On the Structure and Characteristics of User Agent String

Jeff Kline
comScore, Inc.
jkline@comscore.com

Aaron Cahn
comScore, Inc.
acahn@comscore.com

Paul Barford
University of Wisconsin-Madison
comScore, Inc.
pb@cs.wisc.edu

Joel Sommers
Colgate University
jsommers@colgate.edu

ABSTRACT

User agent (UA) strings transmitted during HTTP transactions convey client system configuration details to ensure that content returned by a server is appropriate for the requesting host. As such, analysis of UA strings and their structure offers a unique perspective on active client systems in the Internet and when tracked longitudinally, offers a perspective on the nature of system and configuration dynamics. In this paper, we describe our study of UA string characteristics. Our work is based on analyzing a unique corpus of over 1B UA strings collected over a period of 2 years by comScore. We begin by analyzing the general characteristics of UA strings, focusing on the most prevalent strings and dynamic behaviors. We identify the top 10 most popular User Agents, which account for 26% of total daily volume. These strings describe the expected instances of popular platforms such as Microsoft, Apple and Google. We then report on the characteristics of low-volume UA strings, which has important implications for unique device identification. We show that this class of user agent generates the overwhelming majority of traffic, with between 2M and 10M instances observed each day. We show that the distribution of UA strings has temporal dependence and we show the distribution measured depends on the type of content served. Finally, we report on two large-scale UA anomalies characterized by web browsers sending false and misleading UAs in their web requests.

CCS CONCEPTS

• **Information systems → Web log analysis**; **Traffic analysis**; Data mining; • **Software and its engineering →** *Context specific languages*;

KEYWORDS

User Agent Strings, Character Entropy Matrix, Internet Measurement

ACM Reference Format:
Jeff Kline, Aaron Cahn, Paul Barford, and Joel Sommers. 2017. On the Structure and Characteristics of User Agent String. In *Proceedings of IMC '17, London, United Kingdom, November 1–3, 2017,* 7 pages.
https://doi.org/10.1145/3131365.3131406

1 INTRODUCTION

The diversity of client system configurations (defined in terms of hardware, operating system and applications that access content) in the Internet presents significant challenges for application and content service providers. The key challenge is that content must be delivered in formats that are specifically designed for different types of clients in order to provide the best user experience. Content providers often have complex regression test environments to ensure their pages *render and behave* correctly on different platforms [9, 20]. However, the primary mechanism for ensuring that content is *delivered* in the optimal format is the *user agent string (UA)*, which is transmitted by clients as part of a request for content.

UAs are used by a variety of Internet applications (*e.g.,* web, crawlers, mobile apps, etc.) for content negotiation, but we restrict our focus in this paper to their use in the web[1]. The basic format for a web UA is "one or more product identifiers, each followed by zero or more comments" in a format like `product[/version]` [22]. The intent of the UA is to provide sufficient detail about a client system to enable a server to transmit content in the appropriate format and for debugging interoperability problems [18, 22]. As has been observed in prior studies (*e.g.,* [11]), however, UAs themselves are a compelling source of information about client systems in the Internet writ large.

In this paper, we describe our study of UA characteristics. The goal of our work is to provide a new perspective on UAs that will be useful for content service providers and other entities that utilize UAs, to provide a perspective to the research community on Internet client diversity and dynamics, and to provide a perspective that can inform other potential uses of UAs such as client fingerprinting [11, 12]. Our work is based on analyzing a unique corpus of over 1B UA strings collected over a period of 2 years by comScore[2]. Data collection at comScore is enabled by the placement of tags on partner web pages that are activated when clients access those pages. Prior studies (*e.g.* [8]) have found that comScore tags are among the most prevalent third-party tag deployments across the Internet. UAs are captured by comScore servers on tag activation.

[1]Technically, applications such as web browsers are the *user agents*. In this paper, our use of the term UA refers to user agent strings transmitted by web browsers.
[2]We plan to make a subset of this data available to the community on publication.

To conduct our study, we built a UA processing and analysis infrastructure. Similar to other studies of large data sets, our processing infrastructure is primarily Hadoop-based. The unique aspect of our work includes developing a UA parsing capability that is robust to both standard measurement errors (corrupted characters, etc.) and non-standard UA formats. Categorization and volume counts of historical UAs relied on an archive of UA descriptors that comScore maintains.

