Vasile C. Perta*, Marco V. Barbera, Gareth Tyson, Hamed Haddadi[1], and Alessandro Mei[2]

A Glance through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN clients

Abstract: Commercial Virtual Private Network (VPN) services have become a popular and convenient technology for users seeking privacy and anonymity. They have been applied to a wide range of use cases, with commercial providers often making bold claims regarding their ability to fulfil each of these needs, *e.g.,* censorship circumvention, anonymity and protection from monitoring and tracking. However, as of yet, the claims made by these providers have not received a sufficiently detailed scrutiny. This paper thus investigates the claims of privacy and anonymity in commercial VPN services. We analyse 14 of the most popular ones, inspecting their internals and their infrastructures. Despite being a known issue, our experimental study reveals that the majority of VPN services suffer from IPv6 traffic leakage. The work is extended by developing more sophisticated DNS hijacking attacks that allow all traffic to be transparently captured. We conclude discussing a range of best practices and countermeasures that can address these vulnerabilities.

Keywords: VPN, IPV6, DNS hijacking

DOI 10.1515/popets-2015-0006
Received 11/22/2014; revised 2/16/2015; accepted 2/17/2015.

1 Introduction

Recent revelations regarding massive surveillance projects [1] and the restrictions that some governments impose on their citizens [2–4] have increased the general public's concern regarding untrusted or malicious parties observing and/or manipulating user communications. This has contributed to a rise in the popularity of tools promising end-users a private and/or anonymous online experience [5–9]. Among them, VPN-based solutions are receiving an increasing amount of attention [8, 10, 11]. In fact, the market today is littered with a number of low-cost commercial VPN services, claiming to be able to enhance user security and privacy, or even to provide anonymity, by tunneling their Internet traffic in an encrypted form to an (ideally) trusted remote endpoint.

There are several use cases that may have contributed to this spike in popularity. For example, the use of public networks has increased dramatically in-line with the expansion of the mobile device market. Such infrastructures are ripe for attack (*e.g.,* stealing credentials, snooping, session hijacking [12–14]), leading some users to securely direct their traffic through a VPN tunnel as a solution for safeguarding their interactions [15]. Other users may be attracted by VPN tunnel encryption as a way to avoid unwanted attention, or simply to hide their actions from their ISP or other passive observers. Others turn to VPN services for more pragmatic reasons, wishing to circumvent Internet censorship by tunneling through firewalls [16], or accessing content that is either blocked by their ISP or restricted based on a country's IP addresses (*e.g.,* BBC iPlayer, Hulu, Netflix). In response to the latter, many VPN services allow users to select their exit points so that they can gain IP addresses in a number of different countries or administrative domains. Finally VPN services are widely used by citizens facing government-supported large-scale Internet censorships events, as revealed by recent studies [3, 4].

All commercial VPN service providers support the above use cases to some extent, although their capability to preserve user privacy and anonymity has already raised some questions [17]. In fact, a common misconception is that the word "private" in the VPN initialism is related to the end-user's privacy, rather than to the interconnection of private networks. In reality, privacy and anonymity are features that are hard to obtain, requiring a careful mix of technologies and best practices that directly address a well-defined adversarial/threat model [5, 17]. In other words, there is no silver bullet within this domain. For instance, it is clear that simply tunneling traffic through a VPN cannot provide the same anonymity guar-

***Corresponding Author: Vasile C. Perta:** Sapienza University of Rome, E-mail: perta@di.uniroma1.it
Marco V. Barbera: Sapienza University of Rome, E-mail: barbera@di.uniroma1.it
Gareth Tyson: Queen Mary University of London, E-mail: gareth.tyson@qmul.ac.uk
Hamed Haddadi[1]: Queen Mary University of London, E-mail: hamed.haddadi@qmul.ac.uk. This work was done while the author was at Qatar Computing Research Institute.
Alessandro Mei[2]: Sapienza University of Rome, E-mail: mei@di.uniroma1.it. This work has been partially supported by a Google Faculty Research Grant 2013.

antees of more rigorous (and vetted) systems such as Tor [5]. This does not come as a surprise, as VPNs were not originally intended to provide anonymity and/or privacy.

Still, the appeal that these services have for the general public is very high, perhaps because of their ease of use, their relatively high performance, their effective marketing strategies, and the bold statements the providers make, though in absence of objective evidence in their support. The resulting blind faith that uninformed users may put into these services is thus a worrisome problem that has to be tackled effectively and rapidly.

Within this context, we contribute by shedding light on the privacy and anonymity features of the popular commercial VPN services available today on the market. We use an experimental approach, subscribing to 14 services, downloading their recommended clients on both desktop and mobile systems, and testing them in our lab. Our findings confirm the criticality of the current situation: many of these providers leak all, or a critical part of the user traffic in mildly adversarial environments. The reasons for these failings are diverse, not least the poorly defined, poorly explored nature of VPN usage, requirements and threat models.

This paper is organised as follows. We first survey the tunneling technologies most commonly used by VPN service providers (§ 2), finding that many still rely on outdated technologies such as PPTP (with MS-CHAPv2), that can be easily broken through brute-force attacks [18]. We then show that the vast majority of commercial VPNs clients suffer from data leakage in dual stack networks (*i.e.,* those supporting both IPv4 and IPv6), sending large amounts of traffic over the native interface, unbeknown to the user (§ 3). By exploring various applications, websites and operating systems, we show that significant amounts of traffic are therefore exposed to public detection, while users retain the belief that all their interactions are securely occurring over the tunnel (§ 4). Most importantly, we find that the small amount of IPv6 traffic leaking outside of the VPN tunnel has the potential to actually expose the whole user browsing history even on IPv4 only websites. We further extend this analysis by delineating a DNS hijacking attack that exploits another key vulnerability in many VPN configurations (§ 5). Through this attack, a substantial amount of IPv4 traffic can be leaked from the VPN tunnel too.

It is important to note that, worryingly, the insecurity of PPTP (with MS-CHAPv2), as well as IPv6 and DNS leakage in VPNs are not new to the community [17–20]. Despite this, our study reveals that many commercial VPN services still fail to properly secure user traffic. These low-cost solutions therefore raise many questions in terms of trust and reliability. To the best of our knowledge, we are the first to offer quantified information on the severity of this issue, as well as straightforward countermeasures (§ 6).

2 Commercial VPN services

We begin by surveying a number of commercial VPN services to understand their infrastructures and technologies.

2.1 Overview of Commercial VPN service providers

A large range of commercial VPN services exists today. We therefore begin our study by performing an analysis of the market, registering credentials with 14 services. This set has been selected due to their widespread popularity and advertised features. All the experiments were carried out during the period September – December, 2014. Given the impossibility of objectively measuring it, popularity was approximated with the number of times each VPN service was mentioned in the first 20 Google results corresponding to queries such as “Best VPN” or “Anonymous VPN”. The idea was to identify the subset of providers that the average user would be most likely to purchase, based on public reviews, forum mentions, and so on. Our selection was further augmented with VPN services that, although not among the most popular, advertised distinctive features that were relevant to our study. These include Mullvad, which to the best of our knowledge is the only provider mentioning IPv6 leakage protection; Hotspot Shield, promising WiFi security in untrusted hotspots; and TorGuard, which explicitly targets BitTorrent users. Table 1 lists the providers selected.

2.2 VPN service infrastructure

We next briefly explore the infrastructures used by commercial VPN services, as observed from our experiments. As Table 1 shows, the number of available servers (exit points) can vary significantly across providers, ranging from several hundreds of the top 4 down to less than 10 (a small number of servers could indicate the capability of dynamically adding more resources, based on the service utilisation). Figure 1 presents the distribution of exit points across countries, highlighting a significant bias towards the United States (US). This is probably related to the amount of content that is only accessible from the US, *e.g.,* Hulu, Showtime Anytime, HBO GO. Countries with strict privacy laws (*e.g.,* Netherlands) also seem attractive as VPN tunnel exit points, perhaps driven by users concerned about anonymity.

