WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

CyberWatch - A Unified Web And Browser Threat Detection Platform

Prof. Shital Patil[1], Ujwal Kawale[2], Samiksha Machale[3], Shruti Bhagat[4]

[1]Asst Professor, Dept. of Information Technology, Priyadarshini Bhagwati College of Engineering, Nagpur, Maharashtra, India (affiliated to RTMNU)

ABSTRACT :

Phishing links, insecure websites, risky browser extensions and suspicious files are usually checked with separate tools that report results in incompatible forms. This paper presents CyberWatch, a unified web and browser threat detection platform whose four scanners (URL and phishing, website security, Chrome Manifest V3 extension, and static file analysis) feed a common risk-analysis layer that outputs a Threat Index from 0 to 100, a risk level and a security verdict. The production scanners are rule-based; the Threat Index is an expert-defined ordinal score, not a probability, and the paper gives its formal definition, properties and a validation protocol. Evaluation is reported in separate scopes. Automated functional testing passed 350 of 352 tests (99.43%). On a domain-grouped held-out set of 400 URLs, offline Logistic Regression and Random Forest baselines reached 72.25% (95% CI 67.7–76.4) and 86.00% (82.3–89.1) accuracy, whereas the production heuristic, evaluated offline with live signals disabled at a frozen threshold of $R \geq 50$, detected none of the 212 threats (recall 0%, upper 95% bound 1.8%) and raised one false alarm in 188 benign URLs. We analyse this negative result, specify the hardening needed to close a redirect-related SSRF aggregation gap, document dataset provenance and an unresolved record-count discrepancy, and present screenshots of the implemented platform with the scan evidence they show. A synthetic calibration study reduced macro ECE from 0.158 to 0.010 and is not real-world evidence. Local benchmarks gave sub-millisecond scanner latency (27.76 ms for 1 MB file analysis) and 986.85 requests/s peak throughput with no failed requests; these do not establish production capacity.

Keywords: Threat detection; Phishing URL; Website security; Manifest V3 extension; Static file analysis; SSRF; Risk scoring; Calibration
Platform stack: React, TypeScript, FastAPI, Pydantic, MongoDB, JWT

1. Introduction

A single browsing session can expose a user to a shortened link, an unfamiliar website, a helper extension and a downloaded file. The structure of a URL carries signal about whether an address is dangerous before the page is opened [1, 2]; extensions hold permissions that reach into the session, and malicious ones have been found in official stores [3, 4, 5]; and files remain a malware delivery route for which static inspection is a low-cost first check [6]. These checks are normally made with separate tools, so results are scattered, risk is described in different vocabularies, and nothing shows a combined picture. A further problem is semantic: a score built from expert-chosen weights can rank items within one scanner, but nothing guarantees that the same value means the same likelihood of harm in another scanner.

CyberWatch keeps vector-specific analysis in four scanners and expresses every result through one Common Risk Analysis layer. The contributions are of two kinds.

Engineering contributions. (1) A modular platform with URL, website, Manifest V3 extension and static file scanners behind one risk layer, one record store and three monitoring views; (2) an API boundary with JWT authentication, Pydantic validation and SSRF protection; (3) a clear separation of production functionality from offline machine-learning experiments; (4) a working web interface, documented here with screenshots of each scanner and monitoring view (Section 6).

Research contributions. (5) A formal definition of the Threat Index with stated properties, an explicit statement of what it does not mean, and a validation protocol; (6) a multi-scope evaluation on a domain-grouped dataset that reports a negative result for the production heuristic alongside stronger offline baselines, with confidence intervals; (7) an analysis of reproducibility gaps and SSRF redirect-handling requirements. The novelty claim is limited: the specific integration was not found in the literature reviewed, and no claim is made that CyberWatch detects better than other tools.

2. Problem Statement and Objectives

The problem is the fragmented analysis of web and browser threats: (i) workflows are split across tools per vector; (ii) results use different scales and vocabularies; (iii) non-specialists need one risk representation to prioritise findings; and (iv) extension and page context sit inside the browser.

The objectives are: (O1) implement four scanners in one platform; (O2) give them a common output of Threat Index, risk level and verdict; (O3) support Manifest V3 extension analysis; (O4) protect the backend that handles untrusted input; (O5) test the platform functionally; (O6) establish offline baselines on a leakage-controlled URL dataset and report the production heuristic on the same data; (O7) measure local latency and throughput; and (O8) state precisely what the score means and what remains unvalidated.

3. Literature Review

This section reviews prior work for each analysis vector handled by CyberWatch and for the cross-cutting issues of risk scoring and evaluation. It is a focused narrative synthesis of the sources most relevant to the platform's design, not an exhaustive survey, and each subsection ends with what the work implies for CyberWatch.

3.1. Malicious URL and phishing detection

Early work showed that lexical and host-based features of a URL can separate malicious from benign sites without fetching the page [1]. Sahoo et al. [7] survey the field and organise it by feature representation (lexical, host-based, content-based and contextual) and by learning approach (batch, online and representation learning). Sahingoz et al. [2] compared classifiers, including Random Forest, on URL-text features without blacklists, and the ISCX-URL2016 study [8] characterised lexical features for benign, spam, phishing, malware and defacement URLs. Production systems use signals beyond the address: Whittaker et al. [9] describe automatic large-scale classification of phishing pages that combines URL, page-content and hosting features. Deep models such as URLNet [10] learn character- and word-level URL representations with convolutional networks and avoid manual feature design, but they need large labelled sets and give little per-indicator explanation.

Implication. Lexical baselines are strong enough that a rule-based URL scanner must be benchmarked against them (Section 8); the optional live DNS, TLS and DOM signals of CyberWatch are the counterpart of the non-lexical signals in [9].

3.2. Website scanning and SSRF

A scanner that fetches user-supplied addresses must protect its own network position. The OWASP guidance [11] recommends application- and network-layer controls, robust address validation, validation of resolved IPv4/IPv6 addresses, connecting to the validated address, and restricting automatic redirects; each redirect destination must satisfy the same policy before a connection is made. Jabiyev et al. [12] study how to prevent SSRF attacks in applications, DNS rebinding is a distinct bypass [13], and special-purpose address registries [14, 15] define the ranges a destination policy must reject. The redirect-to-loopback test in this work shows that transport was aborted but the scanner did not aggregate an explicit SSRF/blocked finding: a reporting gap, not evidence that an internal resource was fetched.

Implication. Fetching-based scanners need per-hop validation and typed blocking errors; Table 4 turns this literature into test cases.

3.3. Browser extension security

Carlini et al. [16] evaluated the Chrome extension security architecture and found that extensions, especially those with broad privileges, were often vulnerable, which supports reading declared permissions as exposure. Hulk [3] used dynamic analysis on about 48,000 Chrome Web Store extensions and found malicious ones, including one affecting 5.5 million users, and Jagpal et al. [4] report trends from three years of fighting malicious extensions at Google. Nayak et al. [5] analysed how browser extensions access sensitive data. Risk is also temporal: Pantelaios et al. [17] detect malicious extensions through the changes between versions (update deltas), so a single-snapshot scan can miss a later malicious update. For Manifest V3 [18], Pantelaios and Kapravelos [19] found 154 of 517 malicious extensions (29.8%) remained functional after MV3 conversion and 290 (56%) retained harmful capabilities after adaptation, so MV3 compliance alone is not evidence of safety.

Implication. CyberWatch statically reviews a Manifest V3 manifest: broad host access or sensitive permissions indicate exposure, not malicious intent, and the prior studies are not an accuracy estimate for this scanner. Version-to-version comparison is not implemented.

3.4. Static file analysis

Shalaginov et al. [6] survey machine-learning-aided static malware analysis. Entropy analysis has long been used to locate encrypted and packed code [20], although compressed benign files also have high entropy, so it is a weak indicator on its own. EMBER [21] is an open benchmark for static PE classification (1.1 million binaries; 900,000 training and 200,000 test), and MalConv [22] learns directly from raw bytes of whole executables.

Implication. The file scanner reports structural indicators such as type consistency, entropy and extracted strings (Section 6); these benchmarks are design examples, not evidence of its accuracy, so file results are reported separately from URL classification.

3.5. Risk scoring, calibration and evaluation

Sommer and Paxson [23] argue that machine learning in intrusion detection faces a different problem from other domains because of its open-world, high-cost-of-error setting, and Arp et al. [24] identify sampling bias, leakage and unsuitable assumptions as recurring pitfalls in security ML. CVSS [25] standardises severity, not the probability that an item is malicious; likewise the Threat Index is an ordinal score. Guo et al. [26] show that neural-network confidence can be poorly calibrated, and calibration is studied for several supervised learners [27]. Platt scaling [28], isotonic regression [29], the Brier score [30] and ECE [31] are candidate tools, but all require labelled calibration data.

