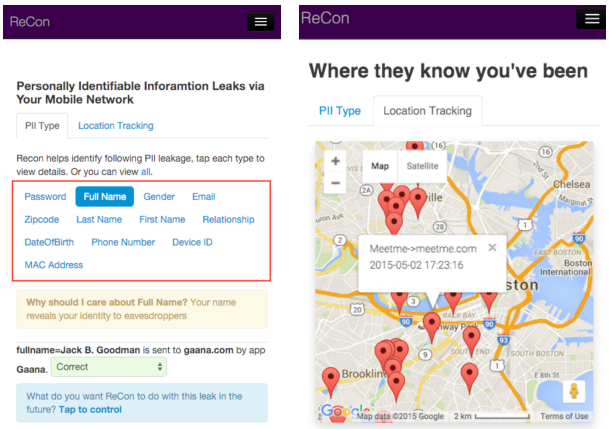
Rao A, Kakhi A, Razaghpanah A, Li A, Choffnes D, Legout A, Mislove A, Gill P. Meddle: Enabling Transparency and Control for Mobile Internet Traffic. *Technology Science*. 2015103003. October 30, 2015. http://techscience.org/a/2015103003/

# Meddle: Enabling Transparency and Control for Mobile Internet Traffic

Ashwin Rao, Arash Molavi Kakhki, Abbas Razaghpanah, Anke Li, David Choffnes, Arnaud Legout, Alan Mislove, and Phillipa Gill

## Highlights

- We built Meddle, which redirects a mobile device's internet traffic to a VPN proxy that we use monitor privacy leaks from apps and traffic differentiation by ISPs
- Testing 309 popular apps, we found that 21% of Android apps leaked Device IDs, and 6% of iOS apps leaked email addresses in unencrypted plaintext
- We found 6 popular iOS apps and 1 Android app leaking passwords in plaintext vulnerable to capture by attackers
- We found 3 US mobile ISPs (BlackWireless, H2O, and SimpleMobile) in early 2015 reduced data-transfer speeds to devices on their networks by up to 65% for connections to YouTube, and sometimes for Netflix and Spotify as well
- We found one ISP in China injecting ads into the Internet traffic of the devices on their network




*Screen captures of a tool built by the authors to display how users' personal information and location leaks over a network*

## Abstract

Mobile devices such as smartphones and tablets have fundamentally changed the way we interact with the Internet—and each other—in many positive ways. Underlying this enormous success are several core challenges that remain difficult to address. Apps track users and leak their personal data; the network performance and neutrality of mobile Internet service providers (ISPs) are generally unknown; and apps inefficiently use available networking resources, leading to suboptimal network performance and energy consumption. Addressing these problems requires not only visibility into the traffic generated by devices, but also control over how, when, and where that traffic is sent to and handled by third parties. Previous approaches to address these problems, such as TaintDroid [21], Glasnost [19], and performance-enhancing proxies, [55] improve visibility and control, but each faces limitations that hamper its effectiveness.

With Meddle, we explore a simpler and more effective strategy to address these problems: using network redirection to improve visibility and control for network traffic from mobile devices. Specifically, we use natively supported OS features (namely, VPN connectivity) to redirect a device's Internet traffic over a secure channel to a trusted server. We developed new systems running atop this server to characterize and control network traffic using controlled and *in situ* studies. Our research builds upon this platform to improve privacy, policy transparency and performance in the mobile environment. We present summary results from our experience using this tool to reveal private information leaked in network traffic from mobile devices. We then show how to reveal mobile ISP performance and policies using Meddle as an in-path vantage point located outside mobile networks.

**Results summary:** When we made mobile network traffic transparent using Meddle, we identified extensive leaks of users' personally identifiable information (PII). This included "run of the mill" tracking of users via unique identifiers and locations, and also the collection of highly sensitive information such as user credentials and contact information. As part of related work, we have developed ways to automatically detect and block/modify this information according to user preferences. We also used Meddle to understand ISP policies and their impact on network performance in the mobile environment. We identified cases of traffic differentiation, content modification, and ad injection. We are opening our system to the public to allow average users to contribute measurements that help monitor policies, and we will publish our results to inform users, researchers, and policy makers.

## Introduction

There has been a dramatic shift toward using mobile devices such as smartphones and tablets as the primary way to access Internet services. Unlike their fixed-line counterparts, these devices offer ubiquitous mobile connectivity via Wi-Fi and cellular data access, and they include a wide array of sensors (e.g., GPS, camera, and microphone). Mobile devices have

fundamentally changed how we interact with the Internet—and each other—in numerous positive ways.

To ensure the continued success of these mobile systems, device operating systems and Internet service providers (ISPs) maintain closed and opaque systems, in part to protect mobile-device users and the network from harm. For example, mobile OSes allow users to install software only from centrally curated stores by default, and mobile carriers block, shape, or modify traffic to prevent any single device from unfairly consuming scarce cellular network resources [25, 53].

However, this closed and opaque environment results in significant collateral damage: users, researchers and policymakers struggle to protect privacy, promote policy transparency and optimize performance. Apps extensively track users and leak their personal information [13, 20, 30], and users are either generally unaware or unable to stop them [18, 31]. Mobile carriers interfere with traffic [37] (e.g., blocking or changing content), and regulators lack the means to hold them accountable [25]. Devices such as proxies and transcoders can optimize performance [3, 22, 54], but researchers cannot explore their potential *in situ* because they require privileged access to carrier networks.

Previous attempts to address these issues faced challenges because of a lack of visibility into network traffic generated by mobile devices and the inability of the devices to control their traffic. Passively gathered datasets from large mobile ISPs [27, 52] provide visibility, but the datasets and the ISPs gathering them give researchers no control over network flows (e.g., to experiment with new proxy designs). Likewise, custom Android extensions provide control over network flows, but measurement visibility is limited to the devices running these custom OSes or apps [15, 21], often requiring warranty-voiding "jailbreaking."

