Leaky Blinders: Information Leakage in Mobile VPNs

Thijs Heijligenberg[1], Oualid Lkhaouni[2], and Katharina Kohls[1]

[1] Radboud University, Nijmegen, The Netherlands
{theijligenberg,kkohls}@cs.ru.nl
[2] Ruhr University, Bochum, Germany
oualid.lkhaouni@rub.de

Abstract. In the mobile domain, VPN applications promise an additional layer of protection for wireless connections. They offer users the choice to improve the security of their connections, however, we only have very limited knowledge about the technical implications that the shift from desktop to mobile applications brings. In this work, we conduct a quantitative analysis of selected Android VPNs and demonstrate how all of them leak packets during an active tunnel. We conduct these measurements for different phones and in varying use case scenarios, including the comparison of Wi-Fi and 4G connections, to get a better understanding of how the mobile setting influences the security of a VPN. While we observe leaks in *all* combinations, some settings particularly cause the transmission of thousands of unprotected packets. We further conduct a series of case studies to provide some first insights on the *causes* for the observed leakage.

Keywords: VPN · Mobile · Information leakage

1 Introduction

VPN applications provide an additional layer of protection to Internet connections. The market size of VPNs is expected to grow from 25 million USD in 2019 up to 75 million USD in 2027 [17], indicating a high commercial potential. VPNs are beneficial in various use case scenarios ranging from casual convenience, e. g., circumventing geofencing for online content, over adding more security to standard Internet connections, to gaining protection from censorship authorities [19]. While conventional VPN applications are well-researched in the context of desktops [13,15,9], we only have limited knowledge about their performance in the mobile domain.

Mobile devices introduce constraints that are not present when running a VPN application on a desktop computer. In most cases, a mobile device has less computational power. It runs on a battery and uses a wireless network connection to either a Wi-Fi or a mobile network. All of these constraints have in common that they limit the performance of an application, e. g., optimizing the

battery usage makes active applications and processes compete for the available resources. Due to these constraints, we assume consequences for *mobile* VPN applications, especially concerning their power and data consumption. Prior work provides different static and dynamic analyses of Android-based VPN apps [8]. However, they mainly focus on the characteristics of the various applications and ignore influencing factors that are critical for the mobile domain. The results of these studies indicate different security issues involving malware in officially accessible Playstore apps, occurrences of TLS interception, or information leakage due to the transmission of untunneled traffic. However, these results are obtained through stationary setups that ignore the critical characteristics in the mobile domain. Examples of this are operating devices without any power connection or connecting to a mobile network instead of a Wi-Fi connection. Both characteristics are directly related to the power and traffic optimizations of a mobile device, and we must assume an impact on the behavior of mobile VPN apps.

In this work, we analyze Android VPN apps with a focus on the key influencing factors of the *mobile domain*. More precisely, we test three popular apps (Turbo VPN, Thunder VPN, Orbot) on three mobile devices that represent different ages of smartphones. Our primary focus is on information leakage, i. e., traffic that leaves or reaches the device without being protected by the active VPN tunnel. We use leakage to indicate how well an application handles a use case scenario in our experimental setup. Our quantitative evaluation demonstrates that *all* apps and devices in our setup transport unprotected traffic while the VPN is supposed to be active. This leakage can result in thousands of unprotected packets that carry potentially sensitive information.

Our setup covers various use case scenarios and combinations of influencing factors. Besides comparing devices and the individual VPN applications, we further investigate the differences between devices with and without an available USB power supply. In another experiment, we evaluate the differences between a network connection through Wi-Fi versus a mobile network connection using 4G. We apply these combinations to typical usage scenarios that resemble varying types of user data traffic, including simple browsing under varying link qualities or data-intensive streaming of multimedia content. Our case studies support our initial assumptions and indicate that the internal performance optimizations affect the security of mobile VPNs.

Our experimental evaluation prepares different starting points for future work on the performance of VPN apps in the mobile domain. This includes various technical aspects derived from our quantitative and qualitative evaluation findings, e. g., the internal resource optimization under varying constraints or the internal dependencies of operating systems and VPN apps that require a proxy interface for tunneling traffic. Finally, we point out user discrepancies that can arise from the consistent leakage of all apps in our test set. Overall, we make the following core contributions.

