Taming The Shape Shifter: Detecting Anti-fingerprinting Browsers

Babak Amin Azad, Oleksii Starov, Pierre Laperdrix, Nick Nikiforakis

Short Paper - Taming The Shape Shifter: Detecting Anti-fingerprinting Browsers

Babak Amin Azad[1], Oleksii Starov[2], Pierre Laperdrix[3], and Nick Nikiforakis[1]

[1] Stony Brook University
[2] Palo Alto Networks
[3] CNRS / Univ. Lille / Inria

Abstract. When it comes to leaked credentials and credit card information, we observe the development and use of anti-fingerprinting browsers by malicious actors. These tools are carefully designed to evade detection, often by mimicking the browsing environment of the victim whose credentials were stolen. Even though these tools are popular in the underground markets, they have not received enough attention by researchers. In this paper, we report on the first evaluation of four underground, commercial, and research anti-fingerprinting browsers and highlight their high success rate in bypassing browser fingerprinting. Despite their success against well-known fingerprinting methods and libraries, we show that even slightest variation in the simulated fingerprint compared to the real ones can give away the presence of anti-fingerprinting tools. As a result, we provide techniques and fingerprint-based signatures that can be used to detect the current generation of anti-fingerprinting browsers.

1 Introduction

Major database hacks and personal information leaks have been the common cyber news headline for the past couple of years. Haveibeenpwned[4], the website that hosts the records of publicly known credential leaks, currently hosts 428 instances of credential leakage from different websites, including some highly popular (e.g. Linkedin and Dropbox). The number of accounts affected by these leaked credentials adds up to over 773 million accounts.

In a similar fashion, the online shopping industry has been the prime target of attackers. In 2019, over 180,000 websites were successfully attacked by Magecart hackers [11]. By implanting malicious JavaScript code on hacked websites, attackers behind these operations steal credit card and payment information of clients upon checkout. According to statistics from the security industry [11], these attacks have so far affected more than 2 million users.

The stolen credentials and credit card information typically end up being sold in bulk in the underground markets [30]. Verification and monetization of the stolen information at scale requires specific tools. Automation is also a vital part of these malicious operations as the size of the data that needs to be verified and then abused becomes increasingly larger. As a result, malicious actors have built automation tools to speed up this process. The existing anti-bot and fraud detection tools and services heavily rely on

[4] https://haveibeenpwned.com/

browser fingerprinting [13]. In order to bypass these mechanisms, malicious actors use specialized browsers that enable them to easily switch fingerprints or simulate a target browsing environment and evade detection. We assembled our list of anti-fingerprinting browsers by searching the underground markets for the tools that malicious actors use, as well as commercial and research projects that promise to defend against tracking. Success stories (e.g., reaching over 90% success rate in carding attempts) and tutorials on configuring and efficiently using these browsers are widely available on different carding forums [1, 2, 9, 10]. Malicious actors use these forums to trade the stolen credit card information and share their latest tips on successful cashout strategies.

Tools such as AntiDetect [22] and Fraudfox [21] are commonly incorporated to mask the browser fingerprints of attackers and evade detection from tools that look for known good (i.e. belonging to a specific benign user) or known bad (i.e. belonging to a previously seen attacker) fingerprints. These browsers not only enable attackers to switch browser fingerprints, they also give them the ability to mimic a victim's environment, such as, setting their timezone and screen resolution to match the victim when visiting websites to make fraudulent purchases or access the hacked accounts.

Even though these tools are popular among attackers, they have not received the attention they deserve from the research community. In this paper, we study the techniques that these tools incorporate to remain undetected and quantify their effectiveness against state-of-the-art, in browser fingerprinting. After analyzing the fingerprintable surface of these tools, we show that we were able to devise fingerprinting-based signatures for all of them which can be used to uniquely identify them. Our findings can be used by the existing anti-fraud systems to precisely identify the usage of anti-fingerprinting browsers.

2 Background

In a typical case of online fraud, multiple entities are involved. Usually, one party is responsible for stealing credentials, which are then sold in bulk to another party to be monetized [28]. The timeliness of these events is crucial. As the stolen information gets stale, it is more likely for the compromised websites or individual victims to have been informed about their information being stolen and invalidate their credentials. In the mean time, to prevent issues with stolen credentials, merchants who process payment information started to incorporate browser fingerprinting to detect fraudulent and automated browsing activities.

