PENINSULA PUBLISHING LLC

SHIFRA

Vol. **(2026)**, 2026, **pp**. 122–140
ISSN: 3078-3186

Research Article

Channel-Agnostic Containment of Scam Stores in SDN Networks

Joel Yao Adedi [1, *], Hui Xu [1]

[1] *School of Computer Science and Artificial Intelligence, Hubei University of Technology, Wuhan, China*

ARTICLEINFO

Article History

Received 22 Feb 2026

Revised: 12 Apr 2026
Accepted 11 May 2026
Published 26 May 2026

Keywords

Network Security,

Intrusion Detection,

Mobile Commerce Security,

Software Defined Networking,

Fraudulent e-Commerce Websites,

ABSTRACT

In recent years, fraudulent shopping websites have continued to pose a serious threat despite heightened attention. Existing, case-specific security ecosystems often lack real-time mitigation, overlooking the myriad social channels, the short-lived nature of attacks, and the open windows that expose mobile commerce and PC-centric vulnerabilities. In this paper, we propose an automated method for gathering and detecting Fraudulent E-Commerce Websites (FCWs) by examining URLs from the very internet fabric such attacks rely on being HTTP, HTTPS, HTTP3 protocols in an SDN network. We relied on an in-network web retrieval system and a feature-enhanced XGBoost classifier built on the BeyondPhish model, to identify and categorize FCWs. Our model achieved a detection rate of 97.28% and a false-negative rate of 2.7%. Finally, detected FCWs are blocked via SDN OpenFlow rules applied to their IPv4 and IPv6 addresses, effectively closing the exploitation window that previous approaches left open for an average of 15 days or more when legal processes were required. By leveraging real-time, in-network web retrieval, classification, and mitigation, we achieved a pipeline detection-to-restriction time of under 30 seconds while blocking fraudulent URLs across all channels, demonstrating strong potential for enhancing mobile-commerce and PC-centric security.

1. INTRODUCTION

The web's flexible, resource-sharing ecosystem has fostered vast, borderless communities of like-minded individuals. However, it has also attracted cybercriminals whose sole purpose is to steal money or coax victims into revealing sensitive information. A common example is phishing, which targets these communities to harvest passwords, bank details, Social Security numbers, TIN/EIN, etc., enabling identity theft and KYC/KYB bypasses [1]–[8]. Most attackers aim to financially defraud users.

Several studies document these attacks. Bitaab et al. [9] examined the ecosystem of fraudulent e-commerce websites (FCWs), revealing that they not only seek financial gain but also operate across distinct categories such as fake online shopping (goods never delivered), pet scams (no pets provided), bogus charities, and counterfeit cryptocurrency or stock schemes. Fake online-shopping sites alone account for 60% of attacks. Price [14] corroborates this finding, showing how pet-scam sites accept payments yet never deliver the animals, resulting in substantial monetary losses. FCWs employ distinct tactics, but their sole objective remains financial fraud. Unlike phishing, which harvests sensitive data, FCWs trick users into paying for goods or services that are never delivered. These attacks rarely aim to impersonate victims; instead, they masquerade as legitimate businesses, often using reputable payment processors such as PayPal, Stripe, Venmo, and Google Pay [12] to convey trust.

Despite extensive research on fake online shops [9]– [20], effective detection and mitigation especially for mobile commerce remain unsolved. First, most studies address only a narrow set of attack vectors [21]. Fraudulent sites reach users not just through search results or paid ads; they also appear in browsers (including in-app), email, SMS, in-app messaging, social shares, and major advertising platforms such as Google, Instagram, LinkedIn, TikTok, Snapchat, and others. Extension-based defenses are fragmented on desktop browsers and even more limited on mobile devices, despite smartphones now dominating shopping [24],[25]. Second, research often ignores post-detection processes. Resolution times are long and heavily manual, particularly in cross border investigations [22], while perpetrators can monetize within the first 24 hours [12]. This creates a critical "opportunity window," as massive fraudulent storefronts typically appear and disappear in less than a year [9],[12]. Consequently, current detection and collection efforts are tied to specific contexts

*Corresponding author email: joeladedi@gmail.com
DOI: https://doi.org/10.70470/SHIFRA/2026/007

search query results, platform listings, blacklists, or browser extensions and do not scale. Mitigations rely largely on extension integrations [16],[17], covering only a small fraction of both mobile and desktop-centric shoppers and leaving the majority exposed. As a result, detection and mitigation remain inadequate across the primary social channels used by most shoppers. Therefore, we require a method that not only collects but also restricts FCWs in real time, independent of the attack channel, while investigative actions are underway. To achieve this, we capture URLs at scale within an SDN-based network, leveraging protocols such as HTTP [18], and HTTPS; this provides true channel-agnostic collection. After capture, each URL is assessed and classified in-network using an enhanced XGBoost model trained on the BeyondPhish dataset and a BigQuery-derived dataset. The model reliably labels URLs, enabling immediate restriction of confirmed FCWs by deploying IPv4 and IPv6 drop rules to Open vSwitch instances, thereby enforcing network-wide URL blocking. This architecture eliminates the previously identified "opportunity window" and resolves the channel-specific challenges of collection and mitigation in mobile commerce security. In this paper we specifically address the following questions:

- (RQ1) How can we capture FCW URLs at scale in a channel-agnostic manner from early-layer network signals?
- (RQ2) How can we integrate a classification model to reduce pipeline latency enough to effectively close the exploitation window?
- (RQ3) How can we apply at-scale network-fabric mitigations and restrictions after detection?

To answer these research questions, we first implemented a Scapy-based solution in our SDN-simulated network that captures and extracts URLs from in-network simulated HTTP and HTTPS requests. The system automatically filters out links from major CDNs and applies a whitelist derived from a 3.9+ million-entry Tranco dataset, eliminating unnecessary classifications. Extracted URLs are then forwarded to an in-network XGBoost classifier, which generates relevant features for real-time URL assessment. Since our use case demands low-latency, in-network evaluation and web retrieval that involve active page fetching and HTML parsing, we restrict feature extraction to data obtainable directly from the URL. Building on the well-known BeyondPhish model, we added several manually selected features and achieved a 97.28% detection rate in our evaluation. Although this exceeds the latest 59.30% reported by ScamMagnifier [12], the latter was measured on different data and under different evaluation conditions, so it does not serve as a directly comparable baseline. Consequently, the comparison is provided for context rather than a strict head-to-head benchmark; the key takeaway is that our feature enhancements deliver robust performance in the current setting. In end-to-end testing, the pipeline achieved an average feature-generation latency of 12.5 seconds, a page-fetch latency of 5.1 seconds, and a model inference time of 45.5 milliseconds. Finally, URLs identified as malicious (FCWs) are dropped within the network by the controller sending rules to the OpenVSwitch, this proved an average drop enforcement latency of 2.2 seconds. In summary, the contributions of this paper are as follows:

- We provide at-scale detection and restriction of FCWs by leveraging an SDN network that not only detects threats across all channels but also enforces rules that successfully block further access to malicious URLs for mobile and PC-centric shopping users, regardless of the medium.
- We close the previously overlooked "opportunity window" that FCW attacks exploit. Although prior works have detected threats in specific channels such as search results efficiently [13], the time required to resolve them still causes financial loss for exposed users.
- We exploit attacker behavior observed in CMS (Shopify, WordPress, Wix, etc.) to enhance the detection metrics originally offered by BeyondPhish, adding novel detection features.
- During detection and classification we maintain a continuously updated list of FCW URLs, enabling the network to bypass re-classification of known sites and instantly apply drop-rules. All drop-rules are TTL-based and expire automatically, ensuring restrictions are applied only when the URLs reappear, without keeping the rules permanently active.

In this paper, we first review the background and related work on fraudulent e-commerce website (FCW) detection and mitigation. We then define the system model and design objectives. Section 4 describes the datasets and preprocessing methods. Section 5 details our feature-engineering and classification approach. Section 6 presents the SDN-based detection and enforcement pipeline. Section 7 outlines the experimental setup, and Section 8 reports the results. Section 9 discusses the findings, and Section 10 concludes with future directions and recommendations for further research. To better support reproducibility, the source code and accompanying materials used in this study are available at https://github.com/.

2. BACKGROUND AND RELATED WORK

Fraudulent e-Commerce Websites (FCWs), as defined by Bitaab et al. [9], are sites that fraudsters use to target users and financially defraud them. These sites display seemingly legitimate products or services to earn trust, often employing trust-building tactics such as trust seals because they boost conversion rates. A/B tests confirm this effect. In a 4-week mobile and 2-week desktop test, AmbienteDirect [37] found that a Trusted Shops badge increased mobile orders by 8%, desktop conversion by 6%, and mobile average order value by 9%. In a 7-week test, Shoe Mart [38] compared Norton/GeoTrust

badges with TrustedSite and achieved a 14% overall lift and a 21% lift on mobile with 97% confidence. The primary goal of FCWs is deception through rapid website rotation: they collect payments and then disappear without delivering any goods or services. To lure victims, they exploit a wide array of channels including direct messages, ads on Facebook, LinkedIn, Instagram, TikTok, Snapchat, Google Ads, email, and search-engine results. Because they appear legitimate and rarely impersonate existing businesses, FCWs pose a significant financial risk to online shoppers, targeting the very platforms most users rely on for mobile commerce.

A recent series of case studies highlights the scale and sophistication of fake e-commerce fraud. BogusBazaar (2021-2024) [39],[40] a China-based network ran 75,000+ fraudulent webshop domains (22,500 active in April 2024), processing 1M+ orders worth $50M+ and victimizing about 850,000 shoppers in Western Europe and the U.S. The group used steep discounts, expired-domain SEO, WordPress/WooCommerce storefronts with minimal branding, and interchangeable payment pages that routed transactions through legitimate processors (PayPal, Stripe, credit-card networks). Cloudflare front-ended servers hosted 200-500 shops per IP, often in the U.S.; victims received no goods or counterfeits and faced card-data theft. SilkSpecter (late 2024) [42] leveraged Black-Friday-style "80%-off" offers, social-media ads, and SEO poisoning to drive traffic to typosquatted .top/.shop/.store/.vip domains. Victims entered PII, CHD, SAD, and phone numbers on realistic storefronts that used Stripe for genuine payments while exfiltrating data. The operation spanned 4,000+ domains, 89+ IPs (about 85% Cloudflare-proxied), obscuring origin and hindering takedown. ERIAKOS (April 2024) [41], a 608-domain campaign impersonated major brands, promoted bogus sales via Facebook/Meta ads (sometimes 100+ ads per domain per day), and displayed sites only to mobile users from ad clicks, serving 404s to desktops or direct traffic. Payments flowed through major card networks and Chinese processors. Shared infrastructure included an Eriakos CDN, Alibaba Cloud/HiChina registrar, and recurring IPs. YLabs/Yarix (2022-2024) [43] actors abused Chinese site-builder services (e.g., SHOPOEM) to mass-produce fraudulent websites that hijacked retailer trademarks. Ads on Facebook and Instagram redirected users to "maintenance" pages for browsers but revealed fake storefronts to smartphone agents, with Meta Pixel tracking. CDN-based hosting (mostly Cloudflare-proxied) held hundreds to thousands of domains each; 14000 domains were identified in two months. These studies demonstrate that fraudsters combine large-scale domain generation, SEO and ad-driven traffic, selective rendering, and legitimate payment processors, while using Cloudflare and CDN rotation to hide infrastructure, making manual takedown and detection extremely challenging.

