Is Your Wallet Snitching On You? An Analysis on the Privacy Implications of Web3

Christof Ferreira Torres
ETH Zurich

Fiona Willi
ETH Zurich

Shweta Shinde
ETH Zurich

Abstract

With the recent hype around the Metaverse and NFTs, Web3 is getting more and more popular. The goal of Web3 is to decentralize the web via decentralized applications. Wallets play a crucial role as they act as an interface between these applications and the user. Wallets such as MetaMask are being used by millions of users nowadays. Unfortunately, Web3 is often advertised as more secure and private. However, decentralized applications as well as wallets are based on traditional technologies, which are not designed with privacy of users in mind.

In this paper, we analyze the privacy implications that Web3 technologies such as decentralized applications and wallets have on users. To this end, we build a framework that measures exposure of wallet information. First, we study whether information about installed wallets is being used to track users online. We analyze the top 100K websites and find evidence of 1,325 websites running scripts that probe whether users have wallets installed in their browser. Second, we measure whether decentralized applications and wallets leak the user's unique wallet address to third-parties. We intercept the traffic of 616 decentralized applications and 100 wallets and find over 2000 leaks across 211 applications and more than 300 leaks across 13 wallets. Our study shows that Web3 poses a threat to users' privacy and requires new designs towards more privacy-aware wallet architectures.

1 Introduction

Web3 has gained tremendous adoption over the past few years. This is mainly fueled by the rise of decentralized applications (DApps) such as the Metaverse, NFTs, and decentralized finance (DeFi). DappRadar.com currently lists over 13,000 DApps across various blockchain platforms [17]. A report from 2022 states that NFTs generated 12 Billion USD in trades and that DeFi even reached a value of 127 Billion USD in total value locked on Ethereum [67]. The promise of Web3 is the ability to run traditional applications in a decentralized way, thus assuring better transparency and privacy. An important aspect of such decentralized infrastructure are wallets, which act as an interface between decentralized applications and the user. Wallets enable users not only to perform common blockchain operations such as managing their credentials (i.e., public and private key pairs) or signing of transactions, but also operations on DApps such as trading tokens or buying NFTs. All of these operations are provided to the user via a convenient and easy to use interface. There are several wallet operators that act as intermediaries and interact with decentralized apps on the behalf of the user. MetaMask is currently one of the most popular wallet operators with over 10 Million active users [12].

While built with the goal of better transparency and privacy, decentralized applications as well as wallets are still based on traditional web technologies, which are prone to privacy issues. Wallets, in particular have access to sensitive user information and are therefore a rich target for attacks. To make matters worse, wallet operators often use centralized providers by default to retrieve information from the blockchain, making them a single point of failure and allowing providers to easily track user activity across DApps. For example, Infura's recent privacy policy update mentions that IP addresses and wallet addresses of users will be collected [15]. Since Infura is the default blockchain provider of MetaMask, this means that Infura is capable of linking wallet addresses with IP addresses of millions of users. While users might accept trusting the wallet operators, they may not realize to what extent they are exposing their wallet information to third-parties.

Wallets operate by injecting a wallet object into the DOM of every website the user visits. This facilitates the interaction between DApps and wallets. DApps can then simply use JavaScript to access wallet information. However, the browser does not take any particular measures to safeguard the wallet object. Thus, any malicious website, third-party, or browser extension can read this object or use this object to trick users into approving malicious actions (e.g., send assets to an attacker-controlled address). While these attack vectors have been exploited in the traditional web in the past, their prevalence in the context of Web3 is yet unclear.

In this paper, we investigate whether wallet extensions are being used to track users online and whether DApps as well as wallet extensions leak the user's wallet address to third-parties. To answer this question systematically, we build a framework that is capable of simulating wallet objects and monitoring access to these objects. Hence, if a website checks the presence of a wallet object in conjunction to several other JavaScript attributes, we deem it as a tracking attempt to fingerprint the user. Moreover, our framework is also capable of automatically interacting with DApps as well as wallets and intercepting any cookies as well as requests made via HTTP and WebSockets. We identify a DApp or wallet to leak the user's wallet address if we find that any of the intercepted cookies or requests include the user's wallet address.

Results. We report three main findings. First, of the 100K websites that we analyzed, $1,325$ of them track users via wallet objects either directly or via third-party scripts. Second, of the $1,572$ DApps that we analyzed, 211 of them leak the user's wallet address to third-parties such as blockchain providers or tracking and analytics platforms. Lastly, of the 100 wallets that we analyzed, 13 wallets leak the user's wallet address to third-parties. All together wallets include over 137 unique third-parties, thereby giving third-parties access to sensitive user information. In summary, our investigation shows that the existing wallet infrastructure is not in favor of users' privacy. Websites are abusing wallets to fingerprint users online, and DApps as well as wallets leak the user's wallet address to third-parties.

Ethics Considerations. Throughout our analysis, we took adequate measures to avoid overloading the websites (e.g., limited ourselves to the landing page). We have informed the websites and third-parties about potentially unintentional data collection from their side.

Contributions. We summarize our contributions as follows:

- We present the first study that systematically measures the prevalence of websites and third-party scripts that use wallet information to track users online. We found evidence that 1,325 websites out of the top 100K websites probe their users for wallets.

- We conduct the first large-scale measurement to assess the leakage of wallet addresses on various DApps and wallets. We find that 211 out of 616 DApps and 13 out of 100 wallets leak the user's wallet address to third-parties.

- We measure the efficacy of 5 popular blocklists and observe that when all combined 44% of the third-parties would not be blocked.

2 Background

We provide background on Ethereum, decentralized applications, wallets, and privacy concerns that might arise when combining all these technologies together.

2.1 Ethereum

Ethereum is a blockchain or distributed ledger where transactions are grouped into batches of blocks and where each block points to its previous block via a cryptographic hash. Blockchains are typically maintained by a distributed peer-to-peer network, which is responsible for broadcasting transactions, appending new blocks, providing access to stored data, and executing smart contracts. Smart contracts are programs that are deployed and executed across a blockchain. As of January 2023, Ethereum has a market capitalization of over 180 billion USD [14], making it the most popular blockchain technology that offers Turing-complete smart contract capabilities. Ethereum peers (i.e., nodes) may expose a JSON-RPC interface [27], which defines an API that users or applications can use to interact with the blockchain (e.g., sending transactions or querying the state of a smart contract). Similar to other blockchains, Ethereum has its own native cryptocurrency (i.e., Ether), that enables users to transfer value across accounts and to pay for transactions. However, unlike Bitcoin for example, Ethereum follows an account-based model. The idea is similar to traditional bank accounts, where users own an account number and other users may transfer currency to this account number. In Ethereum, users do not own an account number, instead they own an account address, which is a unique 160-bit long hexadecimal string. However, similar to a bank account number, addresses act as a unique identifier that can be used to link transactions back to users and which should therefore be shared only with trusted parties.

2.2 Decentralized Applications

Decentralized Applications, also known as DApps, are applications that are accessible via the web, but where either all or some of the parts are hosted on decentralized platforms. However, for ease-of-use, availability requirements, and compatibility with existing technologies (e.g., DNS, HTTP client-server model, etc.), in most cases the user interface (UI) of DApps is hosted on a centralized web hosting service such as AWS. Only parts of the business logic are decentralized via the use of smart contracts. There are a number of different use cases for DApps, ranging from gambling platforms and online games (e.g., CryptoKitties), to decentralized marketplaces and exchanges (e.g., Uniswap). DappRadar currently lists over $3,000$ DApps for the Ethereum blockchain alone [17]. However, to be able to interact with DApps, users are required to use a wallet, which acts as a bridge between the DApp and the user's identity on the blockchain.

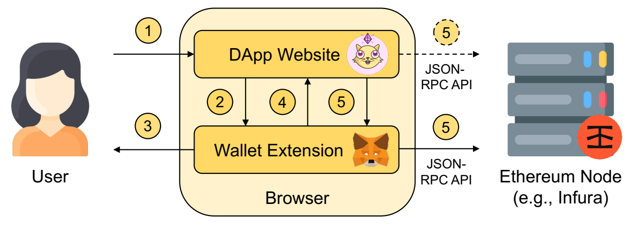

Figure 1: Conceptual flow between a user, a wallet, a DApp and an Ethereum node. ① User visits a DApp website and clicks on "Connect Wallet" button located on the website, ② website requests user's wallet address by calling the wallet extension's Ethereum Provider API [49], ③ wallet extension asks user for permission and user grants it via the wallet's UI, ④ wallet returns wallet address to DApp website, ⑤ website can now interact with the Ethereum blockchain either directly or through the wallet extension which is connected to an Ethereum node via JSON-RPC API [27].

2.3 Wallets

Users typically manage their accounts and cryptocurrency via a wallet. A popular choice are wallets in the form of a browser extension. A browser extension is a software module that users can install in their browser to enhance their browsing experience. Browser extensions have the capability to modify the Document Object Model (DOM) of websites and enjoy access to privileged browser APIs such as browsing history.

MetaMask [47] currently is the most popular wallet extension for Ethereum with over 10 Million downloads on Google Chrome's web store [12]. Wallet extensions such as MetaMask inject a Web3 object into the DOM of any website that the user visits, regardless of whether the website is a DApp or not. Specifically, MetaMask adds a new object called `ethereum` to the existing `window` object, which exposes the Ethereum Provider API [49]. The API enables DApps to interact via JavaScript with the Ethereum blockchain as well as the user's wallet. For example, DApps can call unprivileged properties such as `window.ethereum.isMetaMask`, which will return `true` if MetaMask is installed, but also privileged properties such as `window.ethereum.selectedAddress`, which will return the user's wallet address to the DApp. Wallets are required to ask the user for prior permission and the user needs to grant it before a DApp is able to access privileged properties such as the user's wallet address.

Figure 1 depicts the conceptual flow of a user interacting with a DApp. A user starts by visiting the DApp's website. DApps usually expose a visual UI button on their website, which users must click if they wish to "connect" their wallet to the DApp (i.e., grant DApp access to their wallet). The DApp will then request permission to the wallet via that wallet's injected Ethereum Provider API. The wallet will display a popup to the user asking if it wants to grant permission to the DApp. In case the user grants the access, the wallet returns the requested information back to the DApp. Note that while a subset of the Ethereum Provider API is handled directly by the wallet extension (e.g., signing of transactions), another subset (e.g., retrieving latest block number) is simply forwarded to an Ethereum node (e.g., Infura [35]) via JSON-RPC. Also note that a DApp is not required to rely on a wallet extension to interact with the blockchain. A DApp can simply talk directly to a blockchain node. In fact, many DApps limit their interaction with the wallet extension to the bare minimum of only requesting the user's wallet address and the signing of transactions.