Our analysis begins by considering the basic characteristics of our UA data from both time-series and entropy-based perspectives. We find that UAs collected for our study exhibit variability on multiple time scales. On daily time scales, our analysis indicates how users shift between devices over the course of a day, and between weekdays and the weekend. On longer timescales, we see clear indications of occasional hardware and software updates. The entropy-based representation provides a characterwise measure of the diversity within the space of UA strings over time. Based on this representation, we find that there is structure within the space of strings and show that changes within the UAs distribution tend to be step-wise, not smooth.

The next step in our analysis considers characteristics of both high- and low-volume strings. We find that the most prevalent UAs comprise only 26% of all traffic. We also find that the rank-frequency distribution of UAs exhibits a power-law-like structure. Finally, we observe O(1M) unique UAs on a daily basis. We describe sources of diversity within the UA space and show that the prevalance of mobile browser apps and developer error are also contributing factors.

The final aspect of our analysis considers anomalous characteristics of strings. We report on two instances where high-volume UA strings are inaccurate or unexpected in significant ways. For example, we find evidence of various anomalous events in our data including spurious appearances of a large volume (O(100M)/day) of outdated UAs, which we diagnose as a software misconfiguration.

In summary, the primary contribution of this paper is in characterizing UA strings. Our results on multiscale UA dynamics have implications for content providers. Our result on the power law-like UA prevalence combined with our results showing that many factors have an impact on the distribution of UAs suggests that simple, UA-based device fingerprinting methods are unlikely to be effective. Finally, our results regarding unwanted behaviors support the notion of efforts related to identifying invalid traffic based on UAs [10]. To the best of our knowledge, ours is the first study to consider UA characteristics broadly.

2 DATA

The data we analyze has been collected by comScore, an Internet measurement company who partners with publishers, brands, ad networks and others for the purpose of reporting on online audience behaviors. comScore acquires its data by providing each of its partners with a block of customized JavaScript code. Each partner then embeds this code within the content it serves. For example, if the partner is a publisher, the publisher deploys comScore's code on each of its web pages. If the partner is a brand running an ad campaign, the code is delivered alongside the ad campaign's creative content (*e.g.* an image with an embeded link).

When executed by a web browser, this code instructs the browser to make an HTTP request to comScore's measurement domain which, in turn, logs each request. The data comScore collects includes the cookie, referrer, UA, a timestamp, and other information pertinent to comScore's business. comScore has clear and strict guidelines concerning the use and the protection of personally-identifiable information.

The daily volume of HTTP requests ingested by comScore is O(50B). Since its mission concerns reporting about online behaviors, comScore also maintains several repositories of high-level historical information about the traffic it observes. These repositories include high-level summary information about UAs. comScore also maintains a code-base that categorizes user agents by manufacturer, device type, browser type and so on.

Our data processing systems are a hybrid combination of Apache Hadoop MapReduce tasks and Greenplum (a variant of Postgres) SQL queries.

3 GENERAL CHARACTERISTICS

A fundamental task of web log analysis is the generation of aggregated statistics over browser, device type and operating system. Typically such statistics are derived from the UA. To help in this effort, the HTTP standard describes general features about the structure and information that belongs within the UA string [22, §5.5.3]. Despite this recommendation, we report below that the space of user agents within a typical web log are, in fact, a loosely-structured, very dynamic and extremely diverse set of strings. In short, using the UA as the basis for web traffic categorization is a complex endeavor.

In this section, we focus our discussion around four challenges faced by processes that categorize UAs: *coverage* of the space, effective *partitioning* of the space, *validation* of the categorization process itself, and coping with the *constant evolution* of browsers, device types and to a lesser extent, operating systems.

We begin by describing the problem of covering the space. A consistent feature that we observe, and which we discuss in the next section is that the rank-frequency distribution of UA strings exhibits a power law-like structure. As a consequence, the majority of web requests have UA strings that are unpopular. This *rare-is-typical* phenomenon means that in order to accurately describe basic features of Internet traffic, one must have an accurate description of rarely-seen user agent strings. On a typical day, comScore servers observe O(1M) distinct UA strings.

To expose some of complexity of proper categorization, we drill down on two user agent strings that each generated about 30M records on May 10, 2017. By volume, both ranked among the top 200 overall. The first belongs to a device that runs the Android operating system:

```
Dalvik/2.1.0 (Linux; U; Android 5.1; F100A Build/LMY47D)
```

This User Agent's device type is reported as `F100A`. Categorizing this device as a phone, tablet, game console or by its manufacturer, however, is simply not possible without an external reference that can supply this information. (We believe it represents a Forsa F100 model.)