1. Quantitative analysis: We test three commercial smartphones and three Android-based VPN apps in four scenarios. We use the results of these experiments to compare the traffic leakage under various influencing factors.

2. Qualitative analysis: In a smaller set of targeted measurements, we investigate the performance of VPN apps in characteristic setups, including a lockdown setting and DNS specifics.
3. We provide a detailed discussion of our technical findings and identify different starting points for future work in this context.

2 Technical Background

A Virtual Private Network (VPN) tunnels traffic through a TCP or UDP connection between the user's system and a remote network, which allows access to services and devices in the remote network. Optional encryption through IPSec [6], TLS, or Wireguard [3] provides an additional level of confidentiality.

2.1 Mobile devices

Mobile devices use the same architecture of processes, network devices etc. as home computers, e. g., most mobile devices run on a version of the Linux kernel [7]. The fact that there are different underlying components for Wi-Fi, mobile data, or Bluetooth is abstracted away from applications running in userspace.

Differences to Desktops Mobile devices are typically more resource-constrained than laptops or desktop computers. High screen resolutions and intensive applications draw power and produce heat, and applications differ by their performance requirements [14]. In addition to the graphical processing and the CPU in the phone's chipset, there are different chips for Wi-Fi or mobile connections that increase the overall load on the available resources. On the network side, mobile devices receive a higher number of unsolicited incoming connections, mainly from mobile network sources. While these incoming connections are technically also established by the mobile device after an indication from the network, the phone user has little control over this in practice.

Based on the particular characteristics of the mobile domain, we expect differences in the performance of mobile applications, including VPNs. Due to the additional layer of encryption and the tunneling of user traffic, a VPN introduces a significant overhead for the device. Limited resources might affect the performance of such apps, which eventually leads to security flaws.

Mobile VPN Apps The Android operating system allows apps to register as a VPN by creating a *VpnService* [1]. The developer can provide an IP address of the tunnel endpoint, a route for the traffic (generally expected to be the default route for all traffic), and a DNS server. After establishment, this gives the app a filehandle from which reading equates to getting a message sent into the tunnel, and writing equates to sending a message out of the tunnel. The Android API also allows the application to set allowed or disallowed apps explicitly or enable apps to bypass the VPN.

Modern Android versions provide two additional settings that the user can set for an individual VPN from the settings menu: the option to always have the VPN enabled and to block traffic outside the tunnel, which in the android source code is referred to as *lockdown*. The Android operating system handles these settings, but apps must provide specific functionality relating to device startup to allow the always-on option to work.

2.2 Networks

Mobile devices use either an available Wi-Fi connection or refer to the mobile network made available through an active data plan with one of the network operators. While this wireless Internet connection is transparent to the user, the internal connection handling introduces some technical differences.

Wi-Fi Wi-Fi networks in their most common simple form, as provided by a Raspberry Pi 4 in our test network (§3), are a transmission modem and a gateway behind a NAT (Network Address Translation). The modem is responsible for communicating with the connected devices, which have been appointed individual IP addresses in the local network. The gateway aggregates all traffic and forwards it to the Internet or the encompassing network.

The operating system implements its protocol stack to handle Ethernet connections on the mobile device. The underlying Wi-Fi protocol stack is implemented in either the application processor or a dedicated Wi-Fi chip and resembles the same reference model as other network devices.

Mobile networks Mobile networks consist of multiple base stations and the core network. A base station handles the radio connection with connected mobile devices and sends/receives the traffic to/from the core network. The core network provides a link to the Internet and manages the phone's registration status. It also includes identity management and cryptographic procedures, and handles mobility between base stations or gateways.

On mobile devices, a separate baseband processor implements the mobile network stack. In the case of a mobile network connection, this stack takes over the processing of traffic on the network layer and handles the traffic from that point on.

3 Experimental Setup

The main focus of our analysis is on the volume of leaked traffic during an *active* VPN connection.

3.1 Network Setup

The network setup describes how we provide a wireless access point for the devices under test (cf. Figure 1). This access point is either a Wi-Fi access point (§3.1), or serves as a 4G (LTE) mobile network base station (§3.1).

