Association for Information Systems
AIS Electronic Library (AISeL)

CONF-IRM 2016 Proceedings

International Conference on Information Resources Management (CONF-IRM)

2016

A framework for mobile handset detection

Neil Croft
University of Pretoria, ringtingting@gmail.com

5. A framework for mobile handset detection

Neil Croft
University of Pretoria
ringtingting@gmail.com

Abstract

Due to the increase in social media and mobile media in general, access to these platforms from a number of different mobile devices must be catered for. Mobile Device Detection (MDD) refers to software that identifies the type of mobile device visiting the mobile web and either redirects the end user to a dedicated mobile web site or adapts the rendered output from a web server to best suit the capabilities of the end user's mobile device. Furthermore, handset credentials are, in some cases, used for the purpose of correctly identifying the mobile operating system, which in turn is used to redirect to a mobile application download.

In any web server request, identification of the user is transmitted in a header field known as the User-Agent (UA). Identifiable information present in the request header allows for unique browser identification and the device used in making the request for a web page. A lookup table, comprising of the all known handset capabilities, is the core functionality of a MDD.

Our aim in this paper is to survey the distribution of mobile User-Agents so as to establish an ***attribution*** of mobile browsing detections. In particular, assessing details of mobile User-Agents, proxy requests (intermediary for requests from users seeking resources from other servers), emulated requests (software program that imitates a real handset), perpetuates our ability to census mobile traffic with some degree of accuracy. The approach is to make use of a sample set of real-world mobile aggregated requests. We will analyze and filter these requests by origin, User-Agent, geo-location and sometimes non-industry standard (inserted by mobile network operators, which might contain personally identifiable information) to build our mobile browsing framework. In doing so, we encompass description, identification, nomenclature, and classification of end user mobile handset detections. Finally, we will investigate the significance of our framework to see how unique requests are given nothing other than identifiable browser information.

Keywords
Mobile, handset detection, WURFL, User-Agent, framework.

1. Introduction

With the growing number of mobile Internet-enabled devices in use, providing a usable mobile website or application is more important than ever; but you cannot achieve this without mobile device detection software. This is mainly due to the Device Fragmentation (Wong 2012, Shevchik 2013) or Device Diversity problem that exists today. There are a seemingly infinite number of mobile device configurations in the market today, wreaking havoc on developers and marketers alike. New handsets are released almost daily into new and existing markets with the same or similar configurations. Handset manufacturers and vendors alike rush to launch new handset into

the market before their competitors do, exclusive deals are sometimes signed with network operators further adding to the device fragmentation problem.

The driving force behind device fragmentation is due to users choosing different devices with different functionalities for different reasons (price, phone bundle/basket etc.). So, if there is a way to use these dissimilarities (and sometimes similarities) to partition an audience and design a mobile offering for the various use-segments, then ultimately a better and more relevant experience is provided to the device user. Taking advantage of the differences is where device fragmentation transforms into market segmentation and could spell greater success for a mobile marketing campaign. Providers of mobile device detection software are continually updating their Device Description Repository (DDR) as the relevance, accuracy, interoperability, availability and speed of handset detection is paramount. The Device Description Repository (DDR) is a concept proposed by the Mobile Web Initiative Device Description Working Group (DDWG) of the World Wide Web Consortium (Smith et al. 2007).

The DDR is supported by a standard interface and a core vocabulary and data structure of handset capabilities. Implementations of a DDR are expected to contain information about both mobile devices and web browsers (whether pre-installed or downloaded). If the DDR is not continually updated (with accurate information), the percentage of correct identification is significantly reduced, in which case a number of inaccurate assumptions are made (assuming identity based on a ``similar'' device from the same manufacturer). Providers of mobile device detection software as a service; capture and log all requests, for each device detection. This aids in maintaining a current device repository through the identification of any unidentifiable handsets detected.

Just how much information can we extract from these requests? The information needed to perform such an analysis resides in each and every web request header received at a web server, and, later directed to the mobile device detection software. In saving, reading and matching header information to known devices, the service provider is able to accurately track and monitor trends in mobile activity, make predictions on future mobile trends, capture and analyse statistics and ultimately ascertain user mobile traffic. The result of the mobile traffic analysis contributes to the creation of a framework for mobile detections. The significance of which is demonstrated in the following example. Applications ***must*** detect fraud; however fraud detection tools available today that work in fixed-line computing environments do not necessarily translate to the mobile world. There are a number of methods that can be implemented to help detect fraud in the mobile environment, however these are still in the early stages of development and do not provide transparently across disparate mobile networks. If we begin with an analysis of extractable mobile request information we may only then begin to think of building framework for mobile detection, which may aid in fraud prevention.

