# Demonstrating a Simple Device Fingerprinting System

Michael Rausch, Andrew Bakke, Suzanne Patt, Beth Wegner and David Scott
Mathematics and Computer Science Department
Ripon College
Ripon, WI 54971
Corresponding author: rauschm@ripon.edu

## Abstract

In response to threats to cookies, the dominant technology for tracking on the internet, some companies have developed an alternative form of tracking known as device fingerprinting that does not rely on cookies to identify a visitor. Instead, a profile of the user may be created by querying the browser for identifying information regarding the system characteristics such as browser version, screen size, fonts installed and more.

We created a device fingerprinting system so we could learn how device fingerprinting is accomplished. This system was implemented on our website, which attempted to fingerprint consenting visitors. Our site then attempted to identify returning visitors through the fingerprinting algorithm. We also demonstrated that previously deleted cookies could be regenerated using device fingerprinting. Tests have confirmed that this system can identify returning visitors who are not using cloned machines.

# 1. INTRODUCTION

## 1.1 Overview and Background

By default, the internet is an anonymous place. Generally a visitor to a website does not need to provide identifying information to access a site. There may be portions of the site that are protected by a username and password, but even in these cases visitors can choose whether or not they will surrender their anonymity by logging into the site.

This level of anonymity can be frustrating for those who maintain websites. The managers of websites occasionally wish to uniquely identify users. They have many varied reasons for doing so. Some wish to identify users for analytics purposes, to learn how many different people have seen or read a particular part of a website. Others, such as banks, wish to prevent fraudulent access to accounts by identifying visitors who have login credentials but are unlikely to be the real owners of those credentials. Targeted advertising is another prominent reason for attempting to uniquely identify users. Advertisements tailored to a particular person may be shown if a website is able to uniquely identify a visitor. Regardless of the reasons, some users do not wish to be tracked or have varied privacy preferences.

There are several different techniques that websites can use to uniquely identify users. Cookies are one of the most widely used and oldest technologies for this purpose. Their original purpose was to save state as a visitor moved to different parts of a website, because http is a stateless technology. Since cookies saved state it was possible to create websites with services such as shopping carts, where a visitor could visit multiple pages and the website would remember whether the visitor had selected an item to add to the cart on a previous page. Since cookies could save state and could be tied to a particular visitor it also became possible for websites to uniquely identify visitors using this technology. Cookies are merely a text file that is sent from a website to the visitor's browser which can be fetched later. Cookies can contain text, including a unique identification number.

Cookies are extremely prevalent on the internet. Ayenson found that all of the 100 most popular websites on the internet employ cookies [3]. Privacy advocates and others, concerned about the widespread use of cookies, have developed many techniques and tools to limit the effectiveness of cookie-based identification and tracking. Many web browsers have options which allow users to block cookies from being placed on their computer or an option to delete cookies. Safari blocks third-party cookies by default [16]. Several browsers have privacy extensions such as Ghostery, which can help block trackers.

Those who wished to continue attempting to uniquely identify visitors without using traditional cookie techniques had to develop alternative methods. Many creative solutions were developed, including the creation of zombie cookies, which were discovered by Ayenson [3]. Websites using the zombie cookie technique put copies of a particular cookie in several different storage locations, including the normal place in the browser for holding cookies and unusual places such as in http ETags, HTML5 storage, and Flash cookies (Local Shared Objects). If a user deletes cookies from their normal location but neglects to clear all the copies as well, the normal cookie can be regenerated from the copies once the user revisits the original website that the cookies came from.

Device fingerprinting was another method that could be used to uniquely identify visitors without relying on traditional cookies. Device fingerprinting is based on the hope that enough of a visitor's system attributes can be collected to identify the visitor upon a return visit. These system attributes are often collected by using JavaScript and Flash. JavaScript and Flash give programmers access to many of the system attributes of visiting devices to make websites more reliable and effective. For example, programmers can use JavaScript to learn the type and version of the visitor's browser and operating system so that content can be tailored to different sorts of devices. Fonts are another example. Flash gives programmers access to the list of fonts installed on a visitor's computer so that a program can check to see if a particular font is installed. If it is not detected, an alternative can be used. JavaScript and Flash may give access to many other system attributes to make programming easier, including information regarding which time zone the machine is in, what plugins are installed on a machine (and in some cases their version numbers), whether Flash is installed on the computer, the approximate values of several mathematical constants such as e or pi which can vary from system to system, and the dimension of a user's screen.

Some believe that device fingerprinting is the future of the internet. The press, including Forbes and the Wall Street Journal, suggest that fingerprinting could be replacing cookies as a tracking technology on the internet [2][13]. Recent studies, however, have found evidence of device fingerprinting on only a small percentage of popular websites [1][10].

