Defending Job Platforms from Non-Genuine Applications Using Layered Detection and Anomaly Modeling

Rama Rohit Reddy Gangula
Indeed
rrohit@indeed.com

Vijay Vardhan Alluri
Indeed
vvalluri@indeed.com

Saif Jawaid
Indeed
sjawaid@indeed.com

Dhwaj Raj
Indeed
draj@indeed.com

Udit Jindal
Indeed
ujindal@indeed.com

***Abstract*—Online job-application funnels are increasingly abused by automated campaigns that flood employers with non-genuine submissions, distorting metrics and eroding platform trust. We report on the first production-scale, defense-in-depth system that detects and mitigates such abuse in real time on Indeed.com, a major job marketplace processing tens of millions of applications each week. Our architecture couples lightweight client-side traps like selector obfuscation, distributed honeypots, browser-trust signals, and Google invisible reCAPTCHA with a multivariate Isolation-Forest anomaly model that operates entirely without labelled data. A novel precision-weighted F1 objective steers threshold selection to minimise user friction while preserving coverage. Deployed globally, the system blocks a significant number of fraudulent applications per day and achieves a 10.23% reduction in suspected abuse volume without degrading legitimate conversion. We detail the layered design, feature engineering, unsupervised modelling, and adaptive mitigation pipeline, and distill lessons for practitioners defending high-throughput, adversarial web services where labelled data are scarce.**

***Keywords*—** job-platform security, unsupervised anomaly detection, Isolation Forest, real-time bot mitigation, layered defense architecture, large-scale fraud prevention, precision-weighted thresholding.

I. Introduction

Modern online job marketplaces operate at extraordinary scale. Our platform alone processes tens of millions of applications each week. This volume, coupled with the economic value of each successful placement, has made the application funnel an attractive target for adversaries who automate, script, or mass-submit non-genuine applications. Such abuse pollutes recruiter pipelines, distorts hiring metrics, and ultimately erodes employer trust.

A. The Security Challenge

Detecting fake job applications differs fundamentally from classic spam or credential-stuffing defenses. Adversaries can mimic legitimate "click-and-scroll" patterns [1], distribute traffic across proxy networks [2], and adapt quickly to static heuristics or IP-based rate limits [3]. Meanwhile, defenders must preserve a friction-free experience for genuine job seekers; even a modest false-positive rate risks reputational damage and lost hires.

Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-00-8
https://dx.doi.org/10.14722/prism.2026.23004
www.ndss-symposium.org

B. Constraints

Three constraints shape the problem:

- **Real-time decisions**. Each request must be scored and, if necessary, mitigated within less than 30ms of latency budget for the application funnel.
- **Label scarcity**. Manual ground truth is infeasible at platform scale; any solution must learn from unlabeled traffic [4].
- **Adversarial evolution**. Bots continually refine tactics, rendering brittle rules obsolete [5].

To address these challenges, we developed and deployed a robust, scalable, real-time system on Indeed's global platform. Our approach strategically integrates layered client-side and server-side defenses, centered around an unsupervised Isolation Forest anomaly detection model specifically designed for effectiveness without labeled training data. The key components of our system include:

- A layered detection architecture that integrates UI obfuscation to resist scripted bots, invisible honeypots to flag automation, browser trust signals to assess device legitimacy, and Google's invisible reCAPTCHA to derive behavioral scores.
- A real-time anomaly detection model based on Isolation Forests [6] , designed to detect suspicious interaction patterns such as bursty application behavior, low preview engagement, and inconsistent trust signals without relying on labeled training data.
- A custom F1-penalty evaluation metric that explicitly penalizes false positives, allowing us to optimize for precision while maintaining effective coverage of abusive activity.

A dynamic mitigation layer that applies friction selectively, routing high-risk applications identified through explicit recaptcha challenges, ensuring adversaries incur significant cost while legitimate users experience minimal friction.

II. Related work

Research on defending job-application funnels intersects four streams of security literature.

Abuse and fraud in online hiring or marketplaces. Early work primarily focused on fraudulent job postings — scam listings explicitly designed to lure candidates into deceptive employment schemes. Recently, however, industry reports suggest a shift towards automated systems used by adversaries to mass-submit applications, potentially overwhelming recruiters with large volumes of non-genuine or low-quality applications. Similar patterns have been observed in other online marketplaces, such as gig-work or e-commerce platforms, where low-friction submission forms have attracted automation-based abuse, motivating platforms to adopt anomaly-based defences rather than static rules alone [7] [8].

Unsupervised anomaly detection for security. Unsupervised anomaly detection has emerged as a valuable approach for addressing

security challenges, especially when labeled data is scarce or unavailable. Isolation Forest, a tree-based unsupervised model introduced by Liu et al. [6], has gained popularity because of its efficiency, linear-time complexity, and distribution-free assumptions. Variants leveraging deep learning, such as the Kitsune ensemble of autoencoders [9], have demonstrated their effectiveness even under real-time constraints and limited computational resources. Similar unsupervised methods like one-class SVMs [10], clustering algorithms, and reconstruction-error techniques have also been successfully applied to detect anomalies in related areas such as intrusion detection and fraud prevention. Given these successes, unsupervised methods appear well-suited for detecting fraudulent or automated job-application submissions, particularly because labeled examples of such abuse are often limited and the behaviors of adversaries frequently change over time.