Next, the most common Facebook Mobile App user agent observed on May 10 was:

```
Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_1 like Mac OS X)\
 AppleWebKit/603.1.30 (KHTML, like Gecko)\
 Mobile/14E304 [FBAN/FBIOS;FBAV/91.0.0.41.73;\
 FBBV/57050710;FBDV/iPhone8,1;FBMD/iPhone;\
 FBSN/iOS;FBSV/10.3.1;FBSS/2;FBCR/Verizon;\
 FBID/phone;FBLC/en_US;FBOP/5;FBRV/0]
```

This string has over 250 characters and is rich with information. Its internal structure is likewise complex: to delimit internal fields, the string relies on underscores, commas, dots, semicolons, whitespace, slashes and pairs of brackets and parentheses. This complexity implies that pattern-based categorization algorithms that attempt to parse this string must also be complex. Finally, we note that although this UA is very detailed, it is also very popular. Thus, long and detailed user agents do not equate to enhanced user identifiability.

Next, we turn to the partitioning problem. An accurate partitioning of the UA space must distinguish UAs that are close in edit distance to each other. The issue can be illustrated using Microsoft Edge's user agent:

```
Mozilla/5.0 (Windows NT 10.0)\
  AppleWebKit/537.36 (KHTML, like Gecko) \
  Chrome/42.0.2311.135 Safari/537.36 Edge/12.10136
```

This string agrees with Google Chrome's user agent up to the suffix, `Edge/12.10136`. Despite Edge and Chrome having distinct codebases and histories, they are surprisingly close to each other in the user agent space.

This observation shows that *ad-hoc* rules are a necessary component of UA categorization, which generally confounds validation of the categorization process. More generally, the requirement for *ad-hoc* rules implies that accurate partitioning is an intrinsically complex task. As further evidence of this, we note that the internal ruleset used by comScore is several thousand lines long. Publicly visible evidence of this complexity can be found in various online resources, *e.g.*, see [2, 3, 5, 6].

Some portions of the UA string tend to be more diverse than others, and these regions of diversity are evidence of structure. To expose this structure, and to isolate features that exist in time, we introduce a representation of the UA space that we call a *character entropy matrix*. Two examples of this representation appear in Figure 1. Each row of the character entropy matrix summarizes one day of data. Within each row, the jth column displays the empirical entropy of the jth character of UAs observed on that day. For an intuitive sense of what is being represented by the matrix, the first several characters of many user agents are either `Mozilla` or `Dalvik`. Thus, the first few columns of the character entropy matrix have low entropy as indicated by the red stripe along the extreme left sides of each image. More generally, this representation can identify regions within any population of UAs that are either unusually diverse or uniform. The utility of this matrix is that it effectively quantifies diversity in both space (*i.e.* by character) and in time with a simple and comprehensible algorithm.

The dynamic nature of the UA space means that generating Internet traffic with a realistic profile of UA's is challenging. As was shown in [25], the Athena botnet administrative portal provides functionality that allows the controller to customize the distribution of user agents used by the botnet. A feature common to traffic that is generated by malware is that it has a distribution of UAs that is skewed towards either out-dated browsers or has subfields within the user agent list populated with a limited set of strings. However, in the next section we show that outdated browser versions and inconsistent information within the UA string are, in fact, common and benign features of traffic. Consequently, detection of malicious activity cannot rely entirely on these attributes.

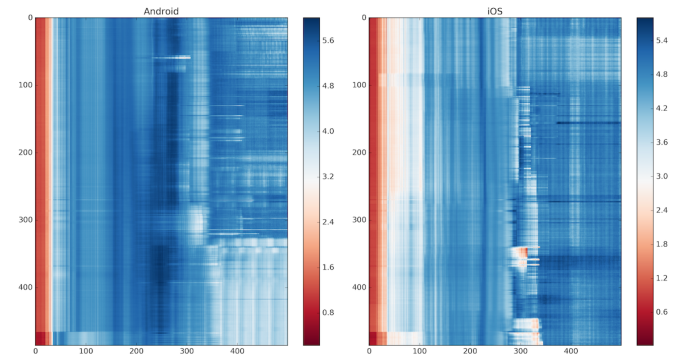

Figure 1: The character entropy matrices are displayed for Android and iOS UAs. The data shown spans 16 months, January 1 2016 corresponds to row 0. The vertical strip of low entropy on the left of each image is the result of many UAs starting with either `Mozilla` or `Dalvik`. For a sense of scale, this image summarizes several trillion web requests.