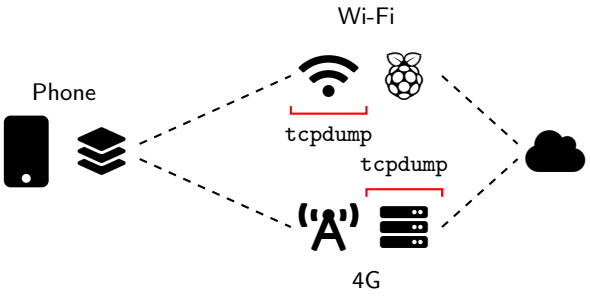

Fig. 1. Experimental setup. The phone connects through a VPN to a wireless access point. We capture traffic at the wireless interface (Wi-Fi) or the core's Internet gateway (4G).

Table 1. Devices and Capabilities.

Device				Experiments			
Name	Short	Android	Chipset	Wi-Fi	4G	Unplugged	Lockdown
Oneplus 8	O8	11	SM8250	●	●	●	●
Poco F2 Pro	PO	10	Qualcomm SM8250	●	●	○	●
Samsung Galaxy S9	S9	10	Exynos 9810	●	●	○	●

Wi-Fi Setup For the Wi-Fi setup, we use a Raspberry Pi 4 that offers network access through a `hostapd` service. The Pi is connected to the Internet and provides a dedicated Wi-Fi network for the devices in our test set. To record traffic, we run `tcpdump` on the wireless interface of the access point and save the resulting PCAP traces on a second machine to avoid any additional file writing load on the Pi. We have complete control over the access point, and we can adjust the network link according to our scenarios.

4G Setup The wireless access point is a base station for our mobile network setup and offers connectivity through a mobile network. To this end, we use an Amarisoft callbox classic [2] configured to a 4G setup. The callbox is connected to the Internet and offers connectivity through the core network's serving gateway. More precisely, the mobile network serves as a NAT providing the connected device with an internal IPv4 address and making requests through its external address. We record traffic using `tcpdump` at the *external* interface of the callbox and apply traffic shaping to the internal interface if necessary for a scenario.

3.2 Devices and App Setup

Our experiments cover devices released between 2018 and 2020, as summarized in Table 1. These devices differ in their hardware capabilities and enable us to analyze the performance of VPN apps on different host machines. All devices in

Table 2. VPN Apps used

App	Version	Downloads	Protocol
Turbo VPN	3.7.4	100m+	`ESP`
Thunder VPN	4.1.2	50m+	`SSL`
Orbot	16.5.2	10m+	`TCP`

our test setup are capable of Wi-Fi and 4G (LTE) connections. We control each phone using ADB, either via USB (in the plugged experiments) or via a network connection (in the unplugged experiments).

The Xiaomi and Samsung provide a relatively clean setup with only a handful of apps installed or activated, representing a realistic but controlled environment. The Oneplus has around 300 popular apps installed to simulate a setting where background traffic and heavy CPU load can occur.

Each device installs the same set of VPN apps as specified in Table 2. The apps in our setup represent different popular choices of free VPN services, with two offering a paid premium option (we refer to the basic service). Despite some conceptual and technical differences, all apps have in common that they route network traffic through at least one additional proxy on the transmission path (cf. §2).

3.3 Parameter Setup

We test three different devices and applications in four use case scenarios. If not noted otherwise, each experiment covers *all* 36 combinations of these parameters. We apply these combinations to different setups that focus on relevant influencing factors.

Scenarios The four different scenarios in our setup represent individual use cases that result in characteristic user data traffic.

Reference. The reference scenario covers simple web browsing where we open the Alexa top 10 websites in individual tabs of the device's standard browser and wait 1 second between new page loads. This setup serves as a reference with a moderate amount of traffic generated. We use the same browsing procedure in the following *link failure* setups.

Link Failure. In the *link failure* scenarios, we artificially add *delay* or *loss* to the transmission link. To this end, we use different combinations of the Linux traffic control settings `tc`. In the Wi-Fi setup, we apply the delay and loss rules to the wireless interface of the Raspberry Pi; in the mobile setting, we select the internal tunnel interface of the callbox. The tunnel interface represents the internal address of the serving gateway, i. e., the gateway that a mobile phone uses for an outbound connection from the core network to the Internet. The assumption behind this scenario is that we force the device into compensating for the lost or delayed traffic, e. g., through initiating retransmissions of packets.