2.4 Privacy Concerns

Tracking is omnipresent on the web. Users are constantly being tracked across websites for purposes of analytics or targeted advertising, either via explicit (e.g., cookies) or implicit (e.g., browser fingerprinting) information. In the past, third-party cookies have been a popular way to track users across the web [45], but most modern browsers nowadays block third-party cookies by default [50, 64]. A popular alternative is browser fingerprinting [38]. The idea is to uniquely identify users based on differences of their browser's configuration (e.g., fonts, screen resolution, plugins, etc.). A fingerprint is generated by combining properties that are exposed to a website via JavaScript. As opposed to cookies, which are stateful, browser fingerprinting is stateless and thus difficult to mitigate without breaking usability (i.e., disabling JavaScript) [53].

Since DApps are developed using traditional web technologies, many DApps also include several third-party tracking scripts. DApps cannot sign transactions without the consent of the user. However, once a user has connected its wallet to a DApp, all third-party scripts embedded within the DApp can access the injected Web3 object via JavaScript. This grants third-party scripts access to sensitive information such as the user's account address or balance, without requiring prior consent of the user. Additionally, without being connected to a wallet, third-party scripts can check for the existence of a Web3 object in the DOM and thus infer that a user owns cryptocurrency and possibly which cryptocurrency and which wallet. This information can be leveraged to augment existing browser fingerprints as it adds additional bits of entropy [23].

While blockchains do provide some level of anonymity (i.e., pseudonymity), they do not provide full anonymity. All transactions from and to a given account can easily be linked to a user's account address, but not necessarily to its real-world identity. Yet, third-party scripts pose a threat to a user's anonymity since they also have access to a user's IP address and thus can potentially link multiple wallet addresses to their respective IP address [41]. In fact, Infura's recent privacy policy update has raised several concerns among the community as it states that wallet addresses and IP addresses will be collected [15].

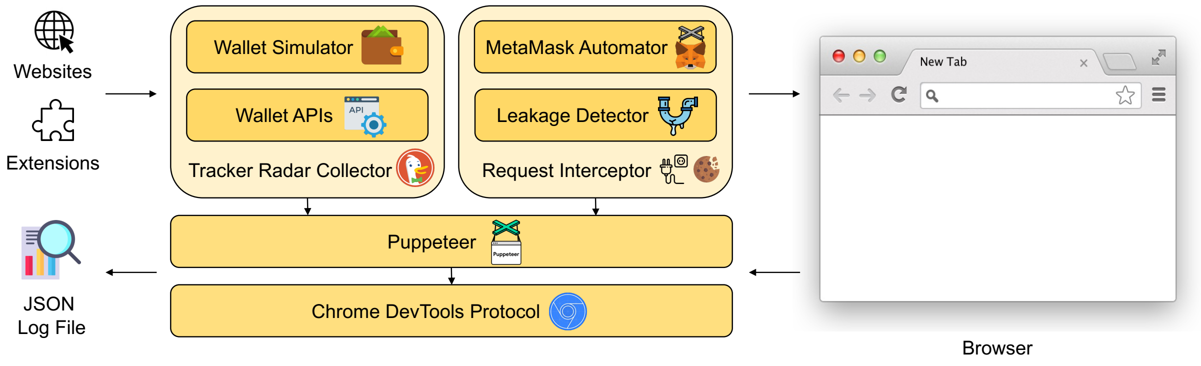

Figure 2: Overview of our measurement framework and its individual components. Our framework takes as input a list of websites or extensions and logs all results in a JSON file. We develop a wallet simulator and integrate it into DuckDuckGo's Tracker Radar Collector (TRC) [19]. We leverage TRC to crawl websites and log any calls to Wallet APIs. We also develop a Request Interceptor that captures cookies as well as HTTP and WebSocket traffic from websites and extensions. Moreover, we create a MetaMask automator that automates the task of setting up MetaMask [47] as well as connecting to DApps, and integrate a leakage detector to find wallet address leaks. Both, TRC and our Request Interceptor are based on Puppeteer [33], which uses the Chrome DevTools Protocol [31] to interact with a browser.

3 Methodology

Next, we describe our approach for detecting Web3-based browser fingerprinting and identifying wallet address leakage across DApps and wallet extensions. A high-level overview of our measurement framework is depicted in Figure 2.

3.1 Web3-Based Browser Fingerprinting

Browser fingerprinting is a prevalent online tracking technique [25, 30, 38, 53, 61]. Our goal is to find evidence of whether websites or third-party scripts are leveraging any of the JavaScript properties that wallet extensions expose, to track users on the web. For that purpose, we use DuckDuckGo's Tracker Radar Collector (TRC) [19] to crawl popular websites and measure their behavior. TRC is a crawler that is designed for large-scale web measurements. It is modular and leverages multi-threading to speed up crawling. It uses Puppeteer [33] under the hood, which is a library that allows developers to control Chromium-based browsers for automation and testing purposes via the Chrome DevTools Protocol [31]. This gives TRC the capability to intercept network requests, read cookies, and instrument JavaScript calls.

3.1.1 Detecting Wallet API Calls

In contrast to OpenWPM [25], another popular crawler which uses inline instrumentation by overriding JavaScript functions and objects with getters, TRC uses the Chrome DevTools Protocol to set breakpoints in the JavaScript engine. These breakpoints cannot be detected by websites and are only triggered when a certain function is called or property is accessed. Whenever the debugger hits any of the configured breakpoints, TRC will collect the JavaScript stack trace (e.g., filename, line number, etc.) and other metadata about the property access or function invocation and store it in a JSON file.

Wallet	TRC Breakpoint	Simulated JavaScript Property
MetaMask	`window.ethereum`	`isMetaMask: true` [48]
Coinbase	`window.ethereum`	`isCoinbaseWallet: true` [13]
Binance	`window.BinanceChain`	`chainId: "0x38"` [6]
Phantom	`window.solana`	`isPhantom: true` [52]
Nami	`window.cardano`	`nami.name: "Nami Wallet"` [9]

Table 1: List of breakpoints added to TRC and JavaScript properties simulated by our wallet simulator.

We set breakpoints for five popular wallets (see Table 1). We started by first only hooking the `window.ethereum` object, and added the other wallet API hooks after manually checking reported scripts during initial test runs. Moreover, after our final crawl we performed a manual inspection of all several scripts and were not able to find any other wallet APIs, which gives us confidence that the four breakpoints in Table 1 are sufficient. However, breakpoints only get triggered if the object actually exists in the DOM. For example, to detect whether websites are trying to identify if MetaMask is installed, we set a breakpoint to be triggered whenever a script accesses the `window.ethereum` object. Thus, in the case of MetaMask, the object `window.ethereum` has to be injected into the DOM for it to be detected by our breakpoint. We therefore simulate each wallet listed in Table 1 by injecting (prior to any script execution) wallet specific properties into the DOM. For instance, to simulate MetaMask we inject the property `window.ethereum.isMetaMask` into the DOM and set it to `true` as defined in MetaMask's documentation [48]. This allows us to hook future accesses by the website, thus catching any access to the object beyond the simulated property. In other words, by hooking `window.ethereum` via the property `window.ethereum.isMetaMask`, we will also be able to detect access to, for instance, `window.ethereum.request`.

3.1.2 Identifying Fingerprinting Behavior

The idea behind browser fingerprinting is to collect a large amount of diverse but stable information about a user's browser configuration, such that when combined together, enough entropy is provided to generate a unique fingerprint that identifies the same user across different sessions and websites. We leverage a similar approach as proposed in [59] to detect fingerprinting behavior in Android applications, and adapt it for JavaScript. TRC already provides a curated list of JavaScript properties and functions, that websites are known to leverage, to generate browser fingerprints [20]. We group each property and function call into one of 22 self-defined categories (see Appendix A). A script is marked as a fingerprinting script if it calls JavaScript properties and functions belonging to at least 10 different categories, where at least one of the categories must belong to a list of 8 explicit browser fingerprinting categories. We tried values of 5, 10, 15, and 20 during earlier experiments with a small set of identified fingerprinting scripts and achieved the best accuracy when using 10 as threshold. Explicit browser fingerprinting categories include JavaScript properties and functions that are heavily used for fingerprinting purposes (e.g., CanvasRenderingContext2D, WebGLRenderingContext, AudioBuffer).

3.2 Wallet Address Leakage

Nothing prohibits DApps or wallet extensions from sharing a user's wallet address with third-parties. This sharing can either happen with or without the knowledge of the DApp or wallet extension. Our goal is to measure whether, how, and with whom DApps and wallet extensions share wallet addresses. To that end, we developed an automator for MetaMask (see Figure 2) that not only automatically installs and sets up MetaMask when visiting a DApp, but also automatically tries to connect MetaMask to the Dapp. Once connected to a DApp or a wallet extension is installed, our request interceptor will intercept any outgoing traffic as well as cookies and search for wallet address leaks.

3.2.1 Connecting MetaMask to DApps

For a DApp to be able to leak a user's wallet address, the user needs to have set up a wallet and connected its wallet to the DApp. As we do not want to repeat this step manually for thousands of DApps, we develop a component called MetaMask automator. First the automator sets up MetaMask. This is done even before visiting the website of the DApp. Our automator starts by launching a fresh instance of a browser and installing MetaMask's wallet extension. Afterwards, it launches MetaMask's UI in a new browser page and automatically clicks on the button to import an existing wallet. Browser extensions, including MetaMask, contain UIs that are essentially HTML pages with JavaScript code. Our automator leverages Puppeteer to extract all HTML elements (e.g., buttons, input fields, etc.) from MetaMask's UI. Puppeteer also provides functions that allows our automator to interact with the HTML elements such as clicking on buttons or typing in text into input fields. Once the "import wallet UI" has loaded, our automator will read a fixed passphrase and password from a file and automatically type in the passphrase and password into MetaMask's UI to import the wallet and finish setting up MetaMask with a wallet address that we control.

