User Tracking: Persistent Cookies and Browser Fingerprinting

Hong Kaing, Michael Risher and Brian Schulte
George Mason University
Volgenau School of Engineering
hong_kaing@yahoo.com, mrisher@masonlive.gmu.edu, brianr.schulte@gmail.com

Abstract

The two most common ways to identify a returning user online is to require a user to provide a username and password and the use of "cookie" data stores. However, usernames and passwords can be stolen and cookies can be disabled and deleted. To make up for these weaknesses, there has been much research done in the areas of "browser fingerprinting" and persistent cookies. A browser fingerprint is simply the collection of readily available information from the client that will help uniquely identify the user, much like how a human fingerprint uniquely identifies a human. A persistent cookie is a cookie, or collection of cookies, that are resistant to deletion. This paper will provide a brief summary of several research studies involving these two topics. It will also document our experience in implementing the persistent cookie concept, which will use as its value our own browser fingerprint identifier.

1. Introduction

1.1 Usernames and Passwords

The concept of identifying users on the internet is not new but new methods of identifying users continue to be developed, evidenced by the continual granting of patents [1] for this very purpose. We are all aware of the concept of targeted advertising, which is an industry that continues to grow. The Interactive Advertising Bureau (IAB) reported online advertising revenue for the first half of 2012 to be at $17 billion. But beyond advertising, there are several other reasons websites desire to properly identify users. Any website that stores bank or credit card information must properly guard that information from unauthorized users in order to prevent fraudulent activity. Content service providers such as Netflix may also want to curb the sharing of accounts between multiple users through the sharing of usernames and passwords.

While usernames and passwords have historically been the preferred method of identifying users, there are some that believe "the age of the password is over [2]." In that *Wired* article, Mr. Honan chronicles how "hackers destroyed my entire digital life in the span of an hour." They were able to get control of his accounts despite the fact that he used long, complex alphanumeric passwords, which is what is recommended. As a result the hackers were able to gain access to his Apple account and delete messages, documents and family photographs. All this was done because they wanted his Twitter handle (aka username), which was "@mat".

The article continues to list out the most common ways passwords are obtained: Phishing, guessing, using keyloggers and password resetting through a company's customer support department. In summary, the article believes the primary reason passwords fail is due to the fact that passwords must be simple enough for humans to remember. This explains why, according to the article, the top two passwords used are "password" and "123456". And there must be contingencies put into place should passwords be forgotten, contingencies such as the customer support password reset that allowed the hackers to gain control or Mr. Honan's accounts.

1.2 Two-factor authentication

While passwords may not be obsolete, we can see numerous examples of where passwords are now being augmented with other means of authentication. The most common being the very popular Two-Factor authentication system [3]. The most common implementation of the Two-Factor authentication system is to not only require the user to enter their own unique password, but then to enter another number generated by a hardware token that is in their possession or to register a phone that is called upon login request.

The Two-Factor authentication system is an improvement over simply requiring a password but physical tokens are more expensive to implement. In fact, a class action lawsuit against the video game company Blizzard has been filed because users were angry over the cost for a required authentication token in order to gain access to their games [7]. Additionally, these tokens may not always be in the possession of the user or simply lost due to their usually small size.

To combat these weaknesses of Two-Factor authentication some companies have developed their own Two-Factor authentication system, such as Facebook's Social Authentication (SA). The SA system requires users who log in from suspicious computers to properly identify friends who

have been tagged in their photo albums. Not only is such system prone to attack by people in the users social circle (who can easily identify friends in photos) but a study has shown that using a web crawling procedure developed in Python, along with the OpenCV toolkit's face detection utility, they were able to successfully guess 22% of SA's tests using only public information (including public photos posted by the victim). However, should any of the victim's friends accept a friend request from the attacker, the success rate jumps to 100% [9].

Even if a company provides the authentication tokens free of charge to the user there are still security problems as there have been documented cases where these tokens have been purposefully given away to unauthorized users [8]. Additionally, contingencies must be put in place so access may be granted in case the token is lost, forgotten are stolen. And these contingencies are usually identical to the ones used to retrieve passwords.

These examples are listed not to infer that Two-factor authentication is not valuable, but simply to show that no authentication system is perfect and users still prefer the lowest cost, least intrusive and most convenient solution available. This is where browser fingerprinting and persistent cookies come in.

