CMP 49414-01 – Human Centered Security and Privacy

Project Proposal

FishNet: Phishing Detection using Extensions

Dr. Reham Aburas

October 8th, 2025

Ahmed Mehaisi	b00094989
Mohammed Imtiaz	b00097219
Mohamed Alawadhi	b00094286

Introduction

Web phishing continues to be one of the most common and damaging methods used by attackers to steal credentials and sensitive information, posing serious risks to user safety, trust, and finances. Despite years of progress in browser security, many users still overlook or misunderstand phishing warnings, while attackers constantly develop new techniques that bypass traditional defenses.

FishNet aims to address this problem through a lightweight browser extension that combines heuristic checks, curated blacklists, domain registry and WHOIS signals, and cloud-based machine learning models to detect phishing pages in real time.

To evaluate FishNet, we will conduct an online survey and semi-structured interviews to better understand users' awareness, interpretation, and trust in phishing warnings. On the technical side, we will also measure system performance using traditional model metrics such as accuracy, precision, recall, and F1-score to validate detection quality.

By the end of this project, we aim to deliver
(i) A functional prototype extension,
(ii) Understand how people perceive and respond to security warnings,
(iii) Deeper insights into decision-making patterns around online safety

Motivation

Phishing remains one of the most widespread and damaging forms of cybercrime. According to the Anti-Phishing Working Group (APWG), more than five million unique phishing websites were detected in 2024, representing a steady rise compared to previous years [8]. A Google and Stanford University study also found that about 42 percent of users fail to correctly identify at least one phishing attempt during real-world testing [9]. In addition, the FBI Internet Crime Report reported over 3.2 billion US dollars in losses related to phishing and social engineering attacks in 2023 [10].

These figures show that phishing is not only a technical problem but also a human-centered security issue. Attackers take advantage of human attention limits, visual trust cues, and users' assumptions about familiar websites. For example, some phishing attempts replace characters in legitimate domain names with visually similar Cyrillic or Unicode characters (for instance, using "apple.com" with a Cyrillic "a" instead of the Latin "a") to deceive users into believing a page is authentic [11]. Even when browsers display security warnings, many people ignore or misunderstand them, increasing their exposure to online risks.

Usable security design can help address this gap by focusing on clarity, consistency, and explainability in browser warnings. Common failure cases include deceptive internationalized

domain names, fake login forms that imitate known brands, and spoofed single sign-on (SSO) prompts. FishNet focuses on detecting these suspicious pages accurately and guiding users toward safer actions through improved warning comprehension and trust-building.

Threat Model:

Assets

- User credentials and personal information entered on web pages.
- Browser integrity and session state.
- FishNet's machine learning models, curated blacklist data, and API keys.
- Telemetry logs and optional user reports that may include metadata.
- Domain and WHOIS-based feature data used by the detection system.

Adversaries

- Phishers who create and host deceptive web pages or fake login portals.
- Cybercriminal groups that distribute large-scale phishing campaigns.
- Attackers attempting to poison training data or evade model detection.
- Malicious intermediaries who intercept or alter network communication.

Motivations

- Financial gain through credential theft and account compromise.
- Identity theft or resale of personal data on underground markets.
- Undermining user trust in browser security mechanisms.
- Testing or bypassing detection models for future exploitation.

System Vulnerabilities

- Weak URL inspection if new obfuscation patterns are not captured.
- IDN and Unicode homograph domain impersonation [11].
- Outdated blacklists or stale local caches missing new campaigns [1][2].
- Incomplete or inaccurate WHOIS and domain registry data used for verification.
- Overly broad extension permissions that expose attack surfaces.
- Human error, where users ignore or misinterpret warnings

Possible Threats

- Credential theft through realistic phishing pages that appear legitimate.
- Data leakage from reports, telemetry, or external API communication.

- Adversarial examples or obfuscated URLs that bypass model detection.

