Preventing Artificially Inflated SMS Attacks through Large-Scale Traffic Inspection

Jun Ho Huh
Samsung Research

Hyejin Shin
Samsung Research

Sunwoo Ahn
Samsung Research

Hayoon Yi
Samsung Research

Joonho Cho
Samsung Electronics

Taewoo Kim
Samsung Electronics

Minchae Lim
Samsung Electronics

Nuel Choi
Samsung Electronics

Artificially inflated traffic (AIT) attacks have become a prevalent threat for businesses that rely on SMS-based user verification systems: attackers use bot accounts to initiate intense volume of artificial SMS verification requests. Malicious telecommunication service providers or SMS aggregators are potential cheating entities. To date, however, there is no published literature formally characterizing AIT attacks or investigating attack detection techniques. Several online blogs provide traffic volume inspection suggestions without revealing implementation details and attack data. We bridge this gap, and for the first time formally characterize AIT attack techniques based on a large-scale dataset consisting of 9.4 million SMS request logs: our analysis reveals that attacks often use short-lived email services, and reuse common prefix values to rapidly generate unverified phone numbers and IMEI numbers. To bypass rate limit policies, bots are programmed to submit a few requests before switching to a different account, phone number or device. This distributed nature of the attack makes detection based on naive historical-event inspection extremely challenging.

We propose a novel AIT attack detection system that monitors such scattered attack orchestration from three different levels: machine learning features are extracted based on a single request information, multiple historical events associated with a user, phone number, or device, and country-wide suspicious traffic that has some ties to the request being inspected. A pivotal country-wide feature, for example, counts the number of distinct phone numbers associated with a given prefix value from the last 24 hour traffic. Based on this three-level feature engineering technique and a fixed threshold, we report 89.6% recall rate (false positive rate: 0.2%) on authentication requests initiated through the web client, and 91.1% recall rate (FPR: 0.1%) on the native application client traffic.

1 Introduction

In December 2022 Elon Musk tweeted *"Twitter was being scammed to the tune of 60M dollars a year for SMS texts..."* and explained the prevalence of artificially inflated traffic (AIT) attacks [5], which are coordinated to generate fake SMS authentication requests through bot accounts, and make huge profit by artificially "running up a tab" and billing authentication service providers. Highly likely suspects are rogue telecommunication service providers (telcos) or SMS aggregators who exploit their own customers. A recent whitepaper estimates the global damage of AIT attacks in 2023 to be about 1.2 Billion USD [1].

Rate limit policies are typically employed to slow down AIT attacks: authentication service providers allow just a several consecutive fail attempts before introducing a lengthy account lock out period to prevent heavy re-submission of two-factor SMS authentication requests. Our real-world SMS request log analysis, however, reveals that latest attacks are knowledgeable about the exact rate limit configurations, and perform widely distributed attacks to bypass them. Attackers use various methods to generate a large volume of fake accounts, phone numbers, and international mobile equipment identity (IMEI) numbers, and program bots to submit just a few requests before switching information – rendering rate limit techniques that monitor user, phone number, or device-specific events mostly ineffective. Numerous blogs published by SMS aggregators discuss general AIT attack trends, and recommend the adoption of rate limit policies without explaining the risks of encountering such distributed attack behaviors. Inspecting SMS conversion rates (successful validation rates) and incoming traffic rates are other typical suggestions – however, formal analysis of attack traffic and implementation guidelines are missing. When and how such information should be inspected and translated into specific rules or machine learning features remain unanswered. Further, we find no published literature attempting to formally characterize or investigate AIT attacks.

To bridge the gap, we analyze a large-scale SMS log dataset consisting of about 9.4 million logs collected from 19 different countries over three month in 2023: based on our own labeling method, we tag 4,947,099 samples as attacks and 4,496,447 samples as benign or genuine requests. Our attack set deep-dive reveals that the following three techniques are common: (1) use of short-lived email services (e.g., disposable email services) to rapidly generate bulk emails and register bot accounts using them, (2) bulk generation of unverified phone numbers and IMEI numbers based on common prefix values, (3) use of outdated devices to ease the process of rooting devices. Understanding the scattered nature of recent AIT attacks and the common attack techniques being used, we propose a novel feature engineering technique that inspects extensive traffic information from *three different levels*: (1) quick inspection of information available from a single SMS request event, (2) multi-event inspection of historical SMS events associated with a given user, phone number, or device, and (3) country-wide inspection of SMS events that have some ties to a request being evaluated. We train a decision tree classifier, and evaluate the model performance based on a test set representing the last two weeks data. Having observed slightly varying attack behaviors among the two client types – web browser and mobile native client – we train two separate classifiers based on different set of features, and evaluate their performance separately. We summarize key paper contributions as follows:

- This paper presents the first systematic analysis of AIT attacks based on real-world SMS authentication traffic logs.
- To timely detect extremely scattered attacks, our novel AIT attack detection technique inspects a *single* SMS event, *multiple* historical events associated with a user or device, and *country-wide* events that are tied to an attack campaign.
- We evaluate the effectiveness of this technique based on a large-scale dataset (9.4 million SMS logs). Based on 18 novel features, including a single-event feature that checks the time difference between "SMS request time" and "email domain first-appearance time," and a country-wide feature that monitors increase in the daily usage rate of a given email domain, we achieve 89.6% recall (0.2% FPR) for detecting attacks initiated through web clients. In the case native client traffic, country-wide features that identify suspiciously large recurrence of phone number or IMEI prefix patterns, together with single-event features that identify outdated system configuration exploits, play a pivotal role in detecting 91.1% of attacks (0.1% FPR). This native traffic tailored model consists of 13 features.

2 Objectives and Requirements

We explain the system design objectives and key deployment constraints identified through discussions with the engineers of the studied SMS authentication service.

2.1 Background: AIT Attacks and Scope

AIT attacks typically involve creation of a large number of bot accounts on a target authentication service, and making several SMS authentication requests through each account to artificially generate high-profile peak traffic. Attackers exploit both web and native clients, initiating SMS-authentication requests through web browsers or native apps installed on mobile devices. Likely suspects are malicious SMS aggregators or telcos that exploit their own customers (service providers) by billing them for such artificially inflated SMS traffic [1].

Authentication service providers try to mitigate AIT attacks by employing rate limit policies: an effective policy, for instance, would only allow several SMS authentication requests to be initiated by the same user account, device (IMEI), phone number, or IP address within a short time window. Any policy violation would trigger a lengthy lock-out period.

Our real-world data analysis, however, revealed that latest AIT attacks are effective in bypassing such rate limit policies: attackers (we presume recruited by SMS aggregators or telcos) seem to invest significant time, effort, and money into acquiring huge volumes of fake phone numbers, IMEIs, email addresses, and bot accounts, and performing distributed attacks using them. To maximize attack efficiency and integrity, short-lived disposable email services are often exploited to quickly register bulk email accounts, and common combination of *valid* prefix patterns are used upon generating fake phone numbers and IMEIs. A large peak traffic identified in Indonesia, for example, comprises 283,157 SMS requests with 168,346 fake phone numbers and 231,796 fake IMEIs exploited during a cycle lasting just 3 days. In an extreme case, we identified attack campaigns designed to submit just a single SMS request per each trackable information. Such sparsely distributed nature of AIT attacks make it almost infeasible to apply traditional multi-event inspection based detection methods. Despite those challenges, common attack patterns are prevalent across multiple countries, indicating that model generalization may be achievable. We further elaborate on the attack characteristics in Section 5.1.

Our primary goal is to optimize the detection rates for highly visible peak traffic cycles that are at least a few times larger in volume than normally observed daily volume in a given country, such that are also associated with suspiciously low daily conversion rates. This is because such high-profile peak traffics account for most of the observed AIT attacks, and the presence of distinct short-lived peak patterns provide strong ground truth evidence about AIT attacks.

2.2 Deployment Considerations

We identified three key business expectations that need to be considered upon building a practical and deployable system.

1. **Explainable features.** To facilitate efficient means to analyze and mitigate real-world occurrences of false positives and false negatives – identifying their root causes – the intuition behind the proposed features and their computational logic need to be clear.
2. **False positive rates (FPRs).** In the context of the studied

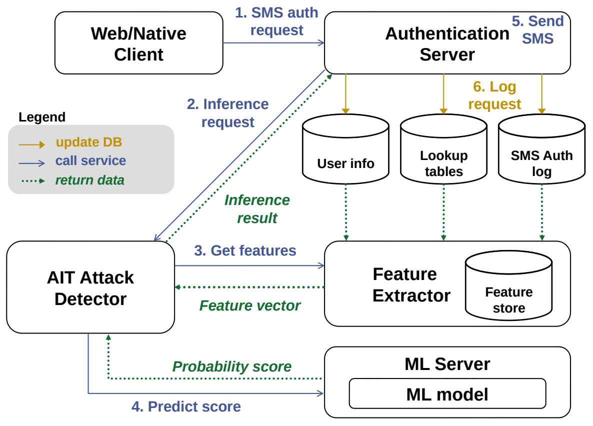


Figure 1: AIT attack detection system architecture overview.

authentication service, about 0.2% of SMS authentication false positives result in “Voice of the Customer” (VoC). If we set our usability objective to “add no more than 1 VoC per week,” two weeks of test data imply our false positive constraint should be about 1,000 false positives during that period. This requirement warrants for a more narrow investigation of FPR-favoring thresholds in the 0.8–0.9 range.

3. **Latency.** Since SMS traffic inspection would occur as part of the overall user login process, attack detection latency should be kept between half a second to a second to ensure unhindered user experience.

3 System Overview

We provide an overview of the proposed AIT attack detection system architecture, and key components used to compute the three-level features and make run-time inference decisions.

3.1 Architecture Overview

Figure 1 depicts an overview of the AIT attack detection system architecture. At the heart of the architecture is the “AIT Attack Detector,” which is responsible for computing an inference score for a given SMS request, making an authentication decision by checking whether the score is lower than a pre-defined threshold.

Authentication phase. A web or native client application is used to make an SMS authentication request. Upon receiving this request, the two-factor “Authentication Server” queries the Detector for an attack inference decision.

Feature computation phase. Once the Detector receives an SMS request, it initiates the “Feature Extractor” to compute the three-level feature sets. User account and device specific information are gathered from the “User Info” database managed by the Authentication Server, and used to compute single-event features such as the time difference between “SMS request timestamp” and “email domain first-appearance timestamp.” Historical SMS log data that are needed to compute multi-event and country-wide features are retrieved through the “SMS Log” database. For instance, a country-wide feature that counts the recurrence of a phone number prefix pattern in last 24 hours is computed based on retrieval of historical log data. We facilitate live inspection by using the current timestamp, and selecting the preceding 24 hour traffic. To improve computation efficiency, reusable static information such as native device/client released-dates are stored as “Lookup Tables” and facilitate fast access.

