Finding Behavioural Biometrics Scripts on the Web Using Dynamic Taint Analysis

Alexandru Bara*, Aswad Tariq†, and Urs Hengartner‡
Cheriton School of Computer Science
University of Waterloo
*abara@uwaterloo.ca, †aswad.tariq@uwaterloo.ca, ‡urs.hengartner@uwaterloo.ca

***Abstract*—Behavioural biometrics have emerged as a transformative security mechanism for the web, leveraging user interaction patterns like keystrokes and mouse movements for authentication. Detecting scripts that perform behavioural biometrics at scale remains challenging due to code obfuscation, dynamic execution, and overlap with analytics scripts. We aim to get an understanding of how widely deployed such scripts are by crawling more than 20K websites, including the Tranco Top 20K list, 500 bank websites, and more than 1K e-commerce websites. Our crawlers can locate checkout and login webpages where sensitive information is entered, making these websites more likely to deploy behavioural biometrics. We develop the first open-source crawler to navigate an e-commerce website to its checkout page, achieving 78% accuracy on Shopify-based websites. Our crawlers rely on a dynamic taint analysis-aware web browser to find websites that use scripts to access keystroke or mouse information and that extract this information to backend servers. We also build a ground truth dataset of behavioural biometrics scripts and create a machine learning pipeline to automatically filter out scripts that show no behavioural biometrics characteristics. Our analysis reveals that behavioural biometrics scripts are deployed on at least 0.31% and potentially up to 0.50% of the Tranco Top 20K websites, with significantly higher adoption on bank login pages. We conclude with recommendations to balance security benefits with privacy risks, advocating for transparency, deobfuscation, and regulatory oversight.**

I. Introduction

In an era of escalating cyber threats, traditional authentication mechanisms, like passwords and two-factor authentication, increasingly fall short of addressing sophisticated attacks, such as credential stuffing, session hijacking, and social engineering. Behavioural biometrics have emerged as a transformative solution, leveraging unique user interaction patterns, like keystroke dynamics, mouse movements, touchscreen gestures, and navigation behaviours, to authenticate users transparently and continuously. Unlike static credentials, behavioural biometrics operate in the background, analyzing real-time interactions with a website to detect anomalies that may signal fraud. For instance, a banking platform might flag a transaction if the user's typing rhythm during login deviated from their historical profile, even if their password was correct.

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2026
27 February 2026, San Diego, CA, USA
ISBN 978-1-970672-06-0
https://dx.doi.org/10.14722/madweb.2026.23023
www.ndss-symposium.org

This technology is particularly critical for securing high-risk workflows, such as financial transactions, healthcare portals, and enterprise systems. However, its adoption raises significant questions: How pervasive are behavioural biometrics on the web? Where are they deployed? Are sensitive pages like logins or checkouts prioritized? Answering these questions is essential for understanding modern web security practices and their privacy implications.

On the web, behavioural biometrics usually rely on JavaScript scripts running in a browser to collect data about user interactions and sending this data to backend servers for analysis. These scripts are difficult to detect due to three key challenges: 1) Dynamic execution: These scripts often operate dynamically, collecting data only during specific user interactions (e.g., form submissions and mouse movement). 2) Obfuscation: Vendors heavily obfuscate code to prevent reverse engineering, masking data flows even from advanced static analysis tools. 3) Overlap with web analytics: Behavioural biometrics scripts share data collection patterns with session replay and heatmap tools, complicating classification.

Existing studies [1]–[4] have focused on finding fingerprinting and tracking scripts but there has not been any approach that detects behavioural biometrics scripts across the broader web. This paper seeks to address this gap as well as provide a ground truth for behavioural biometrics scripts. We address six research questions:

- Prevalence: How widespread are behavioural biometrics scripts on the web? (**RQ1**)
- Sensitive Pages: Are these scripts disproportionately deployed on pages handling sensitive data (e.g., login and checkout pages)? (**RQ2**)
- Login vs. Main Pages: Are login pages more likely to host behavioural biometrics scripts than main pages? (**RQ3**)
- Banking vs. Nonbanking Logins: Do banking login pages employ these scripts more frequently than non-banking login pages? (**RQ4**)
- Checkout Pages: Do checkout pages host these scripts more likely than other e-commerce pages? (**RQ5**)
- Shopify: Does Shopify, the leading e-commerce platform [5], enable behavioural biometrics? (**RQ6**)

We aim to get an understanding of how widely deployed behavioural biometrics scripts are by crawling more than 20K websites, including the Tranco Top 20K list [6], 500 bank websites, and more than 1K e-commerce websites with a focus on Shopify-based ones. Our crawlers can locate webpages where sensitive information, such as a password or credit card information, is entered, making the deployment of additional security mechanisms, like behavioural biometrics, by the webpage administrator more likely. For locating such webpages, we design a novel crawler that navigates from a shopping website's main page to its checkout page with accuracy of 78% on Shopify-based stores.

Our crawlers rely on a dynamic taint analysis-aware web browser, FoxHound [7], to locate websites that use scripts to access keystroke or mouse information and extract this information to backend servers. We improve FoxHound's tracking of fine-grained data flows from user interactions to exfiltration points, boosting its detection effectiveness from 20% to 97% across 15 real-world behavioural biometrics scripts.

To distinguish behavioural biometrics scripts from similar tracking scripts that also collect and extract mouse or keystroke information (e.g., session replay tools [8]–[10]) among our crawled scripts, we develop a classifier trained on a dataset of 126 manually labelled scripts. Each script is characterized by a set of features extracted from dynamic taint analysis, which capture the complexity and behaviour of data flows.

Our paper is the first one to present quantitative evidence for the deployment of behavioural biometrics on the web. We make the following contributions:

- **BBAT**: An open-source behavioural biometrics analysis framework [11] consisting of a significantly enhanced FoxHound browser, the first open-source checkout crawler, and a machine learning pipeline to filter out scripts showing no behavioural biometrics characteristics.
- **Tracking Scripts Dataset**: A set of hundreds of tracking scripts manually classified as either behavioural biometrics or not [12].
- **Empirical Insights**: An analysis showing that behavioural biometrics are deployed on at least 0.31% and potentially up to 0.50% on the Tranco Top 20K websites, with banking login and checkout pages showing higher adoption than main pages.
- **Recommendations**: Recommendations for balancing security benefits with privacy risks, particularly regarding data collection and obfuscated scripts.

II. Background and Related Work

A. Browser Fingerprinting

JavaScript scripts for browser fingerprinting enable the tracking of users across websites by exploiting subtle differences in data exposed through browser APIs. These scripts systematically query a range of APIs that return values indicative of the user's unique browser environment [13]–[19].

Initially, browser fingerprinting scripts were relatively uncommon on the web. In 2013, Nikiforakis et al. [1] demonstrated that only 0.4% of 10,000 websites employed fingerprinting scripts. More recently, in 2024, Boussaha [4] reported that, under a relaxed definition of fingerprinting, approximately 75% of websites employ such techniques. These findings underscore a rapid increase in both the prevalence and sophistication of browser fingerprinting over time.

While browser fingerprinting is often criticized for its potential to compromise user anonymity, it also offers security benefits. Laperdrix et al. [20] discuss how these techniques can be effectively employed for bot detection, vulnerability patching, and enhancing authentication. Antonin et al. [21] explored the integration of browser fingerprinting into multi-factor authentication, reporting that a dozen security-oriented vendors actively collect fingerprint data during sign-up, sign-in, and payment procedures. Senol et al. [22] demonstrated that login pages are more likely to incorporate fingerprinting scripts compared to standard pages, highlighting the strategic use of these scripts in improving security measures. Collectively, these studies illustrate that browser fingerprinting can serve as a valuable component in comprehensive security frameworks.