Large-scale layered anti-abuse systems. Operational anti-abuse systems often employ layered architectures that combine multiple detection strategies to balance speed, accuracy, and usability. For example, X's BotMaker system utilizes multiple layers of rules and machine-learning-based filters arranged in three tiers, significantly reducing spam traffic without negatively affecting genuine user interactions[11]. Similarly, Íntegro pipeline combines graph-based ranking with behavioral classifiers to improve the detection of fake accounts compared to simpler heuristics[12]. Such industry deployments illustrate the advantage of applying precision-driven gating mechanisms at critical user actions, followed by deeper asynchronous analysis and periodic model updates. Given the high volume of job applications and the risk of inadvertently blocking legitimate users, adopting a layered approach may effectively mitigate abuse while preserving platform usability.

Graph and stream methods for coordinated abuse. Graph-based and streaming analytics techniques have been successfully utilized for identifying coordinated and sophisticated abusive behavior, especially when attackers use multiple accounts or coordinated activity to evade detection. Systems such as CopyCatch identify anomalous clusters of suspicious user interactions, like coordinated "likes" or shares within social networks [13]. MIDAS, a streaming-based detection system, flags anomalies by monitoring dynamic edge streams, enabling the identification of abusive activities within extremely short time frames [14]. SynchroTrap further extends these concepts by clustering massive activity logs to detect loosely connected botnets across online social networks [15]. These methods capitalize on structural or temporal deviations from normal user behavior patterns, suggesting their potential applicability to detecting coordinated fake-job-application campaigns, particularly by leveraging timing, velocity, or clustering patterns unique to automation-driven abuse.

Summary. Across the reviewed literature, several key principles clearly emerge: (i) unsupervised anomaly detection methods are particularly effective when labeled data is limited or attackers adapt quickly, making supervised detection less reliable; (ii) layered anti-abuse architectures combining rapid detection at critical user actions with deeper asynchronous analysis can significantly enhance security without compromising user experience; and (iii) graph- and stream-based methods are beneficial for uncovering coordinated or automated abusive behaviors that might evade simpler detection strategies. Drawing from these insights, our work applies these principles to the domain of fraudulent job application submissions. By combining anomaly detection with layered client and server-side defensive strategies, we aim to achieve effective production-grade protection against large-scale application abuse.

III. Our Approach

To meet these constraints, we designed a layered, defense-in-depth system that fuses lightweight client-side instrumentation with an unsupervised anomaly detection service in the backend. Key elements include:

- Selector obfuscation and invisible honeypots that expose DOM-scraping bots without user impact.
- Browser trust signals and Google invisible reCAPTCHA scores that supply passive risk indicators.
- A multivariate Isolation-Forest model trained on six carefully engineered features—covering trust, velocity, and behavioural pacing—thus avoiding the need for labels.
- A custom precision-weighted F1 metric that explicitly penalises false positives during threshold tuning.
- A dynamic mitigation layer that escalates only high-risk sessions to a hard reCAPTCHA challenge, ensuring human applicants proceed unhindered.

A. Impact at Scale

Deployed globally on Indeed's platform, our system has consistently demonstrated real-world effectiveness, including:

- Blocking significant number[1] of fraudulent applications per day.
- Flagging just about 4% of the total traffic as suspicious.
- Achieving a 10.23% reduction in abusive volume.
- Maintaining a false-positive rate below 5%.

These results demonstrate that precision-first, label-light anomaly modelling can defend high-throughput, adversarial services without degrading user experience.

B. Contributions

This paper makes four primary contributions:

- **Problem Formalisation**. We characterise non-genuine job-application abuse and articulate a practical threat model tailored to large job marketplaces.
- **Layered Detection Architecture**. We present the first fully deployed, client-to-server defense stack that combines passive traps with real-time unsupervised scoring in this domain.
- **Precision-Weighted Evaluation Metrics**. We introduce a custom F1-penalty objective that aligns offline tuning with business risk, and we show its effectiveness in threshold calibration.
- **Comprehensive Evaluation**. We report production metrics and operational lessons from protecting a top-tier job platform at global scale.

C. Paper Road-map

Section IV details the background and threat model. Section V describes the system architecture, followed by Section VI on feature engineering. Section VII presents the anomaly-detection model and custom metric, while Section VIII explains the mitigation layer. Section IX reports the evaluation results, Section X concludes with the lessons learnt, and Section XI describes the efforts that will follow this study.

In this context, we next discuss the adversary capabilities and design constraints that guided our defense strategy.

IV. Background and Threat Model

A. Background

At production scale, a major job-marketplace such as ours processes tens of millions of applications each week—traffic that inevitably attracts adversaries seeking to game the hiring funnel for profit or influence. A dominant abuse vector is the non-genuine application: submissions driven by automated scripts, headless browsers, or large proxy farms that mimic legitimate behaviour just well enough to slip past naive rate limits. Besides wasting recruiter effort, these events distort engagement metrics and erode employer trust in the platform. Traditional counter-measures-static CAPTCHAs, IP blacklists, or hand-tuned heuristics—either crumble under evasion or impose unacceptable friction on genuine jobseekers. A more adaptive, defence-in-depth approach is therefore required.

[1]The actual figures of fraudulent applications blocked daily by Indeed are protected and are not available due to internal non-disclosure guidelines.

B. Threat Model

We model abuse at the granularity of individual application sessions, defined as the complete user journey from initial job-detail preview to final application submission.

1) Adversary Goals and Capabilities: The adversary seeks to inject large volumes of spoofed or low-quality applications while remaining indistinguishable from legitimate users. Concretely, we assume attackers can:

- Automate the UI with Selenium, Puppeteer or custom headless browsers that replay realistic click-and-scroll sequences.
- Drive backend APIs directly, bypassing JavaScript, by re-using jobseeker credentials or session cookies obtained elsewhere.
- Obfuscate infrastructure via residential proxy networks, rapid IP rotation, user-agent spoofing and frequent environment resets.
- Iterate quickly, observing mitigation feedback and refining tactics to appear ever more human-like.

