

Back in control — An extensible middle-box on your phone

James Newman* Abbas Razaghpanah⁺ Narseo Vallina-Rodriguez^{†‡}
Fabián E. Bustamante* Mark Allman[‡] Diego Perino[◇] Alessandro Finamore[◇]

Northwestern University* Stony Brook University⁺ IMDEA Networks Institute[†] ICSI[‡] Telefonica Research[◇]

ABSTRACT

The closed design of mobile devices — with the increased security and consistent user interfaces— is in large part responsible for their becoming the dominant platform for accessing the Internet. These benefits, however, are not without a cost. Their operation of mobile devices and their apps is not easy to understand by either users or operators.

We argue for recovering transparency and control on mobile devices through an extensible platform that can intercept and modify traffic before leaving the device or, on arrival, before it reaches the operating system. Conceptually, this is the same view of the traffic that a traditional middlebox would have at the far end of the first link in the network path. We call this platform “middlebox zero” or MBZ. By being on-board, MBZ also leverages local context as it processes the traffic and complement the network wide view of standard middleboxes. We discuss the challenges of the MBZ approach, sketch a working design, and illustrate its potential with some concrete examples.

1 INTRODUCTION

Mobile devices are indispensably handy and fundamentally enigmatic. In just over ten years, they have become the dominant platform for accessing the Internet. This is at least partially due to their closed nature, a design decision clearly stated by one of its main designers — Steve Jobs — “We define everything that is on the phone” [18].

This closed design, and a policed environment where the manufacturer has veto power over every application, is clearly beneficial for security and ensures user-friendly, consistent interfaces. These benefits are not without costs.

The operation of mobile devices and apps is hard to understand. Noticeable changes in performance are as commonplace as difficult to diagnose and even basic questions, such as *why is this website not loading?* or *with what other sites is this website communicating?* do not have ready answers.

This challenge has inspired different approaches to return visibility and control to users and developers. Some tools

will let users change the behavior of applications without accessing the APK [25] or installing a BusyBox-like toolset [3]. Most of them, however, require users to “root” their phones. A 2014 survey with over 14k users found that 63% have rooted their primary Android device [20]. Unfortunately, rooting a phone comes with its own risks. Beyond the fact that this may void a device’s warranty or result on the device being “bricked”, rooting has been linked to mobile malware attacks and adware [9]. In a 2014 announcement,¹ Gartner predicted that 75% of mobile security incidents will be due to mobile application misconfiguration with the biggest threat being devices altered at an administration level.

VPN APIs on mobile devices offer another option to regain control over application’s traffic. VPN APIs enable the development of mobile apps, mostly for improved security and privacy, without requiring root access thus being easy for users to adopt while eliminating/reducing the risk of attacks. Today, however, all apps relying on it are custom, mutually exclusive solutions to specific problems.

Choffnes [7] recently proposed creating Personal Virtual Networks to give users the illusion of a home network no matter what the network they are actually connected to. This paradigm, similar to typical VPN use-cases, addresses the need from users to gain back control over their own traffic when on untrusted networks. The VPN server, fully owned and controlled by the end-user, can implement different policies for network traffic. However, user traffic is still transported to a remote server which lacks access to potentially valuable end-user and device context.

On the network operator side the approach to control and optimize mobile traffic involves placing middleboxes in the network path. Middleboxes are so popular that many enterprise networks now have as many middleboxes as routers [30] serving as firewalls, performance enhancing proxies, NATs, or deep packet inspectors. The advent of Network Function Virtualization (NFV) significantly reduced middlebox deployment cost and time, while current deployment of edge computing infrastructures enables the placement of middlebox-like functions very close to the end-user

¹<https://www.gartner.com/newsroom/id/2753017>

device. While clearly powerful and with an unparalleled network-wide view, middleboxes still show limitations in terms of scalability, especially in presence of mobile users. In addition, as in the case of Personal Virtual Networks, these middleboxes also lack access to potentially relevant context.