B. Behavioural Biometrics

Behavioural biometrics leverage the way users interact with websites to authenticate them. The underlying scripts operate by capturing mouse, keyboard, or touch events via active event handlers. This data is aggregated and transmitted to backend servers, where models employing keystroke, mouse, or swipe dynamics are used to validate the user. To the best of our knowledge, no previous research has examined these scripts or their prevalence on the web.

Keystroke dynamics were introduced by Gaines et al. [23], who analyzed the timing of individual keystrokes to differentiate users from a small sample. Subsequent studies have explored additional features such as the duration a key is held down and the time interval from the initiation to the full depression of a key. Recent approaches have incorporated machine learning models to enhance user differentiation [24]. Samura et al. [25] demonstrated that using a support vector machine allows to distinguish among 112 users with 100% accuracy, showcasing the potential of this authentication method.

Ahmed et al. [26] introduced mouse dynamics by proposing a system utilizing artificial neural networks to distinguish users, achieving a false acceptance rate (FAR) of 2.4649% and a false rejection rate (FRR) of 2.4614%. Subsequent research examined additional features such as movement offset and curvature speed to build more robust models [27]. Notably, Shen et al. [28] managed to reduce the FAR to 1.18% and the FRR to 1.96%. Similarly, swipe patterns, which are analogous to mouse dynamics but designed for touchscreen devices, have been employed in authentication systems [29].

While existing research has shown that browser fingerprinting can enhance security on sensitive pages, such as login screens, behavioural biometrics scripts provide an additional layer of protection. To illustrate their importance, consider a phishing attack on browser fingerprinting demonstrated by Lin et al. [30], where a malicious website harvests a user's email, password, and browser fingerprint. By altering their fingerprint to mimic the victim's, the attacker bypasses the browser fingerprinting security measures deployed on some login pages. Incorporating behavioural biometrics makes such attacks significantly harder as the attacker would have to replicate the authentic behavioural patterns exhibited by the legitimate user during login.

Behavioural biometrics offer more than just enhanced authentication; they are also valuable for distinguishing between real users and bots. They also play a crucial role in preventing various types of fraud, including click and payment fraud or cheating in video games [31].

III. Taint Analysis and Behavioural Biometrics

Taint analysis is "a process used in information security to identify the flow of user input through a system to understand the security implications of system design" [32]. Taint sources are the points where the data that we want to track through an application originates. Taint flow is the path through which tainted data passes through the application. Taint sinks are usually the points where the (maybe processed) tainted data leaves the application.

On a computer, the two main features that behavioural biometrics scripts track are mouse movements and keyboard interactions. Such a script will usually have event listeners for keyboard and mouse events. It will take the properties of the events, like the mouse position or the key press, and send the data to a server for analysis. Taint analysis can be used to find scripts that may be used for behavioural biometrics. Namely, we mark mouse and keyboard events as taint sources and exit points of a script leading to network transmission as taint sinks. Only scripts with a data flow from taint sources to taint sinks may be behavioural biometrics.

Studying this taint flow can be done statically or dynamically. Static analysis examines a script without executing it. Since the analysis does not take into consideration if code gets executed, it may mistakenly detect taint flows for scripts that are on a webpage but that are not active. Static analysis also tends to fail when code is heavily obfuscated [33], as is often the case for JavaScript scripts. Dynamic analysis involves running the code and looking at its behaviour. It is less likely to have problems with obfuscated code since code complexities will be resolved at runtime. For these reasons, we rely on dynamics taint analysis to find behavioural biometrics scripts and websites using them.

A. *Dynamic Taint Analysis-Aware Browser*

We analyzed several dynamic taint analysis-aware browsers [7], [34]–[38] and ended up picking FoxHound [7] since it is open-source, provides fine-grained tracking, allows customization of taint sources, and is actively maintained. FoxHound introduces taint awareness into both SpiderMonkey and Gecko, enabling precise tracking of data as it moves through various browser subsystems. SpiderMonkey is Firefox's JavaScript engine, responsible for parsing, interpreting, and executing JavaScript code. Gecko acts as Firefox's layout and rendering engine, managing tasks such as HTML/CSS rendering, DOM manipulation, networking, and graphics rendering. FoxHound is the state-of-the-art tool when it comes to dynamic taint analysis in browsers.

To enhance FoxHound's ability to track taint flows in behavioural biometrics scripts, we added additional features, which we have forwarded to the FoxHound maintainers.

- Boolean Tainting: We added taint propagation for boolean values and resolved issues with distinguishing between tainted boolean primitives and boolean objects without disrupting JavaScript behaviour.
- Math Library Enhancements: We added taint propagation to the built-in math library, ensuring that operations involving tainted numeric values maintain their taint metadata.
- Typed Arrays: We marked typed arrays as sinks to handle preprocessing scenarios where behavioural biometrics data is staged before exfiltration.
- Web Worker Communication: We enabled taint tracking across web workers by modifying the serialization and deserialization processes in SpiderMonkey.
- Taint Propagation: We identified additional places where the taint was lost and ensured that it was propagated.
- Sink Detection: We improved sink detection mechanisms to account for edge cases, such as hidden form elements accessed in unique ways by some scripts.

B. *Evaluation*

We tested our enhancements against known behavioural biometrics scripts. To find such scripts, we searched the web for vendors that offer behavioural biometrics solutions. Each company was analyzed to confirm that their definition of behavioural biometrics aligned with ours. Once verified, we attempted to locate their scripts by examining official documentation and identifying websites listed as customers. We identified 15 behavioural biometrics vendors (see Table I). For each vendor, we determined the URL of a customer website that deployed the vendor's behavioural biometrics scripts and accessed the URL with FoxHound.

TABLE I
Behavioural Biometrics Vendors.

Ping Identity	ThreatMark	Nudata
Iovation	BehavioSec	BioCatch
CallSign	TypingDNA	Transmit Security
Neuro-Id	Sardine.ai	Accertify
f5	Forter	Human

Without our enhancements, Foxhound tracked taint flows in 3 out of 15 active scripts, achieving an effectiveness rate of 20%. With our enhancements, 14 out of 15 scripts were tracked, resulting in an effectiveness rate of 93%. This represents an almost fivefold improvement in effectiveness. Enhanced FoxHound fails to find a taint flow for one vendor (Forter). Our manual analysis of their heavily obfuscated script reveals that the taint gets lost in a decision tree implemented with many conditional statements. To enable FoxHound to track this flow, we would need to add implicit flow tainting, which is expensive and may lead to many false positives [4].

IV. Checkout Crawler

Web crawlers are automated programs designed to systematically explore and index web content. There are many types of crawlers, each designed to address specific tasks. Several of our research questions focus on sensitive pages that contain private information, such as passwords and credit card numbers. Accordingly, we differentiate between login and checkout crawlers.

Login crawlers specialize in automating authentication workflows to analyze security mechanisms and access restricted content. Three open-source login crawlers are SSO-Monitor [39], Innocenti et al.'s crawler [40], and Double-Edged Sword [22]. They primarily utilize heuristic and regular expression-based methods. When multiple login pages are present, these tools identify only a single candidate, with the exception of SSO-Monitor. We employ an enhanced version of SSO-Monitor through integration of a vision-language model. This model analyzes webpage screenshots and determines interactive elements leading to login pages. Additionally, we incorporate a machine learning classifier that evaluates each screenshot to determine whether it represents a login page. This enhancement provides significant advantages over standard SSO-Monitor by effectively crawling multilingual websites without relying on predefined heuristics or keyword-based methods, thus ensuring more comprehensive coverage of diverse website login flows.