Stress. The *stress* scenario aims to create a high overall burden for the device. This is achieved by opening a webpage with eight embedded 8K videos, resulting in the network link being fully used. The assumption behind this scenario is that by interfering with the limited resources of a device, eventual optimization steps by the operating system might affect the performance of the VPN. Furthermore, the high amount of traffic might affect the internal policies of a VPN app.

Influencing Factors We identify two key influencing factors that are characteristic of the mobile domain. To get a better understanding of these factors, we apply them to *all* of the above combinations. For example, to compare different network setups, we conduct the 36 permutations of devices, apps, and scenarios on a Wi-Fi and a 4G setting, resulting in 72 experiments in total.
Network Setup. We test two different variants of wireless network access. The device connects to our access point and receives an Internet connection through the wireless interface in the Wi-Fi setting. We connect the device to our mobile network in the LTE setting. The assumption behind this comparison is to vary the received signals and trigger the operating system's optimization of the traffic consumption.
Power Supply. As mobile devices only have a limited battery capacity, the power consumption of different apps and services is the target of optimization. To cover the differences between mobile and stationary usage of a device, we conduct experiments in both a plugged-in and plugged-out setting. The assumption behind these two variants is that energy optimization might also affect a VPN app and lead to side effects for tunneled traffic.

4 Dynamic Analysis

In our dynamic analysis, we look at the volume of leaked packets in different combinations of our parameters and setups.

4.1 Metrics

Our primary focus is on the amount of leaked information, i. e., the number and volume of packets that are processed outside the tunnel while a VPN app is active. We define and measure leakage as follows. When a device starts the VPN app, a `VPN connection established` message is sent to the Android log. We take this event as the starting point of the VPN and note the start time. As soon as the VPN app terminates, the log documents a `VPN disconnected` message. We use this timestamp as the end time. In the next step, we filter captured traffic and keep those packets sent between the recorded start and end times. We assume this to be the window in which the VPN tunnel is supposed to be active.

To determine leakage, we then document all IP addresses that transmit traffic using the protocol used by the VPN app (cf. Table 2). All traffic sent to and

received from these addresses is considered VPN traffic, and all other transmissions (IP address ≠ VPN IP) are untunneled traffic. The main metric of interest is the leak's volume, i. e., the sum of the sizes of all packets outside the tunnel. We document the leakage relative to the overall volume of traffic sent while the VPN is active.

We conduct five repetitions for all experiments for each parameter combination and aggregate the results accordingly.

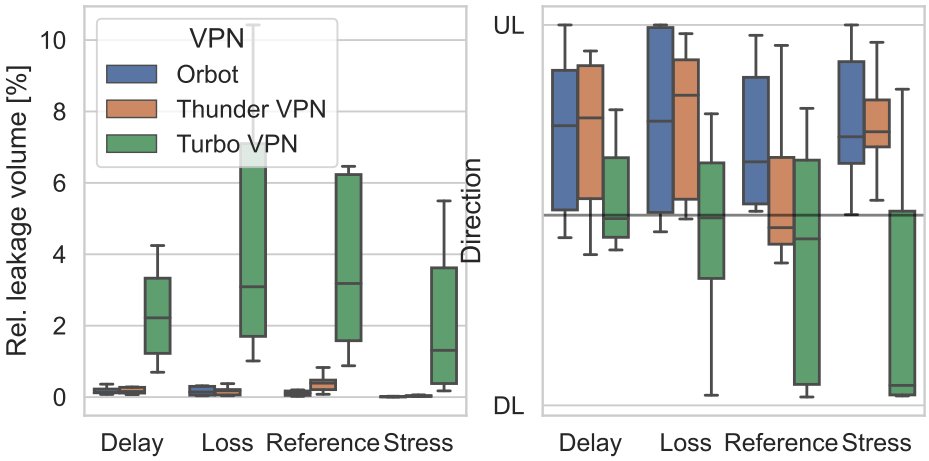

Fig. 2. Relative leakage (left) and distribution of traffic up/downlink (right). We compare the performance of the three VPN apps in our test setup.

4.2 Experiments

Our experiments provide a step-by-step analysis of the different influencing factors. We first give a general comparison of apps and continue with devices, power supply, and network connection. As Turbo VPN has a significantly higher leakage than the other apps, we separate all results after inspecting the apps.

