Free By Design: On the Feasibility of Free-Riding Attacks Against Zero-rated Services

Julian Fietkau, David Pascal Runge, and Jean-Pierre Seifert

Berlin Institute of Technology, Germany
{fietkau, david.p.runge, jpseifert}@tu-berlin.de

Abstract. With free-riding attacks against zero-rated services, an attacker can avoid expenses by transmitting wrongly labeled zero-rated network traffic. Previous research has shown that free-riding attacks can affect network quality and increase costs when applied on a large scale. Hence, various solutions have been proposed to protect networks against this growing threat. While these protections can cure some of the symptoms, they do not solve the underlying issue. In this paper, we argue that secure web services with high bandwidth requirements combined with zero-rating can increase the chances of free-riding attacks. Therefore we show how tunnel-based free-riding attacks can be designed and implemented for secure services, such as instant messaging, cloud storage, and video chats. Furthermore, we evaluate these tunnels in terms of usability and performance and judge their feasibility. In addition, we also discuss the existing countermeasures and how they can be improved. As part of our work, we want to point out that free-riding attacks are a self-imposed problem created by Internet Service Providers that try to create artificial walls within a system that is free and open by design, such as the Internet. To counter this issue, we publish the created tools to support those that are opposed to any form of zero-rating, censorship, and other forms of traffic discrimination.

1 Introduction

Internet access is a crucial aspect of modern life. Consequently, mobile Internet access is established in many parts of the world, and infrastructure costs are shared among users depending on how much everyone is using. Zero-rating is the practice of Internet Service Providers (ISPs) to either exclude traffic of specific applications from the user’s data plan or even offer specific applications for free [14]. Various types and business models have become established in recent years. These can range from self-sponsored zero-rating, where an ISP independently decides to zero-rate a service, up to compound zero-rating, where multiple Content Providers (CPs) bundle their services into one zero-rated product of an ISP [3]. However, each practice of zero-rating is in heavy conflict with the idea of net neutrality [26]. As defined by the EU, the idea of net neutrality is to ensure equal and nondiscriminatory Internet access among all Internet users. This way, innovation without obstacles can be preserved, and discrimination by the ISPs
can be prevented. Without net neutrality, a Content Provider (CP) could pay an ISP to prioritize its traffic to gain an advantage over its competitors.

One of the most prominent examples for zero-rating is the Free Basics initiative by Facebook [9]. This service aims to connect the world with essential online services, especially those who can not afford internet access so far. As of April 2018, 100 million people were using internet.org [9]. While the goal of the Free Basics service appears to be noble, the freely accessible service is very limited to a specific set of services and does not include Facebook’s competitors. The initiative drew much criticism for this, and the Indian telecom regulator has even banned the service, seeing it as digital colonialism [24]. Moreover, other examples are not too different. For instance, Deutsche Telekom has offered its Stream On service until the EU court (EuGH) classified this practice as illegal [18]. Today, the debate about net neutrality is still not settled and will keep us busy for some years to come also, because traffic prioritization is now even part of 5G, the next generation of mobile networks [26].

Another way to think about zero-rating services is to use zero-rated traffic to enable free and unlimited Internet access via free-riding attacks. As the name suggests, the goal of a free-riding attack is to avoid the billing of Internet traffic by ISPs. In the context of zero-rating, an attacker tries to trick the ISP to zero-rate traffic that would be charged otherwise. Recent works on free-riding attacks against zero-rated services show that numerous attack approaches exist [10, 14, 19, 21]. Furthermore, the damages by some of these attacks have already reached a critical level. For example, with a single free-riding attack against a Chinese ISP, damage of half a million US dollars loss has been created by creating 71 TB of wrongly labeled traffic [14]. These numbers show that free-riding attacks are real threats that ISPs and CPs should consider. As we see i) free-riding attacks are a real threat that is growing ii) they cause financial damage to network and content providers and impair network quality; and iii) the next generation of mobile networks will amplify this evolution by standardizing traffic prioritization.

This work will explain how zero-rated services work and present a threat model for free-riding attacks. Based on that, we analyze the different types of zero-rated services to implement our free-riding attacks. This includes attacks via instant messaging, cloud storage, and VoIP services. Afterward, we evaluate the different approaches with respect to usability and performance. According to that, we will discuss existing countermeasures and their effectiveness. In summary, this paper makes the following contributions:

- We demonstrate the feasibility of tunnel-based free-riding attacks against three secure Internet services.
- We evaluate the presented free-riding attacks with respect to usability and performance with real-world applications.
- We discuss the effectiveness of existing countermeasures against free-riding attacks and explain why they fail.
- We point out how the concurrent adoption of zero-rating and secure web services with large bandwidth requirements can amplify free-riding attacks.
- We make our research tools and data accessible via:
  https://bit.ly/freebydesign
2 Background

This section introduces the necessary knowledge about free-riding attacks, zero-rating services, net neutrality and introduces the most related work in this field.

