Not only E.T. Phones Home: Analysing the Native User Tracking of Mobile Browsers

John Pegioudis
FORTH & University of Crete
Greece

Emmanouil Papadogiannakis
FORTH & University of Crete
Greece

Nicolas Kourtellis
Telefonica Research
Spain

Evangelos P. Markatos
FORTH & University of Crete
Greece

Panagiotis Papadopoulos
FORTH
Greece

ABSTRACT

Contemporary browsers constitute a critical component of our everyday interactions with the Web. Similar to a small, but powerful operating system, a browser is responsible to fetch and run web apps locally, on the user's (mobile) device. Even though in the last few years, there has been an increased interest for tools and mechanisms to block potentially malicious behaviours of web domains against the users' privacy (e.g., ad blockers, incognito browsing mode, etc.), it is still unclear if the user can browse the Web in private.

In this paper, we analyse the natively generated network traffic of 15 mobile browser apps under different configurations to investigate if the users are capable of browsing the Web privately, without sharing their browsing history with remote servers. We develop a novel framework (Panoptes) to instrument and monitor separately the mobile browser traffic generated by (a) the web engine and (b) natively by the mobile app. By crawling a set of websites via Panoptes, and analyzing the native traffic of browsers, we find that there are browsers (i) who persistently track their users, and (ii) browsers that report to remote servers (geolocated outside EU), the exact page and content the user is browsing at that moment. Finally, we see browsers communicating with third-party ad servers while leaking personal and device identifiers.

CCS CONCEPTS

• **Security and privacy** → **Web application security**; • **Information systems** → *Traffic analysis*; **Online advertising**.

KEYWORDS

Mobile Browsers, Native Traffic, User Tracking, Browser History

ACM Reference Format:
John Pegioudis, Emmanouil Papadogiannakis, Nicolas Kourtellis, Evangelos P. Markatos, and Panagiotis Papadopoulos. 2023. Not only E.T. Phones Home: Analysing the Native User Tracking of Mobile Browsers. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3618257.3624842

IMC '23, October 24–26, 2023, Montreal, QC, Canada

ACM ISBN 979-8-4007-0382-9/23/10.
https://doi.org/10.1145/3618257.3624842

1 INTRODUCTION

The proliferation of web applications has made contemporary browsers a critical component of our interactions with the Internet. Users use their browser as a gateway to the Web, accessing remote content and services. This functionality makes browsers' role similar to a small but powerful operating system where providers can instantly run their (web) applications locally, on the user's device.

Unfortunately, there are many examples of web providers either delivering malware or spying on their users via tracking their behavior [1–5]. Thus, in recent years, we see a great variety of methods, features [6–8] and products (e.g., ad-blocking extensions or browsers) aiming at detecting and blocking such malicious user tracking behaviors of websites and applications. Thus, *have users won this privacy war? If they block the tracking requests generated by the web engine, can they finally browse in private?*

In this study, we aim at investigating this exact question, motivated by the fact that considering its key role (acting as the user's gateway), the browser can *know what content the user browses and when.* Consequently, there is a hyper-concentration of personal information in the browser (even greater in recent years, as more than 58% of the web traffic comes from the more personal, mobile devices [9]), that enables the browser to accurately and fully estimate the preferences and interests of the user, at any time.

Towards this goal, we develop a first of its kind framework to instrument mobile browsers and monitor the outgoing traffic of both the web engine and the mobile browser app itself. Hence, we are able to analyse the behavior of (i) the websites when rendered in the web engine but more importantly (ii) the browser's, when communicating with second or third-party web entities, or the remote server of the browser vendor (via the so called "phone home" requests).

In summary, the contributions of this study include:

(1) We propose Panoptes: a framework to instrument and monitor separately the mobile browser traffic which is generated by (i) the web engine and (ii) natively by the mobile app.
(2) We implement our framework and by instrumenting 15 different mobile browsers, we crawl 1000 websites. We provide our tool open source to aid the community with mobile browser auditing[1].
(3) By analyzing the captured traffic, we see that natively generated traffic can be as high as 1/3 of the total generated traffic. We also see that when 3 of the tested browsers "phone home", they report to servers located outside the EU (where

[1]Source code of Panoptes: https://github.com/BanForFun/panoptes

the crawling was performed), the exact page and content the user is browsing. One browser, in particular, does so together with a persistent identifier so users can be tracked even if they use Tor [10] or a proxy. In addition, we see 3 other browsers communicating with ad servers or Facebook Graph API while leaking user personal information and other device-specific identifiers.

2 BROWSER TRAFFIC MONITORING

To better understand the behavior of mobile browsers and their actions, we need to monitor their network traffic while we automate their crawling campaigns. The easiest way to achieve this would be via instrumenting browser apps installed in emulated devices. However, this is something that could alter the behavior of websites (i.e., non-realistic web visits) and harm the reproducibility of the crawls [11].