Though it is not yet a popular tracking technique on the internet, we still found the idea of device fingerprinting fascinating and wished to develop a simple device fingerprinting system so that we could deeply understand how this technique works. Our objective was to build a website that would identify a returning visitor without relying on a cookie. We also wanted to demonstrate that it was possible to regenerate cookies through device fingerprinting.

## 1.2 Related Work

People have been interested in remotely identifying devices through peculiarities in the system for many years. One prominent early example was the study conducted by Kohno, Broido and Claffy in 2005 [6]. They found that it was possible to uniquely identify a device connected to the internet by its clock skew. They were able to successfully identify a device even if it was moved to different areas and connected to the internet via a different access method.

Thanks to Eckersley and the Electronic Freedom Frontier, the concept of JavaScript and Flash based device fingerprinting was introduced to hundreds of thousands of people in 2010. In that year, Eckersley and the EFF created a project called Panopticlick. Panopticlick was an experiment that attempted to demonstrate the feasibility of device fingerprinting by creating a website that would fingerprint volunteers and attempt to identify them on return visits. Eckersley published a paper based on the results of the experiment [5]. He found that over 94% of browsers that used Flash or Java had a unique fingerprint. He also showed that his system could identify returning visitors even if their system had changed since the last visit, with a 99.1% success rate.

Our project owes a great deal to the Panopticlick project. Eckersley gave some general information on how the fingerprinting system of Panopticlick worked in his paper but gave few specifics. The paper focused more on proving that enough information could be gathered to successfully identify returning visitors. We wanted a deeper and more detailed understanding of how device fingerprinting is actually accomplished, and we felt that the best way to learn this was to create such a system for ourselves, using Panopticlick as a model.

Since 2010 a number of studies have attempted to discover how fingerprinting is implemented commercially and its impact on the internet. In 2013 Nikiforakis et al. conducted a research project where they found commercial fingerprinting scripts and analyzed them to determine how they worked and how the techniques in the scripts differed from the techniques used in the Panopticlick project [8]. Later in that year separate projects independently concluded (after crawling the internet looking for evidence of device fingerprinting) that fewer than 6% of popular websites implement JavaScript and Flash based fingerprinting [1][10].

## 2. Implementing the System

The goal of the project was to create a device fingerprinting system connected to a webpage so that returning visitors could be identified without relying on a cookie. We created two versions of such a system: first one that used only JavaScript to collect system information and then later a more advanced version that had all of the functionality of the basic JavaScript system in addition to Flash to collect a list of the fonts installed on the visitor’s computer. The webpage associated with each version of the system contains a textbox into which visitors can type some identifying phrase such as their name. Next to the textbox there is a button that will fingerprint the visitor once he or she clicks it.

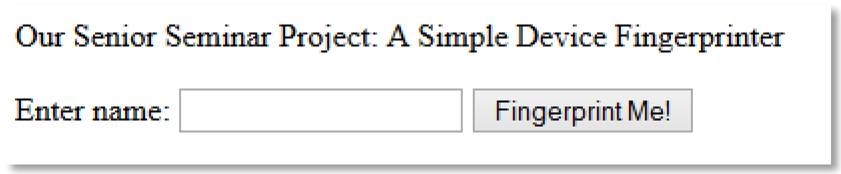

Our Senior Seminar Project: A Simple Device Fingerprinter

Figure 1: Screenshot of webpage on initial visit

If the visitors revisit the homepage in the future, the textbox and button will be replaced with text welcoming them back as returning visitors and displaying the phrases that they typed into the textbox.

Conceptually the systems can be broken into two components. The first, the client side, uses JavaScript (and Flash in the more advanced version) to collect system attributes. The second component, the server side, uses PHP in conjunction with a MySQL database to store and manipulate the data behind scenes, comparing a visitor's fingerprint to fingerprints stored in the database, searching for a match.

