: A curated list of domain suffixes officially documented by Amazon Web Services [3].
- `websites.txt`: A dataset of the top 1000 globally popular domains, used to suppress known good domains and to enable brand similarity scoring. [4]

Examples of domains excluded through these filters include:

- `s3-website.us-east-2.amazonaws.com`
- `s3-accesspoint.dualstack.us-gov-east-1.amazonaws.com`
- `s3-control.dualstack.eu-west-3.amazonaws.com`

4.3 WHOIS and DNS Permutation Handling

Suspicious domains identified in the initial scan are further analyzed:

- **DNS Permutations:** Using `utils/dns_twister.py`, up to 30 typosquatted variants are generated for each domain, based on known brand names.
- **WHOIS Age:** Through `utils/who_is.py`, domain registration dates are retrieved and transformed into a numerical feature: number of days since registration. Missing or malformed WHOIS responses are marked with `-1` and excluded from age-based scoring.

Both permutation results and WHOIS queries are cached using in-memory strategies (`TTLCache` and `lru_cache`) to improve performance and reduce external request load.

4.4 Suspicious Top-Level Domains (TLDs)

In the context of phishing detection, certain top-level domains (TLDs) have been historically and statistically associated with malicious behavior due to their affordability, lax registration policies, or popularity among threat actors for abuse at scale.

To enhance our detection heuristics, we maintain a curated set of suspicious TLDs in the variable `TLD_SUSPICIOUS`. This list includes:

- **Free or extremely low-cost TLDs** (e.g. `tk`, `ml`, `cf`, `gq`, `icu`, `cyou`) — frequently exploited due to minimal cost barriers.
- **Generic and disposable website TLDs** (e.g. `space`, `site`, `host`, `cloud`, `webcam`) — often used to create short-lived phishing landing pages.
- **Marketing-style or emotionally appealing TLDs** (e.g. `buzz`, `shop`, `top`, `fun`, `today`, `mom`) — used to build deceptive branding and impersonation.
- **Geopolitical TLDs with weak oversight or high abuse rates** (e.g. `cd`, `la`, `cc`, `tv`, `to`) — historically found in campaigns targeting global victims.
- **Miscellaneous TLDs flagged in malware feeds or phishing intelligence** (e.g. `uno`, `run`, `faith`, `science`, `xin`).

These TLDs are not inherently malicious. However, they are assigned a non-zero score during phishing risk assessment due to their elevated risk. This feature contributes to the overall phishing score as a lightweight signal in our rule-based system.

5 Feature Extraction and Scoring

Each observed domain is processed through a multi-feature heuristic engine. The following features are extracted:

- Levenshtein similarity to known brand names
- Domain entropy (Shannon entropy)
- WHOIS-based domain age (in days)
- Suspicious TLD presence (from `TLD_SUSPICIOUS`)
- Presence of phishing-related keywords
- Certificate issuer identity and behavior
- Common Name (CN) mismatch in certificate
- Missing OCSP/CRL endpoints
- Short-lived certificate duration ($\leq$ 30 days)
- Brand detected in subdomain component

Each domain is assigned a phishing score in the range **0 to 10**, based on heuristic rules. Final scores are capped at a maximum of 10 points. Higher scores indicate higher likelihood of phishing intent.

Risk Labeling

Domains are categorized into three risk levels:

- `high` — score ≥ 4
- `medium` — score ≥ 2 and < 4
- `low` — score < 2

Rationale for Thresholds: The $\geq$4 threshold for `high`-risk domains reflects the conservative nature of the scoring algorithm. Maximum observed scores are $\leq$7 in practice, as simultaneous critical features—like fresh registration, high entropy, and brand subdomains—are statistically rare.

