Non-consent Data Retrieval While using Web or Email Services

Jose Martinez Rivera, Luis Medina Ramos and Amir H. Chinaei
Department of Electrical and Computer Engineering, Univeristy of Puerto Rico, Call Box 9000, Mayagüez, Puerto Rico, 00681, U.S.A.

Keywords: Security, Privacy, Web Server, Email, HTML, JavaScript.

Abstract: User's private data might be secretly retrieved for or against them every time they browse the web or use email services. This study addresses how these services could secretly retrieve such data. While users may appreciate these techniques as a means of protecting them from hacking, fraud, etc., they may have some privacy concerns against them. We have implemented two approaches to demonstrate how such data retrievals help web and email services to properly identify a user. We have also conducted extensive experiments to measure the success rates of our approaches. Results show 86.217% successful identification for web services and 81.1% successful identification of emails that were attempted to be sent anonymously.

1 INTRODUCTION

With the recent increase of social mediums across the internet, maintaining one's personal information private has been getting more difficult. Just as criminals can steal your identity with some information as simple as your street address or through simple phishing websites, they can use information posted by users on the internet. With the advent of the internet and social networks, people have been more alert of the material they upload to the internet, in hopes of staying safe from such attackers. As end-users upgrade their defenses, hackers upgrade their methods, using social engineering, email spoofing, phishing, etc.

Even with the popular countermeasures that email services provide, such as removing scripting from HTML code and disallowing the embedding of interactive media, adversaries can get someone's information just by having users visit a website or through other legitimate services such as email tracking.

In this research, we focus on studying and developing methods of identifying users through both email and web browsing services, trying to secretly retrieve information about the user directly from reading an email or visiting a webpage. User identification can be a useful tool, especially when settling disputes or trying to identify a user wrongfully accessing a network, for example. The knowledge that comes from knowing if an email was opened—and if so where, when and how—can aid greatly in an investigation. In this research, we propose a novel approach for this purpose. Deterring possible scams can also be carried out within our second contribution in this research. Discussions on the limitations of each approach can definitely inspire end-users by providing guidelines to protect their privacy.

2 RELATED WORKS

There are many research projects in the areas of handling spam and phishing emails, along with protocol proposals and proofs (Abadi and Blanchet, 2005). Among many approaches, HTML and `JavaScript` are known as good means to extract information from computer devices. Sanchez et al. defined a mechanism to battle against spam emails (Sanchez et al., 2011). They have proposed a "support vector machine" that separates end-user devices from legitimate mail servers, using a set of machine features that cannot be easily manipulated by spammers. End-user devices often send particular information within their messages. In particular, viewing protocols and certain keywords in their address help identify end-user devices from mail servers. With this technique, one can establish which emails would likely be part of a spam botnet be no matter if the phishing email is submitted by a hacker mail server or by a contaminated end-user device.

DOI: 10.5220/0005549703160325
In *Proceedings of 4th International Conference on Data Management Technologies and Applications* (DATA-2015), pages 316-325
ISBN: 978-989-758-103-8

Furthermore, in a follow up work, Sanchez and Duan have introduced a sender-centric approach to identify phishing emails by using headers (Sanchez and Duan, 2012). The email headers use properties, such as the "From", "Reply-To" and "Return Path" fields, by which one can properly filter phishing emails with a success rate of 98.7%.

Abadi and Blanchet have designed a protocol to help secure email sending by adding a form of validation to each message sent (Abadi and Blanchet, 2005). Their approach is as follows: an email message will not be accessible to the receiver unless a receipt automatically sent back from the receiver to the sender is read by the sender. This provides identity verification because in the case that an email was spoofed, the one who receives the receipt is the alleged sender and not the spoofer. Hence, the legitimate senders can protect receivers from malicious emails. It is worth noting that the legitimate sender needs some form of protection, since the spoofer may mimic how the message is sent in its entirety. A parity check may prove useful.

