Holes in the Geofence: Privacy Vulnerabilities in "Smart" DNS Services*

Rahel A. Fainchtein* Adam J. Aviv+ Micah Sherr* Stephen Ribaudo*
Armaan Khullar*

* Georgetown University + The George Washington University

Abstract

Smart DNS (SDNS) services advertise access to *geofenced* content (typically, video streaming sites such as Netflix or Hulu) that is normally inaccessible unless the client is within a prescribed geographic region. SDNS is simple to use and involves no software installation. Instead, it requires only that users modify their DNS settings to point to an SDNS resolver. The SDNS resolver "smartly" identifies geofenced domains and, in lieu of their proper DNS resolutions, returns IP addresses of proxy servers located within the geofence. These servers then transparently proxy traffic between the users and their intended destinations, allowing for the bypass of these geographic restrictions.

This paper presents the first academic study of SDNS services. We identify a number of serious and pervasive privacy vulnerabilities that expose information about the users of these systems. These include architectural weaknesses that enable content providers to identify which requesting clients use SDNS. Worse, we identify flaws in the design of some SDNS services that allow *any* arbitrary third party to enumerate these services' users (by IP address), even if said users are currently offline. We present mitigation strategies to these attacks that have been adopted by at least one SDNS provider in response to our findings.

*A shorter version of this paper appears in **Proceedings on Privacy Enhancing Technologies (PoPETS)**, July 2021.

1 Introduction

Vantage points matter on the Internet. Websites often customize or restrict content for clients based on their network locations and perceived geographic locations. This is especially true of media streaming services such as Netflix, Hulu, Pandora, and Amazon Prime Video, that are contractually obligated to restrict audio/video content based on their users' geographic locations. Such websites establish so called *geofences* that enforce location-based access control policies by geolocating clients based on their IP addresses.

However, determined users can apply simple methods to circumvent geography-based blocking by relaying connections through a proxy server located within the fence. Commercial VPN providers describe such abilities when marketing their services [16, 43]. Free solutions such as Tor [12] and open proxies [40, 54] also enable users to bypass geofences. However, popular existing approaches demand some user expertise and often require users to download and operate specialized software. Worse, previous studies show that the use of open proxies may incur severe security and privacy risks [40, 54].

There is a growing industry of *smart DNS* (SDNS) providers that enable an interesting and unstudied method of circumventing geofences. SDNS is simple and does not require additional software. Instead, a user reconfigures their computer's DNS settings to use an DNS resolver operated by a SDNS service. The SDNS resolver "smartly" identifies resolution requests for restricted domains (hereinafter,

fenced sites) and returns proxy servers' IPs in lieu of these domains' correct IPs. The client's machine then directs its traffic to the specified proxy server (since that is the address to which the domains resolve), which is located within the geofence. Finally, the proxy servers relay the clients' communication to and from these requested domains. For non-geofenced (hereinafter, *unfenced*) sites, DNS requests are resolved correctly. Thus, the end-user needs only browse as usual; all SDNS proxy management happens (potentially unnoticed) without additional interaction.

This paper describes an exploration of the privacy and security properties of smart DNS services—to the best of our knowledge, the first such study in the open literature. Through analyzing the architecture and behavior of deployed SDNS systems, we provide descriptions of how SDNS services operate.

Our analysis also uncovers several architectural weaknesses, implementation errors, and system misconfigurations that lead to pernicious privacy leaks, and are pervasive in the SDNS ecosystem:

We demonstrate a simple technique by which any content provider could immediately identify both the use of an SDNS service to access its site, as well as the actual IP address of the requesting client. (We note that numerous content providers have already sought to crack down on SDNS use, perhaps using the technique we describe here.) This would allow the content provider to consistently identify the use of SDNS, without requiring them to continually discover and block proxy servers, and, in so doing, engage in a never-ending "whack-a-mole" arms race with SDNS providers.

More troubling, we describe a design flaw in the architecture of SDNS systems that enables content providers to enumerate the IP addresses of an SDNS service's customers, regardless of whether they are logged in to the service's web portal, or currently use one of its SDNS resolvers for their web browsing. And, as we show through proof-of-concept attacks, the implementations of some SDNS services allow any arbitrary third-party to enumerate these SDNS services' customers. We discuss in detail the ethical considerations of our measurements and the proof-of-concept attacks we conducted.

We also identify a number of authentication and authorization errors, coupled with misconfigurations, that effectively turn some SDNS providers into a distributed network of open proxy servers. That is, we find that several SDNS providers fail to authenticate users who access their proxies, and instead rely only on authentication at their DNS resolvers. We present simple methods for enumerating such open proxies and explain how unscrupulous users could bypass paying for SDNS services while reaping their benefits.

We further find that some SDNS providers proxy more content than advertised. SDNS providers do this by forwarding traffic for websites, for which they do not advertise support, to proxy IPs. This raises the risk of content interception, manipulation, and eavesdropping, both by the SDNS provider and along the extended Internet path this traffic now traverses.

In addition to exploring the privacy and security properties of SDNS services, we also study the landscape of SDNS operators. Our exploration of SDNS services, conducted over more than ten months, strongly suggests that the SDNS marketplace may be more consolidated that it appears. Several of the identified 25 SDNS providers are actually the same entity advertising their services under multiple distinct names and websites. Our probes also exposed the popularity of different content for SDNS providers as well as the SDNS providers themselves. Applying current virtual private server (VPS) costs and advertised SDNS plan costs, we estimate the costs and revenues of SDNS services, and find that they are immensely profitable.

Relevance to Privacy. SDNS is provided by many existing VPN providers, perhaps due to overlap in infrastructure requirements, and SDNS is often advertised alongside VPN products. The manner in which SDNS is marketed differs among providers, with some implying (falsely) that SDNS is itself a privacy-enhancing technology [35, 57, 1]. We found no instances in which SDNS providers describe any added privacy risks.

SDNS does not appear to be a niche industry. At least two SDNS providers (`www.ibvpn.com` and `www.smartdnsproxy.com`) state that they have more

than one million users. Our own measurements largely support this claim.

Our main findings—SDNS customer IP addresses can be easily mined by third parties; SDNS substantially increases users' vulnerability to eavesdropping; and content providers can trivially discover when users attempt to bypass their geofences—all threaten the privacy and/or security of SDNS customers. Although SDNS may not itself be considered a privacy-preserving technology (although it is sometimes marketed as such), the architectural and implementation weaknesses we describe in this paper are relevant to the estimated millions of SDNS users, whose use of these systems may constitute significant and (until now) unexplored privacy risks.

2 Background on DNS

DNS [42] is the mechanism by which hostnames are mapped to IP addresses to facilitate Internet routing. DNS is complex with several important nuances, but conceptually, DNS can be thought of as a distributed database, with mappings between hostnames and their IP addresses stored in *zone files*. Ordinarily, the owner of the domain (i.e., the party that registers the domain) effectively controls this mapping.

Users *resolve*—that is, translate a hostname to its IP address—by querying a DNS resolver. Typically, users use the resolver that is provided in the DHCP response they receive when joining a network; often, but not always, these resolvers are operated by the ISP that provides Internet connectivity. Users also have the option of selecting a different resolver: popular choices include Google's DNS and Cisco Umbrella's DNS resolver.

When receiving a request, a resolver checks its cache for the queried hostname. If it finds an unexpired entry, the cached results are immediately returned. Otherwise, either the resolver returns a reference to another resolver (*an iterative query*) or, the resolver itself relays the request towards another resolver on behalf of the client (*a recursive query*) and ultimately returns the resolved IP. The resolver also caches a copy for a length of time that is defined in the corresponding zone file. The resolver that is responsible for a given domain is known as an *authoritative name server* and it is contacted in recursive queries when the answer is not cached by the other DNS resolvers. We found that all SDNS resolvers support only recursive queries.

While DNS supports both UDP and TCP, the former is much more common. DNS is typically neither authenticated or encrypted. To address this and improve privacy and security, there are three main extensions to DNS that offer additional privacy features: DNSSEC, DNS-over-HTTPS (DoH) [31] and DNS-over-TLS (DoT) [33]. DNSSEC aims to ensure the authenticity of DNS data by incorporating a PKI and using signed and verifiable zone files. (Friedlander et al. provide a good overview of DNSSEC [22].) DNSSEC does not address confidentiality of the DNS request, merely authenticity. DoH and DoT, on the other hand, both provide confidentiality of DNS requests and responses by using TLS. Importantly, SDNS is inherently incompatible with DNSSEC (since SDNS returns modified resolution results), and we found no SDNS providers that support either DoT or DoH.

3 Related Work

There are large, organized efforts at enumerating instances of Internet censorship [6, 20, 56] and there is considerable work that examines methods of bypassing blocking [12, 38, 53]. However, geo-filtering at the server-side is far less well-studied.

Afroz et al. [2] performed a large-scale measurement study and found that geo-filtering was ubiquitous on the Internet. A large number of commercial VPN providers, including many of those listed in Table 1, advertise their services as a means of getting around geo-fences. Khan et al. [37] and Weinberg et al. [59] independently analyzed the VPN ecosystem, with both sets of authors concluding that VPN providers regularly misrepresent the location of their endpoints. Interestingly however, so long as the fenced website similarly misattributes the VPN endpoint's location, this misclassification is not by itself problematic for users wishing to defeat geofences. Poese et al. measure the accuracy of geolocation services and find that errors are fairly common [44].

Server-side filtering of clients has also been explored in the context of preventing anonymous users

(e.g., Tor users) from accessing websites [61, 62, 39, 49]. The approaches used to detect access from anonymity networks [39, 49] and countermeasures to bypass such filtering [61, 62] are specific to the anonymity services being used and differ from the IP geolocation mechanisms used by content providers to impose geofences.

Numerous efforts have attempted to map out and explore the performance of the Internet's domain name system (cf. studies by Callahan and Allman [8] and Jung et al. [36], and measurement platforms such as the RIPE Atlas [47]). We also measure DNS performance (see Appendix D), but focus in this paper on the added costs incurred by choosing remote DNS resolvers. Finally, DNSSEC obviates the benefits of SDNS services by preventing the type of forged DNS resolutions on which SDNS depends (we discuss the impacts of DNSSEC in §4.1). However, DNSSEC has seen slow adoption and even the resolvers that support DNSSEC often fail to validate the authenticity of DNS records [9], indicating that SDNS will likely function for at least the short-term.