Attack inference phase. After the run-time feature computation process, the Feature Extractor creates an one-dimensional feature vector consisting of 18 values (web model) or 13 features (native model), and returns the feature vector back to the Detector. This feature vector is also stored in the “Feature store” for subsequent model training purposes. The Detector then forwards the feature vector to the “ML Server,” which in turn, uses the “ML model” to compute a binary classification probability score, and returns the score back to the Detector. Inference scores close to value “1” indicate an AIT attack whereas scores close to value “0” indicate genuine or benign requests. The Detector then checks the returned score against a threshold, and informs the Authentication Server to proceed with sending an SMS OTP if the score is less than the threshold. Finally, information about inspected SMS requests are logged to the SMS Log database.

Model training phase. We apply a labeling method (explained in Section 4.3) to create attack and genuine labels. Our model training process involves two separate steps. First, pre-computed feature vectors are retrieved from the “Feature Store” to construct a train set. In our evaluation, we use the first two-month data as the “train set,” the first 15 days of the third month as the “validation set,” and the remaining data as the “test set.” The train set is used to train a binary classifier that predicts the probability of a given request being an AIT attack. In our pilot experiments, the Gradient-Boosted Trees (GBTs) algorithm generally reported best detection performance (SVM, random forest, and logistic regression algorithms were also tested) – hence, all results are reported based on GBT classifiers. Tree hyperparameters were optimized based on grid search: 10 “max-depth” 200 “num-trees” showed optimal performance on the web traffic; 5 max-depth and 200 num-trees were optimal settings for native models.

4 Dataset Description

We provide details of SMS authentication logs and the labeling method applied to tag positive/negative samples.

4.1 Dataset Overview

We privately obtained the dataset through an SMS authentication service provider that serves two-factor SMS authentication service to tens of millions of end users residing in 195 countries – end users log in to their accounts to use online stores, manage their devices, access personal data, etc. Top 10 target countries with the most AIT attack traffic between August and October in 2023 from each of the two client channels (web and native) were pre-selected and shared – attack volumes were roughly estimated based on highly-visible peak cycles. Due to one overlapping case, there were 19 countries from both channels – the full list and their attack sizes can be found in Table 3 and 4. The majority of those countries were located on Africa and South Asia continents where the SMS pricing rates are typically very expensive. All 19 countries were also flagged as high risk countries in a recent report [1]. This dataset, in total, consists of 2,915,990 and 6,527,556 samples from the web and native traffic, respectively.

4.2 Log Attributes

The SMS log data contains basic SMS authentication request information including, anonymized user identifier, email domain name, target phone number, request time, IP address, device IMEI number, device model, client and OS version, authentication service type (e.g., account sign in or second-factor set up), application in use, SMS target country, action type (e.g., request sent or successful validation), and SMS pricing information. Information about the client type (web or native channel) used to first register an account, whether a given user has previously registered a "trusted device" (two-factor authentication is waived on a trusted device), and whether the ownership of a given phone number has previously been verified are also available. We note there were some information missing depending on the client type used. On the web side (requests initiated through web browsers), device, client application, and OS related information were missing. On the native side, user identifier and email domain information were missing – features that rely on such information were not considered upon building the native model. We extracted phone number prefix (removing last 4 digits) and IMEI prefix (removing last 7 digits) values, and stored them as separate fields to facilitate more efficient and privacy-preserving feature exploration process. We also obtained additional lookup tables summarizing the first released dates for different OS, native client, and device model versions.

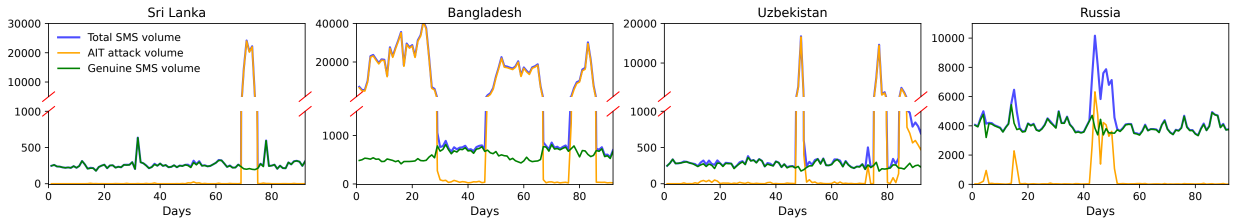
4.3 Labeling Method

Since we were provided with unlabeled data, and our objective was to train a binary classifier through supervised learning, we developed an AIT attack set labeling method tailored to the given SMS traffic dataset. Based on the intuition that highly visible *peak traffic* – daily SMS traffic volume that is multiple folds larger than the normally observed traffic for a given country – represents an AIT attack cycle with high precision, we first designed algorithms to accurately detect the presence of peak traffic: we defined "normal" daily SMS traffic volume and variability for each country by computing 25% trimmed mean and standard deviations across the three month data, and searched for daily traffic volume that is several standard deviations larger than this mean value; we also checked whether the daily conversion rate falls by a suspiciously large value (e.g., 20%) compared to the median computed based on the normal daily traffic. After identifying all peak cycles, we applied the following two labeling methods:

- In the case of web traffic, we searched for attack-triggering users and phone numbers that submitted at least four SMS requests without any successful validation during the same (within-peak) day; we chose "four" as the threshold based on web peak traffic distribution analysis, which revealed that web channel attacks typically submit four requests and switch information to bypass rate limit policies. Surprisingly, in three countries, we observed continuous peak traffic associated with high conversion rates (malicious peak traffics are typically associated with extremely low rates). Here, due to bulk of attacks validating requests, the first labeling rule was inapplicable. To improve labeling quality in such unusual context, we looked for signs of significant elevation in the proportion of a given application being used on a given day and country compared to a pre-defined daily usage proportion. If this difference is larger than a threshold (e.g., 0.3), indicating that a specific application is being heavily exploited by attackers, and the request came as part of a peak volume, we selected associated user and phone number.
- A large bulk of native channel attacks submitted just one or two requests before switching phone or IMEI numbers, and did not validate requests. To accurately identify such scattered behaviors, we simply selected all "phone number and IMEI" information pairs that are found in a given peak traffic, but have never been validated during the three month period.

After identifying a set of malicious users, phone numbers, and IMEI numbers, we labeled all samples associated with that set as positive (attack) samples. Note, although the primary objective of the proposed labeling method was to tag attack samples from peak cycles, this last step may also find a small number of samples that are part of non-peak traffic. All remaining samples are labeled as negative (genuine) samples. To check the validity of our labeling algorithms, we performed eye-tests on all 19 countries, and ensured that the normal daily traffic volume (representing the genuine volume) is consistently present throughout both peak and non-peak cycles. Figure 2 illustrate this technique: the green lines represent the normal traffic volume. Our labeling method was reviewed several times, and eventually endorsed by data experts and security engineers from the service provider. Upon

(a) Web client attacks



(b) Native client attacks

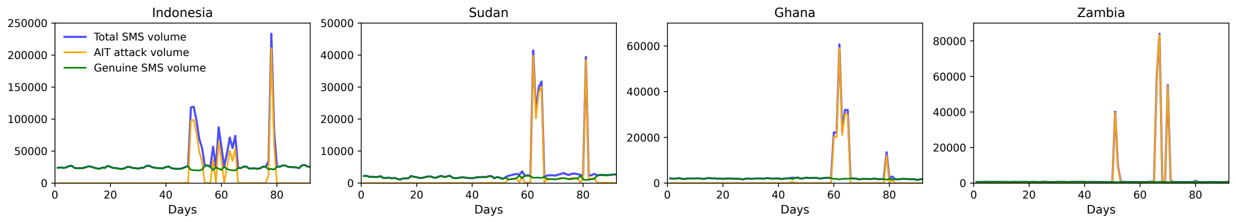


Figure 2: Web client attacks (top) and native client attacks (bottom) in the selected four countries during 3 months, respectively. Blue line, green line, and orange line represent daily total SMS volume, genuine volume, and attack volume, respectively.

completion of the full labeling process, we identified a total of 2,182,994 attack (732,996 genuine) samples from the web traffic, and 2,764,105 attack (3,763,451 genuine) samples from the native traffic. Bangladesh (36%), Ukraine (14%), and Pakistan (10%) comprise the largest attack sets in the web traffic. Indonesia (35%), Niger (13%), and Zimbabwe (12%) are three countries associated with the largest native attack traffic (see Table 8 in Appendix A).

A large peak attack cycle found in Sri Lanka (see Figure 2 (a)), for example, lasted 5 days and submitted 85,002 requests, while exploiting 2,611 fake phone numbers and 3,057 bot accounts during the process. The last peak cycle in Indonesia (see Figure 2 (b)) lasted for 3 days and submitted 283,157 SMS requests through the native client – 168,346 fake phone numbers and 231,796 fake IMEIs were exploited during the attack. Shorter cycles were found too: for example, a peak cycle in Sudan summitted 38,459 requests in a single day.

5 Feature Exploration

We identify representative AIT attack techniques based on the labeled data, and explain how this information can be translated into effective ML features. We distinguish features included in *preliminary* models and those exclusively available to *optimized* (final) models: preliminary models serve as initial baseline systems, which comprise features carefully constructed based on publicly accessible knowledge.

5.1 Known Attack Characteristics

SMS aggregators and security companies have posted blogs describing common AIT attack characteristics and simple detection rules that could be applied [21, 22]. Although implementation details are missing, commonly highlighted recommendations are clear, and involve the adoption of authentication rate limit policies, identifying heavily used accounts, devices, or IPs, and monitoring short-term traffic bursts and extreme drops in conversion rates. Several blogs [10, 14] also identify sequential numbering patterns that are commonly used to generate fake phone numbers, and suggest checking for similarities between phone numbers.

Implementing some of those suggested techniques are straightforward: to monitor heavy reuse, we explored `user-sms-count` and `ph-sms-count` features which simply count the number of requests submitted for a given user account or phone number within the last 24 hours (a time window fixed based on numerous optimization experiments), and included those features in the initial preliminary model. To characterize short-term traffic outbursts and programmatic re-submissions, we experimented with several statistical features: `user-diff-avg` selects all SMS authentication requests submitted by a given user within the last 24 hours, computes the time difference between all two consecutive request pairs, and records the average of those time difference values; `user-time-diff-std` records standard deviation about the average. Similarly, `ph-diff-avg`, `ph-diff-std`, `imei-diff-avg`, `imei-diff-std` features record the time difference average and standard deviation computed based on phone number and IMEI as the grouping information. Note, the user-specific features are only applicable to the web channel model due to missing information in the logs (see Section 4.2); likewise, the IMEI-specific features are solely applicable to the native channel model. These time-difference based statistical features are also included in the preliminary model. To characterize significant reductions in conversion rates we also explored `user-conv-rate`, `ph-conv-rate`, and `imei-conv-rate` features as part of the preliminary model, which measure the proportion of successfully validated requests in the last 24 hours for a given user account, phone number, and IMEI number, respectively. We refer to above features as "multi-event" features.

Identification of sequential numbering patterns in phone numbers, however, is less trivial. A significant portion of observed attacks submit just one request before switching trackable information – it would be infeasible to measure similarity between numbers in this scenario. One might consider measuring similarity against an entire SMS traffic of a given country based on a lengthy time window but such techniques would require a few hundreds if not thousands of similarity scores to be computed per request – this level of computational complexity sits uneasily with the latency expectations (see Section 2.2).

