Device Fingerprinting for User Authentication in Web Applications *

Ang Wei Ming
e0177154@u.nus.edu

Chan Liang Fei Jeremy
e0053073@u.nus.edu

Ong Yong Siang
e0032188@u.nus.edu

Tan Zhen Yong
tan.zhenyong@u.nus.edu

Xu Chen
xuchen@u.nus.edu

ABSTRACT

Device fingerprinting is a modern technique which, instead of using traditional user identifiers (e.g. cookies and local storage), leverages on existing information available from user's device. In this paper, we investigates the various state-of-the-art technologies for device fingerprinting and authentication systems that utilize such fingerprint information. We analyzes the trade-off between the uniqueness of the identifier generated and ease of use for authentication. Lastly, we also build a simple authentication system that can be reused and extended in real-world web applications.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Authentication*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Security

Keywords

Authentication, device fingerprinting, web authentication

1. INTRODUCTION

1.1 Background

Traditional password-based authentication systems require a user to provide two pieces of information: an identifier and a password proving their identity. The weaknesses of such a system largely hinge on the security of said password. For example, users may set weak passwords, allowing the security of their account to be compromised as attackers use brute-force attacks to gain access. On the server side, the storage of these passwords or their representations also prove to be a problem, as they present themselves as valuable targets for attackers looking to compromise a large number of users.

In this paper, we explore an alternative method to identifying and authenticating users using device fingerprinting. Device fingerprinting refers to techniques that collect information about the attributes and configuration of a device for the purpose of constructing a unique identifier for that device. Since this identifier is unique and can be generated without the intervention of the user, it can be used to build a password-less authentication system where the user need only remember their username. The system will then compute a unique identifier for the device and check that with the one stored in its database, allowing the user access if the fingerprints match over a certain threshold.

1.2 What is device fingerprinting?

Device fingerprinting has its roots in its use by online analytics and advertising companies. In order to track a user across different websites, and to serve them relevant advertisements, these companies require a way to be able to identify the same device across different websites. The traditional method relies on placing a tracking cookie that contains a unique identifier on the device when it first visits a website that contains their analytics or advertising script. This would then allow the company to associate visits on different websites that they manage to the same user, and correlate data from those different visits knowing that they come from the same device.

However, this traditional cookie based method presents a few problems, mainly in the form of persistence. Cookies are generally not guaranteed to last for any amount of time, since they can be cleared any time by the user. More recently, as privacy concerns over tracking has arisen, some browsers such as Safari have taken to blocking tracking cookies by default, making cookie based identifiers less reliable.

Therefore, these companies have turned to device fingerprinting as a more effective method of device identification. By computing a unique identifier from the attributes of the device, advertisers can now store a identifier that would persist across users clearing their cookies, since re-computation of the identifier on the same device would yield the same hash. More importantly, device fingerprinting could generate identifiers that were consistent across different browsers

*(Does NOT produce the permission block, copyright information nor page numbering). For use with ACM_PROC_ARTICLE-SP.CLS. Supported by ACM.

on the same machine, or even across minor software and hardware changes.

This enhanced persistence, as well as its use by advertising and analytics companies for the purposes of tracking users, means that device fingerprinting is traditionally regarded as privacy invading, and as a technique that undermines security. However, the same properties that makes device fingerprinting attractive for tracking users also mean that it is a viable authentication method. Since it requires no input, it is far easier for a user to authenticate themselves through a device fingerprinting based authentication system. Moreover, since the fingerprint is tied to the inherent attributes of the device, it is difficult for an attacker to trivially attack and forge an identifier the same way they might brute force a password, since the interaction between the device fingerprint and the device itself may be hard to simulate.

Therefore, we would like to explore the usage of device fingerprinting as a security enhancing technique, specifically in constructing a authentication system that is able to authenticate users with their unique device fingerprint, thus eliminating the need for passwords.