As explained more fully in the next section, the proxies used by SDNS providers inspect Server Name Indication (SNI) TLS headers [4] to extract the hostname requested by the client. Once the requested hostname is obtained, the proxies simply forward TCP traffic between the client and the destination. Such proxies are often called *SNI proxies*, and have been used as building blocks for domain fronting systems [19] (e.g., Tor's meek [12, 17]) and more generally for proxying of Internet traffic. Using ZMap [14] scans and a novel SNI proxy testing tool, Fifield et al. identify approximately 2500 *open* SNI proxies [18] that service public requests. We find that Fifield's list includes some SNI proxies operated by SDNS services, highlighting these services' failure to properly authenticate requests; we explore authentication errors in more detail in §8.

4 Architecture of SDNS Services

There are two phases of SDNS usage: registration and operation.

During the **registration** phase, a user must create an account on the SDNS service via the service's webpage and, depending upon the service, select a payment plan. The user must also register her public-facing IP address with the SDNS service. Many services simplify IP registration by presenting the detected IP address of the user as the default (and sometimes only) option. Next, the user must select a DNS resolver from a list of resolvers operated by the SDNS service. Many services advise the user to select an SDNS resolver that is geographically located nearby. This reduces the network latency incurred during DNS lookups, which, as we show in Appendix D, can significantly impact the user's overall browsing experience. Finally, the user must configure her computer to use the selected SDNS resolver as its primary DNS resolver. All SDNS services provide detailed instructions, complete with screenshots, on how to carry out this process.

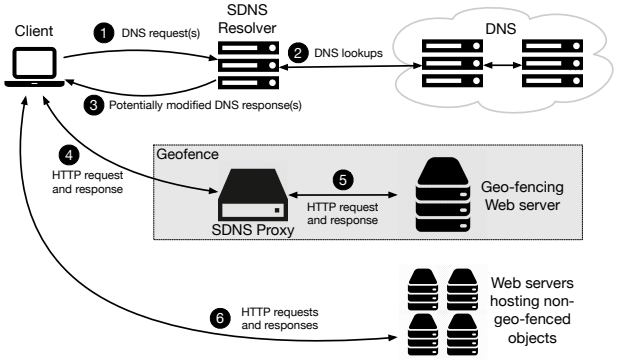


Figure 1: Workflow of the operation phase of a SDNS.

The **operation** phase is depicted in Figure 1. This begins when the user attempts to access geofenced content that is supported by the SDNS service. We adopt the terminology of many of the SDNS services and refer to a geofenced website proxied by an SDNS service as a *supported channel* or, more concisely, as a *channel*. (The term is likely inspired by TV channels; SDNS effectively allows its users to "tune" to "channels" that would otherwise be unavailable.)

Without loss of generality, consider a request for `https://netflix.com`, a channel supported by the user's chosen SDNS service. The user's DNS

requests—for `netflix.com` and for domains that host web objects referenced on that page (e.g., `fls.doubleclick.net`)—are sent to the SDNS resolver (step ❶). For each resolution request, the SDNS resolver either returns the correct IP address (e.g., via recursive lookups, as depicted in step ❷) or returns the IP address of one of its proxies that reside within the geofence (step ❸).

It is worth highlighting that SDNS depends entirely on IP-based authentication to determine whether the requesting user has completed the registration phase. If the user is not registered, the SDNS resolver's behavior differs by SDNS provider. Most providers return a correct (non-proxy) IP when resolving fenced content for non-customers. (As we show in §6, doing otherwise can lead to serious privacy vulnerabilities.) SDNS cannot support more robust forms of authentication since (i) DNS does not support requestor authentication and (ii) proxies cannot rely on web-based authentication mechanisms, such as cookies; HTTPS prohibits the proxy from inspecting session cookies, since TLS encrypts all content between the client and the website.

Returning to our example of a registered SDNS user accessing geofenced content, if the SDNS service supports `https://netflix.com`, the user's configured device will send HTTP/S requests to the proxy IP returned by the SDNS resolver (step ❹). The task of the proxy is twofold: first, it must determine which site is being requested since a single proxy may serve multiple channels (e.g., Netflix, Hulu, and ESPN). If the request is over HTTP, then the proxy can inspect the HOST HTTP header, which is mandatory in HTTP/1.1. For encrypted HTTPS traffic, SDNS exploits TLS' Server Name Indication (SNI) extension [4] that allows the requested site to be communicated as cleartext. SNI is intended to allow a single server to host multiple domains and serve the correct TLS certificate during the exchange. SNI is a popular TLS extension and has been found to be present in 99% of TLS connections [23]. In the context of SDNS, the use of SNI allows the SDNS proxy to interlope on the exchange and learn to which domain (e.g., Netflix) it should send proxy traffic.

Second, the proxy must actually forward the traffic (step ❺). SDNS proxies operate transparently and function as TCP endpoints for both the requesting client (where it poses as the web server) and the web server (where it poses as the client). SDNS proxies merely relay data received through one TCP connection to the other, and vice versa; doing otherwise would disrupt TLS (HTTPS) traffic between the client and the server.

The SDNS service does not necessarily have to proxy all web objects that are included in a requested webpage (step ❻). For example, the SmartDNSProxy provider does not proxy requests to `fls.doubleclick.net`, even though such web objects are referenced on `netflix.com`. This has the advantage that it decreases the proxy's workload and operating cost.

4.1 DNSSEC and Encrypted SNI

SDNS services are entirely incompatible with DNSSEC, since the latter provides origin authentication of DNS records. Of course, SDNS resolvers do not support the DNSSEC extensions, making this incompatibility moot until browsers and/or operating systems begin to require DNSSEC support.

Cloudflare co-introduced and adopted [25] encrypted SNI [46], which eliminates a privacy weakness of SNI by encrypting the requested hostname between the client and the receiving web server. Encrypted SNI would thwart the current SDNS architecture by hindering a proxy's ability to identify the site being requested. However, as of May 2020, encrypted SNI is either unsupported or not enabled by default in the latest release versions of Chrome, Firefox, Safari, Brave, and Microsoft Edge.

4.2 Contrasting with VPN Services

A stark difference between SDNS services and VPNs is that the former has no obvious on/off mechanism. VPNs require starting an application and authenticating to the VPN provider. In most settings, the VPN is not on by default, and there are visual indicators on the desktop when the VPN is in use. User intervention is also required to reestablish the connection to the VPN after machine reboots. That is, the use of the VPN requires *intentional* actions by the user.

In contrast, while SDNS providers offer their customers some helpful instructions and tools to con-

figure their DNS settings, SDNS is much less user-friendly with respect to activation or deactivation. There are no obvious indicators (other than the availability of certain video streaming services) that the SDNS service is in use. Due to this opacity in SDNS services, we posit that many SDNS users will forget the current status of their DNS settings and effectively *always* be performing DNS lookups through the SDNS' resolvers. We discuss the security and privacy implications of continuously using SDNS services in §6, as well as the performance implications in Appendix D.

SDNS is also unlike VPNs in that SDNS does not encrypt content between the user's device and the proxy. It is unable to do so, since its use is entirely invisible to the user's computer. For non-HTTPS traffic, SDNS increases the attack surface by allowing any potential eavesdropper between the client and the proxy to perform man-in-the-middle manipulation. In the case of HTTPS traffic, an eavesdropper can inspect SNI headers to learn which domain names are being requested.

4.3 SDNS Marketplace

To understand the SDNS marketplace, we performed simple Google queries to identify potential providers. We found that many SDNS providers are also VPN providers, advertising SDNS alongside VPN services and usually at a lower cost. SDNS, unlike VPNs, is not a privacy enhancing technology, but the commingling may confuse users about the security and privacy properties of SDNS. In at least two instances (ibVPN and VPNUK), SDNS providers marketed SDNS alongside their VPNs as privacy-enhancing services.

In total, we identified 25 SDNS providers. Using information available on their webpages, we catalogued (i) their prices and subscription plan offerings, (ii) the IP addresses of their DNS resolvers, and (iii) the countries in which the providers appeared to be registered.

The names, monthly costs, and locations of the 25 identified SDNS providers are listed in Table 1, including 15 providers which we focused on as part of our analysis. (These providers were selected based primarily on their search rank when querying Google for SDNS providers and their costs.) The 25 SDNS providers spanned 12 countries, where the country of origin was determined by searching for contact information (i.e., mailing addresses) listed on the providers' web pages. When searching for listed contact addresses, we also noticed that a number of the Turkish SDNS providers mention on their respective webpages that they belong to a single parent company.

Company Aliasing. During our analysis of SDNS services, we gathered evidence that strongly suggests that some of the SDNS providers are in fact the same company advertising under multiple name brands.

We identify numerous instances in which SDNS providers share infrastructure. To do so, we determine a (potentially incomplete) set of proxies used by each SDNS service by querying their DNS resolvers for supported channels (e.g., Netflix), and then comparing the returned IP addresses with a large ground-truth dataset, which was obtained by resolving the hostnames from a distributed network of RIPE Atlas nodes [47]. Our methodology for detecting shared proxies is described in more detail in Appendix A.

We find that the SmartDNSProxy, Trickbyte, and Uflix SDNS services share extensive infrastructure; specifically, we identify 10 proxies that are used by all three providers, seven that are shared between SmartDNSProxy and Trickbyte, and 14 shared proxies between Uflix and SmartDNSProxy. We additionally note that both Trickbyte and SmartDNSProxy's websites are served from the same /16 network, previously shared TLS certificate subjects, and are registered using the same domain registrar.

Upon additional inspection, we also discover evidence implying that CactusVPN and SmartyDNS are likely operated by a single entity. Specifically, these two poviders share at least four proxies, and exhibit similar proxying behavior patterns, which we describe in more detail in §8.

The rationale for operating multiple seemingly (but not actually) distinct SDNS services is unclear. We conjecture that such a strategy may attract more customers, since there are several sites that feature reviews and rankings of SDNS services.[1] Operating

[1]See, for example `https://thevpn.guru/top-smart-dns-`

as multiple services increases the chances of appearing at the top of at least some rankings. This is similar to findings that multiple VPN services may be operated by the same entity [37], perhaps also to gain advantage in VPN review and ranking sites.