Counting the recurrence of "phone number prefix" is another viable solution. But exactly how and when prefix recurrences should be counted, and how those prefix count values should be used during run-time inspection are non-trivial questions that remain unanswered. We propose a novel feature engineering technique that first extracts the prefix value from a given phone number (request being inspected), counts the recurrence of that prefix from an entire 24 hour traffic of a target country, and appends this count value, `ph-prefix-count`, as one of many ML features while constructing the feature vector. The key advantages are (a) only the prefix associated with the inspected phone number is counted, and (b) multiple features are used collectively to reduce false positives.

Due to the AIT attack behaviors diverging slightly, and the disparities in the type of information being collected (or missing in the logs) between the two channels, we analyze the attack characteristics associated with each channel separately.

5.2 Web Channel Attack Characteristics

About 73% of the web-channel attacks submitted four or less requests in the 24 hour time window. The multi-event features, designed to detect frequent re-submissions, will not be effective in detecting the first or second attempts (due to lack of historical information). Understanding this limitation, we shift our focus to inspecting country-wide suspicious traffic or additional side-channel information related to single events.

Short-Lived Email Services (Web-Short-Email). We identified a common technique from the web traffic that exploits short-lived temporary email services (mostly disposable email services) to facilitate rapid and bulk generation of functioning emails, and registration of bot accounts. About 31% of web-channel attacks use bot accounts registered using emails created through such temporary services, which are relatively new, and thus their first appearance date (in the SMS logs) should be recent. Our `em-domain-sms-diff` single-event feature is designed to flag the use of such temporary email services: we compute the difference between the SMS request timestamp and the email domain first-appearance timestamp. Benign accounts are typically associated with well established email services like "gmail" or "hotmail," and the time difference value will be in the several years range. Conversely, bot accounts will report a much shorter value – often in the range of several days or weeks.

Dominant Email Services (Web-Dominant-Email). We also observed a widely contrasting attack trend: about 47% of attacks associated with bot accounts were registered using well-established email services such as "gmail.com," "outlook.com," or "yahoo.com." Hence, such attempts cannot be detected using the `em-domain-sms-diff` feature. To timely detect this campaign, we propose a novel country-wide feature that monitors the daily usage rate of well-known email domains, and flags any suspicious traffic elevations. Our `em-domain-prop-change` feature computes the proportion of SMS traffic initiated by a given email domain in the last 24-hour traffic of a target country, and subtracts the baseline value (pre-computed based on normal daily traffic) from this computed value to measure the change.

Phone Number Prefix Reuse (Web-Phone-Prefix). Another attack technique exploits a large list of unverified phone numbers, and accounts for about 42% of the web attacks. To detect the root attack behavior, we employed the `ph-prefix-count` feature described in Section 5.1. Our country-wide fake phone number pattern investigation revealed that attackers typically keep the first 4–6 digits (which often represent state or telco) the same [4], and change just the last 4–5 digits to generate bulk numbers. Taking advantage of such constraints impeding attackers, we removed the last four digits from a given number, and used the remaining digits as the prefix. Same (reused) phone numbers contributed just once upon counting the recurrence of a prefix value.

Successful Validation (Web-Validation). Surprisingly, in three countries, Azerbaijan, Ukraine, and Uzbekistan, we observed a significant portion of attacks (about 82%) that successfully validated authentication requests – we surmise this attack was designed to bypass rate limit policies and conversion-rate monitoring rules. Similar to the Web-Phone-Prefix campaign, however, the majority of exploited bot accounts were registered with small number of well-known email domains (99%), and programmed to switch between fake phone numbers (estimated to be about 43%). To that end, we expect the two country-wide features, `em-domain-prop-change` and `ph-prefix-count`, to be effective in detecting this attack campaign.

Feature distribution differences between attack samples and genuine samples are depicted in Figure 3 in Appendix B.

5.3 Native Channel Attack Characteristics

Native channel attacks were further scattered: in most cases (96%) bots submitted just one or two requests before switching IMEI numbers or phone numbers.

Phone Number Prefix Reuse (Native-Phone-Prefix). Phone number prefix reuse behaviors were more prevalent in native channel attacks. About 91% of the native attack samples recorded `ph-prefix-count` values greater than 30, while 99.9% of the genuine samples recorded values less than 10 (see Figure 3 in Appendix B). This warrants for an extensive use of the same `ph-prefix-count` feature. In the optimized native model, we also added `ph-prefix-conv-rate`, which measures the conversion rate for a given phone prefix value within the last 24 hours – country-wide conversion rate features were designed to exploit the fact that native attacks typically do not complete validation.

IMEI Prefix Reuse (Native-IMEI-Prefix). We observed a similar technique in the context of IMEI numbers, in which attackers generated bulk unverified IMEI numbers associated with a single mobile device model (see Figure 3 in Appendix B), and programmed bots to use a different IMEI after submitting just one or two requests. Impressively, those fake IMEI numbers typically conformed to the valid IMEI format, allowing them to bypass various IMEI validity checkers. In-depth analysis of the IMEI patterns revealed that attackers often change just the last 7 digits – this is because the first 8 digits of an IMEI number represent the "type allocation code", which are preset values and cannot be changed, and the next 7 digits consist of 6-digit serial numbers and one check digit [3], leaving just the last 7 digits for possible manipulation. Hence, we simply excluded the last 7 digits to identify an IMEI prefix value. An IMEI prefix count feature, however, showed marginal impact in improving performance due to being strongly correlated to `ph-prefix-count` values. In a continued attempt to take advantage of this prefix reuse behavior, we explored `imei-prefix-sms-prop`, which measures the proportion of requests submitted by a given IMEI prefix in the 24-hour traffic volume recorded from a given target country. Having noticed that a large chunk of IMEI numbers were generated through a fixed device configuration (model, OS, and client version), we also examined the effectiveness of adding a `device-sms-prop` feature, which measures the proportion of traffic initiated by a given device configuration in the 24-hour traffic of a target country. This feature was particularly useful in lowering FPRs. We also added `device-conv-rate` to the final model to measure the country-wide conversion rate for a given device model.

Old Native Clients (Native-Old-Client). About 65% of native channel attacks were performed on earlier device models that are at least 6 years old, or using outdated OS or client versions that are at least 4 years old. Conversely, most of the genuine traffic were initiated through latest configurations: only 18% were using earlier models or outdated software versions. To identify those behaviors, we explored three novel single-event features, `device-sms-diff`, `os-sms-diff`, and `client-sms-diff`, which measure the time difference between the current request timestamp and released date of the given device model, OS version, and client version, respectively. We had access to released date lookup tables. Considering deployment, however, those dates can be roughly estimated by using their first log appearance dates.

5.4 Web Model Features

The preliminary model, which has been designed based on the well-known attack trends (see Section 5.1), includes all multi-event features as well as the novel country-wide `ph-prefix-count` feature – consisting of 10 features in total. The final optimized model uses 8 additional features, including all new single-event and country-wide features described in Section 5.2. We explain all 18 features in detail through Table 1. The optimized model also uses five new single-event features – `service-id`, `sms-cost`, `join-channel`, `is-same-country`, and `have-trusted-device` – to capture information about certain authentication services being exploited more during attacks, country-specific SMS pricing rates, whether users registered their accounts through a web or native client (bot accounts are typically registered through web), whether the account registration country matches the SMS target country, and whether a user has previously added a trusted device to skip two-factor authentication. Marking a device trusted likely resembles a genuine user behavior. We measured the feature importance scores using the "mean decrease in PRAUC accuracy" method while training the final optimized model, and sorted the list in Table 1 based on on their importance ranks.

5.5 Native Model Features

The native preliminary model comprises the same set of features as the web version except the IMEI-specific multi-event features replace user-specific features. Along with the new features described in Section 5.3, we added `is-ph-verified` to the optimized native model, which is a single-event feature that checks whether the ownership of a given phone number has been verified before. We presume legitimate users would use phone numbers that have been verified at least once in the past. The final model comprises 13 features, which are explained in Table 2.

6 Evaluation

In this section we report the attack detection (recall) rates for the two channels (web and native) measured based on the validation set and test set, and compare the performance between the preliminary and optimized model versions.

6.1 Software and Hardware Used

We evaluated the system detection accuracy and latency using a server machine equipped with Intel Xeon Platinum 8173M CPU 2.00 GHz (4 cores), 24GB of RAM, and an NVIDIA Tesla T4 GPU card. We trained GBT classifiers using "Tensorflow 2.13.1" and "TensorFlow Decision Forests

Table 1: A summary of the optimized Web model ML features – sorted based on the feature importance score ("Imp. Score"). All multi-event and country-wide features are computed using a time window of 24 hours unless otherwise specified. The feature importance scores are measured using the "mean decrease in PRAUC (area under precision-recall curve)" method.

Imp. Score	Feature Type	Feature	Computation Logic
0.0141	Single-event	`em-domain-sms-diff`	Time difference between SMS request date and email domain first appearance date (converted to days)
0.0036	Country-wide	`ph-prefix-count`	# distinct phone numbers with the same prefix value extracted from a given phone number
0.0031	Country-wide	`em-domain-prop-change`	SMS volume proportion from a given domain − SMS volume proportion from a given domain normal traffic
0.0024	Single-event	`service-id`	Source of SMS request
0.0009	Single-event	`sms-cost`	SMS billing rate for a given country
0.0001	Single-event	`join-channel`	Channel used during initial account registration (mobile or web)
0.0001	Multi-event	`user-sms-count`	# SMS from a given user
0.0001	Single-event	`is-same-country`	SMS target country and GeoIP country are the same (0 or 1)
0.0000	Single-event	`have-trusted-device`	User has marked at least one device as trusted (0 or 1)
0.0000	Multi-event	`user-diff-std`	Standard deviation of time difference values computed between all consecutive pair of events selected for a given user
0.0000	Multi-event	`user-conv-rate`	Successful authentication validation rate for a given user
0.0000	Multi-event	`ph-user-count`	# users associated with a given phone number
0.0000	Multi-event	`user-ph-count`	# phone numbers used by a given user
0.0000	Multi-event	`ph-conv-rate`	Successful authentication validation rate for a given phone number
0.0000	Multi-event	`ph-diff-avg`	Average of time difference values computed between all consecutive pair of events selected for a given phone number
0.0000	Multi-event	`user-diff-avg`	Average of time difference values computed between all consecutive pair of events selected for a given user
0.0000	Multi-event	`ph-diff-std`	Standard deviation of time difference values computed between all consecutive pair of events selected for a given phone number
0.0000	Multi-event	`ph-sms-count`	# SMS from given phone number

Table 2: A summary of the optimized native model ML features – sorted based on the feature importance score ("Imp. Score"). All multi-event and country-wide features are computed using a time window of 24 hours unless otherwise specified.