After setting up the wallet, our automator visits the DApp's website. Once loaded, it searches for a "connect" button by scanning the HTML of the DApp's website for elements that contain keywords such as *"Connect Wallet"*, *"Sign In"*, *"Account"*, etc. We leveraged the 78 DApps by Winter et al. [66] to extract a list of common keywords for connect buttons (see Appendix B). Afterwards, the automator tries to perform a click on every element that it found. It detects the connect button if it finds an element where the click succeeds. Once the automator finds the connect button, it searches for a MetaMask button. Often DApps allow users to connect via different wallets and therefore they let users select which wallet they want to use. The automator finds the MetaMask button by scanning the HTML for elements containing keywords such as *"MetaMask"* or *"Browser Wallet"* (see Appendix B). Some DApps require users to click on a checkbox to agree to the terms and conditions before being able to connect. Our automator handles this case by searching the HTML for checkboxes and selecting them before clicking on any button. After successfully clicking the MetaMask button, a popup window will show up asking the user for permission to connect. Our automator intercepts this popup window and automatically clicks on the "confirm" button to finalize the connection request and give permission to the DApp to access our wallet's information. However, in some cases our automator might not be able to find the MetaMask button via text search because either the DApp uses an image or the text is not detectable. Thus, whenever our automator does not find any MetaMask button, it infers the dimensions of the browser's window and tries to perform hard-coded blind clicks on various offsets starting off from the middle of the window (e.g., 100 pixels to the bottom right, 50 pixels to the top left, etc.).

3.2.2 Intercepting Outgoing Traffic and Cookies

There are multiple ways in which DApps or wallet extensions can exfiltrate wallet addresses. Previous works have only focused on intercepting HTTP GET requests [66]. However, in our work we also intercept HTTP POST requests since DApps and wallet extensions may also leak information via the post body. Moreover, we also intercept WebSocket payloads. WebSockets became a popular alternative to HTTP polling due to their high efficiency (e.g., low latency and fast transmission). They establish a long-lived connection between the DApp or wallet extension and the server. While WebSockets allow for messages to be sent in a bi-directional manner, we are only interested in intercepting outgoing messages (i.e., requests going from the DApp or wallet extension to the server). To that end, we leverage the capabilities of the Chrome DevTools Protocol to intercept network requests to capture any HTTP GET and POST requests as well as outgoing WebSocket messages. Finally, cookies can also be used to exfiltrate wallet addresses. These can either be set by the server or by the client via JavaScript. We therefore capture cookies that are set via the response headers of HTTP requests and also leverage the capability of the Chrome DevTools Protocol to dump any cookies that were set via JavaScript.

3.2.3 Identifying Wallet Address Leaks

We identify wallet address leaks in websites and browser extensions by checking if any of the intercepted traffic (i.e., cookies, HTTP, and WebSockets) contains the wallet address in plain text. More specifically, for cookies, we check whether the value or name of the cookie contains the wallet address. For HTTP GET requests, we check whether the URL of the request contains the wallet address within the GET parameters. For HTTP POST requests, we check whether the post body contains the wallet address. Finally, for WebSockets, we check whether the payload contains the wallet address. However, checking for the wallet address in plain text is not sufficient. Prior studies [24, 55, 57] have shown that many third-parties often obfuscate their leaks by encoding or hashing them. Identifying obfuscated leaks is a challenging task, which often boils down to a brute-force search. We employ Senol et al.'s [55] technique, borrowed from Englehardt et al.'s [24] method, to identify email addresses in obfuscated strings. The method consists of searching for a variety of encodings and hashes within strings, by precomputing a set of strings, which contains all possible encodings (e.g., Base64, URL encoding, LZstring, etc.) and hashes (e.g., MD5, SHA256, MurmurHash3, etc.) of the wallet address. Afterwards, the contents of cookies, HTTP requests, and WebSocket payloads are split into multiple strings by potential separator characters (e.g., '=', '&', etc.) and compared with the strings contained in the precomputed set. This process is repeated until a level of three layers of encodings and decodings is reached.

4 Measurements

We describe our experimental setup and present the results of our large-scale measurement to detect web3-based user tracking and wallet address leakage[1].

Category	DApps	Valid URLs
Collectibles	615	533
DeFi	360	339
Games	291	186
Other	220	158
Marketplaces	97	87
High Risk	149	85
Exchanges	87	80
Gambling	145	74
Social	34	30
Total	**1,998**	**1,572**

Table 2: DApps crawled from DappRadar.com.

4.1 Experimental Setup

We ran all our experiments on a desktop machine with 10 cores and 32GB of RAM. Moreover, we used Chromium version 108.0.5351.0 as our browser and our automator was build based on MetaMask version 10.22.2 for Google Chrome.

Browser Fingerprinting. We measured browser fingerprinting using the top 1 Million Tranco [40] websites as of November 8th, 2022.[2] However, Tranco only provides domains and not URLs. Therefore, we tried matching Tranco domains to URLs using Google's Chrome User Experience (CrUX) Report [11] of November 2022. Whenever a domain did not match any URL contained in CrUX report, we tried inserting the prefixes `http(s)://` and `http(s)://www.` in front of the domains (prioritizing the prefix `https://www.`) and checked whether these were accessible (i.e., got an HTTP response). We skipped domains that were neither accessible nor contained in the CrUX report. We started with the top websites (i.e., highest rank to lowest rank) and repeated this process until we had a list of the top 100K accessible websites. For each website, we limited the maximum crawl duration to 60 seconds and only visited the landing page.

Wallet Address Leakage. We measured wallet address leakage using three different datasets. The first dataset consists of 66 DeFi websites from Winter et al.'s study [66]. The second dataset consists of 1,998 DApps that we crawled from DappRadar.com's top Ethereum Dapps [17]. Table 2 provides an overview of the number of DApps per category. Note that not all URLs listed on DAppRadar.com are valid. For instance, many URLs in the category collectibles are simply pointing to a collection on `opensea.io`. Moreover, some of the URLs are not accessible. We filtered out these URLs and were left with $1,572$ DApps with valid URLs (see Table 2), which we

[1]Our framework and data are publicly available at: https://github.com/christoftorres/Web3-Privacy.

[2]Available at: https://tranco-list.eu/list/6JXYX/1000000.

Rank	Website	Script Domain	Wallet API
74	**tiktok.com**	ttwstatic.com	All
96	**nytimes.com**	googletagmanager.com	`window.ethereum`
160	**xhamster.com**	xhcdn.com	`window.ethereum` `window.BinanceChain` `window.solana`
185	**tiktokv.com**	ttwstatic.com	All
224	**tinyurl.com**	tinyurl.com	All
270	**cnbc.com**	googlesyndication.com	`window.BinanceChain`
359	**mega.co.nz**	mega.io	`window.ethereum`
381	**weather.com**	taboola.com	`window.ethereum`
400	**mega.nz**	mega.io	`window.ethereum`
481	**pexels.com**	pexels.com	All

Table 3: Top 10 most ranked websites calling wallet APIs.

Wallet APIs Combinations	Scripts
`window.ethereum`	210
`window.solana`, `window.ethereum`	13
`window.solana`	11
`window.ethereum`, `window.BinanceChain`	8
`window.BinanceChain`	5
`window.cardano`	4
`window.ethereum`, `window.solana`, `window.BinanceChain`	3
`window.ethereum`, `window.solana`	2
`window.ethereum`, `window.BinanceChain`, `window.solana`	2
`window.ethereum`, `window.cardano`, `window.solana`	1
`window.BinanceChain`, `window.ethereum`	1

Table 4: Observed combinations of explicit wallet API calls.

then crawled during our experiment. Finally, the third dataset consists of 100 popular wallet extensions that we downloaded from Google's Chrome Web Store [32]. We installed and set up each wallet extension manually using separate browser profiles for reproducibility. We stored the password and address of each wallet extension in a separate file such that we can afterwards search for the intercepted requests for wallet address leakage. For each DApp website that we crawled, we limited the maximum crawl duration to 30 seconds and only visited the landing page. To measure wallet address leakage on wallet extensions, we wrote a script to randomly click on 10 clickable HTML elements. The interaction with the wallet extension either stops after 10 elements have been clicked or if 60 seconds have passed.

4.2 Web3-Based Browser Fingerprinting

Wallet API Calls. TRC was able to crawl 96,905 out of 100K websites successfully (i.e., 96.91%). We found 1,114 unique scripts on 1,325 websites which made in total 1,517 JavaScript calls to at least one wallet APIs listed in Table 1. Table 3 lists the top 10 most ranked websites which we found to call at least one wallet API. This list includes websites with millions of daily users such as TikTok and the New York Times. Interestingly, websites such as TikTok called

```
...
f = Math.trunc((new Date).getTimezoneOffset() / -60),
m = Boolean(window.web3 || window.ethereum),
...
return {
  ...
  isAdBlock: n,
  isMetaMaskActive: m,
  ...
}
...
```

Figure 3: Code snippet accessing `window.ethereum` from `https://js.wpadmngr.com/static/adManager.m.js`.

all of our wallet APIs. After inspecting their code we found that these websites detect whether objects were added to the DOM. We checked whether this only occurs via our wallet simulator or if it also happens when visiting TikTok with MetaMask installed. Our check revealed the same results. This is because MetaMask and any other wallet will, similar to our wallet simulator, inject a new Web3 object into the DOM. This will be detectable by those websites and used for either analytical or tracking purposes. We therefore differentiate between explicit calls and implicit calls, where explicit means that a script includes an explicit call to a wallet API in their code and where implicit means that a script implicitly calls a wallet API when searching for new objects that were added into the DOM. In our experiments, we found that browser fingerprinting scripts (see Table 6) often enumerate the entirety of the window object using, for example, `Object.getOwnPropertyNames(window)` to create a unique fingerprint and thereby implicitly calls a wallet API as it is often part of the window object. We found 249 scripts performing explicit calls (22%) and 866 scripts performing implicit calls (78%). Table 4 lists all combinations that we observed of explicit wallet API calls. We observed in total 11 combinations, where a simple call to `window.ethereum` was the most popular call with 210 scripts calling this wallet API.