5 System and Threat Models

There are several actors in the SDNS ecosystem: *SDNS providers* sell geofence-evading services to *customers* in order to provide more unfettered access to geofenced *content providers*. (We also refer to customers as *users*.) The SDNS infrastructure is composed of one or more provider-operated *resolvers* and one or more *proxies*. Additionally, the SDNS resolvers depend on the *traditional DNS infrastructure*, since customers' DNS queries correspond not just to supported content providers (e.g., netflix.com), but also to unproxied domains (e.g., petsymposium.org).

This paper explores the privacy and security implications of SDNS to both customers and SDNS providers, and thus we consider two separate threat models:

Customer Threats. This paper considers attacks on customer privacy that expose a customer's IP address, either to the content provider or to any outside party. We note that such exposure could potentially present legal risks to SDNS users[2], or result in users being banned by content providers.

It is worth emphasizing that, as with other work (cf. [12, 63]), we consider IP addresses to be sensitive information. Indeed, the EU's General Data Protection Regulation (GDPR) and the California Privacy Protection Act of 2018 both consider a user's IP address to be personally identifiable information under certain circumstances [11, 10]. We describe the state-of-the-art in mapping IP addresses to specific individuals in Appendix B, but highlight here that IP-to-individual mappings are commercially available (e.g., from Experian) and that IP-to-individual search engines are also available (e.g., `https://thatsthem.com/reverse-ip-lookup`).

`proxy-providers/` and `http://www.bestsmartdns.net/`.

[2]In the U.S., the use of SDNS may technically violate the Computer Fraud and Abuse Act, which criminalizes "exceed[ing] authorized access" of a computer system and imposes civil liability on the perpetrator [55].

Additionally, we argue that the exposure of customer IP addresses to *any* arbitrary outside party falls well outside of the norms that users expect from their Internet services. We can think of no other example in which a service allows outside parties to enumerate all of its users. Perhaps more importantly, we did not find any SDNS service that advises its customers about any such exposure.

Finally, we consider customers' increased vulnerability to traffic analysis due to the use of SDNS. We consider the additional risk not just to traffic directed towards content providers (which would take longer paths due to proxying) but also other more general Internet traffic that users may not expect to be proxied.

Provider Threats. We also explore the privacy and security risks of operating an SDNS service. These consists of vulnerabilities that either (1) harm the operation of the SDNS provider or (2) reveal potentially sensitive information about its operation.

More concretely, such threats include fundamental weaknesses in the SDNS architecture that allow a content provider to detect (and thus block), in real-time, the use of SDNS. (We note that this is a more powerful attack than attempts to enumerate SDNS proxies, since it avoids an arms race between discovering proxies and spinning up new proxies.) Additionally, we consider to be in-scope attacks that target the financial operation of an SDNS service and allow users to bypass payment and effectively access the service for free.

Finally, our threat model includes the exposure to analytics that enables outsiders to perform competitive analysis on the SDNS provider. This includes the ability of a third-party to estimate the number of users and revenue of an SDNS service, as well as to gauge the relative popularity of the channels that it proxies.

Adversaries and adversarial capabilities. We consider several adversaries that pose threats to either customers or providers. Our **content provider adversary** operates a channel that is targeted for geofencing bypassing by an SDNS provider. The content provider has the ability to modify its website and inspect its own web logs.

The **network eavesdropper adversary** is a passive network observer. We consider two variants of our network eavesdropper: an eavesdropper who is located between the client and the client's SDNS resolver, and a network eavesdropper that is located between the SDNS proxy and the destination (geofenced) website (see §7). An example of the former is the SDNS user's ISP; an example of the latter is a government or AS that monitors or hosts the proxy server. The network eavesdropper can inspect intercepted packets. For the eavesdropper who observes DNS traffic, our attack is effective when the DNS request and response are not encrypted. We are not aware of any SDNS service that supports encrypted DNS resolution (i.e., with either DoT [33] or DoH [31]).

We also present a number of attacks that can be carried out by nearly any arbitrary third-party Internet user; we call this adversary the **Internet user adversary**. The Internet user adversary can exploit the authentication and authorization failures we identify in §8 to use SDNS services without having to pay for them. We show that such attacks are possible and can be carried out by any Internet user.

An Internet user adversary can also probe the caches of SDNS providers' DNS resolvers to infer information about the popularity of the SDNS providers as well as which channels are most often used by the providers' customers (see §9). Here, we require that our Internet user adversary be able to identify the DNS resolvers used by an SDNS provider. We note that resolvers are listed on SDNS providers' websites since SDNS customers need such information to configure their computers to use SDNS.

Finally, the Internet user adversary can carry out the customer enumeration attack (see §6.1). Here, three additional capabilities are needed: the adversary needs to be able to (1) register a domain name (of the adversary's choosing), (2) operate the authoritative name server for that domain, and (3) be capable of sending spoofed UDP packets.

We summarize our main security and privacy findings in Table 2. We note that our threat models exclude geofence circumvention. Although this may be reasonably considered a security threat to content providers (since it bypasses an authentication check), this is the intended function of SDNS services. Our focus in this paper, rather, is to shed a light on the previously undocumented privacy and security risks of SDNS.

6 Privacy Vulnerabilities in SDNS Designs and Implementations

SDNS' architecture and implementations lead to several privacy and security risks, which we describe below. For each vulnerability, we list the relevant threat model defined in §5.

6.1 Client Enumeration Attacks

Threat model(s): Customer

Standard DNS does not support client authentication, and hence SDNS providers must rely on IP-based authentication to identify customers. The use of IP-based authentication, coupled with the ease at which UDP-based DNS requests can be forged leads to serious privacy vulnerabilities. (All tested SDNS services support UDP-based DNS.)

We discovered architectural weaknesses in two SDNS services that allow a third-party attacker (the *Internet user adversary* described in §5) to query the SDNS service and, in so doing, determine whether a target IP address belongs to one of its registered customers. When repeated, this attack allows the attacker to enumerate the IP addresses of these services' customers. For ease of exposition, we refer to an IP address associated with a customer of the SDNS provider as being a *registered* IP. The attack requires no client interaction and will reliably reveal whether an IP address is registered even if the customer is not actively using the SDNS service, or even if it is not currently online.

As discussed in §5, an adversary who learns the IP addresses of SDNS users could potentially also combine this information with existing IP-to-individual to determine the users' identities. This, in turn, could enable targeted cease and desist notifications. Even without resolving particular identities, knowledge of SDNS users' IP addresses is sufficient to deliver abuse notifications to the operators of the users' networks

(e.g., their ISPs), akin to how movie and music trade associations communicate their perceived violations of the U.S. DMCA.

The client enumeration attack requires only that the adversary (i) registers a domain name (of the adversary's choosing) and (ii) operates its own authoritative domain server for that domain. The adversary can be located far from the SDNS service and need not intercept any messages destined to SDNS resolvers. While the attacker can be any Internet user who meets the above criteria, we posit that content providers, trade associations, and content producers (or their copyright holders) might be especially motivated to enumerate the users of SDNS.

The attacker exploits a specific SDNS behavior in which the service's DNS resolvers send distinct responses to customers' and non-customers' requests. At a high level, the adversary uses these two different behaviors to deduce whether an arbitrary IP address is a customer of the service or not. This process can then be repeated for all IPv4 addresses (or more likely, a target set of IP addresses for which the attacker is interested).

Figure 2 presents an overview of our attack. To determine whether an arbitrary IP address, say `1.2.3.4`, is registered, an attacker who controls the domain `attackerdomain.com` forges an otherwise well-formed DNS request to the SDNS resolver for `nonce.attackerdomain.com`, purportedly originating from `1.2.3.4` (step ❶ in Figure 2, *left*), where `nonce` is a unique identifier. If `1.2.3.4` is a registered IP address, the forged query for `nonce.attackerdomain.com` would cause the SDNS resolver to correctly resolve `nonce.attackerdomain.com` to its IP address (step ❷, *left*) via recursive DNS lookups. This would be the case, as the hostname `nonce.attackerdomain.com` does not correspond to any channel supported by the service.

We emphasize that to support general web browsing, SDNS resolvers must correctly resolve hostnames for domain names they do not support. Additionally, the use of a unique nonce prevents `nonce.attackerdomain.com` from being cached at the resolver. This ensures that the request is propagated to the authoritative name server for `attackerdomain.com`, where it can be observed by the adversary. Finally, the IP address for `nonce.attackerdomain.com` (or an error if not found) is relayed back to the SDNS provider's resolver (step ❸, *left*) and forwarded onto the forged IP address X (step ❹, *left*), where it is likely discarded.

The right-side of Figure 2 shows the alternative case in which the IP address `5.6.7.8` is tested and is not registered with the SDNS provider. Here, we rely on a particular behavior of certain SDNS providers; namely, that they *do not* resolve requests from non-customers. We found two slight variations of susceptible behavior. The ibVPN SDNS service responds to non-customer hostname resolution requests by returning a fixed IP address belonging to a website it operates; the website redirects the user to an error page. This scenario is depicted in step ❷ in Figure 2, *right*. In contrast, the VPNUK service does not respond at all to non-customer DNS resolution requests.

Both behaviors allow the attacker to determine whether an arbitrary IP address, in this case `5.6.7.8`, is a customer: if it is, it will receive the recursive lookup request from the SDNS resolver and can observe this request at its authoritative name server; if `5.6.7.8` is not a customer, the request will not appear.

To validate our attack, we performed a proof-of-concept experiment using the ibVPN and VPNUK SDNS providers. We discuss the ethics of our experiment in §11. Our procedure was identical for both systems: we purchased an account on the system and registered our client IP address. We also purchased a domain name and configured an authoritative name server (hosted at Georgetown University) for that domain. We confirmed that requests originating from our client's IP to resolve a unique subdomain were recursively resolved and observed at our authoritative name server.

Next, we confirmed that requests sent from an unregistered IP (also operated by us), either yielded false static responses (ibVPN) or no responses at all (VPNUK); in both cases, the requests originating from the other, non-registered IP address did not propagate back to our authoritative name server.