Other similar approaches are presented in (Ateniese and Nita-Rotaru, 2002) and (Abadi et al., 2002), where the receiver sends a receipt, but it does not need to be read by the sender because of a trusted entity acting as a middleman in the case of a dispute. The trusted third party (TTP) verifies that both sender and receiver act accordingly, with the receipt only being read in case of a dispute between the two. The latter work uses a TTP that strictly verifies if the receipt is sent and read, and only if both are processed, then it is authenticated. Both methods require additional software to be installed in the end-user's device. A proxy is the most likely choice, instead of, e.g., a plug-in. Such proxies offer additional properties, such as increased privacy and spam filtering.

In (Eckersley, 2010), Eckersley proposes and implements a service that can identify the uniqueness of a certain computer. This work details how browser fingerprinting works and how the service can track subsequent visits, up to a 99.1% chance of identifying that unique browser. Eckersley's work is more concerned with measuring uniqueness, while our work emphasizes on how we can use a computer's uniqueness to identify the user. Also, Eckersley's method of obtaining data to compute uniqueness is majorly based off the information in the browser's cookies, while our study tries to identify variables outside of the browser environment.

The authors in (Louw et al., 2007) show some malware installed as browser plugins (or browser extensions). They assumed positions of malware writers and managed to write an extension called "Browser Spy" for Firefox. This browser extension takes complete control of a user's browser space and can observe all the activity performed through the browser while being undetectable. They have developed two mechanisms: one validates the installation integrity of extensions at load-time, the other is an infrastructure for runtime monitoring and policy enforcement to prevent attacks on a browser's core integrity and protect data confidentiality.

3 TERMINOLOGY

This section reviews key terms used in this paper that although are common in computer science and engineering, some readers may not be familiar with. The internet protocol (IP) address and media access control (MAC) address can aid in the identification of a user. The IP address can refer to either a local IP address or to an internet service provider (ISP) assigned address. The local IP address can reveal where in a local area network the user is, while the public IP address is unique to the whole internet. On common households, the local IP address is not useful or important for identifying users since no computer is geographically too far from the router. In a professional environment, such as a university or workplace, the local IP address has been proved to be useful, since we can look up a computer on a dynamic host configuration protocol (DHCP) table and uniquely identify a specific user. Since the IP address is the same for the whole network, we can use the port number to identify the computer, assuming we have access to the router and some type of connection logs.

The local IP address is commonly referred to as the private IP, while the ISP assigned address is referred to as the public IP. There are currently two versions of IP's, version four and version six. Version four was the first version of the internet protocol deployed; it was followed by version six due to IPv4 address exhaustion. Computers and routers use network address translation (NAT) to produce the address most people recognize from its octet form. This process alters the IP header and appends the transformed address, while most often, leaving the rest of the packet untouched.

The MAC address is also used in networking and is recognized as a great identifier. This address is usually more difficult to acquire, since it is tied to the hardware of the computer, more specifically on a network interface card or read-only memory.

However, one still can spoof the MAC address by tricking the operating system of the computer to believe that the network interface card has another address. This method of spoofing is not foolproof, since the scope of the spoofing is limited to a local network and its extensions. Ideally, the MAC address is what is needed to be 100% of the uniqueness of a computer.

4 METHODOLOGY

In this section, we describe two approaches by which one can uniquely identify a user by secretly collecting some data from their computer and compare it against a database. One is an email-based approach and the other is web-based. In the former, we study the structure of an email and protocols commonly associated with them to uniquely identify the sender of an anonymously sent email. In the latter, we study what (private) data a webpage can extract from a visiting user's computer to uniquely identify who is the user or to verify if a user is who they claim they are. In both approaches, we address the limitations too.

Note that in the case that we do not have the user's email activity readily available, we can inject the web based approach in order to compare users, as described in the following sections.

4.1 Data Retrieval through Emails

Our first part of the study revolved around identifying the information that could be obtained through an email. The ultimate goal was to study the possibilities of retrieving the receiver's information, without consent. To identify a single individual, the information we looked to obtain included the media access control (MAC) address, its internet protocol (IP) addresses and other hardware specific information, such as the amount of RAM memory the computer has available, etc.

