Implementing a Digital Fingerprinting System

Overview

While adding unique identifier variables to a sample source will help mitigate the risk of respondents repeatedly completing the same survey, these methods are limited to cookie and URL verification. Through geo-location and deeper assignment of unique IDs, the Digital Fingerprinting System aims to better detect duplicate respondents.

How it Works

Rather than checking against survey connection data, the Digital Fingerprinting System identifies characteristic components of each responding computer and assigns it a unique “fingerprint” based on these components. It then checks all entry attempts against its database for that survey and automatically terminates any duplicate fingerprints.

Respondent machines are identified using the following methods:

- A Geo-location check which retrieves the machine’s country code programmatically, as well as its full country name, city, region, postal code, latitude and longitude, area code and metro (DMA) zone.
- Expanded web caching with greater stickiness (i.e., detecting cookie information on a computer even after a user has cleared their cache).
- Machine fingerprinting which retrieves non-personal machine data such as OS version, display settings, fonts, etc., and then applies an algorithm to determine if the machine is unique.

Digital fingerprinting is added to a survey via the XML Editor and is run in the background, meaning that survey IDs can still be captured and the system will not interfere with any additional deduping methods.

Click here for more information on our default respondent verification methods.

Adding Digital Fingerprinting to a Survey

To implement digital fingerprinting in your survey, simply add whatever variables you would like to use from the table below to your survey XML:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forbiddenCountries=&quot;us,gb&quot;</td>
<td>Stops US/UK respondents from taking the survey</td>
</tr>
</tbody>
</table>
allowedCountries="us"  
Allow ONLY US respondents

geoip="all"  
Adds a virtual question showing all GeoIP-related data

geoip="all"  
Note: geoip="all" requires the collection of user IP addresses, and will not be compatible with a default 139+ survey.

fingerprint="all"  
Captures all deep fingerprinting data (such as fp_html5, fp_flash, fp_etag and fp_browser) but does not act upon it

browserDupes="safe"  
Blocks duplicates based on "safe" dedupe methods, like checking against cookies and unique identifiers

browserDupes="strict"  
Includes the experimental browser fingerprint

Geographical Verification

Certain modern browsers support user-controlled Geo-location, where a web page can ask a user whether they want to share their location with it. This generally works on mobile phones, or in certain cases when using a WiFi connection which has been mapped through "wardriving". Our system does NOT use this method.

Decipher's digital fingerprinting uses a GeoIP data file from a commercially-sourced database provided by MaxMind, an IP intelligence and fraud detection firm. MaxMind's GeoLite data file is saved in data/geocity.dat and the same file is shared between all v2 servers. There are no requirements on survey compat level or browser to use this file, and it's also compatible with SECURE surveys.

This system is entirely dependent on the IP addresses of respondents. Users with "proxies" -- typically ISP gateways where all web traffic is funneled -- will show up as being from the location of the proxy. This is on contract to other Geo-location methods (e.g., those used on a cell phone), which require users' permission to share the location but are more precise.

The same might be the case for users with special VPN services or publicly available proxies, while some well known proxies will be automatically recognized and assigned the "a1" country code. Thus, if you are only allowing "us" as country, a user coming through an anonymous proxy will be blocked (even if the proxy located within the US).
Blocking Respondents by Country

There are two methods for blocking respondents by country. You can either specify which countries you'd like to send to, or those you'd like to avoid sending to.

To block respondents by country, add either the allowedCountries or the forbiddenCountries attribute on the <survey> tag, and set it to a comma-separated list of lower-case country codes to either allow (excluding all others) or forbid (allowing all others). Respondents from any excluded or forbidden country, or those from an IP address which cannot be decoded, will receive a general error message controlled by the invited.geoip language resource:

```
# GeoIP violation
*invited.geoip: You are not permitted to take this survey from your location
```

In addition, they will receive one of the following technical error codes:

- SE-20: generated when IP could not be converted to a country
- SE-21 Country is on forbiddenCountries list
- SE-22 Country is NOT on allowedCountries list.

Afterward, the respondent will be unable to continue the survey, just as if they had started it with an invalid list variable. If you'd like to instead terminate the respondent, you can include a <term> tag.