The distribution of servers across Autonomous Systems (ASes) can also be inspected. A large number of ASes and hosting services are involved in VPN provision. In total, there

Provider	Countries	Servers	Technology	DNS	IPv6-leak	DNS hijacking
Hide My Ass	62	641	OpenVPN, PPTP	OpenDNS	Y	Y
IPVanish	51	135	OpenVPN	Private	Y	Y
Astrill	49	163	OpenVPN, L2TP, PPTP	Private	Y	N
ExpressVPN	45	71	OpenVPN, L2TP, PPTP	Google DNS, Choopa Geo DNS	Y	Y
StrongVPN	19	354	OpenVPN, PPTP	Private	Y	Y
PureVPN	18	131	OpenVPN, L2TP, PPTP	OpenDNS, Google DNS, Others	Y	Y
TorGuard	17	19	OpenVPN	Google DNS	N	Y
AirVPN	15	58	OpenVPN	Private	Y	Y
PrivateInternetAccess	10	18	OpenVPN, L2TP, PPTP	Choopa Geo DNS	N	Y
VyprVPN	8	42	OpenVPN, L2TP, PPTP	Private (VyprDNS)	N	Y
Tunnelbear	8	8	OpenVPN	Google DNS	Y	Y
proXPN	4	20	OpenVPN, PPTP	Google DNS	Y	Y
Mullvad	4	16	OpenVPN	Private	N	Y
Hotspot Shield Elite	3	10	OpenVPN	Google DNS	Y	Y

Table 1. VPN services subject of our study

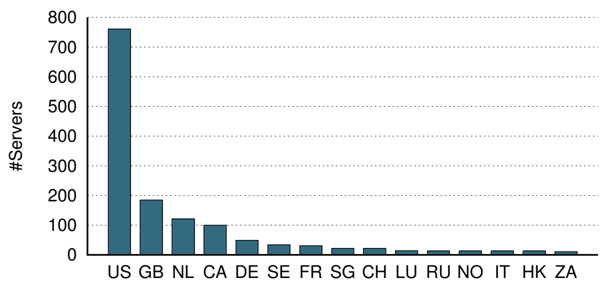


Fig. 1. Countries with most VPN exit locations

are 244 ASes used by just 14 providers. We find that providers tend to place their exit points in a number of different ASes, even on a per-country basis. This might be a resilience decision, since it would be non-trivial for snoopers to monitor all exit points. That said, we notice that there is significant co-location in some countries, with several different providers placing their exit points in a very small number of ASes. For example, in Switzerland, 60% of all exit points can be found in a single AS; investigation of this hosting service revealed very strong privacy guarantees which may attract VPN service providers concerned about monitoring issues. Overall, however, the dominant countries (*e.g.,* the US and UK) tend to avoid this level of co-location, perhaps favouring more resilient placement strategies.

2.3 Anonymity features

We found the most commonly advertised features to be "Access to restricted content" and "Anonymity". Regarding the former, all providers offer similar capabilities by tunneling user traffic towards different countries, although performance, price, and destination numbers may vary. The anonymity claims, however, seem to be exceedingly vague, which is in contrast with the inherently limited anonymity these services are actually capable of providing. This does not come as a surprise, as VPNs, as opposed to Tor, were not originally intended to provide anonymity and/or privacy.

More specifically, if the only objective is to conceal the user IP address from a website, then a VPN may be a viable choice. For multiple reasons, however, it is questionable that these services provide anonymity beyond this minimal objective. First, it is clear that users are not anonymous from their VPN service provider, which must be blindly trusted to not be malicious, and to not disclose the user traffic to third parties (*e.g.,* through subpoena). The data these providers retain about their customers exacerbates this problem further. For instance, we observed that, at registration time, many VPN services (even some of those supporting the more anonymous Bitcoin payments) ask for the user's personal information, or even for a valid mobile number. Similarly, a number of them admit they retain timestamps, the amount of data transmitted, and the client IP address of each VPN connection. In contrast, in Tor no single relay is able to observe traffic at both ends of a circuit. Also, Tor does not require registration or the providing of any identifying information to download or use its software or network, and Tor nodes promptly remove session keys and any other information about a connection once the connection is closed. Most importantly, Tor users are not expected to trust the single relays, but, rather, the fact that the majority of resources of the Tor network are not controlled by a single malicious entity.

Another way user anonymity may be compromised is through end-to-end attacks (*e.g.,* traffic correlation), which do not require collaborating with or compromising the VPN service provider [21–24]. While not sufficient, Tor's use of shared, geographically distributed exit points, in combination with an exit point rotation policy, has the potential to make end-to-end attacks harder [24–29]. On the other hand, the commercial VPN services within this study are much easier prey. For instance, in terms of the geographical diversity of their infrastructure, the sum of all VPN services providers still does not match Tor, whose nodes are, as of today, spread across 468 different autonomous systems (against 244). Also, we note that the client interfaces of the VPN services we tested tend to encourage users to give priority to connection quality over location diversity, offering automated server selection based on geographic proximity and/or network speed, remembering the latest server used, or even presenting the random server selection as an available but "not recommended" option. It is therefore likely that a user will always rely on the same small subset of exit servers, making end-to-end attacks far easier.

While a thorough analysis of these issues falls outside the focus of this paper, we wish to highlight the limited practical protection that these services are capable of offering against user information disclosure or simple traffic correlation attacks.

2.4 Tunnel setup

The clients of the VPN services we screened support one or more of the following tunneling protocols: OpenVPN, L2TP (with IPsec) and PPTP. Users are typically given free choice of the VPN technology to use, except in some cases, where the technology of choice is driven by the plan purchased. For instance, the cheapest VyprVPN plan only allows PPTP to be used. We found the relatively large number of providers that use PPTP to be worrying, as PPTP's authentication protocol, MS-CHAPv2, is affected by serious security vulnerabilities that have been well-known in the community for years [18, 30].

Once the VPN client has initiated the tunnel using one of the above technologies, it creates a virtual network interface (*e.g.,* `ppp0` or `tun0`) and manipulates the host routing table in order to redirect all the traffic towards it. Traffic that passes through the virtual interface gets encrypted and forwarded to the VPN remote entry point via the host's active network interface (*e.g.,* WiFi or Ethernet). Once these steps have completed, the tunnel is fully initiated and all user traffic should be sent via the VPN in an encrypted form. The above mechanism holds true across all providers surveyed. The tunneling protocols described have already undergone thorough security analysis. The remainder of this paper focuses, instead, on the second stage of the VPN client's operation: traffic redirection. Although its use of routing table modification is simple, we note it exposes VPN users to a number of subtle, but critical, privacy vulnerabilities. The problem stems from the fact that routing tables are a resource that is concurrently managed by the operating system, which is unaware of the security requirements of the VPN client. Specifically, small changes to the routing table (both malicious and accidental) could result in traffic circumventing the VPN tunnel, creating serious data leakage over other interfaces. The rest of this paper shows how this fact is sufficient to reconsider many of the claims that VPN service providers make about their security and anonymity provisions.

2.5 Commercial VPN adversary models

In our study, we consider two general types of adversaries for commercial VPN users. Importantly, both represent the type of adversary a user would likely wish to avoid via their use of a VPN service. These are:

1. *Passive Observer:* The adversary operates monitoring points within the native network provider used by the victim (or another pertinent location). They wish to gain access to the traffic, but do not take proactive steps to circumvent the VPN.
2. *Active Attacker:* The adversary controls the point of attachment that the victim connects to. This could be the Internet Service Provider (ISP) or, alternatively, a third party offering connectivity (*e.g.,* in a cafe or hotel). Such an adversary could also masquerade as a trusted WiFi network by faking its SSID [12]. They wish to gain access to the traffic and would take proactive steps to circumvent the VPN.

In all cases, the adversary's objective is to monitor the traffic exiting the host, or even manipulate it (*e.g.,* rate limiting, censorship). Several instantiations of these adversaries are feasible. For example, governmental agencies wishing to monitor civilian activities would be strongly motivated to take on one of these roles. Also, the very same Internet-based services accessed through VPNs (*e.g.,* search engines, social network sites, forums, IRC servers, and so on) may wish to collect identifying attributes of their users, or build a profile of their activities. These adversaries do not represent a comprehensive set of attackers but, rather, a representative sample of important stakeholders. We use them throughout the rest of the paper to highlight the uses of the vulnerabilities we have discovered.

3 IPv6 VPN Traffic Leakage

All VPN services surveyed rely on the correct configuration of the operating system's routing table (§ 2.4). Worryingly, no attempt is made to secure this operation, for instance through monitoring the routing table to ensure that their initial configuration is not changed. Small changes could therefore undermine the security offered by the VPN tunnel. We delay the discussion of more sophisticated routing table attacks to later (§ 5). Here, we delineate a more alarming vulnerability, requiring no accidental or malicious changes to the configuration.

The vulnerability is driven by the fact that, whereas all VPN clients manipulate the IPv4 routing table, they tend to ignore the IPv6 routing table. No rules are added to redirect IPv6 traffic into the tunnel. This can result in all IPv6 traffic bypassing the VPN's virtual interface. Although not a serious issue some years ago, increasing amounts of traffic is now IPv6, bringing the problem to criticality [31]. This attack could be performed by both adversaries detailed in § 2.5.