Checkout crawlers are needed for e-commerce websites. An e-commerce website lets customers look at products and purchase them. The users that purchase these items need to provide a form of payment, such as credit cards, debit cards or gift cards, on a checkout page. A fraudulent transaction is when a form of payment is used without the consent of the user to whom the form of payment belongs. One of the ways to ensure the person completing a purchase is who they say they are is based on behavioural biometrics.

To determine if an e-commerce website employs behavioural biometrics on its checkout page, we need to navigate to the checkout page. Usually, it is not possible to navigate directly (i.e., without adding items to the cart) to this page. We developed a checkout crawler designed to simulate typical user interactions on an e-commerce website. Our crawler navigates from the main page to a product, adds the item to the cart, and attempts to reach the checkout page. To the best of our knowledge, there is no other open-source checkout crawler.

A. E-Commerce Website Flow

Understanding the flow of an e-commerce website is essential for building an effective checkout crawler. We conducted a detailed analysis of the top 100 Shopify stores obtained by taking the first 100 entries from the top 1K Shopify stores [41]. We chose Shopify because it is widely used for creating e-commerce websites [5], offering a diverse range of site structures and purchasing flows.

We aimed to identify common patterns in the user journey from browsing to checkout on an e-commerce website. The flow shown in Figure 1 represents a generalization of the found possibilities. Some websites have only a subset of the shown arrows and pages. The only two pages that are guaranteed in an e-commerce website are the main page and the checkout page, and the only requirement is that there is at least one item in the cart.

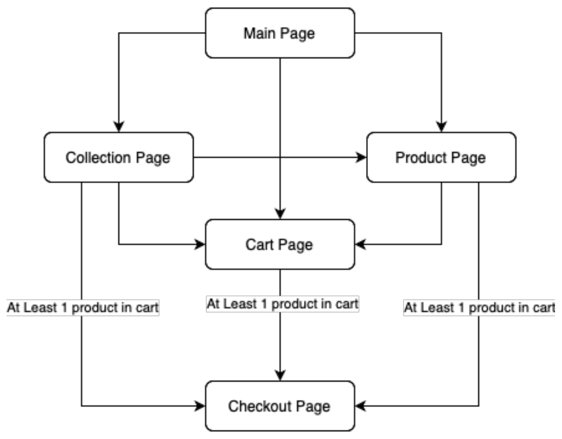
Main Page
Collection Page
Product Page
Cart Page
At Least 1 product in cart
At Least 1 product in cart
At Least 1 product in cart
Checkout Page

Fig. 1. E-commerce checkout flow.

The main page is the landing point of an e-commerce website, often showcasing featured products, curated collections, or promotional content. A collection page organizes products into specific categories, such as a summer collection featuring items for warm weather. A typical product page provides detailed information about a specific item, including a description, price, and specifications. Once products are added to the cart, users can go to the cart page to look at all chosen items. Once the checkout button is pressed, the user is usually directed to a checkout page. For our purposes, successfully reaching this checkout page signifies a successful crawl.

B. Crawler Design

The design of our checkout crawler is guided by the insights gained from analyzing e-commerce website flows. The primary goal was to create a flexible yet effective crawler that could handle a variety of e-commerce structures without being overly tailored to specific cases. Striking a balance between generality and performance was crucial to ensure broad applicability across different platforms.

Our checkout crawler is built on Klein et al.'s crawling framework [42] designed for easy extensibility and customization. We added our own checkout module, which contains the logic used to go from the main page to the checkout page. We released our crawler as open source [43].

1) High Level Design: For each e-commerce website, we first gather 31 web pages (see below) that we are going to crawl. Since all e-commerce websites are different, no one flow can fit them. For this reason, we created four crawling modules that each run a slightly different setup to ensure we have the highest possible coverage. Each crawling module is composed of up to four components. The high-level design can be seen in Figure 2. We run the four crawling modules for each of the 31 web pages.

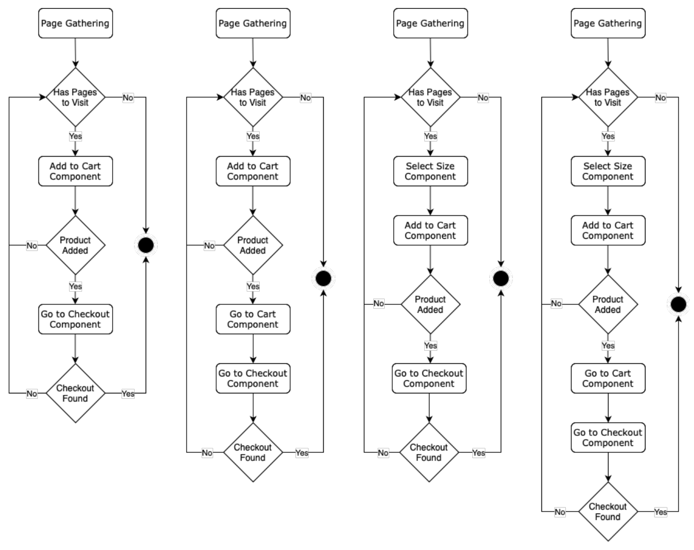

Fig. 2. Crawling modules: Simple, Cart, Size, and Cart and Size (from left to right).

2) Page Gathering: The first part of the checkout crawler is page gathering. From the main page of an e-commerce website, we extract ten random links and store them in a database. For each of the found links, we pick a random link from that webpage and add it to the database. We repeat the process for the new found links. At the end, we will have 31 web pages. Our reasoning behind this approach is as follows: The main page usually does not have many products on it. We extract ten random links hoping that one of them is a collection page or a product page. From our observations, this is frequently the case. For each new web page, we find and record one extra link to find product pages. If we are on a collection page, most of the links will be product page links. So by randomly picking a link, we likely find a product page. We look at only one link for speed; the more web pages we look at, the longer the crawl will take. We found this to be a good compromise between speed and accuracy. Our last choice is to go to a depth of three for the web pages (1 main page and 10 pages at each lower level for a total of 31). Some websites have product pages deep in the navigation chain. Having a higher depth allows us to more likely find these pages.

We stop crawling a website when a checkout page is reached and use the extra stealth plugin [44] to avoid bot detection scripts.

3) Crawling Modules: Our crawler consists of four modules.

Simple Module: This module will go to the page currently being crawled. The module will call the Add to Cart component (see below). If it detects that items were added to the cart, it will call the Go to Checkout component.

Cart Module: The difference between the Simple Module and the Cart Module is that in the Cart Module, when one or more items have been added to the cart, the Go to Cart component is called instead of the Go to Checkout component. After the Go to Cart Module is finished, the Checkout component is called.

Size Module: This module is similar to the Simple Module but before calling the Add to Cart component, it will call the Size Selection component.

Size and Cart Module: This module combines the changes from the Cart Module and Size Module into one. It starts by calling the Size Selection component, followed by the Add to Cart component. If items were added to the cart, the Go to Cart component is called followed by the Checkout component.

4) Components: A crawling module can consist of up to four components.

Size Selection: This component selects the size of a product. The sizes we are looking for are “l”, “large” and “10”. These sizes were chosen based on the most common sizes seen in the top 100 Shopify websites. We currently do not support more complicated selections that involve colours or use dropdown menus.

Add to Cart: This component looks for the following keywords: “add to cart”, “add to bag”, “add to basket”, “add to tote”, and “add”. These keywords were chosen based on our observations from the top 100 Shopify websites. The crawler will take all buttons that contain these keywords and click on up to ten of these buttons. After the “add to cart” button is clicked, some websites redirect users to a product page, while others add the item to the cart without leaving the current page. If we detect that the webpage is no longer the same, we assume that we are now on the correct product page, and we re-run this component. We only do this with a depth of two. We also make sure that if a button from the keywords is pressed properly, we stop pressing buttons that contain one of the keywords. This speeds up the time taken per crawl.