4 DETAILED RESULTS

In this section, we present details of our UA analysis, including characteristics over time, prevalence of different strings, and analysis of anomalous UAs.

4.1 Analysis of the UA distribution in time

The population of UAs changes over time in two distinct ways. First, the continual introduction of new devices, browsers, and version updates implies that the set of UAs increases over time. Meanwhile, old devices and obsolete browsers fall into disuse and eventually vanish from the active population. We quantify this churn over time with the following analysis. We cast the top 10^k UAs seen each Wednesday as a set[3]. We then measure the Jaccard similarity of sets belonging to consecutive Wednesdays to create a time series. Figure 2 shows the resulting time series that spans 70 consecutive weeks starting January 2016. We evaluate the total variation of the time series to quantify the stability of the Jaccard similarity metric.

The total variation of the 10 most prevalent UAs is large while the total variation of the top 1000 is a local minimum. This suggests the following heuristic: from week-to-week, the top 1000 UAs have Jaccard similarity of about 0.7. The set of top 10 and 100 UAs varies more widely from week to week, which can be explained as follows. Each week, several spots among the top 10 UAs are occupied by variants of the Chrome UA. Chrome has a frequent update cycle and its default configuration aggressively updates itself. Whenever a new version of Chrome is released, the spots occupied by Chrome

[3]Traffic volume during a typical week is maximal on Wednesdays.

at the top are replaced with newer versions, and the top browsers fall out of favor.

Android devices generate a large fraction of Internet traffic and the Android space is very diverse. Yet the Android space does not contribute much to the overall churn of the top 100 UAs. Our explanation is that since the Android space is so fragmented, few individual Android UAs accumulate enough volume to be among the top 100 yet they occupy many positions in higher aggregates.

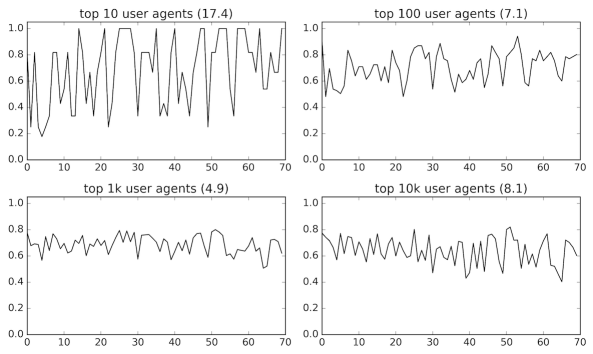

Figure 2: We quantify the churn of the UA population by casting the top UAs seen each Wednesday as a set and measuring the Jaccard similarity of consecutive Wednesdays. The resulting timeseries for the top $10, 100, 1k$ and $10k$ UAs spanning 70 weeks starting January 2016 are shown above. The total variation of these timeseries has a local minimum at 1000 which supports the heuristic that week-over-week Jaccard similarity of the top 1000 UAs is about 0.7.

The second way that the UA distribution changes over time is due to the fact that user habits throughout the day and week are not uniform. Figure 3 is based on publisher traffic and it splits the traffic into several broad categories. The weekly view (top plot) shows data acquired in Sept 2015 while the hourly view (bottom plot) spans 48 hours starting on Wednesday May 10, 2017. The proportion of iPhone traffic varies dramatically, rising from 7% on Monday to 27% on Sunday. The hourly time series shows iPhone traffic doubling from about 7% at 8:00UTC to 14% at 23:00UTC. This supports the notion that evening and weekend browsing habits differ from mid-week workday habits. It also suggests that one's susceptibility to being identified based solely on browser attributes depends on the time-of-day and day-of-week.

We find that an accurate view of the distribution of UAs is not a simple function of scale or reach but also a function of the role one has on the Internet. As mentioned earlier, comScore measures traffic that is associated with publishers, advertisers and ad networks. Each data type provides a distinct perspective on the Internet. Collectively, each group sees a broad swath of Internet traffic with typical daily volume between 5B and 20B records.