1.3 Brower Fingerprinting and Persistent Cookies

With browser fingerprinting and persistent cookies, no additional software or hardware is required. The implementation of both simply uses tools that are most likely already installed in the web browser (JavaScript, Cookies) and information that is easily accessible (User Agent Strings). Advanced users may decide to disable features that can be used to uniquely identify them, such as disabling JavaScript, blocking Flash cookies or altering their User Agent String (UAS). But several studies have shown that these measures may actually have the reverse effect by helping to uniquely identify them as only a small percentage of users go to such extremes as to alter their fingerprint [5, 11].

Let us conclude our introduction by acknowledging that undesired identification of a user always presents the risk of violating privacy rights. However, we have shown clear examples of when identification against the user's wishes is required and debating the issue of privacy rights is not within the scope of this paper.

2. Related work.

A set of persistent cookies termed "Evercookie" was developed by Samy Kamkar in 2010 [6]. Evercookie takes advantage of storing cookies for web clients in multiple locations assuring redundancy if a single cookie is removed from the system. Specifically, Evercookie stores cookies in 13 locations, including: Standard HTTP Cookies, Local Shared Objects (Flash Cookies), Silverlight Isolated Storage, Web History, HTML5 Session Storage and HTML5 Local Storage. So long as a single cookie remains in any location, regeneration of cookies in the other locations is possible. And so long as the Flash Cookie exists, identification across browsers is possible.

The phrase "bits of entropy" refers to how likely a piece of information will be identical between any two random users. For example, if 8 different browsers are equally used by all users, the browser ID is said to have "3 bits of entropy" or a 1 in 2^3 likelihood of having an identical match. For browser fingerprinting, it has been noted that only 33 bits of data is needed to uniquely identify all of Earth's 6+ billion inhabitants since $2^{33} = 8{,}589{,}934{,}592$ [4].

In two separate studies on browser fingerprinting, entropy values are given to different attributes. They are summarized in **Table 1** below:

Table 1: Entropy values of browser attributes

Attribute	Boda Study [10]	Eckersley Study [11]
User Agent String	8.095	10.0
Timezone	2.22	3.04
User ID	9.03	-
All fonts	8.57	13.9
Universal fonts	6.83	-
Detected fonts	7.63	-
Plugins	-	15.4

It should be noted that that Boda study focused on cross browser user identification so it excluded Eckersley's Plugins attribute, since plugins vary between browsers. The highest entropy value for Boda is the User ID. This User ID is script generated identifier derived from the first two octets of the IP address, screen resolution, time zone and list of basic fonts. The highest entropy value for Eckersly is the plugins attribute, which is simply a list of all plugins installed on the browser.

From this table we can see that by using only the User Agent String [12] we should be able to uniquely identify between 273 (Boda) and 1024 (Eckersly) unique users. When the User Agent String is used in conjunction with the User ID or Plugins attribute we can, in theory, identify at least 142,935 unique users. In fact, Eckersley concludes that his fingerprinting algorithm can uniquely identify up to 286,777 unique users.

Another interesting browser fingerprinting technique developed by Mowery [5] is to use benchmark JavaScript execution times to uniquely identify users. The methodology is quite simple, create several custom JavaScript tests and record the execution time for each test as they run on the client's machine. The reasoning behind using these benchmark tests is that system attributes such as the IP address or User Agent String can be falsified, the execution time cannot.

However, even the authors note that there are still several hurdles that must be overcome for these benchmark tests to become usable. The total running time for their 39 JavaScript tests took an average of 190.8 seconds to complete. Clearly this is unacceptable as no user want to wait over 3 minutes for a page (with no actual data to visualize) to load. Additionally, runtimes using the same computer configuration can be very volatile as run times depend on what other programs are currently running at the same time as the JavaScript. Ironically, if you were to disable JavaScript (using NoScript [13]) in an attempt to be "anonymous" their alternative fingerprinting method executes much faster: between 22.2 and 23.3 seconds.

Mowery demonstrated that users who have installed the NoScript plug-in for the Firefox browser could still be uniquely identified. NoScript disables executable code such as JavaScript, Java and Flash. However, since most websites require these plug-ins to function properly users must create a whitelist of websites that can execute code. Mowery's team

created a page that could check for domains in the NoScript white list, utilizing a technique that would load scripts from 689 of the top 1000 sites according to Alexa. For each script that is tested provides 1 bit of entropy for unique user identification.

3. Implementation and Analysis

Our project can be broken down into three different tasks. The first task involves the implementation of the Evercookie concept. Modifications to the Evercookie source code may be necessary to account the for browser changes since the last stable release of Evercookie. Browser specific modifications of Evercookie may need to be implemented in order to get all the advertised features of Evercookie to work.