Phishing Score Calculation Rules

Feature	Condition	Points
Entropy	$\geq 2.8 \rightarrow +0.5$, $\geq 3.2 \rightarrow +1$, $\geq 3.6 \rightarrow +1.5$	+0.5–1.5
Suspicious Keyword	If domain contains terms like `login`, `auth`, `verify`, `account`	+1
Suspicious TLD	Matches entry in hardcoded `TLD_SUSPICIOUS` set	+1
Issuer Risk	Issuer is Let's Encrypt, ZeroSSL, or Actalis **AND** (young age <30 days **OR** suspicious TLD **OR** keyword present)	+1
CN Mismatch	Certificate Common Name not found in SAN (all_domains)	+1
OCSP Missing	OCSP and CRL endpoints missing from certificate	+1
Short-Lived Cert	Certificate validity period $\leq$ 30 days	+1
Brand in Subdomain	Known brand detected in subdomain part (e.g., `paypal.host.com`)	+1
WHOIS Age	0–30 days $\rightarrow$ +3, 31-89 days $\rightarrow$ +2, 30-359 days $\rightarrow$ +1	+1–3
Brand Similarity	Levenshtein ratio ≥ 0.80: 0.80–0.84 $\rightarrow$ +1, 0.85–0.89 $\rightarrow$ +1.5, $\geq 0.90 \rightarrow$ +2.0	+0–2

6 Source Code Overview

6.1 listener.py

This module manages WebSocket communication with the local CertStream server and orchestrates domain analysis using multiple asynchronous workers. Key functions:

- `certstream_client()`: Establishes WebSocket connection and dispatches messages.
- `process_message(message)`: Parses incoming certificate updates and extracts domains.
- `process_domain(...)`: Applies analysis pipeline to each domain.
- `process_worker()`, `csv_writer_worker()`: Asynchronous workers for analysis and logging.
- `main()`: Entry point that launches async tasks.

6.2 phishing_detect.py

Contains core phishing detection logic and heuristic functions. Key components:

- `extract_features(domain, issuer, days, similarity_score, cert)`: Main feature extraction and scoring function.
- `phishing_score(...)`: Aggregates feature signals into final risk score.
- `is_similar(domain)`: Uses RapidFuzz to compute similarity with known brands.
- `has_brand_in_subdomain(domain)`, `contains_suspicious_word(domain)`: Specialized heuristics.
- `calculate_entropy(domain)`: Computes Shannon entropy.
- `is_known_false_positive(domain)`: Filters out known legitimate infrastructure domains.

6.3 stats.py

Handles post-processing of detection results from CSV file and generates visualizations. Implements:

- Deduplication of domains (identical features).
- Risk labeling: assigns `low`, `medium`, `high` based on phishing score.
- Histograms: score distribution, domain lengths, registration ages.
- Scatter plots: score vs entropy, score vs WHOIS age.
- Heatmaps: issuer vs TLD combinations.
- Boxplots: score by issuer, brand, keyword presence.
- Summary tables and top spoofed domain-brand combinations.

7 Visualizations and Output

7.1 Sample Output Fields

The following fields are saved in `output/suspected_phishing.csv` for each suspicious domain observed in real-time:

Field	Description
`timestamp`	Time when the certificate was observed in the CertStream feed
`domain`	Fully Qualified Domain Name (FQDN) extracted from the certificate's all_domains field
`brand_match`	Closest matched brand name from the curated reference list (`websites.txt`)
`similarity_score`	Fuzzy string similarity (Levenshtein ratio) between the observed domain and the closest known brand
`issuer`	Certificate Authority (CA) that issued the certificate (e.g., Let's Encrypt)
`tld`	Top-Level Domain extracted from the observed domain
`tld_suspicious`	Boolean flag indicating if the TLD belongs to the known suspicious TLD list
`has_keyword`	True if the domain contains phishing-related keywords (e.g., `login`, `auth`, `secure`)
`entropy`	Shannon entropy of the domain name — high values may indicate DGA usage
`registration_days`	Age of the domain in days (obtained via WHOIS); -1 if unavailable
`cn_mismatch`	True if the Common Name (CN) does not match any entry in the SAN (Subject Alternative Name) field
`ocsp_missing`	True if OCSP or CRL distribution points are missing from the certificate
`short_lived`	True if the certificate validity period is ≤ 30 days
`brand_in_subdomain`	True if a known brand appears in the subdomain (e.g., `paypal.example.com`)
`score`	Final phishing score (range: **0**–**12.5**) calculated via heuristic rules