The Meddle project explored a simpler, more effective strategy [48, 49]: using indirection to improve visibility and control for mobile network traffic. Meddle allows us to use natively supported OS features to redirect a device's Internet traffic to a proxy server, and to develop new systems running atop this server to characterize and control network traffic. The approach is called Meddle because it uses a middlebox to "meddle" with mobile traffic.

Our Meddle system allows us to explore the potential of new in-network devices without needing privileged access to a mobile ISP; rather, we use software middleboxes that can run virtually anywhere. By operating at the network layer (the "thin waist" of a computer network), Meddle is resilient to the rapid changes in today's mobile network infrastructure and mobile-device software. Further, it provides a practical means to experiment with new protocols and system designs for preserving privacy by revealing ISP performance and policies, and by enabling new services atop in-network devices. This solution does not require rooting (acquiring root access to) devices or deploying hardware, so it is immediately deployable globally. Our prototype currently runs on cloud virtual machines (VMs) worldwide, and it supports deployments on dedicated servers and devices in home or enterprise networks.

## Background

The Meddle project addresses a growing problem in today's mobile Internet systems: a lack of transparency or control over how devices interact with the Internet. Device OSes limit users to installing apps only from a curated store subject to unilateral policies for inclusion [9, 10]. Mobile carriers can and do manipulate network traffic [54], often without user awareness or consent.

Sometimes these policies have positive purposes, i.e., protecting the users who install apps and use cellular networks. For example, app stores control which apps meet reliability, security, and other policy goals. [9, 10] Mobile providers interfere with network traffic to ensure reliable and fair access to constrained resources in cellular networks [25].

However, these policies create significant collateral damage. Previous studies identified privacy, [21, 52] policy, [53] and performance [16, 27, 47] issues in mobile systems. In the paragraphs below, we describe the costs of limited transparency and control.

### Privacy

Threats to our online privacy are pervasive and exacerbated by the vast amount of information readily available on mobile devices. Recent studies show that third parties track most of the time we spend online, and the apps we use leak personally identifiable information (e.g., location, passwords, and phone numbers) over the Internet without our knowledge. [4, 44, 51] Trackers are software libraries that gather information about users' Web and app usage. [29] Personally identifiable information (PII) is a generic term referring to "information which can be used to distinguish or trace an individual's identity." [34] This can include geographic locations, unique identifiers, phone numbers and other similar data. Tracking and leaking PII are not mutually exclusive. In this paper we consider both to be privacy leaks.

Several methods systematically identify privacy leaks from mobile devices and develop defenses against them. Dynamic taint tracking modifies the device OS to track access to PII at runtime [21] using dynamic information flow analysis. This ensures flagging of all access to PII tracked by the OS; however, it can result in large false positive rates (due to coarse-granularity tainting) and false negatives (e.g., when the OS does not store PII such as passwords), and can incur significant runtime overheads that discourage widespread use. Alternatively, static analysis (e.g., using data-flow analysis or symbolic execution) determines *a priori* whether an app leaks PII. [12, 20, 33, 57] This approach does not suffer runtime overhead, but state-of-the-art tools suffer from imprecision, [14] and symbolic execution can be too time-intensive to be practical. Static analysis is limited by obfuscation (a program may be written in a way to avoid detection of PII leaks), and it also does not handle reflection and dynamically loaded code. [60] A recent study [39] finds dynamically loaded code is common, comprising almost 30% of goodware app code loaded at runtime.

These approaches, and follow-up work that extends them, [11, 28, 32, 36, 40, 56, 59] can improve mobile privacy. But they depend on a deployment model that restricts their impact to custom OSes or app stores. Privacy leakage, however, is an evolving target affecting all users and all platforms. Importantly, we note that if a leak occurs, it probably uses the network to do so. Studies using network traces gathered inside a mobile network [29] and in a lab setting [38] identify significant tracking, despite not having access to software instrumentation. Our work builds on these observations to both identify violations of a user's privacy and control privacy leaks regardless of the device OS or network being used.

### Policy Transparency

A key factor enabling the Internet's success is that the network is neutral with respect to the packets it carries. [17] That is to say, networks do not discriminate against or otherwise alter certain types of traffic except for lawful purposes. Under network neutrality, applications not considered in the original Internet can flourish with minimal friction from network providers, even if they compete with the provider's lines of business. There is active discussion on whether to enforce neutrality, and recent FCC rules [24] bar all ISPs from commercially unreasonable practices against Internet traffic. However, there is currently no way to monitor network neutrality in mobile networks, either to inform regulators or enforce policies. In recent work, [55] we found that all US mobile carriers proxy HTTP traffic, and some modify HTTP requests, including transcoding images to reduce size/quality. Our Mobilyzer project [6] and the FCC [23] measure mobile broadband performance but do not detect ISP policies that selectively affect traffic. Existing approaches to revealing these policies in fixed-line networks [19, 41] are limited by mobile-OS constraints on network usage and often require rooting a device to conduct measurements. We will show how we can use traffic redirection as a platform for building tools that improve transparency by reliably detecting network policies in mobile carriers.

### Implications

Each of the existing solutions to address privacy, policy transparency, and performance in mobile networks is limited at least in part by visibility and control over network traffic. To understand and address these problems as they evolve over time, we need an approach that supports the ability to measure and modify network flows from mobile devices. The solutions need to be easy to deploy for a typical smartphone user running any operating system worldwide. The next section proposes an approach that achieves these goals and enables new research in mobile networks.

## Methods

Existing solutions to address privacy, policy transparency, and performance in mobile systems do not have visibility into network flows from mobile devices, the ability to modify them, and/or a deployment model that facilitates incremental, large-scale adoption to ensure broad impact. At first glance, addressing all these limitations seems to impose a high barrier

to success, as it may require custom OSes and/or privileged access to mobile carrier networks.

The key insight that enables our research is that we can in fact achieve these goals today without requiring any privileged access to networks or OS modifications. We achieve network traffic visibility through redirection, i.e., sending all device traffic to a proxy server using native support for virtual private network (VPN) tunnels. Once traffic arrives at the proxy server, we use software middleboxes to enable users and researchers to exert control over mobile-device traffic.