The second task of our project involves the use of browser fingerprinting concept. Our research has uncovered 189 different attributes (see: **Appendix A**) that can be retrieved, primarily using JavaScript, and used to uniquely identify a user. A subset of these attributes will be used to create a hash value that will serve as the value for the Evercookie.

Lastly, the cookies created will be stored on our test server in a database in order to identify any return users. Should the user succeed in deleting all of our created cookies, the client's attributes can still be obtained for the current session based on its fingerprint. These attributes can be compared to the data stored on our server to determine the most likely user.

3.1 Evercookie

We utilize the 3rd party library "Evercookie" in our implementation to combine persistent cookies with browser fingerprinting. Evercookie utilizes various storage locations for cookies, ensuring redundancy in return user identification. None of these storage locations ever explicitly ask the user for permission to store persistent data.

Evercookie uses a multitude of locations and techniques to store a cookie on the client: standard cookies, local shared objects, Silverlight storage, storing in cached images, web history, HTTP ETags, web cache, window.name caching, Internet Explorer userData storage, and various HTML storage locations.

Standard cookies are simply utilizing the default storage location for the client browser. Next, local shared objects are used, which are also known as Flash Cookies. By default, the Flash Player does not require a user's permission to store objects on the file system which is taken advantage of by Evercookie to store their persistent cookies.

Silverlight provides a virtual file system to store information for trusted applications. In our tests, explicit permission was not needed to utilize the virtual storage location, so Evercookie was easily able to store its cookie there as well.

The PNG caching works by storing the cookie inside of an image file. When created, the cookie is passed to a special script that creates an image with each RGB value set to the data of the cookie. It is then sent back to the client, which is instructed to cache it for an extended period of time. When attempting to retrieve the cookie, the script will send back a response of "304 Not Modified", forcing the client to look in its local cache for the stored PNG file.

ETags are used as an additional field used by HTTP for web cache validation. A server can return an ETag value along with a web asset which will in turn get cached by the client. Upon subsequent requests, the client will pass along the ETag value with its request.

Web cache cookies simply utilize the default cache in a client to store a persistent cookie. The space available using the window.name property is also used for cookies. There is storage available for all current browsers through the use of the DOM property window.name. A drawback of this type of storage is that it is cross-domain, meaning that other websites can theoretically be able to read the data as well.

HTML5 also offers multiple storage locations for Evercookie to store its persistent cookies. The global storage that Evercookie previously used is outdated and no longer supported and does not seem to be working with current versions of the browsers in our tests. Local storage is still available and provides storage for each domain to use where a cookie can be stored. The local storage is persistent and has no expiration date meaning that it will reside on the client until a user explicitly deletes the data. Session data is very similar to local storage data, although it is only available until the current session concludes. Due to this restriction, session data is not as reliable as other methods for persistent cookies. Lastly, database storage is provided within HTML5 and allows storage on a local client's database.

The combination of all of these techniques provides astounding persistence of cookies on a client's browser. If any of these cookies are deleted, as long as one still persists, the other locations will be refilled with the cookie and a server can remain able to identify the return user.

3.2 Fingerprinting

The largest difference between our project and Evercookie, is the fingerprinting feature. This allows us to track users even in the event that the user has been able to remove all of the tracking information from their computer.

Our fingerprinting technique is based on the prior work done by many others before. We aggregated a list of over 180 different attributes that can be used to create a unique fingerprint. The list of attributes that we choose to use was designed to accomplish two goals. First, we wanted to choose few enough attributes such that it does not impose a runtime load on the browser. Secondly, we need to choose attributes that can uniquely identify the browser product and version and then uniquely identify the settings and customizations that the user has set within their environment. While JavaScript in critical in being able to generate a fingerprint for the browser, however in order to retrieve the highly unique attributes requires either Adobe Flash or Oracle's Java to access these fields.

The attributes that we choose to use accomplish the goals above. We track the UAS which provides the most uniqueness for the browser product and version. The ability to capture the list of available fonts and browser plugins provide the most uniqueness in identifying the user (see **Table 1**), however, these attributes are easily modifiable by users and we can expect them to be updated at regular intervals.