7.2 Sample Plots

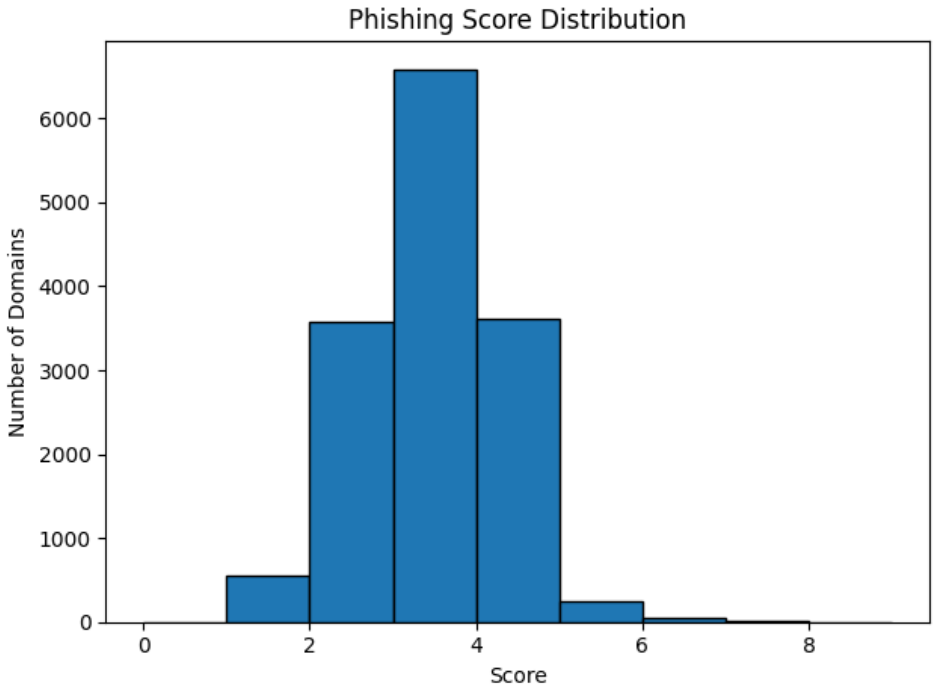

Figure 1: Distribution of phishing scores across detected domains (higher scores indicate higher likelihood of phishing)

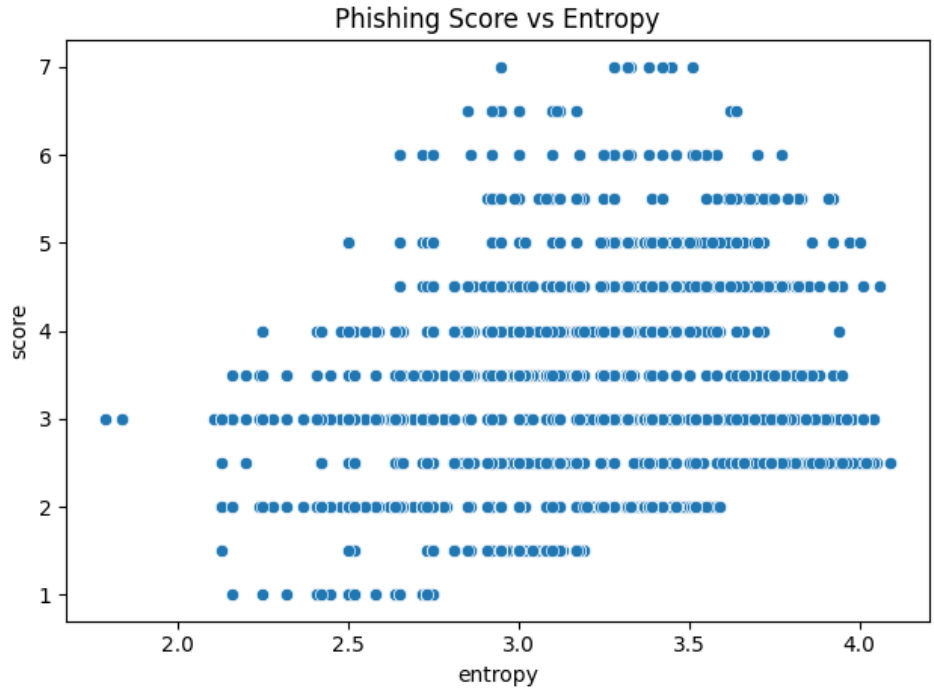

Figure 2: Relationship between domain name entropy (randomness) and phishing score