The aim of this paper is to focus on the cardinality and distribution of mobile User-Agents. In doing so, we are able to account for identifiable information present in web request headers. We consider the possibilities of proxy traffic, emulated traffic, and non-standard network-generated headers. This extracted information aids in the design of a framework for mobile handset detections. The methodology used is as follows: first we need a truly representative sample set from a distributed network capturing web request headers. Secondly we need to accurately extract and match handset information, interpret and comment on the results found. In doing so, we tend

towards (if our sample set is large enough) an accurate representation of the mobile web, based on a breakdown and analysis of distributed mobile requests. Statistical analysis is easier to extract in a traditional web-based environment. However, with the extent of mobile browsing and the need for accurately proving a specific device performed a specific web browsing action, we focus only on mobile browser requests.

This paper is structured as follows: Section 2 provides a background on the technologies and methods used in device identification with a focus on Wireless Uniform Resource Locator (WURFL). Our intension in Section 3 is to identify mobile devices by their User-Agents with exact matches only and exclude bot traffic (automated traffic) in order to provide an accurate reflection. In each case example mobile request headers are given. The significance of the results is used to create the proposed framework which is discussed in detail in Section 4. Possible uses of the framework include design principles in Next Generation Network (NGN) mobile security, privacy and forensic frameworks. Section 5 concludes this paper.

2. Background

Unlike traditional desktop web browsing, there is a tremendous amount of fragmentation when it comes to mobile devices and their browsing. Markup can exist in any of the following standards: WML, HTML, HDML, cHTML, XHTML Mobile Profile, etc. (Saha et al. 2001. In comparison to a standard desktop web browser, a handset and its browsing capabilities will vary according to screen size, ability to support client side scripting, ability to support various image formats, etc. The response to a mobile browsing request is markup sent directly to the handset leaving no opportunity for the web server to adapt to browser limitations. This is compounded with the fact that software updates for mobile browsers are rare (Wurfl 2014) as mobile browsing software is pre-installed.

In this section we cover User-Agents (UA), User-Agent Profiles (UAProf). We also investigate how WURFL uses and extracts device capabilities from these request headers and even serves them as back as part of the response (Passani 2012). Knowing what mobile request information is available is vital for the attribution (practice of attributing information to its source) of mobile browsing detection. In our case, attribution to the source is a requirement due to the construction of our framework.

2.1 User-Agents

When a user request is made to a publicly accessible web server, it often identifies itself, its application type, operating system, software vendor and revision, by submitting a characteristic identification string, also known technically as a User-Agent string. The User-Agent string format is currently specified by Section 14.43 of RFC 2616 (HTTP/1.1) (Fielding et al. 1999). The format of the User-Agent string in the HTTP protocol is a list of keywords with optional comments. For example if a handset manufacturer created a new Mobile Browser and call it ``MobileBrowser'', the User-Agent string might look as follows: MobileBrowser/1.0 Gecko/1.0, where ``MobileBrowser/1.0'' indicates the name and revision and ``Gecko/1.0'' indicates the layout engine and revision. The User-Agent string is often used by web crawlers, also known as bots, for identification purposes to the webserver it communicates with. From a web server standpoint, this

traffic may be excluded from accessing certain parts of a web site using the Robots Exclusion Standard (Sun et al. 2007). This information usually resides in the root directory of the mobile site in a file called robots.txt and defines the handling of automated traffic. This aids in the identification of "real" traffic. At times it has been popular among mobile web developers to initiate User-Agent spoofing where the User-Agent string is manipulated on each web request made to a web server. Usually this is done during the testing phase of program development in order to test a programs outcome with various mobile device emulations. An example of a User-Agent switcher is found in the Firefox browser as an add-on module (Di Stefano et al. 2002) which allows for User-Agent manipulation for the purpose of emulation. We do consider that as a result of User-Agent spoofing, collected statistics of mobile web browser usage may be inaccurate. However, due to the low number of such requests being initiated (and identified) we assume this to be negligible variable in the results.