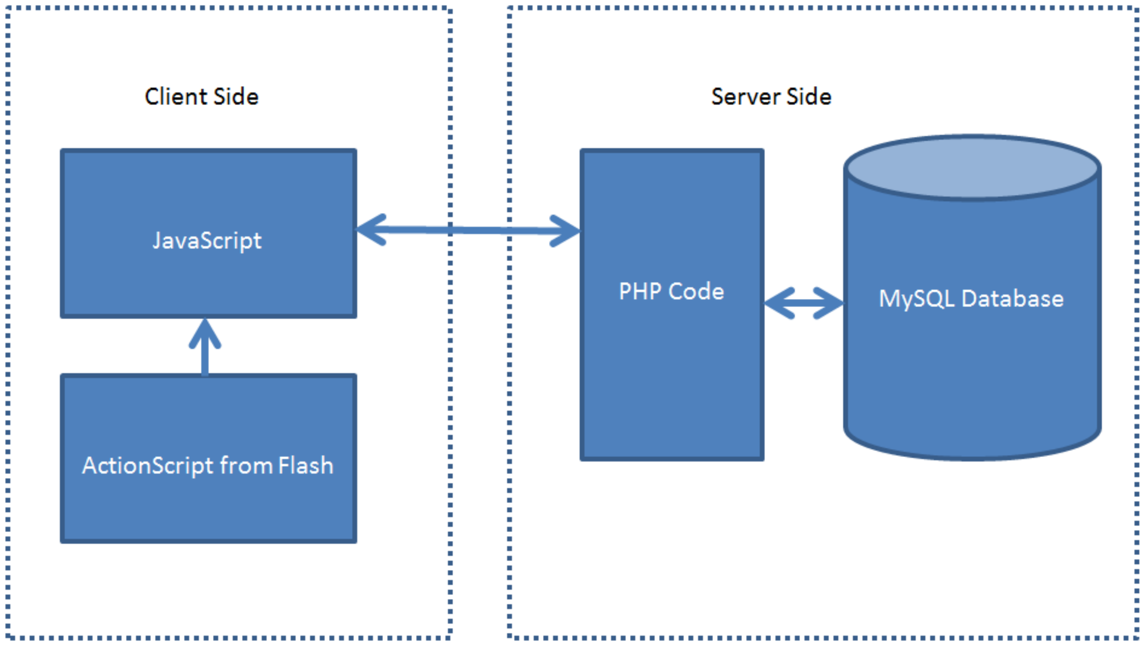
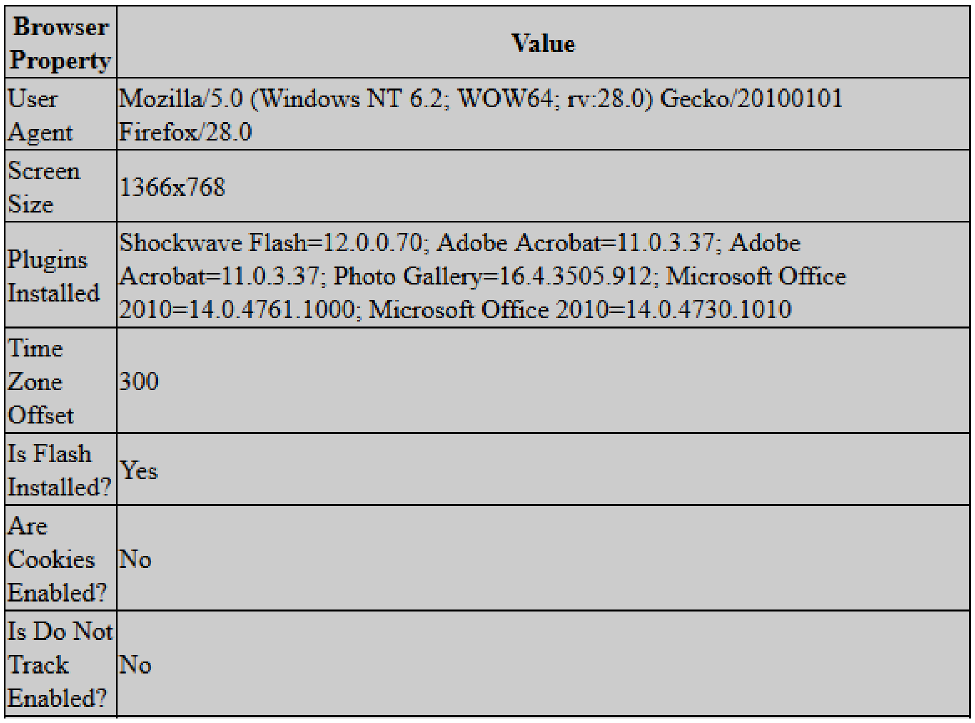


Figure 2: Diagram of device fingerprinting system.

## 2.1 Client Side

We can think of the JavaScript and Flash code as part of the client side as it is executed on the visitor's machine. The JavaScript code collects a number of system characteristics, including the user agent, screen height and width, the time zone offset, whether Flash is installed on the system, whether cookies are enabled, and whether Do Not Track is enabled. This list of characteristics gathered through JavaScript is broadly similar to the information that Panopticlick and commercial device fingerprinters collect with their systems [8]. The webpage associated with our more advanced Flash-enabled version of this system also uses JavaScript to display all the collected information to the user. An example from one visitor is shown in Figure 3.

