



# Bypassing Device Identification

Device Identification is considered by many to be a viable solution to phishing and other client-side attacks due to the fact that even when a phishing attack is successful, the attacker has no way of enrolling his/her device with the website. Notwithstanding, this paper discusses four different, yet very simple attack vectors that can be used to completely overpower device identification.

2007© All Rights Reserved.

Trusteer makes no representation or warranties, either express or implied by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect special or consequential damages. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of Trusteer. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this publication, Trusteer assumes no responsibility for errors or omissions. This publication and features described herein are subject to change without notice.

## Table of Contents

<b>Introduction</b> .....	<b>3</b>
<b>1. Device Identification</b> .....	<b>3</b>
1.1 What is Device Identification .....	3
1.2 Examples of Device Identification .....	3
1.2.1 Regular (HTTP) Cookies .....	3
1.2.2 Flash Local Shared Objects (“Flash Cookies”) .....	4
1.2.3 Browser (and O/S) properties .....	4
1.2.4 A negative example: source IP address .....	4
1.3 Enrolling a new device .....	4
<b>2. Attacks that Defeat Device Identification</b> .....	<b>5</b>
2.1 Malware .....	5
2.2 Session Hijacking .....	7
2.3 Browser Vulnerabilities .....	8
2.4 Man-in-the-Middle .....	8
<b>Summary</b> .....	<b>10</b>
<b>About Trusteer Rapport</b> .....	<b>11</b>

## Introduction

Device identification (also called “machine identification”) is widely used today by financial organizations to reduce fraud. This system “learns” the devices each user uses to access the website and warns when the user, or an attacker, tries to sign into the website using an unrecognized device.

It is a common belief that device identification can be used to defeat most client-side attacks such as phishing, pharming, and keyloggers. This paper demonstrates three very simple attack vectors, all variants of known attacks, which completely bypass device identification. This paper further explains why device identification is not a real-time barrier for attackers and why online businesses cannot rely on it to reduce fraud.

## 1. Device Identification

### 1.1 What is Device Identification

Device identification is a technology that allows a website to uniquely (or almost uniquely) identify a client machine (e.g. desktop, laptop). Device identification plays a major role in online transaction risk analysis. It is based on the observation that a login or a transaction conducted by a user is less risky if it originates from a device (computer) the user has already used in the past.

Device identification is effective against naïve phishing and keylogging attacks where the user’s credentials are compromised and the attacker utilizes them from a different machine.

Device identification can be a combination of several technologies, as surveyed below.

### 1.2 Examples of Device Identification

#### 1.2.1 Regular (HTTP) Cookies

This mechanism operates in the following manner: the website places a permanent cookie on the user’s device, which uniquely identifies the device. The website verifies the cookie each time the user signs in. If the user signs in without the cookie or with an invalid cookie value, the website assumes that the user is connecting from an unrecognized device.

Cookies are commonly used to identify clients. This mechanism is well established through web standards and practice. However legitimate they may be, privacy-oriented users tend to erase cookies frequently, thus diminishing the effectiveness of cookies as a long time device identifier. There are tools which can erase cookies automatically when closing the browser.

### 1.2.2 Flash Local Shared Objects (“Flash Cookies”)

This mechanism is very similar to the cookies mechanism. Instead of storing a cookie on the user’s computer the website stores a unique flash file that it uses to identify the computer.

The Flash client technology (by Adobe) has its own local permanent storage, which is comparable to HTTP cookies. Moreover, the Flash client is ubiquitous, which makes a great deal of sense in its use. Lastly, unlike HTTP cookies, Flash cookies are less known and as such are much less subject to premature erasure.

### 1.2.3 Browser (and O/S) properties

The browser type, version, and related information (operating system, patch level, supported players, screen resolution, time zone) can be obtained either from the User-Agent HTTP request header, or by running Javascript on a page rendered by the browser (using the Navigator DOM object). This information is usually always available. The website is able to store this information in its database and validate it each time the user tries to sign in. However, parts of this information may change (e.g. upgrading the operating system, installing patches, installing/removing support to various media formats), so considerable care and attention must be paid when handling such situations.

### 1.2.4 A negative example: source IP address

A very naïve suggestion would be to use the source IP address as a differentiator between machines. However, this approach is bound to fail due to multiple reasons:

- Dynamically assigned IP addresses (by ISP's) would cause the same device to be regarded as two different devices. However, as long as the user connects from a single ISP, that ISP can be identified (as a group of IP's, rather than identifying the user by a single IP)
- A user may connect through hot-spots. The device remains the same, but the IP's are now owned by different ISP's. This, therefore, effectively dismisses the “group by IP” idea above.
- Corporate/ISP Proxy server triggers all of the users from the respective corporate/ISP to be observed as coming from a single IP.