- Exploitation of delays in model updates or cloud inference.
- Manipulation of domain or WHOIS data to mask malicious activity.

Expected Risks

- Compromise of user accounts or sensitive data.
- Erosion of user trust in browser-based security warnings.
- Reduced detection accuracy leading to false negatives.
- Reputational damage or privacy concerns if telemetry is mishandled.

Possible Defenses

- Defense in depth: local heuristics first, then cloud ML for ambiguous pages.
- Domain intelligence: integrate domain age, registrar, and WHOIS record completeness into model features.
- Regular updates: frequent model retraining and blacklist synchronization [3][4].
- Homograph detection: normalization and punycode verification [11].
- Privacy by design: minimal data sharing, encrypted communication, and user consent for optional uploads [7].
- System security: least-privilege permissions, strict CSP, and signed updates [3].

Related Work

1. Blacklist and feed based detection

   Deployed protections rely heavily on curated feeds to stop known malicious URLs quickly. Google Safe Browsing provides client lookups, while PhishTank offers community submitted reports that are widely used in research and validation. Commercial and research feeds such as OpenPhish add fresher indicators that help reduce time to detection [1], [2], [12].
   Limitations: These feeds depend on users or organizations reporting attacks, so they often react after phishing pages have already spread. This means new or short-lived attacks may not be detected in time.

2. Heuristics and feature based detection

   Classical systems learn from lightweight URL and page features, for example lexical cues, suspicious token patterns, form and password field counts, title or favicon mismatch, and link anchor inconsistencies. Early machine learning work showed that feature based classifiers can outperform static blacklists on fresh attacks and are inexpensive to run in a browser context [13], [15].
   Limitations: These manually designed features can become outdated as attackers change

their methods, and the reasons behind each detection may not be easily understandable to users.

3. Deep learning for phishing detection

   Newer approaches learn directly from raw URLs or HTML. Models such as character level CNNs and hybrid CNN-RNNs capture longer range patterns without manual feature design and often improve generalization to obfuscated URLs [15], [16].
   Limitations: Larger models increase latency and resource usage in extensions, and server hosted inference must address privacy and cost.

4. Domain intelligence and registration signals

   Domain age, registrar reputation, and WHOIS record completeness are useful features in both academic datasets and industry systems. When used carefully, these signals help distinguish short lived phishing domains from long lived legitimate sites, and they complement URL and content features [13], [14].
   Limitations: WHOIS data can be missing, redacted, or inaccurate, so signals need to be treated as probabilistic and combined with other evidence.

5. Usability research on phishing warnings

   Prior user studies show that warning design affects attention, comprehension, and behavior. Concise text, clear actions, and brief reasons are linked to fewer risky choices and lower false accepts. The human-in-the-loop literature emphasizes clarity, learnability, and avoiding fatigue in repeated exposures [16], [17].

   Limitations: Many detection papers stop at offline accuracy and do not test which explanations users actually trust or follow.

6. Extension security best practices

   Open Worldwide Application Security Project (OWASP) guidance for browser extensions recommends least-privilege permissions, a strict Content Security Policy, no inline scripts or remote code, careful handling of any data sent off device, and regular updates and reviews. In short, request only what you need, lock down what can execute, encrypt what you send, and do not ship secrets to the client [3].

Gaps that motivate FishNet

- Fresh attack coverage. Feeds alone miss zero-day and fast flux campaigns. FishNet combines local heuristics with cloud ML so the system adapts quickly without heavy clients.
- Explainability. Users need clear reasons, not just a score. FishNet would explain the rationale behind each decision, mapped to simple language from prior usable-warning work.
- Privacy and minimal data. Many cloud approaches send full content. FishNet sends only compact feature vectors by default, with optional consented uploads for research or debugging.
- Domain intelligence in practice. Public datasets show domain age and WHOIS features are useful, but many systems underuse them due to noise. FishNet integrates domain registry and WHOIS as probabilistic features with safeguards against missing or redacted records.
- Human centered evaluation. A lot of prior art reports accuracy only. FishNet adds a survey and semi-structured interviews to measure clarity, trust, and willingness to comply, alongside technical metrics (accuracy, precision, recall, F1).

