Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

DoS/DDoS Detection and Mitigation System for E-Commerce Platforms

1st Gopika Ratnam Dasari
Department of Computing Technology SRM Institute of Science and Technology Chennai, India

2nd Ram Prakash Alapati
Department of Computing Technology SRM Institute of Science and Technology Chennai, India

***Abstract* - CAPTCHAs will prevent bots but punish legitimate users when used out of context. PIA-Shield minimizes the exposure to CAPTCHA by putting challenges on continuous behavioral risk instead of uniformly applying them. The front-end sends fine grained telemetry, such as product views, clicks, scroll activity, timing and session metadata, to a back-end which aggregates these signals into a single risk score of 0 to 100 with a weighted model. Patterns of repetitive interactions, such as repeated viewing of the same object, and a lengthy period of inactivity, are mapped to four risk bands, i.e., human, medium, high, and bot. The severity of risk is mitigated, with low-risk sessions running uninterrupted, medium and high-risk sessions only subjected to a slider CAPTCHA on sensitive operations like checkout, and bot-identified sessions redirected to a sandbox. The successful CAPTCHA completion reduces the score of the session, which represents recovered trust. Telemetry ingestion and session state are implemented on Redis, scoring and mitigation logic are implemented on a FastAPI service and decision logging is implemented on SQLite to support auditing. Reserving CAPTCHAs during ambiguous or high risk sessions, PIA-Shield shows that behavior based scoring can significantly reduce the CAPTCHA friction and enhance bot detection, with consequences to e-commerce, account security and usable anti abuse systems.**

***Index Terms* - CAPTCHA, Bot detection, Usable security, Session telemetry, Risk-based mitigation, E-commerce security.**

I. INTRODUCTION

CAPTCHAs are common to differentiate between human and automated clients and to curtail abuse like credential stuffing, inventory scraping, and checkout frauds [1], [2]. Users are periodically challenged to solve image, slider, or puzzle puzzles prior to undertaking sensitive tasks. These obstacles have an established price, since they slack down valid users, introduce accessibility obstacles, and may drive individuals to leave flows or evade highly guarded websites [3], [4]. Good bot protection is important, as hackers are continually improving at the sensitive threshold or using crude rules like IP reputation and rate limiting that can be readily circumvented and often punish the innocent user. A better solution is to challenge only when needed by applying continuously, behavior based signals to approximate session risk and to present a CAPTCHA only when the risk is high.

This paper presents PIA-Shield, a system that uses behavior signals — repetitive interaction, inactivity, and session meta-data — to compute a single risk score for a session. The system maps the score to four risk bands (human, medium, high, bot) and applies tiered mitigations: allow, slider CAPTCHA, or sandbox redirect. It also introduces a forgiveness mechanism that reduces risk after a successful CAPTCHA completion, reflecting regained trust. We describe the architecture, scoring model, implementation details, and experimental evaluation.

II. RELATED WORK

CAPTCHAs, usability, and evolution. CAPTCHAs were implemented to differentiate between humans and robots based on tasks that machines are presumed to be more difficult than humans [1]. CAPTCHA Text and image recognition CAPTCHAs were the most frequently used in early applications, but attracted long-term criticism on accessibility and usability [2], [3]. reCAPTCHA and others moved to low friction models, such as checkbox and no CAPTCHA models, which fall back to image or audio challenges when the confidence is low [4]. CAPTCHAs that are based on slider or interaction minimize the use of vision or text, but still result in an additional step and can be automated or crowdsourced [5], [6]. The major drawback here is that the decision to display a challenge is usually made based on crude indicators like IP reputation or rate and thus a high number of legitimate users are challenged.

Bot detection and behavioral signals. Detection of bots has shifted to single challenges to multi signals. Request rate, headers and TLS fingerprinting are some of the traffic and request patterns that are used to identify automated or scripted behavior [7], [8]. JavaScript and browser APIs can be used on the client to capture mouse movements, keystroke timing, scroll movement and touch pattern to create behavioral profiles [9], [10]. They have been applied in fraud detection in banking and e-commerce and abuse prevention in account systems. They can be aggregated by machine learning models to categorize the sessions as human or bot [11], [12]. PIA-Shield pursues this direction, but it applies a explicitly simple

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

rule based scorer with interpretable bands which are human, medium, high and bot instead of a learned classifier. This decision makes the system understandable and simple to tune to any application. One of the future directions of work is the extension of the pipeline with learned models.