Go to Cart: This component is in charge of going to the cart. It will look through all the links on the page and filter them based on which ones contain the word cart in it. It will then sort the links based on length and visit the shortest link. This approach has been taken since almost all webpages from the top 100 Shopify list have a cart page with the link in the format of the website name followed by the word cart.

Go to Checkout: From the most common e-commerce flow, the typical keywords used for checkout are the following: “Checkout”, “Check out”, “Proceed”, “Place Order”, and “Complete Purchase”. This components looks for buttons or text in the HTML that contain these keywords.

C. Evaluation

To evaluate our checkout crawler, we first used a dataset of the top 1,000 Shopify websites [41]. This dataset allowed us to measure performance in a familiar environment.

To test the crawler’s generalizability, we needed a more diverse dataset. We observed that existing datasets of top e-commerce websites often include non-commercial sites, such as news platforms, blogs, and product information websites with no ability to purchase items. To address this, we manually curated a set of 118 truly commercial websites from the top 200 Canadian e-commerce sites [45].

Our crawler has an accuracy of 78% and a precision of 98% on the Shopify dataset, that is, our crawler reaches 78% of existing checkout pages and when it reports reaching a checkout page, it is correct 98% of the time. Our crawler achieved an accuracy of 54% and a precision of 94% for the curated set.

False negatives are websites that have a checkout page but our crawler did not find them. For the Shopify dataset, we observed 199 false negatives. Reasons are 1) bot detection, 2) websites that do not primilarly sell products (e.g., a blogging website mainly hosting blog posts but also selling branded t-shirts), where products may be hard to reach and our strategy of choosing random links likely fails, 3) pages that use keywords that we have not anticipated for the different parts of the checkout process (e.g., an e-commerce store selling baby products that uses "place in the nursery" for the add to cart flow, 4) complicated size selection using drop-down menus or requiring manual entry, 5) websites thare are not in English, and 6) the opening of new tabs or windows with a different URL domain than the main website, causing the crawler to stop. Many of these issues could be addressed by adding more keywords and features to our crawler.

For the curated dataset, we observed 50 false negatives. The reasons are similar to the issues encountered on Shopify websites. We noticed that size selection for non-Shopify websites is a lot more complex than for Shopify websites. Our limited heuristics for selecting sizes is the main reason why our crawler did not perform as well on this dataset.

Whereas accuracy is on the low side for the curated dataset, precision remains high. This is important for our usage scenario, knowing how widely distributed behavioural biometrics scripts are on checkout pages. Low accuracy will lead to underreporting, which is conservative and acceptable, but when finding a checkout page candidate, we want to be sure that it really is a checkout page.

V. Machine Learning Pipeline

Our evaluation in Section III revealed that behavioural biometrics scripts can be effectively detected using dynamic taint analysis, where user-triggered events serve as sources and network transmission mechanisms act as sinks. However, behavioural biometrics scripts are not the only category of scripts exhibiting these data flow patterns. They are also used by session replay, heatmap, and other types of scripts [8]–[10].

Manual classification of taint reports to identify behavioural biometrics scripts is impractical at scale due to obfuscation, vendor diversity, and overlapping behaviours across script categories. Machine learning provides a scalable solution by learning statistical patterns in taint flows. Instead of building a fully generalizable classifier—which our rather small ground truth dataset cannot support—we employ machine learning as a filtering mechanism. The filtering identifies scripts that show no behavioural biometrics characteristics, reducing the number of scripts that require costly manual analysis. This section describes the construction of our ground truth dataset, the features extracted from taint flows, the building of the classifiers and how we combine them for our filtering use case.

A. Building Ground Truth

A machine learning classifier needs a ground truth dataset. However, there are no public lists of websites that use behavioural biometrics and no public repositories of behavioural biometrics scripts from vendors providing such services. Therefore, we need to assemble our own ground truth dataset. This is difficult since behavioural biometrics scripts are often (heavily) obfuscated and usually limited to gathering and transmitting behavioural data, where the analysis of the data happens on the server. This makes it difficult to say whether a script that gathers behavioural data is used for behavioural biometrics or some other purpose, like session replay.

Due to these difficulties, our approach for locating behavioural biometrics vendors and websites of customers deploying their scripts is best effort. Our approach follows two principles: 1) We focus on vendors that provide behavioural biometrics, as announced on their websites. If a vendor provides behavioural biometrics only as one of several services, we do not further consider this vendor. 2) We include a script in our ground truth dataset only if it can be reliably linked back to a known behavioural biometrics vendor, for example, via a copyright statement or other vendor-specific identifiers.

Building our ground truth dataset of taint records from behavioural biometrics scripts requires a web crawler capable of simulating user interactions and a curated list of websites known to use behavioural biometrics scripts.

1) Crawler: Our crawler builds on Klein et al.'s framework [42]. We extended this crawler by integrating a module that simulates genuine user behaviour. This enhancement enables the crawler to trigger events that closely mimic those of a real user, thereby producing more realistic interaction data.

The crawler is configured to execute a predefined sequence of mouse and keyboard events consistently across all websites (see Appendix A) ensuring uniformity during both the ground truth data collection and the large-scale analysis phases.

When executed within our taint-aware browser environment, this enhanced crawler generates taint reports each time a taint flow reaches a defined sink. Integrated event listeners capture and store these reports, thereby facilitating subsequent data analysis.

2) List of Websites: We leveraged the list of web pages discussed in Section III-B representing customers of vendors that offer behavioural biometrics services (see Table I). After obtaining the behavioural biometrics scripts from these web pages, we extracted distinctive keywords from the scripts, which we then used in tools like NerdyData [46] to search across a large dataset of websites and their deployed scripts to find additional customer websites and potential behavioural biometrics scripts. We ended up with 114 customer websites, representing a diverse array of vendors known to provide behavioural biometrics services. It is important to note that not all customer websites actively use these services, and it is possible that some scripts are not active. We added 20 websites to our crawl that are known to contain different web analytics scripts that use mouse and keyboard data. These websites were found by searching for different web analytics vendors and finding their clients.

3) Ground Truth: After executing our crawler across the identified websites, we ended up with taint data from 126 distinct scripts across the known vendors, with some websites contributing multiple scripts while others produced none. To establish ground truth, we manually analyzed each script to determine whether it exhibited behavioural biometrics characteristics. Most of these scripts were obfuscated, making the analysis difficult. We employed a multi-step approach:

Reference to Known Behavioural Biometrics Scripts: We began by comparing each script against the known behavioural biometrics scripts obtained from web pages representing customers of vendors offering behavioural biometrics services (see Section III-B). This initial comparison allowed us to quickly identify scripts with previously documented behavioural biometrics capabilities.

Examination of Script Metadata: For scripts not identified in the first step, we looked at the script headers for any vendor identifiers or metadata. If a vendor name was present, we researched the organization's services to determine if they were associated with behavioural biometrics or related tracking technologies.

Keyword Analysis: In instances where no explicit vendor information was available, we conducted a keyword analysis within the script. Keywords known to be associated with behavioural biometrics activities, such as `wupServerURL` (commonly found in BioCatch scripts) served as indicators of behavioural biometrics functionality.

Analysis of Data Exfiltration Endpoints: When the above methods did not yield definitive classification, we analyzed the endpoints to which the script transmitted data. By examining the destination URLs and conducting public scans, we aimed to associate the data recipients with known entities or services. This helped infer the script's purpose based on the nature of the receiving organization.

Through this comprehensive methodology, we successfully classified all 126 scripts. While 126 scripts may seem limited, it provides a crucial first step toward automated classification. The dataset covers behavioural biometrics scripts and a diverse range of web analytics scripts with different ways of tracking user's actions, allowing the model to learn general patterns. Future work should expand the dataset using additional sources to improve generalization.