This is challenging because of the limitations imposed by email structure. One is able to embed HTML code inside an email, but most popular email services, if not all, filter any HTML email that could contain malicious code. Specifically and most importantly, the script tag is disabled. This filter does not permit any user to execute code on another computer through email, while other formatting tools available in HTML are left untouched. `JavaScript` is our primary way of obtaining information about the receiver, or at least displaying it, and is not available in emails.

One particular approach that seemed to bypass the removal of script tags was the use of Email Tracking Services. The user uses these services to send an email to a specific person (receiver) and track whether the email has been opened or not. Other than tracking the email, the sender is able to view the geographical position of the receiver, its IP address, its operating system, its web browser, the exact time the email opened and how long the receiver had it open. Some of these services work by executing a script when an image (`jpg`, `gif` and `png` formats)—that is hosted by the service provider—is loaded on the webpage or email content. It is a clever approach to remove the script tags, which many email services do provide, but since it is an image, most email servers give the user (receiver) the option to display the image or not. If it is displayed, the script starts running; otherwise, it simply does not run any code.

Another similar technique is to send an invisible image along with the email message to a user of choice. The image is hosted on our own server, and when it is accessed, our server collects the information about the user. Other information, such as the date and time that the receiver has received the email and whether or not the user actually read the email, can be collected too. Furthermore, emails that have not been read can be deleted after a specified time. Notice that not all email services allow the embedding of images inside an email. Some warn the receiver that the email contents require being loaded, as safety precautions. This defeats the purpose of the above-mentioned service, where we do not necessarily want the receiver to know about the security exploit.

Taking advantage of email servers by finding security exploits could provide the solution, but this requires careful study of the server's architecture and may bring up legal ramifications. From this study though, we can probably infer ways to countering attacks of this manner by establishing different security protocols.

We have also embedded Java applets in the HTML code of webpages as well as embedding it in emails, with similar results. Although `JavaScript` does not offer many tools to aid in information retrieval, certainly general-purpose languages (e.g. Java) provide more freedom as to what information one could obtain from the user. This is still a challenging task as Java, similar to `JavaScript`, fails in providing the IP address assigned by an internet service provider (ISP) and only provides the local IP address. The local IP address is not very helpful in identifying a user, because the port number can vary within the same computer, depending on the computer's network adapter and the connected router's configurations. To obtain the public IP address, one would need to use a more online-centered language, such as PHP.

In this research, we also found an issue on sandboxing as a security measure in modern web browsers and email services. The sandbox provides an isolated space for the execution of unknown code, so that it may not tamper with what the system deems inappropriate. This means that our approach would not yield results if our code was entirely sandboxed, meaning no information can be obtained from the user's computer. However, **the fact that the user has gone to great lengths to protect their identity may, in fact, identify them if we use a process of elimination,** taking into account the security measures other users have taken in comparison to that of the victim. In Sections 5 and 6, we detail how we use this approach to identify a sole individual from a group of users.

4.2 Data Retrieval through Web Browsing

The second approach is to exploit web browsing towards user's data retrieval. With `JavaScript` on a web browser, information such as the brand of the web browser, the operating system, can be obtained. The user's public IP address can be obtained from the server connection when the user accesses the webpage. The combination of this information helps to identify the user. When the user is part of a local area network (LAN), multiple users may have the same IP address with different port numbers. Singling out a user from a pool of dozens or hundreds of other users, with common qualities, is challenging yet possible.

The above problem can be solved if the target is not sharing the computer, and only sharing the LAN. The idea is to isolate its machine by verifying other information such as browser information, installed plug-ins, etc. Our method also utilizes the geo-location function present in HTML5, used to successfully locate the user, as long a proxy is not being used and consent is given. Currently, most popular web browsers enable the geo-location function, as long as it is with the user's consent. Internet Explorer, Firefox, Chrome, Safari and Opera all enable the function. This is not restricted to desktop computers. On mobile devices, such as tablets and smartphones, the geo-location function uses the GPS coordinates to pinpoint the exact user's location, down to latitude and longitude. Furthermore, the geo-location function uses online maps, such as Google Maps, to display a map of the surrounding area, along with a marker in the center indicating the user's location.