III. PROBLEM STATEMENT

A. High false positives. Most systems consider benign users to be suspicious and subject them to additional verification or blocking. Triggers can be rough, such as common IPs, VPNs, or high speed but legitimate actions, thus low risk human beings are often incorrectly identified as a bot [1], [2]. False positives undermine trust, load of support and may refer users to competitors. To eliminate the avoidance of unnecessary challenges, it is necessary to move beyond single easily gamed signals to richer session level behavior.

B. User experience degradation. CAPTCHAs and blocks are usually used in a consistent way, i.e., all people are challenged at sensitive points or all people with a suspicious characteristic are. The outcome is frustration to large crowds of legitimate users, longer lines, increased abandonment, and accessibility issues among users with disabilities [3], [4]. In cases where security measures are not tied to the risk, the experience of the users is not enhanced in proportion to the security value.

C. Lack of intent-aware detection. The majority of defenses consider the risk in terms of a limited number of signals, such as IP, rate, or a single challenge, without modeling the behavior of the user over time. Legitimate intent is observed in the patterns of interaction that occur naturally e.g. browsing, stopping, comparing objects and then making a sensitive action. Bots are inclined to rigorous or repetitive behavior. Unconsciously, behavior based measurement, systems are unable to distinguish human like browsing and automation, and false positives and bots are missed [5].

D. Binary decision models. Decisions tend to be binary, i.e., permit or prohibit, or contest or not contest. No halfway covenant like permit but check at checkout only or challenge. Such inflexibility directly leads to needless friction on the part of actual users and crude reactions to abuse [6].

IV. PROPOSED SYSTEM: PIA-SHIELD

A. System Overview

PIA-Shield is a purchase intent conscious bot detection system. It gathers session telemetry on the front end, grades each session, and classifies risk into four categories, and implements one of four mitigation policies, which include transparent access, soft CAPTCHA, strict CAPTCHA, or sandbox isolation. Low risk users are not challenged by a CAPTCHA, and only medium or high risk sessions are challenged at sensitive actions, such as checkout, and bot classified sessions are redirected to a sandbox. Recovered trust is indicated by a successful CAPTCHA completion. The design aims at low false positives and improved usability at the expense of maintaining a good level of protection against high risk and bot traffic.

B. Architecture Design

The system is divided into four layers i.e. frontend, backend risk engine, session storage, and logging or monitoring.

Frontend layer. An example of a web storefront is React, which gathers user behavior and transmits it to the backend. Each meaningful action has its telemetry events including view, add to cart, checkout, heartbeat, and interaction update with session id, event type, product id (where applicable), timestamp, dwell time, and page depth. The front end will periodically retrieve the risk score in place and, at sensitive actions, invokes the mitigation endpoint. In the case of a slider CAPTCHA, the client generates the challenge, captures the slider offset and mouse path of the user, and resubmits. Human drag is characterized by trajectory and timing as opposed to instant or scripted movement. The front end may also be diverted to a sandbox where the backend determines the session to be bot.

Backend risk engine. Three endpoints are exposed by a FastAPI service, one of them being a telemetry ingestion endpoint, which takes in events and optional metadata, another being a scoring endpoint, which provides the current risk score and feature breakdown, and a mitigation endpoint, which provides an action, which may be allow, slider, or sandbox, and in the case of the slider, challenge parameters or verification results. The risk engine incorporates a weighted scorer, which is a combination of baseline trust, repetitive product view penalties, inactivity based penalties and a success mechanism score on CAPTCHA completion. In the case of slider responses, a different verification module compares position accuracy and trajectory to a given solve and logic implementing score reduction.

Session storage layer. Each session has telemetry events attached to a Redis list with the key of session id and a time to live of one hour to limit retention. The metadata of the session, such as the number of last actions, the last action time, the target of the challenge, and the post CAPTCHA score, are stored as a JSON under a different key. This maintains scoring and mitigation state-less as requested and permits the engine to make use of the recent session history.

Logging and monitoring layer. Monitoring and logging layer. All mitigation choices (allow, slider required, slider verified, slider failed, and sandbox) are persisted to SQLite in the form of session id, action, risk label and a field of detail in the form of a JSON. This facilitates auditing, analytics and threshold tuning.