Browser Fingerprinting Prevalence. Following our method defined in Section 3.1.2 to identify browser fingerprinting, we find that 878 scripts (79%) belonging to 1,099 websites (83%) engage in browser fingerprinting and leverage wallet information to enhance the fingerprints they generate. The maximum number of fingerprinting categories collected by a single script was 19 out of 22. Both mean and average number of fingerprinting categories collected by browser fingerprinting scripts is around 12. Also, 71 (8%) of the scripts performing browser fingerprinting, collected wallet information explicitly whereas 808 (92%) of the scripts collected wallet information implicitly. Figure 3 and Figure 4 list each a small snippet from two third-party scripts that were detected by our framework. Both snippets check for the existence of

Category	Websites	Third-Party Calls	Top Website (Rank)	Top Third-Party (Websites)
Pornography & Sexuality	200	138 (69%)	xhamster.com (160)	adsco.re (45)
Computers & Internet	108	56 (52%)	tinyurl.com (224)	cloudflare.com (10)
News & Media	89	65 (73%)	nytimes.com (96)	googlesyndication.com (19)
Finances	79	25 (32%)	opensea.io (1096)	cloudflare.com (7)
Adult Sites	64	46 (72%)	hitomi.la (1066)	adsco.re (11)
Entertainment	48	24 (50%)	bustle.com (2386)	googlesyndication.com (7)
E-commerce	41	16 (39%)	beget.com (1124)	cloudflare.com (6)
Business	40	15 (38%)	bytedance.com (2693)	cloudflare.com (6)
Shopping	39	15 (38%)	moneysavingexpert.com (5419)	cloudflare.com (5)
Games	38	11 (29%)	steamdb.info (4589)	m2.ai (2)

Table 5: Top 10 categories sorted by number of websites detected performing calls to wallet APIs.

```
document.addEventListener("DOMContentLoaded",
 (function() {
  var e = (0, t.getSettings)(),
      n = void 0 !== window.ethereum,
      o = void 0 !== window.BinanceChain,
      a = void 0 !== window.solana;
  ...
  var u = new XMLHttpRequest;
  u.open("post", "/x-api", !0), ...,
  u.send(JSON.stringify([{
    ...
    requestData: {
      model: {
        ...
        key: "ext_detection",
        data: {
          ethereum: n,
          BinanceChain: o,
          solana: a
        }
      }
    }
  }]))
}))
```

Figure 4: Code snippet hosted at `https://static-lvlt.xhcdn.com/xh-shared/js/v1d487c898d.ext-detect.js` accessing `window.ethereum`, `window.BinanceChain`, and `window.solana` wallet APIs to detect whether the user has any of the three wallet extensions installed.

wallet APIs. Figure 4 tries to check whether Ethereum, Binance Chain, or Solana wallet extensions are installed and sends this information back to the third-party server via an HTTP POST request.

Categories. We compared wallet API calls across website categories. Table 5 lists the top 10 categories in terms of number of websites that access wallet APIs. We used SafeDNS's website categorization service [54] to assign a category to each website. As shown in Table 5, Pornography & Sexuality is where we detected the most number of websites (i.e., 200) accessing wallet information, where the most popular website was `xhamster.com` (ranked 160 in Tranco). Moreover, 69% of the wallet API calls were performed by a third-party script, where `adsco.re` is the most widespread third-party with wallet API calls on 45 different websites. Websites with most third-party calls are in the category News & Media (73%), whereas websites with least third-party calls are in the category Games (29%).

Third-Parties. We found 680 websites (i.e., 51%) that include a third-party which calls a wallet API. The wallet API calls originate from 324 third-party scripts which belong to 118 unique third-party domains. Table 6 lists the top 10 third-parties for scripts that perform explicit (upper half) and implicit (lower half) wallet API calls. For explicit calls, we find that the third-party domain `wpadmngr.com` is the most widespread (embedded in 55 websites). For implicit calls, we find that the third-party domain `adsco.re` is the most widespread (embedded in 111 websites).

URL and Code Similarity. When analyzing the URLs of the 324 third-party scripts, we noticed that a large number were similar. Several third-party URLs contain the path `/cdn-cgi/challenge-platform/h/`. We found that these third-parties most likely deploy Cloudflare's Anti-DDoS protection [58], which consists of some JavaScript code that implicitly accesses wallet API information. We found 127 (i.e., 39%) such Cloudflare third-party scripts. We also clustered the remaining 197 third-party scripts based on their code by grouping scripts together which share the exact same JavaScript code. We found 2 clusters, one including the two third-parties `jsdelivr.net` and `unpkg.com` and one including the three third-parties `6347032d45.com`, `wpadmngr.com`, and `ba0182aa75.com`. The former third-parties are content delivery networks hosting the `web3.js` library, which is used by several DApps. The other three third-parties are interesting as we do not know who is running them, but we can see from Table 6, that `wpadmngr.com` and `ba0182aa75.com` are the two most widely deployed third-party scripts calling wallet APIs explicitly. Moreover, as they share the same code, we can infer that they belong to the same organization and that they are together deployed on 94 different websites.

Blocklists. Given that half of the calls to wallet APIs originate from third-parties, we checked whether blocklists could be

Third-Party Name	Third-Party Domain	Third-Party Script	Type	Websites	Min. Rank
-	**wpadmngr.com**	https://js.wpadmngr.com/static/adManager.m.js **(F)**	Explicit	55	1902
-	**ba0182aa75.com**	https://932d007132.ba0182aa75.com/1511a82de1dab2ee0c95006298aa98af.js **(F)**	Explicit	39	18392
xHamster	**xhcdn.com**	https://static-lvlt.xhcdn.com/xh-shared/js/v1d487c898d.ext-detect.js	Explicit	23	160
Taboola	**taboola.com**	https://cdn.taboola.com/scripts/cwc.es5.js	Explicit	22	381
Bustle	**bustle.com**	https://cdn2.bustle.com/2023/bustle/main-148fdc658d.js	Explicit	12	2386
Amazon	**cloudfront.net**	https://d2vjcex1bx9gzc.cloudfront.net/media/tags/goldfinchfinance.js	Explicit	6	18136
Google	**googletagmanager.com**	https://www.googletagmanager.com/gtm.js?id=GTM-P528B3	Explicit	5	96
Adshares	**web3ads.net**	https://app.web3ads.net/-/view.js	Explicit	3	17540
Prospect One	**jsdelivr.net**	https://cdn.jsdelivr.net/npm/@ledgerhq/connect-kit@1	Explicit	3	14898
SpookySwap	**spooky.fi**	https://spooky.fi/static/js/7.2ec90594.chunk.js	Explicit	2	105612
Adscore	**adsco.re**	https://c.adsco.re/ **(F)**	Implicit	111	1066
Cloudflare	**cloudflare.com**	https://challenges.cloudflare.com/cdn-cgi/challenge-platform/h/b/orchestrate/chl_api/v1 **(F)**	Implicit	84	1114
Google	**googlesyndication.com**	https://pagead2.googlesyndication.com/bg/KJeI0sMyo1Q6mjhDM9mKcjS2IqRt95c1wIDqLysfd0M.js **(F)**	Implicit	76	270
CHEQ	**defybrick.com**	https://rock.defybrick.com/placement_invocation?id=65349 **(F)**	Implicit	38	2923
MonetizeMore	**m2.ai**	https://m2d.m2.ai/v/pg-221207-f8d-nc-434e7f97016ae258bb936353072d000e.js **(F)**	Implicit	15	6737
CHEQ	**cheqzone.com**	https://ob.cheqzone.com/clicktrue_invocation.js?id=8911 **(F)**	Implicit	13	2577
-	**zfilm-hd.biz**	https://go.zfilm-hd.biz/cdn-cgi/challenge-platform/h/b/orchestrate/managed/v1 **(F)**	Implicit	9	34846
Anura	**anura.io**	https://script.anura.io/request.js?instance=3688597576 **(F)**	Implicit	8	6035
-	**rageagainstthesoap.com**	https://d.rageagainstthesoap.com/clicktrue_invocation.js?id=11825 **(F)**	Implicit	7	4042
ByteDance	**ttwstatic.com**	https://sf16-website-login.neutral.ttwstatic.com/obj/tiktok_web_login_static/webmssdk/1.0.0.1/webmssdk.js **(F)**	Implicit	5	74

Table 6: Top 10 third-parties with explicit (upper half) and implicit (lower half) wallet API calls. **(F)** Indicates that the third-party script has been flagged by our methodology as a browser fingerprinting script.

a reliable countermeasure. We downloaded the latest blocklists of Disconnect [18], DuckDuckGo [21], EasyList [22], EasyPrivacy [22], and Whotracks.me [65], and counted how many of the detected third-parties are blocked by the individual blocklists. We manually checked all third-parties and left out 10 of them as they are related to benign use-cases such as helper libraries (e.g., web3.js [63]). Figure 5 depicts an overview on the number of blocked third-parties. We observe that Whotracks.me provides the best protection by blocking 46 third-parties (43%). The weakest protection is given by Disconnect with only 13 third-parties blocked (12%). Moreover, we also checked whether installing all blocklists at the same time (i.e., combining blocklists) would improve protection. As seen in Figure 5, the combination of all five blocklists results in blocking 60 third-parties (56%), hence an improvement of 12% as compared to only using Whotracks.me's blocklist.

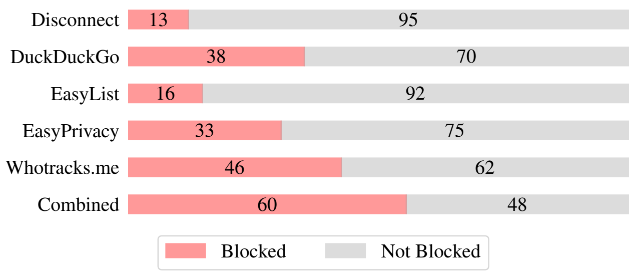

Figure 5: Third-party scripts blocked by popular blocklists.

4.3 Wallet Address Leakage

We analyze to what extent DApps and wallet extensions leak the user's wallet address to third-parties.