Apps Our experiments cover three popular (by download numbers) VPN apps from the Android Play Store. While some apps compensate for the free usage model with advertisements (Turbo VPN, Thunder VPN), the Orbot app provides its core service without any additional content. For our measurements, we focus on the free variants of the apps and compare their performance in our set of four use case scenarios.

Figure 2 summarized the relative leakage per app and further documents the transmission direction of leaked traffic. In our results, it is evident that Turbo VPN causes a significantly higher leakage. While the other apps provide a much

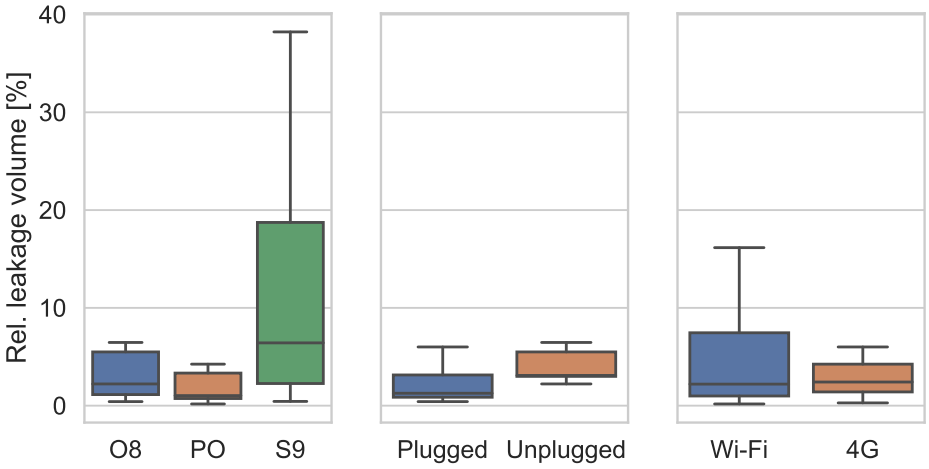

Fig. 3. Relative leakage of Turbo VPN traffic. Plots show the comparison of different devices (left), power supply (middle), and network connections (right).

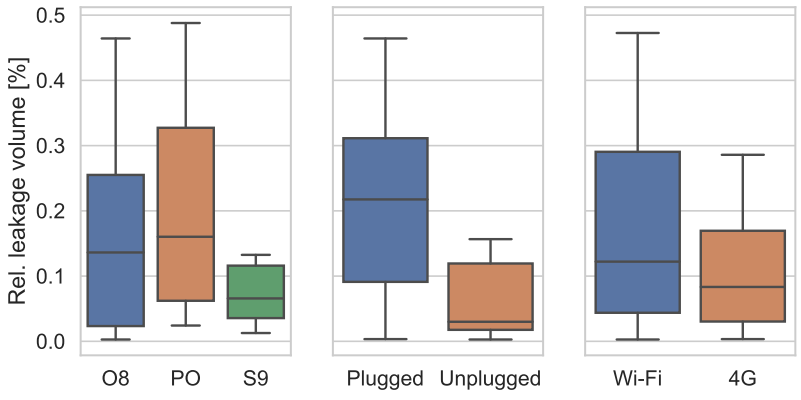

Fig. 4. Relative leakage of Orbot and Thunder VPN traffic. Plots show devices (left), power (middle), and network (right); results are merged for both apps.

lower number and volume of leaked packets, it is worth mentioning that *all* experiments (combinations and repetitions) contain leakage.

We further observe that Turbo VPN traffic consists of significantly more downlink traffic. This indicates that Turbo VPN constantly receives traffic, even though the app operates in the same settings as other candidates. To avoid any bias in the next steps of our evaluation, we separate the results of Turbo VPN from the other apps. This allows us to discuss the factors that influence the heavy leakage in Turbo VPN while also allowing us to look at the results of the other apps without the outliers introduced by Turbo VPN.

Devices The devices in our test setup provide different hardware specifications and were released during the last four years. Consequently, we expect individual performance characteristics in the use case scenarios that we vary in our experiments. We expect that devices with an overall lower RAM and CPU power availability might introduce effects that are represented in the observed leakage, e. g., in the case of processes competing over the available resources.

