НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2026 Том 26 № 3 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2026 Vol. 26 No 3 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2026-26-3-504-516

Weakly supervised web technology identification with rule-augmented machine learning

Rand Deeb[1], Alisa A. Vorobeva[2]✉

[1,2] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] rand.sec96@gmail.com, https://orcid.org/0009-0006-2730-1550
[2] vorobeva@itmo.ru✉, https://orcid.org/0000-0001-6691-6167

Abstract

Identifying the technologies used by web applications is crucial for vulnerability assessment, resource inventory, and research analysis of the digital environment. Traditional tools that rely on static signature rules suffer from fundamental limitations: they require constant manual maintenance, are sensitive to code obfuscation and dynamic loading, and perform poorly when distinguishing features are subtle, distorted, or noisy. The novelty of this work lies in applying a weak supervision method based on behavioral characteristics to create a robust and scalable classifier. This work presents a hybrid approach combining weakly supervised machine learning with rule-based post-processing. Two independent signature-based systems, Wappalyzer and BuiltWith, serve as sources of weakly labeled data; a final positive label for a technology is assigned only when their detections agree. A multi-label Random Forest model is trained on this labeled data. The feature space is constructed from behavioral patterns extracted during an automated browser session, including Hypertext Transfer Protocol headers, cookies, network requests, and structural features of the Hypertext Markup Language and the Document Object Model. To correct model predictions by accounting for typical technology co-occurrences, a post-processing step based on association rules mined from the training data is applied. The experimental evaluation was conducted on a dataset of 8,594 websites for identifying 122 technologies. The Random Forest model demonstrated consistently high precision and recall for common web technologies, such as web servers, content management systems, and front-end libraries. The hybrid approach with post-processing provided a statistically significant improvement in both micro-averaged and weighted harmonic mean of precision and recall, while preserving the results interpretability. On a fixed test set (20 % of the data), the Random Forest model achieved a micro-averaged harmonic mean of precision and recall of 0.750, while the hybrid approach reached 0.763. Association rules affected 23.8 % of the predictions, moderately improving recall. For widespread technologies (Apache, Nginx, WordPress, jQuery), the harmonic mean of precision and recall exceeded 0.840, and for several components (Drupal, Google Cloud) it reached values of 0.960–1.000. The obtained results show that the approach based on weak supervision and behavioral analysis can provide a scalable method for web technology identification. It effectively complements traditional signature-based methods in cases of deliberate code concealment or dynamic execution. This method is promising for automated auditing, enhancing security tool capabilities, and scalable research of the web technology landscape. Future work may focus on improving the feature set and adapting the system for detecting novel and unique technologies.

Keywords

web technologies identification, weak supervision, machine learning, cybersecurity, behavioral analysis, Random Forest, association rules

For citation: Deeb R., Vorobeva A.A. Weakly supervised web technology identification with rule-augmented machine learning. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2026, vol. 26, no. 3, pp. 504–516. doi: 10.17586/2226-1494-2026-26-3-504-516

УДК 004.056

Слабо контролируемое распознавание веб-технологий с применением машинного обучения, усиленного правилами

Ранд Дееб[1], Алиса Андреевна Воробьева[2]✉

[1,2] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] rand.sec96@gmail.com, https://orcid.org/0009-0006-2730-1550

[2] vorobeva@itmo.ru✉, https://orcid.org/0000-0001-6691-6167

Аннотация

Введение. Задача идентификации технологий, используемых веб-приложениями, является ключевой для анализа уязвимостей, инвентаризации ресурсов и исследовательского анализа среды. Традиционные инструменты, полагающиеся на статические сигнатурные правила, обладают фундаментальными ограничениями: они требуют постоянного ручного сопровождения, чувствительны к обфускации и динамической загрузке, а также недостаточно эффективны, если признаки выражены слабо, искажены или зашумлены. В работе предлагается применение метода слабого обучения на основе поведенческих характеристик для создания надежного и расширяемого классификатора. **Метод.** Представлен гибридный подход, сочетающий слабо контролируемое машинное обучение с постобработкой на основе правил. В качестве источника слабо размеченных данных используются две независимые сигнатурные системы Wappalyzer и BuiltWith; итоговая метка о наличии технологии присваивается только при совпадении их решений. На размеченных данных обучается многоклассовая модель случайного леса. Признаковое пространство формируется на основе поведенческих паттернов, извлекаемых из сессии автоматизированного браузера: заголовки протокола передачи гипертекста, куки, сетевые запросы и структурные особенности языка разметки гипертекста и объектной модели документа. Для коррекции предсказаний модели и учета типичных комбинаций технологий применяется постобработка с помощью ассоциативных правил, выведенных из обучающей выборки. **Основные результаты.** Экспериментальная оценка проведена на выборке данных 8594 сайтов для распознавания 122 технологий. Модель продемонстрировала стабильно высокую точность и полноту для распространенных веб-технологий: веб-серверов, систем управления контентом и фронтендом библиотек. Гибридный подход с постобработкой обеспечил статистически значимое улучшение микро- и взвешенной меры гармонического среднего точности и полноты, сохраняя интерпретируемость результата. На фиксированном тестовом наборе (20 % данных) модель случайного леса достигла микромеры гармонического среднего точности и полноты 0,750, а гибридный подход — 0,763. Ассоциативные правила затрагивали 23,8 % предсказаний, умеренно улучшая полноту. Для распространенных технологий (Apache, Nginx, WordPress, jQuery) мера гармонического среднего точности и полноты превысила 0,840, а для ряда компонентов (Drupal, Google Cloud) достигла значений 0,960–1,000. **Обсуждение.** Полученные результаты показали, что подход на основе слабого обучения и поведенческого анализа может стать масштабируемым способом идентификации веб-технологий. Он эффективно дополняет традиционные сигнатурные методы при намеренном сокрытии кода или его динамическом выполнении. Представленный метод перспективен для автоматического аудита, расширения возможностей инструментов безопасности и масштабируемого исследования технологий веб-пространства. Дальнейшая работа может быть направлена на улучшение набора признаков и адаптацию системы для обнаружения новых и уникальных технологий.

Ключевые слова

идентификация веб-технологий, слабо контролируемое обучение, машинное обучение, кибербезопасность, поведенческий анализ, случайный лес, ассоциативные правила

Ссылка для цитирования: Дееб Р., Воробьева А.А. Слабо контролируемое распознавание веб-технологий с применением машинного обучения, усиленного правилами // Научно-технический вестник информационных технологий, механики и оптики. 2026. Т. 26, № 3. С. 504–516 (на англ. яз.). doi: 10.17586/2226-1494-2026-26-3-504-516

Introduction

Modern websites are assembled from heterogeneous components: a server stack and hosting platform, a backend runtime, a Content Management System (CMS) and its plug-in ecosystem, client-side frameworks and libraries, and third-party services such as analytics tags or Content Delivery Networks (CDNs). From a security and measurement perspective, the practical question is often not “what is the source code?”, but which technologies are externally observable and likely deployed, because this determines the exposed attack surface and guides triage during reconnaissance and inventory. In the Open Web Application Security Project (OWASP) Web Security Testing Guide[1], technology and server identification is explicitly described as an information-gathering step for web assessment. To avoid confusion with browser/user fingerprinting (tracking a user across sites), throughout this paper we use the term “web technology identification” to denote the OWASP-style task of inferring deployed technologies from observable web artifacts.

[1] OWASP. Fingerprint web application framework. OWASP Web Security Testing Guide, 2023. Available at: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework (accessed: 04.05.2026).