Imp. Score	Feature Type	Feature	Computation Logic
0.0377	Country-wide	`ph-prefix-count`	# distinct phone numbers with same prefix value extracted from a given phone number
0.0290	Single-event	`is-ph-verified`	Given phone number was previously verified (0 or 1)
0.0223	Single-event	`sms-cost`	SMS billing rate for a given country
0.0162	Single-event	`os-sms-diff`	Time difference between SMS request date and OS version released date
0.0052	Single-event	`client-sms-diff`	Time difference between SMS request date and native client application version released date
0.0043	Multi-event	`ph-conv-rate`	Successful authentication validation rate for a given phone number
0.0033	Country-wide	`imei-prefix-conv-rate`	Successful authentication validation rate for a given IMEI prefix
0.0027	Country-wide	`device-sms-prop`	SMS volume proportion from a given device model in a given country
0.0019	Country-wide	`device-conv-rate`	Successful authentication validation rate for a given device model
0.0016	Country-wide	`imei-prefix-sms-prop`	SMS volume proportion from a given IMEI prefix in a given country
0.0011	Country-wide	`ph-prefix-conv-rate`	Successful authentication validation rate for a given phone number prefix
0.0008	Single-event	`device-sms-diff`	Time difference between SMS request date and device model released date
0.0004	Multi-event	`imei-conv-rate`	Successful authentication validation rate for a given IMEI

1.5.0" libraries in Python, and used relational database and SQL queries to access raw data.

6.2 Dataset and Evaluation Methodology

We divided the three-month dataset into three separate sets: a train set comprising the first two month logs, a validation set comprising the first 15 days of logs from the third month, and a test set comprising the remaining samples. We selected a total of 1,690,953 attack samples and 497,917 genuine samples to train the web model. The native model was trained using 750,420 attack samples and 2,509,922 genuine samples. The details on train set by country are found in Table 9 in Appendix A. We explored several sampling methods to construct a more attack-to-genuine balanced train set (e.g., about 50% from each class) but our validation experiments revealed that using all available samples tend to produce superior results – this is probably because the degree of class imbalance is not substantially high [2]. The validation set was used extensively to optimize individual feature performance, and finalize the feature vectors presented in Sections 5.4 and 5.5. We report the validation set performance in the next section. The two optimized versions were used to measure the final false positive rates (FPRs) and true positive rates (TPRs) on the test set. This final test set consists of 250,240 attack samples and 116,392 genuine samples from the web traffic, and 840,006 attack samples and 664,853 genuine samples associated with the native traffic.

6.3 Validation Set Performance

We measured validation set performance for the web and native models based on two different thresholds (0.8 vs. 0.9) and two model versions (preliminary vs. optimized). We chose two FPR-favoring thresholds based on the guidelines stated in Section 2.2. Validation set TPRs and FPRs are summarized in Tables 3 (web) and 4 (native).

Preliminary vs. optimized. The optimized web model demonstrated superiority in both metrics: overall TPR increased by 3.2 percentage points while FPR recorded 1.0 point reduction under 0.9 threshold. Attacks in Bangladesh heavily exploited Web-Short-Email and Web-Phone-Prefix techniques – since `ph-prefix-count` is available in both models, we surmise `em-domain-sms-diff` was primarily responsible for detecting additional attacks in the optimized model. Regarding Azerbaijan, `em-domain-prop-change` was the main reason for 12.4 percentage point elevation in TPR. Several individual FPRs showed immense improvements: Bangladesh and Azerbaijan, in particular, recorded 11.80 and 2.43 percentage point reductions in FPR. We observed similar trends with the optimized native model and 0.9 threshold. FPR benefits were clear, albeit minor TPR losses: Kuwait, Lesotho, and Niger reported 8.31, 8.46, and 5.35 reductions in FPRs, respectively, based on the optimized model.

Threshold effects. The adoption of 0.9 threshold on the optimized web model led to about 1.1 percentage point reduction in TPR (compared to 0.8 threshold), yet individual FPR enhancements were significant: 0.47, 0.24, 0.53 percentage point reductions in FPRs were reported in Ukraine, Bangladesh, and Azerbaijan, respectively. Similar effects were observed from the optimized native model: 4.50, 1.58, and 5.60 FPR reductions were reported in Niger, Sudan, and Kuwait, respectively.

Web vs. native models. With respect to the optimized models and 0.9 threshold adoption, FPRs were less than 0.2% in both models. Native model reported higher TPR (96.1% vs. 91.2%) – we attribute this to more consistent combination of attack techniques being employed in the native attack set ("N1, N2, N3") across countries.

Low TPR cases. The optimized web model still reported noticeably low TPR for Ukraine (65.9%): the TPR measured based on the first 7 days was 82.3% but fell sharply to 50.7% when the last 8 days of attack traffic was used. We attribute this change to divergence in the two key feature distributions: the proportion of first 7-day traffic with `em-domain-prop-change` greater than 0.2 was about 81% but fell immensely to 51% under the last 8-day traffic; likewise, the proportion of the first 7-day traffic with `ph-prefix-count` greater than 30 was 65%, which fell to just 32% among the last 8-day traffic. In Section 6.7, we investigate the effects of retraining models (with latest data) on stabilizing performance.

High FPR cases. High FPR cases were found in Uzbekistan (2.1%) and Kuwait (2.3%) from the web and native traffic, respectively. Both FPRs are significantly higher than the average. However, all false positives recorded in those two countries incurred during short peak cycles – hence, we expect most user experience issues (genuine users being denied access) to be temporary, and disappear shortly after a day or two, e.g., when an associated phone prefix or email domain is no longer being heavily exploited by an attack. In the case of Kuwait (native), all 260 false positives were associated with `ph-prefix-count` values greater than 90 (suspicious) and only 5 of them recorded conversion rates greater than zero (genuine-like behavior) – implying that a significant portion may actually represent true positives (mislabeled samples), and a very small number of actual false positives may occur when benign users coincidentally use attackers' prefix values or device model during peak attack cycles. Similarly, in the case of web traffic, all false positives in Uzbekistan were found as part of peak cycles, heavily reusing phone prefix values. Among them, 21.6% of false positives simultaneously exploited short-lived email services. Such adversarial characteristics indicate that those samples may also represent true positives.

6.4 Test Set Performance

To emphasize the performance superiority of the optimized model, we compare the test set probability score distributions between the two models (preliminary vs. optimized) in Appendix C. In this section, we report the final test set performance based on the two optimized models.

Threshold effects. The optimized web model under 0.9 threshold reported about 3.8 times improvement in FPR (compared to 0.8) while compromising about 2.5 percentage points in TPR. With respect to the optimized native model, two thresholds led to insignificant differences in both TPRs and FPRs. Considering the tight FPR constraints (see Section 2.2), our recommendation is to use 0.9 threshold for both models.

Web vs. native models. The two optimized models reported similar TPRs and FPRs under the 0.9 threshold, demonstrating their robustness to the test set settings. F1-scores for the web and native models were 95.27 and 95.06, respectively, and precision were 99.92 and 99.95. The web model achieved competitive TPRs in all countries with at least one highly visible peak cycle (top five countries in the table) – all recall rates exceeding 80%. The bottom four countries were associated with minor non-peak traffic (few hundred samples), and we are not too concerned about their low TPRs. The native model reported above 90% TPRs in all countries except for Indonesia (86.5%) and Sudan (77.1%) – as we learned from the validation experiment, those two countries have tendency to perform slightly worse. Attack samples from Zambia, Peru, and Lesotho are associated with small off-peak traffic.

FPR requirements. 0.19% FPR associated with the web model translates to just 221 false positives and 0.11% FPR associated with the native model translates to 731 false positives – the total false positives added from both channels (952) would satisfy the "about 1,000 false positives" requirement.

Low TPR cases. In the case of the native traffic, we recognized one country with low TPR: Sudan (77.1%). In-depth analysis of attack samples with scores less than 0.9 (threshold) revealed that the majority (86%) is associated with much smaller yet stealthy daily traffic volumes. About 21% of the IMEIs and phone numbers used during this suspicious time span were also found active in previous peak cycles, adding confidence that this stealthy volume may indeed represent true positives in real-world settings. These attacks seem extremely scattered though – submitting just one request per IMEI or phone number. Heavy prefix reuse patterns were missing, rendering all country-wide features ineffective against them. Overall traffic volumes initiated by such attack campaigns, however, were small: 9,390 samples in Sudan in the course of two weeks.

High FPR cases. We found a slightly elevated FPR regarding the native traffic in Libya (1.3%). In contrary to what we

Table 3: Web-model TPRs and FPRs measured using the validation set and two threshold values (0.9 and 0.8), and reported separately for the preliminary and optimized model (presented as column headers). W1, W2, W3, and W4 represent “Web-Short-Email”, “Web-Dominant-Email”, “Web-Phone-Prefix”, and “Web-Validation” attack techniques, respectively. “Not-peak” indicates small number of attack samples belonging to off-peak traffic. Countries are sorted based on attack set size. Δ represents changes in TPR/FPR compared to corresponding (same threshold) preliminary model performance.

Country	Attack Techniques	# SMS requests		Preliminary				Optimized			
				thr=0.9		thr=0.8		thr=0.9		thr=0.8	
		Attack	Genuine	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Sri Lanka	W2,W3	85,018	3,525	99.76	0.00	99.77	0.00	99.77	0.00	99.78	0.00
Ukraine	W2,W3,W4	48,127	6,629	66.59	0.09	70.85	0.36	65.89	0.47	68.75	0.94
Bangladesh	W1,W3	41,528	9,493	83.50	11.85	83.66	12.87	96.25	0.05	96.45	0.29
Libya	W2,W3	38,347	870	99.62	0.00	99.62	0.00	99.75	0.11	99.76	0.23
Azerbaijan	W2,W3,W4	21,929	1,521	76.76	2.56	76.95	2.96	89.19	0.13	93.81	0.66
Uzbekistan	W2,W3,W4	6,024	3,513	99.10	2.08	99.30	2.13	99.54	2.13	99.65	2.13
Russia	Not-peak	351	57,274	13.96	0.09	20.23	0.23	28.49	0.04	41.60	0.05
Pakistan	Not-peak	237	17,474	19.41	0.19	22.36	0.33	14.77	0.02	16.46	0.02
Nigeria	Not-peak	126	11,171	20.63	0.00	42.06	0.03	17.46	0.00	21.43	0.01
Morocco	Not-peak	114	7,217	0.88	0.24	1.75	0.86	0.88	0.00	0.88	0.01
Total		241,801	118,687	87.95	1.13	88.87	1.36	91.18 (Δ +3.23)	0.12 (Δ -1.01)	92.23 (Δ +3.36)	0.18 (Δ -1.18)

Table 4: Native-model TPRs and FPRs measured using the validation set and two threshold values (0.9 and 0.8), and reported separately for the preliminary and optimized model (presented as column headers). N1, N2, and N3 represent “Native-Phone-Prefix”, “Native-IMEI-Prefix”, and “Native-Old-Client” attack techniques, respectively. Sorted based on attack set size.