C. Purchase-Intent Aware Scoring Model

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

The score is calculated based on the metadata in the stored telemetry and session. It begins with a base and then imposes penalties on frequent product views and inactivity, which results in a single numeric score of 0 to 100 which is plotted on risk scales and mitigation measures.

Baseline trust initialization. The initial score of a session is 25 out of a 0 to 100 scale, which is the supposition of human unless proved otherwise. Penalty logic does not include heartbeat and interaction update events. When no other user events and no inactivity penalty, the session has a 25 and is considered to be low risk.

Repetitive product view detection. The scorer checks the past n product view events and counts the number of consecutive views which point to the same product id. Repeat consecutively is punishable in the following manner. Two successive views sum up 25. Three add 50. Four add 65. Five or above add 75 and are considered as strong bot evidence. Five or more consecutive product views can score 100 and be considered bot.

Inactivity-based risk accumulation. Sessions that have no user activity over a long period are considered to be more risky. The scorer keeps in session metadata the number of the last user action and the time. At every scoring call, the number of actions that have not been recorded since the last timestamp is then calculated and a penalty of 25 is added to the time elapsed since the last stamp of the previous action. Upon the user re-action, the count and the time are updated and there is no inactivity penalty during that period.

Score reduction (forgiveness model). Upon the successful completion of the slider CAPTCHA by a user in a medium or high-risk session, the system deducts the current score by 40 which is limited to 0. The new value is recorded in session metadata, and used later in making decisions, permitting the session to continue with less friction once it has been verified.

Risk classification thresholds. The numerical result is then associated with four risk levels with 0 to 30 as human, 31 to 60 as medium, 61 to 90 as high and 91 to 100 as bot.
Adaptive mitigation strategy. Low risk (human): transparent access, no CAPTCHA. Medium risk: slider CAPTCHA only at sensitive actions. High risk: same slider CAPTCHA at sensitive actions. Bot: sandbox isolation.

D. System Workflow

User session lifecycle. The start of a session is when the user opens the storefront. The front end either builds or receives a session id that is unique and utilizes it in all further requests. The front end emits telemetry events to the backend as the user browses, adds items, and approaches the checkout and periodically retrieves the risk score. The front end makes a call to the mitigation endpoint when the user tries a sensitive action. Based on the response, the session continues, is issued a slider CAPTCHA or redirected to the sandbox. Successful CAPTCHA solution results in the forgiveness model, and unsuccessful attempts may be retried or escalated. The session data is expired after an hour and then telemetry and metadata are disposed of.

Telemetry data collection. The front end transmits events to the backend through one telemetry endpoint. Each payload includes event type, timestamp, event metadata, product id (when applicable), and dwell time, page depth, or cart status. Events are appended to a Redis list and merged with session metadata. Only events representing user actions — excluding heartbeat and interaction update — feed the repetitive-view and inactivity logic.

Risk score computation. Risk score computation is triggered in response to a dedicated scoring endpoint or as part of the mitigation endpoint. The backend loads all telemetry events and session metadata from Redis and computes the score using the rules described above: baseline initialization, inactivity penalty from metadata, repetition penalty by scanning events in reverse order, and mapping to risk levels. The result — score, risk level, and optional feature breakdown — is returned.

Mitigation decision pipeline. When the client calls the mitigation endpoint, the backend computes or fetches the current score and risk level and returns an action. For low risk, the action is allow. For medium or high risk, if the request is not a sensitive action, the action is allow; otherwise the action is slider. For bot risk, the action is sandbox. In case a slider is needed, the client signs the challenge and provides verification data. The backend implements the forgiveness reduction and gives allow on a successful verification. When it fails, it can either request a retry or escalate. All the results are audited to SQLite.

V. IMPLEMENTATION DETAILS

A. Technology Stack

Frontend: Frontend: React 18 powered by Vite and Axios to communicate with an HTTP. UI takes care of the session state, telemetry, fetches the risk score, and displays the slider CAPTCHA where necessary. Backend: Python and FastAPI, using Pydantic to validate requests and responses, and CORS is on. The risk engine is written in Python and generation and verification of slider CAPTCHA is done in a separate module. Session state: Redis set up through REDIS URL with TTL of 3600 seconds. Logging: SQLite with SQLAlchemy ORM, in which each mitigation decision gets a row in a decisions table.