2.1 Zero Rating Services

Zero-rating is ”a practice that exempts Internet traffic generated through certain applications or access to certain websites from usage charges” as defined by the European Commission [1]. Technically, this is realized by excluding the zero-rated traffic from the customer’s data plan. In some cases, zero-rating will even ”allow users to access content without having subscribed to a data plan at all” [1]. The various types of zero-rating are largely discussed in two publications by the European Commission [1] and in the work of Carillo [3]:

– Single-Service Zero-Rating: The ISP offers free access to one site or service. The CP has a contract with at least one ISP, e.g., Facebook Zero [23] or Wikipedia Zero [27].
– Sponsored Zero-Rating: The CP pays the ISP to offer its services at no charge, e.g., T-Mobile’s Music Freedom, where some music streaming services are zero-rated [25].
– Compound Zero-Rating: A CP bundles services with an ISP, payments are not automatically implied, e.g., the Free Basics Service by Facebook, which includes Facebook, healthcare, news, and weather services [8].
– Faux/Non-Selective Zero-Rating: The CP partners the ISPs to exclude specific action from the user data volume, e.g., download specific apps, advertisements, or order a specific product such as data volume.

2.2 Net Neutrality

Net neutrality describes the idea to ensure equal and nondiscriminatory treatment of all Internet traffic. Accordingly, ISPs should treat every communication equally and not discriminate ”based on IP addresses, domain name, cookie information, TCP port, and others” [28]. One can summarize net neutrality as the idea of treating every packet equally [22]. Without it, an ISP can throttle, prioritize or charge connections arbitrarily. Net neutrality is a fundamental property for fair competition and innovation. When not enforced, wealthy CPs can pay ISPs to prioritize or charge their traffic differently than a competitor’s traffic. This will, above all, harm new CPs with new and alternative services. On the other hand, net neutrality is in the way of efficient traffic management and can even create inequality between customers and their different data demands [11]. In the end, net neutrality and zero-rating contradict themselves because the ISP treats the traffic differently in terms of costs, bandwidth, or latency.
2.3 Free-riding attacks

Free-riding attacks try to avoid the billing of Phone and Internet traffic. The first Free-riding attacks date back to the 1950s when phones were manipulated to enable free long-distance calls. Back then, the first attacks replayed a 2600 Hz tone or the sound of dropping coins to avoid charging. Later on, the term "Phreaking" was coined [12], and after becoming too popular, countermeasures were implemented.

Today’s networks are very different and more complex, but the underlying issues are still the same. Internet traffic is billed on a volume basis and has a limited data allowance; hence attacks try to avoid these limits. For example, DNS tunnelings allow a user to get free internet access via Wifi-Hotspots of Deutsche Telekom [21] and some US network providers as shown by Peng et al. [19]. The traffic will be encapsulated in DNS messages, and sent to a tunnel endpoint with unlimited Internet access. The endpoint resolves the tunneled request and sends the answer back via DNS messages. Besides this technique, many other ways to attack ISPs are known, e.g., spoofed TCP retransmission headers [6] or manipulated VoLTE signaling messages [13].

The first attack against zero-rated services was built for T-Mobile’s BingeOn service and used manipulated HOST headers [10]. Based on this work, Liu et al. have developed more ways to do the same, for instance, by spoofing the Server Name Indication field [14]. As we see, various free-riding attacks have been built to establish free internet access by abusing the protocols that underlie those network services.

2.4 Countermeasures

With zero-rated services becoming more popular in recent years, free-riding attacks also got more attention; hence, industry and academia created several countermeasures.

One of the most recent solutions is ZFree, published by Liu et al. [14]. The solution proposes the cooperation of the CP, ISP, and an ISP assistant. Initially, the CP and the ISP assistant will establish a secure connection between each other. To check the network traffic for free-riding traffic, the CP will hash each network packet and send a message to the ISP over the origin network connection. In parallel, a hash of each packet is sent from the ISP to the ISP assistant. To verify the validity of the traffic, the ISP assistant compares the received hashes. As these hashes are generated at different places and sent over different channels, the packet’s integrity is considered protected by Liu et al. The authors claim to prevent each known injection attacks that are used for free-riding attacks and state that "ZFree is formally verified and secure against free-riding attacks [14]."

Another work of Dusi et al. [5] focuses on the detection of tunnel-based free-riding attacks. The paper describes a traffic fingerprinting mechanism to identify tunnels in network traffic. The core idea is to integrate a statistical characterization mechanism at the IP layer to characterize traffic by observing
network packets’ length and inter-arrival time. In their evaluation, the approach achieves a detection rate of almost 100% by analyzing HTTP and SSH traffic [5]. Since their approach works without DPI, it can be used by ISPs and CPs.

The presented frameworks propose to secure zero-rating against free-riding attacks. However, in Section 6 we will point out the fundamental misconception of these solutions when applied in the context of secure Internet services.

3 The Threat of Free-Riding Attacks

To built upon previous research, we have adopted the threat model created by Liu et al., describing mainly three parties and how they interact [14]:

- The user accessing the Internet and specific applications that are zero-rated. A user can profit if she avoids costs by exploiting zero-rated services with free-riding attacks.
- The ISP providing the Internet access and facilitating zero-rating on a technical level. The ISP profits if no user can execute a free-riding attack and they have to pay their bills. Thus, ISPs want to mitigate all free-riding attacks and increase their profits.
- The CP providing a zero-rated service that wants to improve its performance to profit from a growing user base. A CP is interested in zero-rating to outperform its competitors but needs protection from free-riding attacks.