Several studies, under clearly defined motivations and environment specifics, have examined FCWs to develop detection solutions.

One detection strategy, termed "at-scale FCW detection," targets fraudulent e-Commerce Websites (FCWs) both in the wild and at the point of entry. Bitaab et al. [9] highlighted the scarcity of public FCW datasets and observed that existing defenses such as Google Safe Browsing and Microsoft Windows Defender often miss these threats. Their solution gathered FCWs through crowdsourcing on the /r/Scams subreddit, applied BERT-based sentiment analysis to flag suspicious links, and manually defined key features across content, DNS, URL, and social-media dimensions. This feature set powered BeyondPhish, a machine-learning system that evaluated Random Forest, SVM, and feed-forward neural-network models, achieving high-accuracy classification and enabling large-scale detection in partnership with Palo Alto Networks. Kotzias et al. [10] responded to the surge in e-commerce scams by compiling a dataset of 8,944 online shops, merging known scam repositories with reputable review platforms for benign URLs. They built an automated pipeline that crawled each shop and extracted 111 features covering shop characteristics, network data, HTTP headers, content, and URL metadata. Their system was integrated into the Norton security platform to flag or blacklist malicious URLs. Bitaab et al. [12] focused on proactive FCW detection before blacklists or commercial tools could react. Their labeling workflow combined automated DNS signals with manual verification via BeyondPhish, incorporated an automated checkout test to expose fraudulent transactions with merchant IDs, and deployed browser extension that warns users of suspicious domains while reporting scam indicators. Feature extraction leveraged visual, textual, and structural cues, with checkout-page behavior identified as the most decisive detection point. In a similar vein, Paniagua et al. [19] introduced a machine-learning model that identifies online shops by extracting 11 handcrafted features from HTML, WHOIS, SSL, social-media profiles, and Trustpilot scores; seven of these features are novel, derived from SSL certificate names and e-commerce technologies. Finally, the ScamNet system [11] demonstrated the use of large language models for FCW detection by fine-tuning Llama-3-8B-Instruct through a two-step supervised approach. The second step employed a curated set of 200 manually verified explanation samples, enabling the model not only to classify a website as fraudulent or legitimate but also to articulate the reasoning behind its decision.

One other approach focuses on the "engine search results" category of fraudulent e-Commerce website (FCW) detection. In this line of work, researchers examine browser search results to identify and target FCWs. Carpineto and Romano [13] introduced a search-centric, two-stage learning pipeline designed to spot counterfeit e-commerce sites. Their system, RI.SI.CO., first determines whether a search result points to an e-commerce platform and then classifies the site as legitimate or fake. Each stage uses a dedicated set of features UI elements, shopping-cart functionality, trust seals, SEO patterns, and more enabling the creation of counterfeit-risk charts that visualize brand-level exposure. Following the same process, Wadleigh et al. [15] investigated how counterfeit goods are sold through websites appearing in search-engine results. They collected search results for 225 brand-specific queries and extracted relevant features, including WHOIS data,

pricing, HTML content, brand mentions, and keywords. Using these features, they trained a logistic-regression classifier to detect sites selling fake goods. Their study showed that higher prices and the absence of DMCA notices correlated with a greater likelihood of counterfeit sites.

A final approach targets the "similarities" category of FCW detection. Here, researchers analyze the links and shared characteristics among suspect sites to cluster and identify groups of fraudsters. Beltzung et al. [16] developed several machine-learning models such as Random Forest, Naïve Bayes, SVM, ANN, and XGBoost to detect fake e-commerce shops solely based on structural and design similarities. Their premise is that fraudsters often reuse templates and layouts across malicious sites. The models examine features such as HTML, CSS, JavaScript, and code comments. The solution is deployed via a browser plug-in and a middleware API that issue risk-based, real-time warnings to capture users' attention. Another example, presented by Cheung et al. [17], observed that counterfeit sellers frequently repost identical or reused product images from platforms like Taobao, creating a detection opportunity. Their approach employs a convolutional neural network specifically ResNet to extract visual features from product images, cluster these features, and generate seller profiles as label distributions. The system then flags sellers with a high likelihood of selling counterfeits for manual review. However, most studies that focus solely on detection, alerts, and warnings [9]–[12],[19] still rely on browser extensions for notifications, while only a few also address mitigation. [16],[17]. We note fraudsters reach customers through a variety of channels, this allows them to target users not only on desktops but, more importantly, on mobile devices, which now represent the largest share of online shoppers. The iVend Retail "Global Path to Purchase Survey" (2018) [25] reports that 71.6% of consumers shop on smartphones, the PayPal "Mobile Commerce Report 2018" (Censuswide) [24] notes that the smartphone is the preferred device for online shopping, with 51% of shoppers purchasing weekly or monthly on their phones, the Pew Research Center's 2022 study observes that mobile shopping has become a dominant online purchasing channel in the United States with 76% of U.S. adults buying things online using a smartphone, compared with 69% who do so via desktop or laptop, and that usage rises to 91% among adults aged 18 to 49. These facts create three major limitations in current research:

- Current detection pipelines rely on a narrow set of sources such as search-engine results, similarity-based e-commerce sites, storefront directories, static blacklists, and desktop-only browser telemetry. This overlooks many fraud vectors: short-lived URLs in phishing emails and newsletters, deep-links in WhatsApp Business, Facebook Messenger, in-game chat, in-app browsers/web-views, SMS/OTT messages (iMessage, Telegram), and native ad formats. Limiting collection to browser traffic reduces recall and delays discovery of new campaigns. Valuable information from mobile-centric mediums can be omitted.
- Most defenses are browser extensions that warn/alert after a malicious URL is identified. Extensions suffer three flaws: they leave mobile users unprotected (mobile traffic now exceeds desktop), cannot intercept URLs opened outside browsers (PDF viewers, email clients, system handlers), and depend on users noticing alerts attention wanes quickly. Consequently, attackers shift to mobile apps and other channels, rendering extension-only solutions ineffective. Thus raising the question, "Do we achieve real mitigation when browser extensions are seldom used on mobile phones?"
- Scam operators rapidly rotate domains, advertising IDs, IP blocks, CDNs, and storefronts, making signature-based blacklists quickly obsolete. IP-based blocking on its own is unreliable because shared services (Cloudflare, Akamai, CloudFront) host countless legitimate sites, causing high false-positive rates.

One other important drawback remains the operational opportunity window that fraudsters exploit. As [23] notes, "With the explosive growth of online transactions, traditional manual review methods are no longer scalable or effective." Closing the gap between detection and resolution is therefore essential. Lakkaraju [22] reports that merchant-fraud detection takes roughly 3.6 hours, while resolution typically requires 15 days or more, giving fraudsters ample time to act with [12] showing they can monetize the loss within 24 hours. Similarly, [21] documents the challenges faced by fraud investigators in financial institutions, emphasizing the complexity, delay, and inefficiency of current detection and legal-follow-up processes. Their analysis highlights several systemic obstacles: "Court-admissible evidence must be certified by the jurisdiction that gathered it. . . which can take years," "We refer only about 3–5% of cases to law enforcement, and those rarely succeed," "Fraudsters know they won't get caught; the risk is essentially zero," "We lack enough analysts to review every flagged case," and "Tracking scammers across borders is extremely difficult." Together, these points demonstrate how fraudsters exploit weaknesses that remain entrenched in the very systems designed to protect customers.

To address these issues, our solution provides a scalable method for collecting FCWs over HTTP and HTTPS. An in-network XGBoost classifier detects and classifies the URLs, and the SDN applies IPv4 and IPv6 drop rules only when the same FCW reappears. Reviews are performed against a continuously updated blacklist of previously stored URLs. This ensures that any identified FCW, regardless of the delivery channel, can be contained, addressed, and mitigated across all media.

3. SYSTEM MODEL AND DESIGN GOALS

3.1 Network and SDN Model

In order to address the research needs, we designed a multi-layered experiment that combined BeyondPhish-style features with an SDN-based, in-network mitigation framework. The workflow relied on several tightly integrated Python modules,

each fulfilling a specific role in the pipeline.

The controller runs three application scripts: the controller itself, Osken, and the classifier. At the controller level, the script handles packet capture, filtering, classification, and logging. Incoming packets are first checked against the Tranco whitelist, allowing known benign URLs to bypass further analysis while unknown URLs proceed to classification. The controller is designed to perform precise page fetches, ignoring CDN resources and non-content assets. A preloaded list of previously identified fraudulent e-commerce sites (FCWs) enables instant dropping of recurring malicious domains.

Within the pipeline, Redis serves as a temporary in-memory store to manage batch processing, prevent duplication, and ensure each URL is handled only once per session. The controller maintains bidirectional communication with the Osken application, which enforces policies on the Open vSwitch (OVS) layer. Osken translates a "fraudulent" classification into OpenFlow drop rules targeting HTTP, HTTPS, and HTTP/3 traffic (TCP/UDP, IPv4/IPv6). Each rule includes a time-to-live (TTL) value to avoid permanent blocking and to re-apply only when the same malicious URL reappears. Upon startup, Osken installs a table-miss → NORMAL rule to preserve baseline L2 switching and provide immediate traffic visibility for the controller.

A request-generation script was created to run on a dedicated Ubuntu VM at the client side, simulating user-side traffic within the virtual network. It issues randomized HTTP and HTTPS requests drawn from a combined dataset of 3,100 URLs (the union of the active BeyondPhish and BQ Benign sets). Requests follow a Gaussian distribution to mimic realistic browsing patterns, enabling end-to-end measurement of key metrics such as total packets captured, extraction accuracy, average page-fetch latency, and model-inference latency.

In our pipeline, aiming for completeness we extended BeyondPhish's original model for real-time operation by adding CDN filtering, an updated list of low-cost registrars, and CMS feature extraction from HTML content supporting WooCommerce, Shopify, Wix, BigCommerce, and PrestaShop. The enhanced classifier increased the feature count from 527 to 605 while remaining lightweight enough for real-time inference.