For a full list of country codes, see this document.

Additional Considerations:

- Country codes should only be entered in lower-case (e.g., allowedCountries="us,gb").
- Surveys will fail to load if using an invalid country code.
- The allowedCountries and forbiddenCountries attributes cannot be used simultaneously.
- You are not required to use either attribute to access the GeoIP data; it is also possible to access the data programmatically.
- Users who are logged in and have access to the survey, or surveys in dev/testing mode, will be warned about region violations but allowed to continue.

Tracking GeoIP Data in a Survey

To track respondents' GeoIP data, set geoip="all" on the <survey> tag. This will create a virtual question called vgeoip which will contain a country name, city, region and metro code variable.

```
Note: You must also have the ipAddress attribute enabled to use this setting.
```

Retrieving GeoIP Data in a Live Survey

To retrieve respondents' current IP address data in a live survey, add the geoip object as a condition.

For example:
The `geoip` object has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Example Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>area_code</td>
<td>559</td>
<td>Phone area code (US only)</td>
</tr>
<tr>
<td>city</td>
<td>Parlier</td>
<td>Name of the city (UTF-8 encoded)</td>
</tr>
<tr>
<td>country_code</td>
<td>us</td>
<td>ISO 3166 country code (lower case)</td>
</tr>
<tr>
<td>country_code3</td>
<td>USA</td>
<td>ISO 3166-1 alpha-3 country codes (upper case)</td>
</tr>
<tr>
<td>country_name</td>
<td>United States</td>
<td>Official country name</td>
</tr>
<tr>
<td>dma_code</td>
<td>886</td>
<td>Legacy field. Use <code>metro_code</code></td>
</tr>
<tr>
<td>latitude</td>
<td>36.6265983582</td>
<td>North-South coordinate</td>
</tr>
<tr>
<td>longitude</td>
<td>-119.51940155</td>
<td>East-West coordinate</td>
</tr>
<tr>
<td>metro_code</td>
<td>866</td>
<td>Metro code. List of metro codes (US only)</td>
</tr>
<tr>
<td>postal_code</td>
<td>93648</td>
<td>Zip or postal code (US/Canada)</td>
</tr>
<tr>
<td>region</td>
<td>CA</td>
<td>ISO 3166-2 state code or FIPS 10-4 code</td>
</tr>
<tr>
<td>region_name</td>
<td>California</td>
<td>State name</td>
</tr>
<tr>
<td>time_zone</td>
<td>America/Los_Angeles</td>
<td>Time zone</td>
</tr>
</tbody>
</table>
Note: The country_name and code attributes have 99.5% accuracy (99.8% if using the premium commercial version of the database). All other fields are 83% correct within a 25-mile radius.

Retrieving GeoIP Data in a Virtual Question

It is also possible to retrieve respondents' GeoIP data within a virtual question. To do this, simply add the geoip object directly to the virtual question you'd like to use for tracking.

Note: In virtual questions, geoip will use the ipAddress extra variable, which must have already been configured (any survey using <samplesource> will have it).

Deep Fingerprinting

While normal fingerprinting is based on unique variables passed in the URL (e.g. ?source=1234) or browser cookies set on recent surveys (controlled by browserDupes not being set to a blank value), there are four additional IDs you can use for deeper fingerprinting:

- an "HTML5" cookie (a more modern browser storage method than old style cookies)
- a cookie stored inside Flash Local Storage
- a value stored within the browser cache using the “ETag” method
- a “browser fingerprint” derived from installed fonts, browser version, plugins and screen resolution

These additional IDs can be captured and processed in either the survey or the report.

Enabling Deep Fingerprinting

To enable deeper fingerprinting and capture all four of the additional IDs above, set fingerprint="all". This will create four new extra variables in the data set: fp_html5, fp_etag, fp_flash and fp_browser.

You can examine these fields within the survey itself or in virtuals and terminate data accordingly. The existing cookie variable is stored in the session extra variable.

Note: Deep fingerprinting requires compat="109".