In reality, things can be quite different, e.g., ISP and CP can be the same entity, or in other cases, ISP and CP do not want to protect each other. Hence, motivational aspects of establishing zero-rating practices play an essential role when looking at each party’s willingness to secure a service. With respect to the user role, several use cases can be conceived to substantiate the user’s motivation to execute an attack, e.g., when we imagine the following scenarios:

- A user in a developing country without mobile Internet access has zero-rated Facebook access via Free Basics [23].
- A user abroad downloads a large file over a limited data connection but has zero-rated access to a cloud service provider.
- A user on an airplane wants to browse the web but only has limited access to instant messaging service.
- A user in China cannot access foreign internet contents but has unrestricted access to a secure instant messaging service.
- A user has limited or restricted Internet access at work but unrestricted access to a VoIP call service.

As we see, various use cases for free-riding attacks can be conceived and might resonate with some of our own experiences. Moreover, free-riding attacks do not only exist to avoid billing but also due to political and economic restrictions.
4 Building Free-Riding Tunnels

This section will explain how tunneling can be utilized to build free-riding attacks via secure Internet services. We explain how tunneling works in general, what devices are used for a real attack setup, and how we implemented the tunnel for messaging, cloud storage, and VoIP services.

4.1 General Tunneling Technique

The core idea of tunneling is to encapsulate application traffic into another application’s traffic. This way, the tunneled application can be hidden from controlling network devices between the tunnel entry and endpoint. While only a specific protocol is allowed for the tunnel entry, the tunnel endpoint needs unrestricted or less limited Internet access.

To draw a more concrete picture, we can imagine a user accessing a WIFI hotspot. Without paying the hotspot provider, the HTTP traffic will be blocked. Nevertheless, DNS messages can be exchanged because they are necessary to establish communication. Hence, the real traffic can be encapsulated and sent to a self-hosted DNS service with unrestricted Internet access using DNS messages. This endpoint resolves the tunneled requests and sends the response to the user within a DNS response message.

This idea of tunneling data through other layers can also be applied to create unrestricted Internet access in mobile networks via zero-rated services. Like a DNS tunnel, the solution requires a specific setup that needs preparation.

4.2 Attack Setup

The attacker needs at least one device with some form of zero-rated access and a tunnel endpoint device, e.g., at home, with unrestricted Internet access. For our attacks, we used the following devices and services:

- 1x Samsung Galaxy S3 mini running Android 4.1.2.
- 2x HP Elitebook 840 (Intel Core i5-6300 (2.40 GHz), 16 GB DDR4 RAM, 256 GB SSD, Ubuntu 18.04.5 LTS)
- 1x pre-paid sim card with zero-rating options
- 1x 8 Mbit unrestricted DSL connection “at home”

These components will be arranged as depicted in Fig. 1. In our setup, all devices are located in our office. The mobile device has full 4G connectivity, and the home client is connected via cable to a 1 Gbit internet connection. Depending on the targeted service, a specific mobile network must be selected, and a SIM card must be available. In our case, we have chosen a Portuguese network provider that offers five zero-rating packages called: Video, Social, Music, Messaging, and Cloud. Moreover, the provider also zero-rate its own cloud, TV, and navigation services free of charge. To book one of the packages, the user needs to i) access a specific website of the network service provider, ii) enter his mobile
Fig. 1: A basic setup for a free-riding attack against a zero-rated service.

phone number, and iii) select a package. In our case, we selected the Messaging package for €4.99 per month, which includes 12 different messaging services, e.g., Skype, WhatsApp, Telegram, WeChat, etc. In addition, we booked the Cloud package for €4.99 per month, including Google Drive, Microsoft OneDrive, and Apple iCloud. In comparison, a general data plan costs €9.99 for 1 GB or €14.99 for 5 GB useable within a month. As requested by the network provider, we will not publish the name of the affected company.

4.3 Free-Riding Via Instant Messaging

In the following, we explain how to build a free-riding attack via a zero-rated instant messaging service. We use the Telegram chat in our example, which we chose for its open and simple API. To establish the communication via Telegram we used the TDLib (Telegram Database Library), a cross-platform library to automate Telegram clients. Based on the Telegram chat, we built a document-based HTTP proxy that can transport even large files of up to 1.5 GB. A general overview of our implementation is depicted in Figure 2.