To that extent, we develop Panoptes: the first of its kind framework to instrument instances of various browser applications residing in a physical device and monitor their outgoing network traffic. Our framework is able to capture and differentiate the network traffic generated by the web engine, from the traffic that the browser app natively creates. Panoptes automatically initiates browser instances, visits websites and captures their network traffic.

As a testbed, we utilize a Linux Desktop responsible for the instrumentation of mobile browsing applications and an Android SM-T580 Samsung Galaxy Tablet, running Android 11[2], that hosts the mobile apps, and registers and stores network traffic. In Figure 1, we provide a high level overview of our methodology. To instrument browser instances and automate operations, we use the UI automator of Appium [13], Frida [14] instrumentator and Chrome DevTools Protocol (CDP) [15]. The implementation of our framework is publicly available.

2.1 Crawling campaigns

Before starting every crawling campaign, we reset the browser application to its default factory settings using Appium. Then, we start each browser using Frida and go through the setup wizard manually to test various configurations. To visit a website, we employ CDP and instrument the page object to navigate to a specific domain. For browsers that do not support CDP, we hook into the WebView's functions using a custom Frida script and instrument them accordingly.

To ensure that the auto-complete feature of modern browsers will not pollute our network traces, we navigate to every website directly by using CDP or Frida without typing the domain on the address bar. When we visit the website, we consider that it is ready when the `DOMContentLoaded` event has been triggered, or when 60 seconds have passed since the visit started. Then, we wait for an additional period of 5 seconds to ensure that the website has completely loaded and that all its operations have taken place.

[2]While our work focuses on Android, similar behavior is expected to appear by the same browsers while running on iOS, since past findings have indicated so [12]. Verifying that is part of our planned future work.

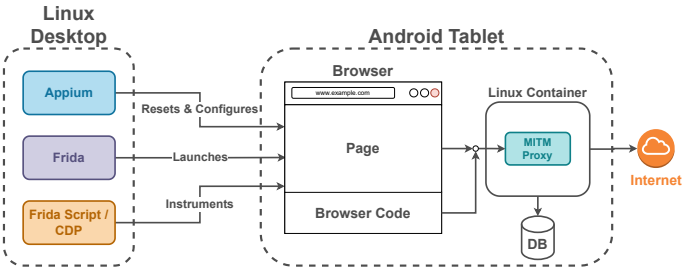

Figure 1: Overview of our framework system design.

Table 1: Our dataset of mobile browsers along with their version number.

Browser	Version	Browser	Version
Chrome	113.0.5672.77	**DuckDuckGo**	5.158.0
Edge	113.0.1774.38	**Dolphin**	12.2.9
Opera	75.1.3978.72329	**Whale**	2.10.2.2
Vivaldi	6.0.2980.33	**Mint**	3.9.3
Yandex	23.3.7.24	**Kiwi**	112.0.5615.137
Brave	1.51.114	**CocCoc**	117.0.177
Samsung	20.0.6.5	**UC International**	13.4.2.1307
QQ	13.7.6.6042		

2.2 Monitoring encrypted browser traffic

To monitor and capture encrypted network traffic, Panoptes utilizes a Man-In-The-Middle (MITM) proxy. Specifically, in the Android tablet, a Debian container with mitmproxy [16] is installed (along with its CA certificate) in transparent mode. We use this mode to proxy all traffic at the network layer without having to configure each browser app[3].

For every browser we analyze, Panoptes extracts their unique kernel UID under which each browser process is running [17] to create *iptable* rules and divert their traffic through the proxy. In addition to this, Panoptes creates rules to block all HTTP/3 traffic, as at the time of crawling, mitmproxy did not support the QUIC protocol. This should not affect the behavior of crawled websites, since they also support older HTTP versions which the browsers automatically fall back to.

2.3 Splitting native & web engine browser traffic

A very important contribution of Panoptes is its ability to differentiate between traffic that (i) was generated in the web engine by the website itself as part of its operations, or due to the actions of the user, and (ii) the traffic generated natively by the browser app itself, as designed by the browser vendor (i.e., native traffic). To achieve this, we make use of CDP and for each browser instance, we intercept all HTTP requests initiated by the website. In the case of the *UC International* browser, we use Frida to hook into an internal API. Then, for each intercepted request, we perform tainting by piggybacking an additional custom HTTP header using the 'x-' prefix that does not interfere with existing headers.

[3]Using a MITM proxy, can cause issues in apps that use certificate pinning by preventing them from issuing some requests (to pinned domains). In this paper, we make no attempt to bypass certificate pinning, and we consider our results as the lower bounds of the leaking that may happen in these browsers.