Proposed Solution

FishNet will be implemented as a browser extension following a clear end-to-end pipeline:

1. Detection Layer: Extract URL, DOM, and page context features using lightweight heuristics. Perform initial checks through cached blacklists and simple rule-based analysis (e.g., form count, IDN/punycode checks).
2. Cloud ML Inference Layer: When results are uncertain, securely send compact feature vectors to a cloud-based ML model hosted via an authenticated API (e.g., Google Cloud Run). The model predicts phishing probability in real time.
3. Warning and Decision Layer: Display a concise warning popup showing the top reasons behind the detection result. Users can choose *Leave Site*, *Proceed*, or *Report*.
4. Reporting and Feedback Layer: With consent, users can report phishing pages to PhishTank and FishNet's internal system. Reports improve blacklists and future model training.
5. Security and Privacy Controls: Follow OWASP extension guidelines [3]: strict Content Security Policy, least privilege permissions, and encrypted communication. Cloud uploads are minimal, anonymized, and consent-based.

Tools

- Languages and Frameworks: JavaScript (Manifest V3), TypeScript, and HTML/CSS for the extension frontend. Other frontend frameworks like Astro, React, Bootstrap etc
- Backend and Cloud: Flask (Python) or Node.js API for model inference; deployed on Google Cloud Run or equivalent managed service [6].
- Machine Learning Libraries: TensorFlow/Pytorch, scikit-learn, and Pandas for model development and evaluation.
- Security and Testing: OWASP ZAP for vulnerability scans and Chrome Developer Tools for extension testing.

Datasets

For model training and offline validation, we will use:

- The UCI Phishing Websites dataset [13], which provides 30 website-based features for supervised learning.
- The UCI PhiUSIIL Phishing URL dataset (2024) [14], which contains updated lexical and host-based URL features.
- A benign subset curated from Alexa and top-ranked legitimate websites to balance the dataset.

Security and Privacy by Design

- Follow OWASP extension security guidance, including least-privilege permissions, strict Content Security Policy (CSP), and no inline or remote code execution [3].
- Apply data minimization by sending only compact, non-identifiable feature vectors. Where limited personal data might appear, apply hashing, anonymization, or tokenization, following data protection principles [7].
- Ensure all communication between the browser extension and the cloud inference API uses secure TLS encryption.
- Protect service keys and API credentials through a server-side proxy and rotate them periodically.
- Enforce HTTPS/TLS on all endpoints and use encrypted storage for any temporary data or cached reports.
- Perform regular dependency audits and threat model reviews to identify and patch security vulnerabilities promptly.

Evaluation Plan

Research Questions:

1. How does the phishing detection system perform in terms of technical accuracy, latency, and reliability?
2. Do FishNet warnings improve user understanding and safe decision-making compared to default browser warnings?
3. How do explanation detail and clarity affect user trust and willingness to comply with phishing warnings?

Technical Evaluation

1. Model and System Metrics
    - True positive rate, false positive rate, precision, recall, and F1-score on a combined corpus from PhishTank, OpenPhish, and benign sites.
    - End-to-end latency for local and cloud inference under realistic network conditions.
    - Resource and cost monitoring for the deployed cloud inference service.

2. Robustness and Security Checks
    - Conduct a threat model and dependency audit following OWASP extension security guidance [3].
    - Perform extensive end to end testing and manual review of reported pages.

Human-Centered Evaluation – Usability Study

Type of Study:
A survey and semi-structured interview–based between-subjects study, comparing the baseline browser warning versus the FishNet warning design.