Using this set of data, we try to answer the question, What was the most common browser? To simplify the results, we limit our focus to Microsoft Edge's rank by volume within each class. Microsoft Edge is consistently among the 10 most popular browser versions and, in contrast to Chrome and Firefox, its version history

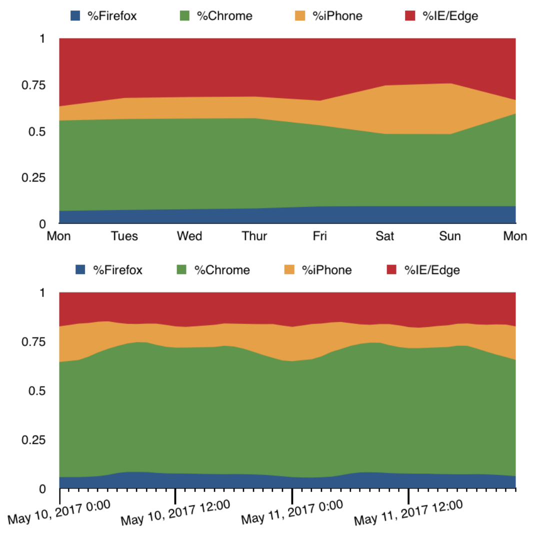

Figure 3: The browser distribution depends on day and time. This is visible in the above time series that show data spanning one week in Sept 2015 (*top*) and the 48-hours starting on May 10, 2017 (*bottom*). The fraction of iPhone traffic on the weekend shown was nearly four times larger than it was on weekdays. The hourly view shows that iPhone traffic varied between 9% (May 10 08:00UTC) and 18% (May 11 02:00UTC).

does not fragment over version updates. The data for this report was observed on May 10, 2017.

Microsoft Edge was the the top-ranked UA within ad campaign data with over 300M records. Within Publisher data, Edge had the 6th largest volume with 400M records while within Ad Net data, Edge ranks 8th overall with 80M records. This shows that the UA distribution is not stable across data type—even when the measurements have both scale and reach.

To elaborate on this point, a web server that delivers billions of advertisements may be serving mostly to mobile apps. Social media companies such as Facebook and Twitter have dedicated apps that influence users' behavior. Ad block usage may correlate with browser choice. Even comparisons across similar publishers can be skewed by current events, content type and audience demographics.

To summarize, answering a straightforward question about browser dominance does not have a straightforward answer. Additionally, this question cannot be adequately addressed simply by aggregating over more data having wider reach.

4.2 Analysis of UA prevalence

A current list of the most prevalent UAs are shown in Table 1. Broadly speaking, this list is stable over time and includes Microsoft Edge, Microsoft Internet Explorer, Google Chrome, iPhone Safari and Firefox. Notably absent from this list is Android, despite Android devices having a larger market share than iPhone. In fact, the top Android-based UA ranks 48th overall. The cause of this discrepancy is fragmentation of the Android market: each manufacturer's device generates a distinct user agent. There is also a great amount of churn in this list: the top Chrome UA from early April ranks 74th in mid May. This is due to aggressive version updates for that browser.

Every UA categorization method faces the problem of describing enough of the UA space so that it applies to an acceptably large fraction of Internet traffic. We call this the *coverage* problem. The challenge that this presents is visible in Figure 4. This figure is based on web log data acquired on the second Wednesdays of Feb, Mar, Apr and May 2017. For each of these Wednesdays, every UA is plotted against its total daily volume and the set of UAs is then ordered by volume. The resulting line closely follows a power-law rule, which is a signature of fat-tailed (*i.e.* rare-is-typical) populations.

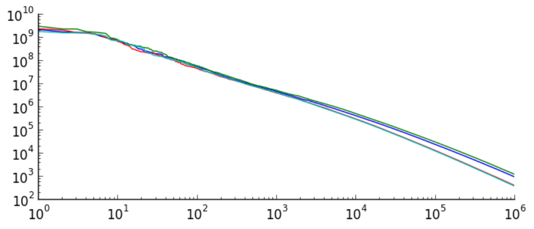

Figure 4: The daily distribution of user agents is well approximated as a power law distribution. This figure displays four such examples, based on on data acquired during the second Wednesday of Feb, Mar, Apr and May 2017. The top 10 user agents comprise about 26% of total volume. The remainder consists of $O(1M)$ user agents and this portion contributes about 74% of total daily volume. This illustrates the covering problem that is faced by user agent categorization processes.