3.1 Why does IPv6 leakage occur?

The vulnerability relies on the nature of IPv4/6 dual stack implementations on common operating systems. Dual stacks have been introduced to smoothly transition between the two protocols (RFC 4213), allowing a network and host to simultaneously operate both IPv4 and IPv6. The problem emerges because common dual stack implementations show preference to IPv6 when available (in line with RFC 6724).

On dual stack hosts, applications can connect to any remote socket using both IPv4 and IPv6 addresses, depending on which protocol version the remote host supports. To decide upon which to use, the host's DNS resolver should attempt to retrieve both address types from the DNS server (*i.e.,* both A and AAAA records). This is what modern address resolution routines, such as POSIX's `getaddrinfo` [32] do, allowing both protocols to coexist (in compliance with RFC 3493). When both IPv4 and IPv6 addresses are returned, the operating system nearly always shows preference to IPv6. There are also other techniques that select the best of the two connections [33]. However, IPv6 will usually be preferred as the IPv4 VPN tunnel introduces overheads that increase the resolution delay. In such cases, the operating system refers to the IPv6 routing table to select the first-hop router, bypassing the changes made to the IPv4 routing made by the VPN client. All IPv6 traffic will therefore exit the host via the native (IPv6) network interface, rather than through the VPN tunnel. This simple observation is the crux of IPv6 leakage.

The above vulnerability occurs whenever a host is connected to an IPv6-enabled network. It is important to acknowledge two scenarios in this regard. First, a point of attachment could *accidentally* support the leakage by providing IPv6 connectivity to a client. This is, of course, dangerous, but the point of attachment is not actively trying to subvert the VPN tunnel (*e.g.,* this could be a Passive Observer, such as the ISP). Second, a point of attachment aware of the vulnerability could *intentionally* enable the leakage by offering IPv6 connectivity and recording all traffic (*e.g.,* an Active Attacker adversary, such as a malicious WiFi AP). The latter is a serious threat, as the attack can be carried out with modest resources. In fact, an AP with no IPv6 connectivity can be easily configured to create a dual-stack WiFi network, as we are going to short shortly (§ 3.2). In reality the adversary does not even need to control the AP. In fact, *any* malicious client connected to a public WiFi can easily create a dual-stack network and inject a rogue Router Advertisement [34] to attract and record all IPv6 traffic.

3.2 Which VPN services are vulnerable?

To explore how different providers react to the above vulnerability, we create a simple testbed. A campus dual stack WiFi LAN is used to connect a variety of hosts running the following operating systems: Linux (Ubuntu 14.04), Windows (8.1 Pro), OSX (Mavericks), iOS 7, and Android (JellyBean, KitKat). The WiFi access point used is an OpenWrt router running IPv6 through an IPv4 tunnel provided by Hurricane Electric's Tunnel Broker service [35]. A /64 IPv6 prefix provided by the tunnelbroker is then advertised on the LAN, enabling the clients to configure an IPv6 address through SLAAC (Stateless Address Autoconfiguration). This configuration, which may very well be used by a malicious WiFi router, allows us to transparently monitor all traffic in the network in the same way an adversary would.

After setting up the network, we test every combination of operating system and VPN client. Each test is performed by executing a small measurement tool simulating a generic IPv6-enabled web application. The tool connects to port 80 of the first address returned by the operating system's resolver for the `wwww.google.com` domain, which is available both via IPv4 and IPv6. This is sufficient to explore how IPv6 traffic is treated by the operating system and VPN. Under perfect circumstances, the connection will be performed through the VPN tunnel, and the WiFi access point will *only* see encrypted VPN traffic.

Our tests reveal that all desktop VPN clients tested, except for Private Internet Access, Mullvad and VyprVPN, leak the entirety of IPv6 traffic (Table 1). We confirm that the reason for this is the combination of the operating system's resolver preference for IPv6 (when available) and the host's IPv6 routing table being left unchanged by the VPN client. One client, TorGuard, acknowledges this problem by offering the possibility of disabling IPv6 traffic (through the advanced settings). However, the option is not enabled by default.

Interestingly, we also notice that even VPN clients that explicitly change the system's DNS server to one they control

(Table 1), still allow for IPv6 addresses to be returned to the end-host, thus supporting the leakage. In other words, the steps so far taken by nearly all providers to deal with this problem, if any, are ineffectual. As already mentioned, the only services that are not affected by IPv6 traffic leakage on desktop OSes are Private Internet Access, Mullvad, VyprVPN, and, if explicitly enabled by the user, TorGuard. During the tunnel setup, their clients completely disable IPv6 on the hosts' network interface(s). This is a rather drastic approach, but it turns out to be very effective, given the possibility of falling back on IPv4 in dual stack networks.

A separate discussion must be presented for mobile OSes. We experimentally observed that, on iOS, all tested VPN services are immune to IPv6-leakage, as IPv6 is completely disabled during the VPN tunnel lifetime. On the other hand, we found that the leakage affects *all* VPN services on Android, including Private Internet Access, Mullvad, TorGuard and VyprVPN.

4 Measuring the Criticality of IPv6 Leakage

We quantify the criticality of IPv6 leakage by investigating the extent to which actual applications (*e.g.,* Web browsers) are exposed to the attack. We combine a number of datasets to investigate this issue from multiple angles. Note that, although this vulnerability only emerges when a host has IPv6 connectivity, recent work highlights a significant upward trend, with IPv6 prefixes constituting 60% of new allocations [31]. Enterprise networks are particularly driving this, yet home networks also have increasing uptake [36].

4.1 How exposed are websites to leakage?

Web traffic forms the bulk of Internet transactions. Arguably, it is also the most sensitive. The most popular web browsers (*e.g.,* Google Chrome, Firefox, Internet Explorer, Safari, Opera) have been natively supporting IPv6 for a while, exposing the browsing of users of vulnerable VPN services. Thus, we begin our analysis by inspecting the Alexa rankings for popular websites that are IPv6-enabled. To do so, we perform AAAA DNS queries against the top 500 websites for every country. Figure 2 presents the list of top IPv6 websites, alongside how many countries count them among their top 500. Overall, we find that 19% of websites support IPv6.

A variety of sites are present. Deciding which ones are sensitive (*i.e.,* the ones users would wish to remain anonymous

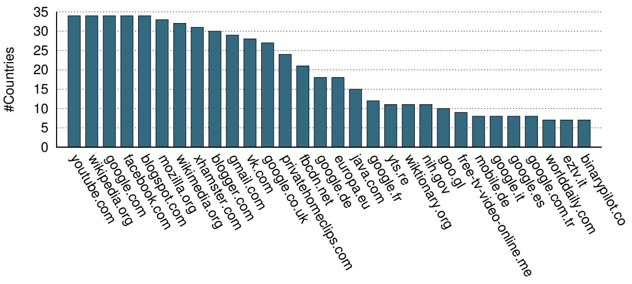


Fig. 2. Topmost popular IPv6-enabled websites, alongside how many countries count them among their top 500

from) is a subjective process. However, we observe various intuitively sensitive IPv6 websites. These include search engines, social networks, blog platforms, adult content providers and illegal video services. It is likely that some users may want to leverage VPNs to stop these sites from knowing their identity, or from tracing their activities. This wouldn't be possible in the case of IPv6 leakage, as all interactions with these sites would circumvent the VPN tunnel and silently occur over the open native interface. One solution to intermediate snooping might be to use a secure protocol (*e.g.,* HTTPS); in fact, 84% of the domains studied support HTTPS. However, this would not protect the user's identity from the website owner. Further, even HTTPS traffic may be susceptible to de-anonymisation by analytics tools [37, 38], or man in the middle attacks enabled by the data leakage (*e.g.,* `sslstrip` [39]). At the very least, this would allow a passive observer to discover which HTTPS websites the user is viewing (with 90% accuracy [37]).

4.1.1 IPv4 browsing history leakage

So far, the discussion has exclusively focused on websites that operate over IPv6. Although these are often big players (*e.g.,* Google, Facebook, Wikipedia), one could argue that they only represent a relatively small portion of the web, which still solely relies on IPv4. Unfortunately, this leaked IPv6 traffic can be extremely harmful for the user privacy, as we discuss now.