We argue for recovering transparency and control on mobile devices through an extensible platform that can intercept and modify traffic before leaving the device or, on arrival, before the traffic is pass on to the operating system.

Conceptually, this is the same view of the traffic that a traditional middlebox would have at the far end of the first link in the network path and so we call this approach “middlebox zero” or MBZ. By existing on-board, MBZ can leverage local context to process the traffic and complement the network-wide view and optimization capabilities of traditional middleboxes. Rather than a dedicated middlebox, MBZ can be extended with user-specified extensions that manipulate traffic for transparency or control. This approach has been used before to monitor network performance [31, 34] and privacy leaks [22], however, MBZ takes the idea a step further. Instead of having standalone, custom-built solutions for each problem, the extensible nature of MBZ allows the user to install multiple, different plugins side-by-side.

MBZ opens up a wide range of opportunities and interesting challenges: from the architecture of an MBZ that could support dynamic extensibility, with minimum overhead and without compromising security, to new forms of interactions between MBZ and traditional middleboxes an MBZ’ed device can run into as it moves between networks. The MBZ extensibility could also enable a service marketplace, wherein third parties can advertise new extensions since no single provider could expect to offer all imaginable needs. This alone brings a plethora of research questions, from how to declare extensions to the most appropriate model for their management and control.

2 MOTIVATION AND BACKGROUND

We now motivate the need for MBZ by first examining the benefits and limitations of existing solutions and related work to control and optimize mobile traffic. We argue that full control and optimization can best be achieved via an extensible middlebox-like platform on the end-user device, capable of collaborating with in-network middleboxes.

2.1 User Device Solutions

When faced with unanswerable questions regarding device behavior or network activity, a popular approach is rooting the device. This returns control to the device’s owner and allows them to install unlicensed applications or gain access to the full Linux kernel with command such as *tcpdump* for network debugging. To root their device, most users turn to

rooting tools such as KingoRoot [16] or SuperSu[32]. While these tools streamline the rooting process, they require technically skilled users, and often the development of custom commands and applications to control and optimize traffic. More problematically, these tools have been linked to malware [12] and adware [9] attacks.

Recently, native support of VPN APIs by the main mobile platforms allowed developers to capture and manipulate traffic generated by applications without root access. This has led to the development of applications for privacy control, measurements, and firewalls such as Lumen [22], AntMonitor [31], Mopeye [34] or NetGuard [11], that can be easily installed and effectively achieve their goals without exposing users to potential attacks.

We see these apps as first steps towards on-board functions for traffic control and optimization, showing growing users’ interest. Each of them, however, have been developed and optimized – from scratch – for a specific function and cannot run concurrently with other similar apps.² We argue for an extensible platform supporting multiple functions in parallel, each of which can be more easily developed by reusing core traffic manipulation code (i.e., an on-board middlebox), and instantiated on demand.

2.2 Network Middleboxes

For companies and ISPs that do not have access to the end device, the most suitable option to control and optimize mobile traffic is to deploy a middlebox in the network. This allows them to gain access to all the traffic generated from any device connected through the middlebox. Middleboxes provide several services, from firewalls to protocol accelerators. For instance, proxies have been used to enable mobile data compression on the fly [1], and previous work focused on how protocols and middleboxes interact [8, 19], how middleboxes can be designed to enable innovation [29], and how devices can establish their own private network [7].

With the advent of NFV and growing popularity of cloud platforms, middleboxes can be developed in software and instantiated on-demand using commodity hardware already installed in the network [6, 13, 15, 17, 21, 30], drastically reducing deployment cost and time. To improve cloud middlebox deployment, several projects studied how middleboxes can be consolidated [28], or made extensible [4]. The current deployment of edge computing infrastructures enables the placement of middlebox-like functions very close to users’ devices, in Central Offices or even in eNodeBs. This provides middleboxes a network-wide view and the opportunity to

²Android only allows one VPN application at the time for security reasons. The right is revoked when another application is granted the VPN permission [5].