Finally, to complete the attack, we acted as the at-

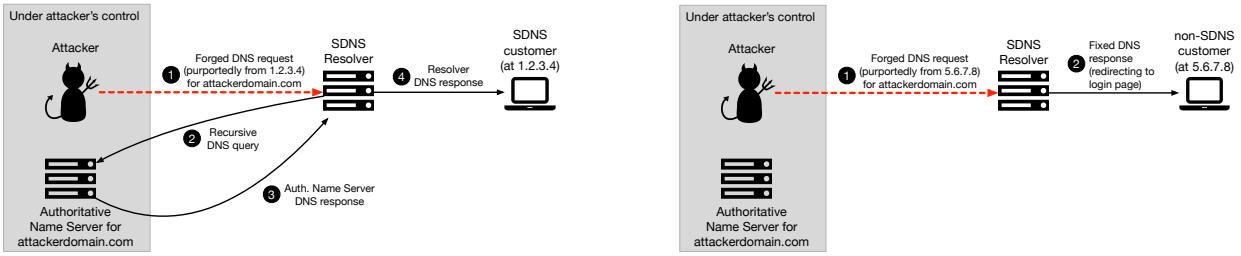


Figure 2: The two possibilities for the client enumeration attack: either the candidate IP address belongs to an SDNS customer *(left)* or not *(right)*.

tacker using a third IP on a different network. The attacker forged two requests: one purportedly from the registered IP and one from the non-registered IP. We confirmed that only the forged requests that purported to be from the registered IP address were relayed to our authoritative name server.

IPv4-space enumeration. We used the ZDNS tool from the ZMap Project [14] to estimate the service capacity of our institution's (i.e., Georgetown University's) DNS server. ZDNS performs highly parallel DNS lookups using lightweight Go threads, and is useful for efficiently resolving a large number of domains against a DNS resolver. We emphasize that we did not use ZDNS against the ibVPN or VPNUK resolvers since ZDNS could potentially disrupt their services. We use the performance measurements of our institution's DNS resolver only to form a rough estimate of the length of time it would require to enumerate all 2^{32} potential IPv4 addresses.

We find that Georgetown's DNS resolvers can resolve the top 10,000 Alexa sites in 7.462s (1340.12 requests/second), while consuming less than 1 MBps. At this rate, it would require approximately 5.3 weeks of continuous queries to enumerate all possible customer IPs (again, under the assumption that the SDNS provider's resolver has comparable performance). While such a sustained rate of access is likely unrealistic, we note that large ISPs can be fully enumerated in under a day (e.g., Comcast has approximately 71 million IP addresses [34]).

Mitigations. Our attack relies on SDNS resolvers that resolve an attacker-controlled domain only when the (purported) requester is a customer. The attack can be partially mitigated by consistently and correctly resolving domains for all hostnames that are not associated with a supported channel. Indeed, one day after we disclosed our attack to ibVPN, the ibVPN service implemented this mitigation.

We emphasize that although this fix disallows arbitrary third-parties from enumerating customers, it will not prevent the operators of a supported channel (e.g., Netflix) from carrying out the attack. For example, the content operator can register subdomains (e.g., *nonce*.netflix.com) and forge a DNS request from a candidate IP X to determine if X is associated with a customer of the SDNS service. The SDNS provider cannot apply the above fix here, since to route around geofences, it needs to respond to the client with an incorrect resolution (containing the IP address of a proxy) when the requested site is a supported channel.

A more robust mitigation is for the SDNS resolver to accurately resolve all resolution requests. When the requested hostname corresponds to a supported channel, the SDNS resolver can ignore the correct IP address and instead return the address of its proxy to the requesting client. However, while this fully mitigates the attack, it also allows a content provider (i.e., channel operator) with knowledge of the SDNS service to precisely measure how often the SDNS service is being applied to bypass its geofilter: it can inspect its own authoritative name server's logs for relayed requests from the SDNS resolver.

6.2 De-proxying by the Content Provider

Threat model(s): Customer & SDNS Provider

It is relatively straightforward for a geofenced content provider (i.e., a website operator) to (i) detect and prevent access from an SDNS service and (ii) identify the true IP address of the SDNS customer. A *de-proxying attack* requires the content provider to insert content into its web page that *does not require a DNS lookup.* By causing DNS resolution to be skipped, the content provider prevents the use of a proxy and forces the client to perform a direct access.

Without loss of generality, consider a content provider `istreamvideos.net`, where `istreamvideos.net` resolves to the IP address `1.2.3.4`. To perform a de-proxying attack, the content provider serves the partial content `<IMG src="https://1.2.3.4/image.jpg?session_id">` where `session_id` is a unique tag that can link the web requests to `istreamvideos.net` with those to `1.2.3.4`.[3] The client's browser will process the above IMG tag and *directly* access the image at `1.2.3.4` since the (unused) SDNS resolver loses the opportunity to return the IP of a proxy. The content provider then checks whether the two linked requests originated from the same IP; significantly differing requesting IP addresses (e.g., from different autonomous systems) indicates the use of an SDNS service.

As a proof-of-concept, we performed a de-proxying attack against ourselves, using the Hide-My-IP SDNS service. Hide-My-IP proxies *all* connections: its DNS resolver returns the IP address of a single proxy regardless of the requested domain. When the proxy receives HTTP requests from the client, it inspects the HOST HTTP header or the SNI TLS header to identify the requested destination, and then acts as a transparent TCP proxy. Since Hide-My-IP proxies all sites, we can trivially become a "content provider" by simply instantiating a web server. As described above, we constructed a simple web page that included an IMG tag whose source ("src") was specified by IP rather than hostname. We confirmed that our web server logs revealed that the domain-based request for the webpage had a different requesting client than the one for the IP-specified image; the former showed the proxy IP address and the latter revealed the client's IP address.

The deproxying attack enables a content provider to learn which of its users use SDNS. Unlike the client enumeration attacks described in §6.1, the deproxying attack may directly implicate a user of the content provider if the provider requires users to first log in before accessing content. It also provides a real-time mechanism for *immediately* detecting the use of SDNS, and is thus a far more practical means of protecting against geofence circumvention than frequently enumerating all SDNS users. Once an SDNS user has been identified, the content provider could either suspend or terminate that user's account, or simply disallow use of the service while SDNS is in use.

There are no clear mitigations to a de-proxying attack, and moreover, de-proxying attacks are particularly worrisome for users who misunderstand the privacy properties of SDNS services. While SDNS services do not advertise anonymity, end-users could be confused about the kinds of protections (or lack thereof) that these services provide, especially when these same providers sell VPN services as their primary offering. This confusion could put end-users in restrictive regimes at particular risk, if they access censored content with an expectation that their accesses are anonymous.

The deproxying attack also presents a threat to SDNS services. We found instances in which content providers blocked access from a handful (but not all) SDNS proxies. This indicates a "whack-a-mole" defense in which content providers attempt to identify and block proxies. This arms race generally works in the SDNS provider's favor, since cloud-hosted proxies can easily change IP addresses. (This same whack-a-mole strategy is also used to find VPN services' egress points.) The deproxying attack avoids this arms race by identifying in real-time the use of SDNS, and thus enabling immediate discovery of SDNS proxy servers as soon as they are utilized. In short, the content

[3] Although it is not particularly common (or advised), some certificate authorities (e.g., GlobalSign [26]) will issue IP-based certificates.

provider can apply this attack to entirely eliminate the utility gained by using an SDNS service.

7 Susceptibility to Eavesdropping

Threat model(s): Customer

SDNS services increase their users' susceptibility to eavesdropping. We explore this increased risk across two dimensions: eavesdropping on DNS requests and eavesdropping on proxies.

7.1 Eavesdropping on DNS requests

A log of DNS queries provides a fairly complete record of which sites and services were accessed by a requestor. SDNS customers configure their computers to send *all* DNS queries to SDNS resolvers, regardless of whether the queries pertain to fenced or unfenced websites. This provides SDNS services with a comprehensive set of potentially sensitive metadata about their customers. We emphasize that this is in stark contrast to using VPNs, whose use can be easily toggled on and off and whose active use is typically indicated by visible cues presented to the user. That is, the "set and forget" configurability of SDNS services, described in §4, has important implications to users' privacy.

Longer paths increase susceptibility to eavesdropping. The architecture of SDNS services risks exposing their users' Internet metadata to third parties, beyond the SDNS provider. DNS requests and responses are usually (and, in the case of SDNS, we believe always) sent unencrypted[4], allowing eavesdroppers between the client and the resolver to learn which hostnames are being requested, and by whom.

For regular (non-SDNS) DNS resolution, the resolver is typically located near the requestor, and is often operated by the requestor's ISP, which we note, learns the sites being requested by virtue of forwarding their traffic. That is, using a local resolver poses little additional privacy risk. Public DNS services, such as those offered by Google, Cloudflare, and Cisco, serve as popular alternatives to relying on local resolution, especially among more technically sophisticated users. We emphasize that the public DNS infrastructure offered by Google, Cloudflare, and Cisco all use IP anycast and are backed by highly distributed networks [28]. For example, DNS resolution requests to the fixed IP address of Google's Public DNS resolvers will often be routed to a resolver located close to the requesting client [30].

Compared with local DNS resolution and with resolution via large, public DNS providers, resolution via SDNS resolvers causes the requests (and responses) to transit longer network paths. We confirm this by counting the number of autonomous systems (ASes) traversed between clients and (1) Google's and Cloudflare's public resolvers and (2) 108 identified SDNS resolvers. We determine the number of ASes by performing traceroutes and using the utility's built-in IP-to-ASN mapping, and then counting the unique ASes observed in the reported network paths. More AS traversals indicate longer paths and consequently increased vulnerability to eavesdropping, since more organizations have the ability to observe the traffic. We place our traceroute clients in five continents, and report our results in Table 3. We find that the average number of AS traversals between the client and its chosen DNS resolver increases when the client elects to use a SDNS resolver. The relative increase in the number of AS traversals ranges from 13% (Japan) to a near tripling in length (Belgium), relative to using Google's or Cloudflare's public DNS service; in the United States, the average number of ASes that observe the DNS requests increases by 55% when SDNS is used. Finally, we note that the use of distant DNS resolvers has been found to be a significant threat to privacy in the context of Tor [30]. We emphasize, however, that unlike with Tor, SDNS users send *all* DNS requests to potentially distant DNS resolvers, not just those that are produced when temporarily browsing anonymously with a specialized browser.

7.2 Eavesdropping on Proxies

SDNS customers are also exposed to an increased risk of eavesdropping through the use of the proxies themselves. Communicating via a proxy increases the surface area for eavesdropping. While directly accessing

[4]The Firefox web browser now uses encrypted DoT [33] to Cloudflare's DNS resolvers by default. However, SDNS users would need to disable this setting.

sites generally uses the shortest paths in terms of the number of autonomous systems traversed [21], relaying traffic through an SDNS proxy requires that it first be transmitted to the proxy and that the proxy separately transmit it to its destination. This process produces longer paths that are more vulnerable to eavesdropping.