B. API Design

GET /health returns {status: ok).
POST /telemetry takes a telemetry event in the form of a JSON and optional metadata and stores the event in Redis and responds with 201 or 400.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

GET /score/{session id} will give the current risk score normalized to a 0-1 range, the risk level, and will include such features as total score, per signal score, and event count.
POST /mitigate/{session id} takes an optional body with a sensitive action flag, slider offset, trajectory, and challenge information. It gives back an action, which may be allow, slider, or sandbox, and detail and risk. It throws a 404 when the session has no events.
POST /simulate takes in humans, bots and a seed, creates synthetic event streams, executes the scorer and gives back precision, recall, false positive rate and notes.

C. Data Storage Mechanism

Redis stores session events as a list with key {session:session id}} and RPUSH to add to the end and LRANGE to read with TTL of 3600. The metadata of the sessions is also stored in Redis under the key {session metadata:{session id}} in a TTL of 3600 and in a JSON format. SQLite stores the decision log in a table called decisions having fields id, session id, action, risk, detail as JSON, and created at, where there is a row per mitigation decision.

D. Real-Time Score Computation Algorithm

Inputs: events list and session metadata from Redis.

1. Set total score = 25; build user events by dropping event type in {heartbeat, interaction_update}. 2. Compute inactivity penalty (25 per full minute of inactivity based on metadata); update metadata. 3. If len(user_events) 2, scan view events in reverse order, count longest run of consecutive same product_id; add penalty: 2→25, 3→50, 4→65, 5+→75. 4. Map total score: 0–30 human, 31–60 medium, 61–90 high, 91–100 bot. 5. On successful CAPTCHA: new score = max(0, current score 40); store in metadata.
Complexity is linear in the number of events per session.

VI. EXPERIMENTAL EVALUATION

A. Test Scenarios

Normal user simulation is a session with mixed product views, dwell times of 2-10 seconds that are realistic and natural pacing. The desired result is a lower score than 30, mitigation allow and no CAPTCHA. False positive rate and user experience impact are measured using this scenario.

Bot scraping simulation. Sessions with multiple view events on the same product id, short dwell (0.1 s), rapid timestamps. Five or more consecutive same-product views trigger the maximum repetitive-view penalty; score ¿ 90 (bot), mitigation sandbox. Used to measure bot detection.

Inventory hoarding simulation. Sessions with rapid or bursty views repeating the same product, with varying run lengths and inactivity. Used to observe when the system classifies as medium/high (CAPTCHA) vs. bot (sandbox), and to tune thresholds and report precision/recall under mixed behavior.

B. Performance Metrics

Detection accuracy: Precision = TP/(TP+FP) and Recall = TP/(TP+FN) for the bot class, reported by scenario. False positive rate: proportion of human-like sessions that receive CAPTCHA or sandbox instead of allow. Response latency: time from client request to response, reporting median and optionally p95/p99. User experience impact: fraction of sessions that see CAPTCHA or sandbox at least once and forgiveness effectiveness (fraction of challenged sessions that receive allow after one successful CAPTCHA).

C. Comparative Analysis

Always-on CAPTCHA: PIA-Shield challenges only medium/high risk at sensitive actions; compare friction rate for normal users and bot detection rate. Binary block/allow: compare false positive rate and friction under binary allow/block vs. PIA-Shield's three-tier policy. The three test scenarios are compared in intent agnostic detection where purchase intent aware signals like repetitive views and inactivity are used by PIA Shield to distinguish between human-like browsing and scraping and hoarding.

VII. RESULTS AND ANALYSIS

A. Risk Score Distribution

Normal user sessions with different browsing and no repetition over time are clustered in the human band with a score of 0 to 30 with a baseline of 25 and no penalties to maintain the score at 25. The highest penalty and a range of 90 to 100 is given to bot sessions that have five consecutive same product views. Two to four consecutive same product views or prolonged inactivity are considered as medium or high. Under controlled experiments, it is bimodal, with the human band at the peak of normal users and high end at clear bots.

