Him of Many Faces: Characterizing Billion-scale Adversarial and Benign Browser Fingerprints on Commercial Websites

Shujiang Wu, Pengfei Sun†, Yao Zhao†, and Yinzhi Cao
Johns Hopkins University, †F5, Inc.
{swu68, yinzhi.cao}@jhu.edu, {p.sun, y.zhao}@f5.com

***Abstract*—Browser fingerprints, while traditionally being used for web tracking, have recently been adopted more and more often for defense or detection of various attacks targeting real-world websites. Faced with these situations, adversaries also upgrade their weapons to generate their own fingerprints—defined as adversarial fingerprints—to bypass existing defense or detection. Naturally, such adversarial fingerprints are different from benign ones from user browsers because they are generated intentionally for defense bypass. However, no prior works have studied such differences in the wild by comparing adversarial with benign fingerprints let alone how adversarial fingerprints are generated.**

In this paper, we present the first billion-scale measurement study of browser fingerprints collected from 14 major commercial websites (all ranked among Alexa/Tranco top 10,000). We further classify these fingerprints into either adversarial or benign using a learning-based, feedback-driven fraud and bot detection system from F5, Inc., and then study their differences. Our results draw three major observations: (i) adversarial fingerprints are significantly different from benign ones in many metrics, e.g., entropy, unique rate, and evolution speed, (ii) adversaries are adopting various tools and strategies to generate adversarial fingerprints, and (iii) adversarial fingerprints vary across different attack types, e.g., from content scraping to fraud transactions.

I. Introduction

Browser fingerprint, an alternative to cookies, is an identifier consisted of a list of client-side features (such as user agent and canvas rendering results) to represent a browser instance. While the initial use of browser fingerprint [30] is for web tracking, e.g., personalized advertisement (which is often considered as a violation of web privacy), recent advances put browser fingerprints in a more positive yet defensive role in preventing cyber-attacks. For example, Azad et al. [10] show that browser fingerprints are used as one important factor for bot detection. For another example, Laperdrix et al. [46] find that online financial transactions often check browser fingerprints to prevent fraud logins and payments.

As always, there exists a tussle between adversaries and defenders: This also applies to browser fingerprint as a defense especially given that it is a weak means in browser identification depending on the entropy. Specifically, adversaries can change their browser fingerprints to bypass existing defenses. Say, for example, fingerprints are used to detect bot: If the number of requests with a certain fingerprint and some other features (e.g., lower-layer ones like IP) exceeds a threshold, the client is detected and blocked as a bot. Then, an adversary can keep changing their fingerprint to avoid being blocked. Such fingerprints forged by adversaries are thus defined as *adversarial browser fingerprints* (or for short adversarial fingerprints), and those traditional fingerprints from user browsers as *benign fingerprints* in the paper.

Intuitively, adversarial fingerprints are different from benign ones because their purposes are to defeat defenses. While intuitively true, the properties of these adversarial fingerprints in the wild are largely under-studied and unknown to the research community. For example, some open research questions include but are not limited to the following: (i) how adversarial fingerprints differ from benign fingerprints in terms of entropy and features, (ii) what strategies and tools adversaries adopt to change fingerprints especially for different attacks, and (iii) how adversarial fingerprints differ from one attack type to another. The answers to these questions will not only shed a light on future detection of adversarial fingerprints (and the associated attacks), but also help researchers to better understand the properties of benign fingerprints for some tracking purposes like personalized advertisement (where adversarial fingerprints should be excluded).

State-of-the-art works conducted small-scale (e.g., at most million-scale) studies for browser fingerprints generally from three major sources: controlled user groups [48], [48], blindly-collected website traffic [36], [50], and honeypot website traffic [53]. However, regardless of their scale, none of prior measurement work compare adversarial and benign fingerprints from the same website. Controlled user groups and honeypot websites only have adversarial or benign fingerprints; blindly-collected website traffic does not differentiate whether a fingerprint is benign or adversarial. That is, none of them answers the aforementioned research questions on the practical difference between adversarial and benign fingerprints in the real world as well as the tools and strategies to generate adversarial fingerprints for different types of attacks.

In this paper, we perform the *first* billion-scale measurement study of adversarial and benign fingerprints collected from 14 commercial websites (including major financial insti-

Network and Distributed System Security (NDSS) Symposium 2023
27 February - 3 March 2023, San Diego, CA, USA
ISBN 1-891562-83-5
https://dx.doi.org/10.14722/ndss.2023.24394
www.ndss-symposium.org

tutes, restaurants, and airlines). Specifically, we cooperate with a security company, F5, Inc.,[1] to collect web traffic and classify traffic using its bot and fraud detection and defense system. Then, browser fingerprints associated with bot and fraud traffic are considered as adversarial and the rest as benign. Note that an external, independent review shows that the bot and fraud detection system reduces costs caused by account takeover attacks by 96% and the total number of fake accounts by 92%.

Our measurement happens between January 2021 and June 2021 and results in 36 billion HTTP(s) requests/responses together with corresponding browser fingerprints from 14 commercial websites. The detection system classifies 42.5% as bot or fraud (i.e., malicious) traffic with five detailed attack types and 57.5% as user traffic (i.e., benign). The fingerprints associated with bot or fraud traffic are considered as adversarial and those with user traffic as benign. We then compare adversarial with benign fingerprints in terms of different metrics like evolution and entropy, and analyze the properties of adversarial fingerprints for generative tools and strategies. For example, we consider that a fingerprint is generated using a scripting tool if all JavaScript features are missing, or a virtual machine (VM) tool if the collected rendering method has consistent VM-related values. For another example, we consider that a fingerprint feature of a browser instance is blocked if its value is empty, or randomized if none of its values appear in benign fingerprints. Our analysis makes the following three observations:

- Observation-1: Adversarial browser fingerprints are significantly different from benign ones on the same website. Only 1.6% of total unique fingerprints are shared between adversarial and benign and the rest are either adversarial (8.1%) or benign (90.3%). They also differ from each other in many aspects, such as evolution speed and entropy. Take evolution for example. Benign fingerprints tend to evolve over time as browser instances get updated; by contrast, adversarial fingerprints are relatively stable because adversaries often bind fingerprints with compromised accounts and abandon fake accounts after one-time use.
- Observation-2: Adversaries are adopting different tools to generate adversarial fingerprints, manifesting varied properties. Specifically, we summarize three general types of tools, i.e., scripting, emulated browser (e.g, headless or full browser), and virtual machine (VM). Scripting tools are the most popular ones especially in scraping web contents, because of its high efficiency; by contrast, emulated browsers and VMs are less popular but more powerful in not only generating adversarial fingerprints but also simulating benign ones.
- Observation-3: The properties of adversarial browser fingerprints vary from one attack to another. Take content scraping and fraud transactions for example. Adversarial fingerprints used in content scraping tend to change very often yet randomly to avoid being blocked by the website; by contrast, adversarial fingerprints used in fraud transactions tend to mimic the characteristics of benign fingerprints and achieve a higher success transaction rate.

[1]F5, Inc. serves 48 of Top 50 fortune companies, which include Top 10 global telecommunication operators, Top 30 U.S. commercial banks, Top 10 large global insurance companies, and Top 10 US retail companies.

Because our measurement study involves private user transactions and a paid commercial security product, unfortunately we have to keep our 36-billion dataset and the bot and fraud detection system private per our agreement with the security company. At the same time, in the spirit of open-science, we are making our fingerprint analysis tool open-source at this anonymous github repository.[2] We would like to encourage future researchers to use our open-sourced fingerprint analysis tool and reproduce the results on other websites.

II. Background

In this background section, we first describe the definition of both adversarial and benign browser fingerprint and then present our threat model. A browser fingerprint is a combination of browser features collected explicitly or implicitly (like via a side channel) at a client, such as user agent, WebGL rendering, and a list of fonts. The traditional definition of browser fingerprint assumes that it is coming from a user-controlled browser, thus called benign browser fingerprint. Being contrary to benign fingerprints, an adversarial browser fingerprint, or for short *adversarial fingerprint*, is forged by an adversary to bypass certain server-side defenses utilizing browser fingerprints. An adversary can use various tools, e.g., scripting tools (e.g., written in Python), emulated browsers (e.g., automated by Selenium), and virtual machines, to craft fingerprint features for adversarial fingerprints.

A. Threat Model

Our threat model takes into consideration of attacks that bypass server-side defenses relying on browser fingerprints. More specifically, our threat model includes the following attacks based on their popularity in real-world web traffics:

- Account takeover attempts (i.e., credential stuffing). An account takeover attempt [11], [67], [76], [82] is that an adversary try to login a large amount of user accounts with different username and password combinations that are often obtained either from underground economy or previous data leaks. Such an attack is also known as credential stuffing.
- Fraud transaction. A fraud [54], [55], [66]is that an adversary initiates an unauthorized transaction—e.g., making orders, personal loan application, sending/transferring money, checking out items, and returning ordered items—on behalf of a victim. Fraud usually happens after a successful account takeover, although some frauds may not require account logins.
- Automated fake account signup. Automated fake account signup is automatic creation of accounts by a bot instead of a human being. Such accounts, according to prior works [15], [16], [32], [68], can be used for many malicious purposes, such as scraping information after the login wall, posting spam or fake reviews, and signing up extra bonus of the websites.
- Aggressive content scraping. Aggressive content scraping [28], [37], [64] is that an adversary crawls a target website disregarding the rate limit and blacklist documented in the `robot.txt` file for their own benefits, e.g., obtaining a competitor's information like airline price. Such aggressive scraping is performed either based on a fake account (called logged-in) or anonymously (i.e., without login).

[2]https://github.com/bfpmeasurementgithub/browser-fingeprint-measurement

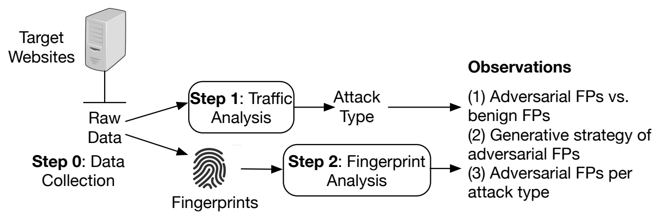
Target Websites
Raw Data
Step 0: Data Collection
Step 1: Traffic Analysis
Attack Type
Fingerprints
Step 2: Fingerprint Analysis
Observations
(1) Adversarial FPs vs. benign FPs
(2) Generative strategy of adversarial FPs
(3) Adversarial FPs per attack type

Fig. 1. An overall architecture of our measurement methodology.

- Giftcard cracking. Giftcard cracking [1] is a brute force attack that enumerates all possibilities of giftcard numbers and tries to spend values before the legitimate user(s).