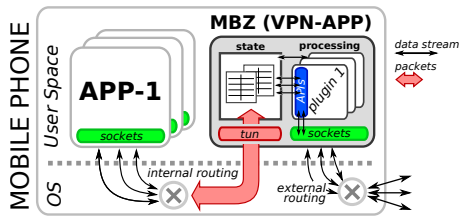


Figure 1: Diagram of MBZ prototype.

jointly optimize aggregate traffic of multiple users, e.g., content placement, caching, network routing. However, they still have limitations that can only be solved by pushing some of middlebox services and features to the end device.

First, despite highly distributed and optimized NFV systems, scalability issues may arise if all users offload all their desired functions to the network. Second, middleboxes lack user context that exists on each end device. Because the middlebox sits within the network, it is for instance difficult to understand if traffic generated by a device was initiated by a given application on that device due to a user action or if it was generated in the background. Furthermore, private information still leaves the user device and is transferred and processed in the network limiting user control. Finally, any benefit or information gained from the middlebox is only effective while the device remains connected to that network or to a particular location of a given network. Once the device leaves and moves to a different network or location, it is difficult to re-connect the traffic to the new location, especially in case of stateful services, where state migration is required and particularly challenging in presence of large number of highly mobile users.

3 MBZ ARCHITECTURE

In this section we present a brief description of MBZ design. We start by discussing how we handle the traffic from apps on the device before outlining how MBZ balances extensibility and security.

3.1 Managing Traffic

To operate as a middlebox, MBZ must be able to handle traffic before it leaves the device after being initiated by an application, and before it is passed to the application after being received off the network.

MBZ uses the VPN interface available as an API on today's mobile phones. This approach has been used before to detect privacy leaks [22] and monitor network performance [31, 34]. MBZ builds on it with a much broader goal: to capture traffic and funnel it through a series of user-specified extensions.³

³Thus, despite our discussion focused on mobile platforms, any platform allowing the creation of virtual interface APIs could leverage an MBZ approach.

As depicted in Fig. 1 the VPN API reveals a tun interface that captures any traffic generated by applications on the device. While a typical VPN application would forward the traffic to a server or proxy, MBZ handles that on the device. To accomplish this, the MBZ must extract flow state from the UDP/TCP and IP packet headers for packets arriving on the tun interface and map the packet to a regular socket (creating sockets as needed). The MBZ must also maintain this state, both for UDP and TCP flows, so that it can marshal data arriving from remote hosts on the sockets back into packets for transmission to the app via the tun interface.

In the case of TCP traffic, when the MBZ reads a SYN packet from the tun interface, it must create a new socket, connect to the endhost target and instantiate state internally. After the OS establishes the socket, the MBZ returns a SYN/ACK via the tun interface to the originating app. Managing state efficiently is critical, as a user-space MBZ is subject to the standard UNIX file descriptor limit and, so, must close sockets and flush state periodically. The explicit connection teardown of TCP connections provides a clear signal for clean-up. However, for UDP's connectionless nature, we leverage inactivity timeouts as do NATs [24]. One workaround would be to reuse sockets for UDP traffic, which can significantly reduce the number of sockets used for DNS resolutions, but does not solve the problem entirely.

3.2 Plugin Management and Security

A key aspect of MBZ is its extensible nature. This is made possible by a framework dedicated to installing and organizing plugins specified by the user. We describe its preliminary design and workings in this section.

We envision the plugins in MBZ to be similar to applications installed on the device. There will be an ecosystem of plugins to be installed and with the user having the ultimate say in which plugins are installed on in MBZ. In order for this to work efficiently and safely, MBZ must ensure that the plugins cooperate to benefit the user without interfering in or hindering the work of other plugins and the user alike.

There are two main factors that MBZ addresses when handling plugin execution. First, plugins are granted a set of permissions to manipulate user traffic similarly to browser extensions. MBZ ensures that every plugin respects those permissions and no plugin abuses its power over the user's traffic. There are several ways MBZ does this, first being that it limits the actions that plugins are permitted to execute on packets. For instance, unless specified by the user, MBZ prevents plugins from collecting user data or routing traffic to a third-party server.