By using the webpage approach, we can comfortably record webpage visits in a database and filter them as such. In Sections 5 and 6, we detail how we use this approach to identify a sole individual from a group of users.

5 IMPLEMENTATION

Requiring a web server to run tests, we implemented a `Node.js` server where we could host the necessary HTML and `JavaScript` code to properly obtain user information. The following two sections describe these approaches.

5.1 Email-based Approach

Using the email's capability to have an embedded HTML code, we constructed a basic message, with the intent of observing its limitations. As expected and described in Section 4.1, images and hyperlinks respond without any issue. We also embedded a `JavaScript` code to verify if any basic functions can be called. As scripts are blocked, no command can run from inside the `JavaScript` code. Furthermore, we embedded Java codes inside the HTML code. In particular, we have developed an applet by which we extract the user's IP address as well as the MAC address. Because the MAC address is unique, it is significantly important towards identifying a specific device. The applet uses the `InetAddress` and `NetworkInterfaces` packages. In particular, we invoke the method `getHardwareAddress()` on an object `NetworkInterface` to get the MAC address. This method returns a byte array and requires some formatting to properly read it. We also obtain the IP address by using the `getHostAddress()` method. This method returns an IP address assigned to a network interface, but only one if there are many. Notice that this method does not always prove to be useful since it only displays the local IP address of the network interface. Yet, if it were to host the applet on the internet, the private IP address would be of more interest.

When the applet runs, it returns the local IP address and the MAC address of the device as long as Java is installed on the computer. The problem is that one cannot run Java applets from emails, just similar to JavaScript codes. Furthermore, we tested opening the Java applet in an HTML file and achieved the same result, because of the aforementioned sandboxing limitations (Section 4.2). Yet, server-based web services where the message contains only HTML code can still be exploited towards this goal. In particular, once the connection is opened, for instance along with the image, the server obtains information—such as the IP address—and uses the time the connection remains open as an indicator of whether or not the message is read. Obviously, the latter cannot evidently prove whether the user has read the message too or has just opened it without reading it.

5.2 Web-based Approach

In contrast to the email-based approach, the web-based approach is more promising towards information retrieval since we are not restricted to limited services. One can freely host any code on a server, with the only restrictions being the ones imposed by web browsers, especially sandbox-based browsers. Since JavaScript has proved useful in a web environment, we have implemented a Node.js server. The server is entirely written in JavaScript, as can host pages be. Once the user opens the webpage, the script embedded in the HTML code of the page extracts information about the user and the device and sends it to the server where it is saved for further use.

This approach extracts information such as type of the operating system, IP address, web browser and its version, plugins installed and a geological map should the user accept for their location to be sent. The MAC address can be obtained too, but it would require prompts for the user to accept and may reveal the server's intentions.

5.3 The Web Server

Following the web-based approach, we setup a Node.js server with MongoDB as our database. Since we receive unstructured data since browser components vary between browsers, storing the data in a MongoDB database seemed ideal. The core files in the web server are app.js and getInfo.js. Other systems include the Node.js software package and modules such as: Express for a flexible web application framework, Jade format/template for HTML files, Socket.io to open a web socket between the user and the server, Forever to continuously run the server and mongojs to interface between MongoDB database and our Node.js server. The app.js script contains the code that represents and runs the hypertext transfer protocol (HTTP) server. Its main function is to retrieve the data from the user who accesses the website and to store the information in the server's database. A secondary function is to match the visiting user's data to previous records to check if it matches. If it does, we can identify the user after multiple visits at different time spans.