Country	Attack Techniques	# SMS requests		Preliminary				Optimized			
				thr=0.9		thr=0.8		thr=0.9		thr=0.8	
		Attack	Genuine	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Zambia	N1,N2,N3	197,759	9,259	99.66	0.52	99.68	0.54	99.73	0.08	99.76	0.27
Zimbabwe	N1,N2,N3	188,380	12,067	99.45	0.47	99.75	0.47	99.74	0.00	99.77	0.20
Indonesia	N1,N3	166,112	346,263	87.99	0.02	88.00	0.02	87.89	0.01	88.01	0.02
Niger	N1,N2,N3	158,644	4,668	99.86	5.91	99.87	5.93	99.49	0.56	99.84	5.06
Ghana	N1,N2,N3	140,455	27,537	98.89	0.58	98.91	0.58	98.70	0.06	98.76	0.17
Sudan	N1,N2,N3	131,970	21,602	87.67	1.83	87.70	1.84	86.88	0.25	87.80	1.83
Lesotho	N1,N2,N3	99,598	1,798	99.92	8.57	99.94	8.57	99.54	0.11	99.57	0.61
Kuwait	N1,N2,N3	78,967	11,470	99.50	10.58	99.54	10.60	96.05	2.27	98.23	7.87
Libya	N1,N2,N3	11,325	18,895	96.64	0.47	96.67	0.47	92.78	0.02	96.81	0.47
Peru	Not-peak	469	135,117	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total		1,173,679	588,676	96.50	0.42	96.57	0.42	96.08 (Δ -0.42)	0.07 (Δ -0.35)	96.46 (Δ -0.11)	0.31 (Δ -0.11)

Table 5: Optimized web model TPRs and FPRs measured using the test set and two threshold values. Countries are sorted based on attack set size available in the test set.

Country	Attack Techniques	# SMS requests		Optimized			
				thr=0.9		thr=0.8	
		Attack	Genuine	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Bangladesh	W1,W3	114,498	9,334	92.02	1.00	94.76	6.57
Ukraine	W2,W3,W4	67,443	6,710	85.68	0.30	88.13	0.80
Azerbaijan	W2,W3,W4	34,323	1,428	81.83	0.21	85.99	0.70
Uzbekistan	W2,W3,W4	23,455	3,211	98.71	1.10	98.92	1.25
Morocco	W2,W3,W4	9,869	6,720	96.99	0.16	97.19	0.52
Russia	Not-peak	363	59,317	20.11	0.06	27.27	0.11
Pakistan	Not-peak	235	15,685	20.85	0.11	22.98	0.17
Nigeria	Not-peak	50	9,936	14.00	0.00	16.00	0.01
Sri Lanka	Not-peak	4	3,318	0.00	0.00	0.00	0.00
Libya	Not-peak	0	733	0.00	0.00	0.00	0.00
Total		250,240	116,392	89.55	0.19	92.07	0.73

Table 6: Optimized native model TPRs and FPRs measured using the test set and two threshold values. Countries are sorted based on attack set size available in the test set.

Country	Attack Techniques	# SMS requests		Optimized			
				thr=0.9		thr=0.8	
		Attack	Genuine	TPR(%)	FPR(%)	TPR(%)	FPR(%)
Indonesia	N1,N2	284,708	402,072	86.52	0.10	86.61	0.13
Niger	N1,N2	200,412	4,653	98.30	0.52	98.31	0.52
Kuwait	N1,N2	159,558	10,540	94.99	0.01	95.05	0.02
Zimbabwe	N1,N2	132,128	14,096	90.76	0.00	90.89	0.00
Sudan	N1,N2	47,849	30,373	77.14	0.02	77.16	0.02
Ghana	N1,N2	13,602	26,957	94.82	0.01	95.24	0.01
Zambia	Not-peak	927	9,302	50.59	0.00	53.83	0.00
Peru	Not-peak	516	143,292	0.00	0.00	0.00	0.00
Lesotho	N1	298	1,814	81.54	0.00	81.54	0.00
Libya	Not-peak	8	21,754	0.00	1.32	0.00	1.42
Total		840,006	664,853	91.11	0.11	91.19	0.13

observed from the validation case studies, however, all false positives in Libya were found from non-peak traffic. Similarly, about 89% of false positives found in Indonesia – where the genuine set size is by far the largest among all countries – were also associated with non-peak traffic. Our deep-dive analysis of such samples revealed that they may represent mis-labeled cases: we identified numerous attack-like feature distributions (e.g., large `ph-prefix-count` and `imei-prefix-sms-prop` values), and no previous history of phone number verification, which strongly indicate that they may have been mis-labeled as negative samples simply because our labeling algorithm disregards non-peak traffic. Another piece of evidence that confirms this hypothesis is their specific device configuration, which was also identified as a heavily exploited configuration in the previous Niger attack. We argue that those mis-labeled samples would represent true positives in real-world settings, and would be correctly classified by our system as attacks. In the case of the web traffic, 86% of the false positives reported from Bangladesh and Uzbekistan were part of peak attack cycles. About 62% of those samples (in peaks) were associated with repeated use of “add a new number” or “password reset” services (`service-id`), which are both adversarial behaviors. Again, such bulk false positives may represent mis-labeled cases.

Window size evaluation. To validate the suitability of using 24-hour time windows, we experimented with 1, 12, 24, and 48 hour time windows for computing features, and compared performance on the test set. A 24-hour time window, for example, implies that we would use all traffic available during the last 24 hours, starting from the current timestamp. TPRs (and FPRs) reported based on those four window sizes were 58.9% (0.20%), 66.2% (0.21%), 91.2% (0.12%), and 86.1% (0.14%) for the web model, and 38.2% (0.05%), 27.6% (0.00%), 96.1% (0.07%), and 91.1% (0.09%) for the native model – showing peak performance with 24-hour windows.

Generalization. To demonstrate generalizability of the two models, we removed two countries with the largest number of test attack samples from the train set, trained new models, and measured performance on those two "unknown" countries. From the web train set, we removed Bangladesh and Ukraine, trained a model, and measured performance on those two countries: 98.0% TPR and 0.57% FPR. Likewise, we removed Indonesia and Niger from the native train set, and measured their performance: 91.0% TPR and 0.04% FPR. We surmise attacks in unknown countries are effectively mitigated because identified "attack techniques" (see Tables 5 and 6) are commonly employed across multiple countries. To measure train set size dependencies, we trained the two models using just the second month (September) data, and evaluated performance based on the test set: the web model reported 90.5% TPR (0.31% FPR), and the native model reported 91.2% TPR (0.11% FPR), demonstrating robustness to reduced train set sizes.

6.5 Robustness to Evasion Attacks

To measure the two models' robustness to evasive feature manipulations we first outline the following assumptions:

1. Attackers will try to manipulate as many features as possible (use values close to genuine distributions) yet still generate heavy peak volumes to maintain profit.
2. To maintain similar peak volumes and cycle duration, attackers will not be able to drastically downsize country-wide (country-level volume monitoring) features.
3. Almost all native attacks use inoperative phone numbers – implying native conversion rate and verification features cannot be manipulated without dramatically increasing costs (purchasing tens of thousands of phone numbers).

Based on those assumptions, we constructed cost-efficient yet comparably pervasive attack sets by first selecting all genuine samples from the test set (attackers replicating benign behaviors), and replacing all country-wide features with values that represent 50, 40, 30, 20, and 10th percentile in the test attack set. We then measured web and native model detection rates based on those sets. As summarized in Table 10 (see Appendix D.1), the web model showed robustness to evasive settings that use the 40th percentile country-wide values; weaknesses were spotted, however, when 30th percentile values were used, allowing about 51% of attacks to bypass the system. The web model broke when 20th percentile values were used (4.1% TPR). Such downsized attacks, however, appear infeasible: to generate similar level of short-lived peak volumes – we assume attackers' target would be around median values – attackers would have to acquire about 48 times more valid phone prefix values (336 vs. 7), and create bulk bot accounts on 6 more email domains (70.3% vs. 12.5%). On average, about 2,658 unique phone prefix values were exploited in each peak cycle. Considering that phone numbers (used in the web countries) are typically 9-10 digits long, and the first 2-3 digits are fixed for mobile carriers or area codes, attackers are left with just 2-4 digits to freely exploit phone prefix values. Hence, acquiring 48 times more prefix values seems impractical. Conversely, the native model maintained robustness across all five evasive settings. We attribute this to abnormally high phone prefix count values (260) and SMS proportions (30%) found even at the 10th percentile positions.

To construct more resilient web models, we applied the identical evasive set generation technique to the train set, and transformed genuine train samples into attack samples based on the country-wide feature values selected at five different percentile positions. Based on five different combination of train sets (evasive set plus original train set), we trained five separate models, and measured their performance on the original test set as well as the evasive test set. As shown in Table 11 (see Appendix D.1), a new model trained with an evasive set representing 30th percentile values reported 82.0% TPR (0.15% FPR) on the test set constructed using 30th percentile values, demonstrating the effectiveness of adding evasive samples in the train set. This same model reported 45.7% TPR (0.15% FPR) on the attack set constructed using 20th percentile values – low ROIs (54.3% success rate) would likely discourage attackers from initiating such expensive attacks. Importantly, all five models reported consistently high TPRs and low FPRs on the original test sets. We also experimented with an extremely expensive native setting that involves attackers purchasing tens of thousands of new phones and validating all SMS requests: even in this setting, the same training technique was effective in lowering attack success rates to just 25% (see Appendix D.2).

6.6 Model Size and Overheads

We measured the model complexity, model training time, and attack inference time for the optimized web and native models. The web model is 7.6 megabytes in size, and contains 200 trees and 121,628 nodes. The native model with less number of features is even lighter, consisting of 200 trees and 6,170 nodes; it is just half a megabyte in size. It only took 617 and 326 seconds to train the two models, respectively.

Table 7: TPRs and FPRs of original (“No retraining”) model, and the three different retraining strategies: models are trained every 3, 5, and 10 days.

Channel	**No retraining**		**10-day retraining**		**5-day retraining**		**3-day retraining**	
	TPR(%)	**FPR(%)**	**TPR(%)**	**FPR(%)**	**TPR(%)**	**FPR(%)**	**TPR(%)**	**FPR(%)**
Web	90.35	0.15	94.69	0.15	95.36	0.18	96.30	0.18
Native	94.01	0.09	94.05	0.07	94.10	0.09	94.09	0.10

In an effort to minimize run-time model inference time, we assumed historical SMS request logs representing the last 24 hours of web/native traffic are pre-loaded in memory through continuous batch execution for each of the studied 19 countries, and retrieving the last 24-hour traffic directly from memory would take negligible query time – in the worst cases (when the daily traffic volumes are at their global peaks), this continuous caching job would use up about 31 (web traffic) and 253 (native traffic) megabytes of memory. Our inference latency measurements therefore excludes this initial data retrieval time, and solely measures the time taken to (a) compute all 18 (web) or 13 (native) features on a given request log, and (b) execute the model predict function. For each of the two models, we randomly selected 10,000 attack samples and 10,000 genuine samples, and measured the mean time taken to complete those two steps. The mean time taken to predict attack samples and genuine samples using the web model were 31.2 ($\sigma = 11.4$) and 17.0 ($\sigma = 4.4$) milliseconds, respectively. We noticed a significant elevation in the mean time when the native model was tested, reporting 329.4 ($\sigma = 156.0$) and 63.4 ($\sigma = 11.7$) milliseconds. We attribute this difference to the increase in daily and peak SMS volumes. Benign users may experience some added delays when heavy-volume AIT attacks are being orchestrated but we argue that such hindered experiences will be temporary as peak attacks typically last only a few days.