Meddle

Meddle uses a VPN to direct a participating mobile device's Internet traffic to a proxy server (Fig. 1, top). The vast majority of mobile devices support VPNs natively, typically to satisfy enterprise clients. We currently support iOS, Android and Windows phones.

Our Meddle proxy sends traffic to a software middlebox (Fig. 1, middle) that can record and modify mobile-device flows. Meddle supports a plugin infrastructure for custom flow processing. Each plugin takes as input a network flow and outputs a network flow (potentially modified or empty). When a packet arrives at Meddle, a software-defined switch [7] determines the ordered set of plugins that the corresponding flow will traverse. Plugins may implement a variety of features such as analyzing PII leakage, optimizing page speed, or blocking connections.

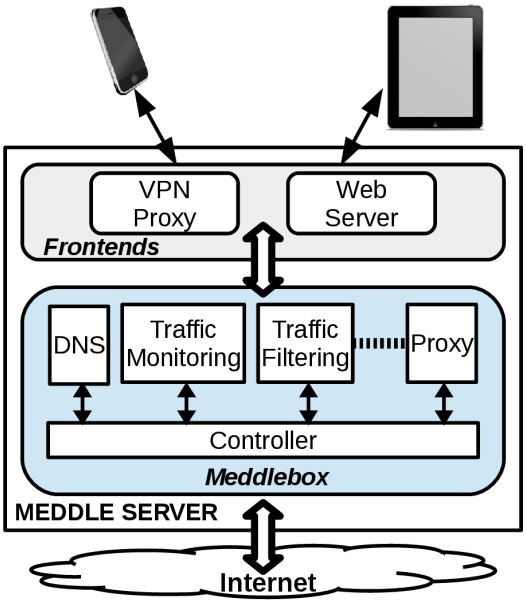



**Figure 1. Meddle architecture. Mobile devices (top) communicate with a Meddle front end (VPN proxy, Web proxy and/or traffic replay server). VPN proxy traffic is forwarded to software middlebox services that measure and/or interpose on network flows before relaying the traffic to the Internet.**

## Deployment models

Meddle can be deployed as (1) a research experimentation platform for controlled experiments (without human subjects), (2) an *in situ* observatory for crowdsourced measurement experiments with human subjects, or (3) a free, open source, stand-alone service for users wanting to benefit from improved transparency and performance. Our work used Meddle (1) to develop and prototype our research ideas, (2) to evaluate their effectiveness "in the wild" and (3) to bring our research products to a broad audience.

Meddle is easy enough to install that even non-expert users can run it. A user configures a VPN on iOS by opening a file and on Android by filling out five fields. We host a cloud-based deployment that is free for users (partially supported by an Amazon Research Grant) to support large numbers of flows for *in situ* experimentation. The initial prototype has reasonably low overheads (a 5- to 15-millisecond extra delay in the US, and a 1–6% increase in power consumption per day [42]). We propose to investigate ways to further reduce these overheads.

Meddle has been deployed with both physical servers and VMs in the US, France, and China. We intend to make the Meddle software publicly available. Those who prefer to run their own Meddle instance (e.g., if they do not want to participate in our study) can deploy Meddle on a physical device such as Raspberry Pi plugged into a home router, a dedicated server in an enterprise, and/or a VM in the cloud. Further, client traffic can be selectively redirected to different Meddle instances (or not at all) depending on the performance and privacy implications. For example, each flow can be redirected to a Meddle instance that minimizes delay between a user's device and destination for performance reasons, and some traffic may not be redirected at all. As a distributed service, Meddle mitigates the impact of any single point of failure. When Meddle is unavailable, the system simply sends traffic directly (unproxied).

## Dataset used in this study

Using Meddle, we collected full packet traces from Internet activity generated by mobile devices. We used this data to study how to map monitored traffic to applications, and to analyze PII leakage. Below, we describe our data-collection methodology, which consists of (1) controlled experiments in a lab setting and (2) IRB-approved "in the wild" measurements gathered from real users during seven months.

**Controlled experiments with apps.** Our goal with controlled experiments was (1) to obtain ground truth information about network flows generated by apps and devices, and (2) characterize the network activity for a large variety of apps in a lab setting. We used this data to understand how apps leak PII in network flows.

**Device setup.** We conducted our controlled experiments using two Android devices (running Android 4.0 and 4.2) and an iPhone running iOS 6. We started each set of controlled

experiments with a factory reset of the device to ensure that software installed by previous experiments could not impact the network traffic generated by each device. Then we connected the device to Meddle and began the experiment.

**SSL bumping.** We used SSL bumping only in controlled experiments where no user traffic is intercepted.

**Manual tests.** We manually tested the 100 most popular free Android apps in the Google Play store and 209 iOS apps from the iOS App store on April 4, 2013. For each app, we installed it, entered user credentials if relevant, interacted with it for up to 10 minutes, and uninstalled it. This allowed us to characterize real user interactions with popular apps in a controlled environment. We entered unique and distinguishable user credentials when interacting with apps to easily extract the corresponding PII from network flows (if the PII are not obfuscated).

**Automated tests.** The second set of controlled experiments consisted of fully automated experiments on 732 Android apps from a free, third-party Android market, AppsApk.com [1]. We performed this test because Android users can install third-party apps without rooting their device.

Our goal was to understand how these apps differ from those in the standard Google Play store, as they are not subject to Google Play restrictions. We automate experiments using adb to install each app, connected the device to the Meddle platform, and startd the app. Then we used Monkey [8], an app-scripting tool, to perform a series of approximately 100,000 actions that included random swipes, touches, and text entries. Finally, we use adb to uninstall the app and rebooted the device to forcibly end any lingering connections. We limited this set of experiments to Android devices because iOS does not provide equivalent scripting functionality.