III. Measurement Methodology

In this section, we present our measurement methodology in characterizing adversarial and benign fingerprints. Figure 1 shows the overall architecture of the measurement, which has three steps: 0) data collection, 1) traffic analysis, and 2) fingerprint analysis. Step 0 collects all the raw data including the entire HTTP(s) requests and responses. Then, the traffic analysis in Step 1 classifies the HTTP request as either benign or malicious with detailed attack type. Lastly, the fingerprint analysis in Step 2 studies the properties of all the fingerprints together with outputs from Step 1, which leads to our three core observations.

Let us start from Step 0 of our measurement study. We cooperate with a security company to collect data from 14 top-ranked target websites (104 subdomains).[3] The collection is performed via both JavaScript implanted into the web applications of target websites and an SDK on the native mobile applications of these target. Then, the collected raw data contains rich information across network-, browser-, and user-levels. The network-level information contains TCP/IP headers (e.g., source and destination IP addresses), device IDs, timestamps, and Source Autonomous System Number (ASN). The browser-level information mainly contains browser fingerprints that are broken down into different features, such as user agent, font list, and canvas images. (Details are shown later in Section IV-B.) The user-level information includes account ID and user behaviors (such as visited links).

A. *Step 1: Traffic Analysis*

Our Step 1, as shown in Figure 2, has two sub-steps: a learning-based bot and fraud detection/defense (which keeps accepting feedback from customer and offline system) and an attack type classification engine.

1) Bot and Fraud Detection/Defense: The bot and fraud detection/defense platform shown in Figure 2 takes raw data as inputs and outputs whether a given HTTP request and response pair belongs to bot or fraud traffic. The key of the platform is a double-model and double-feedback-loop system that is robust to, and can quickly recover from, adaptive bot and fraud attacks. Let us start from the double model. The platform is equipped with two models: (i) an online, real-time model that accepts high-performance features selected from feature engineering (i.e., those that can be computed in a short time) and outputs a decision to target websites, and (ii) an offline model that accepts all features including slow ones for a comprehensive decision.

[3]Due to our agreement as part of the measurement study, we anonymize the 14 companies' websites (with only the category name) belong to the different categories thus we can compare the difference.

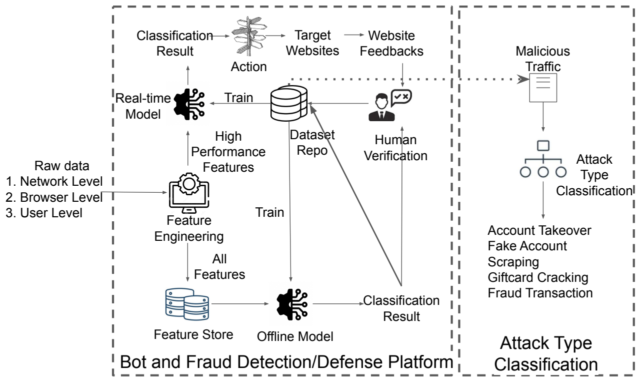

Fig. 2. Details of Step 1: Traffic Analysis

TABLE I. A selective list of features used in real-time and offline models

	Feature Name	Description
Real-time Model	URL	URL in an HTTP request
	Cookie	HTTP cookie
	TCP/IP FP	TCP/IP fingerprint from TCP/IP stack [2]
	IP Address	IP address
	ASN	Autonomous System Number
	Username	User name in hash value
	TLS Fingerprint	Extracted from TLS Client Hello message [35]
	User Behavior	A list of user-behavior pattern, e.g., the presence of mouse moves, touch moves, and key down/up [85]
Offline Model	IP Reputation	Behavioral quality of an IP address and # of malicious requests that it sends
	ASN Reputation	Malice probability of any given active IP in an ASN
	User Reputation	Percentage of benign/malicious requests per user in history
	Device Reputation	Percentage of benign/malicious requests per device in history
	Header Reputation	Header inconsistent rate and percentage of benign malicious requests per HTTP header [86]
	Behavior per session	# of user behaviors per HTTP session

Table I shows a selective list of top important features used in both real-time and offline models based on their contributions to the model's performance. The real-time model mostly adopts high-performance features that can directly obtained from a single HTTP request, such as URL, cookie, TCP/IP fingerprint, and SSL fingerprint. As a comparison, the offline model uses aggregated statistics of features for many HTTP requests over a certain time period. Examples are like the number of benign and malicious request per IP, ASN, username, device, and HTTP header. Note that we only list a selective number of features. In practice, the real-time model has hundreds of features and the offline has thousands (as there are different combinations of feature aggregation).

Note that the initial training set of the double model before the double feedback loop is created using a rule-based approach with human verification. Table II shows a selective list of metrics used to create such an initial training set. The high level idea is to filter web traffic based on abnormality and then

TABLE II. THE INITIAL TRAINING SET CREATION FOR THE DOUBLE-MODEL BEFORE DOUBLE-FEEDBACK LOOP.

Metrics	Abnormality Description
Page view	Abnormally high page views traffic than the average
Referrer traffic	High referral traffic from uncommon websites
Bounce rate	High number of users joining and leaving without activities
Session duration	High or low average session duration
Regional traffic	Spikes in traffic from a certain region
Content refresh rate	High rate of content refreshing
Fake information	Fake (random) information like email, phone and name

verify such traffic using human experts. Take session duration for example. Bots are often automated and therefore may have a much shorter session duration compared with human beings. Similarly, bot accounts often use fake information like a random phone number during registration. Such a training set is initial and further improved in the double feedback loop.

Now, we describe the double feedback loops. The first feedback loop is between websites and the real-time model. That is, if the platform makes incorrect decisions (e.g., bot as benign or benign as bot) and takes wrong actions, such feedback will be collected from the target websites, verified by human experts, and updated to the real-time model during retraining. The second feedback loop is between the real-time and offline models. The offline model will make a separate, (more accurate) decision that is independent of the real-time model. Then, the offline model's analysis results are updated to the dataset repository and thus the real-time model in the second feedback loop. Note that both the real-time and offline models are audited by human experts. More specifically, human experts are involved in the following conditions: (i) customers (e.g., website users) reporting false positives or negatives, (ii) discrepancies between real-time and offline models, and (iii) a massive amount of attacks being reported. In addition, human experts also sample data samples predicted with low confidence scores to double check the accuracy of both models.

The advantage of the double-model and double-feedback-loop architecture is a great reduction of adaptive bot and fraud attacks. The reason is that even if an adversary bypasses the system, the feedback loop can quickly pick the bypass up and update the real-time model to detect such bypasses. An external review team examines the bot and fraud detection/defense product. The examination shows that that the platform product, in a long term, reduces (i) costs caused by account takeover attacks by 96% and (ii) the number of fake accounts by 92%. We would like to point out that although this number is very high at a first glance, it is actually reasonable and also required for a paid, commercial product. On one hand, the platform rarely misclassifies benign traffic; on the other hand, the 4% of account takeover and 8% of fake accounts (as verified by the external team) happen only within a very small amount of web traffic and a small time window and are quickly detected by the platform afterward.

2) Attack Type Classification: After the first step, we further classify malicious traffic into five types as described in our Threat Model (Section II-A). The methodology is based on important features provided by the bot and fraud detection/defense platform. We now describe five different types of attacks and their key features separately. Here are the details: (i) Account takeover attempt: the number of (incorrect) username and password combinations from a given client, (ii) Fraud transaction: when and where transactions happened, (iii) Automated fake account signup: the number of the newly-created accounts of a client, (iv) Aggressive content scraping: whether `robots.txt` is honored during crawling, and (v) Giftcard cracking: the number of tried giftcards of a client.

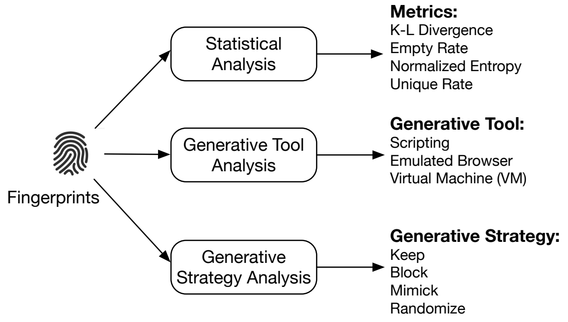

Fig. 3. Details of "Step 2: Fingerprint analysis".

B. Step 2: Fingerprint Analysis

Figure 3 shows the overall procedure of our "Step 2: Fingerprint Analysis". Specifically, this step has three substeps: statistical analysis (which calculates fingerprint metrics), generative tool analysis, and generative strategy analysis.

1) Statistical Analysis: In this substep, we calculate the following metrics and compare adversarial and benign fingerprinting using these metrics.

- K-L Divergence. Kullback–Leibler (K-L) divergence (also called relative entropy) is a statistical distance measuring how one probability distribution differs from the other reference probability distribution. Consider the calculation of the K-L divergence for a specific feature for two sets of fingerprints. For each set, we obtain all the values for the feature and the distribution of the values by calculating the percentage of each value within the set. Then, we compute the K-L divergence between two distributions of two sets. That is, the larger value a K-L divergence is, the more statistical differences two distributions have. In this paper, we randomly divide adversarial and benign fingerprints into two sets and calculate D(adversarial || benign), D(benign || benign), and D(adversarial || adversarial). Each calculation is repeated ten times with different partitions for the average value with standard deviation. Note that we use K-L divergence as a metrics because it can reflect whether the percentage of feature values differs between adversarial and benign fingerprints. Say 90% of benign fingerprints uses Screen Resolution A and 10% uses Screen Resolution B. By contrast, 10% of adversarial uses A, but 90% use B. Then, D(adversarial || benign) is 0.76, which is much larger than zero for identical distributions.
- Normalized Entropy. Normalized entropy,[4] a metrics widely used by prior works [36], [50], is defined in Equation 1:

$$NH = \frac{H(X)}{H_M} = \frac{-\Sigma_i P(x_i) log_2 P(x_i)}{log_2(N)} \quad (1)$$

where x_i represents a fingerprint, $P(x_i)$ is the probability of x_i in the dataset, and N is the total number of fingerprints.

[4]Note that in the context of this paper, "entropy" and "normalized entropy" are used interchangeably as the same terminology for convenience.

The larger NH is, the better the feature is in differentiating different browser instances. We followed the same partition used in the K-L divergence calculation to compute the average normalized entropy value with the standard deviation.

- Empty and Unique Rate. Empty rate is the percentage of fingerprints with an empty value on a certain feature. Then, unique rate defines the percentage of feature values or fingerprints that are either only benign or adversarial.

2) Generative Tool Analysis: In this substep, we analyze adversarial fingerprints and infer the tool that generates these adversarial fingerprints. Specifically, we describe three general types of adversarial tools used in the wild and our detection methodology.