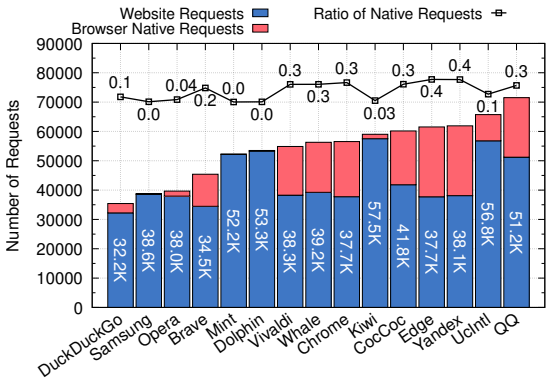

Figure 2: Number of requests generated by the website and the browser. In the cases of Edge and Yandex the ratio of natively generated requests of the brower app (red) to the requests generated by the web engine (blue) can reach as high as 0.38 and 0.39 respectively.

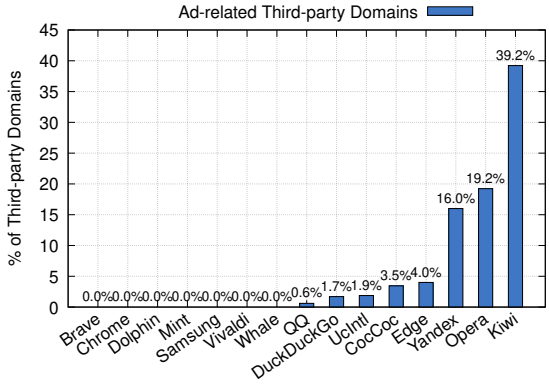

Figure 3: The percentage of domains that browsers send native request being third party and ad related.

Considering that all the network traffic of the Android tablet flows through the MITM proxy, in Panoptes, we have developed a custom MITM add-on to inspect all headers and separate the tainted ones. Thus, when HTTP requests arrive at the proxy, the MITM addon intercepts them at runtime, filters the tainted ones (i.e., requests originated from the website) before removing the additional (custom) header and forwarding them to their original destination. If a request is not tainted, it means that the request was generated natively by the browser app. The two different categories of the requests are finally stored in different local databases.

3 DO BROWSERS TRACK THEIR USERS?

By using the above methodology, we investigate our motivating question: *Can users browse without being tracked*?

Data collection: Hence, we download and install the top [18] 15 mobile browser apps[4] from Google Play Store (see Table 1 for details). Additionally, we (i) obtain the top 500 most popular websites based on the Tranco list [20] and (ii) collect an extra 500 websites that are associated with sensitive information based on the Curlie directory [21]. To achieve the later, from the Curlie directory, we manually select websites associated with sensitive issues regarding Society (e.g., warfare and conflict), Religion, Sexuality and Health (e.g., mental health). Hence, we form a set of 1000 websites [22], and we use Panoptes to visit each of these sites via each of the mobile browsers in our dataset. Our crawls were performed from an EU-based vantage point.

3.1 Natively generated network traffic

Next, we set out to explore the volume of traffic generated natively by the browser app. In Figure 2, we plot for each browser, the number of requests generated (i) by all visited websites (blue) and (ii) the number of requests the browser natively generated (red). Additionally, we compute in the plot (black line) the ratio of the total network traffic the natively generated requests of the browser app amount to. We see that a lot of browsers generate a substantial amount of network traffic that is not related to the website the user has visited. Specifically, for browsers like Vivaldi, Whale, CocCoc, Edge and Yandex, more than 1/3 of the total network traffic (as high as 0.39 and 0.38 in case of Edge and Yandex, respectively) is attributed to native requests that the browser sends to a remote server. It is important to note that this amount of additional (native) traffic is not related at all to the content the user is interested in and navigated to. As measured in the past [23, 24], such unsolicited network traffic consumes system resources and energy from the user's device.

Interestingly, as depicted in Figure 3, 8 of the browsers in our dataset issue native requests to third-party (ad) servers, as classified by the popular Steven Black host list [25]. Surprisingly, almost 40% of the distinct domains to which Kiwi sends native requests to are ad or analytics-related domains, including *rubiconproject.com*, *adnxs.com*, *openx.net*, *pubmatic.com*, *bidswitch.net*, *demdex.net*, etc.. Similarly, 19.2% and 16% of the domains with which Opera and Yandex communicate, respectively, are ad or analytics-related, including *appsflyersdk.com* and *doubleclick.net*. We also see CocCoc and Edge browsers sending native requests to analytics services like *adjust.com*, whereas it is interesting to note that CocCoc browser is an ad-blocking browser that enforces the easylist filterlist in its web engine [26].

In Figure 4, we plot the volume of outgoing network traffic (i.e., HTTP(S) requests) generated by both the browser and the websites. We discover that some browsers generate a substantial amount of traffic. Specifically, in the case of the QQ browser, we see that the additional traffic generated by the browser app can amount to a staggering 42% extra traffic. Of course, such unsolicited and unnecessary traffic can have considerable impact on the user's data plan and performance.