During our experiments we observed that the majority of websites also embed a number of third party "plug-ins" (*e.g.,* ad brokers, trackers, analytics tools, social media plug-ins). The large diffusion of these objects has already raised concerns about a decreasing number of external, large entities being able to get a detailed view of the web browsing activity of all the Internet users [40–42]. A substantial contribution to this leakage is the `Referer` HTTP header, disclosing the exact URL of the visited page in the fetches of each of the third party objects embedded in it. If just a single one of these

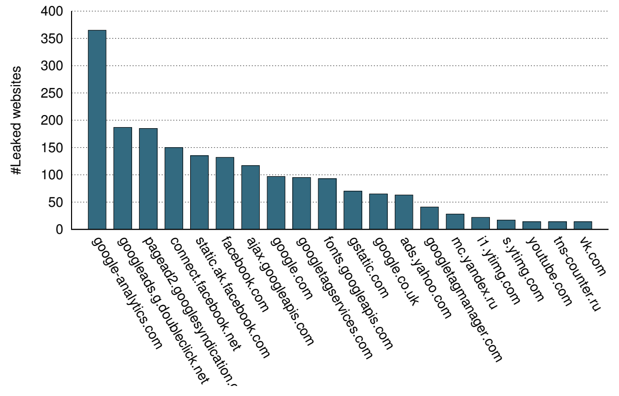


Fig. 3. Top third-parties that leak IPv4-only websites through the `Referer` header. 92% of the Alexa top 1K IPv4-only websites embed objects of at least 1 of these third parties.

fetches were to happen outside of the VPN tunnel (through IPv6 leakage), the actual user IP would be revealed to the relevant third-party, and, perhaps most importantly, the `Referer` header would reveal the page the victim is visiting to any other Passive Adversary.

To quantify this phenomenon, we study the number of IPv6 third party objects that the Alexa [43] top 1K IPv4-only websites embed. The crawling was automated using the Selenium WebDriver [44]. The results reveal that IPv6 third party objects are extremely common. In fact, 92% of the pages we studied contained at least one of them, exacerbating the leakage hugely. Figure 3 presents the top third parties these objects belong to. As expected, these include various mainstream organisations, such as Google, Facebook, and Yahoo, which are among the early IPv6 adopters.

4.2 How exposed is mobile traffic to leakage?

Considering the amount sensitive data stored in smartphones, it is critical to understand the level of exposure that mobile users have to IPv6 leakage. The issue is even more critical considering that, according to our experiments, all of the VPN services we tested on Android leak IPv6 (§ 3.2). The same observations about web browsing leakage we did in § 4.1 thus apply to Android web browsing too.

In order to investigate the existence of other, less obvious sources of leakage, we test the top 100 most popular applications available on the Android Market, fetched using an unofficial Google Play API [45]. In our experiments, we use the dual stack OpenWrt testbed described in § 3.2: We connect an Android device running a VPN client and monitor, in turn, the network traffic generated by each app for 5 minutes. Note that we only use one exemplary VPN service because our experiments show that they all leak in the same manner. Our measurements revealed that, similar to what we observed for websites, almost all apps we tested (80%) indirectly leak sensitive information through third party plug-ins: namely, the embedded advertisements. It is interesting to note that, among the apps surveyed, we found that Google's DoubleClick is the only advertisement engine that supports IPv6. This does not mitigate the leakage, though, as, recently, Viennot *et al.* [46] found that 75% of all Android apps include Google ad libraries, which suggests that all these apps are exposed to the leakage. Also, unlike other Google traffic, advertisements are always served in the clear via HTTP, allowing passive monitoring to easily occur.

Unlike web-based advertisements, we also observe a remarkable amount of information contained with mobile ad fetches. For instance, depending on the application requesting the advertisement, various sensitive data is exposed, including application name, language, location, mobile carrier used, *etc.*. In addition, the ads are sent very frequently (*e.g.,* every 10 seconds) which may even make it possible to monitor app usage over time, and accurately profile the user. Moreover, Castelluccia *et al.* [47] show that by just observing Google advertisements, an attacker can reconstruct a target user profile with high accuracy.

4.3 How exposed is peer-to-peer to leakage?

Anecdotally, VPN services are popular among peer-to-peer users wishing to anonymise their downloads. The open nature of these systems allow us to explore this hypothesis by measuring how many BitTorrent IP addresses belong to VPN service provider exit points. Note that we do not collect any sensitive data that could be attributed to individual users (particularly as user IP addresses often change over time). Our analysis only focuses on aggregated statistics rather than individual user activity. To further ensure this, the exact content shared by users was not recorded. Rather, we used the tracker's labels to categorize each torrent to a particular content (*e.g.,* "Movies"). In addition, all collected data was deleted after the analysis.

We have performed several crawls of the most popular BitTorrent site, Piratebay, collecting information on 33.5K torrents. Whenever a new torrent was published, we repeatedly downloaded its peer list every 5 minutes from the tracker. This collected 2.7M unique IP addresses. We then checked if these addresses were registered with any of the VPN service providers presented in § 2. Note, however, that this is a lower estimate because we do not possess a comprehensive list of all VPN exit points in the world.

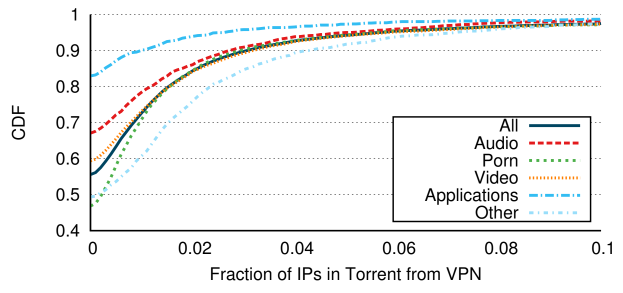


Fig. 4. Fraction of VPN users in swarms

Country	# IPv6	# Peers	% IPv6	# IPv6	# Peers	% IPv6
	29-06-2014			04-03-2014		
US	4281	31429	13.62%	10148	33212	30.56%
Germany	29	303	9.57%	90	239	37.66%
Malaysia	111	2137	5.19%	97	989	9.81%
France	95	2516	3.78%	631	2011	31.38%
Japan	12	400	3%	31	197	15.74%
China	13	557	2.33%	59	305	19.34%

Table 2. Top Countries for Native IPv6 Deployment in BitTorrent

Figure 4 presents the fraction of IP addresses in each swarm that originate from a known VPN exit point (as a CDF). The number of VPN users varies heavily across the different swarms: Diversity can be observed between different content categories, as well as within individual categories. In total, we find that 1.2% of trace entries come from known VPN exit points. We also note that some categories have a higher tendency towards VPN services than others. For example, approximately half of all pornographic torrents contain at least one VPN user. Another interesting point is that VPN users participate in BitTorrent swarms for longer than non-VPN users. As a proportion, they are witnessed in our logs more than 3 times as often as non-VPN users.

All of these VPN service users are therefore potentially leaking IPv6 traffic, via both BitTorrent and HTTP. Unfortunately, our traces cannot reveal which users also use IPv6, as we only observe their VPN tunnel exit address at the tracker. Consequently, we turn to [48], which publicly reports the number of IPv6 BitTorrent peers per country, using the methodology detailed in [49]. Table 2 presents the results for the last two measurement periods. It shows that some countries have a significant IPv6 presence, most notably the US. Such regions would therefore be extremely vulnerable to IPv6 leakage. Many popular clients, including Vuze and μtorrent, already support IPv6, which means this value will increase as soon as native networks introduce support.

5 Strong Adversary: Hijacking the DNS

A key assumption of the above vulnerability is that the host is connected to an IPv6-enabled network. This network could be an unaware ISP or, alternatively, a malicious access point consciously trying to acquire leaked IPv6 traffic. In either case, only IPv6 traffic is vulnerable. We next discuss a more concerted attack, which attempts to create *DNS leakage*, subverting the resolution of a host's DNS queries. Through this, the adversary can resolve all DNS queries to its own local proxies, bypassing the VPN tunnel and gaining control over *both* IPv4 and IPv6 traffic.