***In situ* study.** The controlled experiments described above provided information for a large number of apps running in a controlled setting for a short period of time. To understand the network behavior of devices with real users “in the wild” over longer time periods, we conducted an IRB-approved measurement study with a small set of subjects, from Oct. 15, 2012 to Sep. 1, 2013. The measurement study is ongoing. We report here on a subset of results.

We collected measurement data from 26 devices: 10 iPhones, 4 iPads, 1 iPod Touch, and 11 Android phones. The Android devices in this dataset include the Nexus, Sony, Samsung, and Gsmart brands, while the iPhone devices include one iPhone 3GS, four iPhone 5, and five iPhone 4S. These devices belonged to 21 different users, volunteers for our IRB-approved study. This dataset, called mobWild, consists of 318 days with data; the number of days for each user varied from 5 to 315 with a median of 35 days. For privacy reasons, the SSL-Bumping plugin was disabled for all measurements involving real users.

**Using Meddle to measure ISP policies.** We used Meddle in two ways to understand ISP policies affecting network traffic from mobile devices. First, we used Meddle to capture traffic from iOS, Android, and Windows Phone devices, without needing to root the device or modify the OS. This allowed us to identify traffic to use for testing ISP policies. We use six apps for this: YouTube, Netflix, Spotify, Hangout, Skype, and Viber. We report results from the first three because our experiments did not reveal any differentiation against the last three. Second, we used Meddle to selectively send traffic over encrypted (VPN) and unencrypted channels to measure how ISP behavior changes with and without access to plaintext content.

## Results

Here, we summarize results from using Meddle to identify PII in mobile network flows and reveal ISP policies. Leaks and policies change over time, so that what we present is only a snapshot. Our goal is to continuously monitor changes using Meddle and make our results publicly available to users, researchers, and policy makers.

### Privacy leaks

The two key questions we address are (1) can we observe PII leaked in network traffic and (2) can we develop techniques to automatically detect and block/modify it? We addressed these questions while building a system called ReCon, currently available to the public at http://recon.meddle.mobi (see Fig. 2). For details about this project, please see our technical report [43].

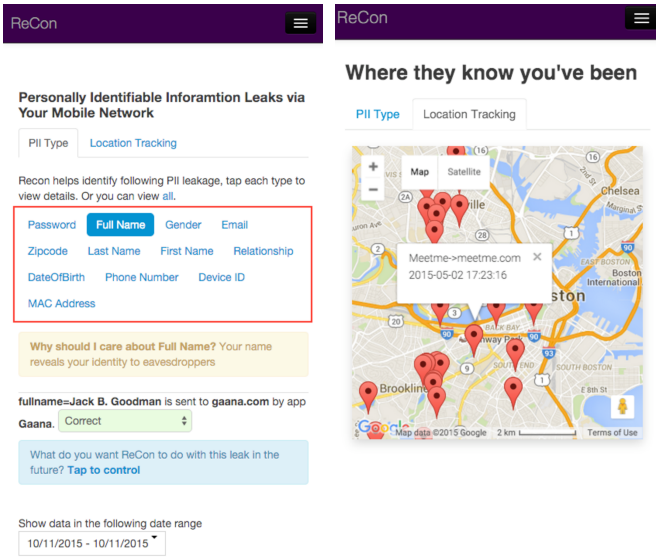



**Figure 2. Screen captures of ReCon (not from a real user). ReCon displays how users' PII leaks over the network, supports filters to modify them, and gathers user feedback providing ground-truth labels for classification.**

Here, we summarize results regarding how apps leak PII when we know the contents *a priori* based on a study that predates the technical report. Specifically, we investigated how PII leaks in plaintext by using network traces collected from apps in 2013 and 2014. We manually tested the 100 most popular free Android apps in the Google Play store and 209 iOS apps from the iOS App Store. We entered unique and distinguishable user credentials when interacting with apps to easily extract the corresponding PII from network flows (if they are not obfuscated). We also conducted controlled, fully automated experiments on 922 Android apps obtained from a free, third-party Android market, AppsApk.com [1]. (Note: The ReCon technical report [43] had results from a 2015 dataset containing new controlled experiments and a different user study.)

We used the traffic traces from our controlled experiments to identify how apps leak PII. For our controlled experiments, we created dummy user accounts with conspicuous, fake contact information. Our goal was to detect if any PII stored on the device leaks over HTTP/S. For our analysis, we focused on the email address, location, username and password used during authentication, device ID, contact information, and the IMEI number. Normal app operations require some of this information; however, the information should never travel across the network in plaintext.

| Store | Platform | # Apps | Email | Location | Name | Password | Device ID | Contacts | IMEI |
|---|---|---|---|---|---|---|---|---|---|
| **App Store** | iPhone | 209 | 13 (6.2%) | 20 (9.5%) | 4 (1.9%) | 6 (2.87%) | 4 (1.9%) | 0 (0%) | 0 (0%) |
| **Google Play** | Android | 100 | 3 (3%) | 10 (10%) | 2 (2%) | 1 (1%) | 21 (21%) | 0 (0%) | 13 (13%) |
| **Third Party** | Android | 732 | 3 (0.4%) | 57 (7.8%) | 3 (0.4%) | 0 (0%) | 85 (11.6%) | 6 (0.8%) | 39 (5.3%) |

**Table 1. Summary of PII leaked in plaintext (HTTP) by Android and iPhone apps. The popular iOS apps tend to leak location information in the clear while Android apps leak IMEI number and Android ID in the clear.**

**PII leaks in plaintext.** Table 1 presents PII leaked by Android and iOS apps in our tests. The IMEI, a unique identifier tied to a mobile device, and the Android ID (tied to an Android installation) are the most frequently leaked PII by Android apps. These can be used to track and correlate a user's behavior across Web services. Table 1 shows that other information including contacts, emails, and passwords was also leaked in the clear. The email address used to sign up for the services was leaked in the clear by 13 iOS and 3 Android apps from our set of popular apps. While only one Android app (belonging to the Photography category) leaked a password in the clear, we were surprised to learn that six of the most popular iOS apps send user credentials in the clear, including the password.