Second, MBZ guarantees resource isolation and management across plugins, similarly to existing NFV platforms.

Indeed, a processing/memory intensive plugin could consume all available resources hurting other applications or plugins performance and potentially impact user experience. Further, given the open nature of MBZ, it could be abused to initiate malicious activities. MBZ monitors plugins' resource usage and disables them if demands increase beyond a set of thresholds which could be set based on users' defined priorities and resources availability. Identifying and fine-tuning these thresholds remains future work. Further, we plan to make our resource management techniques context aware. For instance, it is critical to adapt resource allocation to battery levels, or adjust traffic generated by plugins to connectivity type (e.g., limit data exchange when a device is connected to cellular network to protect the user's data limits).

4 EXAMPLE USE CASES FOR MBZ

The value of an on-board middlebox comes from combining the view of the traffic that traditional middleboxes enjoy with the opportunity to leverage the local—user-, device- and network-specific—context when processing traffic.

We describe several use cases and applications for MBZ. The list is clearly not exhaustive, but meant to illustrate some of the functionality that MBZ enables. As we discussed in § 3, we envision an architecture in which each use case operates as a separate module or plugin to be installed by the user or by the app developer. Some of these plugins could be bundled with MBZ as default plugins, while others could be acquire from a third-party.

4.1 User-Defined Firewalls

Sending all traffic through MBZ enables much control over the traffic, including the ability to monitor, block, redirect, shape and change traffic in different ways. While middleboxes within the network have some of the same abilities, MBZ augments this ability with device context which can enable more complex rules, especially because it can identify which application is generating the traffic. For instance, consider the following firewalling functions that MBZ can facilitate:

Blacklisting: The MBZ can implement blacklists (or whitelists) that block (or enable) traffic to/from certain hosts or domains, as in Fig. 2. This can be done for all traffic or just for certain apps. For instance, one could configure the MBZ to only allow the email application to interact with the user's known IMAP and SMTP servers to thwart efforts to conduct email tracking.

Protocol Usage: The MBZ can prohibit certain protocols in specific instances or adjust protocol settings based on network conditions. For instance, the MBZ can ensure that a banking app uses only encrypted connections.

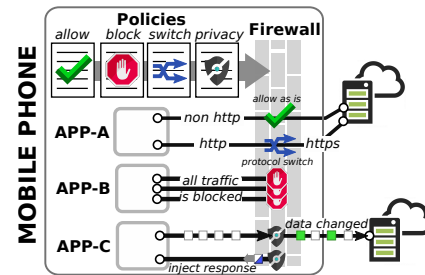


Figure 2: An MBZ extension could act as a firewall on the endhost, enacting user-set rules such as Switch, or Deny on the traffic.

Enhance Privacy: User tracking is omnipresent in the mobile ecosystem [10, 23, 33]. The MBZ is in a unique position to give users the ability to wrest back control over their privacy by allowing them to either replace private information inside network flows with random values before they leave the device, or block such flows entirely. These controls can be precisely tuned using fine-grain details of the communication, such as app, destination (e.g., disallowing known third-party trackers), protocol (e.g., blocking unencrypted protocols), and type of information being leaked.

The MBZ can use a variety of methods to enforce the smart firewalling policies. The exact method employed depends on the specific use case and app pattern. For instance, the MBZ could prevent communication with a blacklisted host by issuing a TCP reset packet to the app instead of instantiating a connection. Alternatively, the MBZ could inject a response to a dubious request letting the user know why their traffic was blocked. This approach could be taken a step further and the MBZ could alert users and determine if the user would like to proceed anyway. Another tool at the MBZ's disposal is re-writing certain portions of the content to obscure sensitive data (e.g., re-writing the IMEI).