- Scripting tools. Scripting tools are simple applications (e.g., written in Python) that send HTTP requests to target websites. On one hand, such tools often cannot implement complex client functionalities driven by JavaScript, e.g., rendering canvas images; on the other hand, they are often very fast, making them scalable to crawl large amount of contents. Because scripting tools generally do not support JavaScript, we detect scripting tools if none of JavaScript features exists in a fingerprint.
- Emulated browsers. Emulated browsers are those with extended or modified functionalities, such as headless browsers, browsers with extensions, and tailor-made browsers with modifications, which are often driven by automated tools like Selenium. Such emulated browsers have the full capability in simulating all different features of browser fingerprint. The default tool is an emulated browser if we cannot determine the tool as scripting or virtual machine (below).
- Virtual machines. Virtual machines (VMs) driven by software such as KVM and VMWare are also used in combination with emulated browsers to emulate real-world users. Virtual machines are often provided by cloud services and also being capable of simulating all different fingerprint features. We detect the generative tool as a VM if the rendering method in the fingerprint has a consistent VM-based renderer and vendor.

3) Generative Strategy Analysis: Once an adversary uses a certain tool, the next step is to choose a strategy in changing or generating features of the browser fingerprint collected in the request sent to a target website. In practice, we observe the following strategies and summarized them below:

- [Keep] Keeping the original fingerprint. The simple strategy from an adversary is to keep the original fingerprint (or certain feature values) of the tool. Consider content scraping for example. While this strategy of keeping the original fingerprint increases the chance of the adversary being blocked, the adversary can still obtain some information under the request threshold for clients with certain fingerprints. If we observe only one feature value for a certain browser instance, we consider that the adversary keeps the value.
- [Block] Blocking certain features in a fingerprint. Another relatively simple strategy is to block the value of a feature. Take canvas rendering for example. An adversary can disable the canvas API in a browser to prevent the target website from getting a valid value. Such a strategy is somewhat effective because benign fingerprints also have missing values especially for some features requiring heavy computations like canvas rendering and font list. If we observe that the feature value is missing in a malicious request, we consider that the adversary blocks the value.

TABLE III. BREAKDOWN OF BENIGN VS. ADVERSARIAL REQUESTS BASED ON ATTACK TYPES

Website	Benign	Adversarial					
		Account takeover attempts	Fraud	Fake account	Scraping		Giftcard cracking
					Anonymous	Logged-in	
All	57.5%	7.0%	0.8%	0.1%	15.0%	19.6%	<0.1%
Rest. A	55.1%	34.5%	3.9%	3.4%	-	<0.1%	3.1%
Bank A	99.8%	0.2%	-	-	-	-	-
Bank B	46.4%	52.9%	<0.1%	0.7%	<0.1%	-	-
Bank C	86.8%	8.7%	4.0%	0.1%	<0.1%	0.4%	-
Finance A	75.5%	24.5%	-	-	-	-	-
Finance B	98.7%	0.1%	-	-	0.5%	0.6%	-
Finance C	77.8%	22.2%	-	<0.1%	-	-	-
Shop A	91.4%	1.2%	7.3%	0.1%	-	<0.1%	<0.1%
Shop B	48.4%	0.2%	1.0%	-	22.5%	27.8%	<0.1%
Airline A	79.4%	0.1%	-	<0.1%	2.6%	17.9%	<0.1%
Airline B	80.7%	9.4%	-	-	3.9%	6.1%	-
ISP A	99.8%	0.2%	<0.1%	-	-	-	-
ISP B	99.5%	0.5%	-	-	-	-	-
ISP C	86.8%	13.1%	-	<0.1%	-	0.1%	-

- [Mimick] Mimicking benign fingerprints. One complex strategy from the adversary is to mimick a different benign fingerprint in each request to the target website. This is relatively hard for an adversary because it needs a large set of benign fingerprints to iterate through for each HTTP(s) request. If we observe that there are multiple feature values for one browser instance and all the values also appear in the benign fingerprints, we consider that the adversary mimicks the value from benign fingerprints.
- [Randomize] Randomizing certain feature values in a fingerprint. Because mimicking is a relative hard strategy, some adversaries also adopt another strategy, which randomizes certain feature values. Take the plugin feature for example. An adversary modifies the plugin API to return a random list of plugins each time. Note that such generated adversarial fingerprints will be unique in most cases and not shared with other benign fingerprints. If we observe that there are multiple feature values for one browser instance and none of the values appear in the benign fingerprints, we consider that the adversary randomizes the value.

Note that there are some gray areas that we cannot determine the adversarial strategies and we just mark them as "Gray". For example, if an adversarial value appears in the benign fingerprints but the frequency is very small (e.g., <0.01% of the feature's values), we cannot decide whether benign users also randomize the value (e.g., via a privacy-preserving browser) or the adversary mimicks a benign user. Then, we mark such adversarial strategies as gray.

IV. DATASET

In this section, we describe the collected dataset. In total, we collected 36 billion HTTP(s) requests: 15.3 billion (42.5% of the total) adversarial and 20.7 billion (57.5% of the total) benign. We also break down the requests based on 14 websites in Figure 4. The results show that Bank B and Airline A takes more than 50% of the total number because of their popularity, and the rest websites take the other 50%.

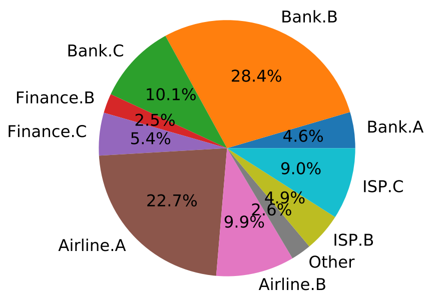

Fig. 4. The Request Percentage Breakdown by Websites (Some websites are grouped.)

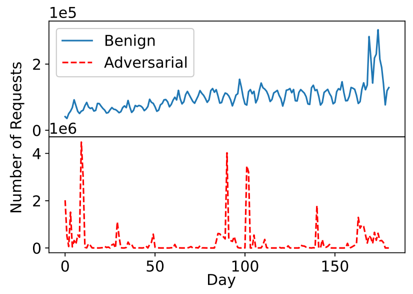

Fig. 5. Number of Benign and Adversarial Requests of Airline A over Six Month

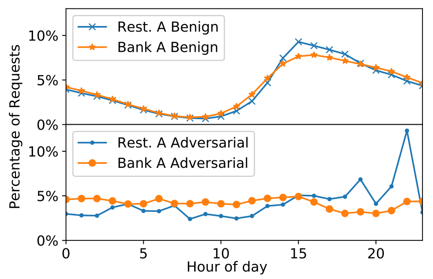

Fig. 6. Number of Benign and Adversarial Requests of Restaurant A and Bank A over One day

TABLE IV. A List of Collected Information (Meta-information + Browser Fingerprint) in Our Measurement Study. Note that all the number are unique values collected from our study.

Name	Description	Example	# Benign	# Adversarial	# Combined
HTTP(s) Request	Requests to target websites	-	20,762,474,611	15,346,176,886	36,108,651,497
Meta-information	-	-	-	-	-
Traffic type	Benign or malicious (account takeover attempts, content scraping, fake account login, giftcard cracking, and fraud transactions)	Label	1	5	6
Account ID	Anonymized account identifier	Hash Value	190,493,755	120,924,767	264,327,700
Destination root(apex) domain	The destination root domain name	-	14	14	14
Destination sub-domain	The destination sub-domain name	-	104	98	104
Source IP	Source Internet Protocol (IP) address	65.78.121.109	793,682,659	144,209,829	804,622,232
Source ASN	Source Autonomous System Number (ASN)	3,356	374,625	204,525	390,238
Browser Fingerprint	A combination of all features	-	1,537,702,813	162,303,779	1,673,234,835
User-Agent	"User-Agent" HTTP header	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/573.36	47,744,084	11,845,304	54,614,360
Historical timestamp	The String representation of a historical date and time	Mon Aug 06 1945 08:16:00 GMT+0900 (Palau Time)	6,276	1,536	6,336
Plugins	The total number of plugins and the hash value of all plugins	10, Hash value	5,238,288	1,374,595	6,595,013
Font list	Anonymized font list	Hash Value	893,115	229,776	1,065,906
Canvas image	The hash value of rendering results of a given image	Hash value	21,626,771	5,715,765	26,338,550
GPU vendor and renderer	GPU vendor and renderer	NVIDIA corporation ANGLE (NVIDA, NVIDA GeForce GTX 1080)	80,903	8,737	82,074
Screen resolution	Screen size, color depth, and available Height/Left/Top/Width.	1440×900×24, availHeight: 823, availLeft: 0, availTop: 25, availWidth: 1440	575,070	54,756	585,857
devicePixelRatio	The ratio of the resolution in physical pixels to the resolution in CSS pixels	2.1	45,118	2,921	45,168

A. Attack Statistics

Table III shows the breakdown of benign vs. adversarial requests based on different websites and attack types. Note that our final attack label of the dataset comes from the company's final database combining multiple labelling sources. That is, the ground truth labels are verified by both the real-time and offline model and the double-feedback-loop system with possible manual verification.

The dominant attack types vary a lot for different websites. For example, content scraping is the most prevalent on Shop B, Airline A, and Airline B, while account takeover attempts are the most prevalent on Restaurant A, Bank B, and Finance A & C. The reason is that shops and airlines usually have abundant product prices and competitors are eager to obtain such information. By contrast, banks and restaurants usually have sensitive customer information so that adversaries want to compromise them for benefits.

1) Attack Pattern Over Time: We illustrate three examples to show the adversarial and benign requests over time. Figure 5 shows the number of benign and adversarial requests of Airline A over the total six-month period. One notable thing is that benign requests are almost uniformly distributed with exceptions in the month of May and June where people are traveling more than winter time and COVID restrictions are partially lifted. By contrast, adversarial requests are concentrated on certain dates where adversaries scrape contents, wait for some days, and do that over again. At the same time, we also show the number of requests of Restaurant A and Bank A during one-day period in Figure 6. This shows a different pattern from the six-month: Adversarial requests are more flat than the benign ones. The reason is that users are less likely to visit a website during early morning while adversaries are usually driven by bots, which do not have such patterns.

B. *Browser Fingerprint Collection and Statistics*

Table IV shows different types of browser fingerprint features and their statistics. We skip the details and only list some descriptions and illustrative examples for each feature in Table IV, because they are all documented by prior works [46], [50]. Note that we do not choose some features, e.g., the support of certain browser functions like cookies, because the unique values of such features are relatively small, which make them less practical in a billion-scale study. We also show the number of unique values of fingerprints and each feature that are broken down by benign, adversarial and combined requests. The total number of unique benign fingerprints is more than nine times larger than that of unique adversarial fingerprints. That is, although many adversaries are mutating fingerprints randomly, they did not mutate fingerprints enough to simulate the behaviors of benign fingerprints. In other words, adversaries still have much room to improve for mimicking the benign fingerprints on the feature level.