4.3.1 DApps

Winter et al. [66]. We compare the performance of our framework using Winter et al.'s [66] DeFi dataset. The dataset consists of 78 DeFi websites, however, 6 websites were down at the time of writing, 2 websites did not support MetaMask, and 4 were duplicates. After filtering, we were left with 66 websites to crawl. While Winter et al. connect manually to each website via MetaMask, we automatically connect to all of them using our MetaMask automator. Table 7 shows a comparison between the leaks measured by Winter et al. and our framework. Winter et al. found that 13 out of the 66 websites (20%), leak the user's wallet address to a third-party, whereas our results show that actually 39 out of the 66 websites (59%) leak the user's wallet address to a third-party. Overall, Winter et al. found 25 leaks whereas we found 2,164 leaks for the same websites. 98% (i.e., 2,131 leaks) are performed either via POST requests, WebSockets, or cookies. 61% of the leaks (i.e., 1,324 leaks) occur via POST requests. This emphasizes that solely analyzing GET requests, as Winter et al. did, is not sufficient. While Winter et al. found that wallet addresses are being leaked to 14 third-parties, our results show that the actual number is much higher, namely 64 third-parties. Table 7 highlights leaks that our framework and Winter et al. have in common (number between parentheses). For example, for `bifi.finance`, we detected 3 leaks which correspond to the same leaks as detected by Winter et al. However, we observe that `1inch.io`, `impermax.finance`, `jelly.market`, and `yearn.finance` did not leak the user's wallet address in our crawls anymore. On closer inspection, we find that `jelly.market` was down and we therefore were not able to collect any data and that the remaining three websites moved towards using their own API to retrieve blockchain data. Interestingly, for `dodoex.io` and `sablier.finance`, the leaks moved from GET requests to POST requests. This

DeFi Website	GET [66]	GET	POST	WebSockets	Cookies
1inch.io	1	0 (0)	0 (0)	0 (0)	0 (0)
aave.com	0	0 (0)	6 (0)	0 (0)	0 (0)
airswap.io	0	0 (0)	0 (0)	1 (0)	0 (0)
akropolis.io	0	0 (0)	0 (0)	777 (0)	0 (0)
alchemix.fi	0	7 (0)	551 (0)	0 (0)	0 (0)
balancer.fi	0	0 (0)	7 (0)	2 (0)	0 (0)
bancor.network	2	1 (0)	30 (0)	0 (0)	0 (0)
barnbridge.com	0	0 (0)	7 (0)	0 (0)	0 (0)
bifi.finance	3	3 (3)	3 (0)	0 (0)	0 (0)
boringdao.com	0	0 (0)	6 (0)	0 (0)	0 (0)
centrifuge.io	0	0 (0)	8 (0)	0 (0)	0 (0)
codefi.network	0	0 (0)	40 (0)	0 (0)	0 (0)
cream.finance	0	1 (0)	0 (0)	0 (0)	0 (0)
debank.com	0	2 (0)	4 (0)	0 (0)	0 (0)
defisaver.com	3	2 (2)	0 (0)	0 (0)	0 (0)
dmm.exchange	0	14 (0)	0 (0)	0 (0)	3 (0)
dodoex.io	2	0 (0)	2 (2)	0 (0)	0 (0)
dydx.exchange	0	0 (0)	43 (0)	0 (0)	0 (0)
enzyme.finance	0	0 (0)	5 (0)	0 (0)	0 (0)
fei.money	0	0 (0)	7 (0)	0 (0)	0 (0)
foundation.app	0	0 (0)	6 (0)	0 (0)	0 (0)
idle.finance	0	1 (0)	3 (0)	0 (0)	0 (0)
impermax.finance	1	0 (0)	0 (0)	0 (0)	0 (0)
indexcoop.com	0	0 (0)	55 (0)	0 (0)	0 (0)
inverse.finance	0	0 (0)	2 (0)	0 (0)	0 (0)
jelly.market	1	0 (0)	0 (0)	0 (0)	0 (0)
liquity.app	0	0 (0)	0 (0)	16 (0)	0 (0)
mai.finance	0	0 (0)	53 (0)	0 (0)	0 (0)
notional.finance	0	0 (0)	27 (0)	6 (0)	0 (0)
o3swap.com	0	0 (0)	70 (0)	0 (0)	0 (0)
opensea.io	0	0 (0)	8 (0)	0 (0)	0 (0)
opyn.co	0	0 (0)	4 (0)	2 (0)	0 (0)
pickle.finance	0	0 (0)	258 (0)	0 (0)	0 (0)
rari.capital	0	0 (0)	16 (0)	0 (0)	0 (0)
rarible.com	3	0 (0)	1 (0)	0 (0)	0 (0)
reflexer.finance	1	2 (2)	1 (0)	0 (0)	0 (0)
sablier.finance	1	0 (0)	6 (3)	0 (0)	0 (0)
truefi.io	0	0 (0)	8 (0)	0 (0)	0 (0)
uniswap.org	0	0 (0)	4 (0)	0 (0)	0 (0)
warp.finance	0	0 (0)	2 (0)	0 (0)	0 (0)
yearn.finance	4	0 (0)	0 (0)	0 (0)	0 (0)
yield.is	0	0 (0)	65 (0)	0 (0)	0 (0)
zerion.io	3	0 (0)	16 (0)	0 (0)	0 (0)
Total	25	33 (7)	1324 (5)	804 (0)	3 (0)

Table 7: Leaks measured by Winter et al. [66] vs our framework. Numbers in parentheses indicate common leaks.

can be due to a change in the API of the third-parties. Moreover, while `alchemix.fi`, `cream.finance`, `debank.com`, `dmm.exchange`, and `idle.finance` did not leak the user's wallet address to any third-parties via GET requests during Winter et al.'s study, our results demonstrate the opposite. Since Winter et al.'s study is already more than a year ago, we assume that those third-party leaks were added after the study was conducted. Finally, we also observed that `dmm.exchange` leaks the user's wallet address to `kyberswap.com` via 3 different cookies set by Mixpanel (see Figure 6 for an example of such a cookie).

DappRadar.com [17]. Winter et al.'s dataset is useful for comparing performance, but it is insufficient to draw any general conclusions due to it being relatively small and only focusing on DeFi websites. Therefore, we crawled DappRadar.com

```
{
  "name": "mp_ff1eea26c19dcf4a7c35ebbc8631e714_mixpanel",
  "value": "%7B%22distinct_id%22%3A%20%220x7e4ABd63A7C8
  314Cc28D388303472353D884f292%22%2C%22%24device_id%22%
  3A%20%22185bc157265a0d-0daab5a6ab23c7-17525635-16a7f0
  -185bc157266f56%22%2C%22%24user_id%22%3A%20%220x7e4AB
  d63A7C8314Cc28D388303472353D884f292%22%2C%22%24initia
  l_referrer%22%3A%20%22%24direct%22%2C%22%24initial_re
  ferring_domain%22%3A%20%22%24direct%22%2C%22wallet_ad
  dress%22%3A%20%220x7e4ABd63A7C8314Cc28D388303472353D8
  84f292%22%2C%22platform%22%3A%20%22Web%22%2C%22networ
  k%22%3A%20%22Ethereum%22%7D",
  "domain": ".kyberswap.com",
  ...
}
```

Figure 6: Cookie set on `dmm.exchange` by Mixpanel for `.kyberswap.com` domain containing user's wallet address: `0x7e4ABd63A7C8314Cc28D388303472353D884f292`.

to obtain a much larger and diverse dataset. We ended up getting 1,572 DApp websites across 9 categories. Our automator was able to automatically connect to 616 (39%) of them. The automator had less issues in connecting to DeFi DApps, with a success rate of 61%. On the other hand, our automator found it hard to connect to High Risk DApps, with a success rate of only 20%. There are several reasons why it was not able to connect to all DApps. Most websites are simply down or our automator is not able to detect the connect button by scanning the HTML. Some websites do not support MetaMask, or require users to either agree on the terms or register and login via an email address and a password before being able to interact with the DApp. Section 5.1 provides a clear breakdown regarding why our automator was not able to connect to certain DApps.

Table 9 summarizes our detected leaks on the `DappRadar.com` dataset. Our framework identified 211 unique DApp websites (35% of the connected DApps) which leak the user's wallet address across 137 unique third-parties. As shown in Table 9, Gambling DApps leak the least (6%), whereas Exchanges leak the most (59%). Our data also shows that 1,400 DApps (89%) embed at least one third-party. On average DApps embed 7 different third-parties. The maximum we observed was 61 third-parties embedded on a single DApp's website. Table 8 lists the top 20 third-parties where the user's wallet address is leaked to. As we can see, most third-parties are JSON-RPC providers (75%) and the rest are tracking & analytics platforms (25%). DApps need to connect to a blockchain node to retrieve blockchain related information. This connection is often performed via JSON-RPC providers. While leaks to JSON-RPC providers are unavoidable, they still may pose a threat to user's privacy as they may collect additional information such as what DApps the user visited or its IP address. Often users do not know to which JSON-RPC provider the DApp is connected to. Leaks to tracking &

#	Third-Party Name	Third-Party Domain	Category	Collects IP Address	DApps	GET	POST	WebSockets	Cookies
1	Infura	**infura.io**	JSON-RPC Provider	Yes	42	0	818	12	0
2	Alchemy	**alchemyapi.io**	JSON-RPC Provider	Yes	39	0	638	772	0
3	The Graph	**thegraph.com**	JSON-RPC Provider	Yes	22	0	230	0	0
4	Sentry	**sentry.io**	Tracking & Analytics	Yes	21	0	122	0	0
5	Google	**google-analytics.com**	Tracking & Analytics	Yes	18	38	6	0	0
6	Alchemy	**alchemy.com**	JSON-RPC Provider	Yes	16	3	132	6	0
7	Amplitude	**amplitude.com**	Tracking & Analytics	Yes	13	0	37	0	0
8	Blocknative	**blocknative.com**	JSON-RPC Provider	Yes	12	0	0	25	0
9	Ankr	**ftm.tools**	JSON-RPC Provider	Yes	11	0	166	0	0
10	Binance	**binance.org**	JSON-RPC Provider	Yes	10	0	99	0	0
11	Ankr	**ankr.com**	JSON-RPC Provider	Yes	9	0	316	0	0
12	Mixpanel	**mixpanel.com**	Tracking & Analytics	Yes	9	17	26	0	0
13	Etherscan	**etherscan.io**	JSON-RPC Provider	Yes	8	38	0	0	0
14	Binance	**ninicoin.io**	JSON-RPC Provider	Yes	7	0	52	0	0
15	Arbitrum	**arbitrum.io**	JSON-RPC Provider	Yes	6	0	67	0	0
16	Avalanche	**avax.network**	JSON-RPC Provider	Yes	6	0	47	0	0
17	Cloudflare	**cloudflare-eth.com**	JSON-RPC Provider	Yes	6	0	19	0	0
18	Google	**firestore.googleapis.com**	Tracking & Analytics	Yes	6	22	17	0	0
19	Pocket Network	**pokt.network**	JSON-RPC Provider	No	6	0	25	0	0
20	Ankr	**polygon-rpc.com**	JSON-RPC Provider	Yes	6	0	22	0	0

Table 8: Top 20 third-parties detected by our framework on the DappRadar.com to which the wallet address was leaked.