3.2 Leaking the user browsing history

As a next step, we analyze the information that the natively-issued (from within the browser app) HTTP(S) requests report to remote servers. When such remote servers are controlled by the browser vendor or a subsidiary of theirs, we call these generated requests *"phone-home"* requests.

[4]We exclude Firefox from our study because it supports different instrumentation protocols [19] which are incompatible with Panoptes.

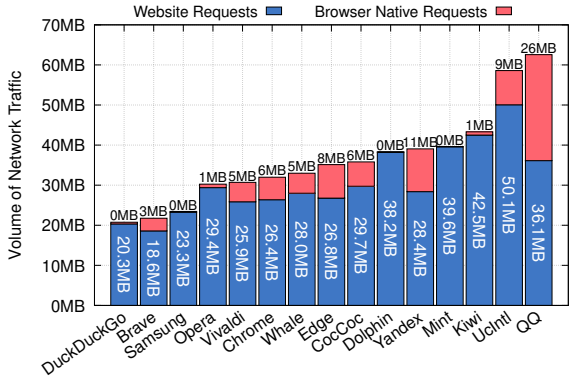

Figure 4: Volume of network traffic generated by (blue) the website and (pink) the browser itself natively. In the case of QQ browser, the additional traffic generated by the browser app can amount to an extra 42%.

```
POST https://s-odx.oleads.com/api/v1/sdk_fetch
body: {"channelId":"adxsdk_for_opera_ofa_final","countryCode":"ANONYMIZED",
"availableServices":["GOOGLE_PLAY"],"languageCode":"EN","appPackageName":"com
.opera.browser","appVersion":"75.1.3978.72329","sdkVersion":"1.12.2","deviceType":
"PHONE","osType":"ANDROID","osVersion":"11","deviceVendor":"Samsung","device
Model":"SM-T580","deviceScreenWidth":1200,"deviceScreenHeight":1920,"operaId":
"7e5d1382f2dd484e9d036519c8a908ddd5de945b100bc9e66582e2ed4ab0b2ab","operator":"",
"connectionType":"WIFI","userConsent":"false","duration":0,"latitude":ANONYMIZED,
"longitude":ANONYMIZED,"positionTimestamp":1683927615,"placementKey":"s569498648
9856","adCount":2,"floorPriceInCent":0,"timestamp":1683927615,"token":"e4818505
a103a7fc3b3e74174c37e570","supportedAdTypes":["SINGLE"],"supportedCreativeTypes":
["BIG_CARD","DISPLAY_HTML_300x250","LEADS","NATIVE_NEWSFLOW_1_IMAGE","NATIVE_
NEWSFLOW_3_IMAGES","POLL","PREBID_INTERSTITIAL","PREBID_NATIVE","SURVEY_SINGLE_
CHOICE","SURVEY_MULTIPLE_CHOICE","SURVEY_FEW_QUESTIONS","VAST_3_URL","VAST_3_XML",
"NATIVE_VAST","VIDEO_16x9"],"supportedFeaturesList":[{"cache":["v1"]},{"video":
["novast"]},{"display":["mrect","mraid2"]}],"omidpn":"Opera","omidpv":
"omsdk-1.3.16-Opera"}:
```

Listing 1: Native ad request issued by Opera

The Yandex case: By parsing the traffic generated by the Yandex browser app, to our surprise, we see that the browser explicitly leaks the user's browsing history via *"phone-home"* requests to its vendor's servers. Specifically, we find that for each page visited, the browser sends a native request towards *sba.yandex.net* with the visited URL (Base64) encoded in the request parameters, along with a second request towards *api.browser.yandex.ru* with the visited hostname, and a unique identifier of the user. This means Yandex company can track the user persistently even if they erase cookies, or change their IP address or use Tor/anonymous proxy or VPN!

To add insult to injury, these user tracking requests are not being issued only the first time the user visits a website, but on each and every page visit. This means that *the remote server knows exactly what and when the user browses it*, thus leaking temporal and behavioral patterns of the user: i.e., when they are online, how many times they have visited a web page, and how much time they spent there before they browse to a different website.

At this point, it is important to note that by reporting the full URL, the remote server has access not only to the visited domain, but also the full list of URL parameters, and thus, the exact content the user consumes (and in some cases, even what actions they performed on the visited website). As an example, for a user browsing a video on YouTube, the remote server learns not only that they visited YouTube, but also the specific video they browsed and when this visit happened. It is easy to anticipate that such a granular access to the user's history leaks interests, sexual preferences, political beliefs, etc.