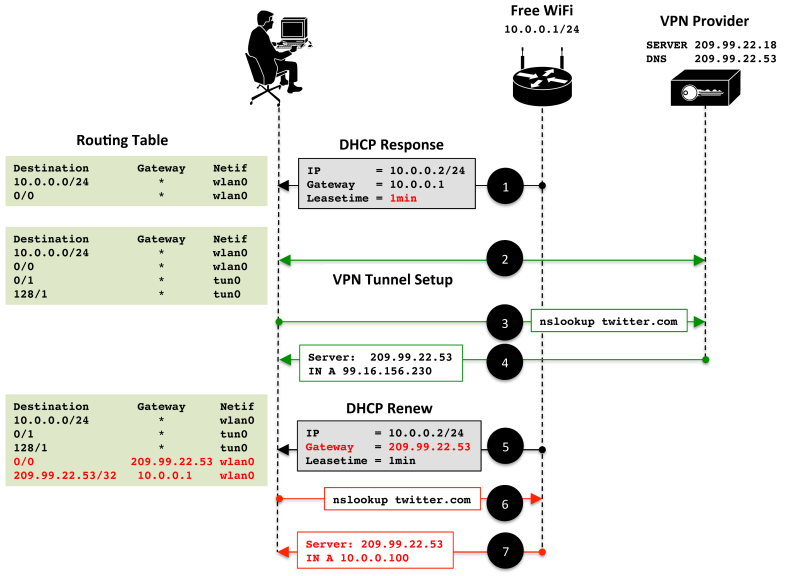
To date, in the context of VPNs, DNS leakage has mostly been related to Windows systems. In particular, Windows does not have the concept of global DNS settings, rather, each network interface can specify its own DNS. Due to the way Windows processes a DNS resolution [50], any delay in a response from the VPN tunnel may trigger another DNS query from a different interface, thus resulting in a leak. In this section, we extend this vulnerability to other platforms (Linux, Android, OSX, iOS) and define a novel attack that allows an adversary to hijack the DNS traffic of many VPN services, even for those whose client explicitly sets a custom DNS server.

5.1 DNS configuration primer

The DNS hijacking attack works by manipulating a host into redirecting its DNS queries to an adversary-controlled server. Despite the criticality of the DNS resolution process, we found that most VPN services do not take significant steps to secure it. We broadly classify the observed DNS configurations into three types (see Table 1):

1. *Default:* Under certain configurations, some VPN clients do not change the DNS settings, leaving the host's existing DNS server as the default. For instance, we observed that the desktop version of HideMyAss (v1.17) does not set a custom DNS when using OpenVPN.
2. *VPN Managed - Third-party DNS:* Most VPN clients override the DNS server settings during setup. Among the 14 VPN service providers we analysed, we found that 8 use third party DNS servers, namely OpenDNS, Google DNS and Choopa Geo DNS.
3. *VPN Managed - Private DNS:* We found that 6 VPN service providers operate their own private DNS service.

From the perspective of an attacker, these three configurations represent varying levels of difficulty. However, as shown below, all can be overcome. In this attack we assume the ad-



```
Destination     Gateway    Netif
10.0.0.0/24        *       wlan0
0/0                *       wlan0
```

```
IP        = 10.0.0.2/24
Gateway   = 10.0.0.1
Leasetime = 1min
```

```
Destination     Gateway    Netif
10.0.0.0/24        *       wlan0
0/0                *       wlan0
0/1                *       tun0
128/1              *       tun0
```

```
nslookup twitter.com
```

```
Server:   209.99.22.53
IN A 99.16.156.230
```

```
Destination       Gateway      Netif
10.0.0.0/24          *         wlan0
0/1                  *         tun0
128/1                *         tun0
0/0              209.99.22.53  wlan0
209.99.22.53/32   10.0.0.1     wlan0
```

```
IP        = 10.0.0.2/24
Gateway   = 209.99.22.53
Leasetime = 1min
```

```
nslookup twitter.com
```

```
Server: 209.99.22.53
IN A 10.0.0.100
```

Fig. 5. Hijacking the DNS through a route injection attack (OpenVPN tunnels)

versary controls the network's gateway (*e.g.,* the WiFi access point). Note that this assumption is not restrictive, as it falls within the typical threat model of commercial VPN services (*e.g.,* securing communications in an untrusted wireless network).

5.2 Hijacking default DNS configurations

The simplest scenario is where the VPN client does not change the victim's default DNS configuration (*e.g.,* HideMyAss over OpenVPN). In this case, subverting DNS queries is trivial. The access point can simply use DHCP to set the victim's DNS server to one that it manages itself. The adversary will then receive all DNS queries generated by the host.

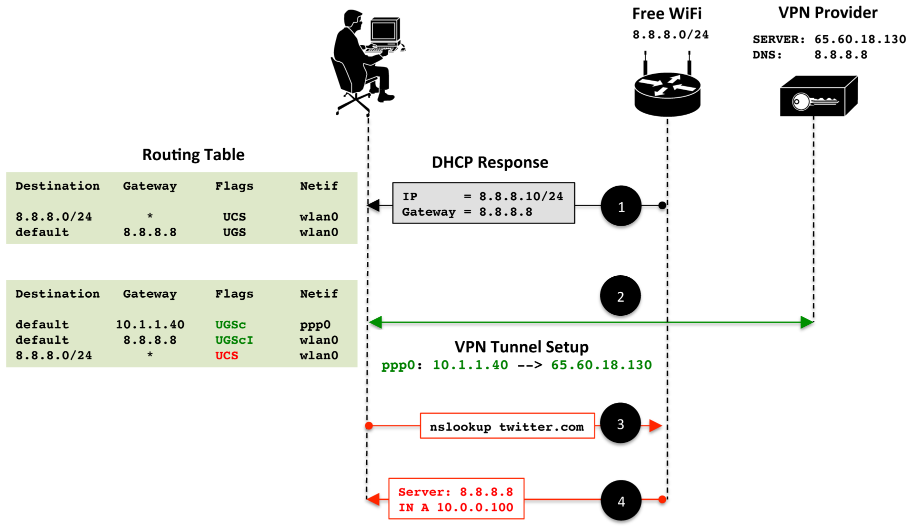
5.3 Hijacking VPN managed configurations

The next scenario occurs when the VPN client overwrites the existing DNS configuration with a DNS resolver specified by the VPN server during the tunnel setup. In such a case, the adversary must take extended steps to hijack the victim's DNS resolver, this time targeting its routing table. The idea is to trigger a configuration change that will make the DNS a local network resource, accessible via the LAN rather than through the VPN. This is possible because the VPNs studied operate under a *split-tunneling* mode, where only traffic directed towards the public Internet gets forwarded through the VPN tunnel; all local hosts (*e.g.,* network printers) are accessed directly. The way the attack is actually implemented depends on the VPN tunneling protocol used, as they modify the client's routing table in different ways. In particular, we found that OpenVPN clients and PPTP/L2TP clients have different ways of configuring the default routes used to forward the traffic through the VPN tunnel. We therefore discuss the two possible implementations (OpenVPN or PPTP/L2TP tunnels) separately below.

5.3.1 OpenVPN tunnels - Route Injection Attack

OpenVPN supports both layer-2 and layer-3 tunnels, implemented through the *tap* ant *tun* virtual interfaces, respectively. Upon tunnel setup, the VPN client needs to set a default route, forwarding the traffic through the secure tunnel. Rather than deleting the existing default route (set via DHCP), the VPN client manipulates the host's routing table by inserting two prefixes: `0/1` and `128/1`, as recommended by the OpenVPN manual [51]. As these are more specific than the `0/0` default route, all traffic is sent through the tunnel's virtual interface (usually `tun0` or `tap0`) instead of the host's native network interface. With this configuration, all user DNS queries are securely sent to the correct DNS server via the VPN.

An outline of the DNS hijacking, through a *route injection* attack, is depicted in Figure 5. The attacker, who controls the access point, first sets a low DHCP lease period (Step 1), forcing the victim to periodically re-request new DHCP information (*e.g.,* after 60 seconds). Through this, the adversary can use DHCP renewals to manipulate the end-host routing table at anytime after the VPN tunnel is established. In particular, the `gateway` option in the DHCP configuration forms



```
Destination    Gateway    Flags    Netif

8.8.8.0/24        *        UCS      wlan0
default        8.8.8.8     UGS      wlan0
```

```
Destination    Gateway    Flags    Netif

default       10.1.1.40   UGSc     ppp0
default        8.8.8.8    UGScI    wlan0
8.8.8.0/24        *       UCS      wlan0
```

Fig. 6. Hijacking the DNS (PPTP and L2TP tunnels)

the basis of the attack. The access point first creates a new virtual interface using the IP address of the VPN's DNS server. Detecting the VPN service provider used is trivial by passively monitoring the remote IP of the tunnel. The provider can then be mapped to the DNS server used by that particular server location.

After this, when the victim performs a DHCP renewal, the access point sets the victim's default `gateway` to that of the newly created interface (*i.e.,* the DNS server's IP address). Upon receiving the new DHCP configuration, the victim first checks if the specified gateway is locally reachable. Using ARP, the client will discover the available (virtual) interface created by the access point that has mirrored the DNS server's IP address. Following this, the client's routing table will be updated. However, since the new gateway is on a different subnet, a new entry is added for this specific IP address (*e.g.,* `209.99.22.53/32` in Step 5). From now on, all DNS queries will be forwarded directly to the fake interface on the access point, rather than through the tunnel. We have confirmed that this occurs without the VPN clients detecting the changes.

