# jFingerprint Datasheet

## jFingerprint

An introduction to jFingerprint, the browser fingerprinting and identification solution.

**Wolf Software Limited**

## Contents

## 1 Background

jFingerprint is a custom browser fingerprinting and identification solution. The solution gives a site owner the ability to uniquely identify a browser without the need for cookies.

The use of cookies comes with a number of issues:

1. Cookies are easy to delete or modify
2. Cookies require consent and notification when not strictly necessary (PECR/EPD 2011)

Due to the issues above jfingerprint has been developed in order to create a unique browser fingerprint for any given browser. As this is accomplished without the use of cookies there is no legal requirement to notify the user and it does not allow the user to opt out, delete or modify the cookie contents.

## 2 How does it work (overview)?

Jfingerprint works by interrogating the browser for as many details as possible including but not limited to:

- Browser information – name, version, platform, useragent, cookies enabled and java enabled.
- Browser supported items – ajax, boxmodel, cssfloat (and many other supported items).
- Plugin information – names, filename, description and version of all plugins installed.
- Viewport information – width and height.
- Computer related information – Screen width * height, colour depth and pixel depth.
- Operating system information – name and version.

This collection of information is combined into a SHA256 hash which gives you a unique fingerprint for any given browser.

# 3 Technical Overview

The plugin has been built using the latest version of jQuery (however it will support much older versions 1.3+) and make use of many of the inbuilt features of the framework.

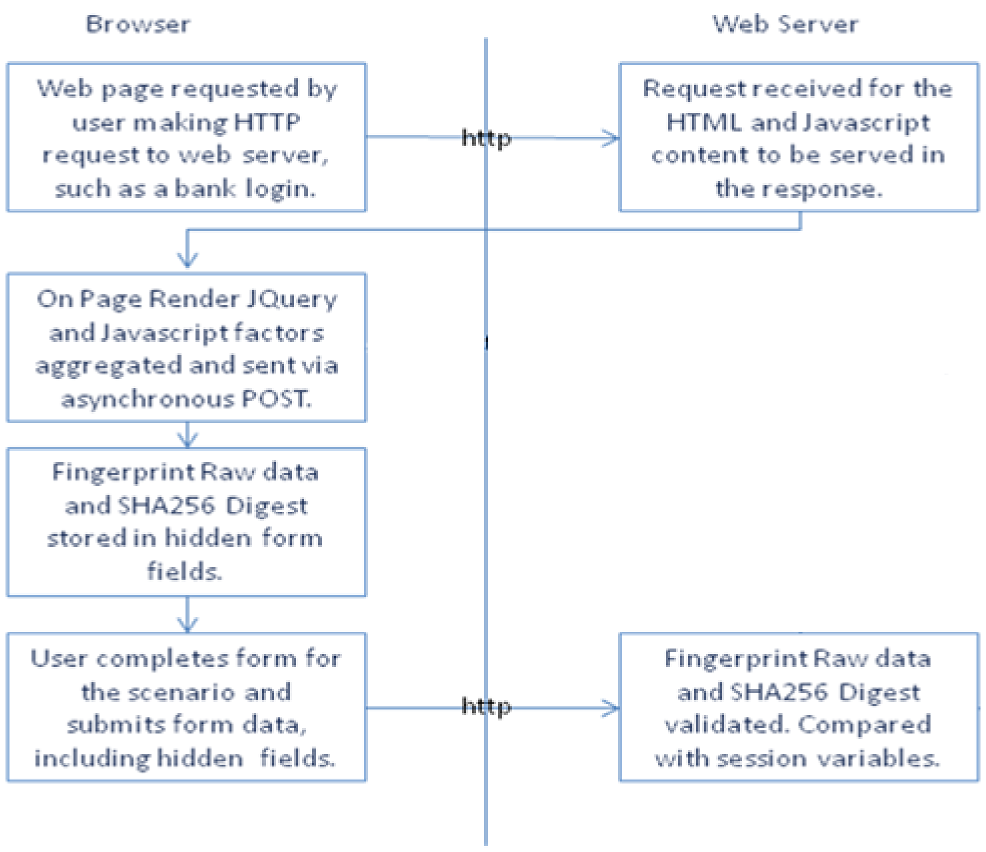
It has a bespoke loader built-in in order to stop It from conflicting with any other frames works that might be in use e.g. prototype or mootools. It will also inject jQuery if it does not already exist, all seamlessly.

The basic methods used for interrogation of the browser are the jquery.support and jquery.broswer sub systems; this allows you to verify what a browser does and does not support and also identify main key features of the browser.

There is also a set of custom functions which are added to interrogate the browser further, the list of categories are detailed below. This is not a complete and exhaustive list of options that can be retrieved, but a list of categories of information that can be retrieved.

## 3.1 Browser / Server Workflow

The following image is the workflow that is carried out in order to create the fingerprint.


Browser
Web Server
Web page requested by user making HTTP request to web server, such as a bank login.
http
Request received for the HTML and Javascript content to be served in the response.
On Page Render JQuery and Javascript factors aggregated and sent via asynchronous POST.
Fingerprint Raw data and SHA256 Digest stored in hidden form fields.
User completes form for the scenario and submits form data, including hidden fields.
http
Fingerprint Raw data and SHA256 Digest validated. Compared with session variables.