What about other browsers? While for Yandex there were past reports [12] indicating potential mishandling of personal data (such as device, network and IP address), the users do not seem to be able to browse without being tracked on other browsers as well. Indeed, as a next step, we analyze the network traffic that all 15 browser apps in our dataset generate. Surprisingly, we see that apart from Yandex, the browsers of QQ and UC International also send the entire visited URL via "phone home" requests, including the full path and the query parameters, thus leaking the full content the user is browsing. Note that, contrary to the other two browsers, UC International does not leak the browser history by issuing native requests to the remote server. Instead, it injects an (obfuscated) JavaScript snippet into every web page the user browses. In the generated requests, UC International leaks to the remote server the user's city-level geolocation and ISP. In addition, we see Edge and Opera browsers reporting every single visited domain to Bing API and Opera Sitecheck (Opera's anti-phishing service), respectively. In addition, 8 out of all 15 mobile browsers in our dataset query Cloudflare's or Google's third-party DNS-over-HTTPS services for the visited domains with the rest (7) of them using the device's local DNS stub resolver.

Incognito mode: Next, we investigate if a privacy-conscious user has the option to stop browsers from leaking their browsing history by browsing in incognito mode. Specifically, we use Panoptes with the incognito modes of Edge, UC International and Opera to crawl a subset of the websites in our dataset[5]. We find that these browsers that leak the browsing history of their users, continue to do so, no matter what mode the user is browsing on. This highlights a gap between what the user expects to happen when they open a browser in incognito mode and what happens in reality.

Reporting visits to sensitive content: It has been shown in the past [27] that sharing sensitive information may lead to the exposure of the user's political beliefs, sexual preferences, etc., and thus, to targeted attacks (e.g., smear campaigns, de-anonymization on social media [28], blackmailing, spear phising [29], etc.). Therefore, as a next step, we set out to explore if the browsers that share the browsing history of the user with remote servers, perform any sort of filtering locally to avoid leaking potential user visits to sensitive content (as per the sensitive categories that Google Ads block [30]). Specifically, we use the Panoptes framework to crawl websites dealing with content from these sensitive categories (i.e., religion, sexual preferences, political beliefs and health). We find that these same browsers (i.e., Yandex, UC International and QQ) continue to leak the entire URL the users visits, and thus, the exact content they browse.

3.3 Leaking PII and device identifiers

Next, we use keyword matching (via regex) and heuristics to extract potential Personally Identifying Information (PII) and device-specific information the browsers may leak via the URL parameters of the natively generated requests. We exclude the Android version

[5]Unfortunately the rest of the browsers that leaked users' browsing history (Yandex and QQ) do not provide an incognito mode.

Table 2: Personally Identifying Information (PII) and device-specific information leaked by the various browser apps. Connection type can be Metered or Unmetered while the Network type can be WiFi or Cellular.

Browser	Device Type	Device Manuf.	Timezone	Resolution	Local IP	DPI	Rooted Status	Locale	Country	Location (lat & long)	Connection Type	Network Type
Chrome	No	No	No	No	No	No	No	No	No	No	No	No
Edge	No	Yes	Yes	Yes	No	No	No	Yes	No	No	Yes	Yes
Opera	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	No	Yes
Vivaldi	No	No	No	Yes	No	No	No	No	No	No	No	No
Yandex	Yes	Yes	No	Yes	No	Yes	No	Yes	No	No	No	Yes
Brave	No	No	No	No	No	No	No	No	No	No	No	No
Samsung	No	No	No	No	No	No	No	Yes	No	No	No	No
DuckDuckGo	No	No	No	No	No	No	No	No	No	No	No	No
Dolphin	No	No	No	No	No	No	No	No	No	No	No	No
Whale	No	No	No	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes
Mint	No	No	Yes	Yes	No	No	No	Yes	Yes	No	No	No
Kiwi	No	No	No	No	No	No	No	No	No	No	No	No
CocCoc	Yes	Yes	No	Yes	No	No	No	Yes	Yes	No	No	No
QQ	Yes	Yes	No	Yes	No	No	No	No	No	No	No	No
UC Int.	No	No	No	No	No	No	No	Yes	No	No	No	Yes

and the device model from our analysis, as such information is reported by default for compatibility purposes by all vendors through the HTTP User-Agent header.

In Table 2, we summarize our findings. Interestingly, we see the Whale browser leaking the local IP along with the rooted status of the device, its network type and the country-level geolocation of the user. Similar geolocation information is shared also by Mint, CocCoc and Opera browsers, with the latter browser sharing also the longitude and latitude coordinates of the user. It is important to note that the information leaked by Opera and QQ is not shared with their vendors, but with ad servers, as described earlier in Section 3.1. An example of such a native request along with the information shared can be seen in Listing 1.

3.4 International data transfers

The General Data Protection Regulation (GDPR) imposes restrictions on the transfer of personal data outside the European Union (EU), to third-party countries or international organizations [31]. To understand where the remote servers (the browsers communicate with when they "phone home") are located, we follow past approaches [32]. In particular, we extract the IP address of every remote server receiving native requests from the tested browsers, and use a popular IP-to-geolocation service [33] to extract its country-level location. We see that while the crawls took place from EU, in case of the mobile browsers Yandex, QQ and UC International which leak in full detail the browsing history of the users, the requests are being received by servers located in Russia, China, and Canada, respectively. This finding exposes a misalignment with the users' expectation regarding online privacy [3, 34], (especially in incognito or "private" mode). Indeed, it is not clear that users are fully aware that their data may end up in countries outside the EU.