2. FINGERPRINT TECHNOLOGY

2.1 Overview of Fingerprinting Techniques

Browser fingerprinting technology usually involves the process of collecting user data from various level of the device, such as user's browser information and the operating system information. [7] In contrast with the state-of-the-art Cookies methodology, such fingerprinting is said to be "stateless" because no additional information (e.g., cookies, tokens) is stored on the user's device.

With the amount of information given away from the browser, they can be identified in a stable and measurable manner [10] by embedding scripts that collect and process those information into the web page, as shown by Nikiforaskis. Eckersley from the Panopticlick project [12] demonstrates that browser fingerprinting technology is a fairly effective user tracking tool, drawing attention and usage from many advertising companies. Almost 500K browsers have their information scanned and recorded on the Panopticlick website, and around 83.6% of them can be identified uniquely. More recently, newer techniques have been discovered that use JavaScript to reveal more information such as operating system and microarchitecture, and even hardware information such as screen resolution. [8] [6] As HTML5 and WebGL becomes more popular tools for websites, new sources of fingerprinting information is added to the arsenal. More importantly, such fingerprinting information is no longer restricted to browser properties because different system configurations will behave differently when rendering images through the canvas component.[9]

Contrary to the common notion of information leaking, none of the above-mentioned practices involves complicated mechanisms such as password hacking, or any sort of attack. The is because so much fine-grained information about the browser and the user can be retrieved as a necessity for web applications to be able to serve content based on different devices effectively. For example, responsive web applications need to know the dimension of the screen, while a video streaming service may need to know the presence of plugins like Adobe Flash. In order to serve these various demands, the browser expose APIs for developers to query such information, e.g., the built-in `navigator` and `screen` objects in JavaScript which reveal the browser name, version, platform information. In addition, browser plugins which add features to the browser also have access to information about the computer that can expose more accurate identifiers.

2.2 Survey of Fingerprinting Techniques

According to Acar et al. [2], browser fingerprinting technologies can be categorized by the approach they use to collect information: (1) JavaScript-based, (2) Plugin-based, (3) Extension-based, (4) Header-based and Server-side. The rest of this section will go through in more detail about each of them.

JavaScript-Based Fingerprinting. JavaScript is the default scripting language for almost all interactive web applications now. It provides functionality such as rendering of animation, submitting of user forms, and altering the content of the page dynamically. Other than the informative objects mentioned earlier (`navigator` and `screen`), JavaScript can also be used to identify users indirectly, such as obtaining browsing history or fine-grained timing information of the JavaScript engine at different browsers.

Plugin-Based Fingerprinting. Plugins are equipped with capabilities to access the software and hardware properties of the underlying system. One of these common plugins is Flash that is used for displaying videos and games in the `.swf` format. In order for the web pages to use Flash effectively, Flash exposes APIs to query information about the operating system, as well as the specific version of Flash installed. Moreover, Flash also exposes APIs that allow one to list all the system fonts, which can vary from user to user. The difference in those fonts installed can again be used as an identifier for the user. Therefore, the more plugins a user has installed, the more information can be exposed to fingerprinting software.

Extension-Based Fingerprinting. A browser extension is a special software component that adds extra functionality to the browser, such as RSS reader or ad-blockers. Extensions are different from plugins as extensions provide additional functionality in the browser itself while plugins enable additional functionality for rendering a web page. Take one of the extensions that expose additional information as an example: `NoScript` [11], which is an extension that blocks executions of Web objects from any URL but those white-listed. As explained by Mowery et al. [8], one can prepare a list of URLs in the web page and see which of them are listed on the white-list. The white-list of the extension then can be used as an identifier to differentiate users from each other.