B. Bot Detection Effectiveness

In the scraping case where there are five or more consecutive views of the same product, all such sessions are identified as bot with a 100 percent recall in simulation. Precision relies on false positives, with human-like sessions with product views different and without long repetition are retained in the human band, leading to high baseline precision. In the case of inventory hoarding, the pattern is used to detect the inventory hoarding, with long same product runs being detected with a high recall, and mixed product hoarding being detected with CAPTCHA rather than sandbox.

C. CAPTCHA Reduction Rate

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

CAPTCHA reduction rate is the percentage of sessions successfully executing a sensitive action without ever encountering a CAPTCHA, that is, the percentage that stay in the human band and are allowed. This rate stands at 100 percent in case of the normal user traffic as opposed to an always CAPTCHA baseline. In the case of mixed traffic, the rate is the ratio of the number of sessions that are human at the moment of the sensitive action.

D. System Scalability

Per session cost O(n) in the number of events and a typical session has a limited number of events in the one hour TTL. Redis round trips and in memory scoring dominate latency of score and mitigate endpoints and responses in the tens of milliseconds range with typical session sizes. The backend is stateless and can be scaled to throughput by adding API instances, whereas Redis and SQLite can be tuned or swapped to more volume.

VIII. SECURITY AND PRIVACY CONSIDERATIONS

Behavioral data usage. Telemetry is only used to the extent required to detect bots and to mitigate risks, and raw events are not utilized to advertise or to do secondary analytics. Events are retained up to the session TTL. Operators are advised to encrypt Redis and the backend and not to store sensitive attributes in telemetry.

Session expiry. An hourly Redis TTL will be used to automatically remove session information, and returning users will be given a new starting score. Decision logs can be stored longer and thus retention policy must be established and published.

Transparency of risk score. It has a rule-based scoring logic, which can be inspected and comprises of a baseline and penalties when repeated and inactive. Operators are able to provide reasons why a session was assigned a certain risk band. Such user facing messages as additional verification required may be added without exposing specific thresholds.

Ethical implications. Behavioral signals may correlate with context (slow networks, accessibility tools); operators should monitor false positive rates across user segments and tune thresholds accordingly. The allow, CAPTCHA, or sandbox of inferred risk is mitigated in a tiered way. The site should provide information that it operates under behavioral signals to prevent abuse, and consent and notice must be in line with the relevant legislation.

IX. RESEARCH GAPS ADDRESSED

Lack of purchase-intent modeling. PIA-Shield interprets session telemetry — repetitive views, inactivity, browsing diversity — to infer intent and separate human shoppers from scrapers and hoarders.

Over-reliance on CAPTCHA. PIA-Shield gates CAPTCHA on risk; only medium- and high-risk sessions are prompted at sensitive actions; low-risk sessions never see a CAPTCHA.

Static threshold systems. PIA-Shield uses configurable, interpretable thresholds (human/medium/high/bot) that can be tuned based on observed false positives and the forgiveness mechanism adapts session risk over time after a successful CAPTCHA.

No adaptive mitigation ladder. PIA-Shield implements a four-level response — allow, slider CAPTCHA at sensitive actions, sandbox — replacing a single binary allow/block decision.

X. LIMITATIONS

Rule-based scoring. The current model is hand-tuned and cannot capture complex or nonlinear patterns. Bots that understand the rules can stay just below bot thresholds. In the future, machine learning will be incorporated to make it more discriminative and automatically tune the threshold.

Limited attack vector coverage. Detecting is most effective with scraping of repetitive same product views and simple automated flows. Advanced human-like bots, distributed scraping, account level abuse, and low and slow strategies need to be covered and analyzed at the cross session.

No cross-session behavioral modeling. Risk is calculated on a per-session basis, therefore, bots can re-churn sessions to reset risk. Persistent identifiers or aggregates and increased privacy and retention concerns would be needed in cross session modeling.

XI. FUTURE WORK

Machine learning integration. Substitute or complement the rule based scorer with a classifier that is trained on labeled sessions to enable automatic threshold tuning and increased resistance to new bot strategies, and maintain interpretability with feature importance or surrogate models.

Cross-session bot correlation. Maintain anonymous aggregates per identifier — count of recent sandboxed sessions, challenge rate — and use these to boost risk across sessions without centralizing raw telemetry.

Federated learning for privacy preservation

Train a common risk model on many sites or tenants without centralizing raw telemetry, with each party computing local updates and only model parameters or gradients are shared, leaving data at the origin.