This investigation resulted in 56 scripts being confirmed as behavioural biometrics scripts. The 70 scripts that were not behavioural biometrics consisted of heatmap, web analytics, session replay and tracking scripts. This annotated dataset now serves as the ground truth for training and evaluating our machine-learning model.

B. Feature Extraction

The next step is to extract meaningful features to train our machine learning model. We use features derived from a script's taint reports but not from the script itself. Script-based features may inadvertently focus on vendor-specific properties of a script, like the presence of a keyword, leading a classifier to excel at identifying scripts from known vendors but performing poorly for unknown vendors. As we will show in Section VII, our strategy leads to the successful identification of several new vendors. To construct our feature set, we process each taint report as follows:

Script Involvement: Extract all scripts that contributed to a given taint report. We store all taint reports associated with each script.

Source and Sink Frequency: For each script, count the occurrences of source events present in its associated taint reports. This count reflects the volume of sensitive data (mouse, touch and keyboard events) that are being exfiltrated by the script. Similarly, count the occurrences of sink events linked to the script. This count illustrates both the frequency and the mechanisms of data exfiltration (e.g., network request, socket communication).

Taint Report Volume: Measure the total number of taint reports generated by each script. A higher number of taint reports suggests more frequent interactions with user-generated events.

Taint Flow Characteristics: Compute the total number of taint flows per script, as each taint report may contain multiple flows. Scripts with a higher number of taint flows likely process and modify user interaction data extensively before reaching a sink. We also track the total number of nodes within the taint flows and sum them up. Each node represents an operation performed on tainted data, such as arithmetic calculations or string manipulations. Behavioural biometrics scripts often perform multiple transformations on user interaction data before sending it to a sink, distinguishing them from simpler tracking mechanisms.

C. Evaluation

Our machine learning pipeline integrates the ground truth labels for each script with the corresponding feature set, which is then used to train a range of machine-learning models. To enhance the model's ability to correctly identify behavioural biometrics scripts, we implemented an oversampling strategy that doubles each script labelled as behavioural biometrics.

We split our data into a training set and a testing set that contains 80% and 20% of the data, respectively. We decided not to go with a validation set because we do not have enough data to split in a 80%, 10% and 10% manner for the training, testing and validation sets. We evaluated five popular classifiers: random forests, SVM, KNN, XGBoost and neural networks. We used a grid search on each model to find the best parameters for them using a 5-fold cross-validation. The best hyperparameters for each model can be found in Appendix B. Using these hyperparameters, we evaluated the models across multiple performance metrics. Our results are given in Table II. Random Forest performs best with an accuracy of 96% and AUC-ROC of 0.99 on the test set.

TABLE II
EVALUATION METRICS FOR MACHINE LEARNING MODELS.

Metric	Random Forest	SVM	KNN	XGBoost	Neural Networks
Accuracy	0.96	0.88	0.92	0.92	0.92
Precision	0.96	0.88	0.93	0.93	0.93
Recall	0.96	0.88	0.92	0.92	0.92
F1-score	0.96	0.88	0.92	0.92	0.92
AUC-ROC	0.99	0.96	0.92	0.99	0.92

To assess the extent of overfitting, we compared model performance on the training and test sets. We observed that all models achieved over 98% precision on the training set, whereas their performance on the test is on average 92% for all models. This discrepancy suggests that the models may have learned dataset-specific patterns rather than generalizable characteristics of behavioural biometrics scripts.

To get past the overfitting limitation, we employ an ensemble approach using all five classifiers. For each script, if any one of the classifiers labels it as a behavioural biometrics, the script is flagged as a potentially behavioural biometrics. Although this strategy increases the number of false positives, it effectively minimizes false negatives. This strategy is consistent with our goal to automatically filter out scripts that show no behavioural biometrics characteristics to reduce manual classification effort. The resulting subset of potentially behavioural biometrics scripts is then manually reviewed. To manually classify such a script as either behavioural biometrics or not, we employed the same approach as in Section V-A. The scripts we cannot classify are classified as potentially behavioural biometrics scripts.

VI. BBAT

Our analysis framework, called BBAT (Behavioural Biometrics Analysis Tool) [11] , is an open-source framework based on the crawling infrastructure developed by Klein et al. [42]. The framework comprises three primary components:

Dynamic Taint Analysis Browser: We use our enhanced version of FoxHound described in Section III to track sources and sinks relevant to our study. (A comprehensive list of sources and sinks is provided in Appendix C.)

Custom Crawlers: We employ three custom crawlers:

1) **Simple Crawler**: This crawler loads a web page and runs our user simulation module (see Appendix A). It is used for analyzing individual pages that are directly accessible via a URL.
2) **Checkout Crawler** (see Section IV). Checkout pages are not directly reachable by URL, as they typically require items to be present in a shopping cart. This crawler navigates to the checkout page and executes our user simulation module on the checkout page.
3) **Login Crawler** (see Section IV). This crawler locates the login page of a given website. As it does not perform user simulation, an identified login page must be subsequently processed by the Simple Crawler.

Machine Learning Pipeline: We apply the machine learning pipeline discussed in Section V.

VII. BEHAVIOURAL BIOMETRICS ON THE WEB

A. Web Crawls

Our crawling took place from a Canadian university network between March and April 2025. Across all studies, the data gathered by FoxHound and the respective crawlers is input to our machine learning classifiers, which predict whether a script implements behavioural biometrics. Finally, we manually confirm positive predictions using the same methodology we used for building our ground truth dataset (Section V-A).

Study 1: Prevalence across the Web We applied the Simple Crawler to the top 20K websites from Tranco's list [6]. The crawler navigates to the main page of each site, where our user simulation module is executed to trigger potential script interactions. This study establishes a baseline for how widespread behavioural biometrics scripting is in general web content.

Study 2: Login Pages This study focuses on login pages. We use two datasets: First, we created a dataset containing 500 banking websites by finding all valid banks in the United States that have a website and selecting the first 500 banks. Second, we created a dataset containing 500 random websites by using the first 500 entries successfully crawled from Study 1.

For each website in the two datasets, we initially used the Simple Crawler to obtain baseline data from the main page. Next, we employed the Login Crawler to specifically retrieve the login pages from both sets. The Login Crawler gave us 1,355 login pages for the bank websites and 1,489 login pages for the 500 random websites. (A single website may have more than one login page.)

Study 3: Checkout Pages This study focuses on the e-commerce sector. We use two datasets: one consisting of the top 1K Shopify websites and another comprising 118 additional e-commerce sites (see Section IV).

For these websites, we first used the Simple Crawler to get our baseline. Then, we used the Checkout Crawler to navigate to the checkout pages. Upon reaching a checkout page, the user simulation module is activated, and all subsequent taint flow data is captured and stored.

B. Results

Table III presents the results for each study and run. 3,207 of the Top 20K websites crawled in Study 1 were no longer active. For 3,515 websites, crawling failed due to issues such as HTTP errors, TLS errors, bot detection, page crashes, or timeouts during crawling; about 2,000 of these failures, like bot detection or page crashes, are crawler specific and may be fixable by improving the crawler. Overall, 13,278 of the Top 20K websited were successfully crawled.

The successful navigations from the main to the checkout page on both e-commerce and Shopify websites (41 out of 104 (39%) and 507 out of 897 (57%), respectively) are lower than those observed in Section IV (54% and 78% respectively).

TABLE III
SCRIPTS AND WEBSITES DETECTED AS BEHAVIOURAL BIOMETRICS (BB). A WEBSITE MAY DEPLOY MORE THAN ONE BB SCRIPT.