**PII leaked from same apps on different OSes.** We observed that the information leaked by an app depends on the OS. Of the top hundred apps for iOS and Android, 26 are available on both iOS and Android. Of these 26 apps, 17 apps leaked PII on at least one OS, 12 apps leaked PIIs only on Android, 2 apps leaked PII only on iOS, while only one app had the same data leakage in both OSes. Of the remaining two apps that leaked PII, one app leaked the Android

ID and IMEI in Android and the username in iOS, while the other app leaked the Android ID in Android and the location in iOS. We posit that the difference in the PII leaks is primarily due to the different privileges that the underlying OS provides these apps (e.g., iOS does not provide access to IMEIs and does not support Android IDs), different implementations of advertising and analytics libraries, or different software development teams.

**PII leaked over SSL.** During our experiments, we observed that apps also sent PII over encrypted channels. We observed that 2 of the top 5 sites that receive PII over SSL are trackers. Our observations highlight the limitations of current mobile OSes with respect to controlling access to PII via app permissions. In particular, it is unlikely that users realize that they grant access to PII only for tracker libraries embedded in an app, even if those trackers (and the information they collect) are not essential to the app's functionality. This problem is pervasive: of the 77 sites that received some PII in the clear or over SSL during our controlled experiments, 35 are third-party trackers.

We note that our observations are a conservative estimate of PII leakage because we cannot detect PII leakage using obfuscation (e.g., via hashing). Regardless, our study shows that Meddle identifies significant PII leaks.

**PII leaked in user study.** We now analyze the PII leaks in the user study. Note that we do not decrypt SSL traffic in this study for privacy reasons.

**Location leaks.** We observe that a bus service app (One Bus Away), an app that manages the iOS homescreen (SpringBoard), and weather apps (TWC, Weather, and Hurricane) are responsible for more than 78% of the flows that send location in the clear. Other apps that do not require location, such as YouTube, Epicurious, and EditorsChoice, also leak the device location. Further, SpringBoard leaked location information for all 11 iOS devices in the user study. We observed that location was leaked up to 14 times per day for one device, sufficient to expose a user's daily movements to anyone tapping Internet connections [26].

**Unique ID leaks.** Apps frequently leak device ID and IMEI in the clear. As in the case of controlled experiments, trackers are the most popular destination for the IMEI leaks. Among the 16 sites that receive these unique IDs in the clear, 10 are trackers; the others include sites for games, news, and manufacturer updates.

| Tracker | Number of devices tracked | | |
|---|---|---|---|
| | Total | iOS | Android |
| doubleclick.net | 26 (all) | 15 (all) | 11 (all) |
| google-analytics.com | 26 (all) | 15 (all) | 11 (all) |
| googlesyndication.com | 22 | 12 | 10 |
| admob.com | 21 | 11 | 10 |
| scorecardresearch.com | 21 | 11 | 10 |

**Table 2. The top 5 trackers contacted by the devices in our dataset. All 26 devices in mobWild contacted doubleclick.net and google-analytics.com.**

In Table 2, we present the top 5 trackers ordered by the number of devices in our dataset that contact them. All the devices in our dataset contacted doubleclick.com, an ad site, and google-analytics.com, an analytics site.

ISP Policy Transparency

In the mobile environment, users have essentially no control over, or visibility into, how their mobile providers handle network traffic. As a result, users and policymakers cannot tell how mobile carriers apply network management policies and have no way to hold them accountable for unreasonable practices. In this section, we describe how to use Meddle to address this problem.

We focus on detecting traffic differentiation, which we define as any attempt to change the performance of network traffic traversing an ISP's boundaries. ISPs may implement differentiation policies for reasons that include load balancing or bandwidth management. Previous work [19, 41, 50, 58] explored this problem in limited settings.

The key challenges in the mobile environment not addressed by previous work are: (1) how to record representative traces in the mobile environment, (2) how to reliably tell when differentiation occurs for arbitrary traffic, and (3) how to deploy this solution to mobile users worldwide.

In a recent publication, [35] we demonstrated how to address these challenges while building the Differentiation Detector system. We summarize the key findings here.

We used a trace record-replay methodology to reliably detect traffic differentiation for arbitrary applications in the mobile environment. First, we used Meddle to record a packet trace from a target application running on a mobile device. Then we extracted the application-layer byte streams to replay over a target network. When testing for differentiation, our client and server replayed these streams, both in plaintext and through an encrypted channel (using the Meddle VPN tunnel). The plaintext flows are our exposed trials (subject to filtering by deep packet inspection, or DPI), while the VPN flows are the controlled trails, based on the assumption that the ISP cannot break encryption to run DPI on the encrypted content. Finally, we proposed and used several validated statistical tests to determine whether other differentiation occurred.

The summary results for US ISPs are presented in Table 3, based on data collected in early 2015. In other networks outside the US we have not yet identified differentiation (though it is likely due to the fact that we have few samples outside the US in terms of users and apps, and not because differentiation does not occur outside the US).

| ISP | YouTube | Netflix | Spotify |
|---|---|---|---|
| Verizon | m | m | m |
| T-Mobile | - | - | - |
| AT&T | m | m | m |
| Sprint | m/p | m/p | m/p |
| Boost | m | m | m |
| BlackWireless | 60% | - | - |
| H2O | 37%* | 45%* | 65%* |
| SimpleMobile | 36% | - | - |
| NET10 | p | p | p |

**Table 3. Differentiation detection results per ISP in our dataset, for three popular apps: YouTube, Netflix, and Spotify. When shaping occurs, the table shows the difference in average throughput (%) we detected. A dash (-) indicates no differentiation, (p) means a "translucent" proxy changed connection behavior from the original app behavior, and (m) indicates that a middlebox modified content in flight between client and server. *For the H2O network, replays with random payload have better performance than VPN and exposed replays, indicating a policy that favors non-video HTTP over VPN and streaming video.**

Our key finding is that three mobile networks (BlackWireless, H2O, and SimpleMobile) shaped traffic in early 2015, leading to a difference in average throughput of up to 65%. Interestingly, these shaping practices always affected YouTube (likely due to its large traffic volumes), but not always Netflix and Spotify. We did not identify differentiation for UDP (user datagram protocol) traffic. Importantly, differentiation can also lead to improved performance; we found that H2O offered better performance to port 80 traffic that was not streaming audio/video. We tested these networks again in August 2015 and did not observe differentiation—likely due to FCC rules, effective in June 2015, that disallow differentiation.