## 3.2 Browser Information

The following browser information is available to the plugin:

- Webkit based browser or not
- Browser name
- Browser version
- Cookie support
- Java support
- Javascript support
- Browser plugins (name, filename, description, version)
- User Agent string

## 3.3 Viewport Information

The following information is available to the plugin:

- Window width
- Window height

## 3.4 Computer Information

The following information is available to the plugin:

- Screen width / available height
- Screen height / available height
- Operating system
- Screen colours
- Colour depth
- Pixel depth

## 3.5 Supported Features

The following information is available to the plugin:

- If a browser is able to create an XMLHttpRequest object
- If the page is rendering according to the W3C CSS Box Model
- If a change event bubbles up the DOM tree, as required by the W3C DOM event model
- If a browser correctly clones the checked state of radio buttons or checkboxes in document fragments
- If the value of a checkbox defaults to "on" when no value is specified
- If a browser can create an XMLHttpRequest object and if that XMLHttpRequest object has a withCredentials property

- If the name of the property containing the CSS float value is .cssFloat, as defined in the CSS Spec
- If the .getAttribute() method retrieves the href attribute of elements unchanged, rather than normalizing it to a fully-qualified URL
- If the browser is able to serialize/insert <link> elements using the .innerHTML property of elements
- If the browser inserts content with .innerHTML exactly as provided—specifically, if leading whitespace characters are preserved
- If cloned DOM elements copy over the state of the .checked expando. (Added in jQuery 1.5.1)
- If cloned DOM elements are created without event handlers (that is, if the event handlers on the source element are not cloned)
- If a browser can properly interpret the opacity style property
- If option elements within disabled select elements are not automatically marked as disabled
- If an <option> element that is selected by default has a working selected property
- If inline scripts are automatically evaluated and executed when inserted into the document using standard DOM manipulation methods such as .appendChild() and .createTextNode().
- If inline styles for an element can be accessed through the DOM attribute called style, as required by the DOM Level 2 specification
- If the submit event bubbles up the DOM tree, as required by the W3C DOM event model
- If an empty <table> element can exist without a <tbody> element. According to the HTML specification, this sub-element is optional, so the property should be true in a fully-compliant browser

Any combination of the above can be used in order to create a fingerprint depending on how specific the site owner wants to be. An upgrade to a browser or plugin might result in a new fingerprint where one isn't wanted, so configuration would need to be made on a site by site basis based on the site requirements.

## 3.6 Additional Information

The plugin has the ability to callout to a remote server (script) in order to inspect additional information about a connection. This can be extended to do anything that is required by the site, for example:

- A repository / database of known 'bad' fingerprints can be accessed.
- A check can be performed to identify proxied or tor based connections.

It will also be possible to extend this functionality further using bespoke solutions that are required on a case by case basis.

# 4 Proxy Piercing

Proxy piercing has an interesting "marketing spin" to it, but it simply means acquiring a PC's actual local IP address.

In order to read a PCs local IP address and deliver it reliably back to a server you must execute some native code on the PC e.g. Java applet, a toolbar, or an application which will then send that IP Address back to the servers using a TCP/IP or UDP socket connection, by-passing the HTTP data stream being sent through the proxy server. This is because all IP addresses that are passed within the x-forwarded-for string will be scrubbed (deleted) by the proxy. Alternate methods for transferring an IP address "transparently" might include methods of encrypting the IP Address using JavaScript or ActionScript into some target data field on the PC that will be transmitted within the HTTP stream's user-agent-string. But since a "smart" proxy is going to re-write ALL of the attributes within the user-agent string, your encrypted IP data is simply going to get dropped on the floor, and lost!

It is also important to remember that most proxy piercing solutions will not work reliably on all platforms both PC based and more importantly as time goes by on mobile devices and tablet computers. They generally rely on settings or applications to be installed which can be blocked or removed by the end user rendering it in-effective. Some even take advantage of known security weaknesses in software which are being patched by the vendors again rendering the solution in-effective.

The question therefore becomes: Is knowing the actual PC's IP address ultimately beneficial for fraud management?

If it was possible to reliably acquire the IP Address transparently (and it's not a unroutable IP address) the answer is going to be yes only sometimes, and its usefulness is going to be temporary at best. An IP addresses are not "owned" by a PC. They are not like license plates assigned to a car. IP addresses are extremely temporary, assigned by an ISP for a real-time connection. But they can be re-cycled as frequently as every time the user reboots their router e.g. possibly everyday. When another PC is re-assigned an IP address that is negatively scored, then you will introduce a false-positive potentially blocking a good customer.

Furthermore if a PC is sitting behind NAT (network address translation) firewall then the PC itself will have an unroutable "local" IP Address e.g. in the range of 192.168.xxx.xxx or 10.x.x.x etc., which will basically tell you nothing.

## 5 Demonstration

A demonstration website has been created in order to allow you to view the solution working on a live website.

This can be accessed at http://jfingerprint.demo.wolf-software.com