3.2 Threat Model

Fraudulent e-commerce websites subtly mimic legitimate businesses and storefronts to lure trusting users into making payments or requesting paid services that are never delivered. Attackers exploit readily available services domain registrars such as GoDaddy, Namecheap, Hostinger; payment processors like PayPal and Stripe; CMS platforms such as Wix, WordPress, Shopify; and social networks including Facebook and Instagram to build a rapid-turnover ecosystem of domains, redirects, and reusable templates. This creates a diverse network that reaches customers while overwhelming existing security controls. Because these sites appear briefly and act quickly, they exploit operational windows that current research often overlooks. As [23] notes, "With the explosive growth of online transactions, traditional manual review methods are no longer scalable or effective." Consequently, fraudsters take advantage of the system's inability to block or mitigate actions across all major attack vectors. Given these assumptions, prevention must occur not only at the browser level but also at the network level. By leveraging the core Internet protocols used by users (HTTP, HTTPS, HTTP/3) we can monitor every channel and because links and URLs are accessible regardless of where they are clicked or advertised from, we can thus leverage SDN-like networks to push restrictive policies for mitigations.

3.3 Design Goals

Due to the inherent nature of FCWs, which exploit opportunity windows in existing mitigation systems, and the limited detection and restriction capabilities of browser-extension applications especially regarding alerts and warnings, we aim to design a preventive detection and mitigation approach rather than relying on post-attack responses that require manual intervention and are unaware of system needs. To achieve this, we target a comprehensive pipeline that delivers high recall, low false-negative rate (FNR), low latency, channel-agnostic network-based detection, scalable SDN enforcement, and, most importantly, suitability for mobile and PC-centric commerce traffic. Because all observations and assessments must occur in-network, we require a Machine Learning (ML) model that is both efficient and lightweight to minimize resource consumption; therefore, we selected the XGBoost model [26],[27] and the Ryu controller [28],[29] for their flexibility, adaptability, and scalability within the network [30].

4. DATASETS AND PRE-PROCESSING

This section outlines the datasets used to train and evaluate our FCW detection pipeline and describes the preprocessing steps applied to URL and domain records. Fig. 1 illustrates how FCW domains from the BeyondPhish corpus are merged with a curated benign e-commerce set, creating subsets that enable real-time testing on live network URLs.

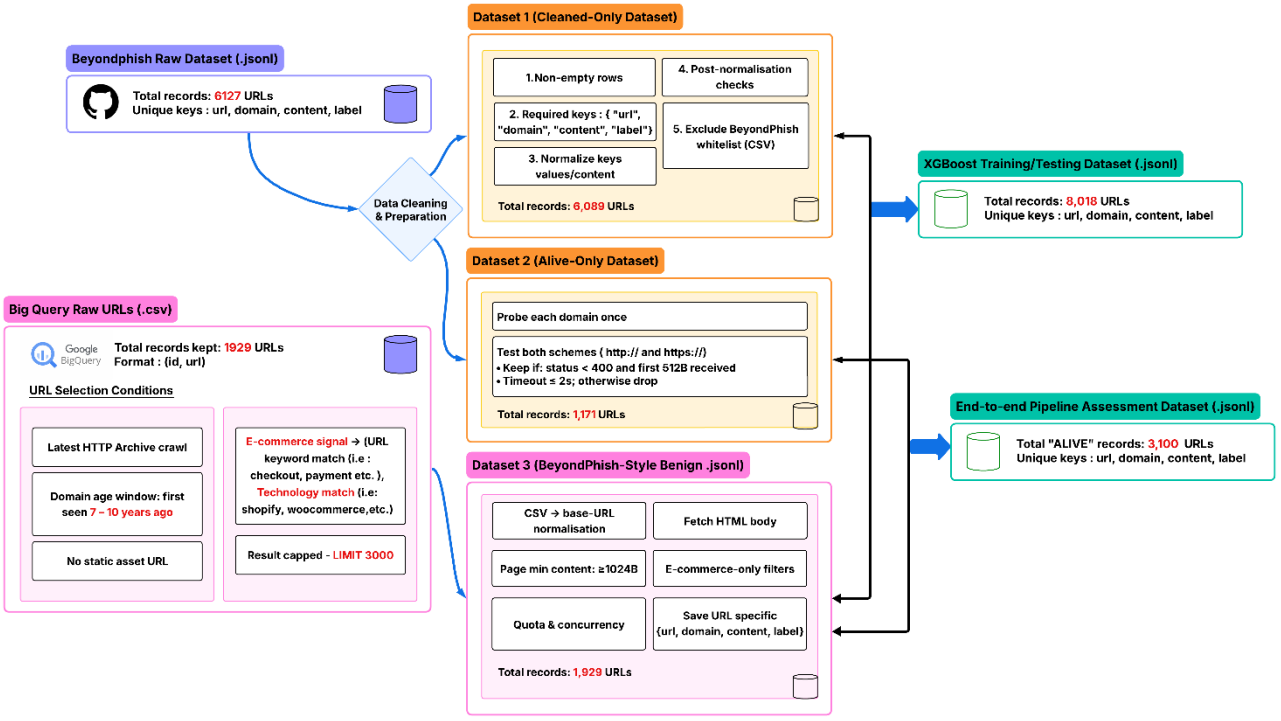

Fig. 1. Overview of dataset acquisition and preprocessing pipeline for FCW and benign e-commerce corpora.

4.1 FCW Corpus and Alive-only Subset (BeyondPhish)

In order to obtain ground truth for our research and begin implementation, we required a ready-to-use dataset that is validated, widely recognized as containing valuable FCW data, and compatible with our network requirements. We selected the publicly available BeyondPhish FCW dataset, which the authors provide via GitHub for academic and research use. This dataset is an ideal foundation for classification because its feature set aligns with FCWs: it combines content, DNS/WHOIS, URL, and social-media cues (e.g., missing or invalid social links, cheap TLDs/registrars, domain age) derived from a large cross-category analysis of FCWs. Its proven accuracy and generalization were demonstrated by high detection rates with low false-positive rates on held-out data, a 14-month user study, and validation against a large third-party (Palo Alto Networks) dataset, showing that the signals transfer beyond the training set [9]. Moreover, their research reveals that blocklists and phishing defenses barely cover FCWs and that public FCW datasets were scarce, justifying our focus and feature selection. The model is interpretable and robust: SHAP and ablation studies identify the most influential signals, and robustness analysis links many evasion techniques to real attacker costs (e.g., aged domains, social followers), informing policy decisions. We noticed alternative approaches are weaker for our goal. Phishing-centric or visual baselines such as Cantina+ [2] underperform on FCWs as demonstrated by [9], and content-only TF-IDF methods miss behavioral and domain signals, resulting in lower recall or higher false-positive rates. Platform-dependent, merchant-ID and review-mining methods rely on external visibility and introduce latency; they do not provide portable, early-layer signals that can be enforced at the network edge. Therefore we adopted BeyondPhish-style features as our baseline because they are FCW-specific, interpretable, validated at scale, and portable to our edge pipeline, unlike phishing-centric or platform methods.
The first dataset, the FCW corpus, was extracted from the public BeyondPhish JSONL file (fields: URL, domain, content, label = 1). After cleaning, removing malformed or empty lines, missing keys, and whitelisted rows the dataset was reduced from 6,127 to 6,089 valid records. This dataset will later be combined with the Benign BQ dataset of 1,929 records to produce a dataset of 8,018 records used to train our classifier. A second sub-dataset, an “alive-only” BeyondPhish subset, was created by probing URLs for HTTP reachability and retaining only active FCW URLs, reducing the original 6,127 entries to 1,171 live records. This subset will be used together with the Benign BQ dataset to create a dataset of 3,100 records used to then automate and simulate real HTTP/HTTPS requests for traffic analysis and latency measurements in the SDN network (see Fig. 2).

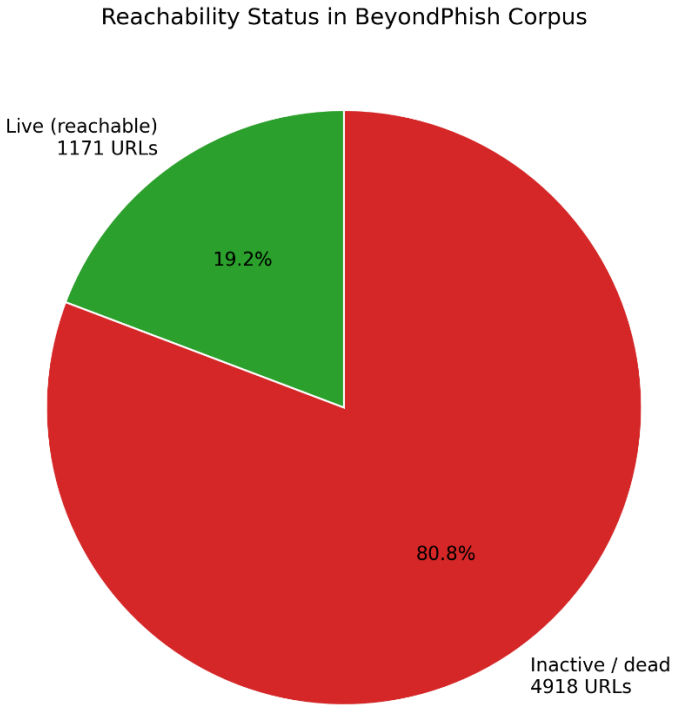

Fig. 2. Status-Distribution of FCWs in BeyondPhish Dataset.

4.2 Benign BigQuery Dataset

The second dataset a benign corpus was created with BigQuery and the HTTP Archive by gathering long-standing e-commerce domains (7–10 years old) as verified legitimate samples that match the descriptions of long-standing legitimate domains described in [9],[12]. We selected pages that displayed clear e-commerce footprints, such as checkout or cart, or that used recognizable CMS platforms (WooCommerce, Shopify, Magento, etc.), and we excluded all static assets. Each page was then converted into a BeyondPhish-like JSONL record, yielding 1,929 entries. (see Fig. 3).

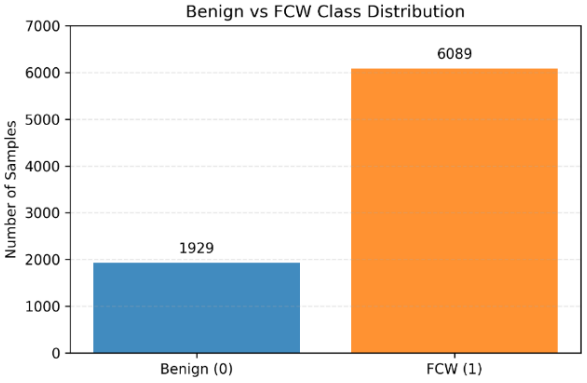

Fig. 3. Distribution of FCWs and Benign Datasets.

4.3 Tranco Dataset

To implement reliable whitelist filtering in our pipeline, we adopted the publicly available Tranco dataset [31], downloaded from its official website. While fraud detection focuses on identifying malicious activity, we can also use this list to recognize legitimate URLs and domains before they reach the classifier or network layer for user access. The dataset contains 3,944,575 URLs, allowing us to quickly verify whether a captured URL is already known as “good,” eliminating the need for further classification and simplifying the process of handling reputable sites compared to tracking fraudulent ones.