3.5 Phoning home when idle

Finally, we examine the native network activity of all browsers in our dataset during idle time. Using Panoptes, we launch each browser and leave them idle (at the start page) without any user interaction for 10 minutes, while monitoring their network traffic. As we can see in Figure 5, even though there is no web activity, the majority of the mobile browsers have frequent communication with their associated remote servers. This communication grows exponentially within the first minute (mostly to update favicons, thumbnails and DNS entries of the websites in the start page), before they reach a relative plateau where they "phone home" to query for updates and send telemetry and analytics). Opera follows a linear growth because of the frequently updated Opera News feed appearing in the starting page.

Yet, we see cases like Dolphin and Mint sending 46% and 8%, respectively, of their native requests to Facebook's Graph APIs. Similarly, CocCoc sends 6.7% of its native request to adjust.com advertisers, whereas Edge is quite active communicating with various different domains, including the second parties msn, microsoft.com, bing.com, but also the third party advertisers and analytics of adjust.com, outbrain.com, zemanta.com and scorecardresearch.com. Finally, we see Opera sending 21.9% and 1.7% of its native requests to doubleclick.net and appsflyers.com, respectively.

4 RELATED WORK

Similar to our study, in [35] authors investigate the "phone home" functionality of the 6 most popular browsers along with its privacy implications. Specifically, they capture the traffic of the browser's web engine and analyze the device and user IDs leaked. Contrary to our work, authors focus on identifiers leaked during specific operations (like browser startup, search auto-complete and update of browser extensions), and show how these identifiers can be used to identify browser instances. In this work, we follow an automated and scalable approach using the Panoptes framework, contrary to the manual analysis of network traffic performed in [35]. Also, we discover that the user's browsing history is explicitly leaked by the browsers, and not because of the auto-complete feature.

Concurrently with our work, in [36] the authors studied the privacy implications of over 400 Android browsing applications, and found that some of these apps not only contain advertising and tracking libraries, but they also leak personal information to third parties. In this study, we consciously select a smaller set of popular browsers that account for 75% of the mobile browser market share worldwide [37]. Our approach allows us to dive deeper in investigating aspects that play a major role in understanding the extent of the problem. First, we perform our study from an EU vantage point (where GDPR is in effect) and show that important and private user

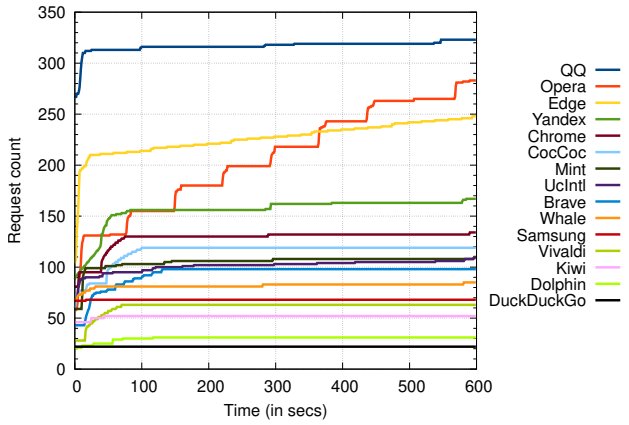

Figure 5: Timeline of native requests while the browser is idle for 10 minutes. The activity of most browsers grows exponentially within the first minute (mostly to update favicons, thumbnails and DNS entries of the websites in the start page), before they reach to a relative plateau. where they "phone home" to query for updates, send telemetry and analytics.

data, such as access to medical sites, is sent to servers outside the EU. Second, we investigate both regular browsing and incognito mode, and show that even in incognito mode some browsers leak the browsing history of their users, exposing the false sense of privacy that incognito mode may give to the users. This highlights the discrepancy between what the users expect to happen when they browse "anonymously" (i.e., incognito mode), and what mobile browsers actually do. Finally, we study separately domain name leaking and full path leaking. Indeed, the latter is more significant as it allows Web entities to have a clear understanding of their users' preferences and actions.

In [38] authors study the native requests of Safe Browsing functionality of Google Chrome and Yandex. They show that Safe Browsing can potentially be used as a tool to track specific classes of individuals. Additionally, they show that data included in Google and Yandex Safe Browsing provide a concrete set of URLs/domains that can be re-identified without much effort. In [39] authors analyzed private browsing modes in modern browsers and discussed their success at achieving the desired security goals. They point out several weaknesses in existing implementations, with the most severe ones enabling a local attacker to completely defeat the benefits of private mode. In [40] authors propose a new whitelist-based browsing mode that aims to act as a middle-ground between regular browsing and private mode, thus, enabling users to get the best of both worlds: the privacy of private mode, along with the convenience of the regular browsing mode.