Header-Based and Server-Side. This is one of the most basic fingerprinting techniques where the fingerprinters use

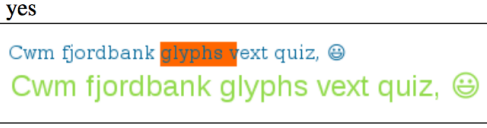
Attribute	Source	Distinct values	Unique values	Example
User agent	HTTP header	11,237	6,559	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36
Accept	HTTP header	131	62	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Content encoding	HTTP header	42	11	gzip, deflate, sdch
Content language	HTTP header	4,694	2,887	en-us,en;q=0.5
List of plugins	JavaScript	47,057	39,797	Plugin 1: Chrome PDF Viewer. Plugin 2: Chrome Remote Desktop Viewer. Plugin 3: Native Client. Plugin 4: Shockwave Flash...
Cookies enabled	JavaScript	2	0	yes
Use of local/session storage	JavaScript	2	0	yes
Timezone	JavaScript	55	6	-60 (UTC+1)
Screen resolution and color depth	JavaScript	2,689	1,666	1920x1200x24
List of fonts	Flash plugin	36,202	31,007	Abyssinica SIL,Aharoni CLM,AR PL UMing CN,AR PL UMing HK,AR PL UMing TW...
List of HTTP headers	HTTP headers	1,182	525	Referer X-Forwarded-For Connection Accept Cookie Accept-Language Accept-Encoding User-Agent Host
Platform	JavaScript	187	99	Linux x86_64
Do Not Track	JavaScript	7	0	yes
Canvas	JavaScript	8,375	5,533	
WebGL Vendor	JavaScript	26	2	NVIDIA Corporation
WebGL Renderer	JavaScript	1,732	649	GeForce GTX 650 Ti/PCIe/SSE2
Use of an ad blocker	JavaScript	2	0	no

Figure 1: List of Attributes Fingerprinted for AmIUnique.org

information from the HTTP headers as identifiers. The HTTP headers contain information such as the IP address or Accept Headers in the request. Such information is readily available for all browsers and sent to servers in the HTTP request.

Combining all these different practices together, the fingerprinter can obtain a large amount of information on a user. Laperdrix et al. [7] demonstrates that one can combine at least 17 attributes from various sources to create unique device fingerprints. As shown in Figure 1, the possible number of combinations of the different values from those parameters is rather large, and the study successfully identified 81% users out of 118,000 users.

2.3 Fingerprinting for Authentication

Despite the existing shortcomings of a password-based authentication methodology, it still remains the dominant method for web authentication. There are multiple ways to enhance the security of password-based authentication, such as the Multi-Factor Authentication model. Device fingerprinting can also be used to augment password-based authentication. A number of commercial fraud detection services have already started using device fingerprinting in identifying fraudulent transactions.[14] [1] A user's device fingerprint is collected automatically and stored alongside their password. The server can use a wide range of fingerprinting methods such as those described in earlier. The server can then verify the device fingerprint later during authentication.

The following sections will discuss the areas to consider when device fingerprinting is used to strengthen web authentication.

2.3.1 Desired Properties

In order to effectively augment user authentication with device fingerprinting, there are a number of properties that the device fingerprint should hold. A set of inappropriate fingerprinting practices may incur too many false negatives in the authentication process, denying legitimate users access or falling back to the most conservative authentication process too frequently. Compared to the general usage of fingerprint in advertising, fingerprinting for authentication requires a higher level of accuracy in distinguishing users. A 80% or 90% accuracy is simply not enough. [3]

In this section, we will be discussing the properties that

make a set of fingerprinting practices a good augmentation for the authentication process:

Stability across different scenarios. It is normal for a user to have their device fingerprint change over time or in different scenarios. For example, a user may be traveling across different time zones or switch between devices. A significant change in the device fingerprints will usually lead to a fallback on a less convenient but reliable authentication process, such as a one-time code sent to the user's phone or email. Therefore, using fingerprint practices that are reliable over time for a particular user is preferred. Having more diverse metrics collected from different fingerprint practices will allow a small subset of values to vary without decreasing the overall stability of the fingerprint.