5.3.2 PPTP and L2TP tunnels

We found the above attack to be ineffectual for PPTP and L2TP tunnels with all VPN service providers studied. The reason is that clients set only one default route, `0/0`, as opposed to the two routes — `0/1` and `128/1`, in the case of OpenVPN. Importantly, before doing this, the existing default route is either removed or de-prioritised by binding it to the local network interface. As such, we find that any route subsequently injected into the routing table by DHCP `gateway` option gets ignored (as it has a lower priority with respect to the default route to the tunnel).

A different strategy (depicted in Figure 6) must therefore be used: The access point assigns the victim an address in a small bogus subnet that includes the DNS server used by the VPN. For instance, if the VPN's DNS server were `208.67.222.222`, then the victim would be assigned an address in the `208.67.222.0/24` subnet (*e.g.,* `208.67.222.10`). This bounds all the traffic towards the subnet, including that towards the DNS server, to the actual network interface (*e.g.,* wlan0) of the victim host. This interface would therefore get priority over the default rule imposed by the VPN client.

Note that this hijacking attack works as well with OpenVPN, although it is more intrusive. In fact, a key difference with respect to the previous attack is that changing the address of the victim through a DHCP renewal will temporarily disconnect the host from the VPN. This side-effect may be avoided if the adversary launches the attack pre-emptively, *i.e., before* the VPN tunnel is actually established, which would negate the need to leverage DHCP renewals.

Finally, note that the adversary has to be careful in excluding the entry point of the VPN tunnel from the bogus subnet. If that wasn't the case, the creation of the VPN tunnel would fail, increasing the chances that the victim notices the attack. While selecting the bogus subnet is trivial when a third-party DNS service is being used (*e.g.,* Google DNS in Figure 6), it becomes tricky if the provider uses a private DNS server, as the subnet must contain at least three IPs (*i.e.,* the DNS server, the gateway, and the victim host). For instance, we observed VyprVPN to use custom DNS servers whose address is very close to the VPN entry point (*e.g.,* 138.199.67.53 for the DNS, and 138.199.67.18 for the entry point). This configuration makes it impossible to split the two IPs into two different subnets, thus thwarting the attack (although perhaps involuntarily). Interestingly, no other VPN service using private DNSes (Table 1) appears to be using this kind of configuration. Further, noe that VyprVPN is still vulnerable to the route injection attack when OpenVPN is used.

5.3.3 Advanced DNS configurations - VyprDNS

A separate discussion has to be made for VyprVPN, whose DNS service – VyprDNS – provides extra security measures to make sure the configuration is working correctly. In particular, we observed that the tunnel setup fails if the client is using a different DNS server than the one managed by VyprVPN. By inspecting the traffic with `tcpdump`, it appears that immediately after the secure tunnel is configured, the VPN client generates 3 random domain name lookups, with all of them returning an error (NXDOMAIN). Importantly, we observed that whenever these queries are sent to a different DNS server, the VPN connection will be immediately closed. This suggests that the client separately contacts the VyprDNS server (using a bespoke protocol) to verify that the queries were properly received and answered, and if this is not the case, the VPN client reports an error and the tunnel setup fails.

Despite implementing this advanced DNS feature, we find that the check is only performed directly after the tunnel has been established. Therefore simply delaying the attack for 60 seconds (using the DHCP lease time), these checks can be circumvented. We experimentally confirmed the efficacy of the route injection attack on VyprVPN when using this delay.

5.4 Attack feasibility

Both versions of the DNS hijacking attack we presented require the adversary to control the DHCP server used by the victim host (*e.g.,* the WiFi router). We do not deem this assumption to be particularly restrictive, as it falls within the typical threat model of commercial VPN services (*e.g.,* securing communications in an untrusted wireless network).

A second, more restrictive requirement is to know the IP address of DNS server in use by the VPN at the victim host. To tackle this, the adversary could passively monitor the client-side IP of the VPN tunnel. This would reveal the VPN service used, which could then be mapped to the relative DNS server (*e.g.,* column “DNS” in Table 1). Note that the mapping may need to take into consideration location too, as we observed some providers to use different DNSes in different servers.

In the case of PPTP/L2TP DNS hijacking the adversary may need to guess the DNS server before the VPN is established. Google DNS and OpenDNS are typically good candidates, given their popularity (§ 1). This is more challenging when the VPN service provider manages its own private DNS service. In these cases, historical information about the victim’s preferred VPN service (and about the corresponding DNS) can be leveraged if the victim has been previously encountered.

Once the victim has been configured to forward its DNS queries to an adversary-controlled DNS server, traffic to all the domains resolved by the victim can be circumvented from the VPN. For instance, the adversary can resolve all the domains to a set of local web caches it operates, thus allowing the access point to seamlessly monitor all web traffic. Approaches such as DNSSEC have attempted to mitigate these risks, yet they are not widely deployed (under 3% of resolvers support DNSSEC [52]). To avoid detection, the access point could even use its own VPN (with the same provider) and forward all traffic; web-based checks (*e.g.,* using whatismyip.com) would therefore show the VPN exit point’s IP address (limiting suspicion).

5.5 Experimental results

We have tested the DNS hijacking against all the VPN clients listed in Table 1, confirming their efficacy. There were, however, some exceptions. The first one relates to Windows 8, which we found to be resistant to the OpenVPN route injection attack (§ 5.3.1) as a side effect of the way it manages its routing tables. More specifically, the `gateway` option in the DHCP renewal message does not result in a custom rule for the DNS server in the routing table (as opposed to Step 5 in Figure 5). For this reason, DNS queries will still be routed through the VPN, thus thwarting the attack. We believe other versions of Windows to be immune to this attack. Note, however, that Windows is still vulnerable to the PPTP/L2TP DNS hijacking (§ 5.3.2).

More recent versions of Android deserve special attention too. Starting from KitKat (Android 4.4.x), we discovered that Android uses firewall rules [53] instead of routing table changes to force traffic to be routed through the VPN tunnel. The firewall rules completely cut the device off from the local network, allowing traffic to be *only* routed through the VPN tunnel, thereby preventing the attack. We stress that, in any case, Android versions prior to KitKat (*e.g.,* JellyBean) are vulnerable to the DNS hijacking attack, whereas both JellyBean and KitKat (thus, potentially, earlier versions too) are still vulnerable to IPv6-leakage (§ 3).

Finally, besides VyprVPN (immune to the PPTP/L2TP DNS hijacking § 5.3.2), Astrill VPN deserves a special mention too, as it is the only VPN service we tested that is not vulnerable to both versions of DNS hijacking (*i.e.,* OpenVPN and PPTP/L2TP). The reason is that Astrill, by default, sets the same IP address for both, the DNS server and the VPN tunnel gateway, which makes it impossible for the adversary to produce a split tunnel and fool the victim host into believing that the DNS resides in the local network.

6 Countermeasures

We now discuss possible countermeasures for the two vulnerabilities.

6.1 IPv6-Leakage

The simplest countermeasure to IPv6 leakage is disabling IPv6 traffic on the host. Although feasible in some cases, not all OSes (*e.g.,* Android) allow applications to do this. Further, this can only be a short term solution in the face of expanding IPv6 adoption. More sensibly, VPN clients could alter the IPv6 routing table to capture all traffic. As previously mentioned, the DNS resolver also plays a crucial role; disabling AAAA queries would likely have a similar effect (although this would leave other resolver systems vulnerable, *e.g.,* BitTorrent trackers that support IPv6). Servers can also assist by exclusively using encrypted communications; this, for example, is the trajectory of Google [54]. However, this would not deal with all threats, as leakage would still occur.

6.2 DNS hijacking

To detect DNS hijacking attacks, an approach similar to the SmartDNS could be used. Unlike VyprVPN, though, the client should check the correct functioning of the DNS repeatedly (*e.g.,* every minute), instead of just at the tunnel's initiation. Still, there is always a chance for the client to detect the attack after user information has already leaked (even with fine grained intervals). A similar issue might affect any solution that monitors the host's routing tables for changes. We thus argue VPN clients should adopt more proactive solutions.

A viable option is that implemented on Android KitKat, that is, the use of firewall rules instead of the routing table to tunnel packets through the VPN tunnel. However, completely isolating the end-host from the local network, like KitKat does, may negatively impact the user experience (*e.g.,* this would leave the device unable to access any local network resource). Plus, the device would not be able to handle DHCP lease renewals, possibly disconnecting it from the Internet.

Another effective solution could be to take complete control of the DNS queries by making sure the DNS server can only be accessed through the tunnel. Configuring the gateway of the virtual interface to be also the DNS resolver (like Astrill does § 5.5) would make it impossible for the adversary to hijack the DNS queries with our attacks.