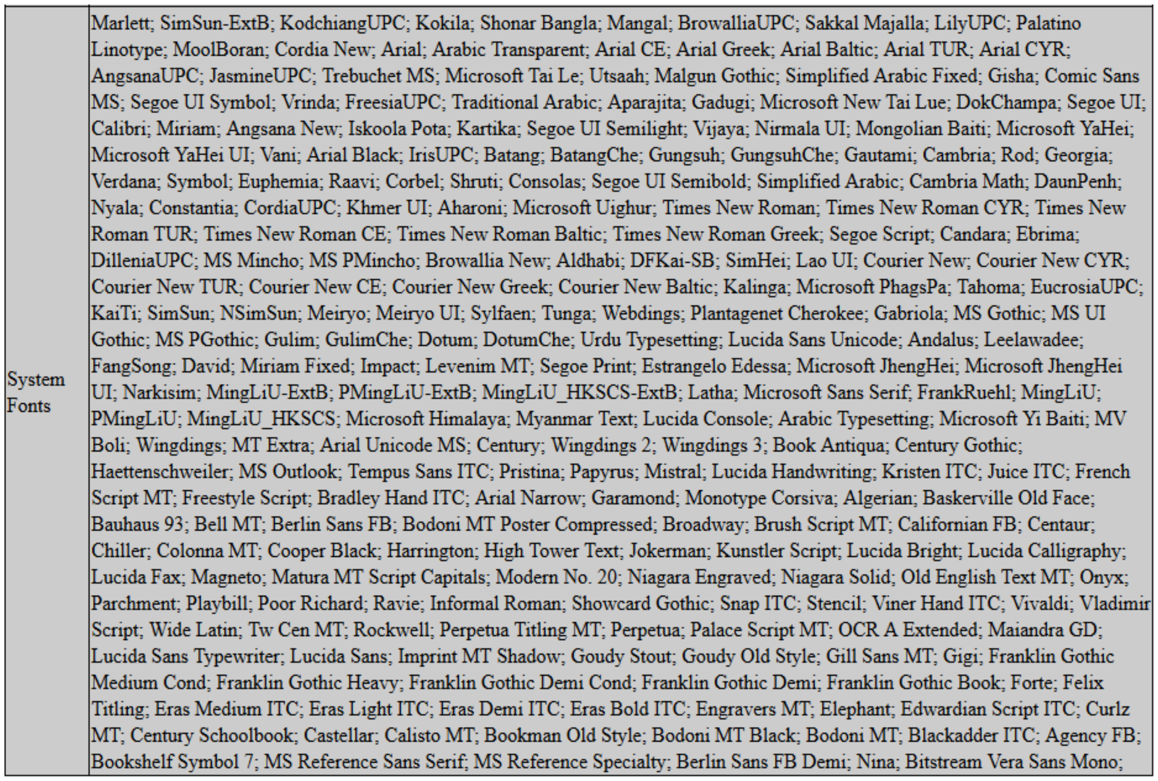


| Browser Property | Value |
|---|---|
| User Agent | Mozilla/5.0 (Windows NT 6.2; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0 |
| Screen Size | 1366x768 |
| Plugins Installed | Shockwave Flash=12.0.0.70; Adobe Acrobat=11.0.3.37; Adobe Acrobat=11.0.3.37; Photo Gallery=16.4.3505.912; Microsoft Office 2010=14.0.4761.1000; Microsoft Office 2010=14.0.4730.1010 |
| Time Zone Offset | 300 |
| Is Flash Installed? | Yes |
| Are Cookies Enabled? | No |
| Is Do Not Track Enabled? | No |

Figure 3: Screenshot of a visitor's system characteristics gathered via JavaScript and displayed on the webpage.

After examining Figure 3 it becomes clear that a large amount of information can be gathered about a visitor's system through JavaScript code. For example, by inspecting the user agent one can learn that the visitor used version 28 of a Mozilla Firefox browser on a Windows NT 64-bit machine to navigate to the website. The visitor's screen size was 1366 by768 pixels. The JavaScript code was able to tell that several plugins were installed on the machine, including Shockwave Flash, Adobe Acrobat, Photo Gallery and Microsoft Office. It was also able to collect detailed version numbers for each plugin. The JavaScript code found that the computer was located in a time zone 300 minutes from Greenwich Mean Time, Flash was installed on the system, cookies were not enabled, and Do Not Track was not enabled. By just using JavaScript a reasonably thorough fingerprint can be developed, because there are not likely to be too many systems using that particular browser on that operating system with those specific plugins and that screen size in that time zone. Our first attempt at creating a fingerprinting system only collected this information and nothing else about a user's device.