Crawl	Successfully crawled websites	Scripts with taint flows exfiltrating mouse or keystroke information	Scripts with taint flows and detected as BB by at least one classifier	BB-detected scripts manually confirmed to be BB	BB-detected scripts where manual confirmation failed (i.e., the scripts may be BB)	Websites that use a confirmed BB script (percentage of successfully crawled websites)	Websites that use a BB-detected script where manual confirmation failed (percentage of successfully crawled websites)
Study 1: Top 20k Crawl	13,278	2,913	652	50	36	41 (0.31%)	25 (0.19%)
Study 2: 500 Bank Main Page Crawl	462	63	15	4	0	3 (0.65%)	0 (0.00%)
Study 2: 500 Random Main Page Crawl	500	163	66	6	13	3 (0.60%)	8 (0.16%)
Study 2: 500 Bank Login Page Crawl	462	191	48	31	0	21 (4.55%)	0 (0.00%)
Study 2: 500 Random Login Page Crawl	500	206	84	16	15	5 (1.00%)	10 (2.00%)
Study 3: 1000 Shopify Main Page Crawl	897	661	392	1	0	1 (0.11%)	0 (0.00%)
Study 3: 118 E-Commerce Main Page Crawl	104	52	23	1	0	1 (0.96%)	0 (0.00%)
Study 3: 1000 Shopify Checkout Page Crawl	507	758	407	0	324	0 (0.00%)	188 (37.08%)
Study 3: 118 E-Commerce Checkout Page Crawl	41	31	12	0	9	0 (0.00%)	4 (9.76%)

This discrepancy is likely due to the compounding effects of FoxHound errors combined with the challenges of navigating multiple pages and steps within each website. Each additional navigation step introduces another potential point of failure, hindering progress toward reaching the checkout page.

Table III uses separate columns for listing scripts where our manual analysis (see Section V-A) of a script flagged to be behavioural biometrics by our classifier confirmed this positive classification and where our manual analysis could make neither a positive nor a negative confirmation. In the second case, the script may be behavioural biometrics but it could also be used for other purposes that collect and transmit keyboard or mouse data, such as session replay.

RQ1 (Prevalence) For Study 1, our analysis reveals that less than 1% of the successfully crawled websites contained a behavioural biometrics script, indicating that these scripts are not yet widespread. However, their presence confirms that they are in active use. Although currently uncommon, we believe that behavioural biometrics scripts are in an early stage of adoption and expect their prevalence to grow.

RQ2 (Sensitive Pages) Table III shows that login pages tend to contain a higher number of behavioural biometrics scripts than other types of pages. 4.55% of bank websites deploy these scripts on their login page. For checkout pages, our manually analysis cannot confidently confirm the presence of behavioural biometrics scripts. However, for 37.08% of the Shopify websites, a Shopify-specific script loads on the checkout page, which we suspect to be a behavioural biometrics script. This script was flagged by our machine learning classifier. More information about this script can be found in RQ6. Our findings indicate that behavioural biometrics scripts are more likely to be deployed on sensitive pages.

RQ3 (Login vs. Main Pages) When comparing main pages to login pages, a similar trend emerges: login pages host a greater number of behavioural biometrics scripts than main pages. This pattern is observed for both banking websites (4.55% vs. 0.65%) and less pronounced for random websites (at least 1.00%, potentially up to 3.00% vs. at least 0.6%, potentially up to 0.76%).

RQ4 (Banking vs. Non-banking Login Pages) The login pages of bank websites are more likely (4.55%) to deploy behavioural biometric scripts than the login pages of random websites (at least 1.00%, potentially up to 3.00%). This suggests that entities handling highly sensitive information, such as banks, are more likely to deploy advanced protection techniques on their login pages.

RQ5 (Checkout Pages) Answering whether behavioural biometrics scripts are more likely on checkout pages is challenging because most of the scripts identified on checkout pages could not be definitively confirmed as behavioural biometrics scripts. Based on the available evidence, it is highly likely that these checkout page scripts are intended to serve a behavioural biometrics function.

Another interesting observation is that many of the scripts confirmed not to be behavioural biometrics scripts that were present on the main page were absent on the checkout page. Therefore, sensitive pages, such as checkout pages, are less likely to include tracking scripts for marketing or analytics.

RQ6 (Shopify) Although we cannot conclusively verify that the checkout script on the Shopify website is a behavioural biometrics script, the evidence strongly suggests that Shopify employs such scripts to prevent fraud. This is because the checkout script is loaded exclusively on the checkout page, is not registered with any recognized analytics provider, and utilizes sources and sinks that are consistent with those observed in other behavioural biometrics scripts. We could not find documentation on Shopify's website on these scripts.

Additional Behavioural Biometrics Vendors While manually confirming the detected potentially behavioural biometrics scripts, we discovered four vendors that offer behavioural biometrics and that we missed in our initial search (see Table I). The vendors are Yofi [47], Cheq [48], Group-IB [49], and Untarget.ai [50]. This shows the usefulness of our machine learning pipeline.

Tracking Script Prevalence In Table III, we observe that only a small percentage of scripts tracking user behaviour (such as mouse movements and keystrokes) are classified as behavioural biometrics scripts. This indicates that most scripts exfiltrating this information are not primarily intended for security purposes. In our analysis, we found that many of these scripts are associated with heatmaps or session replays, while a significant portion are web analytics scripts that also capture user behaviour data.

Potential Behavioural Biometrics In Study 1, we identified 25 websites with scripts that could not be clearly classified as either behavioural biometrics or web analytics. We labelled these as potentially behavioural biometrics scripts. Notably, 14 of these websites were Amazon domains from various regions. All Amazon websites deployed identical scripts that collected the same information and operated in the same manner. A similar pattern emerged in Study 2: out of the eight potentially behavioural biometrics scripts detected, seven belonged to Amazon websites.

The remaining potentially behavioural biometrics scripts in Study 1 were associated with Alibaba and two other unidentified vendors. The Alibaba scripts, in particular, did not provide sufficient evidence to classify them as analytic scripts. They featured the keywords `SecDev` and `SecuritySDK` in their names, suggesting that the scripts are used for non-malicious purposes. These Alibaba scripts were deployed across nine different websites in Study 1 and one website in Study 2.

VIII. Recommendations for Ethical Deployment

To mitigate privacy risks while harnessing the security benefits of behavioural biometrics, we propose a multi-faceted approach.

Disclosure Requirements: Companies should be legally obligated to explicitly disclose the use of behavioural biometrics in their privacy policies. Such disclosures must detail the types of data collected (e.g., keystroke dynamics, mouse movement patterns, touch pressure), the specific purposes for which the data is used (e.g., fraud detection, continuous authentication), and any third parties with whom the data is shared. This transparency not only builds user trust but also facilitates independent scrutiny of biometrics practices.

Obfuscation Restrictions: Regulatory bodies should impose restrictions on excessive obfuscation of behavioural biometrics scripts. This would allow third-party auditors and researchers to verify that biometrics systems adhere to declared practices and that no hidden data-sharing or tracking mechanisms exist. From what we noticed, the main logic happens in the backend. The scripts mostly collect data, which is not enough justification to obfuscate these scripts.

Client-Side Anonymization: To minimize privacy risks, raw interaction data should be processed locally on the client's device. For example, instead of transmitting exact keystroke timings or precise mouse coordinates, systems should compute anonymized features (such as entropy scores or aggregated metrics) that are less sensitive to individual identification while still being useful for authentication. We noticed that some scripts employ these methods.

Opt-Out Mechanisms: Users must be provided with clear, accessible controls to opt out of behavioural tracking.