Companies providing fraud detection services commonly use browser-fingerprinting to track users [4, 5, 7, 27]. By collecting information from users' web browsers, these services build browsing profiles of normal users. This information is then used to filter out fraudulent requests.

State-of-the-art browser fingerprinting identifies users by leveraging features such as HTTP headers and available JavaScript APIs [16, 24]. The act of fingerprinting transcends the actual browser, enabling the identification of the operating system and the underlying hardware [15]. This is typically achieved based on the characteristics of rendered images within an HTML canvas element [14, 25]. Other researchers have focused on other parts of the browsing environment to build more robust fingerprints by extracting the list of available fonts and browser extensions [18, 29]. Fingerprintjs2 [32],

a well-known browser fingerprinting library, compiles the previously mentioned fingerprinting methods in a JavaScript module that can be integrated with any website to collect browser fingerprints of its visitors. Lastly, behavioral features of the user like the use of clicks or touch can be collected to separate interactive user activity from that of an automated client.

3 Anti-fingerprinting browsers

To battle fingerprinting, anti-fingerprinting browsers capable of modifying the content of their fingerprint were created. We categorize the browser fingerprint modification schemes into three groups. Each group has its own benefits and drawbacks as we discuss below:

- **JavaScript Injection:** In this method, JavaScript is injected into all webpages loaded by the browser. This way, JavaScript properties and methods are overwritten to send different information to servers. For example, when a script wants to access `navigator.userAgent` or render a canvas image, it will find the newly injected version instead of the default one. The strength of this approach is the ease of deployment and maintainability. However, prior work has shown that these spoofing extensions may not offer the best protection against fingerprinting as they often present incomplete coverage of JavaScript objects and can create impossible configurations [26].
- **Native Spoofing:** Native spoofing modifies the source code of the browser to return modified values. For some attributes, changing the sent value is as simple as rewriting a string but for other methods like canvas fingerprinting, successful modifications require a deeper understanding of a browser's codebase to find the right methods and modify them appropriately. The strength of this solution is that it can be hard to detect as an inspection of the Document Object Model (DOM) is not sufficient to detect traces of spoofing. However, the downside is that the cost of maintenance can be high, requiring a complete rebuild of the browser after each update.
- **Recreating Complete Environments:** This method consists of utilizing a virtualized browsing environment with a desired configuration on top of the host system. The advantage of this method is that the fingerprint presented to servers is genuine as the components truly run on the system. For the same reason, no impossible configurations can result from such an approach. On the downside, this approach requires more system resources compared to a simple browser extension or a modified browser.

In this section, we analyze research, commercial, and underground tools against fingerprinting, in order to understand whether masking the true fingerprint of a device can help bypass current fingerprinting techniques. Next, we list the tools that are included in this study along with the anti-fingerprinting mechanism they use.

AntiDetect and Fraudfox [JavaScript Injection]. AntiDetect is one of the first tools that surfaced online against browser fingerprinting, gaining visibility from a 2015 article [3]. AntiDetect uses JavaScript injection and relies on a browser extension to change the exhibited browser fingerprint. To improve usability, users are presented with

an interface where they can choose a profile from a pool of existing browser fingerprint profiles. Fraudfox appeared at approximately the same time as AntiDetect and works in a similar fashion by providing an interface to users for selecting the fingerprint they want to expose [21]. Fraudfox offers the option to modify several attributes separately and also targets advanced techniques, such as, font fingerprinting. It uses a custom Windows XP virtual machine and a tool named *OSfuscate* to change the TCP/IP fingerprint of the system in order to confuse `nmap`-like tools that can identify OSes based on the structure of network packets.

Mimic [Native Spoofing]. Mimic is a modified Chrome browser that uses native spoofing to modify its fingerprint [8]. Users can generate various profiles and activate the desired fingerprinting protection. One particularly interesting feature of Mimic is that it gives users the option to either block, or introduce noise into some fingerprinting-related APIs. In contrast to the previously mentioned underground tools, Mimic takes a different approach and advertises itself as a generic solution against browser fingerprinting that can be used for marketing, journalism, cyber investigation, and even web scraping activities.

Blink [Recreating Complete Environments] Blink is a moving-target-style defense against browser fingerprinting. Proposed by Laperdrix et al. [23], this tool assembles a set of components at runtime into a virtual machine. Upon each execution, the virtual machine's environment is modified with new configurations (e.g., timezone, available fonts, etc.) in order to generate an organic browser fingerprint. This guarantees that the exhibited fingerprint is coherent compared to the other tools where the artificial combination of browser properties can easily result in impossible configurations.