6.7 Model Retraining Effects

To investigate the effects of periodically retraining models, we merged the validation set and test set as a single test set (comprising 30 days of test samples), and experimented with 3, 5, and 10-day retraining periods. For instance, to measure the 3-day retraining performance, we started with the original models and recorded inference scores using the first 3 days of the merged test set. We then trained a new model including the samples from those 3 days in the train set, while ensuring proportion of the original (thoroughly validated) train set is always 75% or more of the entire set – this proportion starts from 100% but gradually falls to 75% as more SMS logs become available over time. We always selected most recent samples (latest attack trends) to make up the remaining 25%. We then computed inference scores again using the samples from the next 3 days (4th–6th day). Consequently, we would retrain a new model every 3 days, measure performance based on the following 3 days of test samples, and repeat the process until we cover the entire test set. We then report a single TPR/FPR by combining all results. For comparison, we also measured the baseline performance using the original models without retraining. With respect to web models, FPRs dropped slightly but TPRs improved significantly: 90.4% (no retraining) vs. 96.3% with 3-day retraining (see Table 7). Conversely, we observed insignificant changes in both FPRs and TPRs after retraining native models. We attribute this to already peaking validation set performance (even without retraining), and lack of repetitive sequences of the “N1, N2” campaigns in the test set period. Considering the fast model training times reported in Section 6.6 and minor performance differences between retraining frequencies, we recommend retraining models every 10 or so days based on the original-set retaining strategy described above.

7 Discussion

We revisit challenging case studies, and discuss real-world false positive and false negative implications.

7.1 False Positive Implications

We found high FPR cases in several countries when the web model was used: Uzbekistan (validation set: 2.1%, test set: 1.1%), and Bangladesh (test set: 1.0%). 92% of those FPR cases, however, incurred during short peak traffic cycles. Kuwait encountered high FPR with the native model in the validation settings (2.3%) but most of those false positives were indeed mislabeled samples. Small portion of real false positives, however, imply that unfortunate benign users (e.g., using attackers’ prefix) may experience rejections during or shortly after peak cycles. These observations imply that the adoption of the two models may incur small surge in FPRs when intense AIT attacks are being performed. But we argue that such FPRs would quickly disappear after a day or two (near the end of peak cycles) when blocked phone/IMEI prefix and email domains are no longer heavily reused. Libya and Indonesia, under the test set settings and native traffic, reported a different FPR trend: most of the false positives were associated with off-peak traffic. Our FPR analysis, however, revealed that they were all mislabeled samples; such cases would represent off-peak true positives in real-world settings, and would not affect usability. Nevertheless, we encourage service providers to adopt fallback methods (e.g., mobile authenticator app), and allow users to log in through other second-factor methods in case SMS continues to fail.

7.2 False Negative Implications

Ukraine reported noticeably high TPR (65.9%) with the optimized web model under the validation settings – Web-Dominant-Email, Web-Phone-Prefix, and Web-Validation techniques were employed in a mixed fashion, making it a very challenging attack campaign to detect. Our fine-grained

TPR analysis revealed that the performance changes significantly between the first 7-day traffic (82.3%) and the last 8-day traffic (50.7%) due to high variability in attack configurations and feature distributions. Our preliminary model retraining experiment demonstrated that including the latest dataset (such that would be retrieved through the Feature store in Figure 1) in the train set can be effective in improving robustness to feature distribution variability, and improving overall performance. We leave investigation of model retraining effects to future work.

In the case of the native model, we found one specific case from the test set with low TPRs: Sudan (77.1%). Although the attack samples came from what our algorithms identified as "peak" traffic, all samples scored less than 0.9 (false negatives) were associated with low-profile peak cycles comprising just 9,390 samples during that time span – in consequence, our country-wide features, which rely on rapid accumulation of prefix counts over 24 hours, became mostly ineffective. We argue that the real-world impact of such stealthy attack campaigns would be somewhat limited because of their significantly downsized traffic volumes.

Through those case studies, we learned about covertly executed attacks designed to accumulate long-term profit. As part of future work, we plan to identify and characterize attack campaigns that may be prevalent outside peak cycles, and explore features based on more extensive time windows (e.g., several days or weeks) to build a system for detecting more stealthy attack campaigns.

7.3 Limitations

Despite the fact that our labeling method (see Section 4.3) was designed to primarily inspect peak traffic (periods with very high probability of AIT attacks) and minimize false positives in other non-peak cycles, our individual within-peak labeling rules may still introduce small portion of false positives – this is because genuine users may also try to perform SMS authentication during peak cycles, and may have accidentally entered incorrect one-time codes several times or may have been trying to access an application that was also dominantly being exploited by an attack campaign. Such samples would have been labeled as positive samples too, and may translate to false positives in real-world settings (if our model rejects them). However, it was our design decision to optimize attack sample labeling rates within the identified peak (attack likely) cycles, and achieve uppermost performance in detecting large-volume peak attacks, while minimizing false positives in all other non-peak (normal traffic) time spans. If there is a strong need to further improve labeling accuracy within peak traffic cycles, one may try to seek additional evidence (e.g., service usage history) that indicates benign user behavior, and update labels.

Questions about generalization – applying the system to other unstudied countries – remain mostly unanswered. Our preliminary experiments, however, demonstrated robust performance on unknown countries which have been removed from the train set (see Section 6.4). We attribute this trend to the similarities in the AIT attack techniques that are commonly employed across multiple countries.

Our test results reported low TPRs on non-peak attacks. To gauge the severity of this system weakness, we measured rough sizes of non-peak attacks by selecting accounts or IMEIs that were used to submit more than 20 requests on a given day, and counted associated samples from all off-peak traffic: their volumes were only about 0.35% (web) and 0.02% (native) of peak attack volumes, however, implying low overall severity.

The two models are robust to evasive attacks that try to moderately downsize country-wide feature values. More sparse manipulations would yield about 51% attack success rate on the web traffic; native model continued to show resilience. Training a new web model after including evasive samples in the train set, however, significantly lowered attack success rate to 18%. We argue that low ROIs (profit vs. cost of acquiring new phone numbers, devices, and email accounts) would effectively demoralize such evasive attempts.

7.4 Real-World Deployment: Live Inspection

The proposed system would be integrated with an "Authentication Server" as the first line of defense (see Figure 1), performing live inspection on all incoming SMS authentication requests. Upon receiving a request, the "Feature Extractor" would be instructed to compute all features using last 24-hour traffic information. SMS Logs, which contain this history data, would be pre-loaded to the server memory through continuous batch execution: the memory needed to support this caching activity would be about 31 (web) and 253 (native) megabytes (see Section 6.6). This batch process would enable fast computation, adding about 31 (web) and 329 milliseconds (native) of inference latency during peak attack cycles. The "ML Server" takes features as input and computes an attack probability score. The "Detector" checks this score against a threshold (0.9), and returns the final inference result to the Authentication Server. If the inference result indicates an AIT attack, the request is blocked; otherwise, an one-time code would be sent to the specified phone number. Through this live inspection system we expect about 90% of the peak attack traffic to be blocked. Finally, request details and inference results are logged to SMS Logs: to anonymize PIIs, only the phone number prefix and IMEI prefix should be stored in plaintext (to compute prefix features); distinct occurrences can be counted using their encrypted values.

8 Related Work

This paper presents, to the best of our knowledge, the first formal analysis of AIT attack characteristics, and the systematic implementation and evaluation of AIT attack detection methods. However, we do acknowledge that there have been open articles, albeit non-academic ones, where security practitioners have shared their initial thoughts on this problem so we first relate our work to such posts. Then below, we also relate our work to research regarding detection methods of malicious bot activities in three heavily researched domains: Volumetric DDoS (Distributed Denial of Service), Malicious Web Crawlers, and Credential Stuffing.

AIT attacks. Recent online blogs [21, 22] suggest common set of countermeasures based on well-understood traditional security practices: adopting rate limit policies, detecting heavily-reused IPs or devices, and monitoring conversion rate drops and incoming traffic rates are commonly explained. Web-Phone-Prefix and Native-Phone-Prefix campaigns (see Section 5.3), systematically characterized through our own SMS log analysis, are also mentioned in several blogs [10, 14]: inline with our reports, the authors identify sequential number patterns, and suggest checking similarity between phone numbers. Implementation details, however, are absent – exact techniques for checking similarities efficiently, and translating the results into an actual detection rule or an ML feature remain unexplained. False positive implications are not explained. In Section 5.1, we explain the key challenges in measuring phone number similarities, and propose a novel country-wide feature engineering technique to efficiently capture phone prefix recurrence counts. To slow down bot activity the authors also recommend using CAPTCHA [14, 21, 22] but prior literature [9, 23–25] have shown that most existing CAPTCHAs can be bypassed through automated means.

Volumetric DDoS. DDoS attacks are performed by bots to disrupt a given target by generating traffic volumes that exceed the allowed bandwidth of the target. Research in this domain [6, 7, 15, 18, 27, 28, 30] employ features extracted from packet headers, single network connections, and multiple network connections. For example, a feature vector used in [15] includes protocol, the number of packet from source to destination, and the number of connections with the same destination address. In the case of DDoS attacks the same source and destination pair must be reused to some degree to bring down a target server – in an extreme AIT attack case, however, we found bots sending just a single request and changing every observable information (user account, device, IP, and phone number). This thinly scattered behavior makes it infeasible to apply traditional multi-event monitoring techniques. In this context, our country-wide features, which extract partial information from a given request and look for extensive clues from a bigger traffic, *provide the only known means* to timely detect scattered bot behaviors.

Malicious Web Crawler. Research in this domain [12, 13, 16, 17, 19, 26, 29] focus on bots used to automatically explore the open web and try to exploit any new found end points. The extensive analysis done in recent work with *honeysites* [16] show that such crawler bots exhibit interesting characteristics, and most methods proposed for their detection focus on the differentiating behaviors they typically exhibit in their exploration. Unfortunately, due to the nature of crawlers, most of these detection methods [12, 13, 17, 19, 26, 29] assume a certain volume of activity from a single entity (e.g. 30+ requests is assumed in [13] for any meaningful crawler activity) so they do not translate well into the extremely scattered nature of AIT attacks. As such, both lines of research should be considered orthogonal and complementary to each other.

Credential Stuffing. Bots are also often deployed for *credential stuffing*, online attacks designed to compromise user accounts through remote dictionary or brute-force attacks performed on web authentication servers. Though simple *three strike rules* have been used to thwart these attacks in the past, as attackers evolved to circumvent such restrictions, recent work [8, 11, 20] have proposed methods to differentiate legitimate login attempts from malicious ones. Among them, one recent paper [11] employs relevant features: inspecting number of unique usernames submitted, login inter-arrival time (mean and standard deviation), and login success rates. The first feature, inspecting system-wide count of unique usernames, resembles some of our country-wide feature. The fundamental difference, however, is that their login requests are first grouped by IP address – rendering them less applicable to the scattered attacks studied in this paper. The intuition behind the second (login inter-arrival times) and third (login success rate) features are similar to our own motivations for exploring various time-difference and conversion rate features.