Data Minimization and Retention: Systems should adhere to the principles of data minimization by collecting only behavioural necessary data for authentication. Any collected biometrics data should have clearly defined retention periods after which the data is securely deleted or anonymized.

Third-Party Audits. Regular audits by certified third-party organizations should be mandated. These audits would assess not only the security robustness of biometrics systems but also their compliance with privacy standards and ethical guidelines.

IX. Limitations and Future Directions

Our approach to collect a ground truth dataset of behavioural biometrics scripts was best effort (see Section V-A). It is possible that our manual analysis misclassified a script. Future work should validate these scripts and collect and curate larger sets of such scripts and websites that deploy them.

Our work presents a single snapshot in time. Future research should monitor adoption trends. Longitudinal studies can provide valuable insights into how behavioural biometrics systems evolve and their use on the web.

X. Conclusion

We bridge a critical gap in the detection and analysis of behavioural biometrics scripts by introducing BBAT, a scalable, automated solution for the large-scale detection and analysis of behavioural biometrics scripts. Deploying BBAT across 20K websites, 1K login pages, and over 1K e-commerce websites produced several noteworthy insights: First, behavioral biometrics scripts are used by at least 0.31% and potentially up to 0.5% of popular websites. Second, login pages are more likely to contain behavioural biometrics scripts. Third, many Shopify websites employ a script that we believe is behavioural biometrics on their checkout page. Fourth, we found 19 behavioural biometric vendors with associated scripts that are actively being used by their customers. Four of the vendors were found by our machine learning pipeline.

Behavioural biometrics represent a double-edged sword: their capacity to prevent fraud is substantial, yet their covert operation may break digital trust if misused. This paper advocates for a balance between enhanced security and user privacy by emphasizing transparency, accountability, and user empowerment. As authentication evolves from traditional passwords to behaviour-based mechanisms, it is imperative that regulators, researchers, and industry stakeholders vigilantly guard the boundary between improved security and company surveillance. This will ensure that behavioural biometrics technologies are integrated ethically and improve users' trust.

Acknowledgments

We thank the FoxHound maintainers for their help. This research was supported by Mastercard Canada and the National Cybersecurity Consortium.

References

[1] 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*, 2013, pp. 541–555.

[2] 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*, ser. CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 674–689. [Online]. Available: https://doi.org/10.1145/2660267.2660347

[3] 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)*, 2021, pp. 1143–1161.

[4] S. Boussaha, L. Hock, M. Bermejo, R. C. Rumín, Ángel Cuevas Rumín, D. Klein, M. Johns, L. Compagna, D. Antonioli, and T. Barber, "Fp-tracer: Fine-grained browser fingerprinting detection via taint-tracking and entropy-based thresholds," *Proc. Priv. Enhancing Technol.*, vol. 2024, no. 3, pp. 540–560, 2024. [Online]. Available: https://doi.org/10.56553/popets-2024-0092

[5] BuiltWith, "eCommerce Web Usage Distribution in the United States," https://trends.builtwith.com/shop/country/United-States.

[6] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, ser. NDSS 2019, Feb. 2019.

[7] D. Klein, T. Barber, S. Bensalim, B. Stock, and M. Johns, "Hand sanitizers in the wild: A large-scale study of custom javascript sanitizer functions," in *Proc. of the IEEE European Symposium on Security and Privacy*, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9797375

[8] Microsoft, "Clarity with every click," https://clarity.microsoft.com/.

[9] Fullstroy, "Session summaries make user insights actionable," https://www.fullstory.com/platform/session-replay/.

[10] Q. Metrics, "Session replay, the right way." https://www.quantummetric.com/platform/session-replay.

[11] A. Bara, "BBAT: Behavioural biometrics analysis tool," https://github.com/alexbara2000/BBAT.

[12] ——, "Scripts," https://github.com/alexbara2000/BBAT/tree/main/Results/ML.

[13] J. R. Mayer, ""any person... a pamphleteer:" internet anonymity in the age of web 2.0." [Online]. Available: http://arks.princeton.edu/ark:/88435/dsp01nc580n467

[14] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*, M. J. Atallah and N. J. Hopper, Eds. Springer, pp. 1–18.

[15] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of W2SP 2012*, M. Fredrikson, Ed. IEEE Computer Society, May 2012.

[16] 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*, ser. CCS '16. Association for Computing Machinery, pp. 1388–1401. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2978313

[17] A. Sjösten, S. Van Acker, and A. Sabelfeld, "Discovering browser extensions via web accessible resources," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, ser. CODASPY '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 329–336. [Online]. Available: https://doi.org/10.1145/3029806.3029820

[18] T. Unger, M. Mulazzani, D. Frühwirt, M. Huber, S. Schrittwieser, and E. Weippl, "SHPF: Enhancing http(s) session security with browser fingerprinting," in *2013 International Conference on Availability, Reliability and Security*, 2013, pp. 255–261.

[19] Ł. Olejnik, G. Acar, C. Castelluccia, and C. Diaz, "The leaking battery," in *Data Privacy Management, and Security Assurance*, J. Garcia-Alfaro, G. Navarro-Arribas, A. Aldini, F. Martinelli, and N. Suri, Eds. Cham: Springer International Publishing, 2016, pp. 254–263.

[20] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," *ACM Trans. Web*, vol. 14, no. 2, Apr. 2020. [Online]. Available: https://doi.org/10.1145/3386040

[21] A. Durey, P. Laperdrix, W. Rudametkin, and R. Rouvoy, "FP-Redemption: Studying browser fingerprinting adoption for the sake of web security," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 18th International Conference, DIMVA 2021, Virtual Event, July 14–16, 2021, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 237–257.

[22] A. Senol, A. Ukani, D. Cutler, and I. Bilogrevic, "The double edged sword: Identifying authentication pages and their fingerprinting behavior," in *Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1690–1701. [Online]. Available: https://doi.org/10.1145/3589334.3645493

[23] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, *Authentication by Keystroke Timing: Some Preliminary Results*. Santa Monica, CA: RAND Corporation, 1980.

[24] P. S. Teh, A. B. J. Teoh, and S. Yue, "A survey of keystroke dynamics biometrics," *ScientificWorldJournal*, vol. 2013, p. 408280, Nov. 2013.

[25] T. Samura and H. Nishimura, "Keystroke timing analysis for individual identification in japanese free text typing," in *2009 ICCAS-SICE*, 2009, pp. 3166–3170.

[26] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 3, pp. 165–179, 2007.

[27] P. A. Thomas and K. Preetha Mathew, "A broad review on non-intrusive active user authentication in biometrics," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 1, pp. 339–360, Jan. 2023.

[28] C. Shen, Z. Cai, X. Liu, X. Guan, and R. A. Maxion, "Mouseidentity: Modeling mouse-interaction behavior for a user verification system," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 5, pp. 734–748, 2016.

[29] S. Gupta, A. Buriro, and B. Crispo, "A chimerical dataset combining physiological and behavioral biometric traits for reliable user authentication on smart devices and ecosystems," *Data in Brief*, vol. 28, p. 104924, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235234091931279X

[30] X. Lin, P. Ilia, S. Solanki, and J. Polakis, "Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1651–1668. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/lin-xu

[31] GlobeNewswire, "Denuvo unveils unbotify technology for bot detection," https://www.globenewswire.com/news-release/2023/03/21/2631470/0/en/Denuvo-unveils-Unbotify-technology-for-bot-detection.html.

[32] DeepSource, "Taint analysis," https://deepsource.com/glossary/taint-analysis.