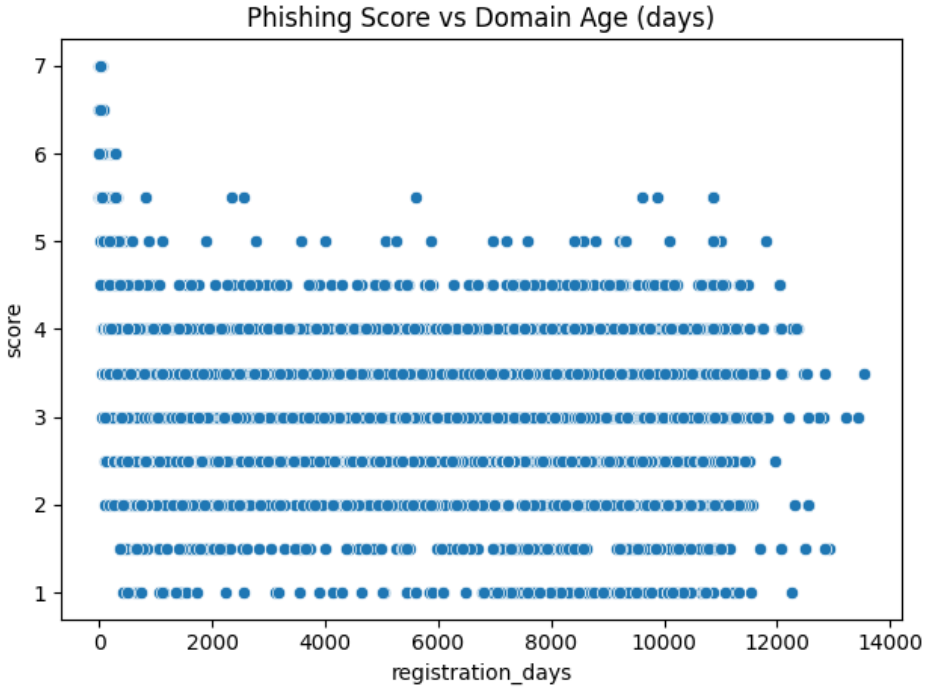

Figure 3: Phishing score distribution relative to domain registration age in days