9 Conclusion

To timely detect advanced AIT attacks designed to bypass rate limit policies, we proposed traffic monitoring at three different levels: single-event, multi-event, and country-wide inspection. Our novel country-wide features that inspect the proportion changes in SMS requests submitted by a given email domain, and single-event features that check for the use of short-lived email services are particularly effective in detecting attacks initiated through web clients: we recorded 89.6% TPR while keeping the FPR around 0.2%. Native client side attacks that misuse large volumes of fake phone numbers are effectively detected using country-wide features that count frequently reused prefix values. Further, single-event features that identify the use of outdated devices and native clients play a decisive role in detecting attacks that are initiated through rooted devices. Taken together, those features are primarily responsible for detecting 91.1% of all native attacks (at 0.1% FPR). Under intense peak attack traffic, our system would add about 31.2 and 329.4 milliseconds of detection latency in the case of web and native traffic. Considering that AIT attack detection systems (other than trivial rate limit policies and traffic volume inspection rules) currently do not exist, these high recall rates and extremely low FPRs are indeed promising.

10 Ethics Considerations

Although the authors' affiliation and the organization that shared SMS request log dataset are independent entities, the two are part of the same global corporate umbrella. While confidentially receiving the dataset – i.e., the dataset never left the corporate private and secure network – we inquired and checked with the internal legal and privacy protection office, and received confirmation that the conducted research and the act of publishing statistical results comply with the terms and conditions stated in users' consent for data collection and use.

Specifically, the collection and use of all personally identifiable information (PII), including phone numbers, IMEIs, and IP addresses, are explicitly stated in terms and conditions that users must read and agree upon creating an account. In fact, all log attributes disclosed in Section 4.2 (used to compute features) are explicitly mentioned in the "what information do we collect," "how do we use your information," and "how do we keep your information secure" sections.

Before conducting the presented research, we contacted internal legal office and privacy protection office, and completed both legal, privacy, and personal data compliance reviews with respect to (1) collecting and storing SMS authentication request logs (including all attributes disclosed in Section 4.2) from the studied authentication service, and (2) analyzing those logs to develop an AIT attack detection system for both web and native channels, and publishing statistical results and findings.

All SMS authentication logs were encrypted and made accessible only within the organization's secure private network – performing two-factor authentication was mandatory to access the data. Access to PIIs were governed with more stringent policies: the authors had to request specific PII access permissions on a daily basis (every 24 hours) to decrypt them. Decrypted PIIs (phone numbers and IMEIs) were merely used for initial AIT attack pattern analyses. Subsequent feature extraction, model training, and evaluation tasks were performed using just the prefix values, and the number of distinct occurrences were counted in the original encrypted form.

Benign end users may benefit from this research in two areas: (1) elimination of high-volume peak traffics would ensure more consistent SMS authentication latency and log in experience, and (2) prevention of genuine phone numbers being exploited by attackers, and users constantly having to deal with spam (unknown) SMS texts.

There is a risk that publishing feature engineering details and experimental results could provide means for adversaries to re-configure attack settings and perform evasive attacks. Our evasive attack analysis (see Section 6.5), however, revealed that such cost-intensive efforts would still lead to modest attack success rates (51% for web and 9% for native channels) yet significantly increase the overall attack preparation costs. Considering low return on investment, we argue it is somewhat unlikely that attackers would heavily invest in evasive attacks. Even if some attackers do proceed with evasive manipulations, overall attack success rates would be significantly lower than when rate limit policies are used alone (state-of-the-art).

Further, the presented feature engineering techniques are designed based on commonly available SMS authentication log attributes – to that end, we believe our features can be adapted quickly to other similar SMS-based authentication services, and facilitate deployment of their own highly effective AIT attack detection systems.

11 Open Science

The SMS authentication log dataset, collected through the real-world use of the studied authentication service, contains personal information and is protected by privacy laws. This dataset, governed by privacy laws and personal information processing policies, cannot be shared without end users' explicit agreements – to that end, sharing the large-scale log data would be infeasible.

Unfortunately, the ML model, feature engineering, and model training source codes cannot be shared because the proposed AIT attack detection system has been deployed on the studied authentication service, and is currently inspecting incoming SMS authentication requests and performing live detection. Source codes, if released, may be exploited by adversaries to find vulnerabilities in the AIT detection system, which would jeopardize system's security and end users' wellbeing. In this context, any code release would violate the internal information security policies, and jeopardize the integrity of formal security reviews conducted prior to system deployment.

References

[1] A2P SMS Under Siege – Artificially Inflated Traffic and its Impact on the Industry. https://info.enea.com/A2P_SMS_Under_Siege.

[2] Datasets: Imbalanced datasets. https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-datasets.

[3] Official Document TS.06 - IMEI Allocation and Approval Process. https://www.gsma.com/newsroom/wp-content/uploads//TS.06-v25.0-IMEI-Allocation-and-Approval-Process.pdf.

[4] National Numbering Plans. https://www.itu.int/oth/T0202.aspx?parent=T0202.

[5] Artificial generation of a2p sms traffic: The twitter case study and beyond. https://lancktele.com/blog/artificial-a2p-sms-traffic-twitter/.

[6] C. Busse-Grawitz, R. Meier, A. Dietmüller, T. Bühler, and L. Vanbever. pforest: In-network inference with random forests. *arXiv preprint arXiv:1909.05680*, 2019.

[7] C. Fu, Q. Li, M. Shen, and K. Xu. Frequency domain feature based robust malicious traffic detection. *IEEE/ACM Transactions on Networking*, 31(1), 2022.

[8] C. Herley and S. E. Schechter. Distinguishing attacks from legitimate authentication traffic at scale. In *Proceedings of the 26th Network and Distributed System Security Symposium (NDSS)*, 2019.

[9] M. I. Hossen, Y. Tu, M. F. Rabby, M. N. Islam, H. Cao, and X. Hei. An object detection based solver for {Google's} image {reCAPTCHA} v2. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2020.

[10] infobip. What is SMS pumping? https://www.infobip.com/glossary/sms-pumping, 2024.

[11] M. Islam, M. S. Bohuk, P. Chung, T. Ristenpart, and R. Chatterjee. Araña: Discovering and characterizing password guessing attacks in practice. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*, 2023.

[12] G. Jacob, E. Kirda, C. Kruegel, and G. Vigna. {PUBCRAWL}: Protecting users and businesses from {CRAWLers}. In *Proceedings of the 21st USENIX Security Symposium (USENIX Security)*, 2012.

[13] S. T. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath. Throwing darts in the dark? detecting bots with limited data using neural data augmentation. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (SP)*, 2020.

[14] A. Labs. SMS Pumping Fraud: How to Spot and Stop It. https://www.arkoselabs.com/toll-fraud/sms-pumping-fraud/, 2024.

[15] J.-H. Lee and K. Singh. Switchtree: in-network computing and traffic analyses with random forests. *Neural Computing and Applications*, 2020.

[16] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis. Good bot, bad bot: Characterizing automated browsing activity. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP)*, 2021.

[17] X. Li, B. Amin Azad, A. Rahmati, and N. Nikiforakis. Scan me if you can: Understanding and detecting unwanted vulnerability scanning. In *Proceedings of the 32nd ACM Web Conference (WWW)*, 2023.

[18] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar. Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches. In *Proceedings of the 30th USENIX Security Symposium (USENIX Security)*, 2021.

[19] A. G. Lourenço and O. O. Belo. Catching web crawlers in the act. In *Proceedings of the 6th International Conference on Web Engineering (ICWE)*, 2006.

[20] S. Schechter, Y. Tian, and C. Herley. Stopguessing: Using guessed passwords to thwart online guessing. In *Proceedings of the 4th IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019.

[21] J. Schuster. Artificially Inflated Traffic (AIT): how to prevent fraud of A2P SMS. https://www.linkmobility.com/blog/artificially-inflated-traffic-ait-how-to-prevent-fraud-of-a2p-sms, 2023.

[22] sinch. What is Artificial Inflation of Traffic (AIT)? https://www.sinch.com/blog/artificial-inflation-traffic-ait-growing-threat-messaging-ecosystem/, 2023.

[23] S. Sivakorn, J. Polakis, and A. D. Keromytis. I'm not a human: Breaking the google recaptcha. *Black Hat*, 2016.

[24] X. Teoh, Y. Lin, R. Liu, Z. Huang, and J. S. Dong. {PhishDecloaker}: Detecting {CAPTCHA-cloaked} phishing websites via hybrid vision-based interactive models. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security)*, 2024.

[25] D. Wang, M. Moh, and T.-S. Moh. Using deep learning to solve google recaptcha v2's image challenges. In *Proceedings of the 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020.

[26] G. Xie, H. Hang, and M. Faloutsos. Scanner hunter: Understanding http scanning traffic. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*, 2014.

[27] G. Xie, Q. Li, Y. Dong, G. Duan, Y. Jiang, and J. Duan. Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation. In *Proceedings of the 41st IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[28] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, 2020.

[29] N. Zhong, J. Liu, P.-N. Tan, and V. Kumar. Discovery of web robot sessions based on their navigational patterns. *Intelligent Technologies for Information Analysis*, 2004.

[30] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu. An efficient design of intelligent network data plane. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security)*, 2023.

Appendix

A Data Set Description

Table 8 summarizes the total number of attack samples and genuine samples labeled for each country. The labeling method described in Section 4.3 was used to identify those attack samples. This table also shows the total number of SMS logs obtained for each country. Table 9 shows the train set construction for each country, summarizing the attack techniques found, and the size of attack set and genuine set.

Table 8: Number of attack samples and genuine samples labeled in each country. "Web" and "Native" indicate web and native client traffic. Countries are sorted based on the total attack set size.

Channel	Country	# SMS requests		
		Attack	Genuine	Total
Web	Bangladesh	783,524	54,535	838,059
	Ukraine	303,667	49,230	352,897
	Pakistan	226,633	100,811	327,444
	Azerbaijan	223,873	13,441	247,314
	Morocco	214,257	40,619	254,876
	Nigeria	208,047	68,605	276,652
	Sri Lanka	85,142	22,084	107,226
	Uzbekistan	50,224	22,089	72,313
	Libya	39,209	9,196	48,405
	Russia	38,418	352,386	390,804
	Total	2,182,994	732,996	2,915,990
Native	Indonesia	955,084	2,206,603	3,161,687
	Niger	361,192	25,475	386,667
	Zimbabwe	320,818	73,779	394,597
	Zambia	247,966	58,662	306,628
	Kuwait	238,598	70,182	308,780
	Ghana	195,678	177,487	373,165
	Sudan	188,988	159,095	348,083
	Lesotho	139,775	9,668	149,443
	Peru	65,290	830,987	896,277
	Libya	50,716	151,515	202,231
	Total	2,764,105	3,763,451	6,527,556

Table 9: Number of attack samples and genuine samples in the train set constructed for each country. "Web" and "Native" indicate web and native client traffic. Countries are sorted by the number of attack samples. W1, W2, W3, and W4 represent "Web-Short-Email", "Web-Dominant-Email", "Web-Phone-Prefix", and "Web-Validation" attack techniques, respectively. "Not-peak" indicates small number of attack samples belonging to off-peak traffic. N1, N2, and N3 represent "Native-Phone-Prefix", "Native-IMEI-Prefix", and "Native-Old-Client" attack techniques, respectively.