There are three things worth noting. First, we compare our study results with prior million-scale studies [36], [50], [53]. The unique values for each feature is proportionally larger than those observed in previous study, showing the potential of using such features in real-world top-ranked websites. Second, the total number of historical timestamp is larger than the number of time zones, because the same time zone may be represented by different names such as "New York Time" and "Eastern Time". Lastly, the unique number of plugins and screen resolution is large. It is because they both contains many subfields, such as filename, version and suffixes for plugins and seven numbers for screen resolution.

Takeaway [Billion- vs. Million-scale]: At the billion-scale (i.e., those being faced by top-ranked commercial websites), unique values of each fingerprint feature are *proportionally* larger than the one observed in previous million-scale studies [36], [50], [53], showing their capabilities in real-world uses.

V. Observations

In this section, we describe our three main observations.

A. *Observation-1: Adversarial vs. Benign Fingerprints*

Our first observation is on the comparison of adversarial and benign fingerprints.

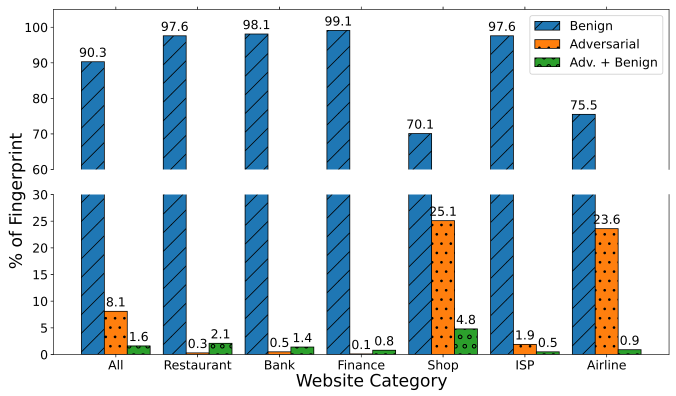

Fig. 7. Distribution of Adversarial and Benign Fingerprints by Different Website Categories

Takeaway [Adversarial vs. Benign Fingerprints]: Adversarial browser fingerprints generated by attackers are significantly *different* from benign ones:

- Only 1.6% of unique fingerprints are shared between adversarial and benign; by contrast, 8.1% are purely adversarial and 90.3% purely benign.
- The K-L divergence between adversarial and benign fingerprints is much larger than that between benign/benign and between adversarial/adversarial.
- Adversarial fingerprints have more empty values compared with benign.
- Benign fingerprints often evolve over time, while adversarial ones mostly stay stable.

1) Overview: We first give an overview of adversarial vs. benign fingerprints by breaking down the percentage of adversarial, benign and combined in Figure 7. Adversarial and benign fingerprints are clearly separated: 90.3% are only benign, 8.1% are only adversarial, and only 1.6% are shared between adversarial and benign. We also break down fingerprint distributions by different website categories in Figure 7.

There are three things worth noting about the breakdown by website categories. First, the percentage of adversarial fingerprints for "Shop", "Airline", and "ISP" websites is large and more than that of adversarial+benign. It is because content scraping is popular for these website as shown in Table III. That is, adversaries create many unique adversarial fingerprints to bypass the scraping threshold enforced by the website. Second, the percentage of adversarial+benign fingerprints for "Restaurant", "Bank", and "Finance" is larger than that of adversarial fingerprints. The reason is that account takeover attempts are generally popular for these websites, where adversaries need to either mimick existing benign fingerprints or use a real browser with benign fingerprints to take over user accounts and then maybe launch fraud transactions. Lastly, the percentage of benign + adversarial fingerprints is much smaller than that of benign fingerprints for all websites. There are two reasons. First, the absolute number of adversaries is much smaller than that of benign users. Therefore, the adversaries do not own a large number of benign fingerprints. Second, it is likely that adversaries do not have a large database of benign fingerprints to emulate.

2) K-L Divergence: Table V shows the K-L divergence of each feature and the combined fingerprint. We have three

TABLE V. Comparison between Adversarial and Benign Fingerprints on Different Metrics

Feature Name	Kullback-Leiber Divergence			Empty rate		Normalized Entropy		Unique rate	
	D(Adv.‖ Benign)	D(Benign‖Benign)	D(Adv.‖Adv.)	Benign	Adv.	Benign	Adv.	Benign	Adv.
User-Agent	1.6±1.2	0.8±1.8	1.9±1.6	<0.1%	<0.1%	3.8±0.5	3.0±0.6	89.6%	58.0%
Timestamp	4.2±5.6	0.5±0.6	2.0±1.5	2.1%	64.4%	1.8±0.3	1.5±0.7	76.5%	3.9%
Plugins	2.1±1.9	0.4±0.3	1.6±1.3	4.4%	46.3%	1.4±0.3	2.2±3.0	99.6%	98.7%
Font list	2.2±1.5	0.8±1.8	1.9±1.5	11.8%	50.1%	2.6±0.7	2.4±1.9	93.6%	75.2%
Canvas image	2.7±1.5	0.9±1.9	1.9±1.5	4.5%	46.6%	2.3±0.4	1.7±0.8	95.3%	82.4%
vendor + renderer	5.7±2.8	0.8±1.7	2.5±2.5	1.4%	86.2%	4.0±1.4	2.0±1.3	90.7%	13.4%
Screen resolution	5.3±3.0	0.4±0.3	2.2±1.5	0.1%	43.1%	4.6±1.2	2.5±1.3	92.3%	19.7%
devicePixelRatio	5.7±4.9	0.9±1.6	2.4±2.5	0.0%	82.4%	2.1±1.2	1.2±0.5	93.6%	1.7%
IP	1.7±0.9	0.4±1.7	1.6±2.3	0.0%	0.0%	1.4±0.3	1.3±0.5	83.2%	7.6%
ASN	3.6±1.5	2.0±1.9	3.0±1.4	0.0%	0.0%	3.7±0.6	3.4±0.5	49.6%	7.6%
Fingerprint	3.8±1.9	0.5±0.5	0.2±0.6	0.0%	<0.1%	10.7±1.6	7.0±2.8	98.3%	83.5%
Fingerprint + IP&ASN	2.6±2.0	0.1±0.2	0.01±0.1	0.0%	0.0%	13.6±1.6	9.1±2.1	97.8%	61.6%

observations. First, D(benign ‖ benign) is always smaller than 1. That is, benign fingerprints are similar to each other. Second, D(adversarial ‖ adversarial) of combined fingerprint (with or without IP and ASN) is small, indicating the similarity among adversarial fingerprints. Interestingly, D(adversarial ‖ adversarial) of each individual feature is always larger than 1.5 (and sometimes 2). This is because although adversaries drastically change one fingerprint feature, other features mostly stay untouched, leading to a small combined K-L divergence value but a large value for separate features. Third, D(adversarial ‖ benign) is generally larger than D(adversarial ‖ adversarial) except for the "User-Agent". That is, the difference between adversarial and benign fingerprints is lager than that between adversarial themselves on the feature and combined level.

3) Empty Rate: Table V also shows the empty rate of each fingerprint feature. The empty rate of adversarial fingerprints is larger than that of benign fingerprints for all the features. The reason is that when adversaries use scripting tools, JavaScript features will have empty values. It is worth noting that benign requests may also have empty values mainly because of two reasons. First, the user may close the tab before the fingerprint collection finishes. Our platform incrementally transfers collected fingerprint features, resulting in empty values that need heavyweight computation. Specifically, the empty rate of font list is the highest, because our platform needs to render many fonts to collect the complete list. Due to a similar reason, the empty rate of canvas image is the second highest. Second, some benign bots, such as Google and Facebook bots, may also have empty values in JavaScript-related features.

4) Normalized Entropy: Table V also shows the normalized entropy of adversarial and benign fingerprints broken down by each feature. The normalized entropy of benign fingerprints is generally larger than that of adversarial fingerprints for all the features except for plugins. This means that fingerprints are generally better to identify browser instances of benign users compared with adversaries. The reason, based on our observation, is that adversaries re-use scripts across different campaigns, leading to duplicates of fingerprints. By contrast, if a fingerprint is unique for a given benign user, it will be less likely shared by other benign users.

5) Unique Rate: The last column of Table V shows the unique rate of each fingerprint feature and the combined value. The unique rates of benign fingerprint features are all very high: This indicates that adversaries probably did not obtain a large database with benign feature values. By contrast, the

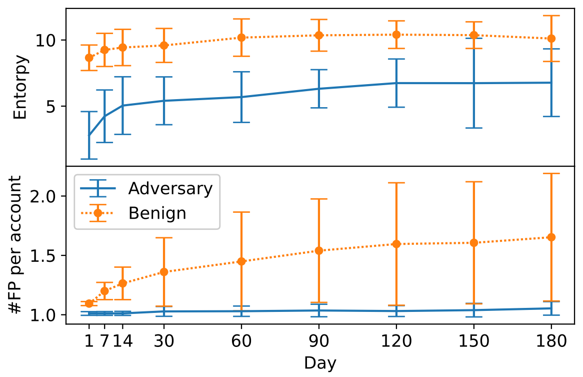

Fig. 8. Evolution of Median and Median Absolute Deviation (MAD) Value of Normalized Entropy and #FP per Account over Six months Period.

unique rates of only three features (i.e., font list, plugins, and canvas image) are high for adversarial fingerprints. The reason is that the space of these three features is large and therefore an adversary can easily mutate the feature and generate many random, unique values. Interestingly, the unique rate of the combined fingerprint drops after adding IP and ASN. The reason is that the number of non-unique fingerprint increases, but the number of unique fingerprint stays mostly the same. That means adversaries are changing IPs when re-using such non-unique fingerprints, while unique fingerprints are truly abandoned after one-time use.

6) Evolution: We use two metrics, i.e., number of fingerprints per account and normalized entropy, to measure the evolution of fingerprints and compare adversarial and benign fingerprints as shown in Figure 8. First, let us describe the median value of number of fingerprints per account, which increases constantly from close to one to larger than 1.5 over time. The reasons are twofold: (i) the browser fingerprint evolves, and (ii) users log in from multiple browser instances. By contrast, the number stays mostly the same at close to one for adversarial fingerprint. The reasons are also twofold: (i) adversaries often abandon fake accounts after one-time use, leading to only one fingerprint per account, and (ii) adversaries often bind fingerprints with compromised accounts, which also result in one fingerprint per account.