4.2 Fine-Grained Traffic Routing

Given that the MBZ intercepts traffic before it leaves the device, the system can re-direct the traffic based on users' preferences rather than the current one-size-fits-all model of traffic forwarding.

Smart Multihoming: Mobile phones often have multiple connectivity options. While traditionally phones only use one network at a time, they are often in locales where multiple networks are available (e.g., cell and a WiFi). This opens several opportunities. For instance, users could identify their network preference at some locations (such as WiFi at work), for all or specific apps (WiFi at work, except for Facebook to skirt a policy block on the WiFi network). Alternatively, the MBZ could monitor performance—see § 4.3—and try to choose the “best” network at the given time.

Smart Tunneling: Users often have the ability to connect to myriad different private networks via tunnels. For instance, companies may force their employees to use VPNs to access content hosted on their private network. Or, privacy-aware individuals may use commercial VPN services or anonymization networks such as Tor to obfuscate the source of the traffic. The MBZ can implement all these policies. Furthermore, MBZ can offer precise control such that a user’s work traffic is sent to their employer’s VPN server while a particular app’s traffic may be directed to a VPN server hosted at user’s home to avoid geo-filters while traveling [7]. This capability of MBZ allows for unprecedented control instead of the current one-size-fits-all approach.

Smart Multipathing: MBZ can also be used to enable the use of multiple paths to aid performance. This could be realized, for instance, by turning normal TCP connections into a series of MP-TCP subflows (which may or may not be divided across the device’s network interfaces). Alternatively, the MBZ could spread requests to multiple instances of some replicated service (e.g., a DNS resolver pool or among edge servers in a CDN).

4.3 Network Troubleshooting

MBZ can observe the performance of all network traffic on the device, and thus can detect performance variations and trends within a user’s normal interactions, and contextualize those observations with link-level insights (e.g., SNR).

Allman and Paxson first described a reactive measurement approach that advocates for considering measurement to be a process and not an event [2]. While studies have used the reactive measurement notion—e.g., for broadband characterization [27]—MBZ provides a generic platform that enables reactive measurements by, for instance, triggering active measurements based on passive observations.

As an example, consider issues caused by the DNS resolution process. A device could—likely in a sampled fashion—trigger alternate DNS and resulting TCP transactions from passively observed traffic. Now the device can monitor multiple transactions that nominally do the same task. At this point we can engage in “what if?” analysis. E.g., would performance be better or worse if hostnames were resolved via a public DNS resolver (e.g., OpenDNS or Google’s Public DNS)? E.g., is the ISP manipulating the DNS responses in some fashion (to monetize errors or for censorship)? The answers to these questions can then lead to configuration changes (e.g., sending DNS queries through a VPN to circumvent censorship).

4.4 Improved Protocol Stacks

Application developers can use a variety of network- (e.g., IPv4 or IPv6), transport- (e.g., UDP/QUIC or TCP) and application-layer (e.g., HTTP(S) or HTTP2) protocols, many

interchangeable, to create mobile applications. However, developers typically stick to a single combination of application- and transport-level protocols for all situations, regardless of network conditions. These limit the adaptability of applications in today’s evolving network conditions, particularly when a given protocol or network element may outperform an equivalent one.

MBZ has the ability to experiment and examine how different combinations of compatible protocols perform over time for a given device and endhost machine, at a given time and network. This historical data can be used by the MBZ to dynamically select a given application’s protocol to optimize application performance and ultimately the user experience. Importantly, this is only possible because of MBZ’s place in the network: the end host. MBZ can implement changes and solutions that can help improve the last mile of the network. Middleboxes within the network do not have this ability.

One instance where a dynamic protocol stack could be beneficial to the user is in scenarios with a high packet-loss and high latency in the last mile. The MBZ can identify when a network becomes lossy and adapt the traffic to such adversary conditions.

For example, MBZ can wrap underperforming TCP packets within a protocol more tolerant to loss, such as QUIC [14, 26], either directly to the remote server or through a proxy. Likewise, the MBZ can also prefetch DNS responses for the domains most relevant for the set of applications running on the device, hence reducing the time required to open a new connection by avoiding unnecessary DNS lookups.