To create a stronger fingerprint we decided to also attempt to gather a list of fonts, so we created another device fingerprinting system that implemented a Flash solution to collect fonts and also collect the other information through JavaScript. Eckersley reported that "plugins and fonts are the most identifying metrics" in device fingerprinting [5]. We did not wish to overlook such an important way of establishing a system's fingerprint, though we could not easily collect this information through JavaScript. Instead the algorithm implemented a solution using Adobe Flash's ActionScript, which is often used to implement Flash based features on websites. We use ActionScript to collect fonts on a visitor's device. When the fonts were collected they were often returned in nonalphabetic order. The list was left unsorted as the order in which fonts are returned can be an identifying feature of the device as well [5]. Figure 4 shows a fairly typical example of the number of fonts that Flash can often detect on a user's device.



| | |
|---|---|
| System Fonts | Marlett; SimSun-ExtB; KodchiangUPC; Kokila; Shonar Bangla; Mangal; BrowalliaUPC; Sakkal Majalla; LilyUPC; Palatino Linotype; MoolBoran; Cordia New; Arial; Arabic Transparent; Arial CE; Arial Greek; Arial Baltic; Arial TUR; Arial CYR; AngsanaUPC; JasmineUPC; Trebuchet MS; Microsoft Tai Le; Utsaah; Malgun Gothic; Simplified Arabic Fixed; Gisha; Comic Sans MS; Segoe UI Symbol; Vrinda; FreesiaUPC; Traditional Arabic; Aparajita; Gadugi; Microsoft New Tai Lue; DokChampa; Segoe UI; Calibri; Miriam; Angsana New; Iskoola Pota; Kartika; Segoe UI Semilight; Vijaya; Nirmala UI; Mongolian Baiti; Microsoft YaHei; Microsoft YaHei UI; Vani; Arial Black; IrisUPC; Batang; BatangChe; Gungsuh; GungsuhChe; Gautami; Cambria; Rod; Georgia; Verdana; Symbol; Euphemia; Raavi; Corbel; Shruti; Consolas; Segoe UI Semibold; Simplified Arabic; Cambria Math; DaunPenh; Nyala; Constantia; CordiaUPC; Khmer UI; Aharoni; Microsoft Uighur; Times New Roman; Times New Roman CYR; Times New Roman TUR; Times New Roman CE; Times New Roman Baltic; Times New Roman Greek; Segoe Script; Candara; Ebrima; DilleniaUPC; MS Mincho; MS PMincho; Browallia New; Aldhabi; DFKai-SB; SimHei; Lao UI; Courier New; Courier New CYR; Courier New TUR; Courier New CE; Courier New Greek; Courier New Baltic; Kalinga; Microsoft PhagsPa; Tahoma; EucrosiaUPC; KaiTi; SimSun; NSimSun; Meiryo; Meiryo UI; Sylfaen; Tunga; Webdings; Plantagenet Cherokee; Gabriola; MS Gothic; MS UI Gothic; MS PGothic; Gulim; GulimChe; Dotum; DotumChe; Urdu Typesetting; Lucida Sans Unicode; Andalus; Leelawadee; FangSong; David; Miriam Fixed; Impact; Levenim MT; Segoe Print; Estrangelo Edessa; Microsoft JhengHei; Microsoft JhengHei UI; Narkisim; MingLiU-ExtB; PMingLiU-ExtB; MingLiU_HKSCS-ExtB; Latha; Microsoft Sans Serif; FrankRuehl; MingLiU; PMingLiU; MingLiU_HKSCS; Microsoft Himalaya; Myanmar Text; Lucida Console; Arabic Typesetting; Microsoft Yi Baiti; MV Boli; Wingdings; MT Extra; Arial Unicode MS; Century; Wingdings 2; Wingdings 3; Book Antiqua; Century Gothic; Haettenschweiler; MS Outlook; Tempus Sans ITC; Pristina; Papyrus; Mistral; Lucida Handwriting; Kristen ITC; Juice ITC; French Script MT; Freestyle Script; Bradley Hand ITC; Arial Narrow; Garamond; Monotype Corsiva; Algerian; Baskerville Old Face; Bauhaus 93; Bell MT; Berlin Sans FB; Bodoni MT Poster Compressed; Broadway; Brush Script MT; Californian FB; Centaur; Chiller; Colonna MT; Cooper Black; Harrington; High Tower Text; Jokerman; Kunstler Script; Lucida Bright; Lucida Calligraphy; Lucida Fax; Magneto; Matura MT Script Capitals; Modern No. 20; Niagara Engraved; Niagara Solid; Old English Text MT; Onyx; Parchment; Playbill; Poor Richard; Ravie; Informal Roman; Showcard Gothic; Snap ITC; Stencil; Viner Hand ITC; Vivaldi; Vladimir Script; Wide Latin; Tw Cen MT; Rockwell; Perpetua Titling MT; Perpetua; Palace Script MT; OCR A Extended; Maiandra GD; Lucida Sans Typewriter; Lucida Sans; Imprint MT Shadow; Goudy Stout; Goudy Old Style; Gill Sans MT; Gigi; Franklin Gothic Medium Cond; Franklin Gothic Heavy; Franklin Gothic Demi Cond; Franklin Gothic Demi; Franklin Gothic Book; Forte; Felix Titling; Eras Medium ITC; Eras Light ITC; Eras Demi ITC; Eras Bold ITC; Engravers MT; Elephant; Edwardian Script ITC; Curlz MT; Century Schoolbook; Castellar; Calisto MT; Bookman Old Style; Bodoni MT Black; Bodoni MT; Blackadder ITC; Agency FB; Bookshelf Symbol 7; MS Reference Sans Serif; MS Reference Specialty; Berlin Sans FB Demi; Nina; Bitstream Vera Sans Mono; |