A full comparison of the tools along with the exact fingerprinting techniques that each of them counters, can be found in Table 1. The main tactic that these tools incorporate against detection is frequent rotation of valid fingerprints. That is, the common elements in browser fingerprints as mentioned both in the literature and popular opensource fingerprinting libraries such as Fingerprintjs2, are configurable.

These values are faked through a large list of valid fingerprints that is either shipped with these browsers or can be easily generated through their interface. For instance, AntiDetect comes with over 4,000 profiles and Fraudfox includes profiles with 90 user-agents and 5 browsers and 6 operating systems. Moreover, users can choose to add noise to certain APIs such as audio context and the canvas API. This variety makes it hard to derive features from the common fingerprinting libraries to uniquely identify these browsers. Interestingly, Fraudfox has been tested against popular browser fingerprinting tools and the successful rotation of fingerprints and removal of tracking information (e.g., Evercookies [6]) has been verified in the underground carding forums [10].

All of the studied anti-fingerprinting browsers, except Blink, which is discussed separately in Section 4, modify or add noise to the existing browser properties. We will discuss in more detail how this type of modification will inherently introduce inconsistencies and demonstrate concrete examples of these inconsistencies and use them to build signatures that uniquely identify these browsers in Section 4.

Table 1: *Overview of the studied tools with the fingerprinting techniques they counter*

Tool	**AntiDetect**	**Fraudfox**	**Mimic**	**Blink**
Type	Injection	Injection	Native	Recreation
Tested version	7.1	1.5.1	1.4.8	1.0
Number of profiles or components	>4,000	600 fonts, 90 user-agents, 85 plugins, 5 browsers and 6 OS	1,000	2,762 fonts, 39 plugins, 6 browsers and 4 OS
Browser used	Firefox 41-48	Firefox 41	Chrome 61	Latest versions of Chrome and Firefox
Network	-	Proxy through *SocksCap64* + Obfuscation of OS Network packet through *OSfuscate*	Built-in proxy management (HTTP, Socks5)	Built-in support for Tor
User Agent	✔	✔	✔	✔
Language	✔	✔	✔	
Screen	✔	✔	✔	
Navigator	✔	✔	✔	✔
Timezone	✔	✔	✔	✔
Date			✔	
Fonts		✔	✔	✔
Plugins	✔	✔	✔	✔
Media devices			✔	
Canvas	Noise (letters in strings)	Noise (fonts and colors)	Noise (fonts and colors)	Noise (change of OS)
WebGL	Blocked	Blocked	Only vendor and renderer	Noise (change of OS)
WebRTC	✔		Block or fake IP address	
Geolocation	✔		✔	
Hardware Concurrency			✔	

4 Detecting the anti-fingerprinting tools

To extract unique characteristics that can be used to uniquely identify each browser, we analyzed each tool using the techniques described by Nikiforakis et al. [26] and Acar et al. [12]. We investigate built-in JavaScript objects, such as, `navigator` and `screen` with and without anti-fingerprinting mechanisms, looking for inconsistencies. According to Vastel et al., existing bot detection schemes already use similar techniques to detect the presence of impossible fingerprints [34]. To the best of our knowledge, we are the first to report on the fingerprintability of dedicated anti-fingerprinting tools.

• **AntiDetect** Since AntiDetect relies on a browser extension, a single line of JavaScript is sufficient to detect injected values. Notably, objects created through JavaScript are easily identifiable as they only contain a `toString` function. In Listing 1 (top), we can clearly see the `getGamepads` function written by the developers to modify the returned value as if it was a native one.

```
navigator.getGamepads.toString.toString()
//Returns "function () { return "function getGamepads() { [
    native code] }";}"
//
//Standard Firefox returns
//"function toString() {
//    [native code]
//}"
```

```
CanvasRenderingContext2D.prototype.__lookupSetter__("strokeStyle
    ").toString()
//Returns
//"function (){
//"use strict";
//this.strokeStyle=settings.strokeStyle}"
//
//Standard Firefox returns
//"function set strokeStyle() {
//    [native code]
//}"

canvas = document.createElement("canvas");
canvasContext = canvas.getContext("2d");
canvasContext.fillStyle = "#ff6600";
canvasContext.fillStyle.toString();
//Returns the color set by the user: "#71cda0"
//Standard Firefox returns the color from the script: "#ff6600"
```