3.3 Tracking the user

In our implementation of persistent cookies, we keep a database on the server comprised of user's cookies and a user number to identify them. On our test site, once a cookie is created by a user's client, it is then uploaded and added to our

database if it is not already present. There are also buttons to return which locations contain the cookie for the user both with and without utilizing Evercookie's mechanism to replace deleted cookies. This allows us to test what methods actually work for attempting to delete these persistent cookies.

We have also implemented the ability to make a guess as to which user a visitor is based on the client's fingerprint. Since our cookie values are based on a hash of various information obtained about a client we can use that information to make a guess as to which user a visitor to our site is. Using this method, even if a user manages to completely erase all evidence of the persistent cookie, if they had previously been registered as a user in our system we can reinstate the cookies based on the browser fingerprint.

Our implementation could have implemented additional features to track a particular user across the internet should the product be included in websites across the Internet. Additionally we could extract information from the browser's header to get the referring website. Combining all of this information would give us a very complete picture of a user's activities across the Internet.

3.4 Limitations

Browsing while utilizing browser's various stealth browsing modes will thwart the attempts of Evercookie to establish a persistent presence on the client. A cookie can be created and stored within the private browsing session, but once it is concluded the persistence is lost as all data that had been stored during the session is erased. Some browsers, however, do not erase all of the locations that Evercookie stores data allowing some functionality during stealth mode browsing.

It is possible for a vigilant user to remove all cookies stored in the various locations that Evercookie uses. However, for the typical user this is not a simple task. For instance, to stop Flash cookies, the user would need to alter the Flash settings, restricting the storage abilities of the Flash player. In order to clear the data saved by Silverlight, the user would need to navigate to the location on the file system where the data resides and manually delete it.

Since the code used for this project is stored on a private server, we had limited access to test a large number of users creating cookies. Because of this we were not able to produce enough results to test for possible collisions.

4. Future Work

For future work with this project we would like to be able to establish a new location to be able to store cookies. All our research indicated that there is no easily available places to store a duplicate cookie for redundancy. One possibility could be to use the file system API provided by HTML5. The issue with this approach would be the explicit permission needed from the user to utilize this functionality. While good for typically usage of file I/O, this would not allow us to covertly track a user as Evercookie can currently accomplish.

Another feature that we would like to capture is GPS data which has become increasingly available with HTML5 APIs and the increased use of mobile devices with built-in GPS devices. For purposes of authentication, we can use GPS data to determine if the user's current location is reasonable when compared to the previously known location given time needed to travel to the new location. For purposes of tracking, a study by de Montjoye [14] was able to show that even anonymized GPS data can be used to uniquely identify someone with as few as 4 points with over 50% accuracy and with 11 points able to achieve over 95% accuracy.

Additionally, improvements are needed to the fingerprinting algorithm to support inexact matching. The attributes used in browser fingerprinting can and will change over time. Browsers and plugins will be upgraded; Plugins and fonts will be installed or removed causing a slightly different fingerprint. While this fingerprint wouldn't match exactly, it would be quite close to the original fingerprint and fuzzy searching should be used to find the closest matching value. This would be tricky because it could be discarding a legitimately new fingerprint value. We need to track these changes in the fingerprint and update the database to track the user across multiple different fingerprints.

5. Conclusion

Tracking return users has been a goal for many organizations across the web. Most sessions are forgotten once a user deletes their cookies and a server can no longer identify them as a return user. We have researched methods in providing resiliency for cookies in clients and have implemented a system, which makes it very difficult to remove tracking measures. We utilize a library called "Evercookie" to perform redundant cookie storage on a client. We combine this with browser fingerprinting to provide unique values for cookies as well a method for reinstating a cookie even after complete deletion by a user. Our implementation provides a resilient method for organizations to be able to identify return visitors to their websites.

6. References

[1] Mu, Ruicao, T. Hu (2012), U.S. Patent No. 8,126,816 B2. Washington, DC: U.S. Patent and Trademark Office.

[2] Honan, Mat (November, 2012). "Kill the Password: Why a String of Characters Can't Protect Us Anymore", *Wired*, available: http://www.wired.com/gadgetlab/2012/11/ff-mat-honan-password-hacker/

[3]Two Factor Authentication: http://en.wikipedia.org/wiki/Multi-factor_authentication

[4] Narayanan, Arvind, "33 Bits of Entropy", Retrieved, 4-6-2013, from: http://33bits.org/about/

[5] Mowery, Keaton, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting Information in JavaScript Implementations", *Proceedings of W2SP 2011*, IEEE Computer Society, Oakland, CA, May 2011, pp. 1-10.