5. FEATURE ENGINEERING AND CLASSIFICATION MODEL

5.1 Feature Space for Fraudulent E-Commerce Websites

To detect Fraudulent E-Commerce Websites (FCW) in the wild, we must first understand the behaviors attackers exhibit in different environments. Feature importance varies with context whether the target is a search-engine result set, a similarity-based filter, or a large-scale FCW detector so studying these behaviors reveals reliable indicators. As a baseline, we adopted the feature set proposed in [9], which was used to evaluate BeyondPhish. These features were selected because, unlike many prior studies, they can be extracted solely from the URL and network characteristics without depending on external services. [9] identified four major feature categories for FCW detection: content-based, DNS-based, URL-based, and social-media-based.

- Content-based: Analyzing HTML source for cues such as valid social-media links (e.g., Instagram, Facebook, Twitter) that fraudsters often embed as non-existent logos, the number of external links (few external links suggest attempts to limit discoverability), and the count of script tags (a high count may indicate malicious activity).
- DNS-based: Examining domain age (malicious domains average 2 years versus 13 years for legitimate sites), registration period (most malicious domains are registered for only one year), registration country (often differs from the host country), host country, the match between host and registration country (legitimate sites are three times more likely to share the same country), and registrar cost (cheap registrars are a strong indicator of fraud)?"
- URL-based: Noting inexpensive top-level domains (e.g., .xyz, .store), the presence of hyphens or digits in the domain (common obfuscation tactics), and the depth of sub-domains (attack sites typically use many sub-domains to keep costs low).
- Social-media-based: Assessing the age and activity of linked social-media profiles on the three major platforms; fraudulent sites often link to newly created accounts with few or no followers, using the links merely as logos.

By integrating these features with the newly added ones and accounting for their interactions, we achieved a detection performance of 97.28% in our evaluation.

5.2 New FCW Classification Features

To improve the classifier's accuracy in detecting FCWs, we focused on features that are both inexpensive and quick for attackers to exploit. Fraudsters typically aim to iterate their attacks as fast and as cheaply as possible, which led us to identify a heavy reliance on specific content-management systems (CMSs) such as WordPress, Shopify, PrestaShop, Wix, and BigCommerce. These platforms simplify website creation and deployment, eliminating the need for extensive coding knowledge and falling squarely into the content-based feature category. The significance of CMS usage is underscored not only by source [32], which notes, "One of the most salient examples is Shopify, which enables a layperson to become a fully functioning online retailer without ever resorting to writing code," and adds, "Not only does it reduce the cost of launching an e-commerce business; it also makes it substantially more accessible, as no direct coding skills are necessary" but also reported by Bitaab et al. [12], their findings show that 58.74% of fraudulent shopping sites were built on Shopify, 19.13% on Wix, 18.69% on Squarespace, and the remaining 3.42% used no well-known e-commerce platform. By combining these CMS-specific features with beyond-phish indicators, we obtain robust, discriminative signals for FCW detection.

New feature categories per CMS are as follow:

- WordPress markers include the "Meta Generator Tag," "WP Content Path," "Includes Path," "REST API Path," and Gutenberg block elements. Additional scripts: "Emoji Script," "Embed Script," "Admin AJAX," and integrations such as "Jetpack." If WooCommerce is present, look for "WooCommerce Markup," "Add-to-Cart Script," and "Cart Fragments Script."
- Shopify telltale assets like the "Shopify CDN," template logic tags such as "Liquid Section Tags," and scripts under "Shopify Globals." Distinctive paths and APIs include "Cart & Checkout Paths," "AJAX Cart APIs," "Payment Buttons," "Theme Assets," "Liquid Snippets," "GraphQL API," and optional "App Proxy Paths."
- Wix indicators comprise static hosts ("Wix Static Domains"), JavaScript "Viewer Scripts," and markup attributes ("Wix Data Attributes"). Other identifiers are "Wix CSS Assets," "Wix Config Variables," "Editor Tags," and media URLs ("Wix Media URLs").
- BigCommerce detection points feature "BigCommerce CDN URLs," "Stencil Theme Assets," template markers like "Handlebars Comments," and storefront scripts ("Stencil API Scripts"). Additional cues include "Cart/Checkout Paths," "Store Data Attributes," and an inline "BC Configuration Object."
- PrestaShop sites reveal themselves through "Modules Path," "Themes Path," and "Product Image Paths." Further signs are controller URLs ("Index Controller"), the global "PrestaShop JS Object," legacy "Classic Modules," template notes ("Module Template Comments"), core asset filenames ("CSS & JS Filenames"), key hook classes ("Product/Cart Hook Classes"), and form actions ("Controller Endpoints")

These CMS-specific fingerprints, together with the existing BeyondPhish features and interactions, create a comprehensive set of discriminative attributes that improve the classifier's ability to identify FCWs efficiently and accurately. As a result, the feature count rises from 527 to 605.

5.3 XGBoost Configuration

We used a predefined hyperparameter set for the XGBoost model within a binary-classification evaluation pipeline. The model was trained with n_estimators = 700, max_depth = 6, learning_rate = 0.05, and both subsample and colsample_bytree set to 0.9. To address class imbalance, scale_pos_weight was computed dynamically as the ratio of negative to positive samples in the training data. We held out 20% of the data for testing and used the remaining 80% for training; from this training portion a further 12% was reserved as a validation slice. The validation set was used to calculate precision and recall at a manually selected decision threshold of 0.41.

It is to be noted that final hyper-parameters values were not fixed in advance; they emerged from repeated empirical testing and tuning. Several alternative configurations were evaluated on the validation split, and the chosen combination consistently achieved the best balance of validation precision, recall, F1-score, and ROC-AUC. Similarly, the threshold was selected empirically as a research-oriented operating point that emphasizes strong detection performance. Consequently, any adjustment to the decision threshold would be motivated by the specific research context, operational goals, and deployment requirements, while also accounting for the initial domain-filtering step used for whitelisting in the pipeline, whose sole purpose remains to drastically reduce FPR in deployments.

5.4 Feature Analysis

To interpret the XGBoost model's predictions, it is essential to identify which features most strongly influence each class and how they shift the decision balance toward "fraudulent" or "legitimate." We therefore calculated feature importance using the mean absolute gain contributed by each feature. We extracted SHAP-like values (excluding the bias term) for every prediction, separated by predicted class. Averaging these contributions per feature yields an importance score for the legitimate and fraudulent classes respectively. Fig. 4 displays the top 15 discriminative features for each class, based on the mean gain approximation from XGBoost decision paths and a .jsonl feature dataset derived from the full data set. The results reveal distinct weightings for each label, highlighting the features that most effectively separate fraudulent domains from legitimate ones.

Rank	Legitimate Feature	L. Score	Fraudulent Feature	F. Score
1	Misses	1.19	Misses	2.00
2	non_standard_tld	0.71	num_script_tags	0.79
3	num_script_tags	0.57	non_standard_tld	0.52
4	sm_account_3	0.46	num_external_links	0.37
5	num_external_links	0.44	cms_wp::WP_EMBED_JS	0.37
6	cms_wp::WP_EMBED_JS	0.24	sm_account_3	0.27
7	cms_presta::PS_PATH_THEMES	0.19	cms_wix::WIX_STATIC_HOSTS	0.17
8	cms_presta::PS_PATH_MODULES	0.17	cms_shopify::SHOPIFY_PAY_BTNS	0.15
9	subdomain_depth	0.17	has_digit_in_url	0.12
10	has_any_cms	0.14	has_any_cms	0.12
11	sm_account_1	0.14	cms_shopify::SHOPIFY_CART_PATHS	0.11
12	host_country_RU	0.14	cms_presta::PS_PATH_THEMES	0.10
13	host_country_JP	0.14	uses_cheap_tld	0.09
14	cms_wp::WP_PATH_WP_JSON	0.11	has_images	0.09
15	cms_bigc::BIGC_STENCIL_ASSETS	0.11	cms_wp::WP_PATH_WP_INCLUDES	0.08

Fig. 4. Top 15 Discriminative Features for Both Classes.

6. FEATURE ENGINEERING AND CLASSIFICATION MODEL

6.1 Overall architecture

To close the opportunity window that fraudsters exploit, we implemented the architecture illustrated in Fig. 5. It outlines the end-to-end identification workflow: traffic monitoring and packet capture, URL extraction, classification, and ultimately blocking or restriction. This solution is designed to answer key questions, how to collect data at scale regardless of medium, how to assess threats regardless of source, and how to restrict or mitigate attacks while minimizing opportunity windows, resolution delays, and other mitigation challenges.

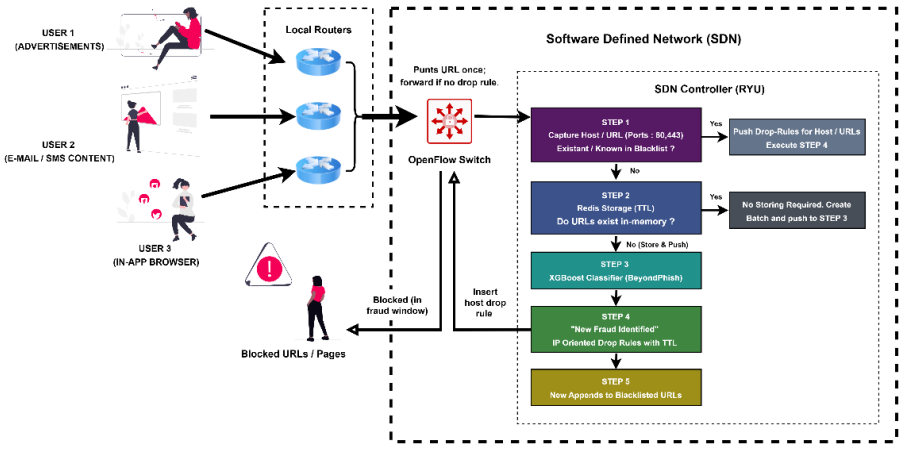

Fig. 5. Architecture of the Detection-Restriction Pipeline with Key Decision Levels.

6.2 Packet Capture, Extraction, and Filtering Process

Because every online store publishes public URLs, we can quickly identify and classify them by capturing traffic from browsers or in-app browsers. Our approach assesses HTTP/HTTPS requests on TCP ports 80 and 443 using Scapy, a lightweight yet powerful packet-capture tool that excels in research and network-testing scenarios (see [33]). Our application, utilizing Scapy, monitors traffic, extracts the domain from the HTTP Host header or the TLS ClientHello SNI field, normalizes it, and then filters out known benign domains. We maintain three CDN-filter lists KNOWN_CDN_EXACT, KNOWN_CDN_SUFFIXES, and CDN_PREFIXES to exclude major content-delivery services (e.g., Google, Facebook, Shopify, Cloudflare, Mozilla) and other static-asset providers. That with the combination of a whitelist derived from the Tranco list (+3.5 million URLs) ensures that approved domains are allowed to pass in-network. When a captured domain matches our blacklist of fraudulent or compromised websites (FCWs), we immediately install a drop rule to block further access. Otherwise, the URL is queued for analysis: it is stored in Redis with a time-to-live (TTL) value to enable deduplication, and duplicate entries for the same session are discarded. URLs that survive this check are added to a batch list for downstream classification and decision-making. Throughout the pipeline we log parsing statistics and latency metrics (see Fig. 6).