6.3 Tor

A solution to the attacks presented in this paper would be that of preferring Tor over a VPN. To securely tunnel client traffic, Tor sets up a local proxy that client applications must be explicitly configured to use, instead of using virtual network interfaces (like VPN clients do § 2.4). Client-to-proxy communications happen through the local loop, whereas connections between the local proxy and the Tor network always happen through secure and authenticated TCP sessions (*i.e.,* TLS). DNS queries can be performed directly through Tor, bypassing any locally configured DNS server. This operational mode thwarts all the attacks presented in this paper. For instance, in the case of dual-stack networks, proxy-to-Tor connections that happen through IPv6 would be secured, preventing private data leakage in the clear. Similarly, malicious attempts to hijack the host DNS configuration would be ineffective, as Tor does not use the address resolution mechanism at the host.

A key requirement, however, is that client applications must be correctly configured to use the Tor local proxy for all traffic. For instance, if the web browser does not forward address resolution requests to the proxy, an external observer will learn the user browsing history. For this reason, the recommended Tor client software distribution (the Tor Browser Bundle) includes a pre-configured Firefox that greatly reduces the risk for accidental leakage. Whether other applications can be properly "Torified" to achieve strong anonymity must be decided on a case-by-case basis [55]. In fact, even tunneling all the application traffic through Tor may not be sufficient to ensure anonymity. For instance, some BitTorrent clients are known to disclose the host IP address directly into the information they send to the tracker and/or to other peers [56], independently of any attempt to conceal it through Tor or a VPN tunnel.

Noteworthy projects are Tails and Whonix [57, 58]. These live Linux distributions transparently tunnel all Internet connections through Tor, and feature a range of applications (*e.g.,* browser, instant-messenger, mail client) selected and pre-configured with privacy and anonymity in mind. These distributions also aim to leave no trace of user activity on the host machine, thus thwarting threats such as cold-boot attacks and various memory forensic techniques. As such, they represent a viable and vetted option for users looking for hassle-free, system-wide online privacy and anonymity.

6.4 Enterprise VPNs

Differently from the commercial VPN services surveyed in this paper, an enterprise VPN gives individual employees secure access to their company network. Although enterprise VPNs might be exposed to these attacks, we argue that their impact is rather limited compared to commercial VPN services. Importantly, in this scenario, access to corporate resources is only possible via the secure tunnel. IPv6 traffic leakage is therefore not possible, as connections to the corporate network outside the VPN tunnel are not allowed (assuming all configurations are done correctly). Instead, hijacking the DNS could leak the names of the resources being accessed within the private enterprise network. However, unless the attacker has detailed knowledge of the corporate network, the name resolution will fail, and the user will notice the error and eventually stop using the VPN.

7 Related Work

To the best of our knowledge, Appelbaum *et al.* [17] were the first to provide a taxonomy of design issues that should discourage the use of VPN services for achieving privacy and anonymity on the Internet. Among other things, they experimentally observed the problem of IPv6 leakage in certain VPNs. We have built on their work to quantify the exact impact of their observations across multiple applications and VPN service providers. Fazal *et al.* [59] describe an attack that allows a rogue client to penetrate a VPN, exploiting VPN clients with a dual-NIC (*i.e.,* a WiFi and an Ethernet adapter) in a WiFi LAN. Gont *et al.* [60] describe a number of practices to prevent security exposures in IPv4 enterprise networks resulting from the native IPv6-support of general purpose operating systems. Among them, they show how a rogue client can impersonate an IPv6 router through Router Advertisement messages, and they discuss the potential of VPN traffic leakage on IPv6 [19].

Related to our study about IPv6 leakage (§ 4), Krishnamurthy *et al.* [40] examine how interconnections between visited sites represent a potential transparent leakage of personal information to third parties. Castelluccia *et al.* [47] instead show how targeted ads expose private user data, allowing the accurate reconstruction of user interest profiles. Olejnik *et al.* [61] experimentally study the uniqueness of web browsing histories, revealing that a large percentage of users have unique browsing histories. Perito *et al.* [62] investigate how usernames allow multiple service profiles belonging to the same user to be linked. Eckersley [63] studies web browser fingerprinting. To the best of our knowledge, we are the first to experimentally show how the most popular VPN services available today are vulnerable to this attack (§ 3.2), to quantify the leakage of popular applications (§ 4), and to study the effectiveness of two new DNS hijacking attacks (§ 5).

8 Conclusions, lessons learned and future work

In this paper we have presented an experimental evaluation of commercial VPN services. Whereas our work initially started as a general exploration, we soon discovered that a serious vulnerability, IPv6 traffic leakage, is pervasive across nearly all VPN services. In many cases, we measured the entirety of a client's IPv6 traffic being leaked over the native interface. A further security screening revealed two DNS hijacking attacks that allow us to gain access to all of a victim's traffic. The most alarming situation is where individuals use VPN services to protect themselves from monitoring in oppressive regimes. In such cases, users who believe themselves to be anonymous and secure will be in fact fully exposing their data and online activity footprint. In countries with state-maintained network infrastructures, it is likely that such monitors could take on the role of any of our adversarial models.

Throughout this study we realised that another worrying aspect of today's market of VPN services is the large misinformation end users are exposed to, which makes it hard for them to properly tell apart vague and bold claims typical of product advertisement campaigns with actual facts. For instance, a simple Google query such as "Tor vs VPN" returns, on the topmost positions, a number of web pages that are not affiliated with the Tor community in any way, even including one from a commercial VPN service provider website stating that, to achieve better privacy protection, a VPN service is likely a better option with respect to Tor.

In order to improve the current situation it is of primary importance to better reach out to the general public through active information campaigns. We believe that a more privacy conscious customer base would force VPN service providers to take serious actions towards securing their services and clients against issues that have been known to the community for a long time [17, 18]. At the same time, users would be able to choose the combination of technologies that better suit their needs.

We believe providing end-users with easily understandable proof of the security (or insecurity) of their VPN services may help mitigate this issue further. We will therefore continue analysing the interconnection between the VPN client and the host operating system, looking for other unusual and unpredictable interactions. In parallel, we will devise new attack strategies, so as to better understand how a concerted effort could undermine VPN security. Our long term goal is to integrate these findings into a user-friendly testing toolkit allowing end users to directly and independently measure the "quality and reliability" of their providers.

Ethical Considerations

Most data we have collected does not raise ethical concerns, as we have primarily probed public services via our own VPN service accounts. The only exception is our collection of BitTorrent users' IP addresses. This was necessary to quantify the ubiquity of VPN users; collecting user IP addresses could not be avoided as it was our only means to measure the usage of VPNs (by comparing against exit point IPs). Although not ideal, we believe it has provided valuable insight into wider privacy issues amortising the risk. To mitigate this issue, all data was securely stored and used *solely* for the aggregated analysis shown in Figure 4. All data was subsequently deleted.

Acknowledgements

We would like to thank our shepherd, Paul Syverson, and the anonymous reviewers for their insightful comments and suggestions.

References

[1] "Global surveillance disclosures (2013-present)," http://en.wikipedia.org/wiki/Global_surveillance_disclosures_(2013-present).

[2] R. Clayton, S. J. Murdoch, and R. N. Watson, "Ignoring the Great Firewall of China," in *Proceedings of the 6th Workshop on Privacy Enhancing Technologies*. Springer, 2006, LNCS vol. 4258, pp. 20–35.

[3] S. Khattak, M. Javed, S. A. Khayam, Z. A. Uzmi, and V. Paxson, "A Look at the Consequences of Internet Censorship Through an ISP Lens," in *Proceedings of the 14th Conference on Internet Measurement*. ACM, 2014, pp. 271–284.

[4] C. Abdelberi, T. Chen, M. Cunche, E. Decristofaro, A. Friedman, M. A. Kaafar *et al.*, "Censorship in the Wild: Analyzing Internet Filtering in Syria," in *Proceedings of the 14th Conference on Internet Measurement*. ACM, 2014, pp. 285–298.

[5] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*. USENIX Association, 2004, pp. 303–320.

[6] R. Stedman, K. Yoshida, and I. Goldberg, "A user study of off-the-record messaging," in *Proceedings of the 4th Symposium on Usable Privacy and Security*. ACM, 2008, pp. 95–104.

[7] "Whispersystems," https://whispersystems.org.

[8] "BestVPN," https://www.bestvpn.com.

[9] "Ultrasurf: the definitive review," https://blog.torproject.org/blog/ultrasurf-definitive-review.

[10] "Five Best VPN Service Providers," http://lifehacker.com/5935863/five-best-vpn-service-providers, 2014.