Adversarial bot resistance.

Add signals that are more difficult to spoof such as browser or device consistency, TLS fingerprinting, and server side trajectory validation. Add controlled randomness in thresholds or challenge choice, add adversarial training with the addition of machine learning, and track detection rates to initiate retraining or new indicators when bots evolve.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 08 , August - 2026

XII. CONCLUSION

This paper discussed the PIA Shield which will decrease the friction of CAPTCHA on the legitimate users and will maintain a high level of mitigation to the bots and abusive traffic. The premise behind it is that the vast majority of cases are either human or bot, and only border cases need to be challenged. The system does not trade off between protection and usability since it scores session behavior continuously and plots the score on a tiered mitigation, which consists of allow, CAPTCHA, and sandbox. Difficulties will not come unless the risk is worth it and a successful solve will reduce the score of the session to ensure that the user is not constantly interrupted.

Real constraints include the rule based scoring, session only modeling, and limited coverage of attacks which inspire future work on machine learning integration, cross session correlation, federated learning, and adversarial resistance. In the case of e-commerce, where each superfluous CAPTCHA may result in abandonment, it is necessary to find the appropriate balance between friction and protection.

ACKNOWLEDGMENT

The authors acknowledge the Department of Computing Technology, SRM Institute of Science and Technology, Chennai, India, as the source of computational resources and academic support of this research.

REFERENCES

[1] M. Hemmampour, C. Zheng, and N. Zilberman, “E-Commerce Bot Traffic: In-Network Impact, Detection, and Mitigation,” Proc. 2024 IEEE 27th ICIN, 2024.
[2] S. Rovetta, G. Suchacka, and F. Masulli, “Bot recognition in a Web store: An approach based on unsupervised learning,” J. Netw. Comput. Appl., vol. 157, p. 102577, May 2020.
[3] M. Gajewski et al., “Data-driven human and bot recognition from web activity logs based on hybrid learning techniques,” Digital Commun. Netw., vol. 10, no. 4, pp. 1178–1188, Aug. 2024.
[4] A. Acien et al., “BeCAPTCHA: Behavioral bot detection using touch-screen and mobile sensors benchmarked on HuMIdB,” Eng. Appl. Artif. Intell., vol. 98, Feb. 2021.
[5] J. Kacel et al., “BOTtrace: A framework for Discriminating Bots and Humans,” in Proc. ESORICS 2024 Workshops, 2024.
[6] M. Uysal, “Revisiting Text-Based CAPTCHAs: A Large-Scale Security and Usability Analysis Against CNN-Based Solvers,” Electronics, vol. 14, no. 22, 2025.
[7] J. M. Llamas et al., “Balancing Security and Privacy: Web Bot Detection, Privacy Challenges, and Regulatory Compliance under the GDPR and AI Act,” Open Res. Eur., vol. 5, 2025.
[8] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, “Botnets: A survey,” Computer Networks, vol. 57, no. 2, pp. 378–403, 2013.
[9] N. Zheng, A. Paloski, and H. Wang, “An efficient user verification system via mouse movements,” in Proc. ACM CCS, 2013, pp. 139–150.
[10] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song, “Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication,” IEEE TIFS, vol. 8, no. 1, pp. 136–148, 2013.
[11] R. Bolton and D. J. Hand, “Statistical fraud detection: A review,” Statistical Science, vol. 17, no. 3, pp. 235–249, 2002.
[12] G. Starnberger, L. Froihofer, and M. Goeschka, “CRHM: A hybrid approach for detecting automated browser interactions,” in Proc. IEEE ARES, 2009, pp. 464–469.
[13] P. A. Grassi, J. L. Garcia, and M. E. Fenton, “Digital identity guidelines: Authentication and lifecycle management,” NIST SP 800-63B, 2017.
[14] S. Bonneau, “Risk-based authentication: How to reduce friction and improve security,” Gartner, 2019.
[15] S. Egelman and E. Peer, “Scaling the security wall: Developing a taxonomy of the obstacles that end-users encounter when trying to implement security advice,” in Proc. IEEE S&P Workshops, 2016, pp. 111–120.
[16] A. Beautement, M. A. Sasse, and M. Wonham, “The compliance budget: Managing security behaviour in organisations,” in Proc. ACM CCS, 2008, pp. 1–10.