Knowing the deployed stack enables concrete security workflows. It allows for Common Vulnerabilities and Exposures triage and targeted scanning by narrowing the search space to likely vulnerable components, such as CMS cores and plug-in ecosystems, server platforms, and dependencies related to Transport Layer Security. It also helps prioritize follow-up checks when multiple technologies co-exist, for instance, a CMS combined with the commerce plug-in and a caching layer. Additionally, it facilitates maintaining a resource inventory for large estates where manual verification is infeasible. The challenge, however, is that modern deployments increasingly hide or rewrite classic markers. This is often done via CDNs, reverse proxies, minification and bundling, and late-loaded resources, which weakens purely static signatures [1, 2].

Signature-based tools, such as Wappalyzer[1], map technologies to hand-crafted fingerprints over Hypertext Transfer Protocol (HTTP) responses, and page artifacts. While widely used, this approach faces persistent limitations:

— Sensitivity to dynamic loading and obfuscation [1, 2];
— A non-trivial maintenance burdens as ecosystems evolve [3];
— Weak or indirect traces for some components which lower recall or produce brittle matches [4].

This work evaluates a pragmatic, measurement-oriented alternative based on weak supervision [5]. The goal is not to propose a new learning algorithm, but to test whether noisy labels bootstrapped from existing detectors can support a multi-label classifier that predicts technologies from behavioral and structural signals collected in live browsing sessions. We focus on externally observable evidence (headers/cookies, network requests, and Hypertext Markup Language (HTML) or Document Object Model (DOM) patterns) and report performance with respect to weak labels.

Rather than treating any single rule engine as ground truth, we use two independent detectors Wappalyzer and BuiltWith[2] only as noisy label sources. For each website and technology, we keep a positive label when both detectors agree (intersection-based labeling), trading label recall for higher proxy precision. These intersection labels train a multi-label Random Forest (RF) operating entirely on features extracted from headless browser sessions: HTTP headers and cookies, full network-request traces, and HTML/DOM structure.

A website typically exhibits multiple technologies simultaneously (e.g., Nginx/Apache at the edge, a CMS such as WordPress, plug-ins such as WooCommerce, a CDN such as Cloudflare, and client-side libraries such as jQuery or Bootstrap). Therefore, we formulate identification as a multi-label problem where each site has a set of labels. Beyond individual predictions, real deployments contain regular co-deployment patterns. To capture these in an interpretable way, we mine Association Rules (AR) from training predictions using the Apriori algorithm [6] and apply them as a conservative post-processing layer that can slightly increase probabilities only when antecedent technologies are predicted with high confidence.

Models are trained on 8,594 websites with 15,394 engineered features and evaluated using an 80/20 held-out split and five-fold cross-validation. We compare a one-vs-rest logistic regression baseline (linear classifier) against a RF and its rule-augmented variant (RF + AR), reporting macro-, micro-, and weighted-F1, per-technology behavior, and interpretability via feature importance and rule structure.

We organize the study around three pragmatic questions:

— RQ1 (Weak supervision for multi-label identification). Can intersection-based weak labels support multi-label models that achieve strong precision/recall for common web technologies from behavioral and structural features?
— RQ2 (Value of association rules). Do interpretable association rules mined from predicted co-occurrences yield measurable improvements in micro-/weighted-F1 while remaining conservative and stable?
— RQ3 (Stability and limits). How stable are aggregate and per-technology metrics under resampling and label imbalance, and what are the observable mechanisms by which runtime signals help under dynamic loading and partial obfuscation?

Within this framing, the main contributions are:

— Weakly supervised technology identification at scale. We construct and release (conceptually) a measurement pipeline that crawls and processes 8,594 live websites into 15,394 engineered features, and empirically demonstrate that weak labels obtained from the intersection of two mature rule engines can train multi-label models with competitive performance on frequent technologies;
— Hybrid modeling with interpretable rule augmentation. We integrate Apriori association rules as a conservative, post-hoc layer on top of a RF classifier, showing small but consistent gains in micro- and weighted-F1 on both held-out and cross-validation settings, and analyzing where rules help (e.g., plug-in ecosystems) and where they have negligible or negative impact;
— Behavioral and structural interpretability. Through feature-importance analysis and rule networks, we characterize which runtime signals (headers, network paths, DOM patterns) drive predictions for representative technologies, and how co-deployment patterns encode de facto relationships between components;
— Discussion of limits and failure modes. We provide a transparent account of the approach limitations including reliance on noisy supervision, volatility on low-support (long-tail) classes, and residual blind spots for purely backend or rarely-exposed components to inform future, more rigorous ground-truth studies.

The focus of this paper is an applied, empirically grounded methodology for web technology measurement and cybersecurity reconnaissance. The contribution lies in the design and analysis of a weakly supervised, hybrid identification pipeline and its interpretability.

[1] Wappalyzer. Identify technology on websites. Available at: https://wappalyzer.com/ (accessed: 04.05.2026).

[2] BuiltWith. Website profiling and technology lookup. Available at: https://builtwith.com/ (accessed: 04.05.2026).

Related work

Signature-based fingerprinting tools, such as Wappalyzer and those following the OWASP testing guides, identify technologies using markers in HTTP headers, Uniform Resource Locators (URLs) and DOM artifacts. These approaches are fast and interpretable but brittle under modern deployment conditions: CDNs and proxies obscure server metadata, while minification and bundling distort patterns at the script and Cascading Style Sheets (CSS) level. Additionally, dynamic content generation can suppress expected signatures [1, 2]. Surveyed work highlights persistent maintenance burdens and recurring false positives/negatives as technology ecosystems evolve [4, 5]. Earlier heuristic or clustering-based attempts reduced manual effort but still struggled with mixed stacks and ambiguous artifacts [7].

In the domain of learning-based identification, machine-learning approaches use content features, network-flow characteristics, or structural HTML/DOM cues to infer web technologies [3, 8]. Multi-label models are particularly suitable for capturing co-deployed components [8], yet progress is limited by the lack of high-quality ground truth: large-scale, manually verified labels are expensive to obtain. Weak supervision provides a scalable alternative by exploiting noisy or heuristic label sources [5]. Studies mining large browser-interaction datasets show the prevalence of technology-identifying artifacts and underscore the need for models that generalize beyond clean, static conditions [9].

Recent advances (2020–2025) indicate a shift toward deep and multimodal architectures that learn representations from raw markup, script artifacts, and behavioral traces [10]. Traffic-centric fingerprinting explores semi- and self-supervised objectives to handle concept drift and reduce dependency on labels [11, 12]. Multi-layer server fingerprinting separates CDN/proxy effects from origin signals [13], while hybrid static–dynamic analyses continue to expose previously unknown fingerprinting scripts or stylistic cues embedded in CSS/HTML [14, 15]. Despite these advances, obfuscation, infrastructural rewriting, and limited ground truth remain core obstacles [1, 2].

Positioning our work relative to prior work, we focus on a pragmatic middle ground: weak supervision applied to behavioral and structural features gathered from live browsing. Where signature-based methods rely on curated fingerprints, and deep traffic-centric models require extensive labeled data or highly instrumented collection, our study treats two mature rule engines (Wappalyzer and BuiltWith) not as definitional ground truth but as noisy label sources. For each technology, we retain a label only when both detectors agree (intersection-based labeling), and train a multi-label classifier under these conditions on behavioral and structural features.