Tunnel entry point The tunnel entry point is built out of two components: an HTTP proxy and a server application to send and receive messages over the tunnel. The HTTP proxy allows a user to access the tunnel. A user only needs to add a proxy to the browser configuration; after the tunnel is initiated, and the communication is routed automatically through the zero-rated tunnel. We implemented the proxy using the Python Flask framework [17]. The tunnel interface is implemented using the python modul python-pty [16]. The user provides credentials for two user accounts of the corresponding tunnel entry and endpoint to initiate the tunnel. The login will be verified with the registered mobile device. Afterward, the tunnel interface will be assigned to a static IP. To connect the tunnel interface to the unrestricted Internet, the tunnel entry-point has to route the traffic accordingly. Usually, the default route goes to the tunnel interface and IP of the tunnel endpoint, which acts as a gateway. Additionally, another route needs to be added to pass the Telegram traffic through the regular connection. From this point on, all packets received at the tunnel interface are
sent to the sender. The sender encodes each message with Base64 and sends the instant messages via send_message() to the tunnel endpoint.

After a message is sent, the message handler waits for the corresponding answer. To receive a message, the message handler needs to be registered to Tdlib. The handler is called every time a new notification arrives via Telegram. This includes messages and notifications of other users’ online status or the creation of new chats. The implemented message handler filters all the relevant messages decodes them and passes them back to the tunnel entry.

To receive a message, the message handler is registered to Tdlib as well and will be called whenever a new notification arrives. At one point in time, a notification is pushed to the message handler by the instant messaging servers. In our case, the notification includes a caption and ID of the new message that needs to be fetched by the message handler of Tdlib using the downloadFile() function and the corresponding ID. When receiving the file, the message handler delivers the caption and document back to the HTTP proxy. The caption contains the HTTP response string, headers, and the document containing the payload of the HTTP response. In the end, the HTTP proxy compiles the valid HTTP response from this and pass it to the tunnel entry point.

**Tunnel end point** The tunnel endpoint components appear similar to the entry point. Instead of the HTTP proxy, it uses a request executor that forwards each HTTP request received from the tunnel interface. The execution is implemented using the Python’s requests module.

Based on the presented approach, any HTTP request can be passed between tunnel entry and endpoint to allow web browsing via a zero-rated Telegram chat. For our proof-of-concept implementation, we only support the HTTP GET method. Furthermore parallel, and out-of-order execution is not considered. To lower the number of requests over the tunnel, the request executor can resolve all external page elements of a website on its own. To do this, we request and merge all website elements into a single HTML file using the Python module webpage2html.
4.4 Free-Riding Via Cloud Storage

Another example for free-riding attacks against zero-rated services can be created via cloud storage services such as Google Drive. To build a tunnel, the Google Drive API needs to be automated, e.g., by using a dedicated Python library such as google-api-python-client [7]. To initiate the Google Drive API, a Google Cloud project has to be created. In this project, we activated the Google Drive API to retrieve our personal API key. The API is limited to one billion queries per day and 10 queries per second for each project user. Instead of sending direct messages, like in the previous solution, the two devices will communicate by uploading and downloading files through the cloud storage service. Hence, a shared storage location for the two user devices has to be created. We decided to use two separate accounts and two shared folders to reduce the risk of reaching the API limits. Each folder is assigned to one tunnel point and hence appears like a unidirectional communication channel. To send messages, a file is uploaded to the respective folder connected to the tunnel entry or endpoint. To receive messages, each tunnel point will regularly poll its respective folder for new content. Each new file that appears is considered as an incoming message. Thus, a simple communication through the cloud storage service has been established.

The implemented modules and how their interaction can be seen in Figure 3. In general, we have refactored large parts of our Telegram solution since both solutions are very similar. The main differences are an additional pull mechanism and adopting the new API functions and limits for the cloud storage service. For example, to send files, the files need to be prepared and created on the cloud storage beforehand, using files().create(). Like the HTTP proxy and request executor, other components did not change and work like described in the previous solution. The polling mechanism is implemented on both sides. It will send out eight requests per second, which is closely below the API limits and can be adjusted by the user. On occurring changes, the polling mechanism will notify and forward a file ID to a downloading component. It will download
the file with the `files().get_media()` function. Depending on which side of
the tunnel, the file will be forwarded to the request executor or passed to the
HTTP proxy. Similar to the previous solution, the created tunnel can be used
for HTTP communication, but also allows the user to download large media files
through the zero-rated connection.

4.5 Free-Riding Via VoIP Calls

In our third free-riding attack, we create a tunnel using the UDP payloads of a
VoIP call service without modifying the VoIP client. As requested by the service
provider, we can not publish the name of the affected product and company.

In general, VoIP calls will be executed via peer-to-peer connections. However,
when the user devices are behind a NAT service, a so-called TURN server will
relay the communication between the two parties. For our attack, we use this
communication to pass through the traffic by rewriting the contents of the sent
packets. The created application flow is shown in Figure 4.

The biggest challenge when creating the tunnel setup is to enable the two
endpoints to rewrite the package contents in an automated way, since both ends
need to know the IP and UDP port of the established connection. To do this, we
extract the connection details from the SIP messages exchanged during the call
establishment. On the caller side, the `REMOTE_ADDR` attribute of the STUN
messages is extracted, while for the callee, the `X-MAPPED-ADDRESS` attribute from
the `Allocate Success Response` message is used. This procedure can be dif-
ferent when using another service provider. To extract these details we use the
Python `scapy` package [20]. After collecting the required session details, the in-
jecting devices can manipulate parts of the UDP data stream and tunnel the
traffic through the VoIP connection. To differentiate between the modified and
the unmodified UDP packets, a 4-byte magic number is added to the payload.
Using `iptables`, the injecting devices can filter the network packets to find the
manipulated ones.