Implication. Shared scores across scanners do not imply shared probabilities; calibration is therefore treated as proposed work with a synthetic feasibility study (Section 8).

3.6. Research gap

The comparison is a focused narrative synthesis, not an exhaustive review. The literature treats each vector largely in isolation: URL classifiers, SSRF defences, extension analysers and file classifiers are studied and evaluated separately, and severity or calibration work is rarely applied across vectors in one platform. CyberWatch's defensible contribution is the engineering integration of four scanners with common records, explanations and monitoring

views. The evidence does not establish that no comparable platform exists, a new detection algorithm, or an advantage over specialised tools. The open gap is validating the production rule set, securing outbound fetching, and determining whether per-vector calibration can give the score a measurable meaning.

Table 1. Summary of the reviewed literature and its implications for CyberWatch.

Area	Representative work	Main contribution	Boundary for CyberWatch
URL detection	[1, 2, 8, 7]	Lexical and host features; supervised classifiers; survey	Heuristic weights are not validated against these baselines
	[9, 10]	Large-scale phishing classification; deep URL representations	Needs data volume and gives limited explanation
Website and SSRF	[11, 12, 13]	SSRF prevention, DNS rebinding defence	Controls need implementation tests; reporting gap open
Extensions	[16, 3, 4]	Extension architecture, dynamic analysis, store experience	Static manifest review is not a dynamic benchmark
	[17, 19]	Update-delta detection; MV3 migration security	MV3 does not imply benign; no version comparison
Static files	[6, 20, 21, 22]	Static analysis survey, entropy, EMBER, raw-byte learning	Structural checks only; no classifier or sandbox
Scoring and evaluation	[23, 24, 25]	Evaluation pitfalls; severity is not probability	Needs realistic, auditable evaluation
	[28, 29, 26, 27]	Calibration methods and metrics	Needs labelled data; only synthetic study here

Table 2. Evidence used by each scanner.

Scanner	Input	Evidence	Executes target?
URL	address string	lexical; optional live DNS/TLS/DOM signals	no
Website	address, then target	transport, redirect, response	fetches only
Extension	MV3 manifest	declared capability	no
File	uploaded file	structure	no

4. Proposed System and Architecture

4.1. Architecture and workflow

Fig. 1 shows the architecture. The frontend accepts submissions and renders results. The backend API authenticates and validates each request and applies SSRF protection to outbound requests. Orchestration routes the submission to one of four scanners, whose findings pass to the Common Risk Analysis layer, the single place where indicators become a Threat Index, risk level and verdict. Each completed scan is stored as a security record read by the Security Dashboard, Recent Security Activity (scan history) and the Behaviour Monitor. An AI explanation component produces text grounded in scanner findings. The offline research environment (dashed) trains and evaluates baselines and is not in the production pipeline. Only the analysis stage differs between scanners; the scanners are never merged into one measurement, the shared layer only gives their outputs the same form (Fig. 2, Table 2).

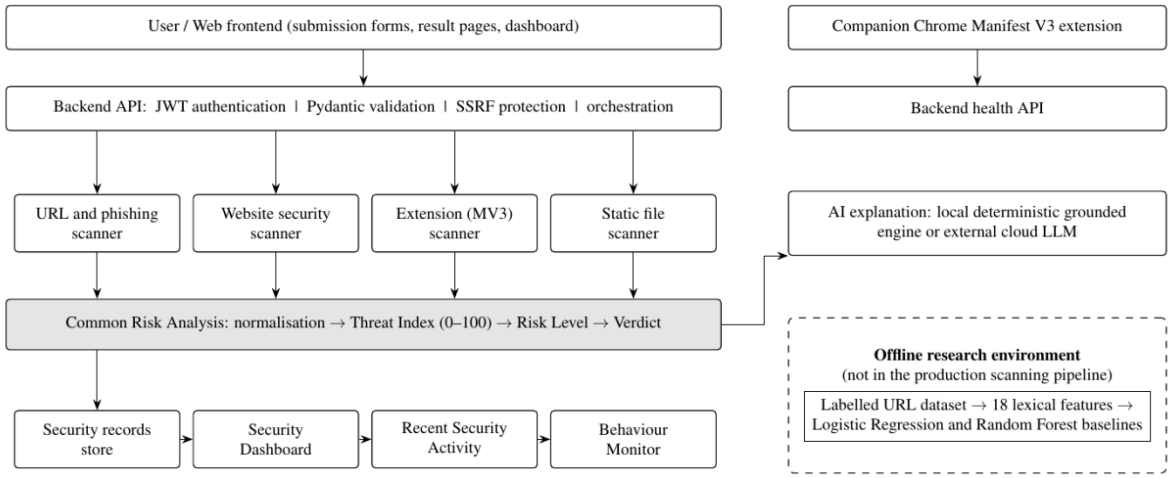


Figure 1. Architecture of CyberWatch. The dashed block is the offline research environment and is not in the production scanning pipeline.

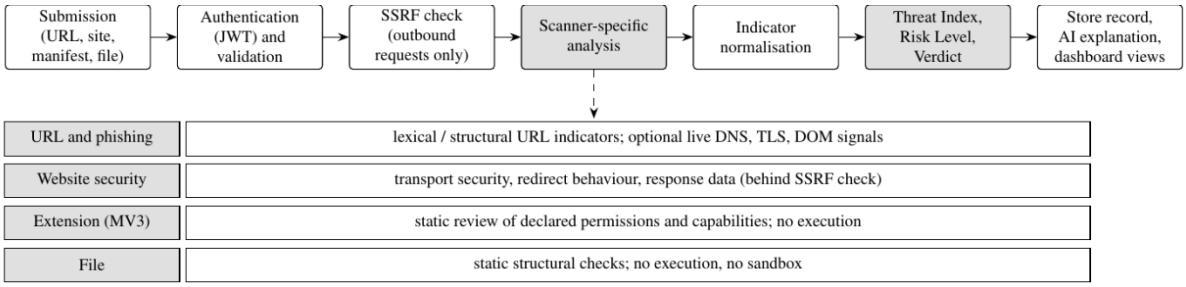


Figure 2. End-to-end CyberWatch workflow and the scanner-specific analysis stage.

4.2. Implementation status

All four scanners, the common risk layer, JWT/Pydantic controls, dashboard views and the local explanation engine are implemented. SSRF protection is implemented with a redirect aggregation gap. The offline LR/RF baselines are offline only. The calibration layer and the full SSRF hardening (per-hop checks, IP pinning) are proposed, not implemented.

4.3. Threat model and scope

The platform handles three kinds of untrusted input: URLs/addresses, uploaded files and extension manifests. The assumed adversary can submit crafted inputs to the API, including addresses that resolve to internal resources; control servers a submitted address points to (including redirects and DNS answers); and supply files and manifests designed to look benign. The adversary has no access to the backend host, database or JWT signing key. Protected assets are the backend's network position, confidentiality of stored records and integrity of reported results. Evasion by an adversary who knows the indicator rules is out of scope, which is a real weakness of any fixed rule set.

5. Methodology

5.1. Scanners

URL and phishing. Evaluates lexical and structural indicators (length, IP-address hostname, subdomain count, scheme, suspicious TLD, punycode, brand keywords, shorteners, unusual ports). Optional live DNS, TLS and DOM signals were disabled in the offline evaluation. **Website security.** Version 2 combines URL checks with observations of the reachable target (transport security, redirects, response data), always behind SSRF protection; the result describes the target at scan time. **Extension.** Statically analyses a Manifest V3 manifest [18] and reports declared capabilities such as broad host access as exposure, not confirmed malicious behaviour. **File.** Validates an uploaded file and checks structure without executing it (e.g. declared type versus real type, entropy, strings); no sandbox is used, and a clean result is a structural assessment, not an antivirus verdict.

Table 3. Threat Index, risk level and verdict mapping.

Threat Index	0–24	25–49	50–74	75–100
Risk Level	Low	Medium	High	Critical
Verdict	Safe	Suspicious	Dangerous	Malicious

5.2. Threat Index: definition and justification

Let x be a validated input to scanner s, $f_{s,i}(x) \in \{0,1\}$ indicate whether indicator i fires, and $w_{s,i} \geq 0$ be its weight. The Threat Index is

$$T_s(x) = \min\{\, 100, \textstyle\sum_i w_{s,i} f_{s,i}(x) \,\}. \tag{1}$$

Hence $0 \leq T_s \leq 100$, T_s never decreases when an indicator fires, and below the cap it equals the sum of the contributions listed to the user, so a result is explained by its indicator list. Levels use thresholds $\tau_1 < \tau_2 < \tau_3$:

$$L(T) = \begin{array}{ll} \text{Low} & T < \tau_1 \\ \text{Medium} & \tau_1 \leq T < \tau_2 \\ \text{High} & \tau_2 \leq T < \tau_3 \\ \text{Critical} & T \geq \tau_3, \end{array} \tag{2}$$

with the verdict (Safe, Suspicious, Dangerous, Malicious) the one-to-one relabelling of the level; the platform uses $(\tau_1, \tau_2, \tau_3) = (25, 50, 75)$ (Table 3).