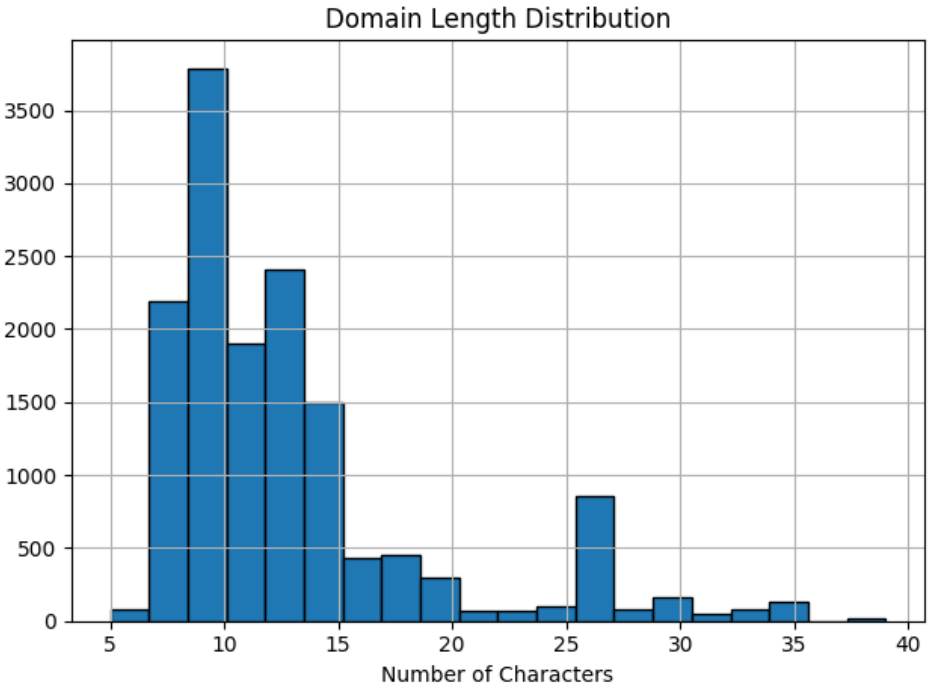

Figure 4: Distribution of character lengths in suspected phishing domains

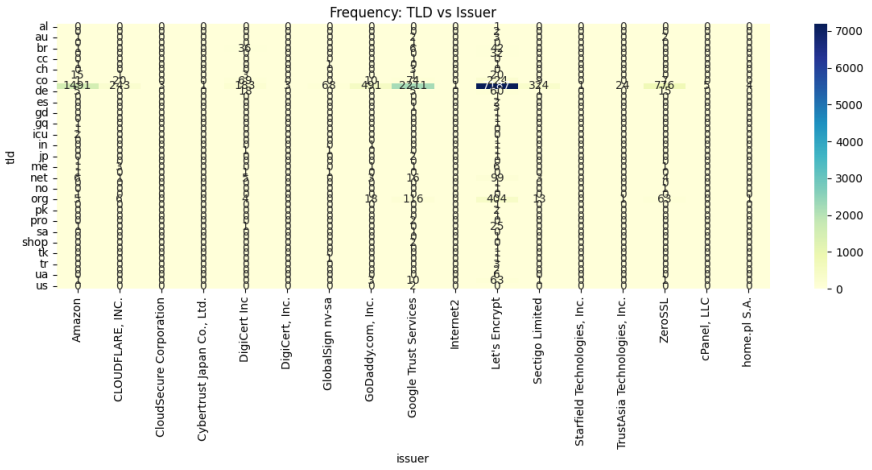

Figure 5: Heatmap showing frequency of top-level domain and certificate issuer combinations

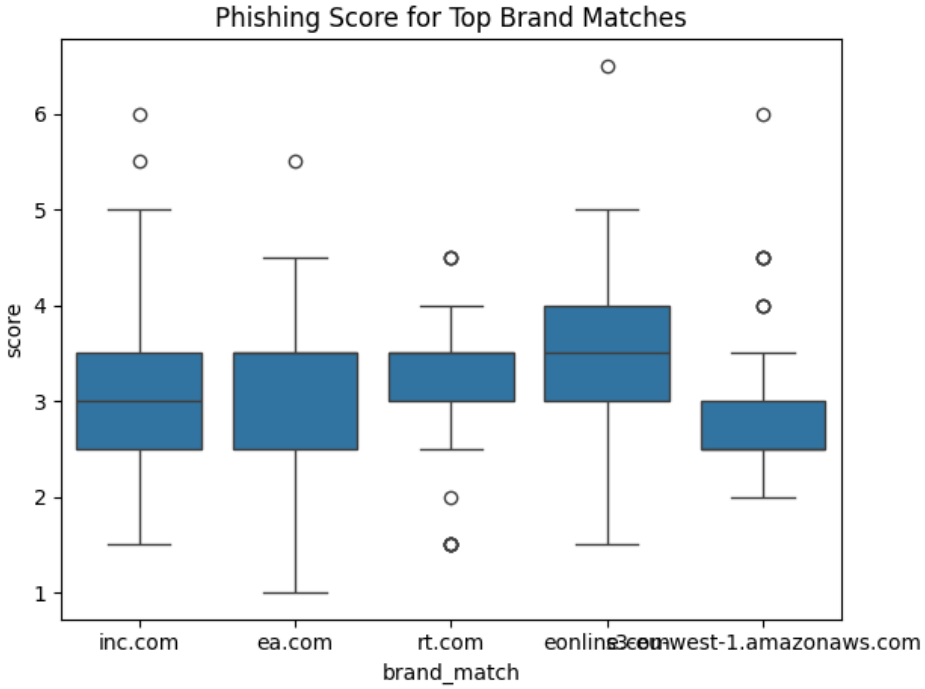

Figure 6: Boxplot showing phishing score distribution for domains targeting top 5 most spoofed brands