Category	DApps	Third-Parties	GET	POST	WebSockets	Cookies
Collectibles	32 (20%)	23 (4%)	38	77	9	0
DeFi	93 (45%)	87 (17%)	319	2533	807	0
Games	22 (30%)	20 (6%)	43	95	6	0
Other	21 (35%)	22 (6%)	32	227	2	0
Marketplaces	21 (45%)	25 (8%)	7	102	4	0
High Risk	3 (18%)	5 (4%)	6	9	0	0
Exchanges	19 (59%)	42 (18%)	46	574	38	0
Gambling	1 (6%)	1 (0%)	0	1	0	0
Social	3 (37%)	5 (4%)	4	25	0	0
Total Unique	211 (35%)	137 (9%)	495	3643	866	0

Table 9: Leaks identified on the DappRadar.com dataset.

analytics platforms are unnecessary and a clear privacy violation. These platforms should not have access to sensible user information such as wallet addresses. For example, Figure 7 shows an HTTP GET request from degens.farm that leaks the user's wallet address to google-analytics.com. We studied the privacy policies of the top 20 third-parties and observe that 95% state that they collect the user's IP address. Pocket Network is the only third-party in Table 8 that does not collect the IP address of its users. We also observe that Infura is the most widespread third-party, with 42 DApps leaking the user's wallet address to Infura. For the DappRadar.com dataset, none of the DApps shared the user's wallet address via cookies. However, similar to Winter et al.'s dataset, most DApps share the user's wallet address via HTTP POST requests, then WebSockets, and finally HTTP GET requests.

4.3.2 Wallet Extensions

We analyzed whether any of the 100 wallet extensions contained in our dataset include third-parties and with whom they share the user's wallet address and potentially even the password or browsing history. Fortunately, none of the analyzed browser extensions seem to leak the user's password. At least we were not able to identify the password in any of

Wallet Extension	Third-Party	GET	POST	WebSockets	Cookies
NuFi	**milkomeda.com (R)**	0	4	0	0
Petra Aptos Wallet	**segment.io**	0	22	0	0
Pitaka	**sentry.io**	0	2	0	0
Nabox Wallet	**mytokenpocket.vip**	0	11	0	0
GameStop Wallet	**loopring.network**	97	0	0	0
	immutable.com	116	0	0	0
Martian Wallet	**dialectapi.to**	16	2	0	0
Crust Wallet	**subscan.io**	0	17	0	0
JulWallet	**swapliquidity.org**	0	1	0	0
Ethos Sui Wallet	**shinami.com (R)**	0	16	0	0
G.U. Smart Wallet	**infura.io (R)**	0	1	0	0
Coinbase Wallet	**fantom.network**	0	1	0	0
	blockscout.com	2	0	0	0
	binance.org	0	1	0	0
	etherscan.io	4	0	0	0
	avax-test.network	0	1	0	0
	bscscan.com	4	0	0	0
	snowtrace.io	4	0	0	0
	arbiscan.io	4	0	0	0
	polygonscan.com	4	0	0	0
	ftmscan.com	4	0	0	0
Ethereum AllInOne	**ethplorer.io**	21	0	0	0
Verto	**pulsechain.com (R)**	0	3	0	0
	volentix.io	25	1	0	0
Total	**24**	301	83	0	0

Table 10: Third-party leaks detected in 100 wallet extensions. (R) stands for JSON-RPC provider.

the requests that we analyzed (including requests to the first-parties themselves). We analyzed the manifest file of each wallet extension and checked whether they can inject content scripts on any website and if they request access to sensible permissions. 89 of the 100 wallet extensions can inject content scripts on any website (i.e., the manifest includes one of the following patterns: 'http://*/*', https://*/*',<all_urls>', *://*/*'). Hence, these 89 wallet could potentially read the URL of the current page and send it to a backend. We also found that 66 wallet extensions request permission for either accessing "history", "tabs", or "activeTab". We visited three different websites (nytimes.com, etherscan.io, and uniswap.org), using each extension and checked whether

```
https://www.google-analytics.com/collect?v=1&_v=j99&a=1044933369&t=event&ni=0&_s=1&dl=https%3A%2F%2Fdegens.farm%2Fwallet&
ul=en-us&de=UTF-8&dt=Degen%27%24%20Farm%3A%20Wallet&sd=30-bit&sr=1512x982&vp=1512x749&je=0&ec=WalletConnected&ea=0x7e4abd
63a7c8314cc28d388303472353d884f292&el=labelForWalletConnect&ev=7.20999590401511e%2B47&_u=aADAAEABAAAAACAAI~&jid=&gjid=&ci
d=437541385.1675387202&tid=UA-201259489-1&_gid=196110690.1675387203&gtm=2wg2105PC69BZ&z=1330733511
```

Figure 7: Wallet address leaked via HTTP GET request to `google-analytics.com` on the `degens.farm` DApp.

there are requests that include any of the three websites. We were not able to detect any extension leaking any of the visited websites. However, we did find that wallet extensions do leak the user's wallet address to third-parties. Table 10 lists the wallet extension that leak the user's wallet address. We found 13 out of 100 analyzed extensions which leak the user's wallet address to at least one of 24 third-parties. In total we found 139 third-parties across all browser extensions. While most wallet extensions only leak the wallet address to a single third-party, Coinbase's wallet extension leaks the user's wallet address to 10 different third-parties. Surprisingly, none of the wallet extensions' third-parties seem to overlap. However, we do observe that `sentry.io` and `infura.io` are present in both of our datasets, DApps and wallet extensions. Although, Infura is a benign platform, since it is a JSON-RPC provider and therefore required for the wallet extension work, the user is not made aware of this connection and the fact that Infura can link requests across websites and infer for example that a user X uses wallet Y and visits DApps A and B regularly. Sentry on the other hand is clearly not benign as it is a tracking & analytics platform, hence sensitive information such as the user's wallet address should not be leaked to such platforms.

5 Discussion

We discuss the limitations of our methodology and elaborate potential countermeasures, including their pitfalls.

5.1 Limitations

Our methodology for detecting wallet API calls is based on TRC which comes built in with an anti-bot detection. However, anti-bot detection solutions are not perfect and thus websites can still detect whether a bot is crawling them and thus behave differently or block access to the website. Moreover, our methodology leverages a wallet simulator that we build to inject fake JavaScript objects into the DOM such that we can simulate wallets without requiring to install them and setting them up. However, our simulator does not simulate a full-fledged wallet. It is limited to the simulated JavaScript properties listed in Table 1 in Section 3.1.1. Thus, third-party scripts could detect our wallet simulator by checking for inconsistencies such as missing JavaScript properties in the different wallet APIs. Although, the likelihood that third-party scripts currently do this is rather low. We did not experience such checks when analyzing the code of third-party scripts manually. However, it could be that in the future third-party scripts will adapt and try to probe for multiple properties of a wallet before making any decisions.

Our MetaMask automator was only able to automatically connect to 39% of the analyzed DApps. Appendix C, provides a detailed breakdown on connection failures that occurred over the DeFi subset of the DappRadar.com dataset. In 24% of the cases, the URLs did not point to a valid DApp and in 14% of the cases the DApp's website was simply down. 3% of the DApps did not support MetaMask. For 18% of the DApps our automator could simply not detect a connect button or MetaMask button within the HTML despite the DApp's buttons containing labels that match the keywords in Appendix B. However, there are also Dapps that contain buttons that do not match any of our keywords (8%) or which represent their buttons as images (8%). 15% require users to give their consent by ticking a checkbox before being able to interact with them. Finally, 7% require users to create an account and login via email and password.

Moreover, during our crawl we only visited the landing page of a website or DApp and might have missed any third-party scripts that perform tracking or leak the user's wallet address. This and the fact that we were not able to connect to all the DApps and limited ourselves to a handful of wallets (extensions are well as simulated wallet APIs), highlights that our results should only be considered as a lower bound.

5.2 Countermeasures

Privacy-conscious users want to prevent their wallet address to be leaked to third-parties, but also minimize their footprint (i.e., the fact that they have a wallet installed on their browser) for online tracking. Initially, wallets would expose the user's unique wallet address to any website without asking the user for prior permission. However, this changed with the release of EIP-2255 [16] which requires wallets to ask user's for permission prior to returning any sensitive information to DApps. However, the permission system is still flawed. Any third-party that is embedded inside the DApp also has access to the sensitive information once, although the user has granted only permission to the DApp and not the third-parties.

Winter et al. [66] proposed a countermeasure which does not prevent wallet address leakage per-se, but limits its usefulness in linking users across DApps as it generates individual wallet addresses for each DApp a user visits. This follows a similar idea that has been proposed in the past to prevent linking users across websites by using different yet consistent web identities across websites [60]. Specifically, DApps always interact with a fake proxy wallet address that is derived from the user's real wallet address. All requests that either go through MetaMask or via an JSON-RPC provider are then intercepted and the fake address is swapped with the user's real wallet address such that the DApp is able to perform actions on real data.

However, this approach has several pitfalls. First, the fact that the user's real balance is returned allows DApps and other third-parties to map the fake address to the real address by scanning the blockchain for an address that has the exact same balance. This is trivial because the balance has a high resolution (e.g., 256-bit resolution in the case of Ethereum) and thus the likelihood that two users having the exact same balance is very low. Second, the interception of traffic as well as the swapping between fake and real requires complex management and is prone to errors. For example, transactions are usually not directly mined and most DApps rely on a transaction receipt which includes a transaction hash that allows them to continuously poll the blockchain for the transaction's confirmation status. Hence, the countermeasure also needs to fake transaction hashes otherwise DApps and third-parties might use this information to obtain the users real wallet address. But in fact this might break the usability of many DApps as they sometimes point to other websites such as Etherscan using the transaction hash. Third, the proposed countermeasure does not hide the existence of a wallet extension from third-parties. Trackers will still be able to detect whether or not a user has a wallet installed on its browser.