We also present the Median Absolute Deviation (MAD) value of adversarial and benign fingerprints in Figure 8. MAD values of benign fingerprints are much larger than those of adversarial and they keep increasing over time. This indicates the diversity of adversarial fingerprint evolution: Some benign

TABLE VI. Benign vs. Adversarial Bots

Bot Name	Type	#FPs	%Requests	Descriptions
Facebook	Benign	3	93.5%	Official bot on PC, Android and iOS.
	Adversarial	4	<0.1%	Different fingerprints from benign ones
		3	6.5%	The same fingerprints as benign one
Google	Benign	8	20.5%	Official bot with different user-agents
	Adversarial	16	5.9%	Different fingerprints from benign ones
		6	73.5%	The same fingerprints as benign one
AppleMail	Benign	5	59.2%	AppleMail bot with different user-agents
	Adversarial	3	40.8%	The same fingerprints as benign one
Amazon Silk	Benign	2	97.6%	Amazon Silk bot on Android and PC
	Adversarial	2	2.4%	The same fingerprints as benign one
Outlook	Adversarial	3	100%	Adversarial Window 10 Outlook bot

users log into multiple browsers and their fingerprints change a lot over time. By contrast, adversarial fingerprints are relatively consistent as shown in their small MAD values.

Second, we compare the median normalized entropy between adversarial and benign fingerprints over time. The entropy of benign fingerprints is relatively stable. That is, the fingerprintability of benign users stays mostly the same over time. By contrast, the entropy of benign fingerprints increases over time from around two to over five. The reason is that more adversaries participate in attacks over time and therefore the fingerprintability also increases over time. Another thing worth noting is that the benign fingerprint's entropy is also higher than that of adversarial, indicating that users are more fingerpintable than adversaries.

7) Case Study: Adversarial vs. Benign Bots: We give a case study on comparing adversarial and benign fingerprints from different bots. Table VI shows the bot names, whether they are adversarial, the number of unique fingerprints, and the percentage of requests in their own bot category. There are two things worth noting. First, although most adversaries mimick the fingerprints of benign bots correctly, there are still a small number that is different from the benign bots, making them easy to be differentiated. Second, the number of unique fingerprints mimicked by adversaries is smaller than the one used by a benign bot. On one hand, it means that certain fingerprints will indicate benign requests. On the other hand, it means adversaries still have rooms to improve their ability in mimicking benign fingerprints.

B. Observation-2: Adversarial Generative Strategies

Our second observation is that adversaries adopts various existing tools with different strategies to generate a large number of adversarial fingerprints with different properties.

Takeaway [Adversarial Generative Strategies]:
- Scripting tools are the most popular in practice due to its high performance.
- "Randomize" and "Block" are the most popular strategies for emulated browser tools.
- "Mimic" is less popular probably because adversaries need to obtain a large database of benign browser fingerprints.

TABLE VII. Adversarial Generative Strategy Distribution based on Adversarial Tools and Their Statistics

Tools	Adversarial Strategy	%Request	%FP	#Req per FP
Scripting	Keeping tools' fingerprints	3.1%	<0.1%	155,522.6
	Mimicking benign fingerprints Disabling JavaScript	77.6%	7.2%	657.5
Browsers	Mimic	1.0%	9.0%	7.1
	Mimic+Block	0.8%	3.5%	14.9
	Mimic+Block+Randomize	0.1%	0.2%	31.5
	Mimic+Randomize	0.2%	1.2%	8.0
	Keep	2.4%	0.4%	348.6
	Block	3.9%	<0.1%	5,151.5
	Block+Randomize	9.0%	60.3%	9.2
	Randomize	0.2%	0.1%	105.3
	Gray	1.2%	7.2%	10.3
VMs	Mimic	0.08%	3.18%	1.9
	Mimic+Block	0.02%	1.55%	1.0
	Mimic+Block+Randomize	0.07%	3.34%	1.35
	Mimic+Randomize	0.03%	1.23%	1.57
	Keep	0.03%	0.06%	32.2
	Block+Randomize	0.03%	<0.01%	18,112.0
	Gray	0.05%	1.33%	2.4

1) Breakdown by Adversarial Tools and Strategies: Table VII breaks down fingerprints and requests by adversarial tools and their corresponding adversarial strategies.

Percentage of Requests. Let us start from the observation on the percentage of requests. Scripting tools are the most popular, taking up over 80% of all the traffic and most adversaries try to mimic benign requests with a benign browser fingerprint (or specifically a benign user agent). The reasons are two-fold. First, script tools are performance efficient in generating a large amount of requests (i.e., 30–133 requests per min for scripting vs. 14–75 per min for emulated browsers). Second, adversaries want their request to look like benign and therefore tend to mimick benign browsers (or bots).

Percentage of Total Fingerprints. We describe the percentage of total unique fingerprints for different adversarial tools and strategies. "Block+Randomize" from emulated browsers is the most popular because feature randomization can quickly generate many different unique fingerprints. By contrast, the percentage of fingerprints for "Keep" in the scripting tools and "Block" in the emulated browsers is small ($< 1\%$) because the number of scripting tools is small and so is the number of possible fingerprints when features are blocked. Note that there are different variations of the "Mimick" strategy. Ideally, the pure "Mimick" is the most effective way because the generated fingerprint is exactly the same as a benign one without inconsistencies. This is also reflected in the % of FP, which is the largest compared with other "Mimick". At the same time, adversaries also combine other strategies with "Mimick" when they run out of benign feature values.

Number of Requests per Fingerprint. The number of requests per fingerprint is very large for the "Keep" strategy in scripting tools and also emulated browsers because adversaries are crafting many requests from unchanged adversarial tools. The number is also large for the "Block" strategy because the number of possible fingerprints is relatively small. Interestingly, the number is also large for the "Randomization", indicating that adversaries are reusing fingerprints when they generate a new, random one.

There are two things worth noting here. First, the order of the number of requests per fingerprints is the opposite of the percentage of fingerprints. For example, the “Block” strategy in emulated browsers has the largest number of requests per fingerprint, but the percentage of fingerprint is the smallest. The reason is that the percentage of request is similar, but the number of fingerprints is much smaller. Therefore, the number of requests per fingerprint is large. Second, the number of requests per fingerprint for “Block+Randomize” is large for VMs but small for emulated browsers. We investigate this and find that a specific attacker using VMs randomizes the canvas value only once but then use the randomized value with blocked feature values to send many adversarial requests. This leads to a very large number of request per fingerprint for VMs. We believe that this is an exception.

Emulated Browsers vs. VMs. We compare the adversarial strategies used in emulated browsers and VMs. The list and percentage of both tools are very similar. It is probably because the attack capabilities are also similar based on our analysis. The adversary using VMs can change any feature values just like those who use emulated browsers. Based on our speculation and manual traffic analysis, the reason that an adversary adopts VMs is that they use cloud services with VMs instead of their own machines. It is worth noting that the percentage of request with VMs is much smaller than that with emulated browsers, because clould IPs have a higher chance to be blocked or detected by a defense system. At the same time, the attack speed of VMs is very small (i.e., 2–15 per minuite).

2) Adversarial Strategy Breakdown by Fingerprint Features: We break down adversarial generative strategies used in emulated browsers and virtual machines by fingerprint features. (Note that scripting tools mostly just manipulate user-agents, i.e., their generative strategies are simple.) Specifically, we separately analyze each fingerprint feature and break down the percentage of requests and the number of values by the corresponding strategies. Note that “Gray” means that either it is challenging for us to decide the specific strategy or the adversary adopts a combination of several strategies together (e.g., part of user agent is blocked and other parts are kept).

Table VIII shows the results of the breakdown. First, the “Block” strategy is the most popular for all the features except for “User-Agent”. The reason is that “Block” is easy to adopt as the adversary does not need to have a database of benign fingerprints or randomize features of existing fingerprints. At the same time, “Block” is effective because benign fingerprints may also have empty values especially when the collection (e.g., for font list and canvas image) takes time. More importantly, some benign clients may also intentionally block certain features such as screen resolution for the privacy purpose. Note that adversaries rarely blocks user agents, because the empty rate of user agents (as shown in Table V) is almost zero for benign users and therefore many websites simply reject requests without a user agent value.

Second, “Randomize” is more popular on three features, namely “Font list”, “Canvas image”, and “Plugin”, than others. The reason is that the feature space for these three is large and can be easily mutated. For example, an adversary can add random noise to existing canvas images to generate a new image; similarly, an adversary can append values to plugin list to randomize its value. As a comparison, “Mimick” is more popular than “Randomize” on features with guessable or known values, such as “User Agent” and “Timestamp”. The reason is that adversary can easily come up with a benign feature value for them and mimick users’ behavior. As a comparison, adversaries need to collect user fingerprint values for some features like canvas image and font list.

Lastly, “Keep” as the default strategy still takes up a considerable amount of traffic. On one hand, even if adversaries do not change fingerprints, they can still collect information in content scraping attack just with limited rate. On the other hand, adversaries may combine “Keep” with other strategies such as “Randomize” and “Block” for generation.

3) A Case Study on Credential Stuffing Attack: In this section, we use credential stuffing as a case study to explain adversarial strategies for a specific attack. Generally speaking, there are two alternate moves for a credential stuffing attack: (i) probe and (ii) takeover attempt. The first move is to probe and determine the threshold for the number of requests associated with a browser instance with a certain fingerprint. Specifically, an adversary launches the probe from fake accounts with a fixed fingerprint. Then, she keeps trying to log into the account with wrong passwords until the browser instance with the fingerprint is blocked (i.e., one cannot even log into an account with the correct password). By doing so, she can obtain the request limit enforced by the website. Note that the adversary has to use a fake account for probing because a user without the real password cannot tell whether the browser instance is blocked or the password is wrong.

The second move is to attempt to take over real-world accounts with credential stuffing. The adversary starts from setuping target accounts and fingerprints. There are generally two strategies: binding and non-binding. On one hand, the most popular strategy is to generate a fingerprint and then bind the fingerprint with the target account in the follow-up attacks. Based on our observation, 98.6% of requests adopt such a binding strategy. On the other hand, some adversaries also do not bind fingerprints with accounts, which takes up the rest 1.4% of requests. After setup, an adversary sends requests to the target website with many account and username combinations with or without the fingerprint-account binding until the guessed limit from the first probing move.

Table IX shows the breakdown of probe and takeover attempts of all the credential stuffing attacks of our dataset. If we cannot decide whether some requests are probes or takeover attempts, we mark them as gray. Probes and takeover attempts are different from each other. First, the number of requests of probe account is much larger than that of victim account. The reason is that adversaries need to repeatedly probe the target website with the probe account and decide the limit. By contrast, adversaries usually do not waste time on one account if the takeover fails. Second, a similar argument applies to the number of requests per fingerprint. The number of requests per fingerprint of takeover attempts is also very close to one because of the account-fingerprint binding. By contrast, because an adversary tries many probes on the probe account with a fixed fingerprint, the number is high for probe attack. Lastly, we also compare the percentage of requests, accounts and fingerprints. Takeover attempts target more accounts with more fingerprints compared with probes because probes usually adopt a fixed number of accounts and fingerprints and reuse them over time to decide the limits.