2.2 User-Agent Profile

The UAProf (User-Agent Profile) specification is concerned with capturing capability and preference information for wireless devices (Quiroga et al. 2011). This information can be used by content providers to produce content in an appropriate format for the specific handset. This information is in the form of an XML document and usually covers the following device attributes: Hardware characteristics (screen size, multimedia capability, etc.); Software characteristics (operating system, etc.); Network characteristics (GSM/GPRS/3G/LTE capable); Browser characteristics (name, version, script support).

Since it is not feasible for a handset to send all the information with each web request, the profiles are stored in publicly accessible repositories, thus a referral via a URL provides access to the UAProf for a particular handset. Theoretically, applications should be able to retrieve this profile programmatically and dynamically re-purpose the content for each device. However, practically, User-Agent Profiles (Quiroga et al. 2011) alone are not sufficient for handset detection. Some of the reasons include: UAProf profiles aren't maintained and are often wrong, legacy handsets devices do not have a UAProf, UAProf are sometimes hosted at private URLs (only applications in the domain of a given operator have access to them), UAProf may not be legally re-host at a different URL and finally, companies such as Google and Microsoft decided to ignore UAProf on many devices. Thus a User-Agent string is the only reliable part of the web request header usable in handset identification.

3. Wireless Uniform Resource Locator

WURFL (Wireless Universal Resource FiLe), originally an open source effort, focused on mobile device detection. It quickly became the de facto standard open source (FOSS) framework for addressing Device Fragmentation or Device Diversity in mobile handsets. WURFL originally an XML configuration file (flat file) has been ported to many databases. It simply contains information about device capabilities and features for a variety of mobile devices (Hatem 2013). WURFL, a Device Description Repository (DDR), has a software component (written in just about every programming language) that maps web request headers to the profile of the user device (desktop, mobile handset, tablet, and recently even the smart TV). WURFL uses the User-Agent string passed along with each web request to a web server. WURFL constitutes a set of proprietary

application programming interfaces (APIs), which match and retrieve capabilities directly from the DDR file. WURFL may reside as a standalone program or sit in the cloud. Wherever the WURFL file is hosted it must be updated with the latest handsets and the over 500 device capabilities for each handset. A portion of WURFL is dedicated to achieving accuracy versus speed using a combination of caching techniques and repository optimisation.

A combination of various capabilities (for example is_wireless_device, brand_name, model_name, mobile_browser, resolution_width, device_os) may be used by an application in a number of combinations. For example, a mobile advertising platform might need to know a handset's resolution width and touch screen capability in order to render the best suited advert back to a requesting handset. As a result of the continued growth of the WURFL file (new device information being added daily) a combination of novel techniques around caching, fall back device (if no accurate match is found) and even new browser inclusion (such as Opera Mini, Firefox mobile and Chrome mobile) have evolved. The extent to which the WURFL file has changed over the years is largely due to the introduction of these new mobile browsers and the complexities which arise from mobile network proxy traffic. For example, BlackBerry devices and Apple IPhone mobile traffic is proxy via the UK and USA respectively. WURFL has remained at the forefront of mobile detection and has evolved into an enterprise offering including products for mobile analytics, high-volume scalable detections.

4. Framework for mobile handset detection

Before beginning with any mobile data analysis and or comparisons of mobile user detections, we first need to understand our data environment and establish and define our sample data set. We begin by investigating the storage of the web browser request headers.

4.1 The Data Environment

The Data Environment includes a redundant network configuration whereby each and every web request header is saved to a flat file database. This aggregated data is generated across a distribution network and sources of the data include: proxies, web servers, mobile ad serving engines etc. The database used, in this case, is MongoDB (Chodorow 2013), an open source document-oriented NoSQL database system. MongoDB is a scalable, high-performance database which is part of the NoSQL family of database systems meaning that instead of storing data in tables as is done in a relational database, MongoDB stores structured data as documents with dynamic schemas. Using a NoSQL database makes the integration of and access to web browsing header information easier and faster than traditional relational databases.