Why an ordinal score. A bounded additive score is transparent and needs no labelled data, which suits a rule-based scanner whose indicators are explained to non-specialists. It supports ranking within a scanner but not probabilistic statements: T is not $P(\text{malicious} \mid x)$, and equal values from two scanners do not imply equal likelihood of harm.

Weights and thresholds. Weights are expert-chosen design parameters. The scan evidence in Section 6 exposes the per-indicator contributions of individual scans (e.g. the extension scan sums to exactly 78), but the full weight table is not reported in this paper. A validation protocol is proposed (not yet performed): fit an ℓ_2-regularised logistic model $\hat{r}(x) = \sigma(b + \sum_i \beta_i f_i(x))$ on a labelled training set under a domain-grouped split, compare β_i with w_i by rank correlation, and choose τ_k on a validation set before a single evaluation on a held-out test set.

Calibration. A calibrated score satisfies $P(y = 1 \mid \hat{r} = p) = p$. Platt scaling fits $\hat{r} = \sigma(aT + b)$ [28]; isotonic regression fits a monotone step function [29]. Quality is measured by

$$\text{Brier} = (1/N) \textstyle\sum_n (\hat{r}_n - y_n)^2, \qquad \text{ECE} = \sum_{m=1}^{M} (|B_m|/N) \cdot |\, \bar{y}_m - \bar{r}_m \,|, \tag{3}$$

where B_m are score bins [30, 31]. A calibrated value is valid only for the population and prevalence of the calibration data.

Illustrative example (hypothetical weights). With $w_1 = 20$ (non-HTTPS), $w_2 = 25$ (IP hostname) and $w_3 = 40$ (brand look-alike), a URL firing f_1, f_2 gets $T = 45$ (Medium); firing all three gives $\min(100, 85) = 85$ (Critical). Changing w_2 from 25 to 30 would move the first URL to $T = 50$ (High) without any change in evidence, which is why scores are only ordinal. These weights are invented and are not the CyberWatch weights.

5.3. SSRF security architecture

Documented controls. JWT authentication [32] gates scanner and dashboard endpoints, Pydantic models reject malformed requests, and the website scanner checks outbound destinations before a request is made [11]. Functional testing found that the transport layer aborted a redirect to a loopback address, but finding aggregation did not label it SSRF/blocked/private-IP; no evidence is available that this gap is fixed, so it is reported as open.

Required hardening (proposed, not implemented). Table 4 lists the test cases a complete design needs. The principles are: validate every redirect hop; validate resolved A and AAAA records, not the hostname; normalise address encodings; check the embedded IPv4 in IPv4-mapped/NAT64 forms [15]; block special-purpose ranges [14]; and connect to the validated address itself (pinning) against DNS rebinding [13]. Errors should be aggregated by a typed exception carrying reason, hop and address. These controls give no absolute guarantee (parser differentials, proxies and IPv6 zone identifiers remain residual risks) and each case must be run and reported before any claim of mitigation.

Table 4. SSRF security test cases (proposed; only ST-8 is documented).

ID	Case	Expected
ST-1	Private IPv4 (10/8, 172.16/12, 192.168/16)	Blocked
ST-2	127.0.0.1, localhost, ::1	Blocked
ST-3	Link-local/metadata 169.254.169.254; fe80::/10	Blocked
ST-4	Decimal/octal/hex IPv4; IPv4-mapped IPv6	Normalised, blocked
ST-5	Hostname resolving to private; mixed records	Blocked
ST-6	DNS rebinding (public then private)	Blocked by pinning
ST-7	Credentials trick (public@127.0.0.1)	Blocked

ID	Case	Expected
ST-8	Public URL redirecting to loopback	Blocked, labelled SSRF
ST-9	Redirect to file:/gopher:; loops, excess hops	Rejected
ST-10	Unicode/IDN hostname tricks	Normalised, checked

5.4. Security records and AI explanation

Every completed scan is written as one security record (target, domain, timestamp, audit status, Threat Index, risk level, verdict, audit checklist, indicators, recommendations, and producing scanner). All three monitoring views read the same records, so they stay consistent, and records are shown only to the owning authenticated user. Alerting is not implemented. A local deterministic engine produces fixed templates grounded in scanner findings with no network access; an external cloud LLM generates stochastic text. A zero-hallucination claim applies at most to the fixed-template engine by design, not to the external model.

6. Implementation and System Screenshots

CyberWatch is a React/TypeScript single-page frontend (Vite, Tailwind CSS) over a Python FastAPI backend served by Uvicorn, with MongoDB as the record store and a companion Manifest V3 extension. Scan pages move through four states (initial, loading, result, error), and optional response fields are handled defensively. The screenshots below were captured from the running development build on 26 October 2026 and show the real scan evidence behind each result; they are local functional demonstrations, not detection-accuracy results.

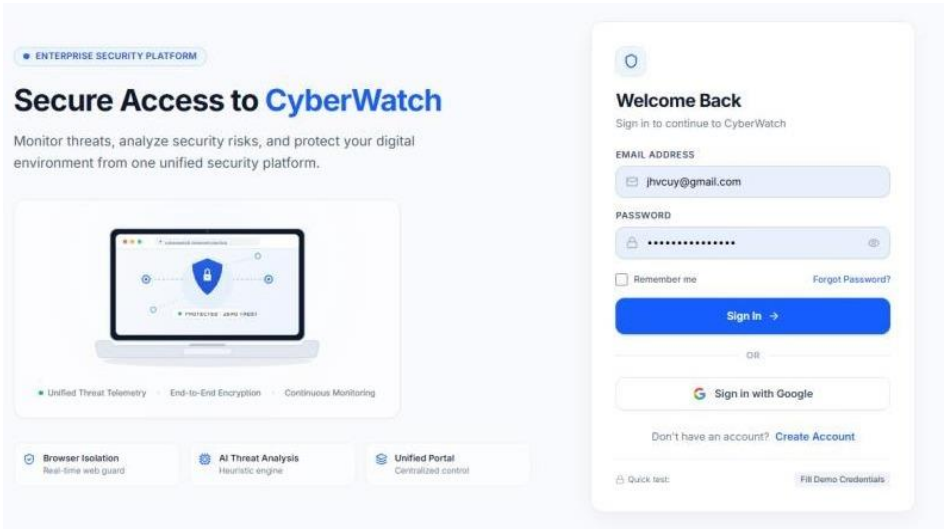


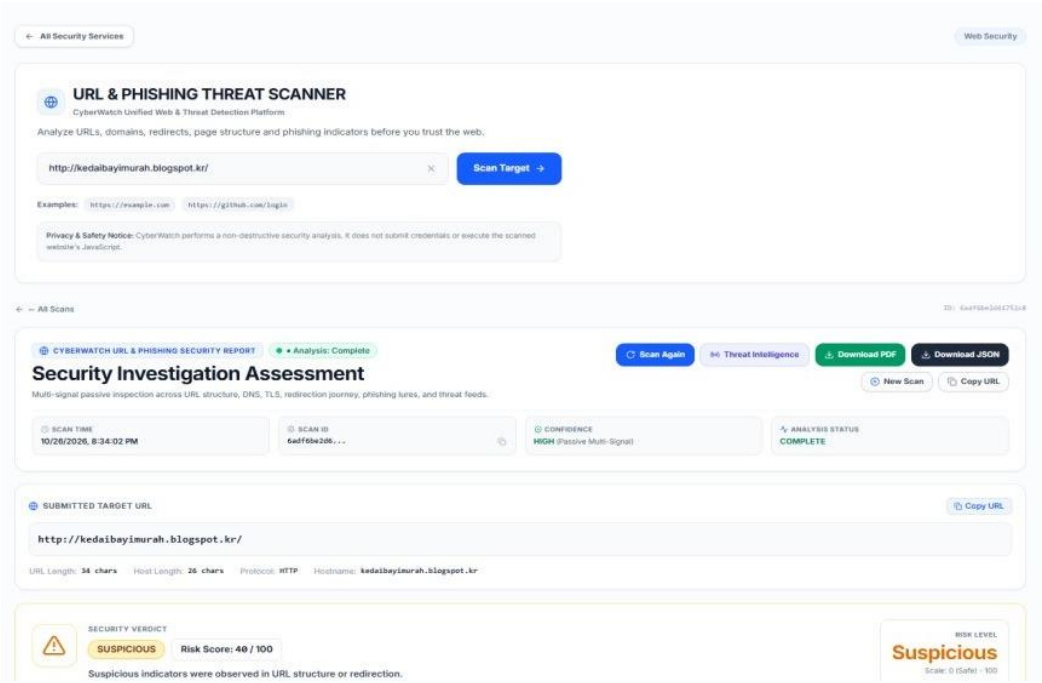
Figure 3. Login page. Access to the scanners and dashboard requires authentication; a JWT is issued at login and attached to every protected request.