The Node.js software provides a lightweight alternative to other server models, perfect for short-term experiments as the one carried out in this study. One drawback of this software is that a session must stay open by a secure shell client so that the server can run. To circumvent this, we can use server scripts as a daemon to keep the software running. We use Forever, a command line interface, which is able to run a script for an indefinite amount of time. The script will stop when we run the command "stop", connection is lost, or server crashes. Yet, Forever ensures that if the script is terminated prematurely, it will execute again.

The other core file, getInfo.js, runs on the client-side and retrieves as much information from the browser and the operating system of the visiting user. Among the obtained information are the web browser that the user used to access the webpage, along with the language that is being used, type of operating system the machine is using (e.g. Windows, Mac, Linux, etc.), information about the plugins that the web browser has installed and the IP address and port number the user is connected to.

The IP address can properly identify a user that is not behind a proxy. However, solely by the IP address, we may not identify an individual who is part of a LAN. We often can filter out users by using their operating system, browser and plugins. Therefore, we can single out a specific computer from a network. The rationale of using plugins is that users tend to visit various websites regularly, and most likely downloads applications that make browsing easier and more efficient. This action is almost unique, meaning each user can have a collection of diverse plugins that almost no other user has. Hence, plugins act as quasi-identifiers.

Furthermore, we can often infer behavioral information about the user. In particular, we can identify an individual if they access the page repeatedly. This approach along with other information, such as the IP addresses and browser information can uniquely identify most users with a very high success rate. The implementation of the server and client-side script also includes generating unique keys that are used for experiments and testing our approach.

6 EXPERIMENTS

6.1 Overview

To prove that our webpage approach works, we conducted a series of tests. To conduct even more stressed tests, we called for experiment participants throughout our department students. Notice that when participants are from a smaller community with a lot of similarities, it is harder to identify them compare to a case that participants are diverse. Furthermore, to have participants confident about the research intentions, we ensured that no information that may harm them if handled wrong was going to be obtained. In particular, *Table 1* illustrates types of information we retrieved. We recorded the application version of the web browser, the default language setting of the user, the type and version of the operating system, the IP address, port number, time of access and the plugins the user has installed. If the user accepts the prompt that appears when the webpage is first visited, we can also obtain the user's coordinate (latitude and longitude). The database stores all this information along with the unique key that is generated, the date and the time the webpage is visited.

A total of 110 participants completed our various experiments, providing a small scale, but useful, preliminary result of the study. Since the population is limited, the numerical results present some deviations. We split the experiments in two, to study each approach independently and also to monitor users' behavior in each approach separately. It is important to note that when the population is restricted to a smaller community (such as students of a department) in which subjects are more likely to present similar devices, it is in fact harder to uniquely identify them correctly.

We sort the information based on which keys have the same IP address, and similar plugins if needed. Then, a list of pseudo-distinct devices is generated based on the information gathered by the server, accompanied with a list of keys that match that same specification. The importance of this approach is that the sort is actually blind, meaning we have no external knowledge on the users who visited the target webpage.

Table 1: Device Information Retrieved.

Parameter	Information
App Version	Contains the user's browser version and type (Firefox, Internet Explorer, etc.)
Operating System	The device's operating system, e.g.: Mac OSX, Linux
Plugins	Plugin components of the browser
Browser Language	User set browser language: English (en-us), Spanish (es-us), etc.
IP Address	The IP Address of the time of connection
Geolocation	(Optional) – User's location coordinates.
Date and Time	The date and time the user accessed the webpage.

The results of the experiments serve to address two main concerns. One is whether we can identify and separate individual users based only on the information of their pseudo-distinct devices. The answer is, with a high success rate, "yes". To confirm it, after the experiments were done, we manually compared the actual users against gathered information. Also, the users were asked to send their information anonymously. We then use the information the server generates, the date and time, to infer behavioral information about the user. In particular, via email, the users were asked to confirm their participation in the test: every day, they received an email from us prompting them to visit our target webpage and report us latest information they see there. On some occasions, some participants did not visit the page, for any reasons they had, although the majority of them continued visiting the page every time they were prompted to. Also, some users may skip visiting the webpage, in which case, they just send their previous unique keys. This added a layer of realistic behavior, since users are not obligated to check their email every day or to visit every page they are asked to. Consequently, the results are skewed because the users' behavior is not controllable. For the experiments, we focus more on desktops and laptops as pseudo-distinct devices, as the following section covers.