2) Non-Genuine Application Criteria: A session is deemed non-genuine when one or more of these signals holds:

- Automated or scripted submission with minimal real interaction.
- Suspicious behavioural patterns such as sub-second page dwell, excessive burst velocity, or clicks on hidden UI elements.
- Anomalous infrastructure signals like low-reputation IPs, mismatched device fingerprints, or missing browser-trust signals.

3) Detection Objectives and Design Constraints: Any production-grade defence must satisfy four hard requirements:

- **Real-time scoring**. Decisions must be completed within the application funnel's sub-second budget to avoid impacting legitimate user experience.
- **Label-free operation**. As clean ground truth is unavailable, detection must learn from unlabeled traffic where genuine sessions dominate.
- **Precision-first mitigation**. False positives carry outsized business cost, so friction is applied only to high-confidence anomalies.
- **Scalability and resilience**. The pipeline must sustain peak traffic and remain robust to feature gaps or adversarial drift.

These constraints drive the layered architecture, feature design, and unsupervised anomaly-detection strategy detailed in the following sections.

V. System Architecture

Our defense system is engineered as a layered, defense-in-depth pipeline that blends client-side deception, high-throughput stream processing, and real-time unsupervised scoring. Figure 2 illustrates the major components and their data flow. We summarise the design goals, then describe each layer from the browser to the mitigation engine.

1) Design Goals:

- **Real-time operation**. All decisions must complete within the $<$30 ms latency budget of the application funnel.
- **Label-free learning**. The detector must learn from unlabeled traffic where genuine applications dominate.
- **Precision-first enforcement**. Friction is acceptable only for high-confidence anomalies; false positives carry outsized business cost.
- **Scalability and resiliency**. The architecture must sustain $>$200 RPS at peak and degrade gracefully under partial failures.

These requirements shaped the layered architecture summarised as follows.

2) Client-Side Instrumentation: The first line of defence raises the cost of automation before any traffic reaches the backend

- **Build-Time Selector Obfuscation (BTSO)**. Static DOM identifiers are replaced with per-build, non-deterministic strings, defeating scripts that rely on hard-coded selectors. The transform executes at build time and imposes zero runtime overhead.

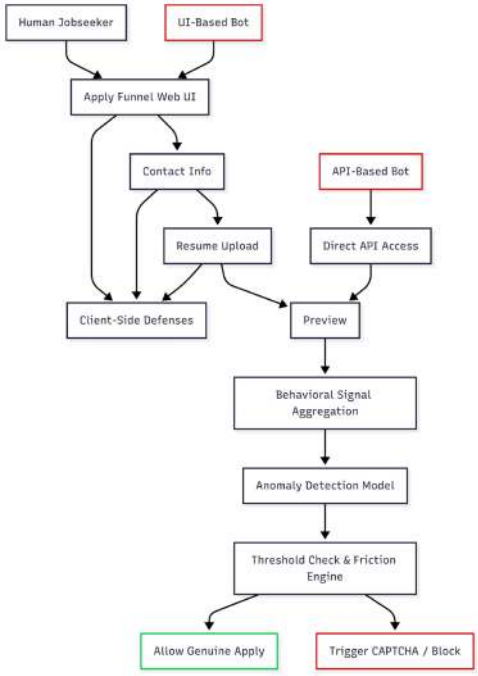

Fig. 1. High-level illustration of the job application funnel, adversarial entry points, and our layered defense approach. Client-side instrumentation across multiple pages captures behavioral signals used by the backend to detect and mitigate non-genuine applications before submission.

- **Distributed Honeypot Traps**. Hidden inputs and decoy buttons are interleaved throughout the apply journey. Any interaction with these elements is a strong indicator of automation yet is invisible to humans.
- **Browser Trust Signals**. Lightweight fingerprint checks (e.g., navigator.webdriver, plugin entropy) flag headless or instrumented browsers with minimal client cost.
- **Smart Hashing via Invisible reCAPTCHA**. Google's invisible reCAPTCHA token is repurposed as a mandatory client hash for backend API calls; missing or invalid tokens trigger an immediate hard challenge, blocking UI-bypass bots.

All signals are logged asynchronously, adding $<$ 2ms to page load and no user-visible friction for $>$ 99% of sessions.

3) Signal Ingestion and Feature Enrichment: Client events stream into Kafka where a stateless enrichment service joins them with contextual metadata (job, account history, geo, device). It computes six real-time features—trust flags, honeypot triggers, reCAPTCHA score, two velocity metrics, and timeToPreview—robust to partial logs and concept drift. The resulting vector is stored in a Redis cache keyed by session and forwarded to the scoring tier.

4) Anomaly Scoring Service: At the Preview-and-Apply which is the final step in the application flow, the cached feature vector is scored by a high-depth Isolation Forest hosted as a gRPC microservice. Median inference latency is 15ms and peak throughput exceeds 200 RPS. The model outputs a continuous anomaly score $\in$ [0,1]. A pre-tuned threshold—selected with our custom precision-weighted F1 metric—maps the score to three risk bands: pass-through, monitor, or mitigate.

5) Mitigation Engine: Sessions in the mitigate band are synchronously routed to a hard reCAPTCHA challenge on the same page. The challenge must be solved before the application is submitted, ensuring bots are blocked while genuine users retain a recovery path. The gate operates fail-open: if scoring is unavailable, applications proceed unimpeded, preventing accidental denial of service.