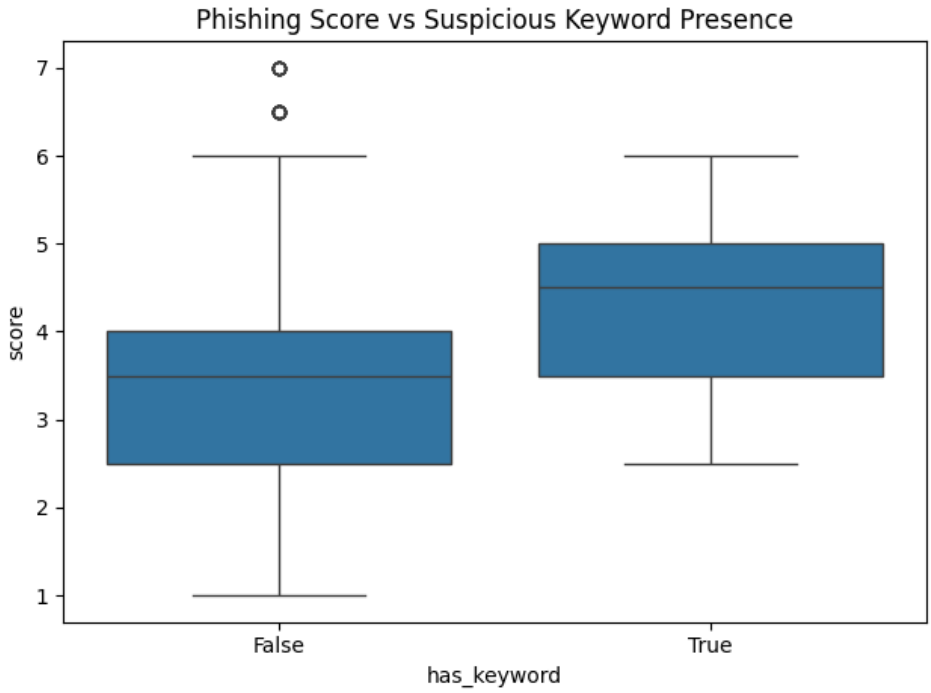

Figure 7: Comparison of phishing scores between domains with/without suspicious keywords

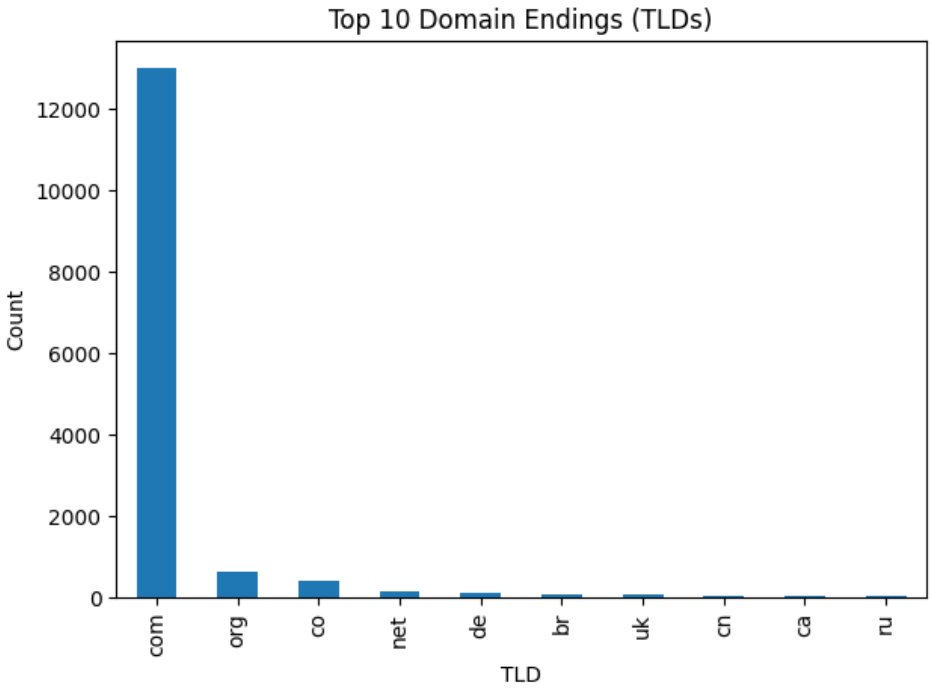

Figure 8: Top 10 most frequently observed top-level domains in suspected phishing URLs