The web browser header data has been collected for a period of one year. Although new handset data is continually added to WURFL XML file, our aim is not to provide a detailed breakdown on an individual handset level but rather identify end-user mobile detection characteristics and ultimately the ***attribution*** of this information. Given the sample set is collected over a significant period of time, from various sources, the data tends towards a complete representation of all possible types of web and mobile browser requests. Due to the sensitivity of this data it was decided not to concentrate on the details of the data set but rather the interpretation of the distribution. It is

important to note, however, that a sample set size of 4 GB of web request headers was processed. It is the view of the author that his is a significantly large enough sample set size for the purpose of the research.

4.2 High performance embedded handset detection in a web server

Developers can retrieve a list of device capabilities by accessing a Device Description Repository (DDR) using a number of programming languages, including java, .NET, Perl, node.js etc. A number of solutions are available to a developer include: a cloud API (application program interface), a standalone program interface and an embedded C++ module plug-in for a number of web servers (including Apache and Nginx). Each implementation has its own benefits. For example, the cloud API uses the latest DDR but lacks in performance due to network lookup (Internet) latency issues. The standalone option suffers from a manual intervention by a developer in retrieving the latest DDR (a zipped archive) and installing it, however, it has significant performance improvements over the cloud solution. Lastly, the embedded web server plug-in is designed for high performance detections. Caching, taking common handsets and pre-loading their capabilities into the server's memory, further enhances lookup speeds.

For our tests, we deployed an Ubuntu 12.04 LTS server with an embedded WURFL module using an Nginx web server. To demonstrate this, a simple PHP script (phpinfo()) responds as part of the web browser header the detected handset (from the User-Agent string interpretation and lookup). For example a request to a test PHP page with the User-Agent string:

``BlackBerry8300/4.2.2Profile/MIDP-2.0Configuration/CLDC-1.1 VendorID/107UP.Link/6.2.3.15.0"

produced the response shown in Figure 1. Notice the PHP Environment variables (returned from Nginx) includes handset information, for example: SERVER[WURFL_CAP_BRAND_NAME]

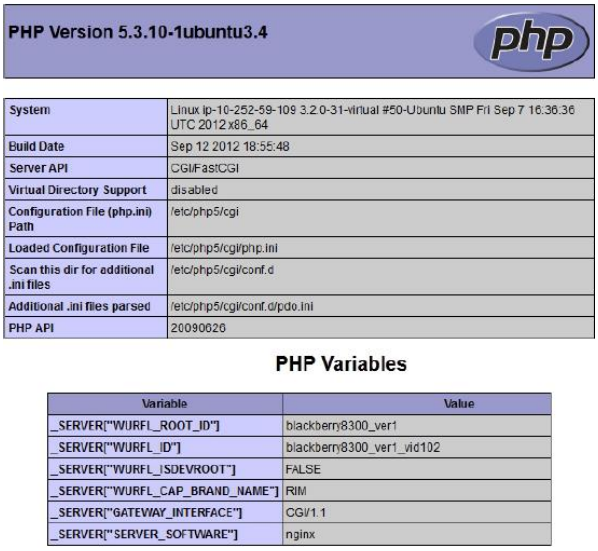

Figure 1: PHP phpinfo() test page

We ran our tests, taking MongoDB web request headers, running them against our high-performance handset detection web server, to extract as much information as possible from the data set. Our main objective: to check the cardinality and distribution of mobile user agents. In

doing so, taking the results we are then able to work towards creating a framework for the attribution of mobile browsing detection. Listing 1 shows a basic example web browser request to a web server made from a BlackBerry handset:

```
GET /dumprequest.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Referer: http://www.google.com/
User-Agent: Mozilla/5.0 (BlackBerry; U; BlackBerry 9700; pt) AppleWebKit/534.8+ (KHTML, like Gecko)
Version/6.0.0.546 Mobile Safari/534.8+
Host: domain.com
Connection: Keep-Alive
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1, *, utf-8
...
```

Listing 1: PHP dumprequest.php page

We were able to categorise the request information as follows:

4.2.1 Handset identification

Using WURFL, we may extract the following information from the request as there is an exact match on the device lookup. A subsequent result may include (amongst other capabilities) the Manufacturer: RIM, Phone Model: BlackBerry 9700, Marketing Name: Bold, Mobile OS: RIM OS 6.0 and Screen Resolution: 480x360 resolution.

4.2.2 IP Address Identification

We may also use geo-location as a means to determine if mobile traffic originated at an IP address associated with a network service provider. We may use one or a combination of the following web browsing headers to extract the IP address from the request.