Listing 1: *Detecting JavaScript injection performed by AntiDetect (top) and Fraudfox (bottom)*

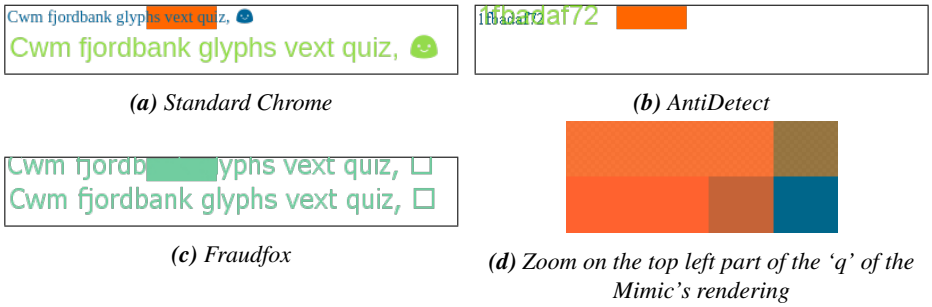
Like other tools relying on JavaScript injection, inconsistencies in fingerprints are possible and frequent. One example is when AntiDetect launches a Chrome profile where one can observe the presence of both *webkit* and *moz* prefixed properties which is impossible as these belong to two different rendering engines. Another example is a mismatch between two attributes where the user-agent reports a 64-bit OS and the `navigator.platform` indicates a 32-bit one.

• **Fraudfox** presents the same shortcomings as AntiDetect as it also relies on the same spoofing method. However, one needs to look elsewhere to find traces of JavaScript injection. As shown in Listing 1 (bottom), the developers directly poison the prototype of specific objects. One can also easily find the parameters that are set in the tool's interface like the exact filling color of the canvas API. This could, in fact, act as a long-time identifier if the user always reuses the same profile without regularly updating the canvas color. Finally, Fraudfox has its own set of inconsistencies. For example, Chrome profiles present *moz*-prefixed properties but no *webkit* ones. Mac profiles show *.dll* extension for plugins instead of *.plugin*.

• **Mimic** is harder to detect compared to the two previous solutions because it does not rely on JavaScript injection. However, the browser is still identifiable through some

unique inconsistencies that come from its database of fingerprints. When spoofing the WebGL Renderer, Mimic always add the *ANGLE* string in front of every value. However, this string can only be found on Windows as Chrome uses the ANGLE backend on this operating system to translate OpenGL API calls to DirectX. On Linux, plugins with the *.so* extension are visible creating an inconsistency if a Windows or a Mac profile is selected. Finally, Mimic presents an incorrect priority in the HTTP language header. The second language should present a priority of 0.9 ("en-US,en;en;q=0.9") but Mimic returns one of 0.8 ("en-US,en;en;q=0.8"). Changing the priority is easily fixable in the profile database but it shows that the smallest detail can render a tool identifiable.

Focus on canvas poisoning. Each tool also has its own canvas poisoning technique, which as we demonstrate is identifiable. Figure 2 illustrates them.



(a) *Standard Chrome*

(b) *AntiDetect*

(c) *Fraudfox*

(d) *Zoom on the top left part of the 'q' of the Mimic's rendering*

Fig. 2: *Renderings of the same canvas test*

AntiDetect changes the letters of a given string and their position. Fraudfox modifies the colors set by a script. This is directly configurable in the interface of the tool. Moreover, since the tool runs on Windows XP, the OS does not have any fonts that support emojis (presence of a green square at the end of the strings). Mimic is different from the other two as the modification is almost invisible for the user. Mimic introduces a small amount of noise but an in-depth analysis reveals that the transparency of some pixels were changed (on the zoomed-in image, the top half of the orange rectangle is more transparent than the bottom half).

Overall, our findings demonstrate that a combination of several tests is sufficient to precisely identify all evaluated anti-fingerprinting tools. The quirks discovered can be corrected but our results confirm that it is difficult to design an anti-fingerprinting tool that is not detectable. For both JavaScript injection and native spoofing, the smallest oversight can make the user stand out, be marked as malicious and invalidate the offered protection.

Blink and the recreation of complete environments

In this section, we showed how the operators of anti-fraud systems can fingerprint anti-fingerprinting tools, based on the latter's inability of perfectly mimicking a non-native browsing environment. Blink, the research prototype by Laperdrix et al. [23] that we introduced in Section 3, sets itself apart from the rest by the fact that it does not attempt to mimick a foreign environment. Instead, Blink assembles a real environment with

different components and launches that environment in a virtual machine. As such, none of the techniques presented in this section can be used to detect Blink since there is no mimicking involved and therefore no inconsistencies to be discovered.