[6] Vega, Tanzina (October 10, 2010), "New Web Code Draws Concern Over Privacy Risks", *nytimes.com*, Retrieved 4-6-2013, from: http://www.nytimes.com/2010/10/11/business/media/11privacy.html

[7] Kain, Erik (November 10, 2012), "Blizzard Responds To Class Action Lawsuit Over Security Concerns", *Forbes.com*, Retrieved: 4-6-201, from: http://www.forbes.com/sites/erikkain/2012/11/10/blizzard-responds-to-class-action-lawsuit-over-security-concerns/

[8] Hill, Kashmir (January 16, 2013) "Software Developer Who Cleverly Outsourced His Job To China Betrayed By His Digital Footprint", *Forbes.com*, Retrieved 4-6-2013, from: http://www.forbes.com/sites/kashmirhill/2013/01/16/software-developer-who-cleverly-outsourced-his-job-to-china-betrayed-by-his-digital-footprint/

[9] Polakis, Iasonas, M. Lancini, Georgios Kontaxis, et. al, "All Your Face Are Belong to Us: Breaking Facebook's Social Authentication.", *Proceedings of 2012 Annual Computer Security Applications Conference*, Applied Computer Security Associates, Orlando, FL, December 2012, pp. 399-408

[10] Boda, Karoly, A.M. Foldes, G.G. Gulyas and S. Imre, "User Tracking on the Web via Cross-Browser Fingerprinting", *16th Nordic Conference on Secure IT Systems*, Copyright 2012 Springer Berlin Heidelberg, pp. 31-46.

[11] Eckersley, Peter, "How Unique Is Your Web Browser?", *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, Copyright 2010 Springer Berlin Heidelberg, pp. 1-18.

[12] User Agent String format: http://en.wikipedia.org/wiki/User_agent#Format

[13] G. Maone. No Script. Online: http://noscript.net/

[14] de Montjoye Yves-Alexandre, Hidalgo, Cesar, Verleysen, Michael, Blondel, Vincent. "Unique in the Crowd: The privacy bounds of human mobility". Scientific Reports Issue 3. 25 March 2013.

7. Appendix A: Available Attributes

List of available attributes along with the method of obtaining the attribute, if known, and how often that attribute can change between user web sessions.

attribute	method	volatility
accept	http header	stable
charset	http header	stable
encodings	http header	stable
language	http header	stable
activex	javascript	stable
geckoactivex	javascript	stable
adobe reader	javascript	Semi-stable
cookie	http header	volatile
user agent	http header	Semi-stable
appName	javascript	Semi-stable
appCodeName	javascript	Semi-stable
appVersion	javascript	Semi-stable
appMinorVersion	javascript	Semi-stable
vendor	javascript	Semi-stable
user agent	javascript	Semi-stable
oscpu	javascript	stable
platform	javascript	stable
securityPolicy	javascript	stable
onLine	javascript	Semi-stable
browser.name	javascript	Semi-stable
browser.version	javascript	volatile
layout.name	javascript	Semi-stable
layout.version	javascript	Semi-stable
os.name	javascript	stable
Operating System	javascript	stable
alpha	javascript	
aolversion	javascript	stable
backgroundSounds	javascript	stable
beta	javascript	
comment	javascript	Semi-stable
cookies	javascript	
crawler	javascript	stable
cssVersion	javascript	stable
frames	javascript	stable
iframes	javascript	stable
javaapplets	javascript	Semi-stable
javascript	javascript	Semi-stable
parent	javascript	stable
tables	javascript	stable
vbscript	javascript	stable
win16	javascript	stable
win32	javascript	stable
win64	javascript	stable
isMobile	javascript	stable
isSyndicationReader	javascript	stable
ActiveBorder	CSS/Javascript	stable
ActiveCaption	CSS/Javascript	stable
AppWorkspace	CSS/Javascript	stable
Background	CSS/Javascript	stable
ButtonFace	CSS/Javascript	stable
ButtonHighlight	CSS/Javascript	stable
ButtonShadow	CSS/Javascript	stable
ButtonText	CSS/Javascript	stable
CaptionText	CSS/Javascript	stable
GreyText	CSS/Javascript	stable
highlight	CSS/Javascript	stable
HighlightText	CSS/Javascript	stable
InactiveBorder	CSS/Javascript	stable