- HTTP_CLIENT_ IP
- HTTP_X_FORWARDED_FOR
- HTTP_X_FORWARDED
- HTTP_X_CLUSTER_CLIENT_IP
- HTTP_FORWARDED_FOR
- HTTP_FORWARDED
- REMOTE_ADDR

We may map a list of known network proxy traffic to the IP address of the request and may conclude that any originating IP address, other than that present on the list, is emulated in nature.

Listing 2 is a PHP code snippet for IP address retrieval from the HTTP header request:

```
function get_ip_address_from_http_header ()
{
  foreach ( array (` HTTP_CLIENT_IP ', `
                    HTTP_X_FORWARDED_FOR ', `
                    HTTP_X_FORWARDED ', `
                    HTTP_X_CLUSTER_CLIENT_IP ', `
                    HTTP_FORWARDED_FOR ', `
                    HTTP_FORWARDED ', `REMOTE_ADDR ')
  as $key ) {
  if ( array_key_exists ($key , $_SERVER ) === true ) {
    foreach ( explode (',', $_SERVER [$key ]) as $ip) {
      $ip = trim ($ip); // to be safe
      if ( filter_var ($ip , FILTER_VALIDATE_IP ,
                          FILTER_FLAG_IPV4 |
                          FILTER_FLAG_NO_PRIV_RANGE |
                          FILTER_FLAG_NO_RES_RANGE) !== false ) {
        return $ip;
      }
    }
  }
  }
}
```

Listing 2: get_ip_address_from_http_header function

4.2.3 Proxy Server Identification

Opera Mini is a web browser designed primarily for mobile phones. Opera Mini is offered free of charge, supported mainly through deals with mobile operators to have Opera Mini pre-installed on their phones, and other sources of revenue such as search advertising deals, licensing and paid bookmarks. Opera Mini's is designed such that it requests web pages through its own Opera Mini servers, which process and compress them before sending them to the mobile phone, speeding up transfer by two to three times and dramatically reducing the amount of data transferred. This benefit is passed onto the user through data charge savings. The pre-processing increases compatibility with web pages not designed for mobile phones.

As Opera Mini proxies all its traffic through its own servers, the originating web browser request is altered by the serving proxy. In most cases, the User-Agent string is altered as the request now originates from Opera Mini. An example of the User-Agent might now be: Opera/9.80 (J2ME/MIDP; Opera Mini/9.80 (J2ME/22.478; U; en) Presto/2.5.25 Version/10.54. In other words, if we were now to make use of our DDR, the result returned would now include the following information: Mobile Browser: Opera Mini 1, Screen Resolution: 176x160 resolution. Proxies who alter the User-Agent string may include (as is the case with Opera Mini) a non-standard web browser header which retains the originating User-Agent before the request reached the Proxy Server. Our web browser header might contain the following header: X-OperaMini-Phone-Ua. We are able to use this header for our handset detection while knowing the web browser request was proxied via Opera Mini.

4.2.4 MSISDN Identification

An Access Point Name (APN) is the name of a gateway between a GPRS (or 3G, LTE etc.) (Dahlman et al. 2010) mobile network and another computer network, in most cases the public

Internet. A mobile device making a data connection must be configured with an APN to present to the network carrier. The carrier will then examine this identifier to determine what type of network connection should be created, for example: what IP addresses should be assigned to the wireless device, what security methods should be used, etc. A network carrier APN, in some instances, is configurable to pass through the mobile number (MSISDN) of the subscriber included in the web request. In doing so, the network carrier may violate the privacy of the subscriber; however this information might be the only truly uniquely identifiable information available, linking a subscriber to a mobile browser request. In some cases (APN dependent), the local Vodafone/Vodacom network carrier passed the MSISDN through as part of the web browser headers as follows:

- HTTP_X_UP_CALLING_LINE_ID
- HTTP_ X_UP_VODACOMGW_SUBID
- HTTP_X_MSISDN

Listing 3 shows the PHP code snippet for the MSISDN extraction from the web browser header, given the presence of the non-standard web browser headers above.

```
function get_msisdn_from_header (){
  $msisdn = `';// Vodafone
  $msisdn = $_SERVER [`HTTP_X_UP_CALLING_LINE_ID '];
  if ( $msisdn == `") { // Vodafone
    $msisdn = $_SERVER [`HTTP_X_UP_VODACOMGW_SUBID '];
  }
  if ( $msisdn == `') {
    $msisdn = $_SERVER [` HTTP_X_MSISDN'];
  }
  return $msisdn ;
}
```