For each of these days, the top 10 user agents comprised approximately 26% of total daily volume while the remaining set of user agents (a set of size $O(1M)$) generates the other 74% of traffic. Effective coverage of the space means that one has accurate descriptions for examples within the long tail. Since the number of distinct examples within the tail is large, accurate and complete categorization is a complex endeavor.

As already mentioned, one source of this diversity includes browser version updates. Another source of UA diversity comes from fragmentation within the mobile sphere. A third source is the combination of mobile device names with in-app browsers. On May 10, over 1.5M Android UAs that had more than 100 records issued requests to comScore's servers. Of these, about 500K appear to be related to the Facebook app.

To summarize, validation of UA categorization methods is too large a task for manual review to be effective. Also, there does not appear to be a natural threshold below which low-volume UA's may be disregarded.

4.3 Analysis of anomalous UA's

We now report on two instances where the UA is configured to inaccurately represent the web browser. Each instance was associated with over O(100M) requests per day.

The first concerns a feature of Microsoft's browsers called the Compatibility View function [4]. This function is informed by an XML-formatted file maintained by Microsoft that essentially contains domain-UA pairs. The purpose of this file is to alter the behavior of Microsoft's browsers based on the domain being visited. We first became aware of this list while reviewing an alert raised by an internal traffic quality monitor that was built to identify anomalies within the distribution of user agents.

The compatibility view list is updated monthly. Its contents includes a wide variety of publishers, government web sites, and financial companies. As of this writing, the current XML file for Edge[4] includes about 1600 domains.

For example, the current Internet Explorer list[5] shows that when visiting the financial site, chase.com, the browser should use the Firefox 22 token, which is represented in the file as:

```
Mozilla/5.0 ($PLATFORM; Trident/7.0; rv:11.0)\
   like Gecko/20100101 Firefox/22.0
```

This UA contains the Microsoft-specific token `Trident` but also contains an out-dated version of Firefox.

The second anomalous instance concerns a dramatic shift within the UA space that occurred during 2015. In April of that year, the daily volume of the following Android UAs suddenly increased from O(10) records per day to O(100M) records per day:

```
Mozilla/5.0 (Android; U; Android 2.1; en-us;)\
  AppleWebKit/525.10 (KHTML,like Gecko)
```

In other words, prior to April, the volume of this UA was backscatter. By July 2015, this UA was among the top 100 worldwide. This event was discovered during a routine manual review of data quality.

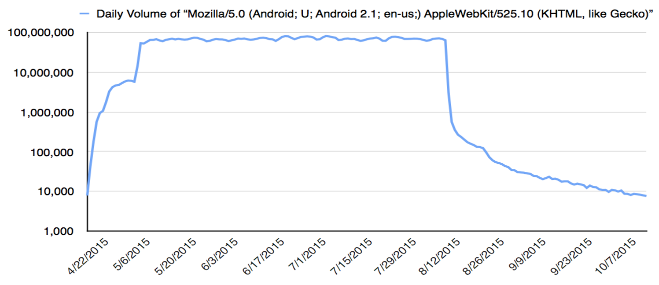

Figure 5: The daily volume of the Android User Agent anomaly of summer 2015.

This User Agent belongs to a device running Android version 2.1. As documented elsewhere[6], this version of Android was most prevalent during 2010–2011. To summarize, the data showed a sudden large shift within the mobile space towards a browser version that was four years out of date. Figure 5 documents the total daily volume of this event.

This event was visible from the perspectives of publisher, brand and ad network. Additionally, no single publisher, ad campaign or other entity appeared to be a target of this traffic. Since comScore also collects browser cookie information, we were able to follow

[4]http://cvlist.ie.microsoft.com/edge/desktop/1432152749/edgecompatviewlist.xml
[5]http://cvlist.ie.microsoft.com/IE11/1426178821/iecompatviewlist.xml
[6]https://en.wikipedia.org/wiki/Android_version_history

Length	Edit Dist	User agent
109	0	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
128	31	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36\ Edge/14.14393
68	58	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
110	3	Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
137	65	Mozilla/5.0 (iPhone; CPU iPhone OS 10_3_1 like Mac OS X) AppleWebKit/603.1.30 (KHTML, like Gecko) \ Version/10.0 Mobile/14E304 Safari/602.1

Table 1: The five most prevalent user agents seen on May 3 2017. The string length reports character count. The edit distance reports the Levenshtein distance from the top-ranked User Agent. The top-ranked user agent from a month prior (April 5) ranked 74th on May 3 and is not displayed.

the evolution of several cookies as they transitioned from a current browser to the anomalous one and back again. This allowed for a partial diagnosis of the the cause: the event appeared to be mostly related to the Facebook app running on Samsung devices. Based on this, we conjecture that a software misconfiguration released in April 2015 by one of Samsung, Google or Facebook was the underlying cause. A more precise diagnosis was not attempted since the event, despite its large scale and obvious inaccuracy, appeared benign.