Figure 4: Screenshot of fonts installed on a visitor's device that were collected via Flash's ActionScript and displayed on the webpage.

The JavaScript code generates a random integer to use as a unique identification number for the collected fingerprint. The JavaScript code also creates a cookie containing this unique identification number and places it on the user's computer.

Once all of the information necessary to create the fingerprint is collected by the scripts it is bundled by JavaScript (along with the user's phrase that was entered into the textbox and the identification number) and sent to a PHP script on a server, which processes the data and stores it into a database.

## 2.2 Server Side

Once a visitor comes to the webpage, enters a phrase in the textbox and presses the "Fingerprint Me!" button located to the right of the textbox, the information gathered via JavaScript (and Flash if available) is sent to a PHP script that is executed on the server. The script checks to see if that exact fingerprint was seen in the past by querying a MySQL database. This database holds all previously collected unique fingerprints, the associated phrases that were entered in the textbox, and identification numbers. If an exact match of every device characteristic is found it means that the fingerprints are identical. The fingerprinting system assumes that all devices with identical fingerprints are in fact the same device. If an exact match for a visitor's fingerprint is found in the database the phrase associated with the fingerprint that was entered into the textbox is fetched from the database and dynamically displayed on the visitor's webpage. If the fingerprint is not an exact match it is stored in the database along with the phrase entered into the textbox and the unique identification number. These data may be used later if the visitor returns.

## 2.3 Cookie Regeneration

In his paper on device fingerprinting Eckersley said that it could be possible to regenerate a cookie that a person had previously deleted by using device fingerprinting. As far as we know nobody has claimed to have tried to accomplish this feat. We wanted to show that this could be done. In the advanced fingerprinting system that employs Flash there exists JavaScript code that creates a cookie with a unique identification number. The cookie is stored in a visitor's browser (if they allow cookies to be set). The number used in the cookie is also stored in a database along with the fingerprint. If a visitor returns to the webpage and the fingerprinting system recognizes the returning visitor it will place a cookie on his or her device that holds the same identification number as the previous cookie (if the previously set cookie had been deleted). In this way cookies with values identical to the previous cookies are regenerated if the old cookies were deleted by a visitor between visits to the website.

## 2.4 Testing

We tested our system with a number of virtual machines. Since the fingerprinting process is so sensitive to software updates, the complete control virtualization offers over the machine (along with the large number of machines that could be easily and inexpensively created) made this a very attractive testing option. We tested with Internet Explorer, Firefox, and Chrome in virtual Windows XP, Windows Vista, and Windows 7 machines. We tested both Firefox and Chrome in Ubuntu. We created exact clones of the Windows XP and Windows 7 machines, and three exact clones of the Ubuntu machine. We also created two machines that were clones of the Windows XP machine except for the fact that these machines had Adobe Reader installed, and we likewise created two further virtual machines that were clones of the Windows 7 machines except for the fact that they had Adobe Reader installed. Altogether we had a total of thirteen virtual machines each with between two to three browsers. We had a total of thirty-five different operating system-browser-plugin combinations.

Each individual browser went through the same procedure. The testing process for the JavaScript-only fingerprinting system and the Flash-enabled fingerprinting system was identical. In the testing process each browser navigated to the webpage associated with the particular device fingerprinting system it was testing, and the tester entered a unique phrase related to the operating system and browser into the textbox found on the webpage. The tester then left and immediately revisited the webpage to see if the unique phrase was displayed on the screen. If the phrase that was previously entered was displayed on the screen it was taken as evidence that the fingerprinting system recognized the return visit. If no phrase was displayed or if the wrong phrase was displayed we took that as evidence that the fingerprinting system did not correctly identify a return visitor. If the more advanced system with the cookie regeneration capability was being tested then at this stage the tester would also check to see if the fingerprinting system had left a cookie on the virtual machine. If it had, the identification number contained in the cookie would be noted. Next, the tester left again, deleted all cookies and revisited the site to see if the unique phrase that had been typed previously was displayed. If it was this was taken as proof that the device fingerprinting system managed to identify the user without the aid of the cookie. At this stage the cookies held in the tester's browser would be checked again to see if a cookie with the exact same identification number had been regenerated.