The tunnel entry and endpoints are very different from those of the previous
two approaches. The sender receives the packet from a tunnel interface and
forges a UDP packet with the corresponding TURN server IP, port, the magic
number, and the packet contents. Afterward, the sender sends the packet to the
TURN server. To improve the performance of the components, a raw socket is
used to send the packets, while `scapy` provides the packet forging template. To
receive packets, the receiver uses a socket as well. This socket is bound directly
to the network interface with a Berkeley Packet Filter (BPF). The BPF checks
for the appropriate IP addresses, ports, and identifier sequence. This way, the
injected traffic can be filtered. The BPF is implemented in assembly based on
an example created by Allan Boll [2]. Overall, the created interface allows very
efficient packet injections into the active VoIP connection to arbitrary tunnel
data through a zero-rated VoIP connection. Since VoIP applications need to be
robust, we assume that the manipulated packets are simply ignored and dropped
by the VoIP clients. However, even if the packets get validated by the client, our
tool supports a feature to manipulate only the media contents of a data stream.
Fig. 4: Layer 3 tunnel through a zero-rated VoIP service.

When establishing the tunnel interfaces, an application can be configured to use the created communication link. To implement this, we used QUICHE, an open QUIC implementation with a low-level API that already provides an easy-to-use example for our use case [4]. For the sake of comparability, we hosted a QUIC HTTP server on our home client that runs HTTP via UDP and can be accessed through the proxy with any compatible browser, such as Chromium.

5 Evaluation

After we presented how free-riding attacks against zero-rated services can be implemented, we now evaluate how the tunnels perform in terms of usability and performance.

For all experiments, we used the setup as defined in Section 4.2. However, the measurements will also depend on other factors, like network connectivity, devices, network stacks, ISP provider, communication distance, etc. Hence, the presented results can only give a limited impression of the tunnel performance. Another setup might cause other results. Nevertheless, the measurements allow us to compare the different approaches of the tunnel with each other and understand their limitations. The experiments described here took place between January and mid-February 2021.

5.1 Usability

This work discusses three HTTP tunnels for three different service types: instant messaging, cloud storage, and VoIP. First, we will discuss the usability of the tunnels and, secondly, measure their performance. To examine the usability, the Alexa top 50 websites of the United States and their basic functionality are tested using the implemented HTTP proxy. Without any optimization, the total loading time of a website was very high due to the high number of requests. To improve this, merging the contents before transmission through the tunnel is recommended. Another way to reduce the load time is to disable JavaScript. Of course, this will remove a lot of functionality, but also reduce the amount of
third-party content. Concerning the paper’s goal, we will focus on the results that have been created when loading and merging the website before transmitting it through the tunnel.

In general, we can state, that the most basic web functionality can be used through the tunnel, such as wikipedia.org or online search via google.com or bing.com. Some of the advanced features like search suggestions did not work, but besides that, the websites are usable. Furthermore, e-commerce websites like amazon.de, social media networks like facebook.com, and twitter.com or news media like nytimes.com and cnn.com are working as usual. Some other popular services we tested in addition, such as youtube.com or maps.google.com, could not be loaded successfully, most probably due to their heavy use of JavaScript.

We measured an average load time for the top 50 websites of 12.65 seconds using an automated browsing session. This load time will vary depending on various factors such as content size, connection setup, user location, etc. The average load time for the 50 websites without an active tunnel though the same mobile connection was 1.83 seconds, which is approx. \( \frac{1}{7} \) of the time required by a tunneled connection.

Besides web browsing, the HTTP tunnels can also be used to download large files. We successfully downloaded a Linux image (650 MB) and a set of MP3 files (52 MB in total) for our usability test. For each case, the download time increased by a multiple of the non-tunneled download duration. Compared to the non-tunneled test, the Google Drive tunnel took approx. 3× as long, while the Telegram and VoIP chat took approx. 2.2× as long when using the same mobile connection. The long delays result from the fact that each file has to be downloaded by the tunnel endpoint first, and only then will it be available for transmission through the tunnel. Nevertheless, while the data transfer takes more time, it is free of charge and hence represents a practical application considering the use cases from Section 3.

5.2 Performance

Since the created tunnels underly different properties, we want to figure out the technical limits of each tunnel.

First, we will analyze the Google Cloud tunnel. To evaluate its performance, we created 1000 random files of 1 KB and sent them through the tunnel. Thereby we measured upload, notification, and download time. The results of these measurements are shown in Figure 5. As we see, the notification time is the bottleneck of this approach due to the polling every 125ms. On the right-hand side, the average time to download a file through the tunnel is shown. Downloading is much quicker than uploading, which stays roughly below 1 second. In total, the average overhead to transfer a 1KB message is around two seconds. In summary, the Google Drive tunnel does not allow low latency communication but allows parallel file transfer and is well suited for large data transmissions.