6.2 Pseudo-distinct Device

We define the devices found on the database as pseudo-distinct, since it is not certain that an entry is a single individual computer. Participants of our experiment are students of our department. They would have access to the supplied university network, and as such, any computer that connects to our webpage will be assigned the private IP address, which corresponds to that of the university. Adding to this, students are provided access to various laboratories, where most computers have the same specifications and the department provides accounts. Such computers are not private, so students tend to not install as much software as they would on their own personal computers. Also, students have limited space on which they can download and install software. As one can guess, these conditions would put our identifying algorithms under a lot of stress.

On the other hand, pseudo-distinct devices share operating systems, IP addresses and plugins are not limited to laboratory computers. Our webpage can also identify whether the user connected through a smart phone or tablets. If the user connects through their service provider, i.e. a cellular phone company, then we can say that their device is more distinct than other devices. Users can also connect through the wireless network. This can show up as a single device due to how the database sorts the keys.

These devices are sure to have various keys associated to them, making it difficult to identify a single user. If access to connection logs were available for this study, we can then compare those to the time and keys provided by the database to properly single out an individual. Not only would we have specific information about the computer used to connect to the webpage, but specific information about the user, if institution complies.

6.3 User Identification Test

The first part of the experiment consists of identifying unknown users based on information gathered from their devices. We prompted the participants to visit our target webpage for six days whenever they can. Every time, a user visits the webpage, they send back us a key as the information they received from the page. We stored those keys and respective participants in a spreadsheet, for manual verification of the results of our algorithm. Such information was obviously hidden from the participants who conducted the experiments, just to avoid contaminating the study. Every time there is a new visitor (not necessarily a new user) in the webpage, their device information is collected and a unique key is generated. The webpage uses the collected information in an algorithm to map the current visitor to one of the previous ones, or to open a new record in the database if it is the first time that the visitor (or its device) is browsing the page. Then, we sort the keys in the database and compare them with the keys sent by users. To confidently prove that a computer is an individual entity, we need all unique keys sent from the same machine to exactly match with information we collected in each visit.

As users' activity is recorded during each visit, we can infer behavioral information about them. This information can give us an idea as to where the user lives, when they usually check their email and what kind of content they view on internet.

6.4 Anonymous Email Test

We also asked participants to create (or use) an anonymous (or private—if sounds anonymous) email and communicate with that email without revealing their real identity. Then, at some point, we prompted all participants both through their real email addresses as well as through their anonymous emails to visit the target webpage. As many users check their email accounts consequently, we anticipated receiving two entries in the database not too distant apart from each other.

We initially gave the entries a grace period of five minutes; yet, the initial test resulted in an appropriate gap of six minutes and thirty-four seconds for our slowest participant. We continued to repeat this test with all participants (through their both email addresses); we compared the keys in the database to those the users sent in. Using keys from this test in conjunction with the keys from the other test we were able to verify what anonymous email account belonged to which actual participants.

Our method provides accurate results when the user is consistent in using the same device to check their email, and as such, participants that used a cellular phone connection or campus internet are harder to identify. We can also add to this that if the participants accessed only one of the email accounts or forgot to check both accounts at almost the same time, we cannot safely deduce whether the two email accounts belong to one user. Note that we mainly used the user's IP address and access time.

We also took into account the number of devices each user has connected from, and comparing it to the number of devices we know belongs to that user. This yields results similar to the first part of the experiment. Results indicate that our approach matched users by 81.1% successes. We assume the user accessed the webpage from both emails through the same device. This assumption takes into consideration users that accessed the website from a different device each time, used a LAN connection to carry out the last part of the test, and did not send every key required for the test.