Three key gaps and motivations recur throughout the literature. First, scalability: web-scale crawls demand automated labeling that remains robust to content churn, yet widely used lists (e.g., Tranco, Majestic[1]) do not contain verified technology annotations [16]. Second, robustness: obfuscation, minification, and infrastructural rewriting degrade both handcrafted fingerprints and model-driven detectors. Third, ground truth: comprehensive multi-label datasets are scarce, limiting the evaluation of learning-based methods [3, 5, 17].

Methodology

As shown in Fig. 1, which illustrates the high-level workflow for rule-augmented weakly supervised identification, the pipeline comprises three stages.

Stage 1. Data preparation.

Stage 2. Model training and baseline evaluation under weak supervision.

Stage 3. Rule mining and augmented inference. The guiding objectives are to reduce dependence on handcrafted signatures, remain robust to obfuscation and dynamic content, and avoid label leakage while preserving reproducibility under weak supervision.

Data preparation

To construct a heterogeneous dataset, we generated candidate domains algorithmically across common Top-Level Domains (TLDs) and valid character combinations. Unlike curated lists such as Tranco [16] or Majestic, this approach prioritizes broad coverage of the web long tail, capturing a more heterogeneous spectrum of technology stacks. Each candidate underwent HTTP/HTTPS liveness testing with retries and conservative timeouts, yielding 10,004 reachable sites. All crawling and text have been complied with robots, and polite-rate throttling was used.

The web crawling pipeline processed each live site through a deterministic headless browser session configured with a clean cache, fixed viewport, and no extensions aligned with prior web technologies identification and measurement studies [7, 18]. Artifacts were collected after DOMContentLoaded and network-idle events, ensuring late-loading resources were included.

Captured evidence included:

— HTML/DOM structure, metadata, unique class and identifier (ID) names;
— HTTP headers and cookies (server frameworks, caching layers);
— favicon hashes (MD5 algorithm) for deduplication-style fingerprints;
— full network request traces (URLs, content types, methods, statuses);
— JavaScript/CSS resource URLs (client-side dependencies).

All artifacts were stored in a modular JavaScript Object Notation and columnar formats, preserving a strict separation between features and labels.

To generate weakly labeled data following principles of programmatic weak supervision, we used two rule-based detectors as noisy label sources $\mathbf{Y}_{\text{weak}}$: Wappalyzer and BuiltWith. For each crawled site and technology, we assigned a positive label only when both detectors agreed on its presence (intersection of predictions), trading some recall for higher label precision. The resulting labels were

[1] Majestic. Majestic Million: the top 1 million websites by referring subnets. Available at: https://majestic.com/reports/majestic-million (accessed: 04.05.2026).

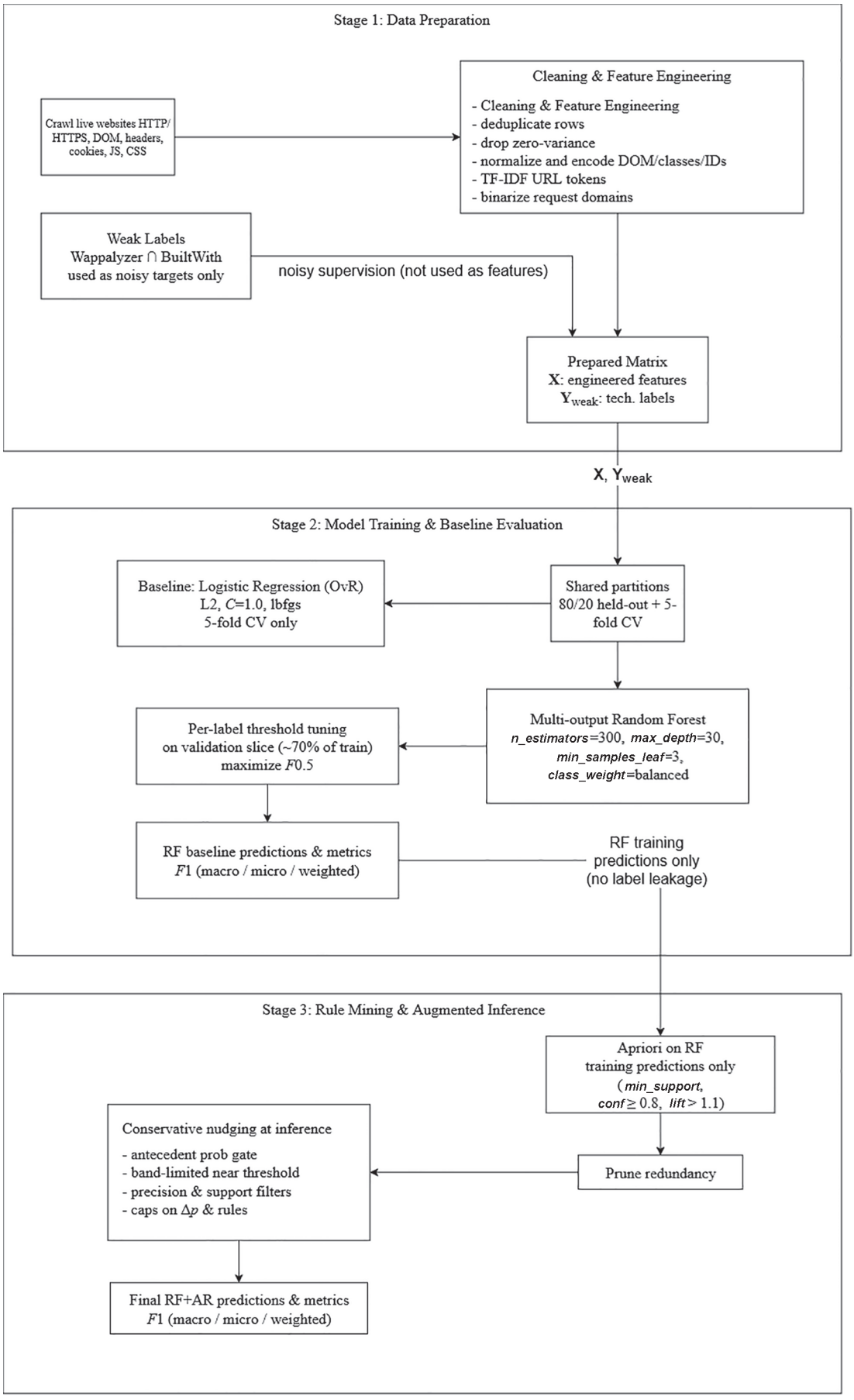

Fig. 1. High-level workflow for rule-augmented weakly supervised identification

used exclusively as targets and were *never* included in the feature matrix, preventing any leakage from the weak supervision sources into the inputs. Schema validation and consistency checks filtered out incomplete or unparsable crawls, producing 8,594 valid websites and 46,737 raw candidate features.

All labels in this study are weak proxies derived from the intersection of Wappalyzer and BuiltWith predictions; they are not manually verified ground truth. Therefore, the reported precision/recall/F1 quantifies agreement with these proxies. This matters especially for server-side technologies whose evidence may be partially hidden from an external observer.

For example, the label Hypertext Preprocessor (PHP) should be read as "externally observable PHP-related signals" rather than "PHP is certainly used and trivially detectable". In realistic deployments, URLs often omit the ".php" suffix due to routing and rewrite rules (front-controller patterns), reverse proxies, CDNs, caching layers, and static fronting; moreover, server-side source fragments such as "<?php" are not exposed to the crawler. As a result, even when PHP is present, externally visible markers (e.g., occasional "index.php"-style paths, "X-Powered-By" variants when not stripped, CMS/plug-in ecosystems (that correlate with PHP) may be intermittent or absent, and the two rule engines may disagree. This explains why PHP precision/recall under weak-label evaluation is below 1.0 without implying that PHP is fundamentally "hard" to detect under full internal access.