TABLE VIII. ADVERSARIAL STRATEGY DISTRIBUTION OF EMULATED BROWSERS AND VIRTUAL MACHINES ON DIFFERENT FINGERPRINT FEATURES (%Req$_{\text{adv.}}$: PERCENTAGE OF ADVERSARIAL REQUESTS; %FINGERPRINT PERCENTAGE OF FINGERPRINT: PERCENTAGE OF UNIQUE FINGERPRINT VALUES)

Feature	Keep		Block		Mimick		Randomize		Gray	
	%Request$_{\text{adv.}}$	%Fingerprint	%Request$_{\text{adv.}}$	%Fingerprint	%Request$_{\text{adv.}}$	%Fingerprint	%Request$_{\text{adv.}}$	%Fingerprint	%Request$_{\text{adv.}}$	%Fingerprint
User-Agent	81.1%	1.9%	<0.1%	<0.1%	6.6%	16.9%	7.4%	76.5%	5.5%	4.7%
Timestamp	25.0%	2.1%	46.4%	<0.1%	0.6%	26.9%	27.8%	70.0%	<0.1%	<0.1%
Plugin	39.2%	40.0%	60.4%	<0.1%	<0.1%	0.1%	0.3%	58.9%	<0.1%	<0.1%
Font list	18.7%	0.3%	68.4%	<0.1%	6.0%	13.4%	6.8%	86.3%	<0.1%	<0.1%
Canvas image	28.9%	10.1%	53.6%	<0.1%	0.1%	9.4%	17.3%	80.5%	<0.1%	<0.1%
Vendor + renderer	40.6%	32.2%	54.4%	<0.1%	0.8%	27.3%	4.1%	40.4%	<0.1%	<0.1%
Screen resolution	26.1%	0.9%	41.3%	<0.1%	6.4%	26.9%	19.9%	67.8%	6.1%	5.3%
devicePixelRatio	20.1%	6.1%	70.9%	<0.1%	2.3%	25.5%	2.8%	63.1%	3.7%	6.3%

TABLE IX. CREDENTIAL STUFFING BREAKDOWN

	%Request	%Account	%FP	#Requests per account	#Requests per FP
Probe	2.0%	>0.1%	1.1%	112.8	46.1
Takeover Attempt	92.9%	99.8%	98.4%	1.3	1.0
Gray	5.0%	0.2%	0.5%	32.9	2.3

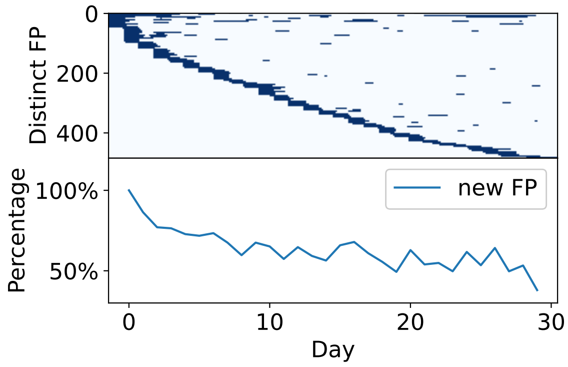

Fig. 9. Daily use pattern of fingerprints (top, marked as blue) and the percentage of new fingerprints (bottom) during a month period for a specific fraud adversary.

Takeaway [Credential Stuffing Case Study]: Our case study shows that credential stuffing adversaries alternate between probes and account takeover attempts.

- A small number of probes with fixed fingerprints are used to determine server-side defense threshold.
- Adversaries usually bind fingerprints with accounts and attack accounts.

4) A Case Study on Fraud Transactions: We present a case study of a specific fraud adversary and their used fingerprints over a month period (when the adversary is active). Figure 9 shows the use pattern of distinct fingerprints on each day (marked as blue, top) and the percentage of new fingerprints (bottom). We observe that the adversary generates about 50% of new fingerprints each day but also reuses another 50% of the old fingerprints over time. The adversary also reuses different old fingerprints over time so that they do not have a big overlaps with the previous day. We believe that such a strategy helps the adversary maximize generated fingerprints and avoid being blocked.

Takeaway [Fraud Transaction Case Study]: A fraud adversary both creates new fingerprints and reuses old ones to maximize the utilization of fingerprints.

TABLE X. BREAKDOWN OF ADVERSARIAL FINGERPRINTS AND STATISTICS BY ATTACK TYPE

Attack Type	Attack tools percentage			D(Adv.‖Benign)	Entropy	Unique rate
	Scripting	Browsers	VM			
Account takeover	84.2%	15.5%	0.3%	2.3±2.3	7.8±2.7	82.7%
Fake account	49.2%	50.6%	0.1%	1.8±1.6	6.5±1.4	61.9%
Fraud	30.0%	69.6%	0.4%	2.2±2.1	7.1±0.9	98.8%
Scraping	93.4%	6.6%	<0.1%	4.1±4.6	7.0±1.4	49.3%
Giftcard Cracking	97.5%	2.2%	0.3%	0.03±0.1	5.8±1.3	96.3%

C. Observation-3: Adversarial Fingerprints over Attack and Account Types

Our third observation is on breaking down adversarial fingerprints by attack and account types.

1) Breakdown by Attack Type: We break down adversarial fingerprints based on the attack types defined in Section IV-A and show the statistics. Specifically, we include four metrics: the K-L divergence between adversarial and benign fingerprints D(Adv.‖Benign), the normalized entropy, the unique rate (i.e., the percentage of fingerprints that are either benign or adversarial), and the breakdown of attack tool percentage (scripting, emulated browser and virtual machine). Table X shows the numbers and we have four observations.

First, emulated browsers are used more often for fake account signups and fraud transactions than other attacks (e.g., giftcard cracking and content scraping). The reason is as follows. Fraud transaction and fake account signups focus on the attack accuracy rather than the total number of requests. For example, a successful fraud transaction is more valuable than many failed ones. Therefore, the adversary wants to mimic a real user to the best that they can and an emulated browser is a better tool compared with scripting tools (which has limited capabilities). As a comparison, some attacks focus more on the performance, e.g., content scraping on the amount of scraped contents. Scripting tools are more suitable for such attacks to obtain a large amount of time-sensitive contents.

Second, the K-L divergence between adversarial and benign fingerprints of content scraping is much larger than that of other attacks. The reason is that adversaries mostly adopt feature randomization to scrape contents, which leads to a big gap between adversarial and benign fingerprints. By contrast, other attacks, such as fraud transactions, rely more on mimicking user browsers, which lead to a relatively small gap. Giftcard cracking is an extreme case, where the K-L divergence score is only 0.03. That is, the adversarial and benign fingerprints of giftcard cracking are very similar.

TABLE XI. Breakdown of # of Accounts and # of Fingerprint per Account by Account Types

Website	User Account		Adversarial Account			
			Bot/Fake		Compromised	
	#Account	#Fingerprint per account	#Account	#Fingerprint per account	#Account	#Fingerprint per account
Rest. A	13,455,375	1.3	265,496	1.0	79,301	3.8
Bank A	-	-	-	-	-	-
Bank B	21,059,672	1.3	32,184,479	1.0	43,257,818	7.2
Bank C	20,557,044	2.2	472,179	1.2	1,464,737	6.6
Finance A	350,378	2.2	3,140	1.1	13,224	6.9
Finance B	1,195,366	1.8	25,128	1.0	10,954	5.6
Finance C	11,474,213	2.6	2,287,748	1.0	787,930	14.9
Shop A	3,023,655	1.7	8,746	1.0	47,720	5.7
Shop B	4,574,309	1.4	26,221	3.3	70,175	3.8
Airline A	17,774,715	1.7	114,986	1.0	279,489	6.4
Airline B	4,385,500	1.7	38,293,398	1.0	756,532	2.3
ISP A	12,550	1.0	17	1.0	46	2.1
ISP B	45,540,156	2.2	152,378	1.1	322,925	6.5
ISP C	-	-	-	-	-	-

Third, we look at the normalized entropy, which is relatively large for account takeover attempts, fraud, and content scraping. The reason might be that adversaries bind fingerprints to browser instances without changing them when launching these attacks multiple times, e.g., multiple attempts to take over accounts and sign up fake accounts.

Lastly, we describe the unique rate, which means the percentage of fingerprints that are only adversarial. Interestingly, the unique rate of gift cracking attack is also very high, meaning that adversaries generate many random fingerprints. Since the K-L divergence score is very low, it means that such random, unique fingerprints only take up a small amount of adversarial requests. In other words, the majority of requests from gift card cracking contain fingerprints that are mimicked from benign users.

Takeaway [Attack Type]:
- Fake account signups and fraud transactions tend to use emulated browsers more often than other attacks like account takeover attempts and giftcard cracking.
- The fingerprints used by content scraping are more different from benign ones compared with other attacks due to the reliance on scripting tools.

2) Breakdown by Account Type: We break down fingerprints and accounts based on account types and show the statistics in Table XI. Note that we do not have account data for Bank A and ISP C and that is why all the numbers are empty for these two websites. There are three things worth noting here. First, the number of fingerprints per user account is generally larger than that per bot or fake account. The reason is that it is usually very cheap or easy for adversaries to create new bot or fake accounts. Therefore, they tend to create a new fake account for attacks rather than reusing existing fake accounts, which risks being blocked or detected. The only exception is Shop B and we checked the data manually. One specific adversary targeting Shop B does reuse fake accounts extensively, which leads to a large number of fingerprints per fake account.

Second, the number of fingerprints per compromised account is much larger than that per user or bot/fake account. It is probably because compromised accounts—originally belonging to a real user—are valuable asset to adversaries. On one hand, such accounts may contain values (like positive balance in a bank's account), which can be obtained by an adversary. On the other hand, these accounts will be less probably blocked by a website, because they originally belong to a user. That is, an adversary may log into a compromised account multiple times with potentially different fingerprints. Note that this is different from previously-mentioned credential stuffing where most accounts have not been compromised and adversaries bind accounts with fingerprints for attacks. Here, the accounts are compromised and adversaries are launching fraud transactions. Moreover, we also observe that compromised accounts are being logged in by different adversaries, which may indicate some underground transactions among adversaries. These different adversaries also have different browser fingerprints, which further increase the number of fingerprints per compromised account.

Lastly, we observe that sometimes adversaries can create many bot/fake accounts for content scraping. Take Airline B for example. The number of bot/fake account is almost ten times lager than that of user account. This brings challenges to websites in maintaining accounts (because the majority accounts are in fact bots).

Takeaway [Account Type]:
- # of fingerprints per user account is generally larger than that per fake account, because adversaries abandon fake account after one-time use.
- # of fingerprints per compromised account is significantly larger than that per user or fake account due to fake account reuse and selling. for profits.

VI. Discussion

Ethics. Our data collection is performed at F5, Inc., which has contracts with our target top-ranked commercial websites. We would like to point out that it is always a trade-off between security and privacy during such a massive data collection effort. On one hand, such data collection is useful to defend against bot and fraud using the company's security product. On the other hand, the collected data inevitable contains potential private information. We did our best to maintain the security product's performance while minimizing the privacy risks.