These long paths are especially risky in the case of SDNS services since connections between users and their proxies are not encrypted. A user may use HTTPS to achieve end-to-end confidentiality of content with the visited website, but the widespread use of SNI allows an eavesdropper situated either between the user and the proxy or between the proxy and the destination to learn the hostnames of all requested URLs.

How much the eavesdropper can learn from this leakage mainly depends on what traffic the SDNS provider proxies. At the extreme, the HideMyIP SDNS service proxies all web requests, regardless of the requested webpage. Clearly, this leaks significant information to an on-path, passive eavesdropper and causes the SDNS provider to incur a very high bandwidth cost.

However, even in cases where SDNS providers take steps to limit unneccessary proxying, they likely still leak substantial information about their users' Internet browsing habits. The SDNS provider ultimately decides which domain names it will route to its proxies and which it will allow its users to access directly. As noted in §4, SDNS services can limit unnecessary proxying by only proxying the content required to make their supported channels work—for example, just those web objects that consider the client's location and enforce the geofence. This can become problematic when a supported channel runs its geo-ip checks on a large CDN and references it using a ubiquitous domain name. In one such case, we noted that Netflix runs one of its geo-ip checks on an Akamai node (akamaihd.net). This effectively requires the SDNS provider to proxy *all* content to akamaihd.net (including that which is not related to any supported channel). For example, the SmartDNSProxy SDNS service proxies some Akamai-hosted webobjects on the Honda motorcars website, despite Honda not being a supported channel.

This "over-proxying" allows an eavesdropper situated between the client and the proxy to learn not only about visits to supported channels, but also other sites that happen to use the same CDN nodes as those channels. We note that although such information may be encrypted, the now-ubiquitous use of SNI may leak information about requested hostnames.

Unadvertised proxying. Given SDNS providers' opportunity to limit costs by only proxying domain names needed to support their advertised channels, we expected SDNS providers to support only the channels that they advertise. However, we additionally identified several instances in which this was not the case. To discern instances of unadvertised proxying, we queried SDNS resolvers for domains from the Alexa website rankings list, and then compared the results to a ground-truth dataset we collected by issuing queries from a distributed collection of RIPE Atlas proxies as well as queries using Google's, Cloudflare's, and our local institution's DNS resolvers. (We exclude HideMyIP from this analysis, since it proxies all connections regardless of destination.) A more detailed explanation of our methodology is presented in Appendix A.

We find that four SDNS providers (SmartDNSProxy, TrickByte, Uflix and VPNUK) omit supported domains from their published channel lists.

For the most part, the uncovered unadvertised channels correspond to pornographic websites. We posit that these sites are intentionally omitted from SDNS providers' websites to avoid detracting from the providers' perceived legitimacy or professionalism. As with over-proxying of domains, the failure of SDNS providers to announce the proxying of these channels poses privacy risks to their customers, since accessing these sites likely traverses longer Internet paths than would occur via direct connections. Beyond the longer paths incurred, all proxied sites leak the SNI hostname of websites visited over a TLS connection. As such, the aforementioned (passive, on-path) eavesdropper can learn that these users access pornographic sites, which sites they visit, and the frequency at which they do so. Moreover, the

SDNS provider can change which domains it proxies at any time, and without warning. As a result, the eavesdropper could potentially gain insights into additional aspects of users' browsing behavior.

8 Authentication and Authorization Failures

Threat model(s): SDNS Provider

As discussed in §4, the workflow of SDNS is a two-step process: (i) upon receiving a DNS resolution request for a supported channel, an SDNS resolver returns the IP address of one of its proxies, and (ii) upon receiving HTTP/S requests from the end user, the proxy then either inspects the `Host` HTTP header or the TLS Server Name Indication (SNI) extension field to infer the destination, and then forwards TCP traffic to and from the location inferred. Critically, SDNS providers should perform authentication (*is the requesting user a registered customer?*) and authorization (*should the traffic to the requested site be proxied?*) at both of the above steps.

In prior work, Fifield et al. used custom ZMap scans [14] to identify approximately 2500 *open SNI proxies* in the wild that used SNI introspection to proxy HTTPS traffic for arbitrary Internet users [18]. We compared our list of identified SDNS proxies to the open SNI proxies found by Fifield et al., and discovered four IP addresses on both lists, indicating that at least some proxies operated by SDNS services do not properly perform authentication. That is, they allow non-paying customers to directly use their proxies to relay traffic (e.g., to bypass geofences). We confirmed that the SDNS proxies allowed non-registered Internet users to proxy content by manually setting the SNI header in HTTPS requests originating from an IP that is not registered with the SDNS service. In all cases, the proxies retrieved the requested content.

To explore whether authentication failures were due to infrequent configuration errors on a small subset of a service's proxies or endemic misconfigurations across all of its proxies, we identified additional proxy servers for eight SDNS providers (see Table 4). To find additional proxies, we noted that SDNS proxy servers sometimes presented distinctive error messages when accessed directly over HTTP, without a modified `Host` HTTP or SNI header that indicated an alternate destination. (Fifield et al. made a similar observation of open SNI proxies [18].) These error messages often warned the user of a DNS misconfiguration and directed her back to the SDNS provider's website. We queried censys.io for this distinctive text to discover more potential proxies.

Table 4 shows the number of proxies we identified for each service, along with whether the proxies were restricted to SDNS customers (open circles) or functioned as open proxies (darkened circles). We tested whether a proxy was open by specifying `Host` HTTP headers (for non-encrypted web traffic) and TLS SNI headers (for encrypted web traffic) in an attempt to proxy. We found that CactusVPN, HideIPVPN, and SmartyDNS all had endemic authentication errors; all of their proxies functioned as open SNI proxies for any requesting Internet user. Oddly, we found no instances of open proxying for unencrypted traffic, even among those three providers. The authentication checks—which are based solely on the requestor's IP address—are performed only for HTTP proxying. We note that Google reports that between 74 to 94% of web requests using the Chrome browser (depending upon computer platform) are over HTTPS [27], suggesting that the failure to authenticate HTTPS requests is sufficient for non-customers to access the majority of the web.

We additionally checked whether the identified proxies would forward traffic to any domain, or limit proxying to its supported channels (as determined by its responses to DNS resolution requests with a proxy's IP address). We use the term *universal* to refer to proxies that forward traffic to any domain, and note that their presence in a given SDNS provider's infrastructure indicates the provider's failure to properly check the authorization of its proxying requests.

As shown in Table 4, we find that the identified proxies operated by CactusVPN, HideIPVPN, ibVPN, SmartyDNS, and VPNUK are all universal. All open proxies are also universal proxies, although the reverse does not hold for ibVPN and VPNUK. Proxies that are open and universal (CactusVPN, HideIPVPN, and SmartyDNS) allow any Internet user to proxy HTTPS traffic to any site, without hav-

ing to register (or pay) for the SDNS service.

9 Information Leakage through DNS Probing

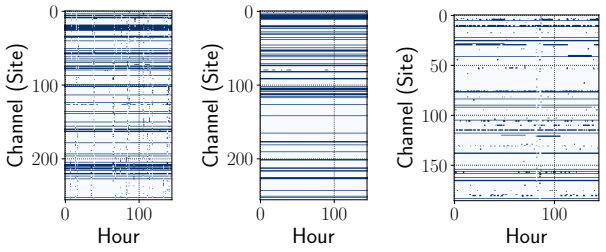
Threat model(s): SDNS Provider

Most SDNS providers advertise a number of channels (i.e., web sites) for which they will proxy traffic. In this section, we describe how DNS cache probing techniques can be used to infer both (i) the relative popularity of channels among a service's customers and (ii) the number of users of an SDNS service. Channel popularity allows us to gauge which sites' geofences are most often bypassed, while the number of users of an SDNS service enables us to estimate the revenue and general profitability of the service. SDNS services do not publish statistics that describe which channels are actually accessed by their customers, nor do they provide the relative popularity of the channels that are accessed. The DNS probing techniques described in this section provide a first glimpse as to how SDNS customers use these services.

9.1 Inferring Channel Requests

To identify the channels requested by SDNS customers, we use the DNS cache snooping technique introduced by Grangeia [29, 5] to determine whether or not the zone record for a hostname is in the cache of a resolver.

By default, clients almost always set the Recursion Desired (RD) bit when sending queries to DNS servers. This is true of many major operating systems (including OSX, Windows, and Linux) and is intended to allow for the possibility of recursive DNS resolution. In brief, the RD bit indicates to the resolver that the client prefers that the resolver perform a recursive DNS lookup. Grangeia's cache snooping technique leverages the behavior that when the RD bit is *not* set, DNS servers (i) respond with the resolved IP address if the entry is in its cache and (ii) return either an error or the root name servers for the requested domain if it is not. (We note that returning the root name servers is the expected behavior for iterative DNS resolutions.) By setting the RD bit to zero, we definitely learn whether the requested hostname is in the resolver's cache.



(a) CactusVPN (b) SmartyDNS (c) Unlocator

Figure 3: Presence (blue) and absence (white) of advertised channels' domain names in SDNS providers' DNS resolver caches.

Figure 3 shows the presence and absence of the hostnames for advertised channels on three SDNS providers over an approximately 5.25 day period beginning on August 21, 2019. We probed each SDNS provider's cache once per hour during this period. (We performed the experiment for other SDNS providers and obtained similar results; they are omitted for brevity.) Specifically, we examined the cache of the first DNS resolver that was listed on the webpages of the three tested SDNS providers. We observe that (i) most of the domain names associated with the advertised channels never appeared in the resolvers' caches and (ii) the few sites that did appear, did so consistently. For example, less than 24% (61 out of 256) of the sites supported by the SmartyDNS proxy ever appeared in its cache; TrickByte had the highest cache saturation with 61% (50 out of 82) of its supported channels appearing at least once in its cache during our measurement period. In summary, while SDNS providers advertise support for a large number of channels, our findings suggest that customers regularly use only a modest fraction of those offerings.

9.2 Deriving Channel Popularity

Based on our prior analysis, we sought to understand the relative popularity of channels provided by SDNS. We estimate how often an SDNS provider's customers request each of the provider's supported channels by assuming the rate at which the SDNS provider resolves requests for a particular hostname is indicative of frequency at which its customers request it.