We can not create such a detailed measurement for Telegram and the VoIP tunnel because we do not control the individual components. For Telegram, the used TDLib library will handle the file transfer on its own. Hence, as shown in
Figure 5, we can only measure the full transmission time, which starts when a message is sent and stops when the other end is notified. As shown in Figure 5, the transmission time is on average beneath 100 ms. Since we can disable the secret chat option, we tried, in addition, to measure what difference this makes. Interestingly, the secret chat results are slightly lower than those of the default chat. However, this difference should not be overrated because network properties represent the most dominant factor.

Likewise, we measured the full transmission time of the VoIP tunnel. As we see in Figure 5, the VoIP latency is even lower compared to the Telegram tunnel and approx. 400 times smaller compared to the Google Drive case. When comparing the cloud service, we can observe that the time overhead is measurable higher than the other two services. This confirms our expected performance difference. As it was mentioned, instant messaging and VoIP services are built to deliver content very quickly and with low latency. Unlike the other approaches, the VoIP tunnel does not induce a remarkable overhead because packets are sent and immediately forwarded by the TURN server without further processing. Moreover, the TURN server is optimized for fast packet forwarding to enable low latency for seamless VoIP connections. Hence, the VoIP tunnel has the lowest latency among all tunnel approaches.

We evaluated the download speed for the Google Drive, Telegram, and VoIP tunnels with another measurement. We created 10 random files with a size of 100 MB on a remote web server. Afterward, we requested each of the files from the mobile user device through the tunnels. During the measurement, all other network connectivity was disabled.

As we can see in Figure 6, we can approximate that a file download takes more than twice as long as it will take to download the file directly. In some cases, it will even take roughly 3x as long compared to a non-tunneled setup. This result depends on the setup since our Lab is connected to a 10 Gbit Internet connection.
Fig. 6: Measurements for the download duration through each of the tunnels for a set of 100MB files with random file contents. At t=0 the download has been initiated by the mobile client.

This allows us to up- and download files to the tunnel end device very fast. Hence, in our case, the bottleneck is the download speed of the mobile network. With another setup, the upload speed of the tunnel end device might impair the results much more than it has done in our setup. This is because Internet connections at home are often asynchronous (download >> upload speed) which impairs the file transfer. However, the task of up-and downloading files via zero-rated tunnels is an excellent use-case for the presented attacks when assuming the tunnel end device is connected to a high-speed internet connection.

6 Discussion

In the following, we will discuss and interpret our findings. We discuss the limitations and also elaborate on some advanced ideas.

6.1 All tunnels are different but useful.

Given the results from the previous section, we can see that each of the free-riding attacks enables Internet connectivity, but also comes with specific limitations and features. One of the easiest setups appears to be the Telegram tunnel; however, download file size and latency are only mediocre. The best solution to download large files via zero-rated connections seems to be the Cloud Storage approach. However, it suffers from the highest latency compared to all other solutions. The VoIP tunnel appears to be a universal solution; however, there are various obstacles to overcome, and much expertise is required to create and maintain the setup. Overall, the presented tunnels are maybe not the most novel, innovative or robust solutions, but they exist to prove a point. Secured Internet services combined with zero-rating can be used to smuggle arbitrary data, allowing cheap and unrestricted mobile communication. With just a little effort and some future developments, e.g., 5G networks, the presented solutions can be leveraged into many applications and enable people to get unrestricted internet access.
6.2 Countermeasures

During our experiments, we have not experienced any substantial impairment by the ISP or CP. Therefore, we either assume that there are no countermeasures in place, or they are not effective. Moreover, the affected provider does not answer our questions if countermeasures are in place to mitigate free-riding attacks like presented. Nevertheless, we want to describe how countermeasures, such as API limits, bandwidth limits, and traffic fingerprinting techniques, can be established to mitigate the proposed attacks.

**API limits** ensure that a user can make only a fixed number of requests in a specific time frame to the API. The limits are created to limit the number of resources used through the API, but also to mitigate risk, e.g., prevent denial-of-service attacks. In many cases, these limits will enforce a boundary on the tunnel’s efficiency and render various tunnel approaches unreasonable. Hence, we designed the tunnels to stay slightly beneath the limits. For the VoIP tunnel those API limits do not exist, but other sanitation mechanisms can be in place. In conclusion, API limits can provide a very useful measurement against the effectiveness of tunnels, but they can not mitigate them entirely.

**Bandwidth limits** can be established by the ISP and the CP to enforce a maximum throughput on the zero-rated connection. Nevertheless, just like API limits, they can limit the tunnels’ effectiveness while also creating issues for benign operations. For example, VoIP audio calls require way less bandwidth than VoIP. Hence, when strictly enforcing appropriate bandwidth limitations can easily protect some implementations. On the other hand, VoIP video services with high-quality demands are good targets since bandwidth requirements are very high.

**traffic fingerprinting** is the process of analyzing the application traffic of each user to identify fraudulent patterns or anomalies to detect malicious users. For our research, we had a closer look at two solutions, introduced in Section 2.4.