As part of this study, we identified other interesting behavior indicating ISP manipulation of traffic in unexpected (and often undocumented) ways. For example, Boost transcodes YouTube video to a lower quality and then caches the downgraded video, violating YouTube caching directives. We also found proxies that reorder TCP headers, inject tracking information, compress content, and change TCP behavior. We found these practices to be pervasive in mobile networks.

Last, we note that ISPs, middleboxes, and client software are known to change Web page content for a variety of reasons, including performance optimization and security. In some cases, a third party changes a page for selfish reasons, e.g., in order to insert ads that generate revenue for that party. While we did not find extensive examples of this behavior in

our experiments, we did identify content injection in China, where a banner ad was replaced by information about the local airport (Figure 3).




**Figure 3: Screen capture of content injection by a Chinese ISP in November, 2013. The highlighted region at the bottom should be an advertisement from a US company.**

## Discussion

We describe Meddle, a platform for gaining visibility and control over network flows from mobile devices. Meddle presents new opportunities for researchers to experiment with middlebox services on Internet traffic generated by mobile users. Meddle also provides several clear incentives for users to participate in the system. We demonstrated the effectiveness of our approach with controlled experiments, a small user study, and case studies of applications built atop Meddle. Our ongoing work focuses on inviting more users to participate in our study (including from developing regions), developing additional meddlebox services, and opening the platform to other researchers.

We now discuss several other topics related to our project.

**Incentives for adoption.** For the user deployment models, Meddle presents several incentives that appeal to a wide range of users, including improved security through encrypted tunnels and device-wide content filters often used for ad-blocking. The applications built atop Meddle serve as additional incentives, namely improving privacy (by automatically identifying and blocking privacy leaks), policy transparency (e.g., identifying traffic differentiation and content modification), and better performance (by accelerating applications). As evidence of incentives for adoption, an existing system (Awazza [2]) that uses APNs and in-network proxy-based performance optimizations has tens of thousands of users.

**Privacy.** An important concern for any Meddle user study is privacy. We developed an IRB-approved protocol [5] (NEU #13-11-17) to address privacy concerns. We obtain informed

consent to enroll users, we encrypted and anonymized all captured flows, we stored the secret key on a separate secure server, and we allowed users to delete their data at any time. For users running Meddle outside the study, Meddle can be configured not to store any network traces or other private information. In addition, we are developing a Raspberry Pi deployment that will allow users to run Meddle on their own hardware in their home network, avoiding the privacy issues entirely.

**Deployment model.** In some cases, our cloud-based deployment model is the right way to implement new services that could be costly to deploy on devices (prefetching) or impractical to deploy in network (detecting ISP service differentiation). In other cases, Meddle provides a practical partial solution to a problem where the complete solution has an impractical cost. For example, identifying privacy leaks from mobile devices can be reliably addressed using information flow analysis [21]. However, due to the overhead of this approach, it is difficult to deploy to users and at scale. Meddle allows us to identify and block unobfuscated PII in network flows from arbitrary devices without requiring OS modifications or taint tracking. Regardless of the ultimate optimal solution, we can use Meddle today to inform the design and deployment of future functionality in OSes and for in-network devices.

In summary, Meddle enables privacy and policy transparency through:

- **Visibility.** Meddle captures all of a device's Internet traffic, allowing us to characterize network flows and interpose on them using software middleboxes. It provides continuous visibility, regardless of the mobile OS, access technology, or apps installed.

- **Control.** Meddle provides simple abstractions for researchers and developers to block, shape, inject or otherwise modify network flows matching various criteria. It also supports applications that operate on collections of flows over time and across users (e.g., Web caching/prefetching, PII detection/blocking and ISP characterization).

- **Deployability.** Meddle can be deployed and distributed quickly, easily, and at scale [45, 46]. To experiment with larger sets of users, we are working with a large ISP to deploy our system in an in-network proxy [2].

## References

1. AppsApk.com. http://www.appsapk.com/.

2. AwaZza. http://www.awazza.com/web/.

3. Data Compression Proxy. https://developer.chrome.com/multidevice/data-compression.

4. Lightbeam for Firefox. http://www.mozilla.org/en-US/lightbeam/.

5. Meddle IRB consent form. https://docs.google.com/forms/d/1Y-xNg7cJxRnlTjH_56KUcKB_6naTfRLqQlcZmHtn5lY/viewform.

6. Mobilyzer. http://www.mobilyzer-project.mobi.

7. Open vSwitch: An open virtual switch. http://openvswitch.org/.

8. UI/Application Exerciser Monkey. https://developer.android.com/tools/help/monkey.html.

9. App review. https://developer.apple.com/app-store/review/, June 2014.

10. Policy guidelines and practices. https://support.google.com/googleplay/android-developer/answer/113474, June 2014.

11. Agarwal Y and Hall M. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. In Proc. of MobiSys, 2013.

12. Arzt S, Rasthofer S, Fritz C, Bodden E, Bartel A, Klein J, Le Traon Y, Octeau D, and McDaniel P. FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In Proc. of PLDI, 2014.

13. Book T and Wallach D. A Case of Collusion: A Study of the Interface Between Ad Libraries and Their Apps. In Proc. of ACM SPSM, 2013.

14. Cao Y, Fratantonio Y, Bianchi A, Egele M, Kruegel C, Vigna G, and Chen Y. EdgeMiner: Automatically Detecting Implicit Control Flow Transitions through the Android Framework. In Proc. of NDSS, 2015.