This last example also points to venues in which MBZ could work in collaboration with middleboxes. This will allow as well MBZ to obtain information about the performance at the core of the network (e.g., congestion), data otherwise unavailable to endhosts.

5 INFORMATION AND PERFORMANCE VALIDATION

In this section we present a proof of concept validation of MBZ. First, we show MBZ can provide additional control and transparency to the user with a sample *snitch* plugin. Second, we show the limited overhead MBZ adds to the traffic. This is a simple demonstration of the potential applications of MBZ and we leave a rigorous evaluation as future work.

5.1 What Third-parties are contacted by the website?

We show how MBZ provides users access to information that would otherwise only be available by rooting the phone or in a network middlebox. In this experiment, we set out to answer the question: *what third-parties are contacted by the website?* We use an instantiation of MBZ in the form of an

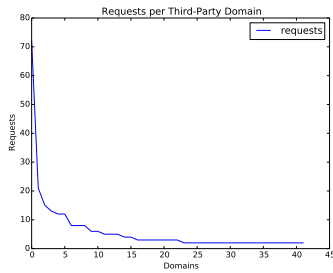


Figure 3: Number of requests per third-party organization from real users running the Snapchat application. Data collected via *snitch* MBZ plugin in the wild.

Android application that has been deployed in the wild [22]. A basic *snitch* plugin is installed through MBZ to investigate the traffic. The plugin passively monitors the connections made by applications and tracks the destination IP address, port, and protocol. This information is then displayed to the user in the MBZ application. For lack of space, we report results for one popular application only: *Snapchat*.

Figure 3 reports the number of requests generated towards different third-party domains. We observe that there are over 40 different organizations (i.e., third parties) contacted and, while most have only 1 request, about 5 organizations account for more than 10 requests each. Additionally, MBZ can identify the protocols used by different applications and share this information with the user. In the Snapchat case, we count a total of 372 third-party flows. The large majority of them are TCP flows (91.7%) while only 8.3% are UDP. Interestingly, some of the UDP flows are leveraged to run the QUIC protocol, indicating that third-party organizations are adopting Google’s new protocol.

5.2 Performance Evaluation

While it is possible to build MBZ using the Android VPN interface, it is critical to ensure that it does not negatively affect the performance of other applications. In this section we present a simple feasibility analysis of MBZ. To this purpose we use a rooted Android phone running *tcpdump*⁴ which would allow us to compare the end-to-end latency experienced by a sample application with and without running our MBZ prototype. While working with a non-optimized prototype, our analysis shows that there is minimal overhead on the end-to-end latency of running applications.

In our experiment, we use a test application that makes successive requests to each of the top 100 Alexa websites while measuring the connection time. We compare the time from when the Java Socket API indicates a successful TCP connection with the TCP connection time on the network interface according to *tcpdump*, i.e., the time between the first

⁴www.tcpdump.org

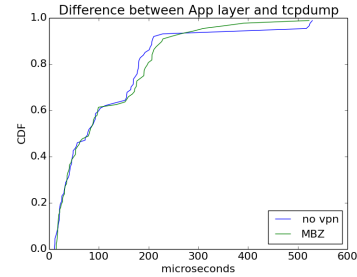


Figure 4: Difference in TCP connection time between *tcpdump* and application level measurement.

SYN and SYN/ACK. We first measure the TCP connection time without an MBZ running on the phone (*no vpn*) as a baseline, so the extra latency is introduced by the Java Virtual Machine (JVM). This allows us to conclude that the JVM adds 77 microseconds (median) for every TCP connection.

To eliminate JVM’s inherent latency, we implement a MBZ prototype in C++ (MBZ). Figure 4 shows the difference between the time reported in *tcpdump* and that on MBZ in microseconds. As the figure shows, the MBZ curve closely overlaps that of the baseline (*no vpn*), mainly due to C++’s ability to treat the tun interface as a socket, making it possible to poll the tun interface along with the remote sockets, which significantly reduces latency compared to alternating between polling the tun and the remote sockets.