[11] "10 Reasons to Use a VPN for Private Web Browsing," http://netforbeginners.about.com/od/readerpicks/tp/Reasons-to-Use-a-VPN-Service.htm.

[12] "Naked Security Blog: What is your phone saying behind your back?" http://nakedsecurity.sophos.com/2012/10/02/what-is-your-phone-saying-behind-your-back.

[13] "Firesheep," http://en.wikipedia.org/wiki/Firesheep.

[14] "Session hijacking," http://en.wikipedia.org/wiki/Session_hijacking.

[15] N. Sastry, J. Crowcroft, and K. R. Sollins, "Architecting City-wide Ubiquitous Wi-Fi Access," in *6th Workshop on Hot Topics in Networks*. ACM, 2007.

[16] "5 Best VPNs in China," http://www.bestvpn-china.com/blog/9690/5-best-vpns-in-china-2014-update/.

[17] J. Appelbaum, M. Ray, K. Koscher, and I. Finder, "vpwns: Virtual pwned networks," in *2nd USENIX Workshop on Free and Open Communications on the Internet*. USENIX Association, 2012.

[18] M. Marlinspike, "Divide and Conquer: Cracking MS-CHAPv2 with a 100% success rate," https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2, 2012.

[19] F. Gont, "Virtual Private Network (VPN) traffic leakages in dual-stack hosts/networks," 2012.

[20] "Best VPN: 4 ways to prevent a DNS leak when using VPN," https://www.bestvpn.com/blog/5184/4-ways-to-prevent-a-dns-leak-when-using-vpn.

[21] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Proceedings of the 8th European Symposium on Research in Computer Security*. Springer, 2003, LNCS vol. 2808, pp. 116–131.

[22] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *Proceedings of the 8th International Conference on Financial Cryptography*. Springer, 2004, LNCS vol. 3110, pp. 251–265.

[23] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proceedings of the 7th Privacy Enhancing Technologies Symposium*. Springer, 2007, LNCS vol. 4776, pp. 167–183.

[24] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: Traffic correlation on Tor by realistic adversaries," in *Proceedings of the 2013 Conference on Computer & Communications Security*. ACM, 2013, pp. 337–348.

[25] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson, "One fast guard for life (or 9 months)," in *7th Workshop on*

Hot Topics in Privacy Enhancing Technologies. HotPETs, 2014.

[26] R. Dingledine and N. Mathewson, “Anonymity Loves Company: Usability and the Network Effect,” in *5th Workshop on the Economics of Information Security*, 2006.

[27] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg, “Changing of the guards: A framework for understanding and improving entry guard selection in Tor,” in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. ACM, 2012, pp. 43–54.

[28] “Tor Project Blog: Improving Tor’s anonymity by changing guard parameters,” https://blog.torproject.org/blog/improving-tors-anonymity-changing-guard-parameters, 2013.

[29] “Tor Project Blog: Traffic correlation using netflows,” https://blog.torproject.org/blog/traffic-correlation-using-netflows, 2014.

[30] B. Schneier, D. Wagner *et al.*, “Cryptanalysis of Microsoft’s PPTP Authentication Extensions (MS-CHAPv2),” in *Secure Networking - CQRE (Secure)*. Springer, 1999, LNCS vol. 1740, pp. 192–203.

[31] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey, “Measuring IPv6 Adoption,” in *Proceedings of the 2014 ACM Conference on SIGCOMM*. ACM, 2014, pp. 87–98.

[32] C. Metz, “Protocol independence using the sockets API,” in *FREENIX Track, 2000 USENIX Annual Technical Conference*. USENIX Association, 2000.

[33] “Happy Eyeballs: Success with Dual-Stack Hosts,” http://tools.ietf.org/html/rfc6555.

[34] “THC-IPV6 Attack Toolkit,” https://www.thc.org/thc-ipv6/README.

[35] “Hurricane Electric Free IPv6 Tunnel Broker,” https://www.tunnelbroker.net/.

[36] A. Dhamdhere, M. Luckie, B. Huffaker, A. Elmokashfi, E. Aben *et al.*, “Measuring the deployment of IPv6: topology, routing and performance,” in *Proceedings of the 12th Conference on Internet Measurement*. ACM, 2012, pp. 537–550.

[37] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, “I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis,” in *Proceedings of the 14th Privacy Enhancing Technologies Symposium*. Springer, 2014, LNCS vol. 8555, pp. 143–163.

[38] S. Chen, R. Wang, X. Wang, and K. Zhang, “Side-channel leaks in web applications: A reality today, a challenge tomorrow,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. IEEE, 2010, pp. 191–206.

[39] M. Marlinspike, “sslstrip,” http://www.thoughtcrime.org/software/sslstrip/.

[40] B. Krishnamurthy and C. E. Wills, “Generating a privacy footprint on the Internet,” in *Proceedings of the 6th Conference on Internet Measurement*. ACM, 2006, pp. 65–70.

[41] B. Krishnamurthy, D. Malandrino, and C. E. Wills, “Measuring Privacy Loss and the Impact of Privacy Protection in Web Browsing,” in *Proceedings of the 3rd Symposium on Usable Privacy and Security*. ACM, 2007, pp. 52–63.

[42] B. Krishnamurthy and C. Wills, “Privacy diffusion on the Web: a longitudinal perspective,” in *Proceedings of the 18th International Conference on World Wide Web*. ACM, 2009, pp. 541–550.

[43] “Alexa Top Sites,” http://www.alexa.com/.

[44] “Selenium WebDriver,” http://www.seleniumhq.org.

[45] “Google Play Unofficial Python API,” https://github.com/egirault/googleplay-api.

[46] N. Viennot, E. Garcia, and J. Nieh, “A Measurement Study of Google Play,” in *Proceedings of the 2014 ACM International Conference on Measurement and Modeling of Computer Systems*. ACM, 2014, pp. 221–233.

[47] C. Castelluccia, M.-A. Kaafar, and M.-D. Tran, “Betrayed by Your Ads!” in *Proceedings of the 12th Privacy Enhancing Technologies Symposium*. Springer, 2012, LNCS vol. 7384, pp. 1–17.

[48] “IPv6-enabled BitTorrent Peers,” https://www.vyncke.org/ipv6status/p2p.php.

[49] M. Defeche, “Measuring IPv6 Traffic in BitTorrent Networks,” 2012, IETF Internet Draft.

[50] “DNS Processes and Interactions,” https://technet.microsoft.com/en-us/library/dd197552(v=ws.10).aspx.

[51] “OpenVPN,” https://openvpn.net/index.php/open-source.html.

[52] A. Herzberg and H. Shulman, “Retrofitting Security into Network Protocols: The Case of DNSSEC,” *Internet Computing, IEEE*, vol. 18, no. 1, pp. 66–71, 2014.

[53] “Security Enhancements in Android 4.4,” http://forum.xda-developers.com/showpost.php?p=48703545.

[54] Y. Elkhatib, G. Tyson, and M. Welzl, “Can SPDY Really Make the Web Faster?” in *Proceedings of the IFIP Networking 2014 Conference*, 2014, pp. 1–9.

[55] “Tor Project: TorifyHOWTO,” https://trac.torproject.org/projects/tor/wiki/doc/TorifyHOWTO.

[56] “Tor Project Blog: Bittorrent over Tor isn’t a good idea,” https://blog.torproject.org/blog/bittorrent-over-tor-isnt-good-idea.

[57] “Tails: The Amnesic Incognito Live System,” https://tails.boum.org.

[58] “Whonix Operating System,” https://www.whonix.org/wiki/About.

[59] L. Fazal, S. Ganu, M. Kappes, A. S. Krishnakumar, and P. Krishnan, “Tackling security vulnerabilities in VPN-based wireless deployments,” in *2004 IEEE International Conference on Communications*, vol. 1. IEEE, 2004, pp. 100–104.

[60] F. Gont and W. Liu, “Security Implications of IPv6 on IPv4 Networks,” 2014, RFC 7123.

[61] L. Olejnik, C. Castelluccia, A. Janc *et al.*, “Why Johnny can’t browse in peace: On the uniqueness of Web browsing history patterns,” in *5th Workshop on Hot Topics in Privacy Enhancing Technologies*, 2012.

[62] D. Perito, C. Castelluccia, M. A. Kaafar, and P. Manils, “How unique and traceable are usernames?” in *Proceedings of the 11th Privacy Enhancing Technologies Symposium*. Springer, 2011, LNCS vol. 6794, pp. 1–17.

[63] P. Eckersley, “How unique is your Web Browser?” in *Proceedings of the 10th Privacy Enhancing Technologies Symposium*. Springer, 2010, LNCS vol. 6205, pp. 1–18.