Channel	Country	Attack Techniques	# SMS requests	
			Attack	Genuine
Web	Bangladesh	W1,W2,W3	627,498	35,708
	Pakistan	W1	226,161	67,652
	Morocco	W1	204,274	26,682
	Nigeria	W2,W3	207,871	47,498
	Ukraine	W2,W4	188,097	35,891
	Azerbaijan	W2,W3,W4	177,621	10,492
	Russia	W2,W4	37,704	235,795
	Uzbekistan	W2,W3	20,745	15,365
	Libya	Not-peak	862	7,593
	Sri Lanka	Not-peak	120	15,241
	Total		1,690,953	497,917
Native	Indonesia	N1,N3	504,264	1,458,268
	Peru	N2	64,305	552,578
	Zambia	N1,N2,N3	49,280	40,101
	Ghana	N1,N2,N3	41,621	122,991
	Lesotho	N1,N2,N3	39,879	6,056
	Libya	N1,N2,N3	39,383	110,866
	Sudan	N1,N3	9,169	107,120
	Niger	N1,N2,N3	2,136	16,154
	Zimbabwe	Not-peak	310	47,616
	Kuwait	Not-peak	73	48,172
	Total		750,420	2,509,922

B Attack Characteristics and Feature Distributions

B.1 Web Attacks

B.1.1 Short-Lived Email Services (Web-Short-Email)

The median `em-domain-sms-diff` for attacks associated with short-lived email services, which we measured roughly using attack samples from Bangladesh, was just 66 days – significantly smaller than the 5,277 days observed in the case of negative samples. The first graph is Figure 3 (a) compares the distribution of `em-domain-sms-diff` values between genuine (blue) and attack samples (orange).

B.1.2 Dominant Email Services (Web-Dominant-Email)

The second graph in Figure 3 (a) compares the distribution of `em-domain-prop-change` values between the two classes. About 75% of attacks associated with well-known services recorded `em-domain-prop-change` values greater than 0.1 (more than 10% increase in 24-hour usage rate) whereas only about 1.0% of genuine traffic recorded 0.1 or greater increase.

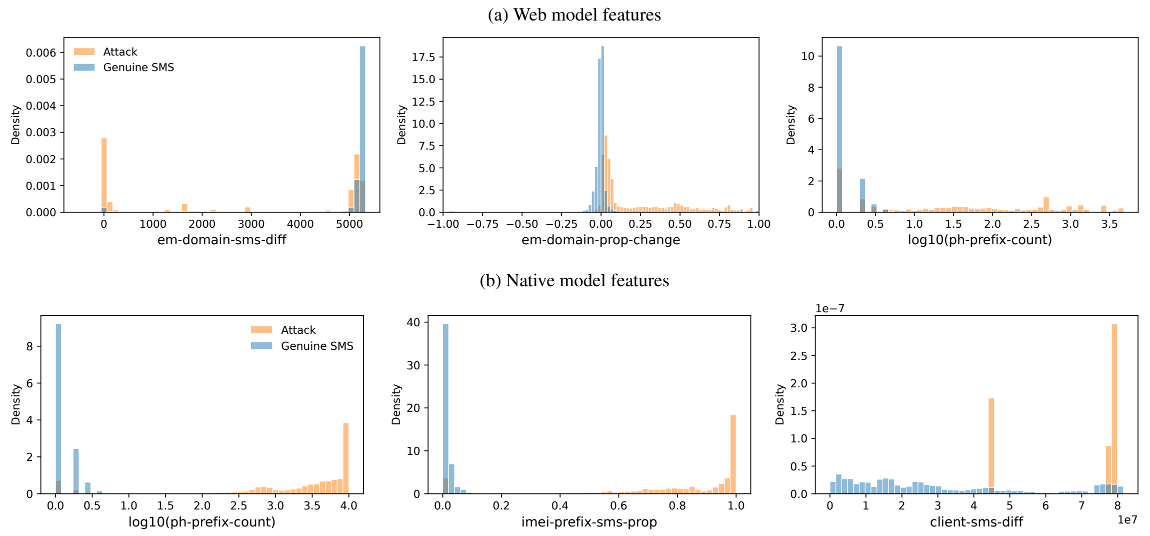


Figure 3: Distribution plots representing web model features (top) and native model features (bottom). Orange and blue lines represent attack set and genuine set distributions, respectively.

B.1.3 Phone Number Prefix Reuse (Web-Phone-Prefix)

About 82% of what appears to be prefix reuse attack recorded `ph-prefix-count` value larger than 100 (i.e., more than 100 distinct phone numbers with the same prefix were found in an 24-hour traffic); the same analysis performed on the genuine traffic reported only a handful of samples with a value greater than 100 (third graph in Figure 3 (a) compares the two distributions).

B.2 Native Attacks

B.2.1 Phone Number Prefix Reuse (Native-Phone-Prefix)

About 91% of the native attack samples recorded `ph-prefix-count` values greater than 30 (a suspicious count), demonstrating a more dominant prefix reuse behaviors on the native side. Conversely, only 0.01% of the genuine samples recorded values greater than 30 – in fact, 99.9% of the genuine samples recorded values less than 10 (the first graph in Figure 3 (b) compares the two distributions). About 92% of the attack samples were associated with `ph-prefix-conv-rate` of zero, while only 21% of genuine samples were associated with zero conversion rate.

B.2.2 IMEI Prefix Reuse (Native-IMEI-Prefix).

About 77% of attack samples were associated with `imei-prefix-sms-prop` larger than 0.2, demonstrating the prevalence of IMEI prefix reuse behaviors. Conversely, only 0.02% of genuine samples showed `imei-prefix-sms-prop` values larger than 0.2 (the second graph in Figure 3 (b) depicts the distribution of the two classes).

B.2.3 Old Native Clients (Native-Old-Client).

The third graph in Figure 3 (b) compares the distributions of the two classes with respect to the `clint-sms-diff` feature. All three attack techniques were prevalent in the four countries depicted in Figure 2 (b).

C Test Set Probability Score Distribution Graphs

To demonstrate the superiority of the two optimized models over their preliminary models, we depict the comparative probability score distribution graphs for the web model in Figure 4 and native model in Figure 5. The graphs on the left represent the two preliminary models, and the blue density bars represent genuine samples in the test set; the orange bars represent attack samples. More distinct separations between the two distributions can be seen in the optimized models.

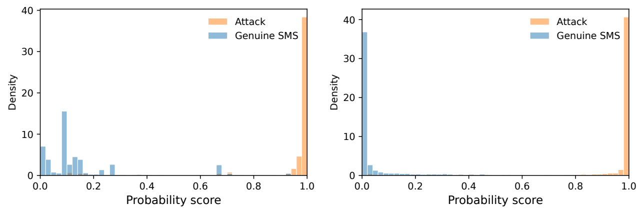


Figure 4: Test score distributions for the web preliminary model (left) and optimized model (right).

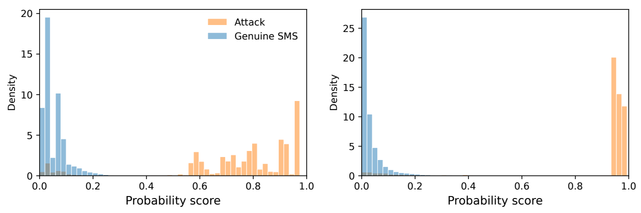


Figure 5: Test score distributions for the native preliminary model (left) and optimized model (right).

D Evasion Attack Details

D.1 Five Different Evasive Attack Settings

The first part of the evasive attack experiment involved measuring the original models' performance against five different evasive settings: to construct five evasive test sets, we selected all genuine samples from the test set, and replaced country-wide features using the values that represent 50, 40, 30, 20 and 10th percentile in the test attack set. Table 10 shows how the two original models' TPR changes across the five different evasive test sets (P_{50}, P_{40}, P_{30}, P_{20}, and P_{10}). "Original" indicates the TPRs measured on the original test set. The web model started showing some weaknesses when 30th percentile values were used for manipulation. Conversely, the native model showed robust performance across all five evasive settings.

The second part of the experiment involved training the web models again after including evasive attack samples in the train set, and measuring their performance on the five evasive test sets. We constructed this evasive train set by selecting the genuine samples from the original train set, and replacing country-wide features with values that represent 50, 40, 30, 20, and 10th percentile in the train attack set. Five separate web models were trained based on the original train set and the newly constructed evasive set. Table 11 shows how the TPRs of those models change across the five evasive test sets. A highly resilient model trained with evasive samples representing 30th percentile values achieved 82.0% and 45.7% TPRs on extensively sparse evasive sets constructed using 30th and 20th percentile values, respectively.

Table 10: TPRs of the original two models measured using the five different evasive test sets constructed by selecting all genuine samples (from the test set), and replacing country-wide values with the 50, 40, 30, 20, and 10th percentile values found in the test attack set. "P_n" indicates an evasive test set generated with n-th percentile values. "Original" indicates the original test set TPR.

Channel	**TPR(%)**					
	Original	P_{50}	P_{40}	P_{30}	P_{20}	P_{10}
Web	89.55	99.98	100.00	49.03	4.06	0.81
Native	91.11	90.82	90.82	90.82	90.82	90.82

Table 11: Performance of web models trained after including evasive attack samples in the train set. Five such models were trained using evasive samples constructed by selecting train set genuine samples, and replacing country-wide features with values that represent 50, 40, 30, 20, and 10th percentile in the train attack set. We measured their TPRs and FPRs based on the five evasive test sets. "P_n" indicates an evasive test set generated with n-th percentile values, and "P_n model" indicates a new web model trained with original train set and n-th evasive train set.

Model	**TPR(%)**						**FPR(%)**
	Original	P_{50}	P_{40}	P_{30}	P_{20}	P_{10}	
Original model	89.55	99.98	100.00	49.03	4.06	0.81	0.19
P_{50} **model**	91.45	100.00	100.00	98.45	5.13	1.48	0.16
P_{40} **model**	90.54	100.00	100.00	99.99	2.84	0.89	0.15
P_{30} **model**	91.90	100.00	100.00	81.97	45.68	1.37	0.15
P_{20} **model**	90.23	99.79	99.56	59.72	4.60	1.45	0.17
P_{10} **model**	89.81	99.85	99.91	40.34	4.17	1.36	0.27

D.2 Extreme Native Setup: Validating All SMS Authentication Requests

We also experimented with an extremely evasive native attack scenario: attackers purchasing tens of thousands of new phone numbers and devices to perform `is-ph-verified` (and setting the value as "1"), which is ranked as the second most important native model feature. Attackers also develop efficient means to validate SMS requests, and transform all converion rate feature values to "1.0." In this widly hypothetical scenario, the original model would break: reporting just 7.5% TPR when 50th percentile country-wide feature values are used for manipulation. Even in this highly improbable scenario, however, a new model trained with an evasive train set constructed using 40th percentile values reported about 75% TPR (0.14% FPR) across all five evasive test set settings. We argue that low ROIs (25% success rates) should discourage such costly evasive efforts.