As an alternative, users could rely on Ad blockers [18, 21, 22, 65] to simply block requests from and to third-party tracking scripts. In Section 4.2, we measured the effectiveness of popular blocklists against the third-parties that we found to access wallet information. The best performing blocklist only managed to block 46 out of 118 third-parties (i.e., 39%). Moreover, even with all the blocklists combined, only 51% of the third-parties are blocked. Blocklists do not scale, they can simply be evaded by deploying the same script to a different domain that is not yet blacklisted. For instance, we found that the script which is hosted on `wpadmngr.com` (top 1 in our list of detected third-party scripts) is identical to the scripts hosted on `ba0182aa75.com` and `6347032d45.com`. Since the two last domains appear to be random, we assume that they might be used by `wpadmngr.com` to avoid blocklists.

6 Related Work

There are several ways to track users online, ranging from classical stateful methods such as third-party cookies [1, 26, 42] to novel stateless methods such as browser fingerprinting [8, 39, 51]. A number of studies have been conducted over the past years in order to measure the prevalence of third-party cookies and novel browser fingerprinting techniques [30, 36, 38, 61]. Essentially, any JavaScript API that provides stable yet user-configuration specific information can be leveraged to generate together with other attributes a unique browser fingerprint. This information may range from simple properties such as screen resolution to more advanced techniques such as canvas fingerprinting [2]. For instance, Englehardt et al. [25] were the first to provide evidence that third-party trackers enhance their browser fingerprinting scripts with information provided by the WebRTC API, Audio API, and Battery Status API. Our work analyses whether trackers are leveraging wallet APIs to enhance their browser fingerprinting scripts to better track users online.

Recently, Senol et al. [55] discovered that a large number of websites leak the user's email address and password to third-parties. In a similar vein, our work aims to shed light into the inner workings of DApps and wallets to uncover if they might leak a user's wallet address or password to third-parties.

Privacy is not only difficult to achieve on the web, but it is also challenging to achieve when dealing with cryptocurrencies. Security and privacy concepts are often not well understood by cryptocurrency users. For instance, Krombholz et al. [37] surveyed over 900 users with respect to their knowledge on security and privacy of Bitcoin. None of the users made a backup of their wallet passphrase on a separate computer. 22% report that they already lost some of their cryptocurrency due to scams or loss of their passphrase. Also, 32% think that Bitcoin is anonymous, despite the fact that transactions can be traced. This is in line with the findings of Mai et al. [44] and Voskobojnikov et al. [62] where users do not understand the concept of public and private keys or believe that transactions are confidential and cannot be seen by third-parties. These works point out that users might have a misconception of wallets with respect to the privacy that they provide.

As more and more online vendors accept cryptocurrencies as a payment method and an increasing number of decentralized applications begin to emerge, the question around linkability and user privacy becomes indispensable. A number of previous works have focused on analyzing the linkability of cryptocurrency transactions [4, 10, 46] including their deanonymization via network-layer attacks [3, 7]. Goldfeder et al. [29] were the first to analyze the intersection between cryptocurrencies and online privacy. The authors find that online trackers are able to collect enough information to link cryptocurrency transactions to online purchases. Béres et al. [5] demonstrate how attackers can link different Ethereum addresses to the same user by analyzing meta information such as time of the day and gas price. Even mixers (i.e., services that shuffle transactions in order to break linkability) have been found to be broken [28, 34, 56]. Users often do not understand how to use mixers properly and use, for example, the same wallet address for depositing and retrieving cryptocurrency, thereby making mixing essentially useless.

Li et al. [43] present a denial-of-service attack against blockchain providers, which are frequently used by DApps and wallets to retrieve blockchain information. Blockchain providers often do not impose a gas limit on certain operations and thus malicious users may exploit this fact to make blockchain providers engage in heavy computations.

The work by Winter et al. [66] is the closest to our work. However, their goal is to analyze the security, privacy, and decentralization properties of popular DeFi front ends, while we aim to analyze the privacy implications of wallets. The authors analyzed 78 handpicked DeFi websites for wallet address leakage and found that 17% of the websites leak the user's wallet address. We found that 59% of the websites leak the user's wallet address. This is because our framework not only analyzes HTTP GET requests but also HTTP POST requests, WebSockets, and cookies. Moreover, while Winter et al. analyzed the websites manually, our work analyzes them automatically. This enables us to perform an automated large-scale study on DApps. Finally, Winter et al. did not analyze whether wallet extensions also leak the user's wallet address and whether websites make use of wallet information to fingerprint users.

7 Conclusion

We present the first systematic study on Web3-based browser fingerprinting and wallet address exfiltration. We built a framework which is capable of detecting JavaScript calls on wallet APIs as well as intercept and search HTTP requests, WebSockets and cookies for leaked wallet addresses. Our framework integrates a wallet simulator which imitates different wallet extensions by injecting wallet-specific properties into the website's DOM, and developed an automator which automatically sets up MetaMask and connects it to DApps. Using our framework we analyzed the top 100K websites and found evidence of 1,325 websites checking the presence of wallet extensions installed within the user's browser. We analyzed 1,572 DApps and found that 211 of them leak the user's wallet address to third-parties. Moreover, we analyzed 100 popular wallets and found that 13 of them deliberately leak the user's wallet address to third-parties. We evaluated countermeasures such as Ad blockers and found that they are not completely effective in blocking all the third-party scripts and leaks detected by our framework. We conclude that wallets pose a serious threat to user's privacy and that new solutions need to be developed that allow users to interact with DApps in a secure and privacy-preserving way.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their valuable comments and feedback. This work was supported by the Zurich Information Security & Privacy Center (ZISC).

Availability

The code that was used to conduct this study as well as the data that was collected during this study is publicly available on GitHub at: https://github.com/christoftorres/Web3-Privacy.

References

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 674–689. ACM, 2014.

[2] Gunes Acar, Marc Juárez, Nick Nikiforakis, Claudia Díaz, Seda F. Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1129–1140. ACM, 2013.

[3] Maria Apostolaki, Cedric Maire, and Laurent Vanbever. Perimeter: A network-layer attack on the anonymity of cryptocurrencies. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 2021.

[4] M. A. Hannan Bin Azhar and Robert Vause Whitehead. A study of user experiences and network analysis on anonymity and traceability of bitcoin transactions. *EAI Endorsed Trans. Security Safety*, 7(25):e3, 2020.

[5] Ferenc Béres, István András Seres, András A. Benczúr, and Mikerah Quintyne-Collins. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2021, Online Event, August 23-26, 2021*, pages 69–78. IEEE, 2021.

[6] Binance. Binance Wallet - API Documentation. https://binance-wallet.gitbook.io/binance-chain-wallet/dev/get-started, January 2023.

[7] Alex Biryukov and Sergei Tikhomirov. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*, pages 172–184. IEEE, 2019.

[8] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via

cross-browser fingerprinting. In *Information Security Technology for Applications - 16th Nordic Conference on Secure IT Systems, NordSec 2011, Tallinn, Estonia, October 26-28, 2011, Revised Selected Papers*, volume 7161 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2011.

[9] Cardano Foundation. Cardano dApp-Wallet Web Bridge. https://github.com/cardano-foundation/CIPs/tree/master/CIP-0030, January 2023.

[10] Wren Chan and Aspen Olmsted. Ethereum transaction graph analysis. In *12th International Conference for Internet Technology and Secured Transactions, ICITST 2017, Cambridge, United Kingdom, December 11-14, 2017*, pages 498–500. IEEE, 2017.

[11] Chrome Developers. CrUX on BigQuery. https://developer.chrome.com/docs/crux/bigquery/, November 2022.

[12] Chrome Web Store. MetaMask. https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn, January 2023.

[13] Coinbase. Coinbase Wallet Injected Ethereum Provider. https://docs.cloud.coinbase.com/wallet-sdk/docs/injected-provider, January 2023.

[14] CoinMarketCap. Cryptocurrency Ranking by Market Cap. https://coinmarketcap.com/, January 2023.

[15] ConsenSys. ConsenSys Privacy Policy Update. https://consensys.net/blog/news/privacy-policy-update/, November 2022.

[16] Dan Finlay, Erik Marks. EIP-2255: Wallet Permissions System - Ethereum Improvement Proposals, no. 2255. https://eips.ethereum.org/EIPS/eip-2255, August 2019.

[17] DappRadar. Top Ethereum DApps. https://dappradar.com/rankings/protocol/ethereum, January 2023.

[18] Disconnect. Disconnect Tracking Protection. https://github.com/disconnectme/disconnect-tracking-protection, January 2023.

[19] DuckDuckGo. Tracker Radar Collector. https://github.com/duckduckgo/tracker-radar-collector, January 2023.

[20] DuckDuckGo. Tracker Radar Collector - API Calls. https://github.com/duckduckgo/tracker-radar-collector/blob/main/collectors/APICalls/breakpoints.js, January 2023.

[21] DuckDuckGo. Web Tracker Blocklist. https://github.com/duckduckgo/tracker-blocklists/tree/main/web, January 2023.

[22] EasyList. EasyList & EasyPrivacy. https://easylist.to/, January 2023.

[23] Peter Eckersley. How unique is your web browser? In Mikhail J. Atallah and Nicholas J. Hopper, editors, *Privacy Enhancing Technologies, 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings*, volume 6205 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2010.

[24] Steven Englehardt, Jeffrey Han, and Arvind Narayanan. I never signed up for this! privacy implications of email tracking. *Proc. Priv. Enhancing Technol.*, 2018(1):109–126, 2018.

[25] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1388–1401. ACM, 2016.

[26] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan R. Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 289–299. ACM, 2015.

[27] Ethereum.org. JSON-RPC API. https://ethereum.org/en/developers/docs/apis/json-rpc/, January 2023.

[28] Simin Ghesmati, Walid Fdhila, and Edgar R. Weippl. Sok: How private is bitcoin? classification and evaluation of bitcoin privacy techniques. In *ARES 2022: The 17th International Conference on Availability, Reliability and Security, Vienna,Austria, August 23 - 26, 2022*, pages 5:1–5:14. ACM, 2022.