Countermeasures: Considering that the user tracking performed by the browser app takes place natively, and gets leaked to remote servers via network requests of the browser app itself, traditional tracker/ad-blocking extensions cannot constitute a useful countermeasure. In [41] authors propose NoMoAds which leverages the mobile device's network interface as a universal vantage point to intercept, inspect, and block outgoing packets from mobile apps. NoMoAds extracts features from packet headers and/or payload to train machine learning classifiers for detecting native ad requests. Similarly, in [42] authors propose a cross-platform system (ReCon) that inspects apps' native traffic and leverages machine learning to reveal potential PII leaks, thus, giving mobile users control of what is shared with remote servers. In [43] authors study and compare the corresponding privacy implications when the user accesses an online service via the web, or via its dedicated mobile (native) app. Finally, they propose an anti-tracking mechanism to intercept and block third party native requests of mobile apps on the OS level based on custom filterlists.

5 SUMMARY AND CONCLUSION

In this work, we study the privacy implications of mobile browser apps and investigate whether users can truly browse the Web in private, even if they do block tracking requests of the web engine. To achieve this, we develop Panoptes, a framework to instrument instances of Android browser applications and monitor separately the mobile browser traffic which is generated by (i) the web engine and (ii) natively by the mobile browser app.

We use Panoptes to study 15 of the most popular browser applications and we collect data by crawling 1000 websites. As a summary of our findings, in this study we:

(1) Analyse the amount of native requests the mobile browsers generate, and find that it can amount to as high as 1/3 of the total generated traffic.
(2) See that when Yandex, QQ and UC International browsers "phone home", they report to their remote servers the exact page and content the user is browsing at real time.
(3) Find that Yandex, in particular, does this reporting together with a persistent identifier, so that users can be tracked even if they use Tor, an anonymous proxy, or a VPN.
(4) Confirm that the browser history leaking behavior happens (i) even when the users are browsing in incognito mode, or (ii) even when the users browse content of sensitive categories related to religion, sexual or political preferences, health issues etc.
(5) Find that, in some browser cases, these browsing history leaks are transmitted to servers located outside EU.
(6) See Opera, CocCoc, Dolphin and Mint browsers communicating with third-party ad and analytics servers, while they are also leaking PII and device-specific identifiers.

6 ACKNOWLEDGEMENTS

This project has received support from the European Union's Horizon 2020 Research and Innovation program under the CONCORDIA project (Grant Agreement No. 830927) and SPATIAL project (Grant Agreement No. 101021808). The authors bear the sole responsibility for the content presented in this paper, and any interpretations or conclusions drawn from it do not reflect the official position of the European Union nor the Research Innovation Foundation.

REFERENCES

[1] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, CCS'16, 2016.

[2] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *The World Wide Web Conference*, WWW'19, 2019.

[3] Emmanouil Papadogiannakis, Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. User tracking in the post-cookie era: How websites bypass gdpr consent to track users. In *Proceedings of the Web Conference 2021*, WWW '21, page 2130–2141, 2021.

[4] Peter Snyder, Soroush Karami, Arthur Edelstein, Benjamin Livshits, and Hamed Haddadi. Pool-party: Exploiting browser resource pools as side-channels for web tracking. *arXiv preprint arXiv:2112.06324*, 2021.

[5] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. Exclusive: How the (synced) cookie monster breached my encrypted vpn session. In *Proceedings of the 11th European Workshop on Systems Security*, pages 1–6, 2018.

[6] Google Developer. Safe browsing apis (v4). https://developers.google.com/safe-browsing/v4, 2023.

[7] Sandra Siby, Umar Iqbal, Steven Englehardt, Zubair Shafiq, and Carmela Troncoso. {WebGraph}: Capturing advertising and tracking information flows for robust blocking. In *31st USENIX Security Symposium*, USENIX Sec'22, 2022.

[8] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy*, SP'21, 2021.

[9] Josh Howarth. Internet traffic from mobile devices (apr 2023). https://explodingtopics.com/blog/mobile-internet-traffic, 2023.

[10] Tor Project. Tor project. https://www.torproject.org/, 2023.

[11] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Benjamin Livshits, and Alexandros Kapravelos. Towards realistic and reproducibleweb crawl measurements. In *Proceedings of the Web Conference 2021*, WWW '21, page 80–91, 2021.

[12] Patrick McGee. Russian tech giant yandex's data harvesting raises security concerns. https://www.ft.com/content/c02083b5-8a0a-48e5-b850-831a3e6406bb, 2022.

[13] Appium. http://appium.io/docs/en/2.0/, 2023.

[14] Frida. https://frida.re/, 2023.