5 RELATED WORK

The HTTP User-Agent string has been used in a number of prior studies to estimate browser version populations, operating system version prevalence, and device populations, particularly mobile handheld devices, *e.g.*, [13, 15, 19, 24]. These prior studies have noted that their population estimates have appeared to be in line with industry estimates, *e.g.*, provided by such companies as `netmarketshare.com`. Third party services such as `udger.com` and `deviceatlas.com` provide UA string parsing libraries and device catalogs to help companies track various subpopulations of visitors to their websites. Maier *et al.* have also used the combination of UA string and client IP address to identify multiple devices residing behind NAT devices [19]. We believe that our results can help inform these string parsing libraries and device catalogs.

The UA string has been commonly used by servers to deliver appropriately-sized content to mobile devices and other devices with constrained screen sizes. This is so common that many web application frameworks contain built-in capabilities for providing different content variants, *e.g.*, [1]. Some researchers have relied on this behavior by spoofing the UA request header in order to analyze differences in content delivered to mobile versus desktop devices [21, 26].

RFC7231 recommends that a client should not use a UA string with "needlessly fine-grained detail" [22] to avoid host fingerprinting, and prior studies have evaluated the potential for the UA string to be used to identify users [7, 11, 27]. Yen *et al.* found that 60–70% of users could be identified through the UA string [27], similar to findings by Eckersley [11]. Our results complement and enhance those of [27] in the following ways: (1) the long tail in observed UAs that we see suggests users may be identified through the UA as the earlier work showed, but (2) our analysis of the time-varying nature of observed UAs suggests that anonymity sets evolve rapidly over hours of the day and days of the week due to user behavior (shifting from one device to another), software updates (*e.g.*, rollout of a new Chrome version), and browser behavior (*e.g.*, Edge target-specific behaviors).

Other studies have evaluated UA strings to detect the presence of malware by using anomalies such as misspellings and inconsistent information in HTTP headers (*e.g.*, iOS and Flash) [14, 16, 23]. Kotov and Massacci studied the source code of several exploit kits and found that server-side malware used the UA string presented by clients as a way to detect potentially vulnerable operating systems and devices [17]; the exploit code would mimic an "innocent" site if the UA indicated a non-vulnerable host.

6 SUMMARY AND CONCLUSIONS

Analysis of User Agent strings transferred during web transactions offers a compelling perspective on understanding client systems in the Internet. In this paper, we describe our study of UAs, which is based on a corpus of over 1B UA strings collected over a period of 2 years by comScore. To conduct this study, we constructed a UA parsing and analysis infrastructure that is robust to the wide variety of strings in our data set. Our analysis of the general characteristics of UA strings reveals that the most prevalent strings comprise about 26% of all UAs and are composed of the expected instances of popular platforms such as Google, Microsoft and Apple. We also find that the rank-frequency distribution of strings is consistent with a power law. Our analysis of the UAs observed at multiple time scales indicates dynamic characteristics that can be explained by day-to-day user behavior and by periodic updates to hardware and software platforms. Finally, we examine UA strings that have been identified as anomalous or pertaining to unwanted or malicious activity. These UA strings reveal that there are instances of unexpected anomalies. In on-going work, we continue to expand and drill down in our analyses. We expect this will reveal a variety of characteristics—especially in the long tail of the data—that will improve analysis, test and content negotiation capabilities.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Theophilus Benson for their feedback. This material is based upon work supported by DHS grant BAA 11-01 and AFRL grant FA8750-12-2-0328. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DHS or AFRL.

REFERENCES

[1] [n. d.]. Action Pack Variants (Ruby on Rails 4.1 Release Notes). http://edgeguides.rubyonrails.org/4_1_release_notes.html#action-pack-variants. ([n. d.]). Accessed August 2017.