6 CONCLUSION

Users and researchers lack dynamic access and control over the networking stack on modern end devices. We advocate locating an extensible virtual middlebox on the end devices themselves. We call this approach “middlebox zero” or MBZ. By being on-board, a MBZ also leverages local context as it processes the traffic and complement the network wide view of standard middleboxes. We discussed the challenges of the MBZ approach, sketch a working design, and illustrate its potential with some concrete examples that, as we argued, would be either cumbersome or not possible to realize using current solutions. We emphasize here that we are merely staking out a position in this paper. While we have developed a proof-of-concept implementation, many questions are left to resolve. Our goal with this paper is to share our ideas with the community in the hopes of gathering early feedback on the MBZ approach.

REFERENCES

- [1] AGABABOV, V., BUETTNER, M., CHUDNOVSKY, V., COGAN, M., GREENSTEIN, B., MCDANIEL, S., PIATEK, M., SCOTT, C., WELSH, M., AND YIN, B. Flywheel: Google’s data compression proxy for the mobile web. In *Proc. USENIX NSDI* (2015).
- [2] ALLMAN, M., AND PAXSON, V. A reactive measurement framework. In *Proc. PAM* (2008).

- [3] ANDERSON, E. BusyBox. <http://busybox.net>, 2018.
- [4] ANDERSON, J. W., BRAUD, R., KAPOOR, R., PORTER, G., AND VAHDAT, A. xomb: extensible open middleboxes with commodity servers. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems* (2012), ACM, pp. 49–60.
- [5] ANDROID DEVELOPERS. VpnService. <https://developer.android.com/reference/android/net/VpnService.html>.
- [6] CHEN, P. F., AND KODIROV, N. Virtual middlebox management for cloud.
- [7] CHOFFNES, D. A case for personal virtual networks. In *Proc. HotNets* (2016).
- [8] CRAVEN, R., BEVERLY, R., AND ALLMAN, M. A middlebox-cooperative tcp for a non end-to-end internet. In *ACM SIGCOMM Computer Communication Review* (2014), vol. 44, ACM, pp. 151–162.
- [9] DEVELOPERS, F. X. KingRoot Malware/Adware root. <https://www.forum.xda-developers.com/android/general/kingroot-malware-adware-root-t35603090>, 2017.
- [10] ENCK, W., GILBERT, P., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. N. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. USENIX OSDI* (2010).
- [11] FAIRCODE. Netguard - no-root firewall, 2017.
- [12] LAB, K. Rooting your Android: Advantages, disadvantages, and snags. <https://www.kaspersky.com/blog/android-root-faq/17135/>, 2017.
- [13] LAN, C., SHERRY, J., POPA, R. A., RATNASAMY, S., AND LIU, Z. Embark: Securely outsourcing middleboxes to the cloud. In *NSDI* (2016), vol. 16, pp. 255–273.
- [14] LANGLEY, A., RIDDOCH, A., WILK, A., VICENTE, A., KRASIC, C., ZHANG, D., YANG, F., KOURANOV, F., SWETT, I., IYENGAR, J., ET AL. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017), ACM, pp. 183–196.
- [15] LEDJIAR, A., SAMPIN, E., TALHI, C., AND CHERIET, M. Network function virtualization as a service for multi-tenant software defined networks. In *Software Defined Systems (SDS), 2017 Fourth International Conference on* (2017), IEEE, pp. 168–173.
- [16] LTD., F. D. T. Kingoroot - the best one click android root apk for free. <https://www.kingoapp.com>, 2017.