Cleaning and feature engineering were performed through a deterministic, modular pipeline that transformed raw artifacts into a structured feature matrix:

— Duplicate and constant removal: Identical rows and zero-variance columns were dropped;
— Feature pruning: Very low-frequency or near-constant features were removed with adaptive thresholds per feature type;
— Name normalization: Randomized HTML IDs (e.g. section-ac3f19) were truncated to improve stability across reloads;
— Categorical encoding: Multi-valued categorical fields (class names, element IDs, and HTTP-level product strings from headers such as Server and X-Powered-By) were vectorized using MultiLabelBinarizer and LabelEncoder [14]. Outputs from the rule-based detectors (Wappalyzer and BuiltWith) were not used as features;
— Network request transformation: URLs were decomposed into path tokens and represented using: Term Frequency–Inverse Document Frequency (TF–IDF) over the top token vocabulary [19], and binary indicators for external request domains (eTLD+1). This captured structural and dependency-level signals while constraining feature explosion;
— Missing values and optimization: Missing fields were imputed with neutral defaults; floats convertible to integers were downcast to reduce memory footprint.

Dataset column C containing lists of network request URLs TF–IDF and binary domain features Parse each URL to extract eTLD+1 domain $d(u)$ Compute TF–IDF vectors over top $k = 20$ URL path tokens Apply MultiLabelBinarizer on unique request domains Concatenate TF–IDF and domain indicators to feature matrix Drop intermediate URL and domain lists.

For the network request transformation step, we applied algorithm to extract and transform network requests. The algorithm requires a dataset column C containing lists of network request URLs and ensures the computation of TF–IDF and binary domain features. The procedure is as follows:

1) Parse each URL to extract the eTLD+1 domain $d(u)$;
2) Compute TF–IDF vectors over the top $k = 20$ URL path tokens;
3) Apply MultiLabelBinarizer to the unique request domains;
4) Concatenate TF–IDF vectors and domain indicators into the feature matrix;
5) Drop intermediate URL and domain lists.

After cleaning and transformation, the final feature matrix consisted of 8,594 rows and 15,394 engineered features. This unified dataset was used across all models and evaluation protocols. Dataset statistics before and after cleaning is shown in Table 1.

We established the final label distribution by processing the outputs from both Wappalyzer and BuiltWith for each site, retaining only those technologies for which the two detectors agreed — the intersection of their predictions. To improve stability by filtering extremely rare labels, we further restricted the label set to technologies appearing on at least 5 websites, yielding 122 retained labels. The resulting weak labels follow a heavy-tailed structure: a handful of ubiquitous technologies (e.g., Apache, jQuery) dominate the head, while numerous niche frameworks populate the long tail. Table 2 reports technology frequencies with support greater than 39 across the dataset for this final label set. For each technology, the table shows the number of websites for which it was present in the final label set (n).

Model training and baseline evaluation

We formulate technology identification as a multi-label classification problem where each website may exhibit multiple technologies simultaneously. The targets consist of binary vectors prefixed with 'tech_', while the feature matrix comprises all engineered indicators.

For our primary classifier, we selected a multi-output RF [20] for its robustness to sparse, heterogeneous inputs and noisy supervision. Unless specified otherwise, hyperparameters were set *n_estimators*=300, *max_depth*=30, *min_samples_leaf*=3, and *class_weight*='balanced'. Rather than using fixed global thresholds, we tuned per-label decision cutoffs on an internal validation slice of each training split. Specifically, for every held-out or cross-validation split, we further partitioned the training portion into 70 %/30 % train/validation; the model was fitted on the 70 %, and per-

Table 1. Dataset statistics before and after cleaning

Metric	Before cleaning	After cleaning
Number of websites	10,004	8,594
Number of features	46,737	15,394

Table 2. Distribution of technologies with support greater than 38 across the dataset

Technology	*n*	Technology	*n*	Technology	*n*
Apache	3,038	jQuery	2,946	PHP	2,504
Google Font API	2,041	Nginx	2,010	Font Awesome	1,747
MySQL	1,665	WordPress	1,639	Bootstrap	1,584
Cloudflare	1,501	jQuery Migrate	1,322	LiteSpeed	1,313
Google Tag Manager	824	animate.css	489	OWL Carousel	480
Elementor	460	Revslider	439	Yoast SEO	431
jQuery UI	345	Modernizr	312	WooCommerce	283
jsDelivr	277	Underscore.js	258	reCAPTCHA	258
Slick	254	Ubuntu	223	Google Maps	222
Lightbox	220	Litespeed Cache	204	Cart Functionality	202
Swiper Slider	174	RoundCube	169	Lua	162
OpenResty	162	Moment.js	146	Windows Server	132
RequireJS	130	Caddy	128	Go	128
IIS	128	prettyPhoto	124	wpBakery	121
Microsoft ASP.NET	120	Prototype	112	Select2	108
Google AdSense	96	Plesk	95	OpenSSL	95
Varnish	92	Joomla	82	CherryPy	82
Node.js	78	Debian	77	CentOS	71
Popper	69	React	69	Ionicons	68
FancyBox	61	WP Super Cache	59	Lodash	58
Drupal	51	Vue.js	51	Osano	47
Ruby on Rails	47	Ruby	47	GSAP	47
Amazon Services	47	DataTables	46	W3 Total Cache	45
WP Rocket	44	Amazon Cloudfront	42	Moment	41
SiteGround	41	YouTube	41	Google PageSpeed	41
Express	41	Python	41	Chart.js	39

label thresholds were selected on the 30 % to maximize $F_{0.5}$, mildly favouring precision over recall. Final metrics (macro-, micro-, and weighted-F1) were always computed on the untouched test portion for that split. All preprocessing and threshold selection were fitted strictly on the training+validation data, preventing any leakage into the test sets.

As a transparent baseline, we employed a multi-label one-vs-rest Logistic Regression (LR) model. It employed ℓ_2 regularization with $C = 1.0$ and the lbfgs solver. LR was evaluated only under five-fold cross-validation; no separate 80/20 held-out split was used. All models and preprocessing components (RF, LR, MultiLabelBinarizer, LabelEncoder) were implemented using scikit-learn [21].

For evaluation, we employed two complementary protocols:

1) Held-out split (main per-label analysis). An 80/20 split was used for RF and RF+AR. All preprocessing, threshold tuning, and rule mining were restricted to the training portion;
2) Cross-validation (robustness check). Five-fold KFold (with shuffling and fixed random seed) was applied. The same folds were used for LR, RF, and RF+AR. In each fold, rules for RF+AR were mined exclusively from RF training predictions.

Macro, micro, and weighted F1-scores were reported alongside class-wise support. Feature importance analysis used impurity-based scores and permutation tests.

Rule mining and augmented inference

We incorporate association rule mining as an interpretable, conservative layer applied after model prediction.

To extract meaningful rules, we mined co-occurrence patterns exclusively from the RF training predictions only, eliminating any possibility of leakage. We employed the Apriori algorithm [6] with thresholds for minimum support and confidence set at no less than 0.8, and lift more than 1.1. The resulting rules were pruned for redundancy and restricted to mid-support technologies to reduce susceptibility to noise.

During the augmented inference phase, these rules act as interpretable priors that nudge the model probabilities rather than overwriting them. Adjustments are applied only when all antecedent technologies in a rule are predicted with high confidence. The influence of any single rule is constrained by precision gates, frequency caps, and

strict per-instance limits on cumulative probability shifts, ensuring stability and preventing runaway amplification of predictions.