Repeatability on same device. Repeatability is defined as the property whereby the same fingerprinting techniques generate the same results if the software, hardware, and network configuration of a device does not change. [3]. It is related to but different from the stability requirement, which is primarily concerned with changes in the device configuration. For some fingerprinting techniques, such as measuring the JavaScript rendering performance (determined and related to the underlying GPU/CPU performance and load at a specific instance), the same device may generate different results with other applications running: for example, trying to login when having a video compilation job running in the background might produce different fingerprints.

Resistance to spoofing. An attacker can potentially spoof a device fingerprint by mimicking the properties or behaviour of the targeted device. Some fingerprinting techniques have a low threshold for this kind of spoofing attack. For example, if the server relies on running a particular JavaScript script at the client side and retrieving the result from the client, then an attack can potentially replay the results to pretend to be the victim. On the other hand, some fingerprint technique are more resistant to such spoofing, such as the IP address of the sources[5] or timestamps indicating clock skew from MAC protocol packets [4].

There are many other requirements that are not listed above in detail, such as the overhead in fingerprint collection, the convenience and level of action required from the user, and personalized identification information (the uniqueness of the fingerprints).

2.3.2 Fingerprint Authentication Solution

In this section, we discuss two strategies of using fingerprinting to augment authentication: (1) fingerprinting at session initialization, (2) fingerprinting throughout an authenticated session, (3) fingerprinting as the primary authentication method.

Fingerprint at the start of session. Fingerprint information is used as an additional authentication dimension alongside the normal password-based authentication (or other primary authentication method), and an authenticated session requires the matching of both the primary response as well as the fingerprint data stored. In order for this strategy to work, the server must be able to store persistently the fingerprint data which is given by the user at registration. Such a strategy can be combined with two-factor authentication. For example, in the event where the device used for receiving SMS is stolen, a fingerprint of the stolen device might reveal the unusual state of it as the fingerprint check failed, which leads to additional security checks.

In general, this strategy can prevent or make the below attacks more difficult:

- **Naive Attack** A naive attack is defined as a conventional password-guessing attack that does not attempt to provide a matching fingerprint. A device fingerprint can effectively counter this attack since the attacker is assumed to launch the attack in a rather different environment.
- **Optimized Password and Fingerprint Guessing**: The attacker might prepare common fingerprint profiles and use it along with to launch a naive password-guessing attack. A device fingerprint can effectively reduce the chances of success under this attack given that the fingerprint information expands the guessing space, and the attacker is only allowed a limited number of trials.
- **Fingerprint Spoofing and Phishing**: Just like a normal password spoofing and phishing attack, the attacker can collect fingerprint information from users on top of the password. The level of spoofing resistance in the fingerprint used will then largely determine the feasibility of this attack.

Fingerprint throughout an authenticated session. In a typical password-based web authentication scheme, the server will return a cookie upon receiving a valid username and password so that the client will use the cookie as a proof of authentication in every request later on. This could be strengthened by validating the fingerprint information at each request. If the set of fingerprints that the server collects remains static, the re-computation of the fingerprint at each request is redundant.

A simple replay attack can be effectively prevented by hashing the fingerprint at the client with a counter before sending to the server, as demonstrated by [13]. However, for a more powerful attack such as XSS attack, it is less effective. The attacker can launch a XSS targeted at the victim, and steal the plain-text device fingerprint and the counter at the client-side. Once obtaining the fingerprint plain-text, the attacker can reproduce the hash by incrementing the counter and recomputing. In addition, such an attack is also possible even if the server dynamically changes the set of fingerprint attributes in each request [15] as the attacker can simply collect additional information about the target device once the hijacking is successful.