[29] Steven Goldfeder, Harry A. Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proc. Priv. Enhancing Technol.*, 2018(4):179–199, 2018.

[30] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit 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 on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 309–318. ACM, 2018.

[31] Google. Chrome DevTools Protocol - Version: 1.3. https://chromedevtools.github.io/devtools-protocol/, January 2023.

[32] Google. Chrome Web Store - Extensions. https://chrome.google.com/webstore/category/extensions, January 2023.

[33] Google. Puppeteer - Version: 19.6.2. https://pptr.dev/, January 2023.

[34] Younggee Hong, Hyunsoo Kwon, Jihwan Lee, and Junbeom Hur. A practical de-mixing algorithm for bitcoin mixing services. In *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC@AsiaCCS 2018, Incheon, Republic of Korea, June 4, 2018*, pages 15–20. ACM, 2018.

[35] Infura. The world's most powerful suite of high availability blockchain APIs and developer tools. https://www.infura.io/, January 2023.

[36] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1143–1161. IEEE, 2021.

[37] Katharina Krombholz, Aljosha Judmayer, Matthias Gusenbauer, and Edgar R. Weippl. The other side of the coin: User experiences with bitcoin security and privacy. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 555–580. Springer, 2016.

[38] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Trans. Web*, 14(2):8:1–8:33, 2020.

[39] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 878–894. IEEE Computer Society, 2016.

[40] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019.

[41] Sebastian Lekies, Ben Stock, Martin Wentzel, and Martin Johns. The unexpected dangers of dynamic javascript. In Jaeyeon Jung and Thorsten Holz, editors, *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 723–735. USENIX Association, 2015.

[42] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska 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, Austin, TX, USA, August 10-12, 2016*. USENIX Association, 2016.

[43] Kai Li, Jiaqi Chen, Xianghong Liu, Yuzhe Richard Tang, XiaoFeng Wang, and Xiapu Luo. As strong as its weakest link: How to break blockchain dapps at RPC service. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

[44] Alexandra Mai, Katharina Pfeffer, Matthias Gusenbauer, Edgar R. Weippl, and Katharina Krombholz. User mental models of cryptocurrency systems - A grounded theory approach. In *Sixteenth Symposium on Usable Privacy and Security, SOUPS 2020, August 7-11, 2020*, pages 341–358. USENIX Association, 2020.

[45] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 413–427. IEEE Computer Society, 2012.

[46] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. *Commun. ACM*, 59(4):86–93, 2016.

[47] MetaMask. A crypto wallet & gateway to blockchain apps. https://metamask.io/, January 2023.

[48] MetaMask. MetaMask Docs - Ethereum Provider API. https://docs.metamask.io/guide/ethereum-provider.html#ethereum-ismetamask, January 2023.

[49] MetaMask Docs. Ethereum Provider API. https://docs.metamask.io/guide/ethereum-provider.html, January 2023.

[50] Mozilla. Third-party cookies and Firefox tracking protection. https://support.mozilla.org/en-US/kb/third-party-cookies-firefox-tracking-protection, January 2023.

[51] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 541–555. IEEE Computer Society, 2013.

[52] Phantom. Phantom Developer Docs - Detecting the Provider. https://docs.phantom.app/solana/integrating-phantom/extension-and-in-app-browser-web-apps/detecting-the-provider, January 2023.

[53] Gaston Pugliese, Christian Riess, Freya Gassmann, and Zinaida Benenson. Long-term observation on browser fingerprinting: Users' trackability and perspective. *Proc. Priv. Enhancing Technol.*, 2020(2):558–577, 2020.

[54] SafeDNS. Website Category Check. https://www.safedns.com/feature/check-website, January 2023.

[55] Asuman Senol, Gunes Acar, Mathias Humbert, and Frederik J. Zuiderveen Borgesius. Leaky forms: A study of email and password exfiltration before form submission. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 1813–1830. USENIX Association, 2022.

[56] Ardeshir Shojaeenasab, Amir Pasha Motamed, and Behnam Bahrak. Mixing detection on bitcoin transactions using statistical patterns. *CoRR*, abs/2204.02019, 2022.

[57] Oleksii Starov, Phillipa Gill, and Nick Nikiforakis. Are you sure you want to contact us? quantifying the leakage of PII via website contact forms. *Proc. Priv. Enhancing Technol.*, 2016(1):20–33, 2016.

[58] Steven. A repository containing research regarding various Anti-DDoS systems. (CloudFlare). https://github.com/scaredos/cfresearch, January 2023.

[59] Christof Ferreira Torres and Hugo Jonker. Investigating Fingerprinters and Fingerprinting-Alike Behaviour of Android Applications. In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, volume 11099 of *Lecture Notes in Computer Science*, pages 60–80. Springer, 2018.

[60] Christof Ferreira Torres, Hugo L. Jonker, and Sjouke Mauw. FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2015.

[61] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-STALKER: tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 728–741. IEEE Computer Society, 2018.

[62] Artemij Voskobojnikov, Borke Obada-Obieh, Yue Huang, and Konstantin Beznosov. Surviving the cryptojungle: Perception and management of risk among north american cryptocurrency (non)users. In *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 595–614. Springer, 2020.

[63] Web3.js. web3.js - Ethereum JavaScript API. https://web3js.readthedocs.io/en/v1.10.0/, May 2023.

[64] WebKit. Full Third-Party Cookie Blocking and More. https://webkit.org/blog/10218/full-third-party-cookie-blocking-and-more/, January 2023.

[65] WhoTracksMe. Bringing Transparency to Online Tracking. https://github.com/whotracksme/whotracks.me, January 2023.

[66] Philipp Winter, Anna Harbluk Lorimer, Peter Snyder, and Benjamin Livshits. Security, Privacy, and Decentralization in Web3. *CoRR*, abs/2109.06836, 2023.

[67] Zhiyuan Sun. Report: DApp daily users surge to 2.4M in Q1 2022 despite headwinds. https://cointelegraph.com/news/report-dapp-daily-users-surge-to-2-4m-in-q1-2022-despite-headwinds, April 2022.

Appendix

A Browser Fingerprinting Categories

Table 11 lists all the JavaScript APIs that our framework uses to detect browser fingerprinting, including the category that we assigned to each API. For example, `*userAgent*` means that any JavaScript API call that includes the string "userAgent" will be collected and assigned to the category browser, and the category browser is not considered as an explicit browser fingerprinting category.

JavaScript API Call	Category	Explicit
`window.ethereum`	Wallet	✗
`window.cardano`	Wallet	✗
`window.solana`	Wallet	✗
`window.BinanceChain`	Wallet	✗
`RTCPeerConnection*`	RTC	✓
`RTCPeerConnectionIceEvent*`	RTC	✓
`WebGLRenderingContext*`	WebGL	✓
`HTMLCanvasElement*`	Canvas	✓
`CanvasRenderingContext2D*`	Canvas	✓
`*Storage*`	Storage	✗
`*indexedDB*`	Storage	✗
`Screen*`	ScreenSize	✗
`*screen*`	ScreenSize	✗
`*cookie*`	Cookies	✗
`Date*`	DateTime	✗
`*DateTimeFormat*`	DateTime	✗
`*getBattery*`	Battery	✓
`*Height*`	WindowSize	✗
`*Width*`	WindowSize	✗
`BarProp*`	WindowSize	✗
`*connection*`	Connection	✗
`*onLine*`	Connection	✗
`*devicePixelRatio*`	ScreenResolution	✗
`*window.name*`	WindowLocation	✗
`*plugins*`	Plugins	✓
`*mimeType*`	Plugins	✓
`*canPlayType*`	Plugins	✓
`*vendor*`	Browser	✗
`*product*`	Browser	✗
`*platform*`	Browser	✗
`*app*`	Browser	✗
`*userAgent*`	Browser	✗
`*language*`	Language	✗
`DeviceOrientationEvent*`	Device	✓
`DeviceMotionEvent*`	Device	✓
`*maxTouchPoints*`	Device	✓
`*hardwareConcurrency*`	Device	✓
`*deviceMemory*`	Device	✓
`*memory*`	Device	✓
`AudioBuffer*`	Audio	✓
`OfflineAudioContext*`	Audio	✓
`*requestMediaKeySystemAccess*`	Media	✗
`*mediaDevices*`	Media	✗
`*enumerateDevice*`	Media	✗
`*mediaCapabilities*`	Media	✗
`Navigator*`	Navigator	✗
`Performance*`	Performance	✗
`speechSynthesis*`	SpeechSynthesis	✓

Table 11: Browser fingerprinting related JavaScript API calls and assigned category.

B List of Keywords Used by Automator

Table 12 lists all the keywords that our automator scans for within a website's HTML to find a "Connect" and "MetaMask" button.

Connect	*"Connect to MetaMask", " Connect Wallet ", "Connect Wallet", "Connect wallet", "connect wallet", "Connect to a wallet", "Connect to wallet", "Connect your wallet", "Sign In", "Connect", "CONNECT WALLET", "CONNECT", "SIGN IN", "WALLET", "SIGN", "sign", "SIGNIN", "Sign Up", "Connect Your Wallet", "Wallet", "Connect a Wallet", "Connect a wallet", "Sign in", "sign in", "connect", "Log in via web3 wallet", "wallet", "account", "Account"*
MetaMask	*"MetaMask", "MetaMask ", "metamask", "Connect Metamask", "Connect MetaMask", "Metamask", "Connect to MetaMask", "browser wallet", "Browser Wallet", "Browser wallet", "Metamask & Web3"*

Table 12: Keywords used by the automator to identify connect and MetaMask buttons on DApp websites.

C Breakdown of Connection Failures

Table 13 provides a breakdown over the reasons that resulted in our automator in failing to automatically connect to the DeFi related DApps from our DappRadar.com dataset.

Connection Failure	#	(%)
Not a DApp	32	(24%)
Button text not detectable	24	(18%)
Consent required	20	(15%)
Website down	19	(14%)
Different button label	11	(8%)
Button is an image	10	(8%)
Login required	9	(7%)
MetaMask not supported	4	(3%)
Requires blockchain network selection	3	(2%)
Captcha	1	(1%)
Total	**133**	**100%**

Table 13: Breakdown of reasons of connection failures by our automator. The breakdown is based on the DeFi subset of our DappRadar.com dataset.