[15] Curlie. Chrome developers. https://chromedevtools.github.io/devtools-protocol/, 2023.

[16] Mitmproxy Project. mitmproxy - an interactive https proxy. https://mitmproxy.org/, 2023.

[17] Android Developers. Process. https://developer.android.com/reference/android/os/Process.html#myUid%28%29, 2023.

[18] statcounter. Top mobile browsers. https://gs.statcounter.com/browser-market-share/mobile, 2023.

[19] Firefox. Remote protocols. https://firefox-source-docs.mozilla.org/remote/index.html, 2023.

[20] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019.

[21] Curlie. The collector of urls. https://curlie.org/, 2023.

[22] John Pegioudis. Website list. https://github.com/BanForFun/panoptes-results/blob/master/1k.txt, 2023. Accessed on September 18th, 2023.

[23] Jiaping Gui, Stuart Mcilroy, Meiyappan Nagappan, and William G. J. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 100–110, 2015.

[24] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. The cost of digital advertisement: Comparing user and advertiser views. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1479–1489, 2018.

[25] Steven Black. Adware & malware hosts. https://github.com/StevenBlack/hosts, 2023.

[26] Coc Coc Browser. The best browser with adblocker. https://coccoc.com/en/chan-quang-cao, 2023.

[27] Artur Janc and Lukasz Olejnik. Web browser history detection as a real-world privacy threat. In *Computer Security – ESORICS 2010*, pages 215–231, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[28] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy*, pages 223–238, 2010.

[29] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 581–590, New York, NY, USA, 2006. Association for Computing Machinery.

[30] Google Ad Manager. Block sensitive categories. https://support.google.com/admanager/answer/2541069, 2023.

[31] Data Protection Commision. Transfers of personal data to third countries or international organisations. https://www.dataprotection.ie/en/organisations/international-transfers/transfers-personal-data-third-countries-or-international-organisations, 2018.

[32] Costas Iordanou, Georgios Smaragdakis, Ingmar Poese, and Nikolaos Laoutaris. Tracing cross border web tracking. In *Proceedings of the internet measurement conference 2018*, pages 329–342, 2018.

[33] Brand Media, Inc. Ip address lookup | geolocation. www.iplocation.net, 2023.

[34] Célestin Matte, Nataliia Bielova, and Cristiana Santos. Do cookie banners respect my choice?: Measuring legal compliance of banners from iab europe's transparency and consent framework. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 791–809. IEEE, 2020.

[35] Douglas J. Leith. Web browser privacy: What do browsers say when they phone home? *IEEE Access*, 9:41615–41627, 2021.

[36] Amogh Pradeep, Alvaro Feal, Julien Gamba, Ashwin Rao, Martina Lindorfer, Narseo Vallina-Rodriguez, and David Choffnes. Not your average app: A large-scale privacy analysis of android browsers. *Proceedings on Privacy Enhancing Technologies*, 1:29–46, 2023.

[37] StatsCounter. Mobile browser market share worldwide. https://gs.statcounter.com/browser-market-share/mobile/worldwide, 2023.

[38] Thomas Gerbet, Amrit Kumar, and Cédric Lauradoux. A privacy analysis of google and yandex safe browsing. *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 347–358, 2016.

[39] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. An analysis of private browsing modes in modern browsers. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, USENIX Security'10, page 6. USENIX Association, 2010.

[40] John Korniotakis, Panagiotis Papadopoulos, and Evangelos P Markatos. Beyond black and white: Combining the benefits of regular and incognito browsing modes. In *17th International Conference on Security and Cryptography*, SECRYPT'20, 2020.

[41] A. Shuba, A. Markopoulou, and Z. Shafiq. Nomoads: Effective and efficient cross-app mobile ad-blocking. In *Proceedings of the Privacy Enhancing Technologies Symposium*, PETS'18, 2018.

[42] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 361–374, 2016.

[43] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, 2014.

[44] Erin Kenneally and David Dittrich. The menlo report: Ethical principles guiding information and communication technology research. *Available at SSRN 2445102*, 2012.

[45] Caitlin M. Rivers and Bryan L. Lewis. Ethical research standards in a world of big. *F1000Research*, 3, 2014.

A ETHICAL CONSIDERATIONS

The execution of this work has followed the principles and guidelines of how to perform ethical information research and use of shared measurement data [44, 45]. Hence, we keep our crawling to a minimum to ensure that we do not slow down or deteriorate the performance of the visited web services in any way, and make considerable effort not to overwhelm the hosting servers. As a result, we crawl only the landing page of each website and visit it only once. We do not interact with any component inside a website, and only passively observe network traffic. In addition to this, Panoptes has been implemented to wait for both the website to fully load and an extra period of time before visiting another website. Consequently, we emulate the behavior of a normal user that landed on a website. In accordance to the GDPR and ePrivacy regulations, we did not engage in collection of data from real users.