Specifically, the company has a legal and ethics review department to ensure that all personal data is securely processed and stored, which obeys privacy laws such as General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). The department also discusses and negotiates with the company's customers (i.e., the 14 websites) to ensure their private data is securely maintained. During data collection, all personal information—such as user name, canvas rendering and even the list of fonts and plugins—is processed via secure one-way hashing. The paper in its current shape has also gone through several rounds of legal and ethics review to ensure that no personal data is involved before being peer reviewed or made public. In addition, all website names are anonymized as part of the agreement of the measurement study to avoid potential leaks of defense strategy used at the server-side.

Attacks other than bot and fraud. Our measurement is to study attacks relying on adversarial fingerprints. Many traditional attacks, such as SQL injection and cross-site scripting, are not relying on adversarial fingerprints. That is, adversaries log into their own account with a benign fingerprint and send a request with malicious payload. Such traffic is categorized as with a benign fingerprint (although the traffic is malicious).

Benign fingerprint randomization. Many browsers and extensions also fake or randomize fingerprints for the purpose of protecting users' privacy. Such fingerprints are categorized as benign instead of adversarial because the traffic analysis will consider the request as benign. Note that the percentage of traffic from such browsers is small. For example, Brave, a popular privacy-preserving browser that can randomize fingerprints, only contributes to $<0.00001\%$ of total traffic on these 14 websites. Therefore, they do not affect our conclusions (e.g., Observation 1 on the difference between adversarial and benign fingerprints).

Accuracy of Bot and Fraud Detection System. We would like to admit that it is challenging to estimate the true accuracy of the company's bot and fraud detection system due to the lack of ground truth data. While this is indeed one limitation of the paper, this is also the best that we can perform even given state-of-the-art approaches in bot and fraud detection. We also want to mention that an external, independent review shows that the detection system reduces costs caused by account takeover attacks by 96% and the total number of fake accounts by 92%.

VII. Related Work

Our work is broadly related to Web Security [12], [22]–[27], [41], [51], [52], [63], or particularly browser fingerprints [46]. We discuss related work below.

Browser Fingerprint Features. Over the past ten years, prior works have proposed ways to improve browser fingerprinting with different features when being used for Web tracking [46], [49]. Eckersley [30] is probably the first to propose and design browser fingerprinting with initial features such as fonts obtained via Flash and user-agent. After Eckersley, people have proposed many methods [3], [4], [6], [14], [18], [21], [33], [50], [58], [59], [69]–[71], [80] to improve browser fingerprint. At the same time, Our study's contribution is to study how attackers utilize features proposed by prior works to generate adversarial fingerprints.

Browser Fingerprint Application. Browser fingerprints can be used for different purposes. First, we describe multi-factor authentication. Spooren et al. [72] show the variability and the predictability of the use of device fingerprinting for risk-based authentication because mobile devices the fingerprints carry a lot of similarity. Laperdrix et al. [44] present the first fingerprinting-based focusing on the canvas-based authentication scheme to replay attacks. Andriamilanto et al [8] collect browser fingerprints throughout 6 months from a population of general browsers as an authentication factor. FPSelect [9] and BrFAST [7] are follow-ups on web authentication. Second, We describe bot and fraud detection. Many prior works [10], [17], [39], [60], [65], [81] analyze bot or fraud traffic using measurement study and their defenses. In particular, Li et al. [53] build so-called "honey sites" to attract bot traffic and analyze browser fingerprints and network behaviors. As a comparison, our measurement study is the first to analyze the difference between adversarial and benign fingerprints.

Browser Fingerprint Detection and Measurement. Prior works have proposed to detect and measure browser fingerprints in the wild. Many prior works [5], [19], [31], [43], [57], [62], [75] conducted real-world measurement study on browser fingerprints using either desktop or mobile traffic. Jueckstock and Kapravelos [40] design VisibleV8, a dynamic analysis framework, to support web security and privacy research. Iqbal et al. [38] present FP-Inspector, a learning based approach to accurately detect browser fingerprinting. Vastel et al. [79] detects browser fingerprint inconsistencies in a small manually-curated dataset. Datta et al. [29] and Merzdovnik et al. [56] evaluate and measure anti-fingerprint techniques. As a comparison, our measurement is the first to perform a billion-scale study on adversarial browser fingerprints.

Extension Fingerprinting. Extension fingerprinting is a technique (different from browser fingerprinting) to detect the existence of certain browser extensions. For example, Starov and Nikiforakis [74] shows the fingerprintability of browser extensions. Trickel et al. [78] present a client-side anti-fingerprinting countermeasure, called CloakX, using client-side diversification to defend against extension detection. Starov et al. [73] design an in-browser mechanism to protect users against extension fingerprinting. Karami et al. [42] present the automated creation and detection of behavior-based extension fingerprints and introduce two novel fingerprinting techniques that monitor extensions' communication patterns. Our measurement work focus on browser but not extension fingerprinting.

Fingerprint Manipulation Tools. Prior works have designed many tools to change browser fingerprints, which can be used to, for example, protect user privacy. For example, Fp-block [77], Fp-Guard [83], Blink [47], Fprandom [45], Deterministic Browser [20], and UniGL [84] propose different ways to modify browser fingerprints and protect user's privacy. Nikiforakis et al. [61] randomly change part of browser fingerprints for enhanced privacy. Baumann et al. [13] add noise to browser canvas rendering and disguise browsers. Fiore et al. [34] provide a coherent design to mimic browser fingerprint and intermingle fingerprints within others. Liu et al. [54] present Gummy Browsers that collect and spoof the browser fingerprinting information without the victim's awareness.

VIII. Conclusion and Future Work

In this paper, we perform the first billion-scale study of adversarial and benign browser fingerprints on 14 top-ranked commercial websites. The study and measurement data spans over half a year from January 2021 to June 2021 and contains over 36 billion HTTP(s) requests and more than one billion unique fingerprints. We classify requests into benign and five different attacks (account takeover attempt, fraud transactions, automated fake account signup, aggressive content scraping, and giftcard cracking) based on a major security product from F5, Inc.

Our key takeaway of the measurement study is that adversarial fingerprints are significantly different from benign ones in terms of metrics like entropy and unique rate. This observation will have a great impact on existing study on browser fingerprints because many of them do not differentiate adversarial fingerprints from benign. Take personalized advertisement as an example, which only need to consider benign fingerprints but not adversarial. Fingerprint entropy will change if adversarial ones are removed from existing studies, which affects personalization accuracy. Similar arguments can be applied to fingerprints as part of authentication. Therefore, we suggest that future researches on browser fingerprint should clearly differentiate adversarial and benign fingerprints when studying fingerprint properties especially when the use cases are related to benign users.

In the future, we envision that our measurement study can be used for the following possible directions. First, our study may shed light on bot and fraud detection using adversarial fingerprints. Such a system may adopt many features studied in the paper such as fingerprint inconsistencies and evolution. Second, our study especially the understanding of adversarial fingerprints may help existing privacy-preserving browser to better design fingerprints to bypass web tracking. Third, our study may lead to further studies on existing fingerprint application scenarios such as two-factor authentication and ads personalization.

Acknowledgement

We also thank anonymous reviewers for their helpful comments and feedback. This work was supported in part by National Science Foundation (NSF) under grants CNS-18-54001, CNS-20-46361, and CNS-21-54404 and Defense Advanced Research Projects Agency (DARPA) Young Faculty Award (YFA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF or DARPA.

References

[1] Prevent gift card cracking: Brute force enumeration. https://www.f5.com/solutions/gift-card-cracking.

[2] TCP/IP stack fingerprinting. https://en.wikipedia.org/wiki/TCP/IP_stack_fingerprinting.

[3] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 674–689.

[4] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1129–1140.

[5] N. M. Al-Fannah, W. Li, and C. J. Mitchell, "Beyond cookie monster amnesia: Real world persistent online tracking," in *International Conference on Information Security*. Springer, 2018, pp. 481–501.

[6] F. Alaca and P. C. Van Oorschot, "Device fingerprinting for augmenting web authentication: classification and analysis of methods," in *Proceedings of the 32nd annual conference on computer security applications*, 2016, pp. 289–301.

[7] N. Andriamilanto and T. Allard, "Brfast: a tool to select browser fingerprinting attributes for web authentication according to a usability-security trade-off," in *Companion Proceedings of the Web Conference 2021*, 2021, pp. 701–704.

[8] N. Andriamilanto, T. Allard, and G. L. Guelvouit, "“guess who?” large-scale data-centric study of the adequacy of browser fingerprints for web authentication," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, 2020, pp. 161–172.

[9] N. Andriamilanto, T. Allard, and G. Le Guelvouit, "Fpselect: Low-cost browser fingerprints for mitigating dictionary attacks against web authentication mechanisms," in *Annual Computer Security Applications Conference*, 2020, pp. 627–642.

[10] B. A. Azad, O. Starov, P. Laperdrix, and N. Nikiforakis, "Web runner 2049: Evaluating third-party anti-bot services," in *17th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2020)*, 2020.

[11] M. H. N. Ba, J. Bennett, M. Gallagher, and S. Bhunia, "A case study of credential stuffing attack: Canva data breach," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2021, pp. 735–740.

[12] Z. Bai, K. Wang, H. Zhu, Y. Cao, and X. Jin, "Runtime recovery of web applications under zero-day redos attacks," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1575–1588.

[13] P. Baumann, S. Katzenbeisser, M. Stopczynski, and E. Tews, "Disguised chromium browser: Robust browser, flash and canvas fingerprinting protection," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 2016, pp. 37–46.

[14] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Nordic conference on secure it systems*. Springer, 2011, pp. 31–46.

[15] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov, "Integro: leveraging victim prediction for robust fake account detection in osns." in *NDSS*, vol. 15, 2015, pp. 8–11.

[16] Y. Boshmaf, D. Logothetis, G. Siganos, J. Lería, J. Lorenzo, M. Ripeanu, K. Beznosov, and H. Halawa, "Íntegro: Leveraging victim prediction for robust fake account detection in large scale osns," *Computers & Security*, vol. 61, pp. 142–168, 2016.

[17] E. Bursztein, B. Benko, D. Margolis, T. Pietraszek, A. Archer, A. Aquino, A. Pitsillidis, and S. Savage, "Handcrafted fraud and extortion: Manual account hijacking in the wild," in *Proceedings of the 2014 conference on internet measurement conference*, 2014, pp. 347–358.

[18] E. Bursztein, A. Malyshev, T. Pietraszek, and K. Thomas, "Picasso: Lightweight device class fingerprinting for web clients," in *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2016, pp. 93–102.

[19] M. Campobasso and L. Allodi, "Impersonation-as-a-service: Characterizing the emerging criminal infrastructure for user impersonation at scale," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1665–1680.