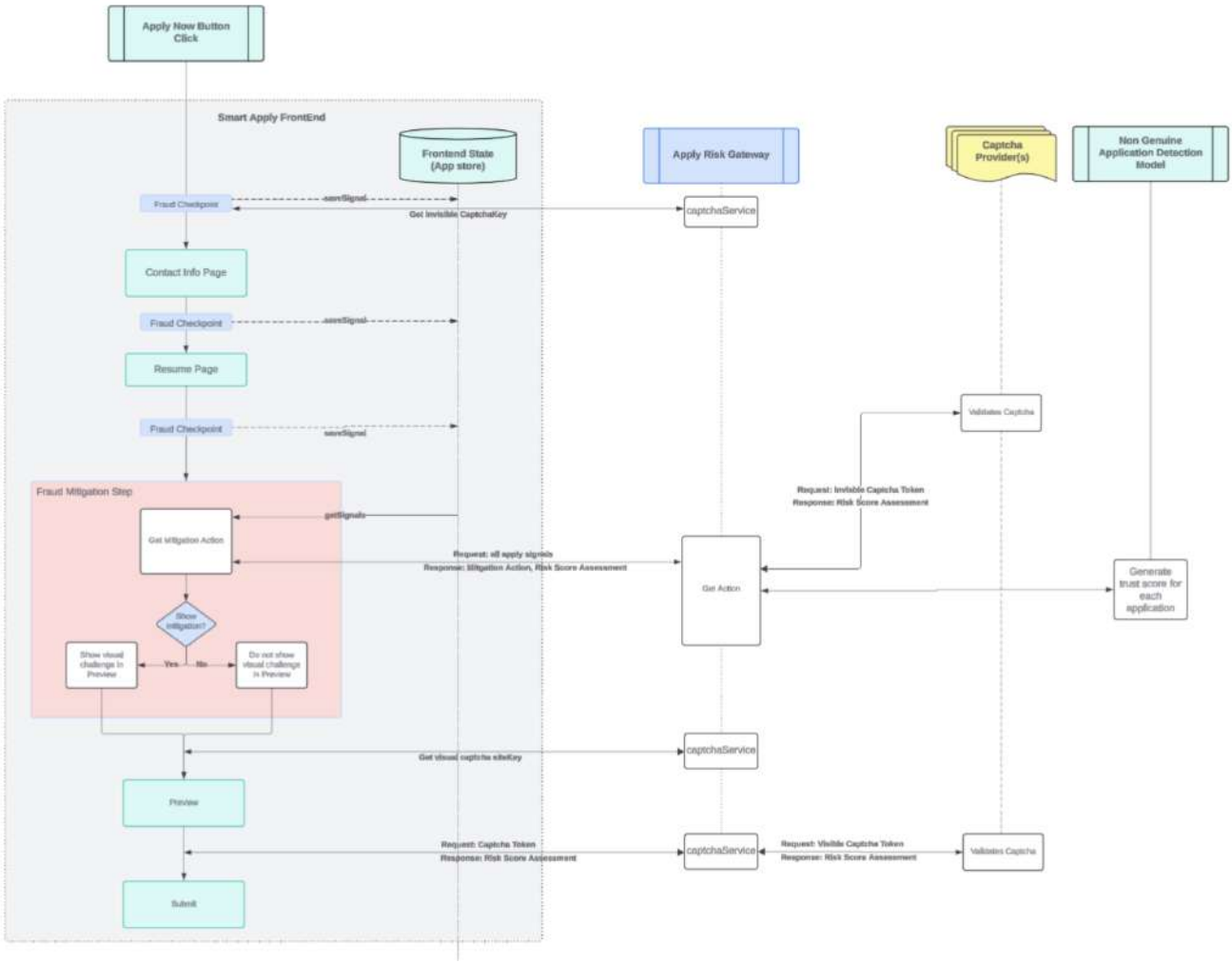

Fig. 2. System architecture for detecting and mitigating non-genuine job applications. The pipeline includes client-side instrumentation (e.g., selector obfuscation, honeypots), backend feature aggregation, an unsupervised anomaly detection model, and a friction engine based on risk scoring thresholds.

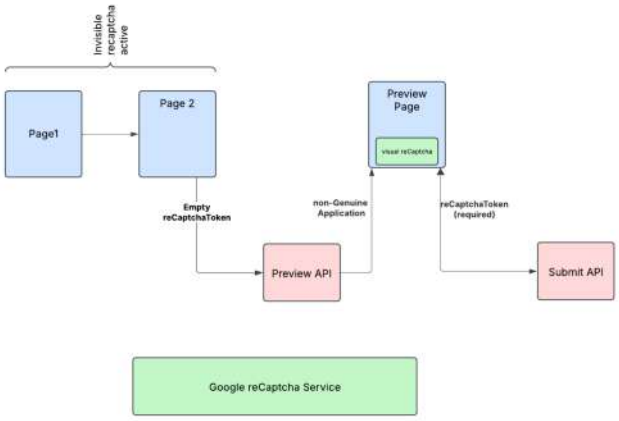

Fig. 3. API-based bot mitigation via client-side smart hashing. Sophisticated bots bypassing the frontend and directly accessing backend APIs are intercepted by requiring a reCAPTCHA-generated token. If the token is missing or invalid, the session is challenged with a visual CAPTCHA, preventing unauthenticated automation from completing the application flow.