Fingerprint as the primary authentication method. The previous two approaches focus on how fingerprint can easily extend the current web authentication methods. We now present an approach where fingerprinting is used as the main authentication method, with supports from 2FA (like email, or SMS). Using fingerprint as the main authentication method requires a few assumptions to be held true:

- Each device should have a unique fingerprint that can be associated with a user's account. From the previous sections results, while this is still not entirely true, it is believed that the potential huge number of attributes from a device fingerprint should set apart each user as more features can be collected from a device with more powerful fingerprinting methods.
- Fingerprints collected from the same device at different occasions need to be relatively stable, and there needs to be ways to re-associate a device to the account: recovery mechanism such as email or SMS can be viable options.
- Even though an attacker has full knowledge of the device's hardware and software information, the attacker should not be able to spoof the device if she/he does not have access to the device.

If the above requirements can be met, device fingerprinting alone could be used in real life for account authentication. In the below sections, we will present a prototype that demonstrates the usage of fingerprint-based authentication method, and also discuss limitations that are closely related to the above requirements, and possible solutions to address them.

3. FINGERPRINT AUTH PLUGIN

This section discusses the design and implementation of our device fingerprinting technique on a web application.

The aim of our project is to showcase the possibility that a unique device fingerprint could potentially replace passwords in an authentication system. We built a web application where users can sign up and login to an account simply using an email. During registration, our web application will generate a unique device fingerprint that pairs with the email provided. Once registered, users can proceed to login using their registered email. During log-in, the application will also generate a log-in device fingerprint and the user will be authenticated based on the existence of email and the similarity between the log-in fingerprint and the registered fingerprint if the email provided exists. This single-field input authentication provides simplicity and ease for users, eliminating the need to remember passwords without compromising security.

The authentication algorithm introduces a degree of complexity whereby the similarity of two device fingerprints is evaluated based on a threshold level. This threshold level is customizable during registration (Figure 2) and explained in the table below.

The algorithm sets up for authentication by assigning a weightage to each component that make up the original fingerprint. This accounts for variations in a device fingerprint.

Security Level	Description
High	Every device component must match
Medium	A 75% match whereby different set of plugins on the same browser will be prevented from login
Low	Using different browsers on the same machine will be allowed

Figure 2: Security Levels For Demonstration

In the current implementation, an email account is tied to one device fingerprint. In other words, the user can only use the device he/she registered the account with. This would be ideal in scenarios where a firm wishes to lock-down usage of an account strictly to a machine for tightened security purposes. However, this would not be ideal on platforms where user should be able to be authenticated across different devices. Hence, we have considered various measures such as using two-factor authentication by sending a one-time key to the registered email when logged in on an unidentified device (also elaborated in Extensions section) for future releases.

3.1 Device Fingerprint Generation

The mechanism to generate a device fingerprint is the key and integral part of our project. We built a plugin, consolidated into one single Javascript file that could analyze and collect a device's information from a series of 27 components listed below. Some components are easy to get as they are avialble in JavaScript, while some are more invovled. For example, fingerprint for WebGL and Canvas required test rendering of objects dynamically to obtain the specific information. Each component is weighted according to its uniqueness across different devices. For instance, pixelRatio is given a higher percentage than userAgent as it would vary more differently. The generated array of values and its weighted percentages are sent as a JSON object to be stored in the database.

- **userAgent**: the User-Agent string of the browser eg. `Mozilla/5.0 (Windows NT 10.0; Win64)`
- **language**: the current language used in the browser, eg. `en-GB`
- **colourDepth**: the colour depth of the device in bits eg. 24.
- **deviceMemory** - the amount of memory on the device in gigabytes eg. 4
- **pixelRatio** - ratio of device pixels to CSS pixels eg. 2
- **hardwareConcurrency** - number of logical processors eg. 8