For the final evaluation, predictions from the RF model and the hybrid RF+AR system were compared under both evaluation setups using macro, micro, and weighted F1-score. Crucially, no step of rule induction or probability adjustment involved the test data, preserving a complete separation between the stages of supervision, rule mining, and evaluation.

Results and discussion

Building on the pipeline and evaluation protocol describer above, this section analyzes the empirical behavior of the models from four angles:

1) aggregate performance on held-out and cross-validation splits;
2) per-technology metrics and the effect of the association-rule augmentation;
3) feature-level interpretability;
4) robustness under obfuscation and dynamic loading.

All models share the same data partitions and feature space.

Aggregate performance

When evaluating on the fixed 80/20 held-out split, the RF baseline attains micro-F1 = 0.742, macro-F1 = 0.447, and weighted-F1 = 0.720. Adding the association-rule layer slightly increases recall on near-threshold cases, yielding micro-F1 = 0.749, macro-F1 = 0.448, and weighted-F1 = 0.728. The absolute gains are deliberately modest: the rule mechanism is precision-gated and band-limited, designed to correct a subset of under-detections rather than to re-rank predictions wholesale. We define rule coverage on a dataset as the fraction of website–label predictions for which at least one association rule adjusts the base RF probability. On the held-out test set, 23.8 % of predictions fall into this category, indicating that the rule layer operates on a sizable minority of cases while leaving the majority of predictions unchanged.

To compare all models under identical resampling conditions, LR, RF, and RF+AR were evaluated using a shared five-fold KFold cross-validation. Table 3 reports mean F1-scores with 95 % confidence intervals across the five folds.

The ordering is stable across folds: LR < RF < RF + AR. The RF substantially improves over the linear baseline in micro-F1, with smaller but consistent gains in macro- and weighted-F1. The augmented RF+AR then adds a further 1–1.5 percentage points in micro- and weighted-F1 while leaving macro-F1 essentially unchanged, which is consistent with a conservative, recall-oriented adjustment on already confident predictions.

Per-technology performance

To examine the top-performing technologies, Table 4 focuses on the 25 technologies with the highest F1-scores under the RF+AR model. These typically correspond to well-supported, structurally distinctive stacks, or components with strong network-level fingerprints.

Core server and backend components (Apache, Nginx, PHP, MySQL), CMS ecosystems (WordPress, WooCommerce), and front-end/CDN tooling (Bootstrap, jQuery, Cloudflare, jsDelivr) achieve F1-scores around or above 0.80. This indicates that the behavioral and structural features we engineered capture stable, discriminative fingerprints for major, well-established technologies.

The PHP metrics in Table 4 should be interpreted under the weak-label protocol: a true deployment may not expose stable PHP markers to an external crawler, and the intersection of Wappalyzer and BuiltWith can be conservative or inconsistent depending on what is observable (headers stripped by proxies, rewritten URLs, caching, or CMS-specific paths). Hence precision/recall below 1.0 is consistent with partial observability and proxy labels, and does not contradict the fact that PHP can be identifiable in simpler settings (e.g., when ".php" endpoints or explicit headers are present).

To isolate the contribution of the rule layer, Table 5 lists the technologies with the largest positive F1 change between RF and RF+AR on the same held-out split.

Most gains occur within coherent ecosystems, especially WordPress-centric stacks and CDN-related labels. These improvements reflect strong co-deployment patterns uncovered during rule mining such as WordPress ⇒ ⇒ PHP and WooCommerce ⇒ (WordPress ∧ PHP). A small number of infrastructure-oriented labels experience slight regressions (the largest being Debian with a –0.13 F1 delta), but these cases are rare and bounded by the conservative gating and band limits on rule activations.

Association-rule structure and interpretability

The extracted rules form a sparse, interpretable network of technology co-deployments. Fig. 2 visualizes these relations as a lift-weighted graph.

Densely connected clusters correspond to familiar ecosystems: WordPress and its plug-ins, front-end tooling (jQuery, Bootstrap, Font Awesome), CDN stacks, and backend groupings such as PHP ↔ MySQL. The technologies that sit in well-supported clusters of this graph are precisely those that exhibit the clearest F1 improvements under RF+AR, illustrating how the

Table 3. Cross-validation metrics (mean [95 % CI]) across five shared folds

Model	Macro-F1	Micro-F1	Weighted-F1
LR	0.440 [0.423–0.457]	0.629 [0.621–0.637]	0.713 [0.707–0.720]
RF	0.453 [0.439–0.466]	0.750 [0.741–0.758]	0.725 [0.716–0.735]
RF + AR	0.454 [0.440–0.469]	0.763 [0.753–0.772]	0.739 [0.729–0.750]

Table 4. Top 25 technologies by F1-score under RF+AR on the held-out test set

Technology	Precision	Recall	F1-score	Support
Drupal	1.000	1.000	1.000	10
Moment Timezone	1.000	1.000	1.000	10
Stimulus	1.000	1.000	1.000	5
Wix	1.000	1.000	1.000	12
Google Cloud	0.923	1.000	0.960	12
CherryPy	0.880	1.000	0.936	22
RoundCube	0.880	1.000	0.936	22
Vercel	0.857	1.000	0.923	6
Prototype	0.917	0.917	0.917	24
RequireJS	0.929	0.897	0.912	29
WordPress	0.810	0.974	0.885	311
MySQL	0.809	0.965	0.880	316
Apache	0.818	0.911	0.862	651
jQuery	0.781	0.951	0.858	548
Polyfill	1.000	0.750	0.857	12
Cloudflare	0.971	0.755	0.849	265
Osano	1.000	0.733	0.846	15
Nginx	0.875	0.816	0.844	396
PHP	0.838	0.851	0.844	475
Popper	1.000	0.722	0.839	18
Elementor	0.745	0.886	0.809	79
Bootstrap	0.710	0.936	0.807	280
React	0.842	0.762	0.800	21
WooCommerce	0.682	0.957	0.796	47
jsDelivr	0.923	0.692	0.791	52

Table 5. Largest positive and negative F1 changes due to association-rule augmentation (held-out split)

Technology	Support	F1 (RF)	F1 (RF+AR)	Δ F1 (F1 (RF+AR) – F1 (RF))
Yoast SEO	68	0.25	0.39	+0.14
WooCommerce	47	0.55	0.62	+0.07
jQuery	548	0.78	0.83	+0.05
Google Cloud	12	0.96	1.00	+0.04
Google Font API	386	0.68	0.72	+0.04
OWL Carousel	75	0.58	0.60	+0.02
Revslider	82	0.68	0.70	+0.02
MySQL	316	0.86	0.87	+0.01
WordPress	311	0.86	0.88	+0.01
PHP	475	0.77	0.79	+0.01
Debian	20	0.60	0.47	–0.13
UNIX	17	0.75	0.71	–0.04
Font Awesome	331	0.77	0.75	–0.03
jQuery Migrate	238	0.81	0.79	–0.02
Nginx	396	0.89	0.87	–0.01