6) Production Monitoring and Feedback: A dedicated observability stack tracks score distributions, friction rates, conversion impact, and drift metrics in real time. Weekly audits join model logs with downstream hire outcomes to estimate false-positive precision and trigger threshold recalibration when required. Warm-start retraining pipelines allow friction-free rollout of updated models behind feature flags.

This layered architecture satisfies our design goals: it thwarts UI and API automation early, enriches sparse behavioural signals, applies unsupervised scoring at scale, and introduces user friction only where the risk justifies it. The next section details how careful feature design enables the Isolation Forest to separate genuine and non-genuine sessions with high precision.

A. Risk Band Detection Mechanism

To briefly describe the full detection and mitigation pipeline from the browser to backend action, we present Algorithm 1, which outlines the end-to-end defense flow.

VI. Feature Analysis

Our anomaly-detection model relies on a compact, six-feature vector purposely designed for precision, latency, and robustness in an unlabeled, adversarial setting. The guiding principles were (i) high

Algorithm 1 End-to-End Defense Pipeline for Non-Genuine Application Detection

Input: Browser session events E (DOM interactions, timing, device info)
Output: Decision d$\in$ {Allow, Monitor, Challenge}

Client-Side Phase
Apply Build-Time Selector Obfuscation (*BTSO*) to DOM elements
Insert invisible honeypot elements into the application flow
Capture browser trust signals (e.g., *navigator.webdriver*, *plugin entropy*)
Generate Google invisible reCAPTCHA token
Forward E and token to the backend

Feature Extraction Phase
Compute feature vector F = {*isTrusted*, *honeypot_triggered*, *recaptcha_score*, *velocity_per_minute*, *velocity_per_hour*, *timeToPreview*}
Store F in Redis cache
Forward F to anomaly scoring service

# **Anomaly Detection Phase**
score $\leftarrow$ IF_Model.predict(F) // Isolation Forest score $\in$ [0,1]
if score $\leq$ T1 **then**
    $RiskBand \leftarrow Pass-Through$
else if $T1 < score \leq T2$ **then**
    $RiskBand \leftarrow Monitor$
else
    $RiskBand \leftarrow Mitigate$
end if
#**Mitigation Phase**
if $RiskBand = Pass-Through$ **then**
    return Allow
else if $RiskBand = Monitor$ **then**
    log & tag session
    return Allow
else if $RiskBand = Mitigate$ **then**
    Trigger reCAPTCHA challenge
end if

signal-to-noise, where every feature should add discriminative power; (ii) real-time availability, where all signals must be computable inline in the $<$ 30ms budget; and (iii) resilience to evasion, where features should be difficult for bots to spoof consistently.

A. Feature Set Overview

These six signals provide complementary perspectives: passive trust (isTrusted, recaptcha_score), active traps (honeypots_triggered), behavioural dynamics (velocity pair), and temporal engagement (timeToPreview).

To further validate the discriminative power of these features, Figure 4 presents a SHAP summary plot showing their relative importance and directional influence on the anomaly score. Notably, higher values of honeypot_triggered, velocity_per_minute and velocity_per_hour (red points) have a strong negative impact on the model's output, driving it towards the non-genuine classification, while higher values of features like timetoPreview, recaptcha_score and isTrusted provide positive trust signals. This analysis aligns with our empirical observations and confirms that the minimal six-feature vector is both interpretable and high-leverage.

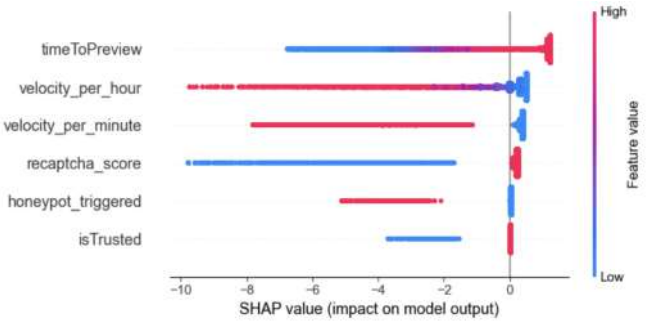

Fig. 4. SHAP summary plot visualizing feature importance and the directional impact on anomaly scoring. Higher values of velocity features are the strongest indicators of automation

B. Signal-Specific Details

- **isTrusted**. Set when critical clicks originate from an authentic browser context. Highly precise: $>$ 98% of legitimate sessions emit it, whereas many automation frameworks miss required event attributes.
- **honeypot_triggered**. Fired when a hidden input/button is accessed. Because the elements are invisible and selector-obfuscated, the feature offers near-zero false positives while catching unsophisticated DOM scrapers.
- **recaptcha_score**. A continuous, latency-free trust score generated by Google's invisible reCAPTCHA. Mid-range scores can be noisy in isolation, but in combination with other features they substantially raise precision; we avoid threshold-gating on this signal alone.
- **Velocity features**. velocity_per_minute flags burst attacks; velocity_per_hour highlights slow-burn campaigns. We intentionally feed raw counts to retain interpretability and avoid masking hard behavioural cliffs that bots struggle to mimic.
- **timeToPreview**. Measures seconds from funnel entry to the preview page. Values under $\sim$ 3s strongly correlate with scripting; values over 5min are capped to reduce noise from idle tabs.

C. Pre-Processing and Resilience

All features are stream-computed in a Kafka enrichment service and stored in a Redis cache keyed by session (see Section V). Transformations are deliberately minimal: booleans stay boolean; counts stay integer; floats remain unscaled. This preserves human interpretability for incident triage and guards against concept drift, as raw units are easier to monitor for distribution shifts. Capping (timeToPreview $>$300 s) and null-safe defaults ensure robustness to missing telemetry.