## 3. Results and Discussion

Neither of the two fingerprinting system could tell the difference between exact clones. This result was expected, because the algorithm relies on differences in a device's characteristics to tell them apart. If two devices have characteristics that match exactly (the same browser, plugins, fonts, time zone, etc. as a clone would) the algorithm assumes that they must be the same device. The inability to tell between clones is a weakness of this form of device identification. Cookies, in contrast, could be used to tell the difference between clones because one clone could be given a cookie with one unique identification number and the other clone could be given a cookie containing a different unique identification number. The two clones could then be distinguished by their differing identification numbers. While the failure of the fingerprinting systems to distinguish between clones is disappointing it was not unexpected.

Aside from the aforementioned problem with exact clones, the simple JavaScript-only device fingerprinting system was successful in all of its tests. Altogether it was able to distinguish 14 different operating system-browser-plugin combinations from each other. It is encouraging that this simple method was so successful but it may not gather enough information to effectively distinguish among a larger number of different devices.

The Flash-enabled fingerprinting system also successfully passed all of its tests, aside from the problem of the exact clones. It too was able to identify fourteen unique operating system-browser-plugin combinations from one another. In addition it was always able to regenerate cookies that had been deleted between visits to the webpage with the same unique identification number that they had previously held. This Flash-enabled system was much more difficult to debug than the previous JavaScript-only system. We had to ensure that this system would not crash if a user did not have Flash enabled. We eventually successfully fixed the system so that if a user did not have Flash enabled it would simply gather all of the information it could through JavaScript and attempt to fingerprint the device using only that information. It was also difficult to ensure that the fingerprinting algorithm worked effectively in all the different browsers we tested. It was especially challenging to make the webpage display correctly in Internet Explorer. In the end the issues were resolved, and the resulting Flash-based system can generate a much stronger fingerprint than the JavaScript-only system because it has the ability to detect fonts. The stronger fingerprint generated by the Flash-enabled system should allow the system to identify different devices in a larger population more effectively than the JavaScript-only system.

## 4. Conclusion

The aim of the project was to create a system that could identify a returning visitor to a webpage without relying on the use of cookies. To achieve this goal we created two different systems. One system used JavaScript to query the visitor’s browser for system characteristics, while the other system used both JavaScript and ActionScript to collect system characteristics. In both cases these characteristics were sent to a PHP script, which turned the characteristics into a fingerprint that was stored in a MySQL database. This database of fingerprints was used to recognize returning visitors. A secondary goal of the project was to demonstrate that deleted cookies could be regenerated by a device fingerprinting system upon a visitor’s return to the website. Our testing found that the systems could not tell the difference between devices that were clones. In all of our other test cases the project was successful: the device fingerprinting system correctly identified returning visitors and regenerated cookies. The system works on a wide variety of browsers and devices.

*Future Work*

Our testing can be expanded and made more rigorous in several different ways. One possible tactic would be to simply use many more virtual machines in the testing process. We could emulate many more kinds of operating systems, including Macintosh and mobile operating systems. We could also try to have a wider variety of browser versions and plugin versions on each of the virtual machines. Using these methods we would be more likely to uncover possible bugs in the system, while continuing to demonstrate the feasibility of device fingerprinting. Of course, the drawback to this method of testing is that it does not provide substantial and convincing evidence that device fingerprinting works in the real world.

The best way to test this system is to have many people visit the website with their own devices and browsers. With this method, after agreeing to the process, a visitor would have his or her device fingerprinted by the system. On a return visit, our fingerprinting algorithm would attempt to check the fingerprint and guess the visitor’s identity. The algorithm’s prediction would be verified with the cookie. With this process it would be possible to calculate useful information like the percentage of correct guesses, the percentage of false-positives, etc. that our algorithm makes in the real world. It would be very beneficial to go through the verification process of having many visitors (preferably many thousands) visit the website for testing purposes, but there are difficulties with this manner of testing. It is challenging to drive that much traffic to a webpage. Visitors must not only visit the page, they must also return, because the purpose of the system is to recognize returning visitors. In addition, a visitor must not delete their cookie in between visits, otherwise there would be no way of verifying the algorithm's guess. It would also be difficult to get a good random or typical distribution of website visitors to the site. It is possible that those who visited our website would be particularly privacy or security conscious and have devices that are configured differently from devices used by typical people browsing the web. Despite these challenges, this testing method of having thousands of real visitors come to the fingerprinting website is the best and most convincing way of demonstrating the success of the system.