[20] Y. Cao, Z. Chen, S. Li, and S. Wu, "Deterministic browser," ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 163–178. [Online]. Available: https://doi.org/10.1145/3133956.3133996

[21] Y. Cao, S. Li, E. Wijmans *et al.*, "(cross-) browser fingerprinting via os and hardware level features." in *NDSS*, 2017.

[22] Y. Cao, Z. Li, V. Rastogi, and Y. Chen, "Virtual browser: A web-level sandbox to secure third-party javascript without sacrificing functionality," ser. CCS '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 654–656. [Online]. Available: https://doi.org/10.1145/1866307.1866387

[23] Y. Cao, X. Pan, Y. Chen, and J. Zhuge, "Jshield: Towards real-time and vulnerability-based detection of polluted drive-by download attacks," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ser. ACSAC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 466–475. [Online]. Available: https://doi.org/10.1145/2664243.2664256

[24] Y. Cao, X. Pan, Y. Chen, J. Zhuge, X. Qian, and J. Fu, "Malicious code detection technologies," Dec. 15 2015, US Patent 9,213,839.

[25] Y. Cao, V. Rastogi, Z. Li, Y. Chen, and A. Moshchuk, "Redefining web browser principals with a configurable origin policy," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013, pp. 1–12.

[26] Y. Cao, Y. Shoshitaishvili, K. Borgolte, C. Kruegel, G. Vigna, and Y. Chen, "Protecting web-based single sign-on protocols against relying party impersonation attacks through a dedicated bi-directional authenticated secure channel," in *Research in Attacks, Intrusions and*

Defenses, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham: Springer International Publishing, 2014, pp. 276–298.

[27] Z. Chen and Y. Cao, "Jskernel: Fortifying javascript against web concurrency attacks via a kernel-like structure," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020, pp. 64–75.

[28] E. Chiapponi, M. Dacier, O. Thonnard, M. Fangar, M. Mattsson, and V. Rigal, "An industrial perspective on web scraping characteristics and open issues," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2022, pp. 5–8.

[29] A. Datta, J. Lu, and M. C. Tschantz, "Evaluating anti-fingerprinting privacy enhancing technologies," in *The World Wide Web Conference*, 2019, pp. 351–362.

[30] P. Eckersley, "How unique is your web browser?" in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2010, pp. 1–18.

[31] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1388–1401.

[32] B. Erşahin, Ö. Aktaş, D. Kılınç, and C. Akyol, "Twitter fake account detection," in *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2017, pp. 388–392.

[33] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 107–124.

[34] U. Fiore, A. Castiglione, A. De Santis, and F. Palmieri, "Countering browser fingerprinting techniques: Constructing a fake profile with google chrome," in *2014 17th International Conference on Network-Based Information Systems*. IEEE, 2014, pp. 355–360.

[35] S. Frolov and E. Wustrow, "The use of tls in censorship circumvention." in *NDSS*, 2019.

[36] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 309–318.

[37] A. Haque and S. Singh, "Anti-scraping application development," in *2015 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2015, pp. 869–874.

[38] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1143–1161.

[39] H. Jonker, B. Krumnow, and G. Vlot, "Fingerprint surface-based detection of web bot detectors," in *European Symposium on Research in Computer Security*. Springer, 2019, pp. 586–605.

[40] J. Jueckstock and A. Kapravelos, "Visiblev8: In-browser monitoring of javascript in the wild," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 393–405.

[41] Z. Kang, S. Li, and Y. Cao, "Probe the proto: Measuring client-side prototype pollution vulnerabilities of one million real-world websites," in *NDSS*, 2022.

[42] S. Karami, P. Ilia, K. Solomos, and J. Polakis, "Carnus: Exploring the privacy threats of browser extension fingerprinting." in *In Proceedings of the 27th Network and Distributed System Security Symposium (NDSS)*, 2020.

[43] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting mobile devices using personalized configurations," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.

[44] P. Laperdrix, G. Avoine, B. Baudry, and N. Nikiforakis, "Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2019, pp. 43–66.

[45] P. Laperdrix, B. Baudry, and V. Mishra, "Fprandom: Randomizing core browser objects to break advanced device fingerprinting techniques," in *International Symposium on Engineering Secure Software and Systems*. Springer, 2017, pp. 97–114.

[46] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," *ACM Transactions on the Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.

[47] P. Laperdrix, W. Rudametkin, and B. Baudry, "Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, 2015, pp. 98–108.

[48] ——, "Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 878–894.

[49] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, "Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[50] S. Li and Y. Cao, "Who touched my browser fingerprint? a large-scale measurement study and classification of fingerprint dynamics," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 370–385.

[51] S. Li, M. Kang, J. Hou, and Y. Cao, "Detecting node.js prototype pollution vulnerabilities via object lookup analysis," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 268–279. [Online]. Available: https://doi.org/10.1145/3468264.3468542

[52] ——, "Mining node.js vulnerabilities via object dependence graph and query," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 143–160. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/li-song

[53] X. Li, B. A. Azad, A. Rahmati, and N. Nikiforakis, "Good bot, bad bot: Characterizing automated browsing activity," in *2021 IEEE symposium on security and privacy (sp)*, 2021, p. 17.

[54] Z. Liu, P. Shrestha, and N. Saxena, "Gummy browsers: Targeted browser spoofing against state-of-the-art fingerprinting techniques," *ACNS: Applied Cryptography and Network Security*, 2022.

[55] M. Lu, Z. Han, Z. Zhang, Y. Zhao, and Y. Shan, "Graph neural networks in real-time fraud detection with lambda architecture," *arXiv preprint arXiv:2110.04559*, 2021.

[56] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, "Block me if you can: A large-scale study of tracker-blocking tools," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 319–333.

[57] A. Mirian, J. DeBlasio, S. Savage, G. M. Voelker, and K. Thomas, "Hack for hire: Exploring the emerging market for account hijacking," in *The World Wide Web Conference*, 2019, pp. 1279–1289.

[58] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in javascript implementations," *Proceedings of W2SP*, vol. 2, no. 11, 2011.

[59] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," *Proceedings of W2SP*, vol. 2012, 2012.

[60] N. Naik and P. Jenkins, "Discovering hackers by stealth: Predicting fingerprinting attacks on honeypot systems," in *2018 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 2018, pp. 1–8.

[61] N. Nikiforakis, W. Joosen, and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 820–830.

[62] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 541–555.

[63] X. Pan, Y. Cao, S. Liu, Y. Zhou, Y. Chen, and T. Zhou, "Cspautogen: Black-box enforcement of content security policy upon real-world websites," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 653–665. [Online]. Available: https://doi.org/10.1145/2976749.2978384

[64] K. Parikh, D. Singh, D. Yadav, and M. Rathod, "Detection of web scraping using machine learning," *Open access international journal of Science and Engineering*, pp. 114–118, 2018.

[65] R. Puri, “Bots & botnet: An overview,” *SANS Institute*, vol. 3, p. 58, 2003.

[66] S. X. Rao, S. Zhang, Z. Han, Z. Zhang, W. Min, Z. Chen, Y. Shan, Y. Zhao, and C. Zhang, “xfraud: Explainable fraud transaction detection on heterogeneous graphs,” *Proceedings of the VLDB Endowment*, no. 3, pp. 427–436, 2021.

[67] S. Rees-Pullman, “Is credential stuffing the new phishing?” *Computer Fraud & Security*, vol. 2020, no. 7, pp. 16–19, 2020.

[68] S. R. Sahoo and B. Gupta, “Real-time detection of fake account in twitter using machine-learning approach,” in *Advances in computational intelligence and communication technology*. Springer, 2021, pp. 149–159.

[69] T. Saito, K. Yasuda, T. Ishikawa, R. Hosoi, K. Takahashi, Y. Chen, and M. Zalasiński, “Estimating cpu features by browser fingerprinting,” in *2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE, 2016, pp. 587–592.

[70] T. Saito, K. Yasuda, K. Tanabe, and K. Takahashi, “Web browser tampering: inspecting cpu features from side-channel information,” in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2017, pp. 392–403.

[71] I. Sanchez-Rola, I. Santos, and D. Balzarotti, “Clock around the clock: Time-based device fingerprinting,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1502–1514.

[72] J. Spooren, D. Preuveneers, and W. Joosen, “Mobile device fingerprinting considered harmful for risk-based authentication,” in *Proceedings of the Eighth European Workshop on System Security*, 2015, pp. 1–6.

[73] O. Starov, P. Laperdrix, A. Kapravelos, and N. Nikiforakis, “Unnecessarily identifiable: Quantifying the fingerprintability of browser extensions due to bloat,” in *The World Wide Web Conference*, 2019, pp. 3244–3250.

[74] O. Starov and N. Nikiforakis, “Xhound: Quantifying the fingerprintability of browser extensions,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 941–956.

[75] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki *et al.*, “Data breaches, phishing, or malware? understanding the risks of stolen credentials,” in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1421–1434.

[76] K. Thomas, J. Pullman, K. Yeo, A. Raghunathan, P. G. Kelley, L. Invernizzi, B. Benko, T. Pietraszek, S. Patel, D. Boneh *et al.*, “Protecting accounts from credential stuffing with password breach alerting,” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1556–1571.

[77] C. F. Torres, H. Jonker, and S. Mauw, “Fp-block: usable web privacy by controlling browser fingerprinting,” in *European Symposium on Research in Computer Security*. Springer, 2015, pp. 3–19.

[78] E. Trickel, O. Starov, A. Kapravelos, N. Nikiforakis, and A. Doupé, “Everyone is different: Client-side diversification for defending against extension fingerprinting,” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1679–1696.

[79] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy, “Fp-scanner: The privacy implications of browser fingerprint inconsistencies,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 135–150.

[80] ——, “Fp-stalker: Tracking browser fingerprint evolutions,” in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 728–741.

[81] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, “Fp-crawlers: studying the resilience of browser fingerprinting to block crawlers,” in *MADWeb’20-NDSS Workshop on Measurements, Attacks, and Defenses for the Web*, 2020.

[82] K. C. Wang and M. K. Reiter, “Detecting stuffing of a {User’s} credentials at her own accounts,” in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2201–2218.

[83] K. Weldemariam, “Fpguard: Detection and prevention of browser fingarprinting,” in *Data and Applications Security and Privacy XXIX: 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13-15, 2015, Proceedings*, vol. 9149. Springer, 2015, p. 293.

[84] S. Wu, S. Li, Y. Cao, and N. Wang, “Rendered private: Making GLSL execution uniform to prevent webgl-based browser fingerprinting,” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1645–1660.

[85] C. Yang, L. Wang, H. Cao, Q. Yuan, and Y. Liu, “User behavior fingerprinting with multi-item-sets and its application in iptv viewer identification,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2667–2682, 2021.

[86] R. Zegers, “HTTP header analysis,” *University of Amsterdam Master Thesis*, 2015.