Listing 3: get_msisdn_from_header function

As it may not always be possible to get a mobile device id number from its browser request, a possible solution is to generate a ``unique'' number (in our case, with php, we used the session_id() number mixed with a number generated with the rand() function) and then store it in a cookie that never expires (refer to Listing 4}. This is not a perfect solution, as it will not work if cookies are disabled or if the user clears his cookies, but mobile device mostly have cookies enabled, and the users don't seem to clear them often.

```
Cookie:
ci_session=a%3A4%3A%7Bs%3A10%3A%22session_id%22%3Bs%3A32%3A%2215ed7fdc01fa9ca778c48159d5be80ad
%22%3Bs%3A10%3A%22ip_address%22%3Bs%3A15%3A%22105.228.178.128%22%3Bs%3A10%3A%22user_agent%
22%3Bs%3A50%3A%22Mozilla%2F5.0+%28Linux%3B+Android+4.2.2%3B+en-
za%3B+SAMSUNG+%22%3Bs%3A13%3A%22last_activity%22%3Bs%3A10%3A%221403338596%22%3B%7D49d6b8
a54c5ef54ff321e1786db34215; PHPSESSID=c00tqlhhv2jr6t5u141jutklr0;
_pk_id.3.ba30=accd5838528bd401.1403338973.1.1403339119.1403338973.; _pk_ses.3.ba30=*; __atuvc=3%7C25
```

Listing 4: Mobile Cookie Session Example

4.3 Why a handset detection framework?

Why are we concerned with mobile identification and building a framework? It is evident from such reports (Goredema 2012) that South Africa's increased use of mobile phones, access to social media and Internet contribute to an increase in cybercrime. The scope of activities which could fall within the definition of mobile cybercrime is potentially broad, ranging from purely malicious or intimidation invasions of privacy, to the theft and abuse of personal identity particulars and the fraudulent manipulation of electronic data to commit fraud. If we do not consider to survey and structure a framework, we cannot begin to design a framework to combat fraud for example.

In the previous section we spent some time performing a detailed extraction of web browsing request data. We detailed that on top of the web request lies the following: IP address extraction (proving a request was proxied through a legitimate network access point or third party), Proxy extraction (given the fact that some requests are proxied through third parties), Handset extraction (using a DDR to determine a generic device, partial find or fully detect) and then finally finding non-standard web browsing headers with information uniquely identifying the subscriber (e.g. MSISDN). The categorisation of the data lies in the presence of all of these identified components of a mobile request. Figure 2 illustrates our proposed handset detection taxonomy. The attribution extraction along with the network information is used by the framework in implementation of the objective or specific case. An example of the objective is provided in Section 4.3.1.

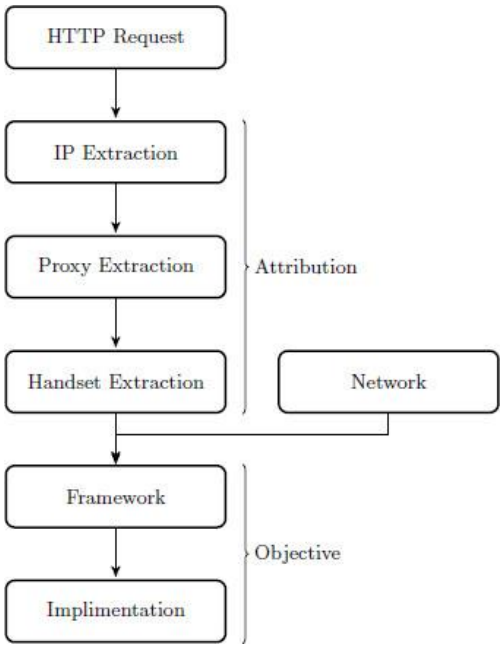

Figure 2 Proposed Mobile Detection Taxonomy

4.3.1 Framework analysis – a case example

Using our framework, consider a crime is committed using a mobile device via a mobile browser. As an example, suppose a fraudulent insurance claim is made using this mobile device. In doing so, the perpetrator has left his browsing activity in the insurer's web browsing server log files.