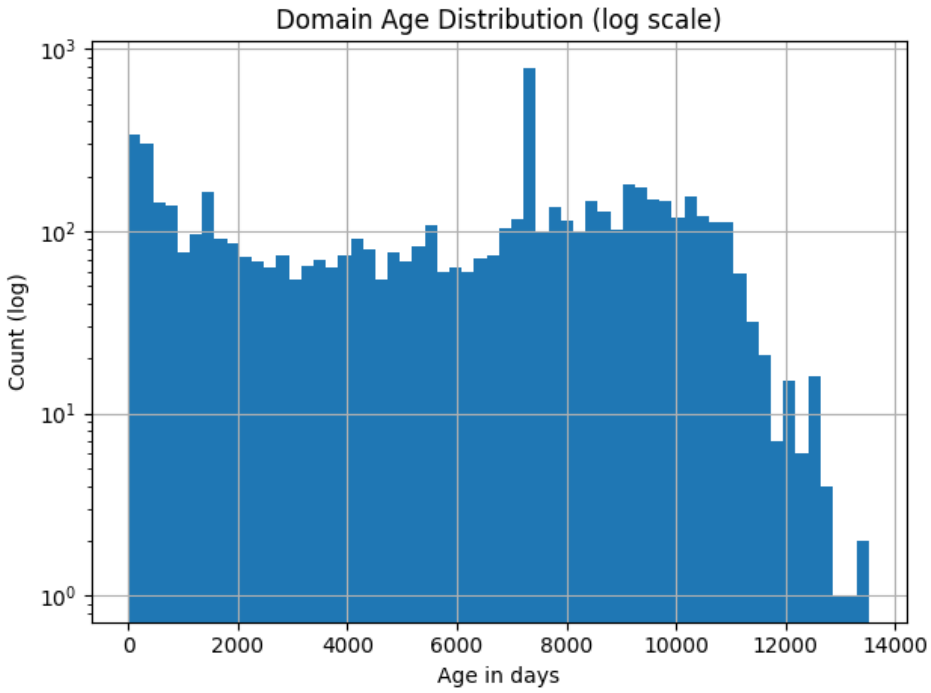

Figure 9: Distribution of domain registration ages (log scale) showing concentration of newly-registered domains

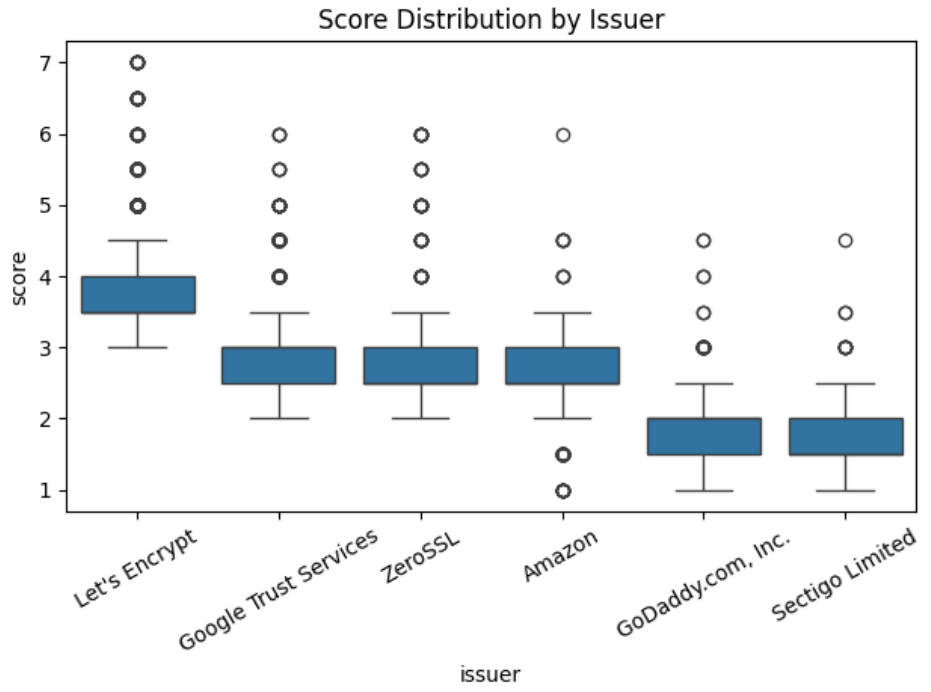

Figure 10: Phishing score distribution by top 6 most common SSL certificate issuers