The first solution, called Tunnel Hunter, is proposed by Dusi et al. [5]. The authors claim to identify tunnels using a statistical fingerprinting mechanism at the IP layer that will analyze the inter-arrival time of network packets [5]. Since their approach works without DPI, it can be used by ISPs as well as CPs. Given the proposed attacks, we see two major disadvantages with this approach. First, Tunnel Hunter will bind many resources and cannot deal with long-living sessions in real-time. Secondly, the communication is cryptographically secured; hence packet contents will be padded to mitigate side-channel attacks. This will greatly impair any traffic analysis and hence blind the Tunnel Hunter analysis as well.

Another solution is called ZFree and has been introduced by Liu et al. [14]. In summary, ZFree will try to find tunnels by fingerprinting the transferred traffic between ISP and CP. While this integrity check allows to detect attacks that rewrite packet contents, such as the request masquerading and response modification attacks, it can not mitigate tunnels via secured Internet services. The application traffic does not change on its route through the network and is only de- and encapsulated on the user’s end devices.
In summary, the proposed countermeasures can not be effective since they typically rely on the detection of malformed traffic. They assume exploitation of lower-level protocols, while free-riding attacks via secure services will keep the integrity of the application traffic and use features on a higher level, which are in most cases confidential.

6.3 Effective protection against secured free-riding attacks

As diverse as the presented tunnels are, as diverse the mitigations have to be designed to prevent free-riding attacks. For some specific types of zero-rating services, a collaboration between ISP and CPs can be considered to detect malicious traffic. One example is the Google Drive tunnel because the CP has the right and possibility to examine the user’s content. However, such a countermeasure will be in great conflict with the user trust, privacy, and goals of secure Internet services. Furthermore, this implies that the CP wants to collaborate with the ISP, which is not always the case. However, the ISP can enforce rules or policies that force CPs to mitigate free-riding attacks as best as they can. We think one of the best solutions will be to implement certain thresholds and limits for each user to render most free-riding attacks useless. Based on this, the affected vendor has already implemented the volume for each zero-rating package to a maximum amount of 10GB. However, the zero-rated volume is still cheaper than the general data volume, and the cloud services of the ISP itself are still available without any limitation.

Maybe, mitigations alone will not solve the underlying issue. New ways will be found to create similar attacks, as we have learned from the times of Phreaking. In the future, new attacks will be executed by injecting data into video streams or into 5G high-bandwidth channels dedicated to special functions, such as car-to-car communications or emergency services. Hence, another solution to counter the growing thread of free-riding attacks can be found on an economic and political level. Mobile communication networks should not be restricted by artificial paywalls that only exist for monetary interest. Just like the flat-pricing models have become the default for cabled Internet connections, a similar model should be implemented for mobile networks. The ISPs can still profit from the volume-based tariffs, ensuring an equal distribution of costs to all users.
7 Conclusion

On our route to overcome the zero-rating practice of ISPs, we have evaluated several free-riding attacks. These attacks are intended to provide a technical argument against the growing adoption of zero-rating services. As we see, the large diversity of Internet services create various ways to bypass and exploit zero-rating offers to get unlimited Internet access for little money.

The extensive adoption of cryptography has further exaggerated this problem because countermeasures that maybe worked so far are now rendered ineffective. When the communication between the user device and the CP is secure, the ISP alone can not detect and defend against zero-rating attacks that are driven through the application layer. The only way to identify such an attack will be to cooperate with the CP if he is willing to help. However, for many applications, such as messaging services or VoIP applications, the communication will even be end-to-end encrypted, making the detection very difficult. As we see, the concurrent adoption of free-riding services and secure web services with large bandwidth requirements can heavily amplify free-riding attacks and make them almost inevitable.

With our work, we don’t want to discuss the presented attacks only; we want to shed light on the general conflict between zero-rating practices and secure Internet services. In our opinion, the best solution to this problem is not on a technical level. When net neutrality policies are enforced, the ISP cannot treat Internet traffic differently, which keeps the freedom of information exchange and fosters competition and innovation. This was essential for the growth of the Internet, and it will be necessary for its future. Likewise, this problem should not be imported into the future generation of networks, like 5G network slicing, which allows the creation of isolated networks with different priorities. A paying client can dynamically allocate the virtual slices to improve their service. In the end, prioritizing one application also means another one will be deprioritized [15]. We assume that free-riding attacks, as implemented in this work, can be used to abuse features, such as network slicing, to drive traffic through virtual high-speed networks. As we see, the issue is already adopted into the next generation of mobile networks and might create just another way for free-riding attacks.

The underlying reason why free-riding attacks exist is not only due to vulnerable protocols. It is a self-imposed issue created by Internet Service Providers that try to create artificial walls within a system that is free and open by design, such as the Internet.