- **screenResolution** - width and height of the current screen eg. $864, 1536$
- **availableScreenResolution** - width and height of the current screen available eg. $824, 1536$
- **timezoneOffset** - offset from the GMT timezone in minutes eg. -480
- **timezone** - current timezone eg. $Asia/Singapore$
- **sessionStorage** - whether the HTML5 session storage API is available eg. $true$
- **localStorage** - whether the HTML5 localStorage API is available eg. $true$
- **indexedDb** - whether the HTML5 indexedD API is available eg. $true$
- **addBehavior** - whether addBehaviour is available eg. $false$
- **openDatabase** - whether WebSQL is available eg. $true$
- **cpuClass** - CPU class of the device eg. `Win32`
- **platform** - platform of the device eg. `Win32`
- **doNotTrack** - whether the Do Not Track setting is on eg. $false$
- **plugins** - list of plugins installed
- **canvas** - canvas fingerprint of browser
- **webgl** - WebGL fingerprint of browser
- **webglVendorAndRenderer** - WebGL vendor and renderer of browser
- **adBlock** - whether the browser has an ad blocker installed eg. $false$
- **touchSupport** - whether the device has a touchscreen interface, with no. of touchpoints eg. $10, false, false$
- **fonts** - detection of all user installed fonts
- **audio** - audio fingerprint of device
- **enumerateDevice** - list of all media input and output devices, such as microphones or webcams

The idea of a single-file Javascript plugin considers scalability whereby it could be imported into any web application for the purpose of generating a device's fingerprint.

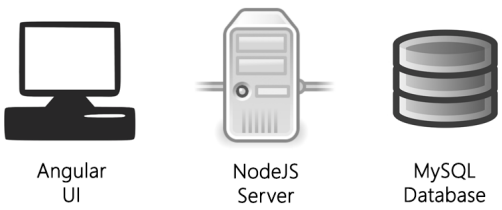


Figure 3: Demo Flow of Web Application

3.2 Demo Web Application

The Javascript plugin above for generating device fingerprint was fitted into our streamlined web application to exemplify the fingerprint-based authentication.

As shown in Figure 3, our web application consists of three components - User Interface (UI), Server and a Database.

The front-end user interface that displays the registration and login pages was built with Angular 6.0, designed with Material Design Bootstrap for a consistent and fluent design. Background animation was added to create an aesthetic appeal that blends into the "security" theme. The UI is responsible for input validations, sending and receiving data to-and-from the backend (NodeJS) server. It triggers the Javascript function of our fingerprint plugin to generate a device fingerprint and thereby package both user input email and device fingerprint into a JSON object, passing it to the NodeJS server.

The NodeJS server utilizes Express 4.0 for communication of data between front-end and database. The specific set of APIs are exposed to the UI such that only the responses and requests made by the UI would be routed for operations. For storing and retrieving of data from the database, it uses Flex.Query Processor Library to execute SQL statements to a MySQL instance. The server also stores and execute the algorithm logic for registration and login authentication.

- **Sign Up Flow:**
  1. User signs up with an email and select a security level (Figure 4)
  2. UI generates device fingerprint JSON object containing 27 device components
  3. UI sends device fingerprint and email of the registering user to backend server
  4. UI displays the table of 27-component device fingerprint to user (Figure 5)
  5. Backend stores device fingerprint and email into MySQL database
- **Login Flow:**
  1. User logs in with email
  2. UI generates device fingerprint JSON object containing 27 device components
  3. UI sends device fingerprint and email to backend server

4. Backend checks whether there exists any record for email and runs an algorithm to match each of the 27 device fingerprint components based on the security level chosen by the user
5. UI redirects to a login success page if email and device fingerprint were successfully matched. Otherwise, UI redirects to a login failure page if either email or device fingerprint failed to match