Testing in this manner could give the necessary data to implement a solution that would make the fingerprinting system much more robust to changes that a user's system may undergo between visits. As mentioned previously, after the algorithm collects all of the information it can on the visiting system's characteristics, it searches the database for an exact match of that information. If it does not find an exact match it will not recognize returning visitors, even if the information collected from the first visit is nearly identical to the information collected on the return visit. A small, simple software update on a visitor's computer could ruin the fingerprint. If the system were tested by having a large number of people visit the site, leave, and return some time later it would be possible to determine how quickly a user's system changes over time and what sorts of changes are likely. Using this information it would be possible to modify the device fingerprinting algorithm so that it could make reasonable guesses as to whether a particular visitor to the website is a new or returning visitor in the absence of an exact match in the database of previously collected fingerprints. For example, it might be possible to compare two fingerprints (one from a current visitor and one that was created in the past) and find that they differ only by the version number of the browser, and the current visitor's browser version number is higher. Using the data from our proposed test our updated fingerprinting algorithm might decide to guess that the current visitor is a returning visitor and that the visitor's browser was updated since the last visit. The effectiveness of this updated version of the fingerprinting algorithm could in turn be verified by having more visitors come to the site and return later, again using the cookie as a verification method.

By following these testing schemes and implementing these changes the device fingerprinting system could become an effective way to identify returning visitors to a website with a high degree of certainty and without relying on the use of cookies.

## 5. References

[1] ACAR, G. JUAREZ, M. NIKIFORAKIS, N. DIAZ, C. GURSES, S. PIESSENS, F. PRENEEL, B. 2013. FPDetective: Dusting the Web for Fingerprinters. *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security* (CCS '13). ACM, New York, NY, USA, 1129-1140. DOI=10.1145/2508859.2516674

http://doi.acm.org/10.1145/2508859.2516674

[2] ANGWIN, J. AND VALENTINO-DEVRIES, J. 2010. Race Is On To ‘Fingerprint’ Phones, PCs. The Wall Street Journal. Published Dec 1 2010.

[3] AYENSON, M. WAMBACH, D. SOLTANI, A. GOOD, N. HOOFNAGLE, C. 2011. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. http://ssrn.com/abstract=1898390

[4] Device Fingerprinting. Oracle. 2013. http://docs.oracle.com/cd/E27559_01/admin.1112/e27207/finger.htm

[5] ECKERSLEY, P. 2010. How Unique Is Your Web Browser? https://panopticlick.eff.org/browser-uniqueness.pdf

[6] KOHNO, T., BROIDO, A. AND CLAFFY, K. 2005 Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93--108, May 2005. http://www.caida.org/publications/papers/2005/fingerprinting/

[7] Multi-Layer Device Fingerprinting. Kount. 2013. http://www.kount.com/products/complete/multi-layer-device-fingerprinting

[8] NIKIFORAKIS, N. KAPRAVELOS, A. JOOSEN, W. KRUEGAL, C. PIESSENS, F. VIGNA, G. 2013. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. *IEEE Symposium on Security and Privacy*, 2013, pp. 541-555. https://lirias.kuleuven.be/bitstream/123456789/393661/1/cookieless_sp2013.pdf

[9] OLSON, S. 2009 Device Fingerprinting Techniques – Several Choices. June 4, 2009. https://www.iovation.com/blog/device-fingerprinting-techniques-several-choices

[10] RAUSCH, M.  GOOD, N. AND HOOFNAGLE, C. 2013. Searching for Indicators of Device Fingerprinting in the JavaScript Code of Popular Websites. Unpublished Manuscript. http://www.truststc.org/education/reu/13/Papers/RauschM_Paper.pdf

[11] SOLTANI, A. CANTY, S. MAYO, Q. THOMAS, L. HOOFNAGLE, C. 2009. Flash Cookies and Privacy.  http://ssrn.com/abstract=1446862

[12] How Device Fingerprinting Works. *ThreatMetrix*. 2012. http://www.threatmetrix.com/how-device-fingerprinting-works/

[13] TANNER, A. 2013. The Web Cookie Is Dying. Here’s The Creepier Technology That Comes Next. *Forbes.* http://www.forbes.com/sites/adamtanner/2013/06/17/the-web-cookie-is-dying-heres-the-creepier-technology-that-comes-next/

[14] VIRTUAL REALITIES. "United Virtualities Develops ID Backup to Cookies." United Virtualities.31 Mar. 2005. 07 July 2009. http://www.unitedvirtualities.com/UV-Pressrelease03-31-05.htm

[15] WALL STREET JOURNAL. What They Know: The Business of Tracking You on the Internet.

[16] What is Safari. Apple. 2013. http://www.apple.com/safari/what-is.html