URL scanner (Fig. 4a; login in Fig. 3). The submitted target http://kedaibayimurah.blogspot.kr/ (34 characters, host length 26, protocol HTTP) was analysed passively across URL structure, DNS, TLS, redirection and phishing lures. The report shows Risk Score 40/100, risk level and verdict *Suspicious* (Medium band, consistent with Table 3), confidence "High (passive multi-signal)", scan ID 6adf6be2d6..., and actions to rescan, download PDF/JSON and copy the URL. A privacy notice states that the scan is non-destructive and does not submit credentials or execute the target's JavaScript.

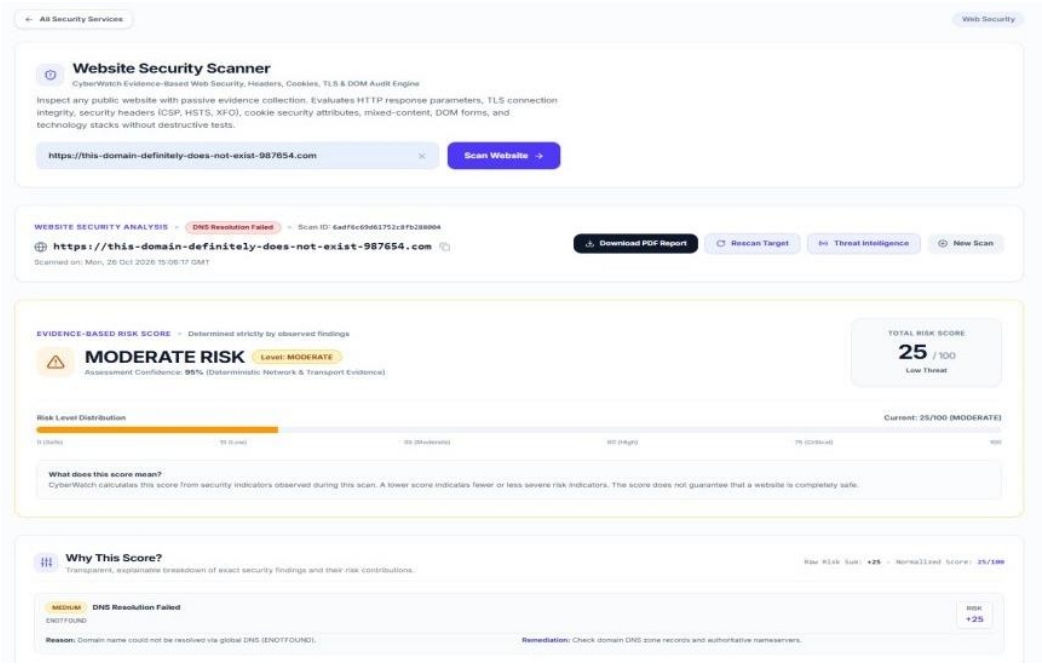
Website scanner (Fig. 4b). A deliberately non-existent domain (this-domain-definitely-does-not-exist-987654.com) was submitted and produced the status *DNS Resolution Failed*. The evidence-based score is 25/100 with a single contributor, "DNS Resolution Failed" (ENOTFOUND, +25, raw sum 25, normalised 25/100), with the stated reason and a remediation hint, and assessment confidence 95% (deterministic network evidence). The page explains that a lower score does not guarantee a site is safe. This page labels the band "Moderate", while the history view labels the same 25/100 result *Suspicious*, matching the 25–49 band.

Extension scanner (Fig. 4c). The uBlock extension (v25.5.1) was assessed from its store listing as *Critical*, 78/100, from 10 evidence-backed contributors: declarative rule-based network filtering (+2), declarative rule execution feedback (+5), browser tab metadata access (+3), passive network request inspection via webRequest (+7), dynamic script injection via scripting (+5), broad website access <all_urls> (+12), dynamic code evaluation eval/Function (+8), remote script injection routine (+12), static cookie-access routines (+6) and a potential sensitive-data exfiltration routine (+18). These sum to exactly 78, matching Eq. (1) below the cap. The banner states that extension code was not executed and that the analysis is a static audit; the score therefore reflects *capability and exposure*, not confirmed malicious behaviour, and a widely used content blocker legitimately needs several of these permissions.

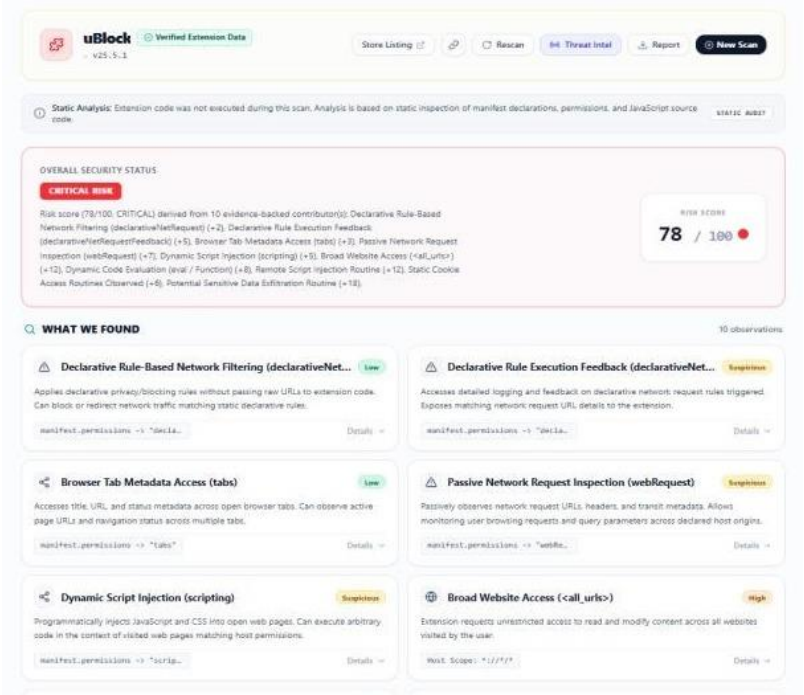
File scanner (Fig. 4d). A sample file (81.6 KB, real type "Generic Binary Data", SHA-256 shown) scored 0/100, *Low risk*, with zero risk contributors. The security summary lists file integrity, file type, extension/format consistency, Shannon entropy (5.324/8.00, moderate) and extracted strings (250 strings, 0 suspicious APIs flagged) as normal. The page carries the assessment note that the score is not a guarantee that the file is safe, and tabs for file details, strings, static analysis, reputation, YARA matches and embedded URLs (1).



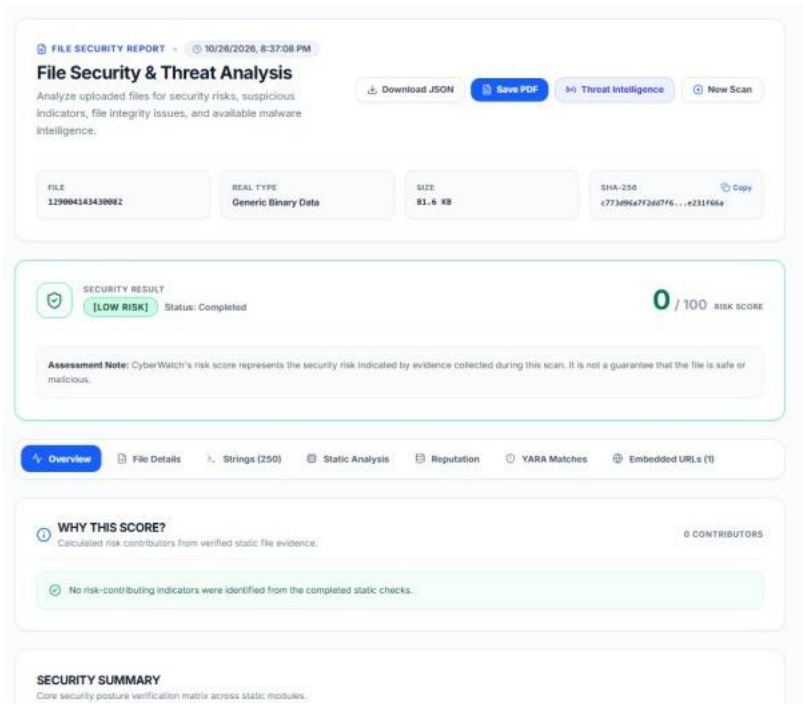
(a) URL & Phishing Threat Scanner: Risk Score 40/100, verdict Suspicious.