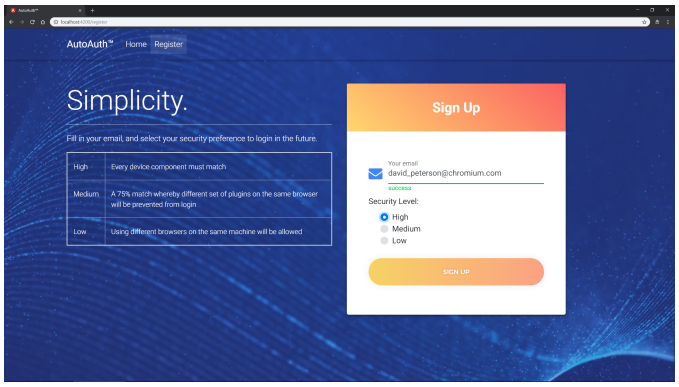


Figure 4: Registration Page

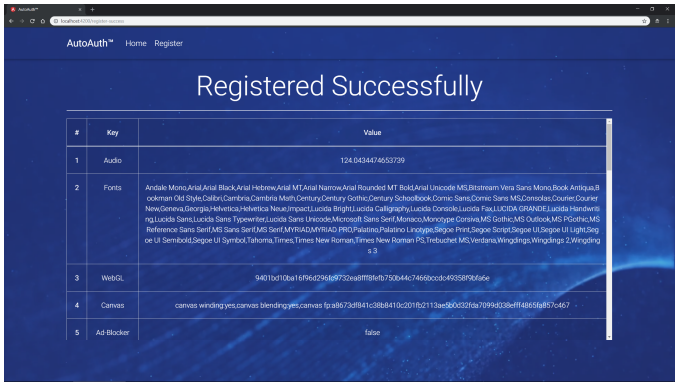


Figure 5: Demo Page to Show User Fingerprint Value

4. EXTENSIONS

4.1 Improvements to be made

In our current implementation, we allow for variations in the components of a fingerprint by assigning a weight to each component that represents the unlikelihood of the value changing. These weights are used during the authentication stage to calculate how different the current fingerprint is compared to the original fingerprint stored in the database. However, in order to support such variations, there are few other issues that we have to consider.

Accuracy or Convenience. The tradeoff between accuracy and convenience is one such issue that we did not cover in detail. Even though our implementation includes various security levels, we did not investigate the optimal threshold level that should be set for the different levels of security or the optimal algorithm to calculate the amount of variation for each component.

Increasing the threshold for the amount of variation will increase the accuracy of authentication while at the same time provide inconvenience to the users since they are unable to make any changes to their device configuration. On the other hand, if the threshold is lowered, though the authenticity of the users might be affected, users are instead able to log in successfully even though they have installed a new font or changed their screen resolution after they registered their fingerprints. To aid with the optimal threshold level, custom algorithms for check each component can also be implemented. For instance, a change in the language component from en-US (English - United States) to en-SG (English - Singapore) can have a lower amount of variation compared to a change from en-US to ru-RU (Russian - Russia). Further studies will have to be conducted to find the optimal threshold level, the weights for each component as well as the algorithm to check each component.

Another concern regarding accuracy and convenience is that there is no one security level that is able to fit different use cases. For a certain use case, using a medium security level may be sufficient while for another specific use case, we may have to use a low security level. One method to go about solving this issue is to allow for more detailed customization by the users themselves. For example, if a certain user travels frequently due to work, the user will have the value of the timezone component change regularly. In this case, it is better to reduce the effect the timezone component has on the amount of variations. Therefore, the timezone component can adjusted a lower weight just for this specific user.

Change of Hardware Configurations. Though most devices connected to the internet are mobile devices such as smartphones, tablets and laptops, where the hardware are unlikely to change, there are still a significant number of desktop computers being used. The hardware of desktop computers can change over time since users will likely upgrade their computer's components as they get outdated. This can reduce the success rate of fingerprint authentication and will require the users to re-register their fingerprint when the new device fingerprint differs too much from the original. Instead of updating the fingerprint every time a variation is detected, a proposed solution is to normalize the device fingerprint over time. This allows for the device to go through different configurations over a period of time while still maintaining both the accuracy and convenience. It is also able to handle the use case mentioned previously where a the timezone component can also change over time when a user is travelling over multiple countries.