Figure 3 (Turbo VPN, left) and Figure 4 (Orbot, Thunder VPN; left) summarize the performance for the different devices under test. Our results indicate that the median leakage for Turbo VPN is higher and that the S9 experiences a series of outliers. We attribute this to the overall weaker performance of Turbo VPN and the weaker hardware capabilities of the S9. No significant outliers are visible for the other apps, and we observe leakage of less than 1%. We conclude that Turbo VPN introduces outliers, while we leave a detailed evaluation of the S9 internals in this setup to future work. Future work should continue with a detailed analysis of the app internals and how they interfere with the OS.

Power We compare the performance differences for plugged and unplugged setups, e. g., the devices receive power via USB or are unplugged from any power source. In these experiments, we limit ourselves to the Oneplus 8. We perform a full test of scenarios for the Wi-Fi setup.

Again, we observe how Turbo VPN differs from the results of the other apps, as it indicates a higher leakage for the unplugged setting. In all other cases, a constant power supply leads to more unprotected traffic. We assume that battery power leads to a higher degree of performance optimizations in the OS, which eventually leads to more constraints for the VPN apps. Future work should investigate OS optimizations and how they affect different apps, e. g., categorized by access and interface usage.

Network Next, we analyze the differences between a Wi-Fi and a 4G mobile network setup. While we do not expect significant differences on the packet level (the underlying network connection is invisible to the VPN app), mobile traffic might lead to a different optimization strategy for the operating system. An example of this is minimizing mobile traffic consumption for specific use cases.

In our experiments, the use of 4G resulted in less leakage. The hypothesis can explain that the phone is more stringent with giving resources to background

processes while leaving the VPN app untouched. Future work should continue this by monitoring the power consumption and device battery status.

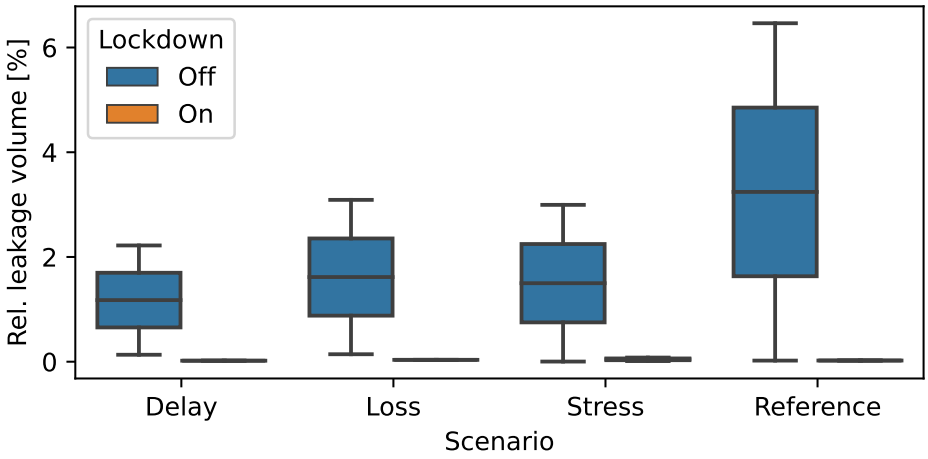

Fig. 5. Comparison of different scenarios with the lockdown option disabled and enabled.

4.3 Case Study: Lockdown option

In addition to the quantitative evaluation of the above parameter combinations, we conduct targeted case studies for specific VPN characteristics. We begin with the lockdown function that blocks all traffic outside a VPN tunnel. Figure 5 provides a comparison of scenarios with and without the lockdown function enabled. We observe that the option drastically reduces leakage in all scenarios. However, none of the setups yielded truly zero leakage, and further research is required to investigate what information the leaked packets contain. We did not test Thunder VPN in this variation as it does not support the prerequisite *always-on* option.

4.4 Case Study: DNS traffic

A portion of the leaked traffic consists of DNS messages bypassing the VPN. As a second case study, we test whether a domain responds to a DNS request. This gives us a strong indication that this domain serves as a name server. In all our experiments, the combined DNS traffic amounted to around 1% of traffic by packet count. This indicates that while DNS is part of the leaked traffic, it is not the leading cause. Unprotected DNS messages, when sniffed, can lead to a privacy leak as they do give a hint on the user's behavior, but further research is required to ascertain how impactful the DNS traffic leaked from the DNS apps is.