The ID capturing mechanism runs in the browser, on the first survey page loaded. Thus, those variables will be blank on the first page, and only have a value on the second and subsequent pages. A value may remain blank if detection failed to run. It's conceivable that on extremely slow machines or machines with poor network connection the user could submit the first page before the detection finished to run. No attempt is currently made to prevent that, such as disabling the Continue button.

The HTML5 Method

The current dupe detection method is based on cookie. The first time our system encounters a respondent, it sets a cookie for them by sending a HTTP Header named Set-Cookie. The browser will then remember the unique value stored there for 30 days and send it back every time it visits any survey on the same domain (e.g., v2.decipherinc.com
and survey.ebay.com have different cookies). In a survey, this value can be retrieved by examining the session extra variable.

The HTML5 method stores a unique ID within the “HTML5 Local Storage” system. In Google Chrome, you can review local storage for a domain by opening Tools -> Developer Tools -> Resources then selecting Local Storage. There, you will see a beacon_id with the value.

Note: This method is compatible with the latest versions of most modern browsers.

The unique HTML5 ID defaults to the "session" ID as used for cookies. If in captured data the session is different than fp_html5, you’ll know the user has been here before and either one or the other has been lost or cleared. The ID will always be a string of 16 random characters (e.g., "5d36v2kz0jpbkv3s").

The ETag Method

The “ETag” method uses a browser’s cache to store a unique ID tied to a specific external page. The server will essentially ask the browser to cache the page together with the unique ID, which again defaults to the session ID.

Note: The name has been chosen to conceal from the user the purpose of this file.

When the user returns, that cached ID can be retrieved, even if the user has cleared their cookies -- as long as the user has not cleared the cache. It will be available in the fp_etag variable.

The ETag method has the added advantage of being cross-domain. Because the shared data is available to all web pages, a user taking a survey at survey.paypal.com and later at survey.ebay.com can be given the same ETag ID. The cookie, HTML5 and Flash methods are only able to store a user's unique ID for at particular domain.

To examine your “ETag” cookie, open about:cache in Google Chrome, and search for appversion.js. You will see the page as saved by Chrome, including the ETag field.

The Flash Method

Adobe Flash has its own cookie-like system. The "Flash" method takes advantage of that by creating an invisible Flash object. This small Flash file tries to retrieve a stored value. If unable to retrieve it, it will instead set it to default to the current session ID.

Once it has that value, it will communicate with the browser page and save the ID for submission to the server. It will be available as fp_flash.

You can view such stored data for all domains here.

The Browser Fingerprint

This method uses Flash and browser-specific Javascript to capture a number of accessible system configurations that may identify duplicate respondents, even if all cookies and cache is deleted.
The following settings contribute to the browser fingerprint, stored in the `fp_browser` extra variable:

- List of fonts on the user’s system (retrieved via a Flash object)
- Screen resolution and depth (e.g. 1920 * 1200 pixels, 32-bit color)
- List of plugins on the system and their versions (e.g. Adobe Flash, Acrobat and Windows Media Player versions)
- Browser version
- Other browser specific information that may identify additional plugins

The `fp_browser` ID is expected to be unique, but relatively short lived. The ID may not be the same for some users who have a faster 6-week cycle on browser versions (e.g., Chrome/Firefox), but it should catch a user that clears all their cookies and cache (including Flash cookies) and returns within a few days to the same survey.

The full ID is composed of 4 sections:

- A fingerprint of installed plugins (e.g., MG0IS2KpaX9X+S9oPqvzrg)
- A fingerprint of installed fonts (e.g., 9SujGFofFjpW7TFxIlpe8w)
- The screen resolution (e.g., 1920,1200,24)
- Browser version and other browser settings that are not volatile (e.g., 3TrMt90DPNgX)

Note: Only the full ID is checked against the dupe database, and it’s expected that only the full ID can provide enough entropy to avoid false positives; however, you can extract parts of the `fp_browser` variable to use as you wish.