[2] [n. d.]. Panopticlick. ([n. d.]). https://panopticlick.eff.org/

[3] [n. d.]. Udger. ([n. d.]). https://udger.com/resources/ua-list

[4] [n. d.]. Understanding the compatibility view list. https://msdn.microsoft.com/en-us/library/gg622935(v=vs.85).aspx. ([n. d.]). Accessed August 2017.

[5] [n. d.]. UserAgentString. ([n. d.]). http://www.useragentstring.com/pages/useragentstring.php

[6] [n. d.]. WhatIsMyBrowser.com. ([n. d.]). https://www.whatismybrowser.com/developers/tools/user-agent-parser/browse

[7] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. 2011. User tracking on the web via cross-browser fingerprinting. In *Nordic Conference on Secure IT Systems*. 31–46.

[8] Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. 2016. An Empirical Study of Web Cookies. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 891–901. https://doi.org/10.1145/2872427.2882991

[9] Shauvik Roy Choudhary, Husayn Versee, and Alessandro Orso. 2010. Webdiff: Automated identification of cross-browser issues in web applications. In *IEEE International Conference on Software Maintenance (ICSM)*. 1–10.

[10] Media Ratings Council. [n. d.]. Invalid Traffic Detection and Filtration Guidelines Addendum. ([n. d.]). http://mediaratingcouncil.org/101515_IVT%20Addendum%20FINAL%20(Version%201.0).pdf

[11] Peter Eckersley. 2010. How unique is your web browser?. In *International Symposium on Privacy Enhancing Technologies Symposium*. 1–18.

[12] EF Foundation. [n. d.]. Panopticlick. ([n. d.]). https://panopticlick.eff.org/

[13] Aaron Gember, Ashok Anand, and Aditya Akella. 2011. A comparative study of handheld and non-handheld traffic in campus Wi-Fi networks. In *International Conference on Passive and Active Network Measurement*. 173–183.

[14] Martin Grill and Martin Rehák. 2014. Malware detection using HTTP user-agent discrepancy identification. In *Information Forensics and Security (WIFS), 2014 IEEE International Workshop on*. 221–226.

[15] Sunghwan Ihm and Vivek S Pai. 2011. Towards understanding modern web traffic. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 295–312.

[16] Nizar Kheir. 2013. Analyzing HTTP user agent anomalies for malware detection. In *Data Privacy Management and Autonomous Spontaneous Security*. 187–200.

[17] Vadim Kotov and Fabio Massacci. 2013. Anatomy of exploit kits. In *International Symposium on Engineering Secure Software and Systems*. 181–196.

[18] Balachander Krishnamurthy. 2001. *Web protocols and practice: HTTP/1.1, Networking protocols, caching, and traffic measurement*. Addison-Wesley Professional.

[19] Gregor Maier, Fabian Schneider, and Anja Feldmann. 2010. A first look at mobile hand-held device traffic. In *International Conference on Passive and Active Network Measurement*. 161–170.

[20] Ali Mesbah and Mukul R Prasad. 2011. Automated cross-browser compatibility testing. In *Proceedings of the 33rd International Conference on Software Engineering*. 561–570.

[21] Jitu Padhye and Henrik Frystyk Nielsen. 2012. *A comparison of SPDY and HTTP performance*. Technical Report MSR-TR-2012-102.

[22] J. Reschke and R. Fielding. 2014. RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. http://tools.ietf.org/html/rfc7231. (June 2014).

[23] Christian Rossow, Christian J Dietrich, Herbert Bos, Lorenzo Cavallaro, Maarten Van Steen, Felix C Freiling, and Norbert Pohlmann. 2011. Sandnet: Network traffic analysis of malicious software. In *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. 78–88.

[24] Fabian Schneider, Bernhard Ager, Gregor Maier, Anja Feldmann, and Steve Uhlig. 2012. Pitfalls in HTTP traffic measurements and analysis. In *International Conference on Passive and Active Network Measurement*. 242–251.

[25] Kevin Springborn and Paul Barford. 2013. Impression Fraud in On-line Advertising via Pay-Per-View Networks. In *USENIX Security*. 211–226.

[26] Paul J Timmins, Sean McCormick, Emmanuel Agu, and Craig E Wills. 2006. Characteristics of mobile web content. In *Hot Topics in Web Systems and Technologies, 2006. HOTWEB'06. 1st IEEE Workshop on*. 1–10.

[27] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. 2012. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. In *NDSS*.