15. Challen G. Phonelab testbed. http://www.phone-lab.org.

16. Chen X, Jin R, Suh K, Wang B, and Wei W. Network Performance of Smart Mobile Handhelds in a University Campus WiFi Network. In Proc. of IMC, 2012.

17. Clark D. Network neutrality: Words of power and 800-pound gorillas. International Journal of Communication, 2007.

18. Consolvo S, Jung J, Greenstein B, Powledge P, Maganis G, and Avrahami D. The Wi-Fi Privacy Ticker: Improving Awareness & Control of Personal Information Exposure on Wi-Fi. In Proc. of UbiComp, 2010.

19. Dischinger M, Marcon M, Guha S, Gummadi K, Mahajan R, and Saroiu S. Glasnost: Enabling end users to detect traffic differentiation. In Proc. of USENIX NSDI, 2010.

20. Egele M, Kruegel C, Kirda E, and Vigna G. PiOS: Detecting Privacy Leaks in iOS Applications. In Proc. of NDSS, 2011.

21. Enck W, Gilbert P, Chun B, Cox L, Jung J, McDaniel P, and Sheth A. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In Proc. of USENIX OSDI, 2010.

22. Farkas V, Héder B, and Nováczki S. A Split Connection TCP Proxy in LTE Networks. In Inf. Comm. Tech., 2012.

23. FCC announces "Measuring Mobile America" program. http://www.fcc.gov/document/fcc-announces-measuring-mobile-america-program.

24. FCC. Protecting and promoting the open internet. https://www.federalregister.gov/articles/2015/04/13/2015-07841/protecting-and-promoting-the-open-internet, April 2015.

25. Federal Communications Commission. In the matter of protecting and promoting the open internet. GN Docket No. 14-28, May 2014.

26. Gellman B and Soltani A. NSA tracking cellphone locations worldwide, Snowden documents show. Washington Post, December 4 2013. Retrieved from http://www.washingtonpost.com/.

27. Gerber A, Pang J, Spatscheck O, and Venkataraman S. Speed Testing without Speed Tests: Estimating Achievable Download Speed from Passive Measurements. In Proc. of IMC, 2010.

28. Gibler C, Crussell J, Erickson J, and Chen H. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. In Proc. of TRUST, 2012.

29. Gill P, Erramilli V, Chaintreau A, Krishnamurthy B, Papagiannaki D, and Rodriguez P. Follow the Money: Understanding Economics of Online Aggregation and Advertising. In Proc. of IMC, 2013.

30. Grace M, Zhou W, Jiang X, and Sadeghi A. Unsafe Exposure Analysis of Mobile In-app Advertisements. In Proc. of WISEC, 2012.

31. Han S, Jung J, and Wetherall D. A Study of Third-Party Tracking by Mobile Apps in the Wild. Technical Report UW-CSE-12-03-01, University of Washington, 2012.

32. Hao S, Liu B, Nath S, Halfond W, and Govindan R. PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps. In Proc. of MobiSys, 2014.

33. Jeon J, Micinski K, and Foster J. SymDroid: Symbolic Execution for Dalvik Bytecode. Technical Report CS-TR-5022, University of Maryland, College Park, 2012.

34. Johnson C. US Office of Management and Budget Memorandum M-07-16. http://www.whitehouse.gov/sites/default/files/omb/memoranda/fy2007/m07-16.pdf, May 2007.

35. Kakhki A, Razaghpanah A, Koo H, Li A, Golani R, Choffnes D, Gill P, and Mislove A. Identifying traffic differentiation in mobile networks. In Proc. of IMC, 2015.

36. Kim J, Yoon Y, Yi K, and Shin J. SCANDAL: Static Analyzer for Detecting Privacy Leaks in Android Applications. In Proc. of MoST, 2012.

37. Kreibich C, Weaver N, Nechaev B, and Paxson V. Netalyzr: Illuminating the edge network. In Proc. of IMC, 2010.

38. Krishnamurthy B and Wills C. Privacy Diffusion on the Web: A Longitudinal Perspective. In Proc. of ACM WWW, 2009.

39. Lindorfer M, Neugschwandtner M, Weichselbaum L, Fratantonio Y, van der Veen V, and Platzer C. Andrubis - 1,000,000 Apps Later: A View on Current Android Malware Behaviors. In Proc. of BADGERS, 2014.

40. Lu L, Li Z, Wu Z, Lee W, and Jiang G. CHEX: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. In Proc. of ACM CCS, 2012.

41. Mahajan R, Zhang M, Poole L, and Pai V. Uncovering performance differences among backbone ISPs with Netdiff. In Proc. of USENIX NSDI, 2008.

42. Rao A, Kakhki A, Razaghpanah A, Tang A, Wang S, Sherry J, Gill P, Krishnamurthy A, Legout A, Mislove A, and Choffnes D. Using the Middle to Meddle with Mobile. Technical report, Northeastern University, 2013.

43. Ren J, Rao A, Lindorfer M, Legout A, and Choffnes D. ReCon: Revealing and controlling privacy leaks in mobile network traffic. Technical report, Northeastern University, 2015.

44. Roesner F, Kohno T, and Wetherall D. Detecting and Defending Against Third-Party Tracking on the Web. Proc. of USENIX NSDI, 2012.

45. Sekar V, Egi N, Ratnasamy S, Reiter M, and Shi G. Design and implementation of a consolidated middlebox architecture. In Proc. of USENIX NSDI, 2012.

46. Sherry J, Hasan S, Scott C, Krishnamurthy A, Ratnasamy S, and Sekar V. Making middleboxes someone else's problem: Network processing as a cloud services. In Proc. of ACM SIGCOMM, 2012.

47. Sommers J and Barford P. Cell vs. WiFi: On the Performance of Metro Area Mobile Connections. In Proc. of IMC, 2012.

48. Spinellis D. Another Level of Indirection. In A. Oram and G. Wilson, editors, Beautiful Code: Leading Programmers Explain How They Think, chapter 17, pages 279–291. O'Reilly and Associates, 2007.