Conclusion. From the analysis of different influencing factors, we can conclude that the choice of a VPN app serves as an amplifier for the observed characteristics. All setups have in common that they do not function *without* any leakage, and even the otherwise effective lockdown function does not entirely prevent this. Less reliable VPN apps lead to more downlink traffic, which indicates that incoming traffic is less likely to be protected by the tunnel. Our evaluation is a first starting point and helps identify future work directions. In particular, the dependencies between operating systems and applications should be analyzed in settings characteristic of mobile usage.

5 Directions for Future Work

Our work indicates that the mobile domain introduces different influencing factors that affect the performance of VPN apps. This underlines the initial assumption of facing specific challenges that can affect the security of an App and, eventually, the privacy of its users. From our findings, we derive important directions for future work.

5.1 Android Internals

To avoid the requirement of root privileges, mobile VPN apps are limited to using the API that Android exposes for tunneling traffic. At the same time, the internal baseband and application processors implement the different network stacks responsible for processing the incoming and outgoing traffic. As soon as network effects like delays or packet loss affect the connection, these parts of the operating system must respond to the incidents. This directly affects the performance of the VPN apps. Future work should investigate how apps and operating systems interact and how different factors like the power supply or network connection influence the security of a VPN.

5.2 VPN App Internals

Similar to the impact of the operating system, the specific implementation of an app influences how different use case scenarios can be handled. As we observed significant differences between the VPN apps in our experiments, a valuable next step is an evaluation of the internals of such apps. While prior work already covers static analyses, a dynamic approach would allow for covering relevant characteristics of the mobile domain.

5.3 User Expectations

Besides the technical aspects of apps and their host devices, users' expectations are a critical factor in assessing the security of mobile VPNs. Our experiments show constant leakage for all combinations of scenarios and parameters. Although the relative amount of leaked packets can be low, all untunnelled data

has the potential to carry sensitive information. This might cause discrepancies with user expectations and emphasizes the fact that a mobile OS is mostly opaque to users [11]. Future work must analyze such user expectations, the technical understanding of VPN apps, and differences between the perception of mobile versus desktop VPN solutions.

6 Related Work

Prior work provides an extensive measurement study on more than 200 commercial VPN providers that involves static and dynamic analyses of the apps and the resulting traffic[8]. While their work demonstrates how the VPN permissions of Android facilitate different kinds of information leakage or connection manipulations, this existing study only covers a stationary, single-device setup using a Wi-Fi connection. In contrast, our work focuses on the specific characteristics of a *mobile* setting. More precisely, we incorporate different network setups and the capabilities of different classes of devices, all in the presence of individual use case scenarios. Our results underline the assumption that these influencing factors affect the performance of a mobile VPN and should not be isolated in quantitative analyses.

There are different reasons to use a VPN application. Tunneling traffic through a trusted party can help to improve the overall privacy, e. g., by avoiding tracking [4] or fingerprinting [16] attacks. Another use case scenario is the circumvention of Internet censorship, where the authorities of a jurisdiction limit the access to certain services and information sources on the Internet. Prior work monitors worldwide incidents of Internet censorship [18] and investigates ways to circumvent the resulting limitations [10,12].

A different line of work focuses on the manual inspection of popular VPN services regarding their network characteristics or the infrastructure of the different providers. Results indicate misconfigurations and bugs that lead to leakage of DNS and IPv6 information [13,5]. Follow-up work repeats this with a specific focus on VPN applications in the Android ecosystem [8,20]. These studies show different instances of malware, manipulation of connections and TLS interception, or traffic leakage. The findings of these studies were later confirmed [9].

7 Conclusion

VPNs offer an additional layer of protection for user traffic. While such apps are increasingly popular, we only have limited knowledge about the implications of switching from stationary to mobile settings. Optimization strategies of operating systems that aim to reduce the battery usage and consumed traffic when connected to a mobile network might impact the protection capabilities of a VPN, leading to traffic and, thus, information leakage. In this work, we analyze the information leakage of Android VPN apps in different use case setups to assess the impact of critical characteristics of the mobile domain. Our results indicate that in *all* combinations of devices, apps, and scenarios, a certain amount

of traffic remains unprotected by the tunnel. In some cases, the combination of influencing factors leads to thousands of leaked packets. Our results indicate different directions for future work and emphasize the need to consider the unique aspects of mobile VPN usage.