(b) Website Security Scanner on a non-existent domain: DNS resolution failed, score 25/100, with the "Why This Score?" breakdown.



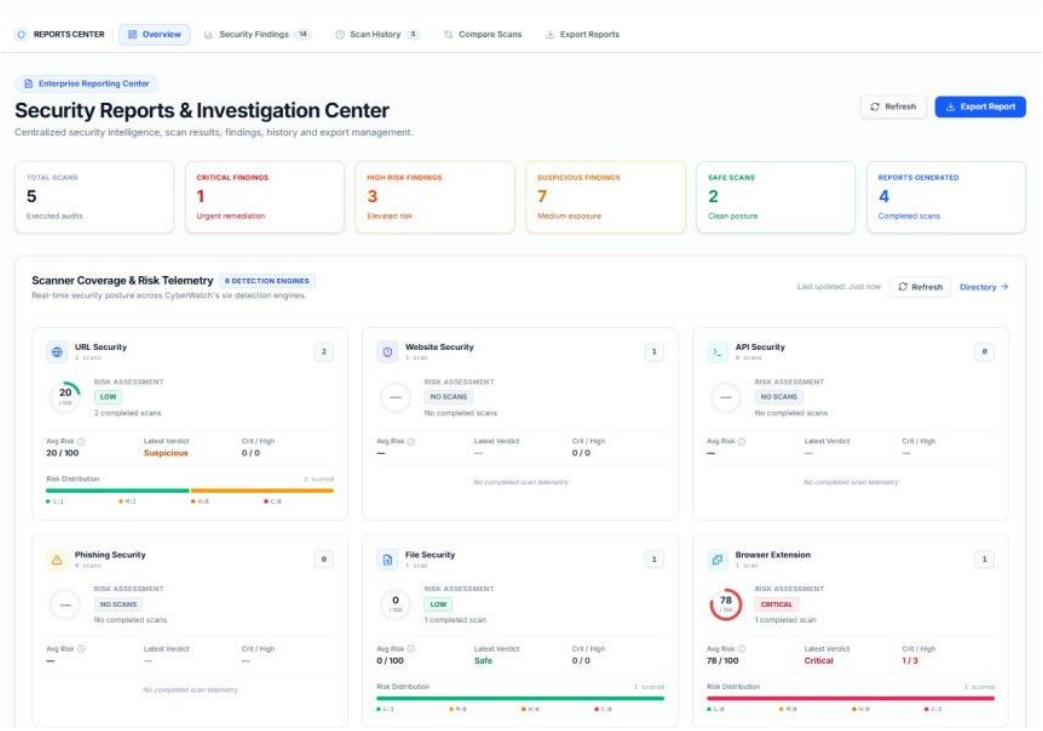
(c) Browser Extension Scanner: uBlock v25.5.1, Critical 78/100, with static observations and per-indicator contributions.



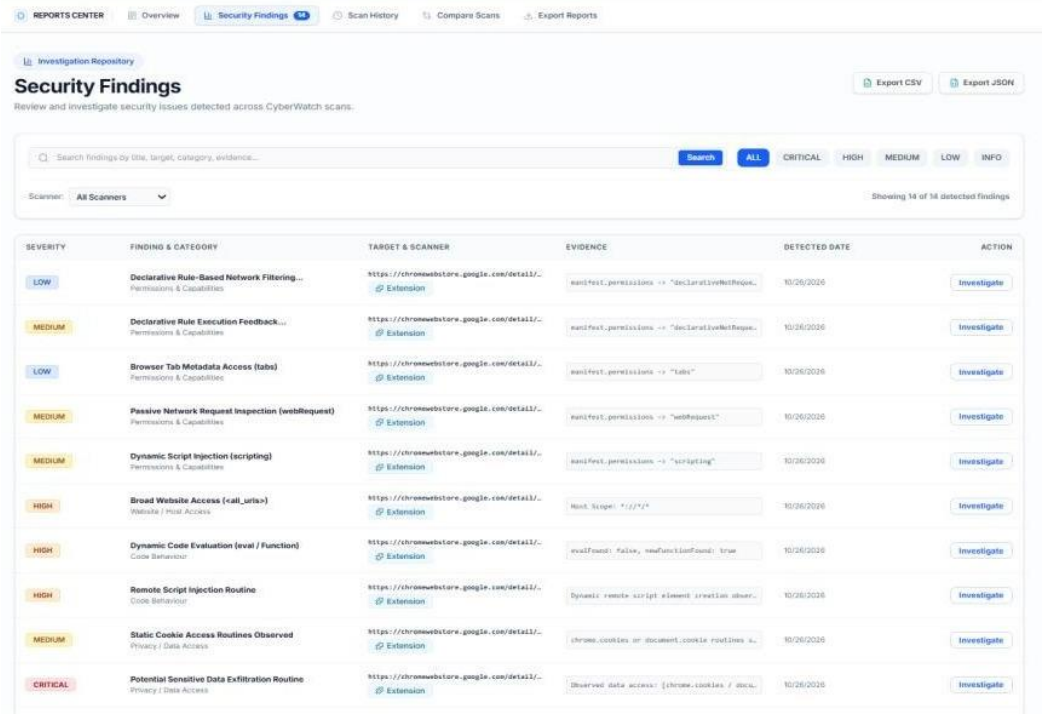
(d) File Security Scanner: 81.6 KB file, 0/100 Low risk, all static checks normal.

Figure 4. Scan result pages of the four CyberWatch scanners (captured from the running development build, 26 October 2026).

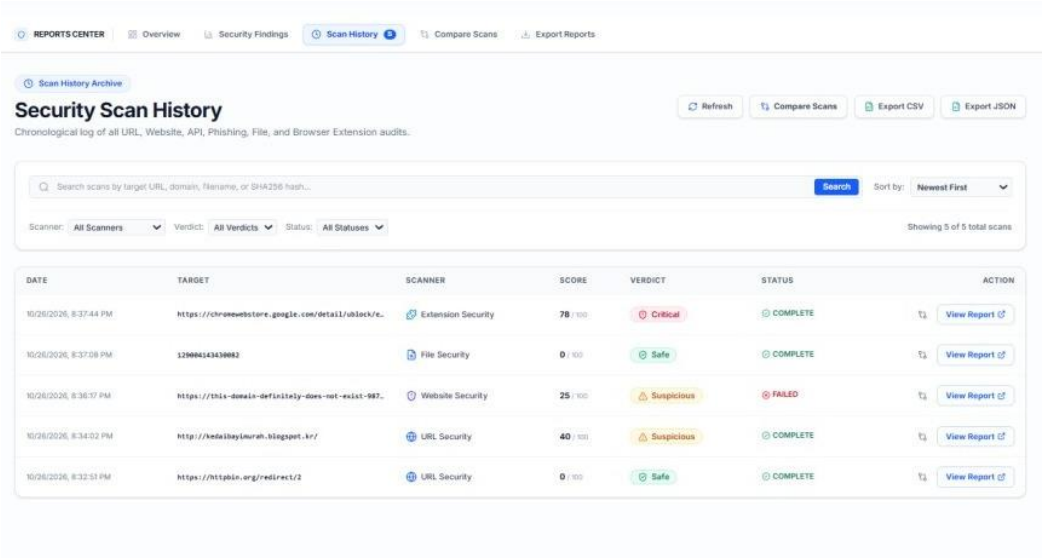
Monitoring views (Fig. 5). The Security Reports & Investigation Center overview aggregates all scans: 5 total scans, 1 critical finding, 3 high-risk findings, 7 suspicious findings, 2 safe scans and 4 reports generated, with one telemetry card per detection engine (URL, website, file and browser extension have scans; the interface also lists API and phishing engine cards with no scans, which are outside the four scanners studied here). The Security Findings page lists 14 findings across scanners with severity, category, target, evidence snippet and date, and supports severity filters, search, and CSV/JSON export. The Scan History records the five scans in time order (Table 5), so a URL, a website, a file and an extension scan made minutes apart appear in one list.