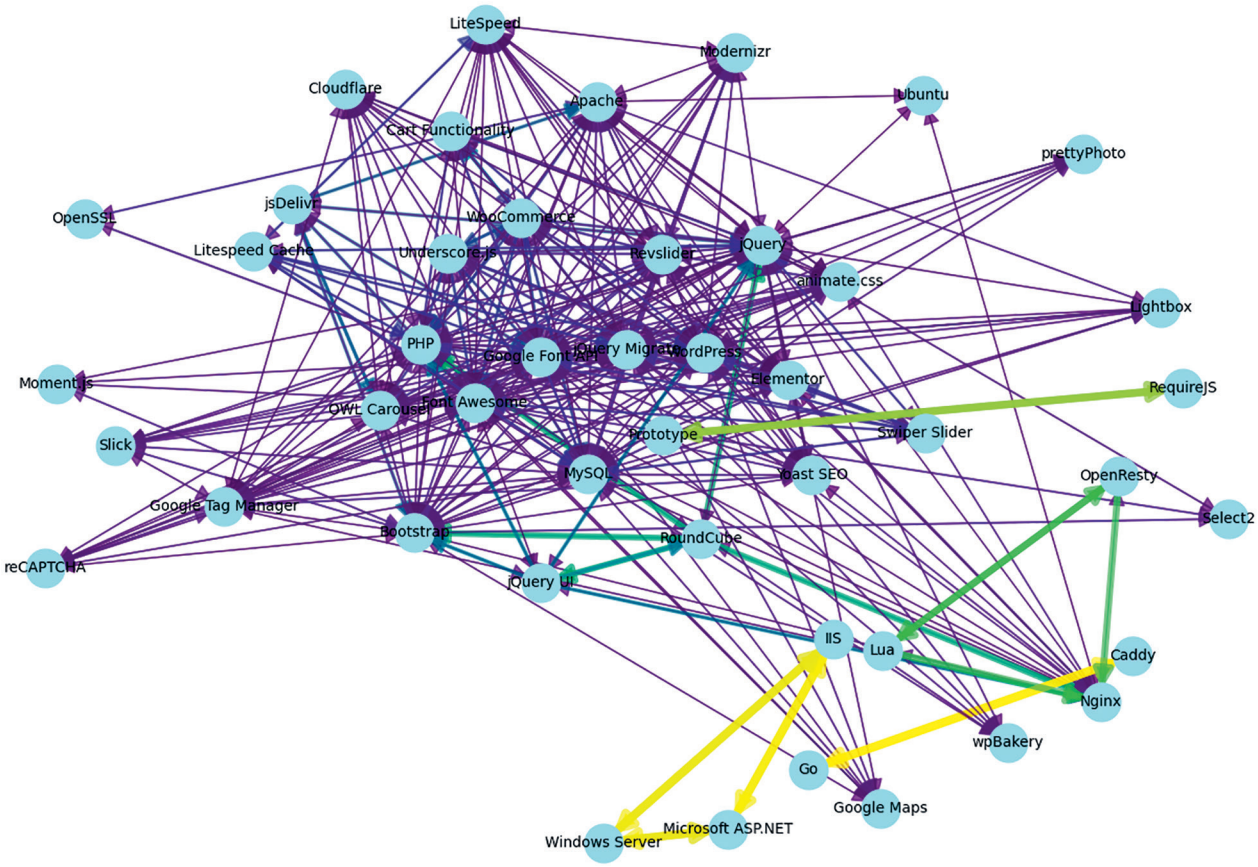

Fig. 2. Association-rule network (lift >1). Edges denote frequent co-deployments in training predictions; edge thickness and color both encode lift magnitude, with thicker and brighter edges indicating stronger co-deployment association

association rules operationalize RQ2 by providing both priors and explanatory structure.

Feature-level interpretability

To elucidate what the RF learns from the engineered feature space, we analyze per-label feature importance. We rank features by their contribution to predicting representative technologies and validate these rankings through permutation importance checks.

For instance, in the case of PHP, the model assigns high importance to variants of the ‘x-powered-by’ HTTP header and to TF–IDF weights for canonical path segments such as ‘wp-includes’ and ‘index.php’.

Conversely, for jQuery, the most influential features are JavaScript asset URLs containing ‘jquery(.min).js’ and co-occurring CSS layout classes commonly found in Bootstrap-style templates. The top 20 features for jQuery, ranked by their importance, are illustrated in Fig. 3.

As anticipated, the detection of Apache primarily relies on server-banner fields (e.g., ‘server’, ‘server_name’) and content-type patterns, reflecting classic HTTP-level fingerprints established in prior work.

Similarly, identifying Microsoft ASP.NET depends on distinctive features such as the ‘VIEWSTATE’ form field, ‘ASP.NET_SessionId’ cookies, and request paths to ‘.aspx’ endpoints, which collectively form clear discriminative markers for this technology stack.

Finally, the detection of WordPress and its WooCommerce extension utilizes a combination of DOM class attributes and path patterns. WordPress is strongly signaled by elements like ‘wp-content’ and ‘wp-includes’, whereas WooCommerce detection emphasizes specific script IDs (e.g., ‘woocommerce-js’) and request patterns associated with cart and checkout flows. These coherent, semantically meaningful patterns confirm that the model learns interpretable cues rather than relying on arbitrary or noisy artifacts, directly addressing the interpretability concerns outlined in RQ2.

Robustness to obfuscation and dynamic loading

The crawling setup described in above records runtime network activity, which enables detection when libraries are loaded indirectly or deliberately hidden. For instance, Listing 1 illustrates a minimal example code where TailwindCSS is injected via a Base64-encoded loader; while the static HTML contains no explicit signature, the resolved request appears in the network trace.

In contrast, under static inspection, no Tailwind-specific tokens appear. However, behavioral features capture the decoded URL and its domain; TF–IDF tokenization of paths and domain indicators then surface the latent dependency. Moreover, similar mechanisms support the strong performance of CDN libraries and JavaScript frameworks in Table 4 and 5, illustrating how behavioral evidence can surface dependencies that static HTML alone would fail to reveal. In turn, this addresses the mechanism underlying