References

1. Vpnservice reference (2022), https://developer.android.com/reference/android/net/VpnService
2. Amarisoft callbox series (2022), https://www.amarisoft.com/products/test-measurements/amari-lte-callbox/
3. Donenfeld, J.A.: Wireguard: Next generation kernel network tunnel. In: NDSS. pp. 1–12 (2017)
4. Englehardt, S., Narayanan, A.: Online tracking: A 1-million-site measurement and analysis. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 1388–1401. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978313, https://doi.org/10.1145/2976749.2978313
5. Fazal, L., Ganu, S., Kappes, M., Krishnakumar, A.S., Krishnan, P.: Tackling security vulnerabilities in vpn-based wireless deployments. In: 2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577). vol. 1, pp. 100–104. IEEE (2004)
6. Frankel, S., Kent, K., Lewkowski, R., Orebaugh, A.D., Ritchey, R.W., Sharma, S.R.: Guide to ipsec vpns:. (2005)
7. Globalstats statcounter "mobile operating system market share worldwide" accessed 4-2-2022 (2022), https://gs.statcounter.com/os-market-share/mobile/worldwide
8. Ikram, M., Vallina-Rodriguez, N., Seneviratne, S., Kaafar, M.A., Paxson, V.: An analysis of the privacy and security risks of android vpn permission-enabled apps. In: Proceedings of the 2016 internet measurement conference. pp. 349–364 (2016)
9. Khan, M.T., DeBlasio, J., Voelker, G.M., Snoeren, A.C., Kanich, C., Vallina-Rodriguez, N.: An empirical analysis of the commercial vpn ecosystem. In: Proceedings of the Internet Measurement Conference 2018. pp. 443–456 (2018)
10. Khattak, S., Javed, M., Khayam, S.A., Uzmi, Z.A., Paxson, V.: A look at the consequences of internet censorship through an isp lens. In: Proceedings of the 2014 Conference on Internet Measurement Conference. pp. 271–284 (2014)
11. Liu, B., Andersen, M.S., Schaub, F., Almuhimedi, H., Zhang, S.A., Sadeh, N., Agarwal, Y., Acquisti, A.: Follow my recommendations: A personalized privacy assistant for mobile app permissions. In: Symposium on Usable Privacy and Security (2016)
12. Nobori, D., Shinjo, Y.: Vpn gate: A volunteer-organized public vpn relay system with blocking resistance for bypassing government censorship firewalls. In: 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). pp. 229–241 (2014)
13. Perta, V.C., Barbera, M., Tyson, G., Haddadi, H., Mei, A., et al.: A glance through the vpn looking glass: Ipv6 leakage and dns hijacking in commercial vpn clients (2015)
14. How much ram do you need in a smartphone? (2019), https://www.androidauthority.com/how-much-ram-do-you-need-in-smartphone-2019-944920/

15. Ramesh, R., Evdokimov, L., Xue, D., Ensafi, R.: Vpnalyzer: Systematic investigation of the vpn ecosystem. In: Network and Distributed Systems Security. NDSS' 22, isoc (2022)
16. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated Website Fingerprinting through Deep Learning. In: Network and Distributed System Security Symposium. NDSS '18, The Internet Society, San Diego, CA, USA (Feb 2018)
17. Statista: Size of the virtual private network (vpn) market worldwide in 2019 and 2027 (2022), https://www.statista.com/statistics/542817/worldwide-virtual-private-network-market/
18. Sundara Raman, R., Shenoy, P., Kohls, K., Ensafi, R.: Censored planet: An internet-wide, longitudinal censorship observatory. In: In ACM SIGSAC Conference on Computer and Communications Security (CCS) (2020)
19. Wired: The attack on global privacy leaves few places to turn (2017), https://www.wired.com/story/china-russia-vpn-crackdown/
20. Zhang, Q., Li, J., Zhang, Y., Wang, H., Gu, D.: Oh-pwn-vpn! security analysis of openvpn-based android apps. In: International Conference on Cryptology and Network Security. pp. 373–389. Springer (2017)