To perform this analysis, we use the popularity inference technique of Rajab et al. [45], which operates as follows: a request for resolving hostname H is sent to a DNS resolver D. In its reply, D returns the time (TTL_l) at which the entry for H will be expunged from its cache. The maximum possible TTL ($\text{TTL}_{\max}$) can be obtained by querying the authoritative name server for H. Rajab et al.'s technique issues a probe for H once per $\text{TTL}_{\max}$, which allows for computing the *refresh* time (the time at which the cache entry for H was most recently refreshed) as $T_r = T_p - (\text{TTL}_{\max} - T_l)$ where T_p is the time of the probe. This allows for the computation of the average rate λ at which H is requested from D as

$$\lambda \approx \frac{R}{\sum_{i=1}^{R}(T_{r_i} - T_{r_{i-1}} - TTL)}$$

where R is the number of probes for H [45].

We implemented Rajab et al.'s algorithm and deployed it on 11 resolvers belonging to 11 different SDNS providers (i.e., we used one resolver per SDNS provider). Three of the 11 resolvers reported erratic or otherwise erroneous TTLs, and we excluded these resolvers from our analysis. For the remaining eight SDNS services, the average aggregate request rates, measured in requests per hour, are listed in Table 5. We report the 10 most frequently requested hostnames for each SDNS service. We additionally compute 95% confidence intervals of λ by applying the central limit theorem [45]; this allows us to gauge our confidence in the results based, in part, on the number of probes we performed. (Hostnames that have a large $\text{TTL}_{\max}$ value resulted in fewer probes.)

Our findings reveal that streaming video—the target offering for many of the SDNS providers—is by far the most common destination for SDNS customers. Interestingly, SDNS providers commonly resolve queries for popular news (cnn.com) and social media (instagram.com) sites. This suggests that SDNS customers regularly use SDNS resolvers and do not reserve their use for accessing geofenced content.

In Appendix C, we use similar techniques to estimate the number of customers and revenue for several SDNS providers.

Limitations to Popularity Measures. To minimize the volume of our requests, we target only one DNS resolver for each SDNS provider. Probing the unexamined DNS resolvers may yield different results. Additionally, as shown in Table 5, the confidence intervals can be large, sometimes overwhelming a site's estimated arrival rate (λ). In such cases, these results should be viewed with some skepticism. Finally, our results assume DNS servers correctly follow the DNS protocol as described in the RFC [42]. Although, by definition, SDNS resolvers do not always return correct DNS responses, we observe that the tested SDNS resolvers appear to generally adhere to the DNS RFC, with the sole exception of returning false IP addresses for proxied channels.

10 Discussion

Some of the attacks identified in this paper are inherent to the design of SDNS systems, and are difficult to remedy. In particular, the real-time SDNS user identification attack (§6.2)—which can also identify SDNS proxy servers—is effective because SDNS can *only* proxy traffic that first requires a DNS resolution. Fundamentally, the attack exploits the defining characteristic of SDNS services (i.e., the assignment of proxies via DNS resolution), and thus we believe it is unlikely that an SDNS provider can counter this inherent weakness. On the other hand, the client enumeration attack (§6.1) exploits an implementation flaw in some SDNS systems, and can be remedied; in fact, one provider implemented a fix after we disclosed the attack.

The increased risk of traffic analysis (§7) is similar to the risk that arises from using VPN services: longer Internet paths generally provide more opportunities for eavesdropping. However, unlike with VPNs, SDNS has no easy on/off switch, and we suspect that many SDNS users configure their computers to use SDNS resolvers and do not restore their settings after using SDNS. This "set and forget" functionality is unusual for VPNs, which typically require user interaction to enable. We conjecture that SDNS use therefore provides a more persistent level of susceptibility to eavesdropping than VPNs due to the likely longevity of using SDNS resolvers.

Encrypting DNS traffic between a client and the SDNS resolver using either DoH [31] or DoT [33] mit-

igates some of the eavesdropping risks. However, encrypted DNS is still subject to traffic analysis which can leak the sites being resolved [7, 48, 32] to on-path eavesdroppers. Perhaps more importantly, not all major operating systems support DoH or DoT. We believe at least for the short-term that it is unlikely that SDNS providers would add support for DoH/DoT, since doing so may increase the level of technical sophistication required to configure SDNS.

The authentication and authorization failures (§8) leverage poor design decisions by the vulnerable SDNS services. Applying IP-based authentication at both the resolver and the proxy prevent unauthorized use. (It is worth emphasizing that IP-based authentication does not provide strong authentication.)

Finally, the exposure to analytics (§9) uses DNS cache probing techniques that are generally applicable to DNS resolvers. They are arguably especially problematic however in the SDNS setting since the sole purpose of SDNS services is to bypass geofencing, and hence determining how often these services are used for each channel could be useful to assess the potential criminal culpability or legal liability of the providers. To prevent such information leakage, DNS resolvers (whether SDNS resolvers or ordinary resolvers) could advertise the maximum TTL value, although such behavior would clearly violate the DNS specification [42].

11 Ethical Considerations

At all times, we sought to minimize risk, both to the users of SDNS services and the services themselves. Our experiments were guided by the principles outlined in the Menlo Report [13]. We use this guideline to elaborate on the ethics of our study:

Respect for persons. We avoided causing harm to individual users through our measurements and proof-of-concept attacks by not targeting specific individuals (other than ourselves). In validating the client enumeration attack in §6.1, we used a small-scale proof-of-concept in which we spoofed only our own IP addresses. We did not attempt to discern whether any IP addresses, other than those operated by the authors, belonged to customers of the vulnerable SDNS services. We did not issue more than a handful of queries to the SDNS provider's resolver, and our DNS queries were all well-formed and conformed to the DNS standard [42].

Similarly, in §9, when identifying SDNS usage rates and the popularity of channels, our measurements consisted of sending a relatively low volume (one request per hostname per the hostname's TTL_{max}) of well-formed DNS queries to a DNS resolver. This volume of requests is negligible compared to the request arrival rate of SDNS providers (see Table 5). This measurement provides respect for individuals and persons in that it does not directly interfere with the normal activities of the SDNS service nor its customers. In brief, we used SDNS resolvers exactly as they were intended to function, and merely inspected the returned result (specifically, its TTL value) to derive statistical inferences.

Beneficence. At all stages of our study, we sought to reduce harm and maximize the benefits of the research. Foremost, we designed our experiments to avoid overwhelming the public DNS or SDNS resolvers by rate limiting our requests, and we only submitted well formed DNS, HTTP, and HTTPS requests throughout.

Most relevant, after identifying the client enumeration attack, we performed responsible disclosure and notified the operators of the two affected services (VPNUK and ibVPN).We received a response from ibVPN the following day, informing us that they had (partially) mitigated the attack (see §6.1). The goal of disclosing the attack was to decrease the associated risks identified and increase the overall benefits of the research.

Justice. Our measurements are just in that we spread out all probes to a broad set of SDNS providers, treating each equally in our experimentation without targeting specific services over others.

Respect for Law and Public Interest. We designed our study to be both lawful and in the public interest. First, we paid or used free-trials for all of the measured SDNS providers; we did not use these services surreptitiously. Additionally, we considered and reported on the privacy and security ramifications of using these services, which is in the public interest, and when we identified potential vulnera-

bilities, we responsibly disclosed them to the SDNS providers. In taking this action, we consulted with our institution's general counsel to ensure that we took action in a legally responsible manner (based on our jurisdiction).

12 Conclusion

This paper presents the first study of smart DNS (SDNS) services in the wild. We identify a number of architectural weaknesses in currently deployed SDNS services that affect the privacy and security of end users, who may be confused with the privacy properties of SDNS as compared to other services that can avoid geographic restrictions, such as VPNs. We show that content providers can trivially detect SDNS users who access their sites, identify these users' unobscured IP addresses, and enumerate *all* customers who are registered for these SDNS services (whether they access the content provider's website or not). Worse, some settings of SDNS allow *any* arbitrary third party to enumerate all customers of an SDNS service, even when those users are offline. These vulnerabilities were confirmed and repaired by an SDNS provider after responsible disclosure.

We also identify a number of authentication and authorization errors in the setup of SDNS, with many services relying only on IP-based authentication at the DNS server and neglecting to perform these checks at the proxy server. The failure to properly authenticate and authorize users effectively transforms SDNS providers' proxies into a distributed network of open SNI proxy servers. We describe a straightforward method of discovering these open proxies and discuss how unscrupulous users could use SDNS services while bypassing payment. Additionally, we show that some SDNS providers proxy content that is not advertised as being supported, raising further privacy concerns about traffic interception and manipulation.

Although SDNS provides customers with the ability to seamlessly bypass geofenced content, this comes at the expense of a large number of privacy and security risks, including the exposure of SDNS service usage and individual browsing habits.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful feedback, and especially to our shepherd, Tariq Elahi, for the many suggestions that helped us improve this paper. We also would like to thank Tavish Vaidya for the many fruitful conversations about SDNS services.

This paper is partially supported by the National Science Foundation under grants #1718498 and #1845300. The opinions, findings, and conclusions or recommendations expressed in this paper are strictly those of the authors and do not necessarily reflect the official policy or position of any employer or funding agency.

References

[1] All Checkout Plan Links. `https://www.vpnsecure.me/other-products/all-products/`, Accessed 13, September 2020.

[2] Sadia Afroz, Michael Carl Tschantz, Shaarif Sajid, Shoaib Asif Qazi, Mobin Javed, and Vern Paxson. Exploring Server-side Blocking of Regions. *arXiv preprint arXiv:1805.11606*, 2018.

[3] Amazon. Alexa Top 1 Million. `https://s3.amazonaws.com/alexa-static/top-1m.csv.zip`, (Accessed on 09/27/2018).

[4] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546, Internet Engineering Task Force, 2003.

[5] S. Bortzmeyer. DNS Privacy Considerations. RFC 7626, Internet Engineering Task Force, 2015.

[6] Sam Burnett and Nick Feamster. Encore: Lightweight Measurement of Web Censorship with Cross-origin Requests. *ACM SIGCOMM Computer Communication Review*, 45(4):653–667, 2015.

[7] Jonas Bushart and Christian Rossow. Padding Ain't Enough: Assessing the Privacy Guarantees

of Encrypted DNS. In *Workshop on Free and Open Communications on the Internet (FOCI)*, 2020.