If the user does not check both emails one after the other, this method has less reliability. Also, in this method, time of connection is very useful, in particular in small populations of suspects. The ideal case would be for the case of one suspect. If that one user checks both emails at the same time, we can identify that user with a very high confidence.

6.5 Results

As mentioned earlier, 110 participants completed our two types of experiments. Every participant had access to the campus wireless network. This makes it harder to uniquely identify a user from another. Yet, it would be interesting to apply the algorithm to bigger and more diverse population scattered along a bigger geographical area. The important question there would be whether the diversity of users would cancel out (or at least significantly reduce) the stress imposed with the big population.

In the seven-day period of the experiments, the webpage was visited 475 times by non-mobile devices. This gives an average of 4.32 visits per user. We found out that 86.17% of the user entries were unique (in the sense of pseudo-distinct device components, explained above). Users connected to the server using 162 pseudo-distinct devices. This provides an average of 1.5 pseudo-distinct devices per user. This will probably grow in future as more computing devices will access the internet.

In the user verification test, 37 out of the 110 participants were selected. Individual users were recorded to have visited the webpage through their home computers, campus supplied computers, smart phones and tablets connected through both the campus supplied wireless network and their cellular phone carrier connection. The campus supplied computers will set up every time the user logs in for the first time. This means that everything a user downloads in a specific computer may not be available to them on another computer. As such, the plugins the browser uses will not be reliable in identifying a user across different machines. In this case, it is better to use the time of connection if we have access to user logs. If users access the webpage every time through their personal computer, the results yielded will be more reliable. We note that identifying a computer is easy, but for identifying a user, we will need other parameters such as an email address, since it is a more personal tool. (Email verification test proves this).

When a user enters the webpage the algorithm in the server, matches browser components to previous entries based on the Operating System, Web browser, browser language & Plugins. The IP Address was not directly dependable since user's IP Address will change with respect to which location they're in. This initial algorithm yielded an 86.17% successful matching. It would have been a higher percentage but users tend to update their plugins: for instance update a Microsoft Word plugin from version 2010 to 2013. Another added measure to solve this issue was match with a possible saved IP Address. For this issue we introduced the algorithm in Figure 1 to match a user's sequential visits more successfully; this could possibly increase the chance that the user will be correctly matched up to 95.2% (distinct from other computers that access the page).

```
1.  Input: x ➔ "browser software components"
2.  Data Set: B ➔ "unique stored browsers"
3.  M ➔ "Set that stores possible matches"

4.  FOR: b in B
        factor = getFactor(b,x)

        IF: factor = 1
            RETURN B

        ELSE IF: factor > 0.80
            M.add(b)

5.  IF: M.length > 1
        "Match by IP Address"

6.  FUNCTION getFactor (b, x):
        IF: b equals x
            RETURN 1
        ELSE:
            compability_factor (float)

            FOR:  p in b[plugins]:
                FOR: y in x[plugins]:
                    IF: p equals y
                        compability_factor++

            IF:  compability_factor = x[plugins].length
                RETURN 1
            ELSE:
                RETURN compability_factor / b[plugins].length
```

Figure 1: Device Sequential Detection Algorithm.

When we reduce the domain and assume the user will use the same personal computer to access the webpage (which is usually the case), with the algorithm in *Fig. 1*, we can sequentially detect a user's machine through their browser with better accuracy since we track it down by both IP Address and plugin changes. The 0.80 factor is from the worst-case scenario is that users change up to three plugins (updates or downloads) during the period we experimented but other plugins remain intact. For example, if a user has 13 plugins and they change two plugins, 11/13 of the plugins will be the same. This means 84.6% of the plugins are the same before and after the plugin updates.

The anonymous email test provides a more useful method of identifying a user; if, for example, we want to prove whether a suspect is a scammer (or a spoofer). These results tend to venture more into behavioral data. Our worst-case scenario uses the approach taken in the first part of the experiments. Users were recorded to check their various email accounts from a myriad number of devices and not always recorded their unique keys. The email verification test shows that 81.1% of the machines were completely matched through email services.