It seems, therefore, that source IP addresses are not a good choice for device identification.

## 1.3 Enrolling a new device

There are three main operational problems associated with device identification. The first problem is enrolling the system for the first time. The second problem is allowing users to connect from multiple places (for example from home and from work) and the third problem is allowing

users to connect from public computers. In all three scenarios, the website does not have any identifying information on the device used by the user. The website therefore needs to learn these device identifiers and place a cookie or a flash file on each. However, before doing that, the website needs to be sure that the request was made by the authentic user and not by an attacker who stole the user's username and password and is now connecting from an unfamiliar device.

The above is often referred to as elevated security procedure, which compensates for the riskier situation encountered (wherein the user does not log in from an enrolled device). Once the user successfully logs in through this elevated security process, the new device is typically registered with the user.

The elevated identification process typically involves asking the user questions that only the user is supposed to know ("out of wallet" or "challenge" questions). Oftentimes, such questions are user-specific (the user may have registered them earlier, possibly offline), and are not necessarily "predictable" (i.e. different users may be asked different questions). This way, even when an attacker collects user information through a phishing attack or a keylogger, the attacker will have a hard time registering his/her device in order to sign in due to those "out of wallet" questions. Another method used in an elevated identification process is sending an activation key via SMS to the user's cellular number. The user then needs to provide the activation key in order to complete the sign in process. Attackers that steal sign-in credentials and try to sign in from a different device will not be able to get the activation key and complete the sign in process.

## 2. Attacks that Defeat Device Identification

This section reviews three classes of attacks that completely bypass device identification.

### 2.1 Malware

Using a malware is probably the easiest way to bypass device identification. A malware is a piece of software that sits on the victim's computer. A malware that overcomes device identification would collect device identification parameters from the victim's computer and send them to the attacker. The simplest way to achieve that is by copying the JavaScript code used by websites to enforce device identification and then run it on the victim's machine. The JavaScript code collects all the information required to complete a successful device identification process. Note that since it is a JavaScript code, the attacker can actually view it by simply choosing "view source" from the browser's menu. In addition, the attacker needs to collect all cookies and flash objects that are associated with the targeted website.

Here is a summary of the steps an attacker needs to perform in order to bypass device identification using a malware:

1. Browse to a targeted website (for example a bank application) which uses device identification

2. In the website's login form, view source and look for the JavaScript code that collects device parameters (such as OS version. See [1.2.3](#) for more information)
3. Copy this code and use it in the malware to collect the same parameters from the victim's computer
4. If the targeted website is not using a JavaScript code to collect device parameters then it probably uses HTTP headers to collect this information. In this scenario, include a code that copies all HTTP headers from the victim's computer
5. Add a simple code that collects all cookies and flash files that are associated with the targeted website
6. Add a code that sends this information back to the attacker
7. Distribute the malware and wait for the results

Deleted: 1.2.3

Once the attacker receives this information from the malware a few additional steps are required in order to ultimately sign into the victim's account:

1. place the cookies and the flash files on the attacking computer
2. install a local proxy such as Paros<sup>1</sup> which allows you to control and edit requests your browser sends to the website
3. browser to the attacked website
4. when receiving the login form with the device identification JavaScript, edit the next request manually and provide the collected identification parameters as inputs, instead of the input the JavaScript collected from your computer
5. using the same technique, change the HTTP headers to match the headers collected from the victim's computer

The above described attack is very simple to execute. Building the malware requires no real expertise and can be completed in a few hours of work, even if the attacker starts from scratch and does not use available code. Building an environment that simulates the victim's environment is also very simple to do and requires only a couple of hours of work.

To sign into the victim's account the attacker would probably use a publicly available network such as a Starbucks WiFi network and would avoid botnetted computers as these computers are often black-listed by device identification solutions. A slightly more sophisticated attacker would also collect network parameters from the victim's computer, such as the ISP used by the victim. The attacker can then register to the same ISP and launch the attack while connected to the ISP network. This would overrule any suspicion the website might have in the attacker's IP address.