Despite Blink's attractiveness for defeating fingerprinting-based, unwanted online tracking (since users can keep changing their fingerprints and therefore break the linking of browser sessions), we argue that Blink's utility is limited for attackers. This is because, an attacker who tries to match the fingerprinting of a victim user, must utilize Blink to recreate the entire browsing environment of their victim. This requires not just the installation of the appropriate software, but even the purchase of the appropriate hardware (e.g. to match the number of threads in the victim's CPU and how the victim's graphics card renders complex 3D scenes). All of this is clearly possible for highly targeted attacks but also highly unlikely for the monetization of credentials, since the investment in assembling the right environment can exceed the profit from the stolen credentials.

5 Related Work

Prior work can be split into the study of underground markets, browser fingerprinting, and bot-based fraud detection.

Singh et al. studied the underground ecosystem of credit card fraud [28]. They describe the different methods that attackers use to steal credit card information. These methods range from POS malware to exploitation of a vulnerability. Given the difficulty and risk associated with monetizing stolen credentials, attackers often resort to selling these illicitly obtained credentials to other attackers specializing in monetization. The authors then go over the existing channels to monetize the cards (e.g. by delivering high-end goods purchased with stolen credentials to unsuspecting users who believe they are working for a shipping company and will then re-ship the goods to another destination [19]). Other works focused on trafficking of fraudulent twitter accounts in the underground markets [31]. Fallmann et al. discussed their finding on probing these markets [17] and Thomas et al. assessed the effect of data breaches on the activities of underground markets [30].

In the realm of browser fingerprinting, researchers keep identifying features that can be extracted from browsers and make browser fingerprints more robust [14, 15, 18, 25, 29, 33]. As fingerprinting-based fraud detection tools incorporate these features into their techniques, the tools used by attackers must also account for them (such as accounting for canvas-based fingerprinting, as described in Section 4).

One of the challenges in the study of JavaScript files and fingerprinting scripts is instrumenting the various API calls and monitoring them. VisibleV8 is a Chromium based browser that is easy to maintain over time and provides the ability to monitor JavaScript API calls [20]. The authors used their customized browser to analyze the prevalence of scripts that query for bot and browser automation artifacts on popular Alexa websites.

6 Conclusion

In this paper, we showed that anti-fingerprinting tools are capable of bypassing the protection of state-of-the-art fingerprinting techniques by masking the components that

are queried by fingerprinting libraries. We analyzed their masking techniques (i.e., JavaScript injection, native spoofing, and the recreation of complete environments) and described the process of identifying fingerprinting-based inconsistencies which can be used to identify them and block them. Our analysis showed that all tools that attempt to mimick non-native environments are unique fingerprintable and therefore can be identified by anti-fraud systems, through the use of our proposed fingerprinting vectors. Finally, we discussed the difficulty of fingerprinting tools that are based on the recreation of browsing environments and the reasons why these tools are highly unlikely to be used in generic, non-targeted attacks.

Acknowledgements: We thank the anonymous reviewers for their helpful feedback. This work was partially supported by a gift from Amazon and the National Science Foundation (NSF) under grants CNS-1813974, CNS-1617902, and CMMI-1842020.