Deep Deduping

By default, it is possible to enable browser deduping by adding the `browserDupes` attribute to the `<survey>` tag:

- `default` -- if not specified, this means the same as `browserDupes="cookie"` for compat 8+ surveys and no dupe checking for older surveys
- `blank` -- i.e., if `browserDupes=""` is set in the `survey.xml`, then no dupe checking is done

To enable deeper deduping, you can set this to one of the following values:

- `safe` -- this enables deduping based on cookies as well as the HTML5, ETag and Flash methods below. These are safe methods to always enabled, as there’s no chance of false positives
- `strict` -- this also dedupes on the browser fingerprint. This is currently experimental, and precision is not yet known.

Additional Considerations:

- The safe/strict methods can only work after the user has loaded the first page into his browser and submitted it, giving Flash time to collect information. This means that a user may be blocked on the second page after pressing Continue once and this will look like a dropout.
- Users caught by safe/strict browser dupes are not terminated; they will simply be unable to continue. To instead terminate and redirect users, add a `<term>` tag.
- Users who are logged in and have access to the survey in dev/testing mode will be warned about dupes, but allowed to continue.
• Legacy browser may not be able to use all available detection methods:
  ◦ Cookies: all supported
  ◦ HTML5: requires a modern browser (IE9, Firefox, Chrome).
  ◦ ETag: no special requirements, all normal browsers supported
  ◦ Flash: requires Flash 9.0+ (released in 2006)
  ◦ Browser: requires Flash 9.0+ for font detection
• Mobile browsers: Opera Mini will support only the “ETag” method.
• In case the detection mechanism is unable to use a method, it will still continue to try the other methods. This is not expected present the respondent with any error messages.

---

### Styles and Other Resources

The GeoIP system uses a single language resource, with an error message shown when a violation occurs (e.g., forbiddenCountries or allowedCountries requirements were not met):

```plaintext
# GeoIP violation
*invited.geoip: You are not permitted to take this survey from your location
```

If browserDupes="strict" or "safe" is enabled, and any of the 4 or 3 respectively captured IDs have completed the survey before, then the standard dupe style is displayed:

```plaintext
# Already completed
*survey.invited.used: [survey.static-error;title=@(invited.used),error=@(invited.used)]
```

This uses the invited.used language resources:

```plaintext
# Shown when someone has already gotten through the survey
*invited.used: It seems you have already finished this survey.
```

To detect respondents, the following styles are emitted on the first page of a survey:

```plaintext
# Generate a unique ID via ETag cache
*survey.fingerprint.etag:
<script src="$(host)/page/appversion.js"></script>
<input type="hidden" name="__fp_etag" id="__app_version" value="">
```

The browser fingerprint detection is more complex and uses a [third party script](https://support.focusvision.com/Decipher/0003_Build/Advanced_Programming/006_Putting_it_all_Together/004_Setting_up_Sar) to enumerate plugins in Internet Explorer:

```plaintext
*survey.fingerprint.browser:
# Generated from http://www.pinlady.net/PluginDetect/PluginDetect%20Generator.htm
<script src="[versioned /s/support/plugindetect.js]"></script>
<script>
 $(Survey.detectPlugins);
</script>
<input type="hidden" name="__fp_plugins" id="__fp_plugins" value="">
<input type="hidden" name="__fp_screen" id="__fp_screen" value="">
```
The Flash one uses `swfobject.js` to embed a very small SWF file:

```html
*survey.fingerprint.flash:
@if includeOnce('swfobject22.js')
<script src="[versioned /s/iq/swfobject22.js]"></script>
@endif
<div id="fp_flash_container"></div>
<script>
 $( function() {
   swfobject.embedSWF("/s/flex/fp.swf", "fp_flash_container", "0", "0", "9.0.0", false,
   {session_id: $(gv.request.session|js)});
});
</script>
<input type="hidden" name="__fp_flash" id="__fp_flash" value="">
<input type="hidden" name="__fp_font" id="__fp_font" value="">
```

Note: The SWF must not be referenced using `[versioned]` to avoid domain restrictions if it were to be served from `static.decipherinc.com`.

https://support.focusvision.com/Decipher/0003_Build/Advanced_Programming/006_Putting_it_all_Together/004_Setting_up_Sar
Updated: Thu, 10 Jan 2019 20:09:13 GMT
Powered by