---

<sup>1</sup> <http://sourceforge.net/projects/paros>

## 2.2 Session Hijacking

HTTP is stateless and does not include inherent mechanisms to track users. To bypass these problems, web applications are relying on session identifiers that are usually in the form of cookies or HTTP parameters. When the user accesses a website, the website generates a session identifier (a unique number or string) and sends it to the user as a cookie or a parameter embedded inside a form. As a result, the user's browser sends this session identifier back to the website with each HTTP request. This allows the website to easily track HTTP requests originating from the same user. When the user authenticates to the website, the website marks the specific session as authenticated and from now on any HTTP request that includes the specific session identifier would be considered as coming from an already authenticated user. This means that if the attacker manages to steal or hijack the user's session identifier, the attacker would be able to access the user's account without having to authenticate.

The easiest way of executing a session hijacking attack is through a cross-side-scripting (XSS) attack. This attack takes advantage of a website vulnerability in which the website displays content that includes un-sanitized user-provided data. If the website is vulnerable to XSS the attacker can craft a URL that once clicked by the victim would send the victim's session identifier to the attacker. The URL appears legit as it points to the targeted website. However, it allows the attacker to run arbitrary JavaScript code in the victim's computer. This code can grab the victim's session identifier and send it to the attacker. More information on XSS can be found on OWASP's website<sup>2</sup>.

It is reported that about 80% of websites suffer from XSS vulnerabilities. Overall, XSS vulnerabilities are extremely hard to mitigate but are very easy to exploit, therefore making this particular attack vector so compelling.

As XSS allows attackers to run arbitrary JavaScript code on the victim's computer and steal cookies, it is also a perfect platform to bypass device identification. Here are the steps an attacker needs to follow to execute this attack:

1. discover an XSS vulnerability in the targeted website
2. Craft a URL to exploit the vulnerability. The URL should run a JavaScript that directs the user to a fraudulent sign in page. It should steal the targeted website's cookies, flash and identification parameters (see previous section for more information on extracting the JavaScript from the targeted website). It should also grab the victim's credentials when the victim attempts to sign in
3. Distribute the link via email or any other channel (for example IM, search engines, advertisements)
4. Receive the collected information
5. Repeat the process described in the previous section to sign into the victim's account

---

<sup>2</sup> [http://www.owasp.org/index.php/Cross\\_Site\\_Scripting](http://www.owasp.org/index.php/Cross_Site_Scripting)

The attack is very easy to execute. The only barrier here is finding an XSS vulnerability in the targeted website. However, experienced hackers can find such vulnerabilities in almost any website and in a very short timeframe.

To fight this attack the website can use HTTPOnly cookies<sup>3</sup> that prevent a JavaScript from reading the content of cookies. However this mechanism is still not widely supported (e.g. not supported by FireFox) and there are some attacks that bypass it completely<sup>4</sup>.

Another limitation of XSS attacks is the difficulty of accessing flash files through JavaScript code. There are several implementation specific ways to evade the above limitation, such as invoking a Flash object using an OBJECT tag. However, device identification mechanisms would usually accept either cookies or flash, considering that flash can be removed by the user. Thus an attack that grabs the victim's cookies and device parameters would succeed.

## 2.3 Browser Vulnerabilities

Browser vulnerabilities are common. The SecurityFocus Vulnerability database<sup>5</sup> consists of a wealth of vulnerabilities in Mozilla Firefox, Microsoft Internet Explorer, and even Adobe Flash Player.

Attackers can easily exploit existing browser vulnerabilities to access cookies on the victim's computer and use this information to bypass the device identification mechanism. This is accomplished using existing unpatched vulnerabilities, as many users do not patch their systems or using newly released vulnerabilities for which a patch does not yet exist. Such vulnerabilities are sold today by hacking communities for a few hundred dollars.

Exploiting browser vulnerabilities does not require much expertise. It usually involves running the exploit code on the phishing website. Further, device identification mechanisms are completely vulnerable to this attack vector as they totally rely on the browser's security model for their own security. As a result, any breach in the browser's security model can immediately affect this mechanism.

## 2.4 Man-in-the-Middle

In a man-in-the-middle (MITM) attack, the fraudster sits between the consumer and the website and can read, insert, and modify at will, all traffic that passes between the two. In MITM the consumer is directed to a proxy server controlled by the fraudster. This can be achieved in various ways. For example, a slightly modified phishing attack could direct the user to a proxy server instead of a fraudulent website. A slightly modified pharming attack could achieve the same result. Some experts would actually argue that directing the user to a proxy server is much easier

---

<sup>3</sup> <http://msdn2.microsoft.com/en-us/library/ms533046.aspx>

<sup>4</sup> <http://seclists.org/webappsec/2006/q2/0181.html>

<sup>5</sup> <http://www.securityfocus.com/vulnerabilities>

than building a fraudulent website. Widely available tools<sup>6</sup> can help attackers achieve this goal without much effort.