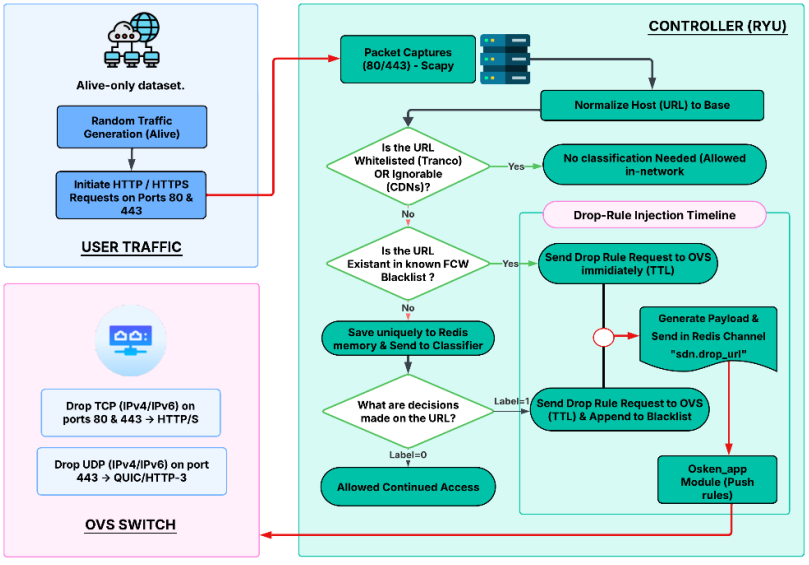

Fig. 6. Simulation and Level-Decision Logics.

6.3 Real-time Classification and Decision Logic

The batch-processing pipeline runs at regular intervals, gathering a set of recently observed base domains extracted from packets, deduplicating them with Redis, and handling them in bulk. At each interval it invokes the prediction/assessment function, which constructs a feature vector for every URL, normalizes the data, and calls the XGBoost model to classify

each domain. Only predictions with a label of 1 indicating potentially malicious behavior are kept for enforcement; all others are discarded or rather by default permitted in the network (see Fig. 6). The batch results of label=1 are written to disk and serves as the blacklist pre-classification filtering, and any domain flagged as malicious triggers an immediate drop action, blocking further interaction. Throughout the workflow, latency metrics such as feature-construction time and model-inference time are captured and stored in "STATE" for monitoring. Feature vectors consist of domain lexical characteristics, WHOIS data (e.g., registrar privacy guard, domain age), country level one-hot encodings, external-link counts, CMS-specific scripts (WordPress, Shopify, Wix), image-filename patterns, and derived combinations from BeyondPhish and manually added CMS features. Primary data sources are IP2Location for geolocation, BeautifulSoup for HTML parsing, and regular expressions for pattern extraction, all exposed by BeyondPhish features and manually selected ones. These features together enable precise behavioral discrimination in real-time malicious-URL detection.

6.4 SDN Enforcement Module

In the network, following Scapy extraction of URLs/domains during filtering and normalization, each URL is first checked against the list of already-classified “bad” URLs. This prevents redundant classification and enables rapid enforcement. The check also verifies whether the base domain has been observed before and explicitly blacklisted. When a new domain, passed the point of filtering and classification is labeled malicious (label=1), the pipeline builds a structured payload containing the domain, its top-K (K=2) resolved IPs, and the target ports (default 443 and 80). The payload is either sent directly to the OS-Ken SDN controller or published asynchronously via Redis to the ‘sdn.drop_url‘ channel for high availability concerns. The controller’s OS-Ken SimpleBlocker13 module listens on this channel, parses the JSON, and installs OpenFlow v1.3 drop rules with hard timeouts. These rules block TCP and, optionally, UDP (to stop QUIC on 443) using the highest priority to guarantee enforcement. Within OS-Ken’s application, the controller calls an iteration over all connected datapaths and install OFPFlowMod DROP entries for each malicious IP-port pair. The match format adapts to the IP version (IPv4 vs. IPv6), and barrier messages confirm the flow entries for immediate effect. A background Redis worker thread enables cross-process drop signaling, and the controller resolves any URLs to IPs that are not already supplied.

7. EXPERIMENTAL SETUP

This section outlines the testbed, software, and configurations used to perform the required SDN network simulation and testing of both the XGBoost classifier and the detection-to-restriction pipeline. In the implementation we used the GNS3 virtual environment for network simulation as it integrates seamlessly with various platforms including Linux-based routers, firewalls, and SDN tools such as Ryu and Open vSwitch allowing realistic testing without physical hardware [35]. Its support for live traffic generation, packet capture [36], and interfaces with tools like Scapy makes it ideal for academic and experimental research in network security and programmable networking (see Fig. 7).

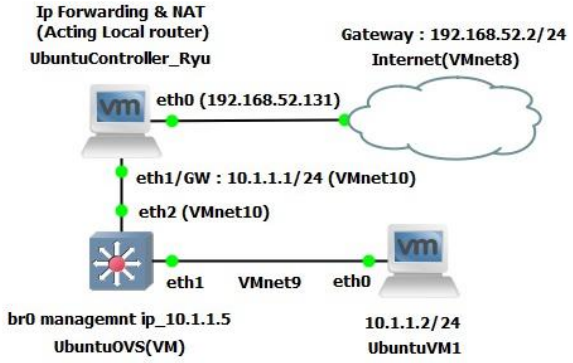

Fig. 7. GNS3-Simulated FCWs detection SDN Network.

We used VMware Workstation 17 to host three Ubuntu virtual machines, each serving a specific function.

- The Controller VM runs packet-capture, filtering, and classification modules and incorporates the OS-Ken library an SDN controller framework built on Ryu and maintained by OpenStack. OS-Ken provides a lightweight, modular system for managing OpenFlow-enabled switches via Python [34] and maintains continuous communication with the OVS switch.
- The Open vSwitch (OVS) VM runs OVS software to evaluate drop-rule policies.
- The Client VM generates HTTP/HTTPS requests for end-to-end testing.

To ensure reliable controller operation, we configured the controller VM as the local router by enabling NAT and IP forwarding. This setup lets every virtual machine reach the Internet through the GNS3 cloud, which provides shared-network connectivity. Consequently, the client VM obtains full Internet access, which is essential for feature-extraction,

classifier validation, and real-world demonstration. We permanently enabled IP forwarding on the controller VM, applied NAT with iptables on the ens33 (eth1) interface, and made the firewall rules persistent using the iptables-persistent package. For SDN control, we created a Python virtual environment, upgraded pip, and installed OS-Ken without build isolation to guarantee compatibility with Python 3.12. The controller then listens on TCP port 6633, enabling communication with OVS instances and allowing drop-rule commands via the OS-Ken application.

On the switch, we configured OVS to maintain a constant connection to the controller. Upon connection, OVS receives a default flow that permits normal traffic (‘sudo ovs-ofctl add-flow br0 actions=NORMAL ‘), allowing traffic to pass through the switch. The virtual bridge br0 was assigned the IP 10.1.1.5/24, with routing rules for intra-subnet communication and gateway access via 10.1.1.1; DNS resolution was set in/etc/resolv.conf. For robustness, OVS was placed in secure fail-mode (preventing fallback switching if the controller becomes unreachable) and pointed to the controller VM’s IP on TCP port 6633, establishing the control channel for flow-rule enforcement.

8. RESULTS

8.1 Model-Level Intrusion Detection Performance

8.1.1 Confusion Matrix Analysis

The confusion matrix provides a clear breakdown of prediction outcomes being true positives, true negatives, false positives, and false negatives. It reveals not only how many predictions were correct or incorrect, but also the specific types of errors, allowing us to see whether the model tends to miss positive cases or falsely label benign ones in the 8,018 sample test set (see Fig. 8).

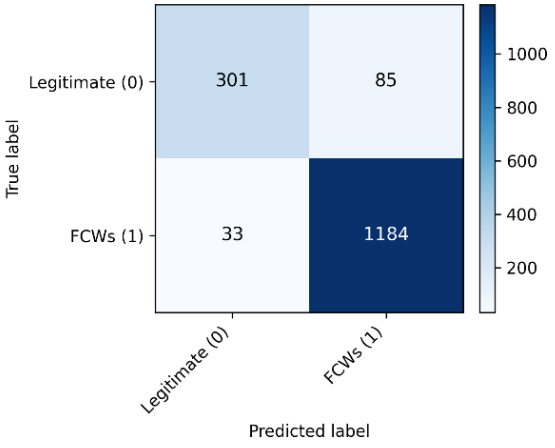

Fig. 8. Confusion Matrix Heatmap.

8.1.2 Accuracy, Recall, Precision, and Miss Rate Analysis

The Accuracy gives a quick snapshot of overall correctness obtained at 92.63%. Recall (true-positive rate) shows the proportion of actual threats captured 97.28% while the false-negative rate indicates the 2.7% being missed. Precision answers the question: of the items we flagged, how many were truly malicious? It stands at 93.30%. The F1 score, which balances recall and precision, is 95.25%, providing a single metric that reflects both missed threats and false alarms.

8.1.3 Assessment of False Positives

The false-positive rate (FPR) at the chosen operating point is 22.02% and should be reported explicitly. Although this figure might suggest that all flagged sites are blocked, its real-world impact is mitigated by upstream filtering and URL-state handling especially the Tranco whitelist and the tracking of known legitimate URLs. Consequently, the reported FPR reflects classifier performance on the development dataset (8,018 records), not on the end-to-end pipeline, which was evaluated only on live URLs (3,078 records). Thus, the classifier-level FPR does not directly translate into the same level of disruption for live traffic.

8.1.4 ROC and Precision–Recall Curve Analysis

The ROC curves illustrate the trade-off between detecting threats (TPR) and generating false alarms (FPR) across various thresholds, achieving an AUC of 96.92% (see Fig. 9). In contrast, PR curves focus more directly on positive-class performance, which is especially important when positives are scarce. Together, these curves provide a comprehensive view of the model’s ability to distinguish bad from good at any threshold (see Fig. 10).