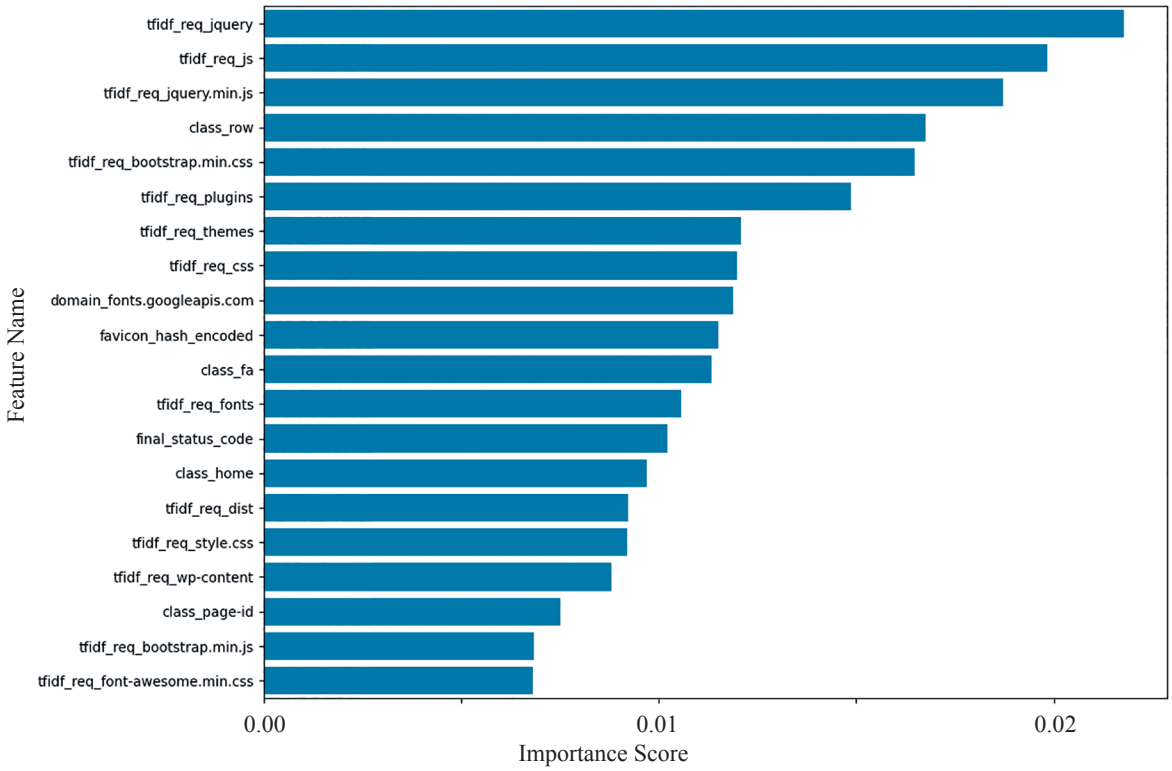

Fig. 3. Top-20 features ranked by importance for jQuery

both RQ1 and RQ3: the models leverage runtime signals to detect technologies even when traditional signatures are weakened or absent. Nevertheless, our evaluation of obfuscation-related robustness is necessarily qualitative. Beyond the synthetic example in Listing 1, we do not perform a controlled, large-scale obfuscation benchmark, and we do not simulate targeted adversarial evasion. Therefore, the claims for RQ3 should be interpreted as evidence of plausible capability, not as a comprehensive adversarial robustness assessment.

Listing 1. Base64-encoded resource loader detected via behavioral signals (code example)

```
<!DOCTYPE html>
<html lang=»en»>
<head>
<meta charset=»UTF-8»>
<title>TailwindCSS Dynamic
   Injection</title>
<script>
   const tailwindBase64 = «aHR0cHM6
      Ly9jZG4udGFpbHdpbmRjc3MuY29t»;
   function loadTailwind() {
      const script = document.
         createElement(‘script’);
      script.src =
         atob(tailwindBase64);
      document.head.
         appendChild(script);
   }
   document.addEventListener(‘DOMC
      ontentLoaded’, loadTailwind);
</script>
</head>
</html>
```

Conclusion

This study demonstrates that weak supervision can be a practical and scalable foundation for web-technology identification when high-quality ground truth is unavailable. By treating the consensus of two rule-based detectors as a source of noisy labels, we trained a multi-output Random Forest on behavioral and structural features extracted from live browser sessions and augmented it with interpretable association rules mined from the model own predictions. This approach operates independently of predefined signature templates, relying instead on behavioral and structural evidence collected from live browsing.

The results confirm the viability of this hybrid method. The Random Forest model substantially outperforms a linear baseline, and the conservative association-rule layer provides small but repeatable gains typically one to two percentage points in micro- and weighted-F1 while maintaining stable macro-F1.

For well-supported technologies, including major server stacks, Content Management System platforms, and front-end tooling, the approach achieved F1-scores above 0.80, proving that weak labels can be leveraged to extract robust, discriminative patterns at scale. Interpretability analyses revealed that the model learns semantically meaningful features (e.g., specific request paths, header fields, Document Object Model structure, and runtime network traces), while the rules capture coherent co-deployment patterns, such as the strong association between WordPress, Hypertext Preprocessor, and WooCommerce.

However, the inherent limits of weak supervision are apparent. Label noise introduces uncertainty, performance on low-support technologies in the long tail remains volatile, and the dataset inherits the biases of the source crawl. Our obfuscation-related evidence is qualitative rather than the result of a controlled adversarial benchmark, so claims of robustness should be interpreted as plausible capability rather than as a formal security guarantee.

In direct answer to our research questions, the experiments indicate that (RQ1) noisy labels derived from intersecting two rule engines can support multi-label models with strong precision and recall on common technologies using behavioral and structural features alone; (RQ2) conservative association-rule augmentation provides small but reproducible gains in micro- and weighted-F1 and yields interpretable co-deployment structure; and (RQ3) the proposed approach shows stable aggregate metrics under modest resampling, while the use of runtime network traces and dynamic loading signals offers a promising, though qualitatively evaluated, path toward greater resilience under obfuscation.

In conclusion, weak supervision emerges as a viable and complementary paradigm for scalable web-technology measurement. It does not replace traditional signature-based methods but offers a powerful alternative for environments where they falter, paving the way for more adaptive and large-scale analysis of the evolving web ecosystem.

References

1. Skolka P., Staicu C.-A., Pradel M. Anything to hide? studying minified and obfuscated code in the web. *Proc. of the World Wide Web Conference (WWW 2019)*, 2019, pp. 1735–1746. doi: 10.1145/3308558.3313752
2. Sarker S., Jueckstock J., Kapravelos A. Hiding in plain site: detecting JavaScript obfuscation through concealed browser API usage. *Proc. of the ACM Internet Measurement Conference*, 2020, pp. 648–661. doi: 10.1145/3419394.3423616
3. Applebaum S., Gaber T., Ahmed A. Signature-based and machine-learning-based web application firewalls: a short survey. *Procedia Computer Science*, 2021, vol. 189, pp. 359–367. doi: 10.1016/j.procs.2021.05.105
4. Zheng R., Ma H., Wang Q., Fu J., Jiang Z. Assessing the security of campus networks: The case of seven universities. *Sensors*, 2021, vol. 21, no. 1, pp. 306. doi: 10.3390/s21010306
5. Zhang J., Hsieh C.-Y., Yu Y., Zhang C., Ratner A. A survey on programmatic weak supervision. *arXiv*, 2022. arXiv:2202.05433. doi: 10.48550/arXiv.2202.05433
6. Agrawal R., Imieliński T., Swami A. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 1993, vol. 22, no. 2, pp. 207–216. doi: 10.1145/170036.170072
7. Berg A., Lamberg N. Automatic Fingerprinting of Websites. *Technical report*, 2020.
8. Shi Y., Yu W., Zhao Y., Jia Y. A web application fingerprint recognition method based on machine learning. *Computer Modeling in Engineering & Sciences*, 2024, vol. 140, no. 1, pp. 887–906. doi: 10.32604/cmes.2024.046140
9. Rizzo V., Traverso S., Mellia M. Unveiling web fingerprinting in the wild via code mining and machine learning. *Proceedings on Privacy Enhancing Technologies*, 2021, vol. 2021, no. 1, pp. 43–63. doi: 10.2478/popets-2021-0004
10. Qiang W., Ren K., Wu Y., Zou D., Jin H. DeepFPD: browser fingerprinting detection via deep learning with multimodal learning and attention. *IEEE Transactions on Reliability*, 2024, vol. 73, no. 3, pp. 1516–1528. doi: 10.1109/TR.2024.3355233
11. Bahramali A., Bozorgi A., Houmansadr A. Realistic website fingerprinting by augmenting network traces. *Proc. of the ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 1035–1049. doi: 10.1145/3576915.3616639
12. Deng X., Li Q., Xu K. Robust and reliable early-stage website fingerprinting attacks via spatial-temporal distribution analysis. *Proc. of the ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1997–2011. doi: 10.1145/3658644.3670272
13. Topcuoglu C., Onarlioglu K., Jabiyev B., Kirda E. Untangle: multi-layer web server fingerprinting. *Proc. of the Network and Distributed System Security Symposium*, 2024, doi: 10.14722/ndss.2024.24497
14. Bird S., Mishra V., Englehardt S., Willoughby R., Zeber D., Rudametkin W., Lopatka M. Actions speak louder than words: semi-supervised learning for browser fingerprinting detection. *arXiv*, 2020. arXiv:2003.04463. doi: 10.48550/arXiv.2003.04463
15. Lin X., Araujo F., Taylor T., Jang J., Polakis J. Fashion faux pas: implicit stylistic fingerprints for bypassing browsers' anti-fingerprinting defenses. *Proc. of the IEEE Symposium on Security and*