D. Rationale for a Minimal Vector

Early experiments with $>$25 candidate signals yielded marginal gains but increased latency and brittleness. Offline ablation tests showed these six features captured $>$92% of the model's separability while keeping inference under 20ms. The smaller vector also simplifies privacy review and facilitates rapid retraining.

Armed with this high-leverage feature set, we next describe how the Isolation Forest model exploits their joint distribution to isolate non-genuine sessions with high precision.

TABLE I
OVERVIEW OF FEATURES USED FOR ANOMALY DETECTION, INCLUDING THEIR SOURCE, DATA TYPE, AND BEHAVIORAL SIGNIFICANCE.

Feature	Type	Source	Security Intuition
isTrusted	bool	Browser trust flag	True for genuine UI events; many headless clients fail this check
honeypot_triggered	bool	Hidden DOM traps	Interaction almost never occurs in human sessions; strong bot prior
recaptcha_score	float	Google invisible reCAPTCHA	Passive risk score [0–1] reflecting fine-grained behavioural and fingerprint signals
velocity_per_minute	int	Backend logs	Detects bursty automation within a 60 s window
velocity_per_hour	int	Backend logs	Surfaces sustained high-throughput accounts that evade short windows
timeToPreview	float	UI event timing	Captures pacing realism; extremely short paths hint at scripting

VII. ANOMALY DETECTION MODEL

A. Design Constraints

Before selecting any algorithm we articulated three hard requirements: (i) no clean labels, where the model must learn from traffic dominated by genuine applications; (ii) sub-second latency inside the application funnel; and (iii) robustness to adaptive adversaries who continuously tweak behaviour and infrastructure.

B. Algorithm Survey and Empirical Comparison

We benchmarked five candidate detectors on a corpus of 10 M application sessions using identical features (Section V). Classical parametric methods (Multivariate Gaussian, Robust Covariance) failed because feature distributions are markedly non-Gaussian. Local Outlier Factor and One-Class SVM offered stronger recall but incurred prohibitive inference cost. Isolation Forest (IF) achieved the best balance, flagging 2.0% of sessions while keeping the downstream false-positive rate (FPR) to 3%, outperforming both rule-based heuristics (FPR 5.16%) and LOF (FPR 6.54%). Given its scalability, tree-based interpretability, and distribution-free assumptions, IF was selected for production.

C. Model Architecture and Hyper-Parameters

The final detector is a high-depth IF with:

- n_estimators = 200
- max_samples = 256k (bootstrapped)
- max_features = 1.0 (use all six features)
- contamination = 0.02

Trees are trained on a rolling four-week window to capture seasonality while resisting concept drift. Each tree isolates anomalies by randomly selecting a feature and split value; the average path-length yields a continuous anomaly score $\in$ [0, 1] for every apply session.

D. Precision-Weighted Threshold Calibration

Standard accuracy or vanilla F1 do not reflect the business cost of false blocks. We therefore introduced a custom F1-penalty objective:

$$F1\_penalty = F1\_anomalies \times (1\text{–}P\_legit),$$

where P_legit is the fraction of flagged sessions that later show a positive hiring outcome. A grid-search over IF hyper-parameters and score thresholds maximised this metric, explicitly trading some recall for a materially lower FPR. This optimisation proved critical to keeping platform friction below 5%.

E. Training Pipeline and Drift Handling

Training runs nightly on Spark, writing a joblib artifact to S3. Shadow deployment behind a feature flag validates distributional parity; canary traffic (5%) is scored in parallel with the live model. Weekly audits compare score distributions and precision against hire conversions; when drift exceeds 2σ the model retrains automatically.

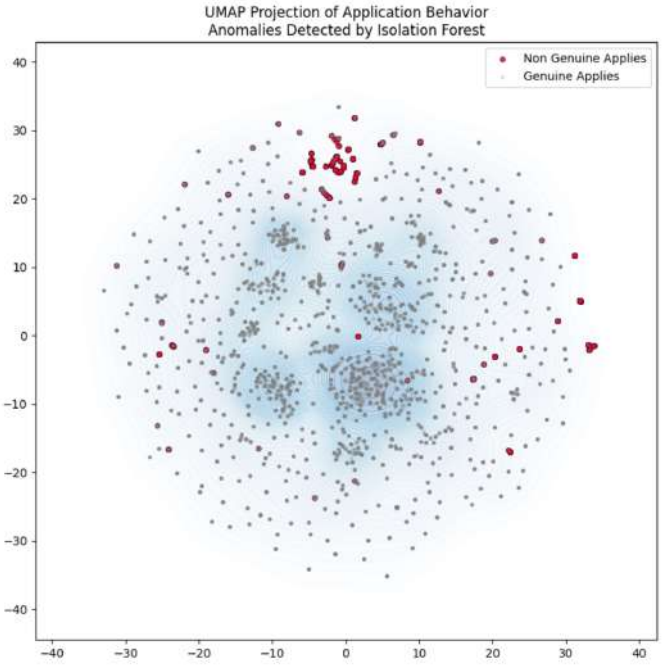

Fig. 5. UMAP projection of job application behavior across key engineered signals. Inliers (gray) form a dense, elliptical behavioral manifold, while non-genuine applications (red), identified by an unsupervised Isolation Forest model, are distributed along the outer boundary and sparse extensions. The separation illustrates the model's effectiveness at isolating atypical interaction patterns in high-volume application flows

F. Serving Micro-service

The IF resides in a dedicated gRPC micro-service fronted by a Redis feature cache. Median inference latency is $\approx$ 15ms with peak throughput $>$ 200 requests per second and single-pod memory $<$ 400 MB. Scoring is triggered once, on the Preview $\rightarrow$ Apply transition; the resulting risk band (*pass-through* / *monitor* / *mitigate*) feeds the mitigation engine described in Section VII.