- [17] LU, H., SRIVASTAVA, A., SALTAFORMAGGIO, B., AND XU, D. Storm: Enabling tenant-defined cloud storage middle-box services. In *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on* (2016), IEEE, pp. 73–84.
- [18] MARKOFF, J. Steve jobs walks teh tightrope again. *New York Times* (January 12 2007).
- [19] MEDINA, A., ALLMAN, M., AND FLOYD, S. Measuring interactions between transport protocols and middleboxes. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement* (2004), ACM, pp. 336–341.
- [20] POLICE, A. [Weekend Poll] Is Your Phone Rooted. <https://www.androidpolice.com/2016/06/12/weekend-poll-is-your-phone-rooted/>, 2016.
- [21] QAZI, Z. A., TU, C.-C., CHIANG, L., MIAO, R., SEKAR, V., AND YU, M. Simple-fying middlebox policy enforcement using sdn. In *ACM SIGCOMM computer communication review* (2013), vol. 43, ACM, pp. 27–38.
- [22] RAZAGHPANAH, A., VALLINA-RODRIGUEZ, N., SUNDARESAN, S., KREIBICH, C., GILL, P., ALLMAN, M., AND PAXSON, V. Haystack: In situ mobile traffic analysis in user space. *CoRR abs/1510.01419* (2015).
- [23] REN, J., RAO, A., LINDORFER, M., LEGOUT, A., AND CHOFFNES, D. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proc. of ACM MobiSys* (2016).
- [24] RICHTER, P., WOHLFART, F., VALLINA-RODRIGUEZ, N., ALLMAN, M., BUSH, R., FELDMANN, A., KREIBICH, C., WEAVER, N., AND PAXSON, V. A multi-perspective analysis of carrier-grade nat deployment. In *Proc. ACM IMC* (2016).
- [25] ROVO89. Xposed Module Repository. <http://repo.xposed.info/>, 2018.
- [26] RULA, J. P., NEWMAN, J., BUSTAMANTE, F. E., KAKHKI, A. M., AND CHOFFNES, D. Mile high wifi: A first look at in-flight internet connectivity. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web* (2018), International World Wide Web Conferences Steering Committee, pp. 1449–1458.
- [27] SÁNCHEZ, M. A., OTTO, J. S., BISCHOF, Z. S., CHOFFNES, D. R., BUSTAMANTE, F. E., KRISHNAMURTHY, B., AND WILLINGER, W. Dasu: Pushing experiments to the internet’s edge. In *Proc. USENIX NSDI* (2013).
- [28] SEKAR, V., EGI, N., RATNASAMY, S., REITER, M. K., AND SHI, G. Design and implementation of a consolidated middlebox architecture. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (2012), USENIX Association, pp. 24–24.
- [29] SEKAR, V., RATNASAMY, S., REITER, M. K., EGI, N., AND SHI, G. The middlebox manifesto: enabling innovation in middlebox deployment. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks* (2011), ACM, p. 21.
- [30] SHERRY, J., HASAN, S., SCOTT, C., KRISHNAMURTHY, A., RATNASAMY, S., AND SEKAR, V. Making middleboxes someone else’s problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 13–24.
- [31] SHUBA, A., LE, A., GJOKA, M., VARMARKEN, J., LANGHOFF, S., AND MARKOPOULOU, A. Antmonitor: Network traffic monitoring and real-time prevention of privacy leaks in mobile devices. In *Proc. S3 Workshop* (2015).
- [32] TECHNOLOGY, C. C. M. Supersu. <http://www.supersu.com/>, 2016.
- [33] VALLINA-RODRIGUEZ, N., SUNDARESAN, S., RAZAGHPANAH, A., NITHYANAND, R., ALLMAN, M., KREIBICH, C., AND GILL, P. Tracking the Trackers: Towards Understanding the Mobile Advertising and Tracking Ecosystem. In *Proc. of the Workshop on Data and Algorithmic Transparency (DAT)* (2016).
- [34] WU, D., CHANG, R. K. C., LI, W., CHENG, E. K. T., AND GAO, D. Mopeye: Opportunistic monitoring of per-app mobile network performance. In *Proc. USENIX ATC* (2017).