Method and Procedure:
Participants will first complete a short online survey presenting simulated phishing scenarios and screenshots of warnings. They will rate clarity, trust, and intent to comply.
Next, a 10–15 minute semi-structured interview will explore their reasoning and understanding of the warning design. Responses will help identify usability issues and guide interface improvements.

Independent Variables (IVs):

- Warning type: baseline browser vs. FishNet warning.
  Explanation detail: short vs. detailed reason message.

Dependent Variables (DVs):

- Perceived clarity: how well participants understand the warning.
- Trust in system: confidence in the accuracy of the warning.
- Willingness to comply: likelihood of following the safety recommendation.
  Decision time: average time to make a safe choice.
- Decision accuracy rate: proportion of correct safe responses in simulated tasks.
- Usability score: using the System Usability Scale (SUS).
- Qualitative feedback: themes related to comprehension, trust, and perceived usefulness.

Confounds and Controls:
Possible confounds include prior phishing awareness, browser familiarity, and attention bias. We will collect demographic data, randomize question order, and use neutral wording to reduce bias.

Participants and Sampling:
A sample of 15–20 participants selected from the average population (with convenience sampling) will be recruited. No personal data or sensitive identifiers will be collected.

IRB Consideration:
No formal IRB application will be made for this project. The usability study will involve only the researchers' close friends and family members, and participation will be entirely voluntary. No sensitive personal data or identifying information will be collected, and responses will be used solely for academic evaluation and system improvement.

Timeline

Week	Goals
Week 7	Initialize project files, set up extension structure, and obtain required API keys.
Week 8	Implement core feature extraction and heuristic rules; design the initial warning popup
Week 9	Integrate blacklist lookups and build a small server tool to manage cache updates.
Week 10	Develop and train the cloud-based ML model using phishing and benign datasets.
Week 11	Connect the extension to the cloud inference API; add consent dialog and secure proxy.
Week 12	Test detection accuracy and UI; tune thresholds and complete security checklist.
Week 13	Conduct the usability study; analyze feedback and finalize report and slides.
Week 14	Present the project and submit all deliverables.

References

[1] Google, Safe Browsing API Documentation, 2024. Available: https://developers.google.com/safe-browsing/v4

[2] PhishTank, API Information, 2025. Available: https://phishtank.org/api_info.php

[3] OWASP, Browser Extension Vulnerabilities Cheat Sheet, 2025. Available: https://cheatsheetseries.owasp.org/cheatsheets/Browser_Extension_Vulnerabilities_Cheat_Sheet.html

[4] Chrome Developers, Manifest V3 and Service Worker Guidance, 2024. Available: https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3

[5] Google Cloud, Cloud Run Best Practices for AI Inference, 2025. Available: https://cloud.google.com/run/docs/configuring/services/gpu-best-practices

[6] Information Commissioner's Office (ICO), Guide to Data Minimisation (UK GDPR Principle), 2025. Available: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/data-protection-principles/a-guide-to-the-data-protection-principles/data-minimisation/

[7] Anti-Phishing Working Group (APWG), Phishing Activity Trends Report, 2024. Available: https://apwg.org/trendsreports

[8] E. Bursztein et al., Understanding Phishing Risk, Google and Stanford University Research, 2024.

[9] Federal Bureau of Investigation (FBI), Internet Crime Report 2023, Internet Crime Complaint Center (IC3), 2024.

[10] ICANN, IDN Homograph Attacks and Unicode Security Considerations, 2023.

[11] OpenPhish, Phishing Database and Feeds, 2025.

[12] UCI Machine Learning Repository, Phishing Websites Dataset, 2015.

[13] UCI Machine Learning Repository, PhiUSIIL Phishing URL Dataset, 2024.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs,” Proc. KDD/WWW, 2009.

[15] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor, “Crying Wolf: An Empirical Study of SSL Warning Effectiveness,” USENIX Security, 2009.

[16] L. F. Cranor, “A Framework for Reasoning About the Human in the Loop,” UPSEC, 2008.