InactiveCaption	CSS/Javascript	stable
InactiveCaptionText	CSS/Javascript	stable
InfoBackground	CSS/Javascript	stable
InfoText	CSS/Javascript	stable
Menu	CSS/Javascript	stable
MenuText	CSS/Javascript	stable
ScrollBar	CSS/Javascript	stable
ThreeDDarkShadow	CSS/Javascript	stable
ThreeDFace	CSS/Javascript	stable
ThreeDHighlight	CSS/Javascript	stable
ThreeDLightShadow	CSS/Javascript	stable
ThreeDShadow	CSS/Javascript	stable
Window	CSS/Javascript	stable
WindowFrame	CSS/Javascript	stable
WindowText	CSS/Javascript	stable
Components	javascript	Semi-stable
max connections per host	javascript	stable
using proxy	javascript	stable
cookies enabled	javascript	Semi-stable
JS cookies enabled	javascript	Semi-stable
server cookies enabled	javascript	Semi-stable
HTTP only cookies	javascript	Semi-stable
Can JS read cookies?	javascript	stable
meta tag cookies	javascript	stable
max JS cookies per server	javascript	stable
max size per cookie	javascript	stable
browser history	CSS/Javascript	volatile
date/time	javascript	volatile
Date locale format		stable
timezone offset	http header	stable
diff between client and server time	http header	Semi-stable
directX	IE	Semi-stable
HTML support	javascript	stable
XML support	javascript	stable
Views support	javascript	stable
StyleSheets support	javascript	stable
CSS support	javascript	stable
CSS2 support	javascript	stable
Events support	javascript	stable
UIEvents support	javascript	stable

MouseEvents support	javascript	stable
MutationEvents supports	javascript	stable
HTMLEvent support	javascript	stable
Traversal support	javascript	stable
Range Support	javascript	stable
DNT	http header	Semi-stable
.NET framework	IE	stable
flash version	Flash	volatile
flash version	javascript	volatile
navigator.plugins	javascript	volatile
flash platform	Flash	stable
flash major version	Flash	stable
flash build version	Flash	volatile
flash capabilities	Flash	stable
flash JS bridge	javascript	stable
font count	Flash	volatile
font list	Flash	Semi-stable
font count	java	volatile
font list	java	Semi-stable
google gears	javascript	Semi-stable
geolocation support	javascript	Semi-stable
IP city	http header	volatile
IP zip	http header	volatile
IP lat/lon	http header	volatile
gzip browser	javascript	stable
gzip JS	javascript	stable
gzip CSS	javascript	stable
http version	http header	stable
image format support	html/javascript	stable
country	http header	Semi-stable
host name	http header	volatile
IP address	http header	volatile
java version	javascript	volatile
java enabled	javascript	Semi-stable
java support	javascript	Semi-stable
js support	javascript	Semi-stable
no script support	javascript	stable
js version support	javascript	Semi-stable
data tainting support	javascript	stable
encoded Jscript support	javascript	stable
Jscript support	javascript	stable
external javascript support	javascript	stable
system language	javascript	stable
user language	javascript	stable
browser language	javascript	stable

JS constants/calculations (accuracy)	javascript	stable
MathML support	http	stable
MIME associations	javascript	stable
cache control		
pragma		
document.all support	javascript	stable
anchors	javascript	stable
forms	javascript	stable
getElementById	javascript	stable
getElementsByTagName	javascript	stable
documentElement	javascript	stable
images support	javascript	stable
layers support	javascript	stable
links support	javascript	stable
frames support	javascript	stable
regex support	javascript	stable
option support	javascript	stable
Security.getProviders()	java	stable
crypto. “hash algo”	javascript	stable
crypto.algorithms	javascript	stable
open office installed		Semi-stable
available plugins		volatile
helper components		volatile
prefetch support		stable
security manager version	javascript	Semi-stable
quicktime installed	javascript	Semi-stable
quicktime version	javascript	volatile
realplayer installed	javascript	Semi-stable
realvideo installed	javascript	Semi-stable
real jukebox installed	javascript	Semi-stable
realOne installed	javascript	Semi-stable
screen dpi	javascript	stable
screen resolution	javascript	Semi-stable
pixel depth	javascript	stable
color depth	javascript	Semi-stable
font smoothing	javascript	stable
buffer depth	javascript	stable
update interval	javascript	Semi-stable
shockwave installed	javascript	Semi-stable
silverlight installed	javascript	Semi-stable
silverlight version	javascript	volatile
supports silverlight	javascript	stable
soundcard	javascript	stable
svg feature support	javascript	stable
WAP support		stable
windows media player installed		stable