8.1.5 Ablation Outcomes of CMS-Related Features

To evaluate the impact of the newly introduced CMS-related features, we conducted an ablation study on a held-out test set using a consistent evaluation protocol. We compared the full model (containing all CMS cues) with two reduced variants: one that omitted only the raw CMS fingerprint flags, and another that removed the entire CMS feature block (raw flags, presence bits, and CMS-based interaction features). The three-model ROC analysis showed that the full model achieved the strongest discrimination; performance dropped when raw CMS flags were excluded and declined further when the whole CMS block was removed. This confirms that CMS-related features add meaningful predictive value beyond simple threshold selection (see Fig. 11).

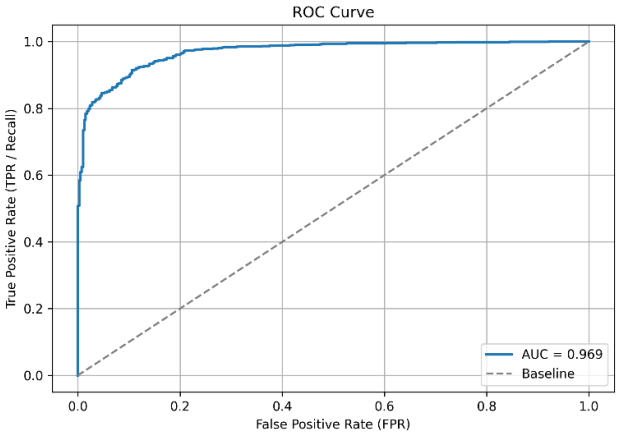

Fig. 9. Receiver Operating Characteristic (ROC) Curve.

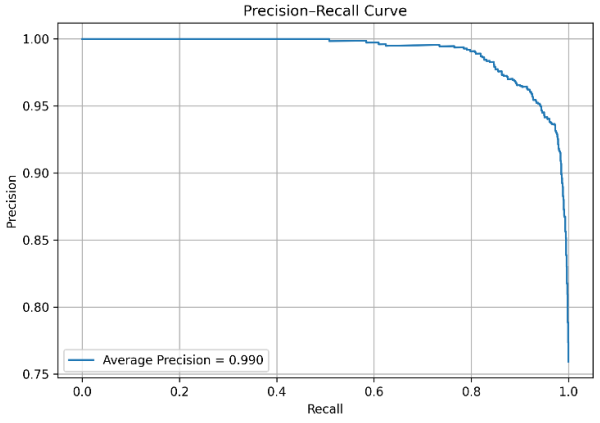

Fig. 10. Precision-Recall (PR) Curve.

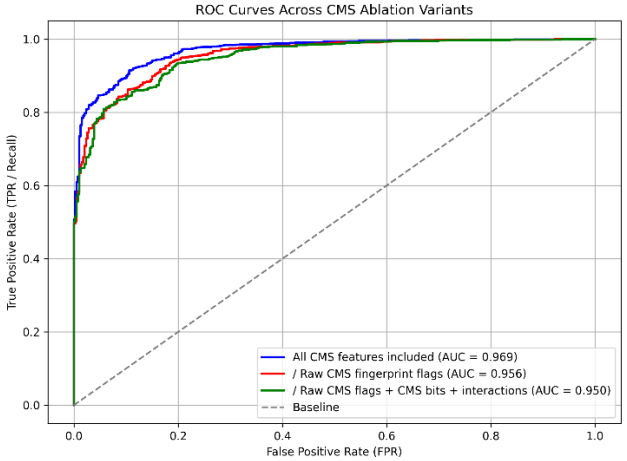

Fig. 11. ROC comparison of the full model and CMS feature ablation variants.

8.1.6 Empirical Cumulative Distribution Function (ECDF) of Features

Fig. 12 displays the ECDF of 10 high ranked discriminatively engineered feature, contrasting legitimate and fraudulent websites. We began by ranking features for each class according to their mean absolute contribution ("Score"). After selecting the top k=25 features, we identified the common subset and limited it to the first 15 entries, ordered as 0, 1, 2, 3, 4, 5, 6, 7, 13, 14 for the ECDF representation. The ECDF lets us compare the full distribution of a feature across the two groups, not just their averages. Most features show a clear separation, although a few such as non-standard TLDs, CMS-related features have nearly identical distributions. For the "Misses" (missing features returned), the legitimate curve rises earlier, indicating that legitimate sites tend to have fewer misses, while the fraudulent curve is shifted to the right, reflecting higher miss counts. This visualization reveals how often and under what conditions specific features can differentiate the two classes when used together, enabling effective classification of both legitimate and fraudulent sites.

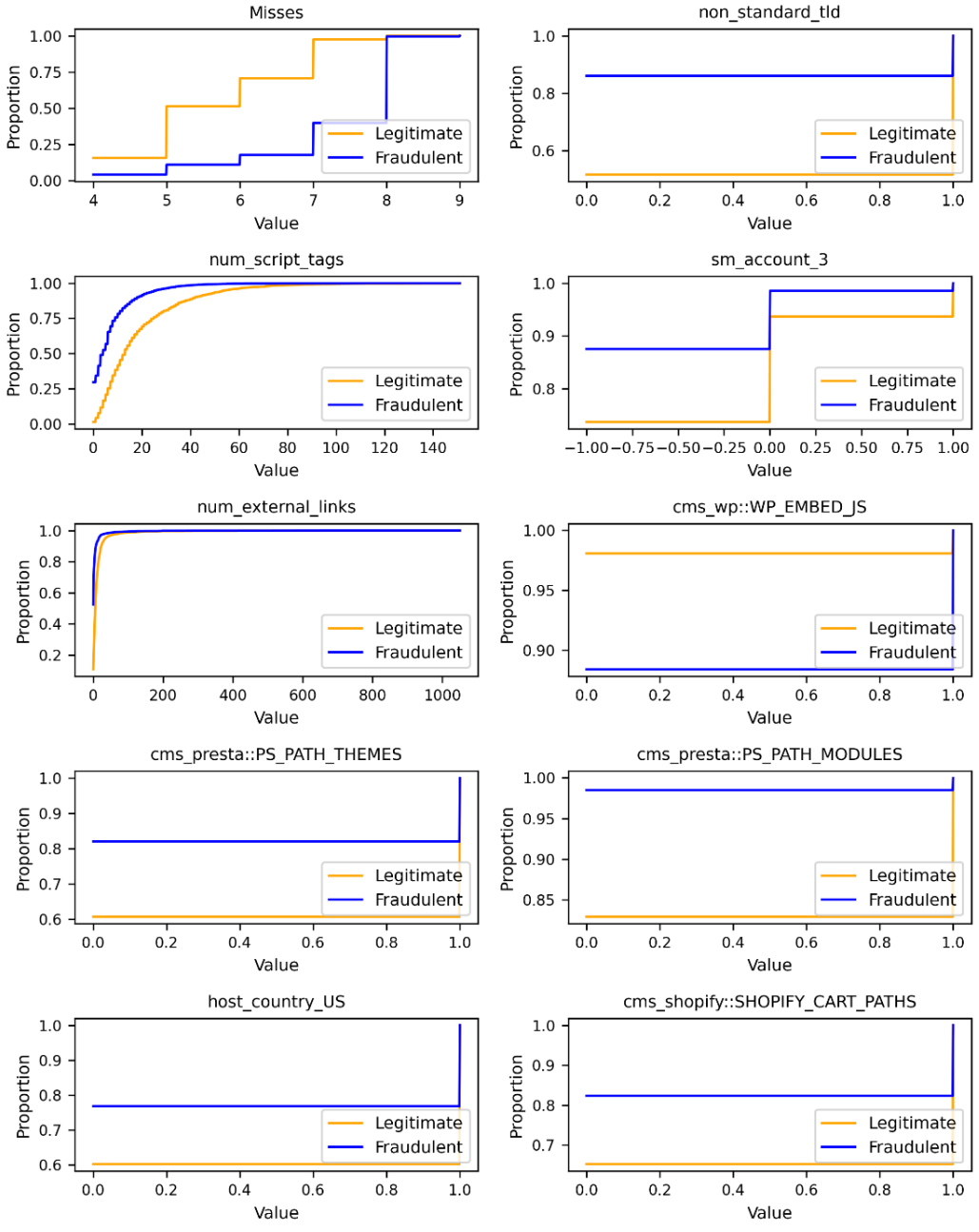

Fig. 12. ECDF plots of the distribution of 10 high ranked features across different labels.

8.1.7 Contextual Benchmarking Against Existing FCW Detection Approaches

Table I places our study within FCW detection literature, showing our results alongside prior systems. The enhanced BeyondPhish XGBoost model achieved 97.28% recall and a 0.953 F1-score. Because other studies use different data, features, and conditions, these figures serve as contextual benchmarks, not direct comparisons. Unlike most approaches that only detect and report, our work also adds in-network enforcement for real-time containment.

TABLE I. CONTEXTUAL COMPARISON OF FCW DETECTIONS UNDER DIFFERING EVALUATION SETTINGS

Approach / Study	**Recall (↑)**	**F1 Score (↑)**	**Enforcement Strategy**
Our model (BP-XGBoost) + Feature Enhancements	**97.28%**	**0.952**	**In-Network**
BeyondPhish (2023)	72.79–91.92%	0.810–0.92	Report-Based
BeyondPhish Tested CheckPhish (2020)	18.87%	0.012	Report-Based
Cantina+ (2011)	79.72%	0.769	Phishing-Focused
RealTime – L. Beltzung et al. (2020)	69.07%	0.773	Report-Based
Scammagnifier (BeyondPhish, 2025)	59.30%	N/A	Report-Based
Scammagnifier + Auto-Checkout (2025)	76.74%	N/A	Report-Based
Netcraft Extension	23.25%	N/A	Report-Based
Scamdog Millionaire (2023)	89–95%	**0.86–0.97**	Report-Based
Wadleigh et al. (2015)	54.4–76.9%	N/A	Report-Based
Beltzung et al. (2020)	93–96%	**0.95–0.97**	Report-Based
Sánchez-Paniagua et al. (2021)	70.59–94.12%	0.6809–0.75	Report-Based
BeyondPhish – ScamNet (2025)	90.10%	90.19%	Report-Based

8.2 Pipeline and SDN Enforcement Performance

8.2.1 Controller Initialization and Redis Subscription

Fig. 13 depicts the controller's runtime readiness. It shows the OSKen application launching in verbose mode, loading the OpenFlow handler, subscribing to the Redis channel sdn.drop_url at localhost:6379/0, and completing the OpenFlow handshake with the switch (DPID shown). Once the handshake finishes, the application installs the baseline table-miss NORMAL rule, confirming that the control plane is active and ready to enforce drop-request policies.