[8] Thomas Callahan, Mark Allman, and Michael Rabinovich. On Modern DNS Behavior and Properties. *ACM SIGCOMM Computer Communication Review*, 43(3):7–15, 2013.

[9] Taejoong Chung, Roland van Rijswijk-Deij, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In *USENIX Security Symposium (USENIX)*, 2017.

[10] California Code. California Consumer Privacy Act of 2018 (1.81.5 CA Civ Code §1798.145).

[11] Council of European Union. EU General Data Protection Regulation (GDPR):Regulation (EU) 2016/679 of the European Parliament and of the Council of 27, April 2016.

[12] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, August 2004.

[13] David Dittrich, Erin Kenneally, et al. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. Technical report, US Department of Homeland Security, August 2012.

[14] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. ZMap: Fast Internet-wide Scanning and its Security Applications. In *USENIX Security Symposium (USENIX)*, 2013.

[15] Experian. How well do you know customers - really? `https://www.experian.com/marketing-services/identity-resolution`, Accessed on 12, September 2020.

[16] ExpressVPN. What Is VPN? `https://www.expressvpn.com/what-is-vpn`, (Accessed on 05/08/2019).

[17] David Fifield. meek. `https://trac.torproject.org/projects/tor/wiki/doc/meek`.

[18] David Fifield, Jiang Jian, and Paul Pearce. SNI Proxies. `https://www.bamsoftware.com/computers/sniproxy/`, 2016.

[19] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant Communication through Domain Fronting. In *Privacy Enhancing Technologies Symposium (PETS)*, 2015.

[20] Arturo Filasto and Jacob Appelbaum. OONI: Open Observatory of Network Interference. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[21] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.

[22] Amy Friedlander, Allison Mankin, W. Douglas Maughan, and Stephen D. Crocker. DNSSEC: A Protocol toward Securing the Internet Infrastructure. *Communications of the ACM*, 50(6):44–50, June 2007.

[23] Sergey Frolov and Eric Wustrow. The Use of TLS in Censorship Circumvention. In *Network and Distributed System Security Symposium (NDSS)*, 2019.

[24] Henry Gemmer. 10 Best Unlimited Bandwidth VPS Providers. `https://uncensoredhosting.com/unlimited-bandwidth-vps/`, 2019.

[25] Alessandro Ghedini. Encrypt It or Lose It: How Encrypted SNI Works (Blog Post). `https://blog.cloudflare.com/encrypted-sni/`, (Accessed on 05/09/2019).

[26] GlobalSign. What is an OrganizationSSL Certificate? `https://www.globalsign.com/en/ssl/organization-ssl/`, (Accessed on 09/07/2019).

[27] Google. HTTPS Encryption on the Web. `https://transparencyreport.google.com/https/overview?hl=en`, 2020.

[28] Google Public DNS. Frequently Asked Questions. `https://developers.google.com/speed/public-dns/faq#anycast`, 2020.

[29] Luis Grangeia. DNS Cache Snooping. Technical report, Securi Team—Beyond Security, 2004.

[30] Benjamin Greschbach, Tobias Pulls, Laura M Roberts, Philipp Winter, and Nick Feamster. The Effect of DNS on Tor's Anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2017.

[31] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, Internet Engineering Task Force, 2018.

[32] Rebekah Houser, Zhou Li, Chase Cotton, and Haining Wang. An Investigation on Information Leakage of DNS over TLS. In *International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*, 2019.

[33] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, Internet Engineering Task Force, 2016.

[34] ipinfo.io. Comcast Cable Communications, LLC (AS7922). `https://ipinfo.io/AS7922`, Accessed on 9/11/2019.

[35] IronSocket. Access Your Favorite Social Network Sites. `https://ironsocket.com/`, Accessed on 12, September 2020.

[36] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Transactions on Networking*, 10(5):589–603, 2002.

[37] Mohammad Taha Khan, Joe DeBlasio, Chris Kanich, Geoffrey M. Voelker, Alex C. Snoeren, and Narseo Vallina-Rodriguez. An Empirical Analysis of the Commercial VPN Ecosystem. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2018.

[38] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg. SoK: Making Sense of Censorship Resistance Systems. *Proceedings on Privacy Enhancing Technologies (PoPETS)*, 2016(4):37–61, July 2016.

[39] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Damon McCoy, Vern Paxson, and Steven J Murdoch. Do You See What I See? Differential Treatment of Anonymous Users. In *Network and Distributed System Security Symposium (NDSS)*, 2016.

[40] Akshaya Mani, Tavish Vaidya, David Dworken, and Micah Sherr. An Extensive Evaluation of the Internet's Open Proxies. In *Annual Computer Security Applications Conference (ACSAC)*, December 2018.

[41] Zhuoqing Morley Mao, Charles D Cranor, Fred Douglis, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *USENIX Annual Technical Conference (USENIX-ATC)*, 2002.

[42] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, Internet Engineering Task Force, 1987.

[43] NordVPN. Advantages & Benefits of VPN (Virtual Private Network). `https://nordvpn.com/features/`, (Accessed on 05/08/2019).

[44] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, April 2011.

[45] Moheeb Abu Rajab, Fabian Monrose, and Niels Provos. Peeking Through the Cloud: Client Density Estimation via DNS Cache Probing. *ACM Transactions on Internet Technology (TOIT)*, 10(3):9, 2010.

[46] Eric Rescorla, F. Oku, N. Sullivan, and C. Wood. Encrypted Server Name Indication for TLS 1.3. IETF draft tls-esni-03, Internet Engineering Task Force, 2019.

[47] RIPE Network Coordination Centre. RIPE Atlas. `https://atlas.ripe.net/`, 2020.

[48] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. Encrypted DNS ⇒ Privacy? A Traffic Analysis Perspective. 2020.

[49] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. Characterizing the Nature and Dynamics of Tor Exit Blocking. In *USENIX Security Symposium (USENIX)*, August 2017.

[50] Technology Analysis Branch of the Office of the Privacy Commissioner of Canada. What an IP Address Can Reveal About You. `https://www.priv.gc.ca/media/1767/ip_201305_e.pdf`, May 2013.

[51] ThatsThem. Find People for free using an IP Address. `https://thatsthem.com/reverse-ip-lookup`, Accessed on 12, September 2020.

[52] ThatsThem. Ptivacy Policy. `https://thatsthem.com/privacy-policy`, Accessed on 13, September 2020.

[53] Michael Carl Tschantz, Sadia Afroz, Anonymous, and Vern Paxson. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy (Oakland)*, 2016.

[54] Giorgos Tsirantonakis, Panagiotis Ilia, Sotiris Ioannidis, Elias Athanasopoulos, and Michalis Polychronakis. A Large-scale Analysis of Content Modification by Open HTTP Proxies. In *Network and Distributed System Security Symposium (NDSS)*, 2018.

[55] U.S. Code. Computer Fraud and Abuse Act (18 U.S. Code §1030).

[56] Benjamin VanderSloot, Allison McDonald, Will Scott, J. Alex Halderman, and Roya Ensafi. Scalable Remote Measurement of Application-Layer Censorship. In *USENIX Security Symposium (USENIX)*, 2018.

[57] VPNUK. Secure Virtual Private Networking with VPNUK. `https://www.vpnuk.net/`, Accessed on 12, September 2020.

[58] Guohui Wang, Bo Zhang, and T. S. Eugene Ng. Towards Network Triangle Inequality Violation Aware Distributed Systems. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[59] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In *ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2018.

[60] W. Wimer. Clarifications and Extensions for the Bootstrap Protocol. RFC 1542, Internet Engineering Task Force, 1993.

[61] Zhao Zhang, Tavish Vaidya, Kartik Subramanian, Wenchao Zhou, and Micah Sherr. Ephemeral Exit Bridges for Tor. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2020.

[62] Zhao Zhang, Wenchao Zhou, and Micah Sherr. Bypassing Tor Exit Blocking with Exit Bridge Onion Services. In *ACM Conference on Computer and Communications Security (CCS)*, November 2020.

[63] zzz (Pseudonym) and Lars Schimmer. Peer Profiling and Selection in the I2P Anonymous Network. In *PET-CON 2009.1*, March 2009.

A Additional Methodological Details and Ecological Findings

To bypass geofencing restrictions, SDNS services rely on a network of strategically placed proxies and DNS resolvers. As part of our analysis and to better understand these services' ecosystem, we attempted to answer the questions: *where are SDNS resolvers and proxies located?; who hosts them?; and what was the likely motivation behind these choices?*

As a first step, to better understand to whom the SDNS providers are catering their services, we determine the geographic location of the SDNS providers' DNS resolvers.

To reduce network latencies incurred during DNS resolutions, SDNS providers often recommend that their customers select a SDNS resolver that is in close physical proximity. Understanding where SDNS providers place their resolvers thus provides hints as to where they envision the best opportunities for attracting customers. Using MaxMind's geolocation service, we map the listed DNS resolvers for each proxy to a location, and report the countries with the most resolvers in Table A.1. With the exception of the United States, we note that the locations of potential customers mostly differ considerably from the locations in which the SDNS providers operate (see Table 1). In short, it appears that SDNS providers mostly tailor their services to international customers.

While the locations of the resolvers indicate potential customer markets, the proxies' locations correlate to the geofences that the SDNS providers aim to bypass. However, due to the nature of the modern web and the prevalence of content distribution networks (CDNs), identifying SDNS providers' proxy servers proved complicated.

At a high level, the task entails querying an SDNS resolver for a hostname and identifying whether the returned IP address was (i) accurate or (ii) that of a proxy server. In reality, unfortunately, DNS resolution is fairly complex. Rather than consistently returning a single IP, multiple queries to a single domain name on a normal (non-SDNS) resolver return the IP of the host that can serve the website's content fastest, given the current network state and the requester's network location. When accounting for the widespread use of CDNs, this also means seemingly unrelated sites can be resolved to the same IP address (since multiple sites can share CDN replicas). In short, it is difficult to enumerate all possible valid IP addresses that belong to a given hostname; this is especially true of popular sites (including the channels supported by SDNS providers), since such sites tend to heavily rely on CDN services.