6.6 Behavioral Data

Along with the information that we set out to retrieve, we obtained also some extremely useful data concerning how a user acts on the internet. On a LAN connection, we can use its operating system to identify someone in a population containing a large range of different OS's. This can help us isolate a group of users to handle issues they may present. Plugins present very useful information concerning the user's browsing habits. Plugins also reflect what software they have installed on their computers, making it an extremely useful marketing tool.

From the information we obtained, we can clearly see uniqueness in the parameters: 333 different plugins, 10 different language settings, 4 different operating systems, 304 different IP Addresses and 96 different browser app versions.

The plugins contain information about media players that a device has installed and information directly related to helping the browsing experience of certain websites. These websites include popular social networks, so developers who are interested in understanding where a user spends most of their time online can help them target those users, and either incorporate elements of those social networks or implement their own version of software that proves useful to users. Another application is video game developers as plugins often reflect what games they play. This can give developers insight on whether their competition is strong, or whether they have market dominance.

We have already presented some important identifying information one can easily obtain form users browsing the internet, concerning the software properties. Furthermore, certain programming languages—such as C++—can obtain information about what type of hardware, e.g. video card model and type of the processor, is used. This gives the developer more useful insight into how they should build their software, and what kind of usage they can expect from the user.

7 CONCLUSION

Users' security and privacy have been important factors in communication and computer sciences. This study has shown the limits of what type of data one can retrieve from a user without their consent and how useful such data can be towards identification. We showed that the combination of IP address, port number, operating system, browser version, and plugins installed could provide amazing insight as to what the identity of a particular user is and how that user behaves on the internet.

As an application, this non-consent information retrieval can highly identify the sender of an anonymously sent email. This can be applied to various users to increase chances of success by process of elimination.

Our web server implementation was able to provide the amount of information necessary to identify someone on the internet. Keeping track of a user throughout a period of time proves to be difficult, even though someone uses different devices connected through different networks. Our experiments show that we can extract useful information about that individual through periodic observations; by recording their online behavior and the browsers' components and configurations. We found that 86.17% of the users were successfully uniquely identified. In particular, in the case of verifying whether a suspected user has malicious intent, we can apply the second part of our experiment. Through careful observation, we can infer behavioral data. Although the smaller the group of suspects is, the more accurate their information and our matching is, we would suspect that the diversity of users and machines in bigger groups would significantly reduce—if not cancelling it out—the stress imposed by the big population as future direction of this research.

ACKNOWLEDGEMENTS

We would like to acknowledge the Department of Electrical and Computer Engineering at the University of Puerto Rico at Mayagüez for funding

this research.

REFERENCES

M. Abadi and B. Blanchet, 2005, "Computer-Assisted Verification of a Protocol for Certified Email," *Science of Computer Programming,* pp. 3-27.

F. Sanchez, Z. Duan and Y. Dong, 2011, "Blocking Spam By Separating End-User Machines from Legitimate Mail Server Machines," in *CEAS*, Perth, Australia.

F. Sanchez and Z. Duan, 2012, "A Sender-Centric Approach to Detecting Phishing Emails," in *ASE International Conference on Cyber Security*, Washington D.C.

G. Ateniese and C. Nita-Rotaru, 2002, "Stateless-Recipient Certified E-mail System Based on Verifiable Encryption," *Proceedings of The ryptographer's Track at the RSA Conference on Topics in Cryptology,* pp. 182-19.

M. Abadi, N. Glew, B. Horne and B. Pinkas, 2002, "Certified Email with a Light On-line Trusted Third Party:," *Proceedings of the 11th International Conference on World Wide Web,* pp. 387-395.

P. Eckersley, 2010, "How Unique Is Your Browser?," in *Proceedings of the Privacy Enhancing Technologies Symposium*.

M. Ter Louw, J. Soon Lim and V. N. Venkatakrishnan, 2007, "Extensible Web Browser Security," in *4th International Conference, DIMVA*, Lucerne, Switzerland, pp. 1-19.