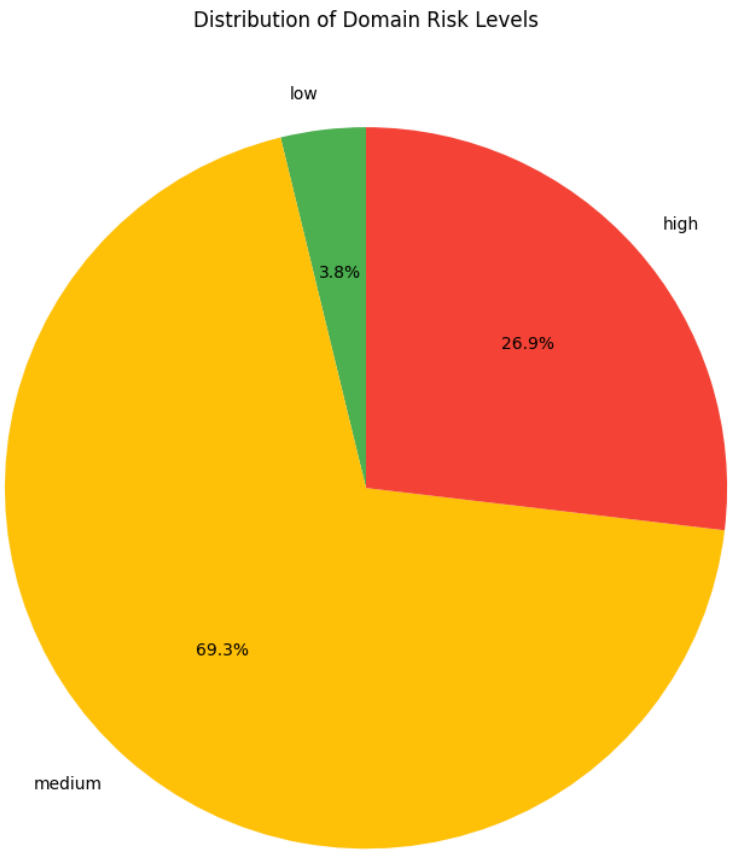

Figure 11: Proportion of domains classified by risk level (low/medium/high) based on phishing score thresholds

8 Future Work

- Add machine learning-based phishing classifier
- Build web dashboard for real-time alerts
- Support for other log sources beyond CertStream
- Crosscheck with Google Safe Browsing, Virus Total and other blacklists if the domains have been detected as malicious.
- `TLD_SUSPICIOUS` list can be extended dynamically as new abuse patterns emerge, or linked to live abuse intelligence feeds (e.g. Abuse.ch, PhishTank).

References

[1] D-Rickyy-b. *certstream-server-go - A CertStream-compatible server*. GitHub repository. https://github.com/d-Rickyy-b/certstream-server-go

[2] x0rz. *Phishing Catcher - CertStream public endpoint not responding*. GitHub Issue #86. https://github.com/x0rz/phishing_catcher/issues/86

[3] Amazon Web Services. *Amazon S3 endpoints and domain structure*. Official AWS Documentation. https://docs.aws.amazon.com/general/latest/gr/s3.html

[4] Bezhan Mukhidinov. *Top 1000 Domains* https://gist.github.com/bejaneps/ba8d8eed85b0c289a05c750b3d825f61#file-websites-csv

[5] Richard Penman. *python-whois*. PyPI project. https://pypi.org/project/python-whois/

[6] Max Bachmann. *RapidFuzz: Fast string matching in Python and C++*. Documentation. https://maxbachmann.github.io/RapidFuzz/

[7] *dnspython - DNS toolkit for Python*. Documentation. https://dnspython.readthedocs.io/

[8] Thomas Kemmer. *Cachetools: Extensible memoizing collections and decorators*. GitHub repository. https://github.com/tkem/cachetools

[9] Steve Ragan. *What are phishing kits? Web components of phishing attacks explained.* https://www.csoonline.com/article/565879/csos-guide-to-phishing-and-phishing-kits.html

[10] *Certificate Transparency*. https://certificate.transparency.dev

[11] *DNS Twister* https://dnstwister.report

[12] Marcin Ulikowski. *DNS Twist*. https://github.com/elceef/dnstwist