The attacker's proxy server acts as a relay station between the consumer and the attacked website and passes requests and responses back and forth. The fraudster can read all traffic, modify traffic, and even inject traffic on behalf of the consumer or the website.

MITM was supposed to have been defeated by SSL mainly because the SSL protocol incorporates an inherent protection against MITM, namely the certificate warning. However, the protection provided by SSL assumes that the user indeed browses to the correct website, whereas over the last several years, the harsh reality of phishing attacks demonstrates that users cannot be trusted with the task of differentiating between real and fraudulent URLs. Not to mention that only a tiny fraction of users bother to even check the SSL certificate. Hence a MITM is a very feasible attack even when SSL is used.

A MITM attack that defeats device identification works as follows:

1. The attacker starts with a simple phishing or pharming attack that directs the victim to the attacker's proxy server
2. the attacker's proxy server relays the victim's traffic (e.g. login request) to the targeted website and the victim gets the login form
3. the victim fills in and submits the form
4. the attacker stores the victim's sign-in credentials (e.g. username and password) and forwards the form to the targeted website
5. The targeted website processes the form but is unable to find the device identification cookie or flash file. The reason is that the victim's browser did not pass the cookies and the flash files to the attacker's proxy server as its domain does not match the targeted website domain
6. Since the request lacks the required cookies and flash files, the targeted website requires the victim to undergo elevated security procedures, which translates simply to asking the victim some questions or sending an activation code via SMS
7. The attacker's proxy server forwards these question to the victim
8. In today's Internet world, users are used to being asked elevated security, out-of-wallet questions in various situations (ironically they got used to it through the various FFIEC-induced security solutions implemented by banks). Users have a scarce idea when such questions should be asked and when not (in fact, relying on users to recognize usual/weird website behavior has been quite clearly shown in several researches to be futile). Therefore, the victim answers whatever questions he/she is asked. Alternatively the victim enters the activation code received via SMS or email.
9. The attacker's proxy server forwards those answers to the targeted website (and as a by-product, also gets to know the answers).

---

<sup>6</sup> <http://www.computerweekly.com/Articles/2007/08/15/226213/man-in-the-middle-phishing-attacks-to-surge-says.htm>

10. As a result, the targeted website sends new device identification cookies and/or flash files to the attacker's proxy server and enrolls the attacker's proxy server as a recognized device.

The net effect is that the attacker's proxy server is now an enrolled device for the victim. This means that from now on, the attacker can use this machine to sign in anytime with the username and password grabbed from the victim's sign in process. The attacker now has a computer that is identified by the targeted website as the victim's computer.

The sad conclusion is that MITM de-facto defeats the whole purpose of device identification.

## Summary

Device identification is a widely used mechanism to fight fraud. However, it can be effortlessly defeated by attackers. Malware can easily steal device identification parameters, and simpler attacks such as phishing and pharming can be upgraded to simple man-in-the-middle attacks that completely defeat device identification. Session hijacking through cross-site-scripting can also evade device identification. The entire concept is exposed to browser vulnerabilities that can be used to bypass the mechanism. Further, some attacks such as man-in-the-browser are insensitive to device identification to begin with. The bottom line is that device identification is only good against very naïve attacks where the attacker is completely unaware of device identification.

Device identification systems can be improved to overcome these attack vectors. However, for that to take place, a dedicated agent must run on the user's device and cryptographically control the identification process. Rapport from Trusteer implements such a solution. Rapport's device identification is cryptographically generated and changes from one login to another. Stealing Rapport's device identification code using one of the attacks described in this paper is of no use to the attacker as it cannot be used again. Cryptographically changed device identification is one of various security mechanisms used by Rapport to prevent fraud.

## About Trusteer Rapport

The Rapport Protection Layer from Trusteer takes a revolutionary new approach to consumer security by eliminating the root cause for all client-side attacks. The Rapport Protection Layer is a very "lightweight", small "footprint" application (330k) which requires no user interaction. Once downloaded, Rapport runs completely in the background, and effectively disables the "last mile" attacks which cannot be directly controlled by existing conventional applications on the desktop or by remote fraud-detection and authentication systems. Rapport protects against: phishing, pharming, keyloggers, man-in-the-middle, man-in-the-browser, and session hijacking attacks.

[www.trusteer.com](http://www.trusteer.com)

[sales@trusteer.com](mailto:sales@trusteer.com)

+1(646)247-5669