Acknowledgements: The authors want to thank the Review Committee for the valuable feedback and comments. The project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 952684. Opinions, views, and conclusions are those of the authors and do not reflect the views of anyone else.
References

1. Aetha, DotEcon Ltd, European Commission, Directorate-General for Competition, Oswell Vahida: Zero-Rating Practices in Broadband Markets: Final Report. (2017), http://dx.publications.europa.eu/10.2763/002126

2. Allan Riordan Boll: Allan’s Blog: Raw sockets with BPF in Python (2011), http://allanrbo.blogspot.com/2011/12/raw-sockets-with-bpf-in-python.html

3. Carrillo, A.J.: Having Your Cake and Eating it Too? Zero-Rating, Net Neutrality and International Law (2016), https://papers.ssrn.com/abstract=2746447

4. Cloudflare Inc.: QUICHE an implementation of the quic transport protocol and http/3 as specified by the IETF, https://github.com/cloudflare/quiche

5. Dusi, M., Crotti, M., Gringoli, F., Salgarelli, L.: Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting 53(1), 81–97 (2009), https://doi.org/10.1016/j.comnet.2008.09.010, http://www.sciencedirect.com/science/article/pii/S1389128608003071

6. Go, Y., Jeong, E., Won, J., Kim, Y., Kune, D.F., Park, K.: Gaining Control of Cellular Traffic Accounting by Spurious TCP Retransmission. In: NDSS Symposium 2014 (2014). https://doi.org/10.14722/ndss.2014.23118

7. Google: google-api-python-client. The official Python client library for Google’s discovery based APIs (2020), https://github.com/googleapis/google-api-python-client

8. Internet.org: Free Basics by Facebook - Apps on Google Play (2020), https://play.google.com/store/apps/details?id=com.freebasics&hl=en

9. Internet.org: Internet.org - Our Mission (2020), https://info.internet.org/

10. Kakihana, Y.: T-Mobile: Why We Love 5G. In: Proceedings of the 2nd Global 5G Forum. 2nd Global 5G Forum (2019)

11. Kraemer, J., Wiewiorra, L., Weinhardt, C.: Net neutrality: A progress report 37(9), 794–813. https://doi.org/10.1016/j.telpol.2012.08.005, http://www.sciencedirect.com/science/article/pii/S0308596112001450

12. Lapsley, P.: Exploding the Phone. Grove Press (2014)

13. Li, C.Y., Tu, G.H., Peng, C., Yuan, Z., Li, Y., Lu, S., Wang, X.: Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 316–327. CCS ’15, Association for Computing Machinery. https://doi.org/10.1145/2810103.2813618, https://doi.org/10.1145/2810103.2813618

14. Liu, Z., Zhang, Z., Cao, Y., Xi, Z., Jing, S., Roche, H.L.: Towards a Secure Zero-rating Framework with Three Parties. pp. 711–728 (2018), https://www.usenix.org/node/217562

15. Lohninger, T.: 5G and Net Neutrality (2019), https://media.ccc.de/v/36c3-10711-5G_net_neutrality

16. montag451: Montag451/pytest (2020), https://github.com/montag451/pytest

17. Online, H.: Eugh: Zero rating verletzt netneutralitaet (2020), https://www.heise.de/news/EuGH-Zero-Rating-verletzt-Netzneutralitaet-4894216.html

18. pallets: The python micro framework for building web applications. (2020), https://github.com/pallets/flask
19. Peng, C., Li, C.y., Tu, G.H., Lu, S., Zhang, L.: Mobile data charging: New attacks and countermeasures. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 195–204. CCS ’12, Association for Computing Machinery. https://doi.org/10.1145/2382196.2382220, https://doi.org/10.1145/2382196.2382220
20. Philippe Biondi: Secdev/scapy (2020), https://github.com/secdev/scapy
21. ProSec GmbH: Die schwachstelle in fast allen hotspots! — prosec (2017), https://www.prosec-networks.com/blog/nach-wannacry-prosec-wannasurf/
22. Service, C.R.: The Net Neutrality Debate: Access to Broadband Networks. CreateSpace Independent Publishing Platform, crs report r460616 - 2018 edition edn.
23. Sid Murlidhar: Fast and free facebook access with 0.facebook.com (2010), https://web.archive.org/web/20140116060937/http://www.facebook.com/notes/facebook/fast-and-free-facebook-mobile-access-with-0facebookcom/391295167130
24. Solon, O.: ‘It’s digital colonialism’: How Facebook’s free internet service has failed its users (2017), https://www.theguardian.com/technology/2017/jul/27/facebook-free-basics-developing-markets
25. T-Mobile: Free Unlimited Streaming — Music Freedom Simple Choice Plan — T-Mobile (2020), https://www.t-mobile.com/offers/free-music-streaming
26. Thomas Lohninger, Benedikt Gollatz, Cornelia Hoffmann, Erwin Ernst Steinhammer, Ludger Benedikt Defaa, Ali Al-Awadi, Andreas Czák: Report: The Net Neutrality Situation in the EU (2019), https://epicenter.works/sites/default/files/2019_netneutrality_in_eu-epicenter.works-rl.pdf
27. Wikimedia Foundation: Wikipedia Zero - Wikimedia Foundation Governance Wiki (2018), https://foundation.wikimedia.org/wiki/Wikipedia_Zero
28. Wu, T.: Network Neutrality, Broadband Discrimination (2003), https://scholarship.law.columbia.edu/faculty_scholarship/1281