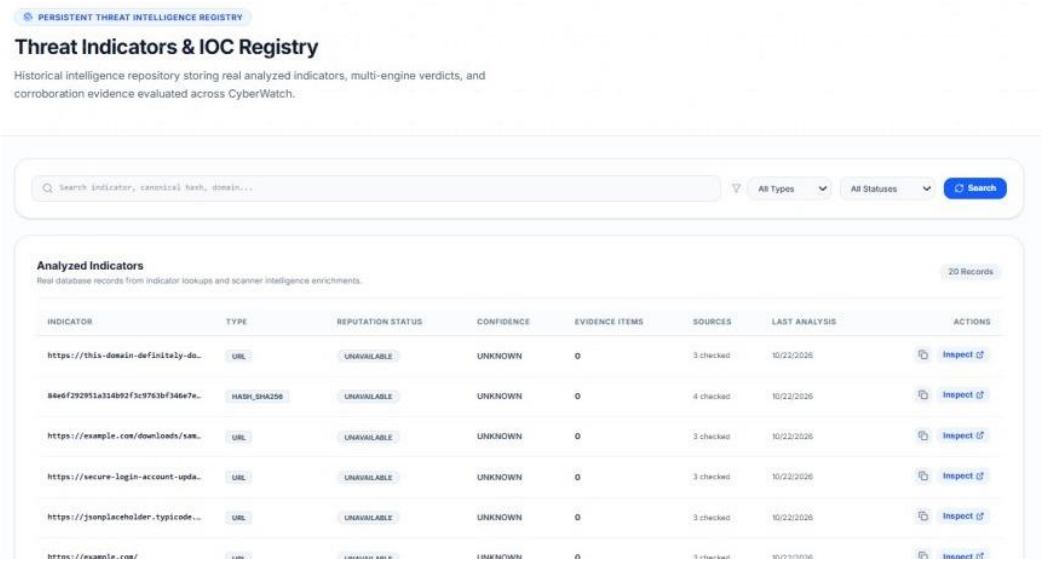
(a) Security Reports & Investigation Center overview with scanner coverage and risk telemetry.



(b) Security Findings: 14 findings with severity, scanner, evidence and detection date.



(c) Security Scan History (Recent Security Activity): five scans across four scanners.



(d) Threat Indicators & IOC Registry: reputation status is "Unavailable" for all indicators (no external threat intelligence).

Figure 5. Monitoring views of CyberWatch built on the shared security records.

Table 5. Scan records shown in the Scan History view (Fig. 5c), 26 Oct 2026 (times PM).

Time	Scanner	Target	Score	Verdict
8:37:44	Extension	uBlock (Web Store listing)	78	Critical
8:37:08	File	81.6 KB sample file	0	Safe
8:36:17	Website	non-existent domain	25	Suspicious∗
8:34:02	URL	kedaibayimurah.blogspot.kr	40	Suspicious
8:32:51	URL	httpbin.org/redirect/2	0	Safe

∗Scan status *Failed* (DNS resolution error); all other scans *Complete*.

The Threat Indicators and IOC Registry lists 20 analysed indicators (URL and SHA-256 types); every entry shows reputation status "Unavailable" and confidence "Unknown" because no external threat-intelligence source is integrated, which is consistent with the stated limitations.

7. Experimental Setup

Experiments ran on a local machine (Ryzen 7 7435HS, 8C/16T, Windows 11, 15.82 GB RAM; Python 3.13.5, Node.js 24.18) with network tests confined

to loopback (127.0.0.1). The following details are not reported and therefore limit exact reproduction: library versions, random seeds, Logistic Regression and Random Forest hyperparameters beyond those stated (RF: 100 trees), the tuning procedure, the load-testing tool and payload, and the external LLM provider and model.

7.1. Dataset provenance and consistency

The audit reports 650,234 raw records; class counts (benign 427,881, defacement 96,062, phishing 93,731, malware 32,520) sum to 650,194, agreeing with 650,234 minus 40 records excluded for invalid types. A later package summary reports 651,191 raw records; the two cannot both be correct. This paper uses 650,234 because the class counts reconcile with it; the discrepancy is recorded as a limitation. Removing 40 invalid, 12,416 normalised duplicates and 9 conflicting-label records gives 637,769, whereas the report gives 637,770 for the clean pool (a minor second inconsistency). The audit records a SHA-256 hash of the dataset (a500a136d455...1317d) identifies the exact file used. Constituent feeds include ISCX-URL-2016 [8], Malware Domain Blacklist, a safe-web repository, PhishTank and PhishStorm under a CC0 1.0 compilation licence; users of the dataset should check the licence terms of each constituent feed independently.

Sampling and splitting. The balanced sample keeps 1,000 benign and 1,000 threat URLs (defacement 432, phishing 422, malware 146), whose threat mix (43.2%, 42.2%, 14.6%) equals the raw mix. Splitting was by eTLD+1 domain group with zero shared domains between train (1,600 URLs, 1,225 groups) and test (400 URLs, 318 groups; 188 benign / 212 threats), preventing domain leakage. Balancing makes prevalence 50% against 34.2% threats in the raw data and a far lower rate in real traffic, so accuracy and precision do not transfer; at 5% prevalence the Random Forest rates (TPR 83.96%, FPR 11.70%) would give precision of about 27%, if the rates transferred at all. Other biases remain: benign and threat URLs come from different feeds so feed style can act as a shortcut, the data are not time-split, and group splitting does not control feed or temporal leakage [24].

7.2. Features, models and metrics

Eighteen lexical/structural features were extracted without network calls: URL, hostname and path length; counts of dots, hyphens, @, ?, =, slashes and digits; subdomain count; HTTPS flag; IP-address flag; URL and host entropy; suspicious-TLD, punycode and brand-keyword flags. A StandardScaler was fitted on the training split only; Logistic Regression (ℓ_2) and Random Forest (100 trees, Gini) were evaluated at probability threshold 0.50. Fig. 6 shows the workflow. These are offline baselines and are not deployed. With threats as the positive class we report accuracy, precision, recall, F_1 and FPR; because raw predictions are unavailable, confidence intervals are 95% Wilson score intervals [33] from reported counts and are optimistic since URLs in a domain group are not independent. Paired tests (McNemar [34], DeLong [35]) need per-sample predictions and are listed as required work.

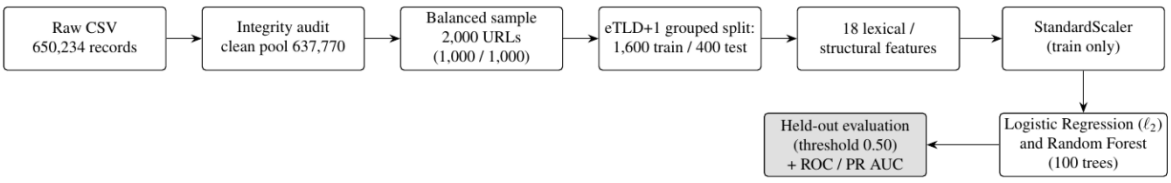


Figure 6. Offline machine-learning experimental workflow (not part of the production scanning pipeline).

8. Results

8.1. Functional testing

The functional test campaign executed 352 automated tests across 21 test files and 11 subsystems: 350 passed, 2 failed, none blocked (pass rate 99.43%). One failure is the redirect-related SSRF aggregation gap; the other is a test against postman-echo.com that exceeded its 20-second limit, attributed to external latency or rate limiting. A 99.43% functional pass rate is not 99.43% detection accuracy. A separate QA summary (128 tests, 98.4%, scope not documented) is not combined with this result.

8.2. Offline machine-learning baselines

Table 6 and Fig. 7 give held-out results on 400 URLs. Random Forest is better on every reported metric and its accuracy interval does not overlap that of Logistic Regression, although a paired test is still required; it also reduces both error types (false positives 29 to 22, false negatives 82 to 34). With only 23 malware test URLs, the malware recall of 73.91% has a 95% interval of 53.5–87.5%, so no ranking of the models on malware is supported (Table 7). The phishing gap is large relative to sampling uncertainty (RF 73.4–87.8% versus LR 41.2–59.7%). Random Forest importances are led by path length (16.91%), slash count (16.28%), dot count (10.31%), subdomain count (10.05%) and URL entropy (7.48%); because path-structure features dominate, part of the performance may reflect dataset artefacts such as deep paths on compromised defacement sites, and validation on an independent time-split dataset is needed.

Table 6. Held-out results on 400 URLs (95% Wilson CI where computed).

Metric	Log. Reg.	Rand. Forest
TP/TN/FP/FN	130/159/29/82	178/166/22/34
Accuracy	72.25% (67.7–76.4)	86.00% (82.3–89.1)
Precision	81.76%	89.00%

Metric	Log. Reg.	Rand. Forest
Recall	61.32% (54.6–67.6)	83.96% (78.4–88.3)
F1-score	70.08%	86.41%
FPR / FNR	15.43% / 38.68%	11.70% / 16.04%
ROC-AUC / PR-AUC	0.8392 / 0.8414	0.9378 / 0.9422

Table 7. Threat-class recall on the test split.