[33] A. Sjösten, D. Hedin, and A. Sabelfeld, "EssentialFP: Exposing the essence of browser fingerprinting," in *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*. IEEE, 2021, pp. 32–48. [Online]. Available: https://doi.org/10.1109/EuroSPW54576.2021.00011

[34] R. Kanyal and S. R. Sarangi, "PanoptiChrome: A modern in-browser taint analysis framework," in *Proceedings of the ACM on Web Conference 2024*. ACM, pp. 1914–1922. [Online]. Available: https://dl.acm.org/doi/10.1145/3589334.3645699

[35] Q. Chen and A. Kapravelos, "Mystique: Uncovering information leakage from browser extensions," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. Association for Computing Machinery, pp. 1687–1700. [Online]. Available: https://dl.acm.org/doi/10.1145/3243734.3243823

[36] S. Lekies, B. Stock, and M. Johns, "25 million flows later: large-scale detection of DOM-based XSS," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. Association for Computing Machinery, pp. 1193–1204. [Online]. Available: https://dl.acm.org/doi/10.1145/2508859.2516703

[37] R. Karim, F. Tip, A. Sochůrková, and K. Sen, "Platform-independent dynamic taint analysis for JavaScript," vol. 46, no. 12, pp. 1364–1379, IEEE Transactions on Software Engineering. [Online]. Available: https://ieeexplore.ieee.org/document/8511058

[38] M. Tran, X. Dong, Z. Liang, and X. Jiang, "Tracking the trackers: Fast and scalable dynamic analysis of web content for privacy violations," in *Applied Cryptography and Network Security*, F. Bao, P. Samarati, and J. Zhou, Eds. Springer, pp. 418–435.

[39] L. Jannett, M. Westers, T. Wich, C. Mainka, A. Mayer, and V. Mladenov, "SoK: SSO-MONITOR - the current state and future research directions in single sign-on security measurements," in *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*. IEEE, pp. 173–192. [Online]. Available: https://ieeexplore.ieee.org/document/10628996/

[40] T. Innocenti, M. Golinelli, K. Onarlioglu, A. Mirheidari, B. Crispo, and E. Kirda, "OAuth 2.0 redirect uri validation falls short, literally," in *Proceedings of the 39th Annual Computer Security Applications Conference*, ser. ACSAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 256–267. [Online]. Available: https://doi.org/10.1145/3627106.3627140

[41] Aftership, "Top 1000 Shopify Stores 2024," https://www.aftership.com/store-list/top-1000-shopify-stores.

[42] D. Klein, M. Musch, T. Barber, M. Kopmann, and M. Johns, "Accept all exploits: Exploring the security impact of cookie banners," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3564625.3564647

[43] A. Bara, "Checkout crawler," https://github.com/alexbara2000/checkout-crawler.

[44] berstend, "puppeteer extra plugin stealth," https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra-plugin-stealth.

[45] BuiltWith, "Canadian Top Ranked eCommerce Websites," https://builtwith.com/top-sites/Canada/eCommerce.

[46] NerdyData, "NerdyData," https://www.nerdydata.com/.

[47] Yofi, "Yofi.ai," https://www.yofi.ai.

[48] CHEQ, "Cheq," https://cheq.ai/dg-brand-gtm-security-platform/.

[49] Group-IB, "Group-ib," https://www.group-ib.com.

[50] Untarget.ai, "Untarget.ai," https://untarget.ai.

Appendix A
User Simulation

Our crawler is configured to execute a predefined sequence of events consistently across all websites, ensuring uniformity during both the ground truth data collection and the large-scale analysis phases. The sequence of simulated events is as follows:

1) **Click Events:** Four click events are triggered at random locations on the webpage, with a 250ms delay between each click.
2) **Double Click Event:** A single double-click event is executed.
3) **Mouse Movement:** The cursor moves from one random location to another.
4) **Mouse Down and Mouse Up:** A mouse press-and-release action is performed.
5) **Keyboard Events:** Key presses, along with key-down and key-up actions, are simulated.
6) **Copy-Paste Action:** A copy-and-paste event is triggered.
7) **Scrolling:** The crawler simulates both upward and downward scrolling.
8) **Screen Resize and Orientation Change:** The screen is resized to a random dimension, followed by an orientation change.
9) **Touch Events:** A multi-touch interaction is simulated by triggering touch events at random positions on the page.

Appendix B
Hyperparameters

Hyperparameters used for our machine-learning models:

- **Random Forest**: Max depth = None, Min samples per leaf = 1, Min samples per split = 2, Number of estimators = 50.
- **SVM**: Regularization parameter $C = 10$, Gamma = `auto`, Kernel = `linear`.
- **KNN**: Metric = `euclidean`, Number of neighbors = 7, Weights = `distance`.
- **XGBoost**: Learning rate = 0.2, Max depth = 5, Number of estimators = 50.
- **Neural Network**: Activation function = `relu`, 2 hidden layer with 50 neurons in each, Solver = `adam`.

Appendix C
Taint Sources and Sinks

Table IV shows all the sinks and sources we used for our dynamic taint analysis.

TABLE IV: Sinks and Sources used in Dynamic Taint Analysis.

Sinks	**Sources**
element.after	KeyboardEvent.charCode
element.before	KeyboardEvent.keyCode
EventSource	KeyboardEvent.key
Function.ctor	KeyboardEvent.altKey
Range.createContextualFragment(fragment)	KeyboardEvent.ctrlKey
WebSocket	KeyboardEvent.shiftKey
WebSocket.send	KeyboardEvent.metaKey
XMLHttpRequest.open(password)	KeyboardEvent.location
XMLHttpRequest.open(url)	KeyboardEvent.repeat
XMLHttpRequest.open(username)	KeyboardEvent.isComposing
XMLHttpRequest.send	MouseEvent.screenX
XMLHttpRequest.setRequestHeader(name)	MouseEvent.screenY
XMLHttpRequest.setRequestHeader(value)	MouseEvent.pageX
a.href	MouseEvent.pageY
area.href	MouseEvent.clientX
document.cookie	MouseEvent.clientY
document.writeln	MouseEvent.x
document.write	MouseEvent.y
element.style	MouseEvent.offsetX
embed.src	MouseEvent.offsetY
eval	MouseEvent.ctrlKey
eventHandler	MouseEvent.shiftKey
fetch.body	MouseEvent.region
fetch.url	MouseEvent.movementX
form.action	MouseEvent.movementY
iframe.src	MouseScrollEvent.axis
iframe.srcdoc	PointerEvent.pointerId
img.src	PointerEvent.width
img.srcset	PointerEvent.height
innerHTML	PointerEvent.pressure
insertAdjacentHTML	PointerEvent.tangentialPressure
insertAdjacentText	PointerEvent.tiltX
localStorage.setItem	PointerEvent.tiltY
localStorage.setItem(key)	PointerEvent.twist
location.assign	PointerEvent.isPrimary
location.hash	Touch.identifier
location.host	Touch.screenX
location.href	Touch.screenY
location.pathname	Touch.clientX
location.port	Touch.clientY
location.protocol	Touch.pageX
location.replace	Touch.pageY
location.search	Touch.radiusX
media.src	Touch.radiusY
MessagePort.PostMessage	Touch.rotationAngle
navigator.sendBeacon(body)	Touch.force
navigator.sendBeacon(url)	TouchEvent.altKey

Sinks	**Sources**
object.data	TouchEvent.metaKey
outerHTML	TouchEvent.ctrlKey
script.innerHTML	TouchEvent.shiftKey
script.src	
script.text	
script.textContent	
sessionStorage.setItem	
sessionStorage.setItem(key)	
setInterval	
setTimeout	
document.source	
srcset	
track.src	
window.open	
window.postMessage	
TypedArray.setElem	
TypedArray.setFromArray	
Element.append	
BrowsingContext.PostMessage	
TextEncoder.encode	
HTMLInputElement.setValue	
Worker.PostMessage	