References

1. AntiDetect tool, only way to cashout from stolen credit cards (2015), https://www.ehacking.net/2015/03/antidetect-tool-only-way-to-cashout.html
2. Fraudfox makes it easier for thieves to empty bank accounts (2015), https://www.pcworld.com/article/2872372/this-tool-may-make-it-easier-for-thieves-to-empty-bank-accounts.html
3. Post by Brian Krebs on AntiDetect (2015), https://krebsonsecurity.com/tag/antidetect/
4. DataDome: Protect your website from bot traffic (2017), https://datadome.co/
5. Distil Networks: Bot Mitigation & API Security (2017), https://www.distilnetworks.com/
6. Evercookie (2017), https://github.com/samyk/evercookie
7. ShieldSquare: Bot Mitigation & Protection (2017), https://www.shieldsquare.com/
8. Multilogin – The Most Advanced Browser Fingerprinting Protection Ever Created - Enter Mimic (2018), https://multiloginapp.com/advanced-browser-fingerprinting-protection-ever-created-enter-mimic/
9. AntiDetect 7 and FraudFox VM, best carder protection (2019), https://imgur.com/a/ycxFTtz and https://bazaar.blockstamp.market/listings/view/QmX9VGTz2HziSqL7kjNSGjPe8UHDrdyyxZwXyQbBgTbWcN-antidetect-7-fraudfox-vm-full-version-of-both-applications-best
10. Fraudfox tool in and underground carding forum (2019), https://imgur.com/a/6xmYPgN and https://www.verifiedcarder.ws/threads/fraudfox-tool-cracked.21485/
11. Magecart Skimmers Spotted on 2M Websites (2019), https://www.darkreading.com/endpoint/magecart-skimmers-spotted-on-2m-websites/d/d-id/1336011
12. Acar, G., Juarez, M., Nikiforakis, N., Diaz, C., Gürses, S., Piessens, F., Preneel, B.: FPDetective: dusting the web for fingerprinters. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (2013)
13. AminAzad, B., Starov, O., Laperdrix, P., Nikiforakis, N.: Web Runer 2049: Evaluating Third-Party Anti-bot Services. In: DIMVA (2020)
14. Bursztein, E., Malyshev, A., Pietraszek, T., Thomas, K.: Picasso: Lightweight device class fingerprinting for web clients. In: Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices (2016)

15. Cao, Y., Li, S., Wijmans, E.: (Cross-)Browser Fingerprinting via OS and Hardware Level Features. In: NDSS (2017)
16. Eckersley, P.: How unique is your web browser? In: Privacy Enhancing Technologies. Springer (2010)
17. Fallmann, H., Wondracek, G., Platzer, C.: Covertly probing underground economy marketplaces. In: DIMVA (2010)
18. Fifield, D., Egelman, S.: Fingerprinting web users through font metrics. In: Proceedings of the 19th international conference on Financial Cryptography and Data Security (2015)
19. Hao, S., Borgolte, K., Nikiforakis, N., Stringhini, G., Egele, M., Eubanks, M., Krebs, B., Vigna, G.: Drops for Stuff: An Analysis of Reshipping Mule Scams. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security (2015)
20. Jueckstock, J., Kapravelos, A.: VisibleV8: In-browser monitoring of JavaScript in the wild. Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC (2019)
21. Kirk, J.: This tool may make it easier for thieves to empty bank accounts. https://www.csoonline.com/article/2871248/fraud-prevention/this-tool-may-make-it-easier-for-thieves-to-empty-bank-accounts.html
22. Krebs, B.: 'AntiDetect' Helps Thieves Hide Digital Fingerprints. https://krebsonsecurity.com/2015/03/antidetect-helps-thieves-hide-digital-fingerprints/
23. Laperdrix, P., Rudametkin, W., Baudry, B.: Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification. In: 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2015) (2015)
24. Laperdrix, P., Rudametkin, W., Baudry, B.: Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. In: 37th IEEE Symposium on Security and Privacy (2016)
25. Mowery, K., Shacham, H.: Pixel Perfect: Fingerprinting Canvas in HTML5. In: Proceedings of W2SP 2012 (2012)
26. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy (2013)
27. PerimeterX: Anti Bot Protection - Protect Against Bot Attacks. https://www.perimeterx.com/
28. Singh, A.: The Underground Ecosystem Of Credit Card Fraud. Black Hat Asia (2015)
29. Starov, O., Nikiforakis, N.: XHOUND: Quantifying the Fingerprintability of Browser Extensions. In: 38th IEEE Symposium on Security and Privacy (2017)
30. Thomas, K., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., Markov, Y., Comanescu, O., Eranti, V., Moscicki, A., et al.: Data breaches, phishing, or malware? understanding the risks of stolen credentials. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2017)
31. Thomas, K., McCoy, D., Grier, C., Kolcz, A., Paxson, V.: Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In: USENIX Security (2013)
32. Vasilyev, V.: FingerprintJS2: Modern & flexible browser fingerprinting library. https://github.com/Valve/fingerprintjs2
33. Vastel, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: FP-STALKER: Tracking Browser Fingerprint Evolutions. In: 39th IEEE Symposium on Security and Privacy (2018)
34. Vastel, A., Rudametkin, W., Rouvoy, R., Blanc, X.: FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In: Starov, O., Kapravelos, A., Nikiforakis, N. (eds.) NDSS Workshop on Measurements, Attacks, and Defenses for the Web (2020)