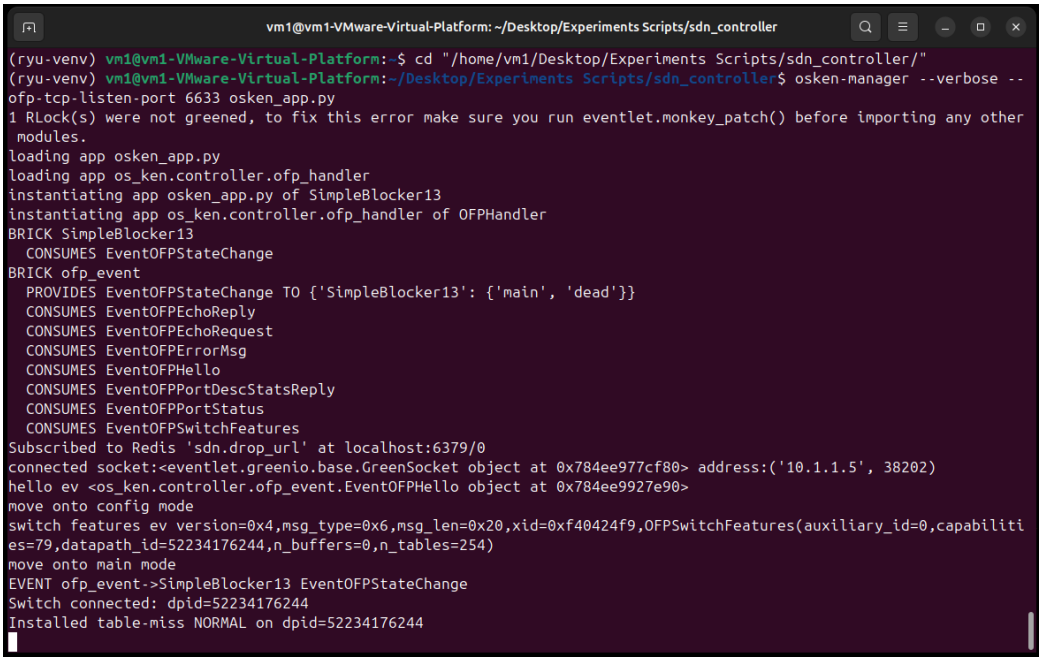

Fig. 13. OSKen Controller Startup, Redis Channel Subscription, and Switch Connection Procedure.

8.2.2 OVS-Level Drop Rule Enforcement

Fig. 14 illustrates the drop rules applied at the OVS level and shows how the controller enforces IP-based containment. It installs high-priority OpenFlow DROP entries on br0 for specific destination IPs (e.g., 142.214.215.226, 13.223.25.84, 54.243.117.197) and blocks HTTP/HTTPS traffic by dropping TCP flows on ports 80 and 443, with additional UDP/443 drops where needed, thereby preventing outbound connections to these endpoints directly at the switch.

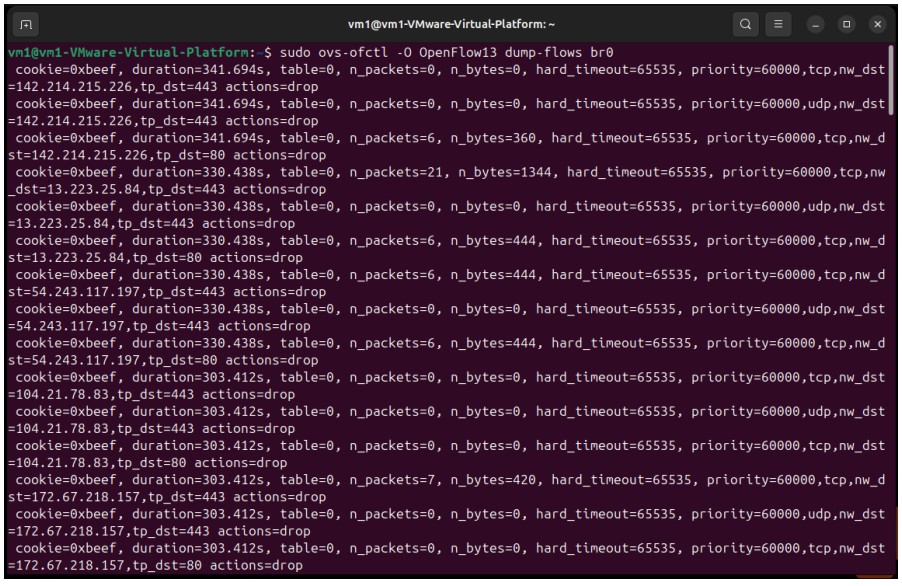

Fig. 14. Sample OVS Drop Rules for IP-Based Containment.

In pipeline test rounds, we measured four key metrics:

8.2.3 Extraction latency (average 0.062 ms):

Indicates the overhead of parsing incoming URLs before any deeper analysis. It reflects the system's responsiveness at the first touchpoint and helps pinpoint bottlenecks in input-collection mechanisms.

8.2.4 Feature-build latency (average 12,521 ms):

Measures the time required to transform raw URLs into structured features ready for model inference. This metric captures the computational cost of engineering domain-specific signals such as lexical, WHOIS, or CMS attributes.

8.2.5 Inference latency (average 45.53 ms):

Shows how quickly machine learning model makes decisions. Faster inference is essential for real-time detection systems, where delays can lead to missed threats or slower mitigation.

8.2.6 Rule-install latency (average 2,294 ms):

Records the time the SDN controller needs to push a drop rule to Open vSwitch after a malicious domain is identified. This latency reflects the agility of the enforcement pipeline and is critical for preventing further communication with harmful endpoints.

These comprehensive metrics give the clearest view of system responsiveness under real-world conditions, integrating detection, decision-making, and enforcement. They also incorporate blacklist-verified URL enforcement, eliminating the need for extra feature development or reclassification, which improves efficiency and reduces overall latency by removing process delays. These applications enable us to reduce resolution time by leveraging previously identified opportunity windows through preventive measures rather than reactive post-event actions, thereby decreasing customers' financial losses.

9. DISCUSSION

In this section we discuss risk reduction for mobile shoppers, the limitations of our work, and ethical considerations. Our approach protects users by extending beyond browser-extension alerts whose coverage is limited to both enforce restrictions and warn users. By continuously assessing the URLs accessed by a device, the SDN framework combined with the classification metrics presented in Section 8 can detect threats in under 30 seconds. This rapid response closes the window that attackers exploit to defraud shoppers, a window that typically remains open for 15 days or more [22]. The

solution is designed to complement existing defenses on currently unchecked channels such as in-app browsers, mobile browsers (e.g., Chrome), email clients, direct-messaging apps, and social-media advertisements (Facebook, Instagram, etc.). By doing so, it mitigates concerns identified by [21] including legal and jurisdictional delays, rare prosecutions, investigative bottlenecks, victim denial, and cross-border legal complexities.

The primary threat to validity stems from visibility constraints on website URLs. For the pipeline to operate, a site must be publicly reachable and allow content analysis; many of our features rely on observable CMS characteristics.

Attackers can evade detection by restricting access to their pages through techniques such as bot protection, rate limiting, JavaScript-rendered content, geo-restrictions, IP blocks, login walls, or deliberate obfuscation. These measures can degrade classifier accuracy because they hide the very page content cues we depend on. CMS-based sites, however, very often expose predictable structures e.g., stable URLs and paths, meta tags, page titles, asset links, template markers, and plugin scripts that our model leverages to improve detection while considering the "keeping costs low for attackers". To reduce false-positive rates, we first query a live FCW URL corpus; domains not present are sent for classification. We also filter benign URLs using the Tranco dataset (+3.5 million entries), which efficiently eliminates known, low-risk domains. Another limitation is that for unknown URLs, the controller must fetch the page to extract CMS-related and site-level features before classification. Although this boosts detection accuracy, it introduces processing and network overhead in high-traffic environments, especially when many new URLs arrive rapidly. Repeated fetching, parsing, and feature extraction increase latency, reduce controller responsiveness, and hinder scalability. Therefore, the approach should be tested under heavy load, examining throughput, queuing effects, and the trade-off between richer feature extraction and real-time performance. One final limitation shows how the proposed detection approach is constrained by the sheer diversity and rapid turnover of CMS and site-building ecosystems used in real-world scam-shop campaigns. As demonstrated by BogusBazaar [39],[40], ERIAKOS [41], SilkSpecter [42], and SHOPOEM [43], attackers do not rely solely on mainstream platforms such as WordPress or Shopify; they also exploit lesser-known or specialized systems like oemapps and other provisioned storefront frameworks. This creates a practical challenge for CMS-cue-based detection because unknown, newly emerging, or quickly rotated platforms can exhibit feature patterns that the classifier has not yet learned. Consequently, evasive operators can shrink detection coverage by switching to unfamiliar toolchains, reusing niche infrastructures, or altering implementation details faster than feature updates can be incorporated. Relying only on known CMS cues therefore limits generalization, especially when attackers deliberately rotate across non-mainstream systems to evade recognition [44].

The current implementation aims to provide resolution time without granting attackers access to users, while also handling false-positive URL cases. When a URL is mistakenly flagged, it should be then added to the whitelist and removed from the blacklist to prevent repeated reclassification. The approach first targets assessment of the pipeline classifier and enforcement latency metrics. Second, it enables tracking of legitimate, existing website URLs, which is simpler and more reliable than attempting to monitor newly created sites without verification. In essence, the system prioritizes trust in known entities over unknown ones [45][46].

Together, these processes aim to (1) lower false positives, (2) maintain a high detection rate across all channels, and (3) enforce network-level restrictions that close the opportunity gaps previously exploited at scale [47].

10. CONCLUSION AND FUTURE WORK

In today’s mobile-commerce security landscape, tools intended for legitimate use are increasingly hijacked by fraudsters posing as trustworthy entities. Although recent advances have improved detection of online-shopping attacks, most methods overlook two critical factors: the post-detection response and the channels through which attacks are launched, which create an “opportunity window.” To address this gap, we built an automated, large-scale detection system for Fraudulent E-Commerce Websites (FCWs) that combines an enhanced XGBoost classifier trained on BeyondPhish features plus newly engineered attributes with a software-defined networking (SDN) application. The SDN component can impose flexible, network-level restrictions, offering broader coverage of fraud-prone channels than browser-extension-only solutions. We view this pipeline as a stepping stone for further research, as it provides deeper insight into attacker behavior and enables stricter mitigation. Because the pipeline relies on manually selected features and incurs non-negligible computational overhead, future work will explore dynamic feature and fingerprint updates that evolve as rapidly as fraud tactics. Potential approaches include deep reinforcement learning (DRL), federated learning (FL), and mixture-of-experts (MoE). Lightweight implementations of these techniques could reduce network load while improving detection precision, building on the initial introduction of large language models (LLMs) for attack detection by ScamNet [11] and keeping in-network resource usage minimal for broader coverage. Another important next step is to expand the CMS- and URL-level detection pipeline beyond the single dataset of 8,018 records used. Testing on larger, more diverse corpora would provide a firmer basis for evaluating robustness across various FCW ecosystems, languages, regions, and campaign styles. Future work should therefore incorporate multiple independently gathered datasets and perform cross dataset validation to more thoroughly assess generalization, transferability, and resilience to previously unseen fraud patterns.