Furthermore, suppose the evidence is typed with standard and non-standard (some of which is uniquely identifiable) web browser header details. We consult the mobile detection taxonomy in order to prove the attribution of such an event's occurrence. In this case, it may be common to consider the match (conditional) probability $P(E = ab|S = ab)$ under the hypothesis of innocence, where $E = ab$ and $S = ab$ means the evidence and the suspect have non-identifiable (or even partially identifiable) information, represented by a with identifiable information, represented by b provided in the web browser header(s). If we are to consider the product rule as a means to evaluate the match probability of a mobile user's mobile browsing profile then the above match probability is evaluated as follows:

$$P(ab|ab) = 2p_a p_b; a \neq b$$

$$P(aa|aa) = p_a^2$$

where p_a and p_b are receptively the proportions of handsets browsing with web browser request header information ab in the defined population of requests. The population is determined from the statistical analysis of all browser requests, excluding bot traffic, of all mobile browser requests for a particular service. We may introduce a parameter to measure the uncertainty or likelihood about the proportions in the suspect's population; however this is left to future work.

5. Conclusions

In this paper, we assess details of web browsing requests in order to identify survey mobile traffic with some degree of accuracy. We made use of a set of real-world mobile aggregated requests analysing these using a DDR. We identified information that was identifiable and non-identifiable which lead to the building of our proposed mobile detection taxonomy. The taxonomy's categorisation of the data lies in the presence of IP address extraction, proxy extraction, Handset extraction all of which forms part of attribution extraction. The analysis thereof made use of match (conditional) probability in determining the likelihood the evidence and the having both non-identifiable (or even partially identifiable) information together with identifiable information (of the subscriber). This paper shows the possibility of reaching a state of attribution of mobile handset detections forming the basis for handset detection frameworks.

Acknowledgements

The author would like to thank Steve Kamerman from ScientiaMobile Inc for his assistance with regards to key aspects of mobile detection.

References

Chodorow, K. (2013) "MongoDB: the denitive guide" O'Reilly Media, Inc.

Dahlman, E. and Parkvall, S. and Skold, J. and Beming, P. (2010) "3Gevolution: HSPA and LTE for mobile broadband." Academic press.

Di Stefano, A. and Pappalardo, G. and Santoro, C. and Tramontana, E. (2002) “A multi-agent reective architecture for user assistance and its application to e-commerce.” In Cooperative Information Agents VI, Springer. pp. 90-103.

Fielding, R. and Gettys, J. and Mogul, J. and Frystyk, H. and Masinter, L. and Leach, P. and Berners-Lee, Y. (1999) “Rfc 2616, hypertext transfer protocol”

Goredema, C. (2012) “Africa: Predatory cyber crime in south africa - current risks and realities. Transnational Threats and International Crime Division, Institute for Security Studies (ISS), Cape Town. http://www.ngopulse.org/article/predatory-cyber-crime-south-africa-current-risks-and-realities.

Passani, L. (2012).“Http and mobile: The missing header.” http://scientiamobile.com/blog/post/view/id/25/title/HTTP-and-Mobile%3A-The-Missing-Header.

Quiroga, J. and Marin, I. and Rodriguez, J. and Berrueta, D. and Gutierrez, N. and Campos A.M. (2011) “From UAProf towards a universal device description repository.” MobiCASE, volume 95 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 263-282. Springer.

Saha, S. and Jamtgaard, M. and Villasenor, J. (2001) “Bringing the wireless internet to mobile devices. Computer. pp. 34-36:54-58, June.

Shevchik, L. (2013) “Mobile device fragmentation: It's only going to get worse.” http://blog.newrelic.com/2013/05/23/mobile-device-fragmentation-its-only-going-to-get-worse.

Smith, K. and Sanders, D. (2007) “Device description repository requirements 1.0. World Wide Web Consortium” NOTE-DDR-requirements-20071217, December.

Sun, Y. and Zhuang, Z. and Giles, C.L. (2007)“A large-scale study of robots.txt.” In Proceedings of the 16th International Conference on World Wide Web, WWW, ACM. pp.1123-1124, New York, NY, USA

Wikipedia. (2014) “Wurl.” http://en.wikipedia.org/wiki/WURFL.

Wong, R. (2012) “In mobile, fragmentation is forever. Deal with it.” techcrunch. http://techcrunch.com/2010/03/04/mobile-fragmentation-forever.