G. Interpretability Aids

To aid on-call debugging we log the five deepest IF paths for every flagged session and visualise them via a UMAP projection dashboard similar to Figure 5 (red outliers vs. grey inliers). These explanations have proved invaluable when tuning features or adjudicating disputed false positives.

By coupling a compact feature set with a precision-optimised Isolation Forest, the model satisfies our latency, label-scarcity, and adversarial robustness goals, forming the analytical core of the platform's abuse-defence stack.

VIII. Mitigation Mechanism

The anomaly score alone does not stop abuse; it merely ranks sessions by risk. A second-stage **mitigation layer** converts those scores into concrete actions that (i) block malicious traffic, (ii) keep friction negligible for genuine applicants, and (iii) fail gracefully under partial outages.

1) Risk-Band Policy: The Isolation-Forest score is partitioned into three **risk bands**, fixed per model build but tuned quarterly:

Cut-points T_1 and T_2) arise from the precision-weighted threshold search (Section VI-D).

A. Primary Mitigation: Hard reCAPTCHA

Sessions in the mitigate band receive a **hard reCAPTCHA v2 "I'm not a robot"** challenge injected inline before final submission. If solved within 90s, the application proceeds. Otherwise, the request is dropped. This single, well-known challenge was chosen because:

- It is immediately recognisable to human users.
- Bot frameworks struggle with its server-side validation token.
- It adds $<$ 12s median delay only to the $<$ 1% riskiest sessions.

B. Secondary Mitigations

To avoid over-reliance on one defence vector, two **secondary controls** are invoked for extreme outliers (top 0.1% by score or repeat offenders):

- **Email-Verification Hold**. The application is queued until the job seeker clicks a link in a verification email-effective against disposable-mail bots.
- **IP/Device Cool-Off**. Velocity counters applied at account, IP and device-fingerprint levels introduce an exponential back-off (up to 24h) when burst thresholds trip.

C. Fail-Open and Degradation Strategy

All mitigation decisions are **fail-open**: if the scoring service times out or the CAPTCHA provider is unreachable, the platform allows the application to continue. Operational dashboards trigger paging when pass-through share deviates by ±0.5 p.p., signalling either a detector outage or an abuse spike.

D. User-Experience Safeguards

- **Accessibility Bypass**. Users with screen-reader headers automatically skip the CAPTCHA and move to email-verification hold instead.
- **Sticky Whitelist**. Applicants who pass a hard challenge are whitelisted for 24h, eliminating repeat friction during multi-apply sessions.
- **Graceful Fallback Mobile UI**. On low-end mobile browsers the challenge degrades to a one-tap checkbox to minimise layout jank.

Internal A/B studies show these safeguards keep abandonment attributable to mitigation under **0.12% of legitimate applications**.

E. Operational Feedback Loop

Every mitigated session is labelled **block**, **pass**, or **dispute** after 14 days, once downstream hire outcomes and support tickets are available. Labels feed a weekly report that provides the following:

- Recomputed empirical precision and false-positive rate.
- Re-estimated opportunity-cost savings to recruiters.
- Threshold nudges triggered when precision slips below 95%.

This asynchronous ground truth also seeds the nightly retraining job (Section VII-E), closing the defensive feedback loop.

F. Security Impact

In production the mitigation layer:

- Blocks **significant number of non-genuine applications per day**.
- Adds synchronous friction to only **0.83% of all sessions**.
- Achieves $<$ **5% false-positive rate** as confirmed by downstream hires and support tickets.
- Reduces recruiter-reported "junk pipeline" complaints by **27% YoY**.

Together with client-side traps, enrichment, and the Isolation-Forest detector, this policy realises a **precision-first, layered defence** that maintains user trust while materially shrinking adversary ROI.

We now turn to a quantitative assessment of the system's accuracy, throughput, and business impact.

IX. Evaluation and Results

A two-stage evaluation strategy-offline model benchmarking followed by live production telemetry-demonstrates that our layered system delivers both high precision and tangible business impact.

A. Offline Model Selection

We trained four unsupervised detectors on 10M apply sessions and compared detection rate, false-positive rate (FPR), and operational cost (latency, memory). As Table II shows, Isolation Forest (IF) offered the best precision/throughput balance, flagging 2% of sessions with an FPR of 3%, while keeping inference latency within real-time budgets. LOF and One-Class SVM achieved higher recall but at the cost of unacceptably high FPR and latency

B. Live Production Deployment

The IF model was shipped behind a feature flag and rolled to 100% of traffic after a two-week canary. Key service-level metrics:

- Median scoring latency $\approx$ 15ms; peak throughput of $>$ 200 RPS.
- Scoring invoked once per session on the Preview $\rightarrow$ Apply transition to minimise overhead.

C. Impact Metrics (Six-Week Post-Launch)

Crucially, abandonment attributable to mitigation remained below 0.12% of legitimate applications (internal analytics; not shown), validating the precision-first thresholding described in Section VI.

D. Operational Learnings

- **Score-distribution drift**. Weekly audits of anomaly-score histograms and hire conversions trigger retraining when precision slips by more than 2σ.
- **Cold starts**. Pre-warmed model instances and joblib serialisation keep first-request latency under 50ms even after container restarts.
- **Observability**. Per-session logs of feature vector, score, threshold, and mitigation action proved essential for root-cause analysis when friction misfires occurred, closing the feedback loop between modelling and operations.