Funding:

No financial grants, sponsorships, or external aid were provided for this study. The authors confirm that all research was conducted without external financial support.

Conflicts of Interest:

The authors declare that there are no conflicts of interest regarding this publication.

Acknowledgment:

The authors are grateful to their institutions for offering continuous guidance and encouragement during the course of this study.

References

[1] A. Oest, Y. Safaei, A. Doupé, G.-J. Ahn, B. Wardman, and G. Warner, "Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis," in Proc. APWG Symp. Electron. Crime Res. (eCrime), 2018, pp. 1–12, doi: 10.1109/ECRIME.2018.8376206.

[2] G. Xiang, J. Hong, C. P. Rosé, and L. F. Cranor, "CANTINA+: A feature-rich machine learning framework for detecting phishing web sites," ACM Trans. Inf. Syst. Secur., vol. 14, no. 2, pp. 21:1–21:28, 2011, doi: 10.1145/2019599.2019606.

[3] C. Ardi and J. Heidemann, "AuntieTuna: Personalized content-based phishing detection," in Proc. NDSS Workshop Usable Security (USEC), San Diego, CA, USA, Feb. 2016, doi: 10.14722/usec.2016.23012.

[4] T. K. Panum, K. Hageman, R. R. Hansen, and J. M. Pedersen, "Towards adversarial phishing detection," in Proc. 13th USENIX Workshop Cybersecurity Experimentation and Test (CSET), 2020.

[5] S. Marchal and N. Asokan, "On designing and evaluating phishing webpage detection techniques for the real world," in Proc. 11th USENIX Workshop Cybersecurity Experimentation and Test (CSET), 2018.

[6] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, "PhishTime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists," in Proc. 29th USENIX Security Symp. (USENIX Security), Aug. 2020, pp. 379–396.

[7] PhishTank, "PhishTank." [Online]. Available: https://phishtank.org

[8] OpenPhish, "OpenPhish." [Online]. Available: https://openphish.com

[9] M. Bitaab et al., "Beyond Phish: Toward detecting fraudulent e-commerce websites at scale," in Proc. IEEE Symp. Security Privacy (SP), San Francisco, CA, USA, May 2023, pp. 2566–2583, doi: 10.1109/SP46215.2023.10179461.

[10] P. Kotzias, K. Roundy, M. Pachilakis, I. Sanchez-Rola, and L. Bilge, "Scamdog Millionaire: Detecting e-commerce scams in the wild," in Proc. Annu. Comput. Security Appl. Conf. (ACSAC), Austin, TX, USA, Dec. 2023, pp. 29–43.

[11] M. Bitaab et al., "ScamNet: Toward explainable large language model-based fraudulent shopping website detection," in Proc. AAAI Conf. Artif. Intell. (AAAI), Feb. 2025, pp. 27841–27848.

[12] M. Bitaab et al., "SCAMMAGNIFIER: Piercing the veil of fraudulent shopping website campaigns," in Proc. NDSS Symp., Feb. 2025.

[13] C. Carpineto and G. Romano, "Learning to detect and measure fake e-commerce websites in search-engine results," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI), Leipzig, Germany, Aug. 2017.

[14] B. Price and M. Edwards, "Resource networks of pet scam websites," in Proc. APWG Symp. Electron. Crime Res. (eCrime), 2020.

[15] J. Wadleigh, J. Drew, and T. Moore, "The e-commerce market for 'lemons': Identification and analysis of websites selling counterfeit goods," in Proc. Int. Conf. World Wide Web (WWW), Florence, Italy, May 2015, pp. 1188–1197.

[16] A. Lindley, O. Dinica, L. Beltzung, N. Hermann, and R. Lindner, "Real-time detection of fake-shops through machine learning," in Proc. IEEE Int. Conf. Big Data (BigData), Dec. 2020, pp. 1–9.

[17] M. Cheung, J. She, and L. Liu, "Deep learning-based online counterfeit-seller detection," in Proc. IEEE INFOCOM Workshops, Honolulu, HI, USA, Apr. 2018, pp. 51–56, doi: 10.1109/INFCOMW.2018.8406896.

[18] G. Sancheti, "The presence of counterfeit products in online marketplace: Causes and implications," Master's thesis, MICA (Mudra Institute of Communications, Ahmedabad), India, 2021.

[19] M. S. Sánchez-Paniagua, E. Fidalgo, E. Alegre, and F. J. Martino, "Fraudulent e-commerce websites detection through machine learning," in Proc. Int. Conf. Hybrid Artif. Intell. Syst. (HAIS), vol. 12886, Bilbao, Spain, Sep. pp. 267–279, 2021.

[20] W. Mostard, B. Zijlema, and M. Wiering, "Combining visual and contextual information for fraudulent online store classification," in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI), Thessaloniki, Greece, Oct. pp. 84–90 ,2019.

[21] C. V. Amasiatu and M. H. Shah, "The management of first party fraud in e-tailing: A qualitative study," Int. J. Retail Distrib. Manag., vol. 47, no. 4, pp. 433–452, 2019.

[22] S. Lakkaraju, "Using machine learning to combat e-commerce fraud," Int. J. Inf. Technol. Manage. Inf. Syst. (IJITMIS), vol. 16, no. 1, pp. 844–859, 2025.

[23] S. Yin and X. Luo, "A review of learning-based e-commerce," in Proc. Int. Conf. Intell. Syst. Knowl. Eng. (ISKE), 2021, pp. 483–490.
[24] PayPal, "The Mobile Shopping Boom," White Paper, 2018. [Online]. Available: https://www.paypalobjects.com/digitalassets/c/EMEA/research/Mobile_Commerce_Report_2018_Final.pdf
[25] R. P. Jones and R. C. Runyan, "Conceptualizing a path-to-purchase framework and exploring its role in shopper segmentation," Int. J. Retail Distrib. Manag., vol. 44, no. 8, pp. 776–798, 2016.
[26] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud," in Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp), 2018, pp. 251–256, doi: 10.1109/BIGCOMP.2018.00044.
[27] A. El-Dalahmeh et al., "An intrusion detection system using the XGBoost algorithm for SDVN," in Advances in Computational Intelligence Systems, Proc. UKCI 2023. Singapore: Springer, 2024, pp. 390–402.
[28] A. Montazerolghaem and S. Imanpour, "Evaluation and performance analysis of the Ryu controller in various network scenarios," arXiv preprint arXiv:2505.19290, May 2025.
[29] S. Asadollahi, B. Goswami, and M. Sameer, "Ryu controller's scalability experiment on software defined networks," in Proc. IEEE Int. Conf. Current Trends Adv. Comput. (ICCTAC), Feb. 2018, doi: 10.1109/ICCTAC.2018.8370397.
[30] IP2Location, "IP2Location LITE: Free IP Geolocation Database for Developers." [Online]. Available: https://lite.ip2location.com/ip2location-lite
[31] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," arXiv preprint arXiv:1806.01156, 2018.
[32] G. Dushnitsky and B. K. Stroube, "Low-code entrepreneurship: Shopify and the alternative path to growth," J. Bus. Ventur. Insights, vol. 16, Art. no. e00251, Nov. 2021, doi: 10.1016/j.jbvi.2021.e00251.
[33] R. Raj et al., "SCAPY—A powerful interactive packet manipulation program," in Proc. ASTESJ Conf., Dec. 2018.
[34] Ken, "tutorial-ken," GitHub repository. [Online]. Available: https://github.com/scc333-networking/tutorial-ken
[35] D. Lal, B. Ghorbani, and S. Vaghri, "A survey on the use of GNS3 for virtualizing computer networks," Int. J. Comput. Sci. Eng., vol. 5, no. 1, pp. 49–58, Dec. 2016–Jan. 2017.
[36] Verizon, "2023 Data Breach Investigations Report (DBIR)," Verizon Business, 2023. [Online]. Available: https://www.verizon.com/business/resources/reports/2023/2023-data-breach-investigations-report-dbir.pdf
[37] Trusted Shops, "AmbienteDirect case study: Boosting conversion rates with the Trustmark." [Online]. Available: https://business.trustedshops.com/clients/ambientedirect-casestudy
[38] TrustedSite, "TrustedSite certifications generate 14% conversion lift over Norton Shopping Guarantee for The Shoe Mart," TrustedSite Blog, Jul. 9, 2020. [Online]. Available: https://blog.trustedsite.com/2020/07/09/trustedsite-certifications-generate-14-conversion-lift-over-norton-shopping-guarantee-for-the-shoe-mart/
[39] M. Marx, "BogusBazaar: A criminal network of webshop fraudsters," SRLabs Research, Aug. 5, 2024. [Online]. Available: https://srlabs.de/blog/bogusbazaar
[40] C. Kunz, "38C3: BogusBazaar gang still operating thousands of fake stores," heise online, Dec. 29, 2024. [Online]. Available: https://www.heise.de/en/news/38C3-BogusBazaar-gang-still-operating-thousands-of-fake-stores-10221661.html
[41] EclecticIQ Threat Research Team, "Inside Intelligence Center: Financially motivated Chinese threat actor SilkSpecter targeting Black Friday shoppers," EclecticIQ Blog, Nov. 14, 2024. [Online]. Available: https://blog.eclecticiq.com/inside-intelligence-center-financially-motivated-chinese-threat-actor-silkspecter-targeting-black-friday-shoppers
[42] YLabs, "Threat actors leverage Chinese SHOPOEM platforms to spread infamous scam campaign," Aug. 1, 2024. [Online]. Available: https://labs.yarix.com/2024/08/shopoem-scam-campaign/
[43] Federal Bureau of Investigation, "2023 Internet Crime Report," IC3, 2023. [Online]. Available: https://www.ic3.gov/annualreport/reports/2023_ic3report.pdf
[44] J. Hall, "Facebook Targeting: User Acquisition," SlideShare, 2014. [Online]. Available: https://www.slideshare.net/slideshow/facebook-targeting-user-acquisition/38104250
[45] R. Leppert, "About 1 in 3 Americans say they've had an online shopping scam happen to them," Pew Research Center, Nov. 19, 2025. [Online]. Available: https://www.pewresearch.org/short-reads/2025/11/19/about-a-third-of-americans-say-theyve-had-an-online-shopping-scam-happen-to-them/
[46] Liquid Web, "Who Do Americans Trust With Their Data in 2025? The Digital Trust Report," 2025. [Online]. Available: https://www.liquidweb.com/white-papers/digital-trust-report/
[47] M. Faverio and M. Anderson, "For shopping, phones are common and influencers have become a factor—especially for young adults," Pew Research Center, Nov. 21, 2022. [Online]. Available: https://www.pewresearch.org/short-reads/2022/11/21/for-shopping-phones-are-common-and-influencers-have-become-a-factor-especially-for-young-adults/