Литература

1. Skolka P., Staicu C.-A., Pradel M. Anything to hide? studying minified and obfuscated code in the web // Proc. of the World Wide Web Conference (WWW 2019). 2019. P. 1735–1746. doi: 10.1145/3308558.3313752
2. Sarker S., Jueckstock J., Kapravelos A. Hiding in plain site: detecting JavaScript obfuscation through concealed browser API usage // Proc. of the ACM Internet Measurement Conference. 2020. P. 648–661. doi: 10.1145/3419394.3423616
3. Applebaum S., Gaber T., Ahmed A. Signature-based and machine-learning-based web application firewalls: a short survey // Procedia Computer Science. 2021. V. 189. P. 359–367. doi: 10.1016/j.procs.2021.05.105
4. Zheng R., Ma H., Wang Q., Fu J., Jiang Z. Assessing the security of campus networks: The case of seven universities // Sensors. 2021. V. 21. N 1. P. 306. doi: 10.3390/s21010306
5. Zhang J., Hsieh C.-Y., Yu Y., Zhang C., Ratner A. A survey on programmatic weak supervision // arXiv. 2022. arXiv:2202.05433. doi: 10.48550/arXiv.2202.05433
6. Agrawal R., Imieliński T., Swami A. Mining association rules between sets of items in large databases // ACM SIGMOD Record. 1993. V. 22. N 2. P. 207–216. doi: 10.1145/170036.170072
7. Berg A., Lamberg N. Automatic Fingerprinting of Websites. Technical report. 2020.
8. Shi Y., Yu W., Zhao Y., Jia Y. A web application fingerprint recognition method based on machine learning // Computer Modeling in Engineering & Sciences. 2024. V. 140. N 1. P. 887–906. doi: 10.32604/cmes.2024.046140
9. Rizzo V., Traverso S., Mellia M. Unveiling web fingerprinting in the wild via code mining and machine learning // Proceedings on Privacy Enhancing Technologies. 2021. V. 2021. N 1. P. 43–63. doi: 10.2478/popets-2021-0004
10. Qiang W., Ren K., Wu Y., Zou D., Jin H. DeepFPD: browser fingerprinting detection via deep learning with multimodal learning and attention // IEEE Transactions on Reliability. 2024. V. 73. N 3. P. 1516–1528. doi: 10.1109/TR.2024.3355233
11. Bahramali A., Bozorgi A., Houmansadr A. Realistic website fingerprinting by augmenting network traces // Proc. of the ACM SIGSAC Conference on Computer and Communications Security. 2023. P. 1035–1049. doi: 10.1145/3576915.3616639
12. Deng X., Li Q., Xu K. Robust and reliable early-stage website fingerprinting attacks via spatial-temporal distribution analysis // Proc. of the ACM SIGSAC Conference on Computer and Communications Security. 2024. P. 1997–2011. doi: 10.1145/3658644.3670272
13. Topcuoglu C., Onarlioglu K., Jabiyev B., Kirda E. Untangle: multi-layer web server fingerprinting // Proc. of the Network and Distributed System Security Symposium. 2024. doi: 10.14722/ndss.2024.24497
14. Bird S., Mishra V., Englehardt S., Willoughby R., Zeber D., Rudametkin W., Lopatka M. Actions speak louder than words: semi-supervised learning for browser fingerprinting detection // arXiv. 2020. arXiv:2003.04463. doi: 10.48550/arXiv.2003.04463
15. Lin X., Araujo F., Taylor T., Jang J., Polakis J. Fashion faux pas: implicit stylistic fingerprints for bypassing browsers' anti-

Privacy (SP), 2023, pp. 987–1004. doi: 10.1109/SP46215.2023.10179437
16. Le Pochat V., Van Goethem T., Tajalizadehkhoob S., Korczyński M., Joosen W. Tranco: a research-oriented top sites ranking hardened against manipulation. *Proc. of the 26th Annual Network and Distributed System Security Symposium*, 2019, doi: 10.14722/ndss.2019.23386
17. Zhang M.-L., Zhou Z.-H. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 2014, vol. 26, no. 8, pp. 1819–1837. doi: 10.1109/TKDE.2013.39
18. Altulaihan E.A., Alismail A., Frikha M. A survey on web application penetration testing. *Electronics*, 2023, vol. 12, no. 5, pp. 1229. doi: 10.3390/electronics12051229
19. Abusnaina A., Jang R., Khormali A., Nyang D., Mohaisen D. DFD: adversarial learning-based approach to defend against website fingerprinting. *Proc. of the IEEE INFOCOM 2020 – IEEE Conference on Computer Communications*, 2020, pp. 2459–2468. doi: 10.1109/INFOCOM41043.2020.9155465
20. Breiman L. Random forests. *Machine Learning*, 2001, vol. 45, no. 1, pp. 5–32. doi: 10.1023/A:1010933404324
21. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2825–2830.

fingerprinting defenses // Proc. of the IEEE Symposium on Security and Privacy (SP). 2023. P. 987–1004. doi: 10.1109/SP46215.2023.10179437
16. Le Pochat V., Van Goethem T., Tajalizadehkhoob S., Korczyński M., Joosen W. Tranco: a research-oriented top sites ranking hardened against manipulation // Proc. of the 26th Annual Network and Distributed System Security Symposium. 2019. doi: 10.14722/ndss.2019.23386
17. Zhang M.-L., Zhou Z.-H. A review on multi-label learning algorithms // IEEE Transactions on Knowledge and Data Engineering. 2014. V. 26. N 8. P. 1819–1837. doi: 10.1109/TKDE.2013.39
18. Altulaihan E.A., Alismail A., Frikha M. A survey on web application penetration testing // Electronics. 2023. V. 12. N 5. P. 1229. doi: 10.3390/electronics12051229
19. Abusnaina A., Jang R., Khormali A., Nyang D., Mohaisen D. DFD: adversarial learning-based approach to defend against website fingerprinting // Proc. of the IEEE INFOCOM 2020 — IEEE Conference on Computer Communications. 2020. P. 2459–2468. doi: 10.1109/INFOCOM41043.2020.9155465
20. Breiman L. Random forests // Machine Learning. 2001. V. 45. N 1. P. 5–32. doi: 10.1023/A:1010933404324
21. Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., et al. Scikit-learn: machine learning in Python // Journal of Machine Learning Research. 2011. V. 12. P. 2825–2830.

Authors

Rand Deeb — PhD Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0006-2730-1550, rand.sec96@gmail.com
Alisa A. Vorobeva — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, sc 57191359167, https://orcid.org/0000-0001-6691-6167, vorobeva@itmo.ru

Авторы

Дееб Ранд — аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0006-2730-1550, rand.sec96@gmail.com
Воробьева Алиса Андреевна — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, sc 57191359167, https://orcid.org/0000-0001-6691-6167, vorobeva@itmo.ru

Received 10.12.2025
Approved after reviewing 27.02.2026
Accepted 18.05.2026

Статья поступила в редакцию 10.12.2025
Одобрена после рецензирования 27.02.2026
Принята к печати 18.05.2026