Multi-device Support. Adding multi-device support to our current implementation is as easy as registering multiple device fingerprints for each user account. However, though this is possible, supporting multiple devices for a single user comes with its own limitations such as scalabilty. Having multiple fingerprints for one user means that when logging into the account, the server would need to check the current device fingerprint with the list of fingerprints registered under that specific user. As the size of the fingerprint is larger than regular passwords and each components have to be checked with its own algorithm, a user with a significant

number of devices will take a much longer time to authenticate. Further more, because our implementation allows for variations in the device fingerprints, we will be unable to tell which fingerprints stored on the server is the current fingerprint's variation. This prevents us from using the method where we normalize a fingerprint over time.

4.2 Real World Implementation

Before we start using fingerprint authentication in a real world application, it is very important to ensure that a 2FA (Two Factor Authentication) is in place. This is because in the event where a user's device fingerprint changes significantly more than the allowed threshold, we are still able to use 2FA to authenticate the user, and at the same time reregister the user's device fingerprint into the database.

5. CONCLUSION

In a nutshell, device fingerprinting is an effective and convenient method of device identification and we have examined the possibility of transforming this key idea into web-based user authentication using unique fingerprints.

By combining various components of a device's properties, we could increase the fingerprint entropy for an individual's device, correspondingly unique to an individual.

Although an device's fingerprint could change over time, we have considered that a user can be re-authenticated using a one-time password sent to their registered email.

It is worth noting that fingerprinting alone however, cannot prevent attacks such as session hijacking where a perpetrator can simply collect additional information about the target device once the hijacking is successful.

All in all, we think that fingerprinting is useful and provides a viable authentication method allowing users to be authenticated easily and such technology can be explored further in future security implementations.

6. REFERENCES

[1] T. 41st Parameter. The 41st parameter announces new real-time product as world's first standard for online pc identification. http://www.the41.com/buzz/announcements/41st-parameter-announces-new-real-time-product-worlds-first-standard-online-pc-0, 2018.

[2] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gurses, F. Piessens, and B. Preneel. Fpdetective: Dusting the web for fingerprinters. In *the 2013 ACM SIGSAC conference on Computer communications security*, pages 1129–1140. ACM, 2013.

[3] F. Alaca and P. van Oorschot. Device fingerprinting for augmenting web authentication: Classification and analysis of methods. In *ACSAC '16*. ACM, 2016.

[4] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz. On the reliability of wireless fingerprinting using clock skews. In *ACM WiSec*, 2010.

[5] R. Beverly, R. Koga, and K. Claffy. Initial longitudinal analysis of ip source spoofing capability on the internet. *ISOC whitepaper*, 2013.

[6] K. Boda, A. M. Foeldes, G. G. Gulyas, and S. Imre. User tracking on the web via cross-browser fingerprinting. *Information Security Technology for Applications*, 7161:31–46, 2012.

[7] P. Laperdrix, W. Rudametkin, and B. Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Proceedings - 2016 IEEE Symposium on Security and Privacy*, pages 878–894, September 2016.

[8] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. Fingerprinting information in javascript implementations. *Web 2.0 Security, Privacy*, 2(1):11, 2011.

[9] K. Mowery and H. Shacham. Pixel perfect : Fingerprinting canvas in html5. *Web 2.0 Security, Privacy 20 (W2SP)*, pages 1–12, 2012.

[10] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium*, page 541. IEEE, 2016.

[11] NoScript. Noscript. http://noscript.net/, 2018.

[12] Panopticlick. Panopticlick. https://panopticlick.eff.org, 2018.

[13] D. Preuveneers and W. Joosen. Smartauth: Dynamic context fingerprinting for continuous user authentication. In *ACM SAC*, page 218502191, 2015.

[14] M. D. Site. Device tracking add-on for minfraud and proxy detection services. http://dev.maxmind.com/minfraud/device/, 2018.

[15] T. Unger, M. Mulazzani, D. Fruhwirt, M. Huber, S. Schrittwieser, and E. Weippl. Shpf: Enhancing http(s) session security with browser fingerprinting. In *8th Int. Conf. Availability, Reliability and Security*, pages 255–261, September 2013.