Class	N	Heuristic	LR	RF
Defacement	80	0.00%	72.50%	90.00%
Phishing	109	0.00%	50.46%	81.65%
Malware	23	0.00%	73.91%	73.91%
All threats	212	0.00%	61.32%	83.96%

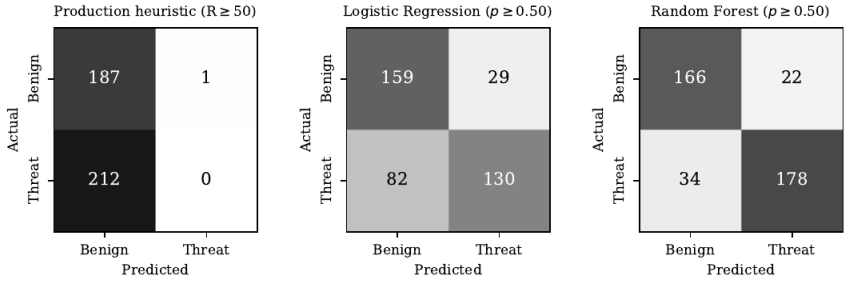


Figure 7. Confusion matrices on the 400-URL held-out set for the production heuristic, Logistic Regression and Random Forest.

8.3. Production heuristic: evaluation and failure analysis

Evaluated offline on the same 400 URLs with live DNS, TLS and DOM signals disabled, at a frozen threshold of $R \geq 50$, the production heuristic produced TP = 0, TN = 187, FP = 1, FN = 212: accuracy 46.75% (41.9–51.6), precision 0%, recall 0% (upper 95% bound 1.8%) and FPR 0.53%. This is not a full live-production benchmark and must not be read as the performance of the offline models. Every threat scored below 50 on this feature set. Score distributions were not reported, so it is unknown whether scores separate the classes at a lower threshold; computing ROC-AUC for the heuristic is the first required step. Candidate explanations (untested): (a) the threshold of 50 coincides with the High band and may require several indicators to fire together, which lexical-only URLs rarely do; (b) the indicators target classic phishing cues (IP hosts, shorteners, brand look-alikes) that are rare where 43% of threats are defacement; (c) the weights are expert-set, possibly balanced assuming live signals, and never fitted to data; (d) the Random Forest relies on path length and slash count, which the indicator list does not capture directly; (e) the offline harness could differ from the production path. None has been tested; the experimental plan to do so is in Table 8.

Table 8. Required experiment to improve and validate the heuristic (proposed; no results claimed).

Step	Procedure	Output
1	Export heuristic scores for all URLs; score histograms by class	ROC-AUC, PR-AUC
2	Choose R by group-aware CV on training data only; freeze	Threshold
3	Single evaluation on the untouched test split	Metrics table
4	Indicator ablation; refit weights as ℓ_2 logistic model; add path-depth features	Weight table
5	Add live DNS/TLS/DOM/reputation on a recent, time-aware split	Live-signal results
6	Compare with LR/RF on identical splits (McNemar, DeLong, group bootstrap)	p-values, CIs
7	Repeat on an independent dataset	External validity

8.4. Synthetic calibration study

The study was run in simulation: four synthetic vectors (URL, website, extension, file) with 10, 12, 8 and 6 binary indicators and malicious prevalence 0.25, 0.15, 0.05 and 0.30; indicator rates drawn from assumed ranges; correlation 0.3; 30 independent worlds, each fitting calibrators on 2,000 items per

vector and testing on 20,000 (parameters as in the authors' earlier simulation study). The raw index is poorly calibrated (macro ECE 0.158); Platt scaling and isotonic regression reach 0.015 and 0.011 and a learned logistic model 0.010 (Fig. 8). The assumptions are strong and prior shift at deployment would degrade calibration. The study shows calibration is feasible in principle; it is not evidence about real phishing or malware detection, and the calibration layer is not part of the platform.

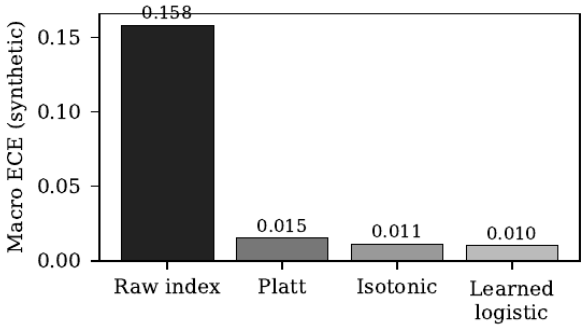


Figure 8. Macro expected calibration error in the synthetic study (30 worlds).

8.5. Performance and load testing

Each module ran five warm-up and 35 measured iterations using monotonic timers on the loopback interface (Table 9, Fig. 9). With 35 samples, P95 is effectively the second-largest value, so tails are coarse. The rule-based scanners respond in well under a millisecond; 1 MB file analysis is the costliest local operation (27.76 ms mean, roughly 36 MB/s). The local deterministic engine is in-memory, whereas the external LLM needs a TLS round trip and generation, making its mean latency more than four orders of magnitude higher. Timings recorded by the functional test harness include mocking and assertion overhead and are not comparable.

Table 9. Subsystem latency (35 measured iterations).

Component	Mean (ms)	P95 (ms)
URL scanner	0.0836	0.1770
Website scanner	0.1237	0.3696
Extension scanner	0.1612	0.3020
1 MB file analysis	27.7631	34.9311
Health API	0.5605	0.9208
Local deterministic AI	0.0181	0.0255
External cloud LLM	1226.1243	1534.2583

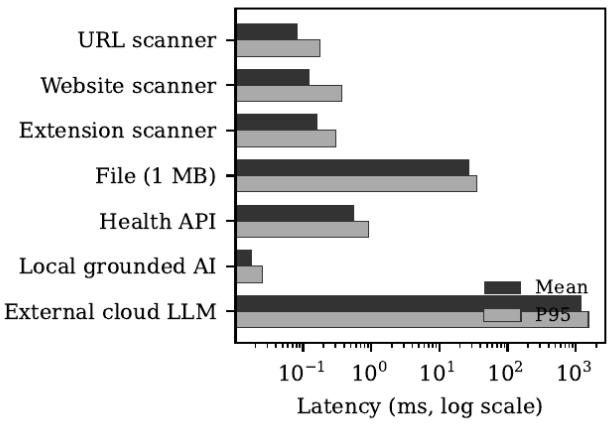


Figure 9. Mean and P95 latency per component (logarithmic scale).

Concurrency levels of 1, 5, 10, 25 and 50 virtual users issued 100, 150, 200, 250 and 300 requests (1,000 total); all completed and none failed (Table 10, Fig. 10). Throughput peaked at 986.85 req/s at C25 and fell to 946.78 at C50 while mean latency rose roughly in proportion to concurrency; throughput times mean latency is about 25 at C25 and 50 at C50, matching the user count, which is consistent with a closed-loop generator and a server near saturation. Peak CPU was 93.4%, resting CPU about 1.5%, and resident memory rose from 93.73 to 119.05 MB. Limitations: one machine, a load generator probably competing for CPU, short runs, an unspecified endpoint mix and no network latency; results are not a production capacity guarantee.

Table 10. Local load-testing results.

Users	Req/s	Mean (ms)	P95 (ms)	RAM (MB)
C1	656.68	1.51	2.23	99.06
C5	760.72	6.54	10.52	105.59
C10	948.86	10.50	15.40	110.95
C25	986.85	25.20	35.38	113.17
C50	946.78	52.39	72.61	119.05

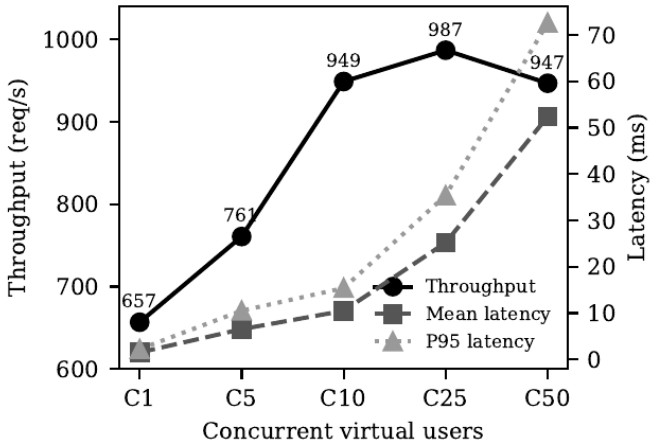


Figure 10. Throughput, mean latency and P95 latency versus concurrency.

9. Discussion and Threats to Validity

Table 11 separates what each evaluation scope shows. The strengths are a coherent modular design, a common output vocabulary, a working interface (Section 6), a high functional pass rate and low local latency. The main weakness is detection effectiveness of the production heuristic: under offline lexical-only conditions it missed every threat, whereas a Random Forest on the same 18 features detected 83.96%. This suggests that scoring weights, the frozen threshold or missing structural indicators, not the information alone, limited the heuristic, though the cause is undiagnosed. The screenshots illustrate the explainability benefit (every score decomposes into listed indicators) but also its risk: a capability-based extension score (78/100 for uBlock) must not be read as a malice finding.