49. Stoica I, Adkins D, Zhuang S, Shenker S, and Surana S. Internet Indirection Infrastructure. In Proc. of ACM SIGCOMM, 2002.

50. Tariq M, Motiwala M, Feamster N, and Ammar M. Detecting network neutrality violations with causal inference. In CoNEXT, 2009.

51. The Wall Street Journal. What They Know - Mobile. http://blogs.wsj.com/wtk-mobile/, December 2010.

52. Vallina-Rodriguez N, Shah J, Finamore A, Haddadi H, Grunenberger Y, Papagiannaki K, and Crowcroft J. Breaking for Commercials: Characterizing Mobile Advertising. In Proc. of IMC, 2012.

53. Wang Z, Qian Z, Xu Q, Mao Z, and Zhang M. An Untold Story of Middleboxes in Cellular Networks. In Proc. of ACM SIGCOMM, 2011.

54. Weaver N, Kreibich C, Dam M, and Paxson V. Here Be Web Proxies. In Proc. PAM, 2014.

55. Xu X, Jiang Y, Flach T, Katz-Bassett E, Choffnes D, and Govindan R. Investigating transparent web proxies in cellular networks. In Proc. PAM, 2015.

56. Yan L and Yin H. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In Proc. of USENIX Security, 2012.

57. Yang Z, Yang M, Zhang Y, Gu G, Ning P, and Wang X. AppIntent: Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection. In Proc. of ACM CCS, 2013.

58. Zhang Y, Mao Z, and Zhang M. Detecting Traffic Differentiation in Backbone ISPs with NetPolice. In Proc. of IMC, 2009.

59. Zhang Y, Yang M, Xu B, Yang Z, Gu G, Ning P, Wang X, and Zang B. Vetting undesirable behaviors in Android apps with permission use analysis. In Proc. of ACM CCS, 2013.

60. Zhauniarovich Y, Ahmad M, Gadyatskaya O, Crispo B, and Massacci F. StaDynA: Addressing the Problem of Dynamic Code Updates in the Security Analysis of Android Applications. In Proc. of ACM CODASPY, 2015.

## Authors

Ashwin Rao is a postdoctoral researcher in the NODES research group at the University of Helsinki. He received his PhD while working in the DIANA project team at Inria Sophia Antipolis. Prior to joining Inria, he was a Master's student in the School of Information Technology at the Indian Institute of Technology Delhi.

Arash Molavi Kakhki is a PhD candidate in Computer Science at Northeastern University in Boston, MA. He received his MSc from Imperial College London and BSc from the Sharif University of Technology. His research interests broadly lie in the areas of networking, network measurements, online privacy, and social networks.

Abbas Razaghpanah is a third-year PhD student in the computer networks group at Stony Brook University. Prior to joining Stony Brook University, he completed his bachelor's degree at Amirkabir University of Technology in Tehran. Prior to that, he attended the National Organization for Development of Exceptional Talents (NODET), where he studied mathematics and physics. Abbas has received a runner-up award in the ACM Research Competition for his work on detecting traffic differentiation and the Open Technology Fund Information Controls Fellowship.

Anke Li is a PhD student in the computer networks group at Stony Brook University. His research interests lie in computer networks and network measurement. More specifically, his recent work focuses on building systems to investigate and monitor various issues happening today in different networks and places. These issues, such as online information control and traffic differentiation, are generally related to net neutrality principles.

David Choffnes is an Assistant Professor in the College of Computer and Information Science at Northeastern University. His research is primarily in the areas of distributed systems and networking, with a recent focus on mobile systems and privacy. Much of his work entails crowdsourcing measurement and performance evaluation of Internet systems by deploying software to users at the scale of tens or hundreds of thousands of users. He is a co-author of three textbooks, and his research has been supported by the NSF, a Google Faculty Research Award, the Data Transparency Lab, VidScale, M-Lab, and a Computing Innovations Fellowship.

Arnaud Legout is a Research Scientist at Inria Sophia Antipolis. He earned his PhD in 2000 from the Institut Eurecom, France. His main research interests are privacy, social networks, and peer-to-peer systems, with a strong focus on large-scale measurements and experiments. Most of his research activities are addressing problems with a strong societal impact.

Alan Mislove is an Associate Professor at the College of Computer and Information Science at Northeastern University. He received his PhD from Rice University in 2009. Prof. Mislove's research concerns distributed systems and networks, with a focus on using social networks to enhance the security, privacy, and efficiency of newly emerging systems. He is a recipient of

an NSF CAREER Award (2011), and his work has been covered by the *Wall Street Journal*, the *New York Times*, and the CBS Evening News.

Phillipa Gill is an Assistant Professor in the Computer Science Department at Stony Brook University. Her work focuses on many aspects of computer networking and security, with a focus on designing novel network measurement techniques to understand online information controls, network interference, and interdomain routing. She currently leads the ICLab project, which is working to develop a network measurement platform specifically for online information controls. She has received the NSF CAREER award, Google Faculty Research Award and best paper awards at the ACM Internet Measurement Conference (characterizing online aggregators), and Passive and Active Measurement Conference (characterizing interconnectivity of large content providers).

We would like to thank several contributors to the Meddle research project, including Walid Dabbous, Arvind Krishnamurthy, Jingjing Ren, Amy Tang, Shen Wang, Justine Sherry, and Nick Martindell. We also thank the participants in our user study.

**Referring Editor:** Latanya Sweeney

---

## Citation

Rao A, Kakhi A, Razaghpanah A, Li A, Choffnes D, Legout A, Mislove A, Gill P. Meddle: Enabling Transparency and Control for Mobile Internet Traffic. *Technology Science*. 2015103003. October 30, 2015. http://techscience.org/a/2015103003/

---

## Data

For data on traffic differentiation. See http://dd.meddle.mobi/codeanddata.html

We are in the process of preparing data on privacy leaks for publication while still ensuring privacy and security safeguards and will update with links as soon as possible.