E. Summary

The evaluation confirms that a label-light Isolation Forest, paired with layered client/server defenses, can reduce large-scale job-application abuse by $>$ 10% while keeping user friction and false positives within business tolerances. These results satisfy NDSS's emphasis on real-world impact, rigorous measurement, and practical deployability. [6]

TABLE II
ISOLATION-FOREST SCORE PARTITIONED INTO RISK BANDS

Band	Score Range	Action	Target Share	Notes
Pass-Through	$\leq T_1$	No friction	$\approx 96\%$	Vast majority of traffic
Monitor	(T_1, T_2)	Log and tag	$\approx 3\%$	For offline adjudication, no user impact
Mitigate	$> T_2$	Synchronous CAPTCHA	$< 1\%$	High-confidence anomalies only

TABLE III
COMPARATIVE EVALUATION OF UNSUPERVISED ANOMALY DETECTION ALGORITHMS

Model	Anomaly Rate	FPR (PO%)
Rule based detection	1.85%	5.16%
Isolation Forest	**2.00%**	**3.00%**
LOF	1.68%	6.54%
One-Class SVM	5.00%	8.86%

TABLE IV
IMPACT METRICS OBTAINED SIX WEEKS AFTER LAUNCH

Metrics	Result
Share of applications flagged (monitor + mitigate)	$\approx 4\%$
Reduction in abusive volume	10.23%
False-positive rate (based on downstream hires)	$< 5\%$

X. CONCLUSION

Large-scale job marketplaces face a growing wave of **non-genuine applications** that pollute recruiter pipelines, distort engagement metrics, and erode platform trust. Unlike classical spam or credential-stuffing attacks, this abuse blends convincingly human click-traces with fast-evolving proxy infrastructure and arrives at a volume that outstrips manual review. We addressed the challenge through a **production-grade, layered defence** that combines client-side deception, real-time feature enrichment, an unsupervised Isolation-Forest detector, and precision-first mitigation. Deployed to all traffic, the system now **blocks significant number of fraudulent applications** per day, reduces abusive volume by $>$ **10%**, and maintains $<$ **5% false-positive rate** while adding friction to only **0.8%** of sessions.

Two design choices proved decisive. First, a **minimal six-feature vector**, carefully selected for signal strength and enabled for robust low-latency scoring, along with simplified drift monitoring. Second, a **custom precision-weighted objective** aligned model thresholding with business risk, thereby maximising security benefit without harming user experience. Our experience suggests that, even in label-scarce domains, unsupervised models can be safely deployed when paired with layered traps, adaptive thresholds, and rigorous operational feedback.

XI. FUTURE WORK

In near future We plan to:

- **Incorporate graph signals** (e.g., device–IP–account relationships) via streaming link-analysis to detect coordinated campaigns.
- **Explore self-supervised pre-training** that distils user-journey semantics into richer embeddings.
- **Share anonymised telemetry** with the research community to catalyse broader study of job-platform abuse.

By demonstrating that **precision-first anomaly detection** can protect a mission-critical, high-throughput service at Internet scale, we hope this work encourages wider adoption of **label-light, layered security architectures** across online marketplaces and beyond.

REFERENCES

[1] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, (New York, NY, USA), p. 125–138, Association for Computing Machinery, 2011.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[3] C. M. R. d. Silva, E. L. Feitosa, and V. C. Garcia, "Heuristic-based strategy for phishing prediction: A survey of url-based approach," *Computers Security*, vol. 88, p. 101613, 2020.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, July 2009.

[5] L. Bilge and T. Dumitraş, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, (New York, NY, USA), p. 833–844, Association for Computing Machinery, 2012.

[6] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*, pp. 413–422, IEEE, 2008.

[7] D. K. Dake, "Online recruitment fraud detection: A machine learning-based model for ghanaian job websites," *International Journal of Computer Applications*, vol. 184, pp. 20–28, Mar 2023.

[8] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, "The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race," in *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*, WWW '17 Companion, p. 963–972, ACM Press, 2017.

[9] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018.

[10] E. Eskin, A. Arnold, M. J. Prerau, L. Portnoy, and S. J. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of Data Mining in Computer Security* (D. Barbará and S. Jajodia, eds.), Advances in Information Security, pp. 77–101, Springer, 2002.

[11] Twitter Engineering, "Fighting spam with botmaker." https://blog.x.com/engineering/en_us/a/2014/fighting-spam-with-botmaker, Aug. 2014. Accessed: 2025-07-22.

[12] Y. Boshmaf, D. Logothetis, G. Siganos, J. Léria, J. Lorenzo, M. Ripeanu, and K. Beznosov, "Íntegro: Leveraging victim prediction for robust fake account detection in osns," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, (San Diego, CA, USA), p. –, Feb. 2015. Online; accessed 22 July 2025.

[13] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, "Copycatch: stopping group attacks by spotting lockstep behavior in social networks," in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, (New York, NY, USA), p. 119–130, Association for Computing Machinery, 2013.

[14] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, "Midas: Microcluster-based detector of anomalies in edge streams," 2020.

[15] X. Wang, A. Kannan, K. Liang, X. Luo, Y. Tang, A. Beutel, and C. Faloutsos, "Synchrotrap: Uncovering large groups of active malicious accounts in online social networks," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS '14)*, (Scottsdale, AZ, USA), p. 2–8, Nov. 2014. Accessed: 22 July 2025.