Table 11. What each evaluation scope measures.

Scope	Key result	Does not show
Functional	350/352 passed	detection accuracy
ML baselines	RF 86.00%, LR 72.25%	real-traffic accuracy
Heuristic	recall 0%, $R \geq 50$	live-signal performance
Calibration	ECE 0.158→0.010	real-world accuracy
Latency, load	986.85 req/s peak	production capacity
Screenshots	scan evidence, 5 scans	accuracy; they are demonstrations

Internal validity. Group-wise splitting prevents domain leakage, but feed-specific URL style, surviving duplicates and possible tuning on the test set (not documented) could inflate baseline scores; the heuristic threshold was fixed beforehand, which avoids tuning bias but may be a poor operating point. **Construct validity.** "Malicious" covers defacement, phishing and malware, which are different phenomena; functional pass rate and detection accuracy measure different constructs. **External validity.** The test set has 400 URLs, 50% threats, from public feeds of limited age; results need not hold for current traffic, other languages, shortened or obfuscated links, or the website, extension and file vectors, which have no labelled evaluation. **Conclusion**

validity. Intervals ignore domain clustering and are optimistic; no paired test was possible; P95/P99 over 35 samples are coarse.

10. Limitations, Ethics and Future Scope

Limitations. The heuristic was evaluated only offline without live signals and its zero recall is unresolved; weights are fixed and not fully reported; the dataset is a 2,000-record balanced sample (146 malware records, 23 malware test URLs) without a time split; raw predictions, seeds and some hyperparameters are not available, so ROC curves, paired tests and exact reproduction are not yet possible; the SSRF redirect gap is open and DNS-rebinding protection is not claimed; calibration is synthetic; performance was measured on one machine; several reported values (raw record count, later-stage accuracies, test-suite size) are inconsistent across project documents and are not combined; and extension and file results describe exposure and structure, not confirmed behaviour. **Ethics.** All URLs come from public datasets, no user data were collected, and negative results are reported in full. Results should not be used to claim the platform protects users.

Future scope (proposed). Complete the experiment of Table 8; implement and test the controls of Table 4; calibrate the Threat Index on real labelled data; evaluate learned models against the heuristic; add reputation sources (the IOC registry in Fig. 5d currently reports none), sandboxed analysis and explainability evaluation [36]; and load-test a deployed multi-user system.

11. Conclusion

CyberWatch integrates URL, website, Manifest V3 extension and static file scanners behind a common Threat Index, risk level and verdict, with JWT, Pydantic and SSRF controls at the API boundary and a working monitoring interface. Functional testing passed 350 of 352 tests and local benchmarks showed low latency and error-free local load runs. Offline baselines reached 86.00% (Random Forest) and 72.25% (Logistic Regression) held-out accuracy, but the production heuristic detected no threats under offline lexical-only conditions, the SSRF redirect gap is open, the dataset record counts are inconsistent across project documents, and calibration is so far only synthetic. The contribution is an integrated and transparently evaluated platform together with a concrete protocol for validating its detection, security and scoring claims, not a demonstrated detection advantage.

Data and code availability. The authors' earlier manuscript refers to a public repository for the platform. Exact reproduction of the offline experiments additionally requires the dataset audit scripts, the split manifest and per-sample prediction files, which are not part of the material evaluated in this paper.

REFERENCES

[1] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in Proc. 15th ACM SIGKDD, Paris, 2009, pp. 1245–1254.

[2] O. K. Sahingoz, E. Buber, O. Demir, B. Diri, "Machine learning based phishing detection from URLs," Expert Syst. Appl., vol. 117, pp. 345–357, 2019.

[3] A. Kapravelos et al., "Hulk: Eliciting malicious behavior in browser extensions," in Proc. 23rd USENIX Security Symp., 2014, pp. 641–654.

[4] N. Jagpal et al., "Trends and lessons from three years fighting malicious extensions," in Proc. 24th USENIX Security Symp., 2015, pp. 579–593.

[5] A. Nayak, R. Khandelwal, E. Fernandes, K. Fawaz, "Experimental security analysis of sensitive data access by browser extensions," in Proc. ACM Web Conf. (WWW '24), 2024, doi: 10.1145/3589334.3645683.

[6] A. Shalaginov, S. Banin, A. Dehghantanha, K. Franke, "Machine learning aided static malware analysis: A survey and tutorial," arXiv:1808.01201, 2018.

[7] D. Sahoo, C. Liu, S. C. H. Hoi, "Malicious URL detection using machine learning: A survey," arXiv:1701.07179, 2017.

[8] M. S. I. Mamun et al., "Detecting malicious URLs using lexical analysis," in Network and System Security, LNCS 9955, Springer, 2016, pp. 467–482.

[9] C. Whittaker, B. Ryner, M. Nazif, "Large-scale automatic classification of phishing pages," in Proc. 17th Network and Distributed System Security Symp. (NDSS), 2010.

[10] H. Le, Q. Pham, D. Sahoo, S. C. H. Hoi, "URLNet: Learning a URL representation with deep learning for malicious URL detection," arXiv:1802.03162, 2018.

[11] OWASP Foundation, "Server-Side Request Forgery Prevention Cheat Sheet." [Online]. Available: https://cheatsheetseries.owasp.org/

[12] B. Jabiyev, O. Mirzaei, A. Kharraz, E. Kirda, "Preventing server-side request forgery attacks," in Proc. 36th ACM/SIGAPP SAC, 2021, pp. 1626–1635.

[13] C. Jackson, A. Barth, A. Bortz, W. Shao, D. Boneh, "Protecting browsers from DNS rebinding attacks," in Proc. 14th ACM CCS, 2007.

[14] M. Cotton, L. Vegoda, R. Bonica, B. Haberman, "Special-purpose IP address registries," RFC 6890, IETF, Apr. 2013.

[15] R. Hinden, S. Deering, "IP version 6 addressing architecture," RFC 4291, IETF, Feb. 2006.

[16] N. Carlini, A. P. Felt, D. Wagner, "An evaluation of the Google Chrome extension security architecture," in Proc. 21st USENIX Security Symp., 2012.

[17] N. Pantelaios, N. Nikiforakis, A. Kapravelos, "You've changed: Detecting malicious browser extensions through their update deltas," in Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS), 2020.

[18] Google, "Chrome extensions: Manifest V3," Chrome for Developers. [Online]. Available: https://developer.chrome.com/docs/extensions/

[19] N. Pantelaios, A. Kapravelos, "Manifest V3 unveiled: Navigating the new era of browser extensions," arXiv:2404.08310, 2024.

[20] R. Lyda, J. Hamrock, "Using entropy analysis to find encrypted and packed malware," IEEE Security & Privacy, vol. 5, no. 2, 2007.

[21] H. S. Anderson, P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," arXiv:1804.04637, 2018.

[22] E. Raff et al., "Malware detection by eating a whole EXE," arXiv:1710.09435, 2017.

[23] R. Sommer, V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in Proc. IEEE Symp. Security and Privacy, 2010.

[24] D. Arp et al., "Dos and don'ts of machine learning in computer security," in Proc. 31st USENIX Security Symp., 2022.

[25] FIRST.Org, "Common Vulnerability Scoring System version 4.0: Specification document." [Online]. Available: https://www.first.org/cvss/

[26] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, "On calibration of modern neural networks," in Proc. 34th ICML, PMLR vol. 70, 2017, pp. 1321–1330.

[27] A. Niculescu-Mizil, R. Caruana, "Predicting good probabilities with supervised learning," in Proc. 22nd Int. Conf. Machine Learning (ICML), 2005.

[28] J. C. Platt, "Probabilistic outputs for support vector machines," in Advances in Large Margin Classifiers, MIT Press, 1999, pp. 61–74.

[29] B. Zadrozny, C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in Proc. 8th ACM SIGKDD, 2002, pp. 694–699.

[30] G. W. Brier, "Verification of forecasts expressed in terms of probability," Mon. Weather Rev., vol. 78, no. 1, pp. 1–3, 1950.

[31] M. P. Naeini, G. F. Cooper, M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in Proc. AAAI, vol. 29, 2015, pp. 2901–2907.

[32] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)," RFC 7519, IETF, May 2015.

[33] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," J. Amer. Statist. Assoc., vol. 22, no. 158, pp. 209–212, 1927.

[34] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," Psychometrika, vol. 12, no. 2, pp. 153–157, 1947.

[35] E. R. DeLong, D. M. DeLong, D. L. Clarke-Pearson, "Comparing the areas under two or more correlated receiver operating characteristic curves," Biometrics, vol. 44, no. 3, pp. 837–845, 1988.

[36] N. Capuano, G. Fenza, V. Loia, C. Stanzione, "Explainable artificial intelligence in cybersecurity: A survey," IEEE Access, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3204171.