We identify proxy IPs using a two-phase approach: at a high level, we first identify a set of *candidate IPs* we believe may be SDNS proxies, and then verify them. To generate candidate IPs, we queried 10 SDNS providers' resolvers for two sets of domains: the Alexa top 1,000 most popular sites [3], and the hostnames of the channels advertised as being supported by the SDNS provider. We then compared the returned list of IPs against a *ground truth* dataset generated by making over 32,000 DNS requests to Google's and CloudFlare's DNS resolvers, as well as requests from RIPE Atlas probes to their local resolvers. The latter was included to increase the geographic and network diversity of the requesting DNS clients. The ground truth dataset was constructed using DNS queries conducted between February 14 and March 31, 2019, and again between April 25 and May 3, 2019. Finally, we generated a candidate list of hostname-to-resolved-IP pairings by first considering the responses from SDNS resolvers and then eliminating entries for which an IP in the same /24 appeared in the ground truth dataset.

To verify the candidate IPs as proxies, we attempted to fetch content via a candidate proxy from both a machine that was registered with the SDNS service and one that was not. Conceptually, if the candidate IP is not a proxy and is a legitimate IP address that serves content for the site, it should serve the content regardless of the requestor's IP; on the other hand, if it *is* a proxy, then the proxy should only serve content for the IP that is associated with one of its customers. That is, we expect actual proxies to serve requests that originate from a registered IP, but to deny the same content requests from IPs that are not associated with the SDNS provider.

Using two machines, one whose IP we registered with the SDNS service, and one whose IP was not registered, we sent well-formed HTTP/S requests (with the HOST HTTP and SNI headers properly set) to the candidate proxy using `curl`, and compared the results. We confirm a candidate IP as an actual proxy if and only if the HTTP/S request from the registered machine was successful (i.e., resulted in a 200 OK HTTP response) and the request from the non-registered machine was not.

Overall, we were able to definitively identify 54 distinct proxy IPs across five of the evaluated SDNS providers.

Table A.3 lists the most common countries where proxy servers are located. We note that the most popular locations—the United States, the United Kingdom, and India—are also the nations that host a large fraction of the channels offered by SDNS providers. This suggests that proxies are indeed placed close to content providers.

B Mapping an IP Address to an Individual

Although mapping an IP address to an autonomous system (AS) is straightforward, the process of matching an IP to an actual individual is more difficult, error-prone, and less well-understood. This is due to a number of different factors including (but not limited to) the widespread use of DHCP for IP address allocation, and ISPs' widely varying policies for managing pools of available IP addresses. The former is especially problematic, since DHCP does not set any explicit requirements for how long each IP address is allocated to a single entity (e.g., a household) [60]. Nor does DHCP require a lessee to notify the DHCP server if it relinquishes an IP address before its lease expires [60].

In the context of the IP enumeration attack described in §6.1, for each IP address found to be registered with an SDNS provider, the attacker can determine the ISP, AS, and rough geographic location (with the caveat that IP geolocation services are not always accurate) from which the IP address originates.

However, the attacker can use additional datasources to sometimes hone in on the household or even individual who leases a particular IP address. Many companies collect publicly available records and mine information from sources such as social media, web beacons, browsing histories, and user cookies placed across the Internet to create "people databases"—vast databases that contain detailed profiles of Internet users. These profiles often include an Internet user's full name, address, gender, age, phone number(s), email(s), date of the profile's last update, and—most relevant to our attacks—the user's IP address [51].

Using these people databases as a primary backend, companies such as ThatsThem [51] offer search tools that allow anyone to map an IP address to an individual or household. We note that these search tools do not have complete data, and it is difficult to assess their accuracy. However, traditional and far more well-established data brokers, such as Experian, also offer IP-to-individual mapping services [15].

It is worth noting that, in addition to their public facing offerings, consumer data collection companies frequently buy and sell collected information from/to each other, meaning that once data about an individual is collected by one company, that information likely propagates to others [15, 52].

In summary, an adversary who learns an SDNS user's IP address could use these people databases to learn not only the identity of the SDNS customer, but also additional information (e.g., address and email). For this reason IP addresses are sometimes considered personally identifying information under both GDPR and the California Consumer Privacy Act of 2018 [11, 10], and that the Privacy Commissioner of Canada issued a report detailing the privacy risks of exposing IP addresses [50].

C Estimating the Number of Customers and Revenue

Again borrowing from the technique of Rajab et al. [45], we can use the average request rate (λ) to

form a rough approximation of the number of users (n) of an SDNS provider. We denote λ(site) as the value of the aggregated (total) average request rate for a given site. Further, let λ_c(site) be the expected request rate for a *client* accessing the site. That is, while λ(site) denotes the total requests per unit time for the site, λ_c(site) is the number of requests due to a given user. Then, assuming Gamma distributed arrival times, Rajab et al. shows that the number of users n of an SDNS provider is:

$$n = \frac{\lambda(\text{site})}{\lambda_c(\text{site})}$$

Rajab et al. reports that λ_c(google.com) is 2.63 requests per hour [45]. We can compute λ(google.com) for the various SDNS providers using the technique described in the previous section, and thus compute n. Here, we perform our probes (i.e., DNS requests for google.com) over an approximately 11 hour period beginning on September 6, 2019. To limit the load on the SDNS providers' resolvers, we send probes to a single resolver per SDNS provider.

Table C.5 lists the empirically measured average request rate for google.com and the derived number of users for six SDNS services' resolvers. (The remaining providers did not consistently respond properly to DNS requests to resolve google.com, and are excluded from the Table.) Note that the number of estimated users in Table C.5 is based on traffic to a single resolver (per service) and thus likely undercounts the total number of users of a service.

Of the successfully tested providers, we find that CactusVPN has approximately 16K users using a single one of its resolvers, while the other SDNS services have significantly fewer users accessing their tested resolvers.

Using the pricing information presented in Table 1, we can then estimate the revenue for each SDNS provider by multiplying the estimated number of users by the price-per-user. This should be considered a conservative (low) estimate of the provider's revenue, since our probing DNS requests target only the first listed DNS resolver for each SDNS provider.

We can estimate profit margins for an SDNS provider based on the expected costs of running proxy servers. Proxies relay content to/from supported channels, and we consider a near-worst case scenario in which all SDNS users continuously stream high-quality video. Here, we use Netflix's reported bandwidth requirements of 3 GB/hour (6.67 Mbps) to estimate SDNS providers' bandwidth needs. SDNS providers can easily support such rates with VPS providers. In particular, there are a number of VPS providers that provide uncapped (sometimes called *unmetered*) 1 Gbps links [24] for approximately $ 10 per month. We note that a single 1 Gbps link can support 150 SDNS customers (each of whom consumes 6.67 Mbps). The revenue from 150 SDNS customers far exceeds the bandwidth costs. For example, CactusVPN charges $ 4.99 per customer, per month, for a revenue of $ 748.50 and profit of $ 738.50 (after subtracting the $ 10 VPS cost) per 150 customers. The profit per customer is thus $ 4.92 per month, yielding a profit margin as high as 98.6%. Using these general assumptions, we provide estimates of the profits of SDNS providers in Table C.5.

Limitations to Profit and Revenue Estimation. Our analysis relies on a number of assumptions, including the expected distributed arrival times and the accuracy of λ_c(google.com) reported by Rajab et al. in 2010. We note that the client enumeration attack presented in §6 constitutes a far more accurate method of determining the precise number of SDNS customers, although the attack only works for a subset of SDNS providers. (Due to obvious ethical concerns, we did not perform the enumeration attack described in §6 to measure SDNS usage.)

Additionally, as discussed above, a more complete revenue exploration of an SDNS service would include the infrastructure costs of resolvers, as well as the fact that not all proxies are fully utilized. Further, our analysis ignores the (potentially high) costs of customer support and maintaining an infrastructure for billing.

D Performance of SDNS Services

DNS resolution is a frequent operation not only for web browsing, but also for myriad other applications and system services. For SDNS users, DNS resolution requests are handled by the SDNS resolver (absent local caching, which is rare). Relative to using a local ISP-provided resolver, resolving hostnames through SDNS imposes significant performance costs: local DNS resolvers are typically located near the requesting client [41], producing low roundtrip times. In contrast, SDNS providers have scant offerings of DNS resolvers from which to choose, making it unlikely that the chosen resolver will be close to the client. Second, due to the prevalence of CDNs, the mapping from hostnames to IPs may be dependent upon the location of the client IP and the DNS server. SDNS resolvers serve diverse clients and hence its cached entries (and, consequently, its returned IPs) are less catered to any particular client's network location.

We empirically measure the cost of using SDNS resolvers by performing DNS lookups to the top 1000 Alexa sites [3] via (i) SDNS resolvers, (ii) Google's free DNS resolver at `8.8.8.8`, and (iii) our local institution's default resolver. We note that Google's public DNS service uses IP anycast to ensure that resolution requests are routed to the closest resolver [28]. For each resolver and Alexa hostname, we performed five DNS resolutions (without caching) from a client machine on our institution's wired network.

We find that, unsurprisingly, the local resolver has the lowest median response time, closely followed by Google's DNS server. Using local resolution or Google's highly replicated DNS server offers orders of magnitude better response times than any of the tested SDNS providers. For example, the HideIP resolver impose approximately a 2500% overhead in median response time relative to our local resolver.

We also consider how the use of SDNS services affects overall browsing experience, as measured using web page load times. Here, the effects of slow DNS resolutions could be compound by the number of web objects embedded on a webpage, since objects at difference domains (or subdomains) require DNS resolution.

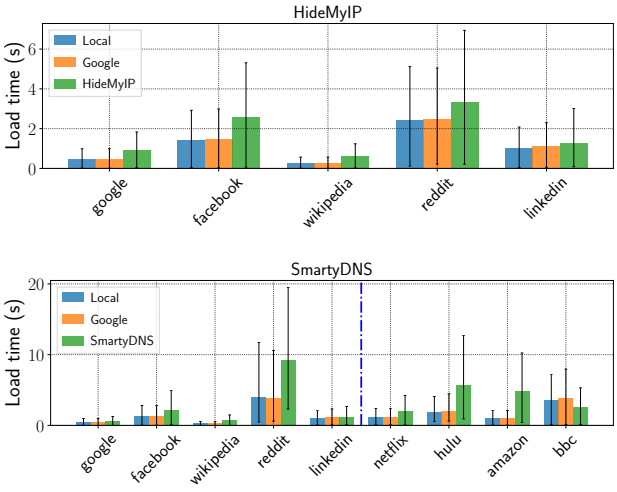


Figure D.1: Page load times for top pages of various sites, when using the local DNS server, Google's open DNS server at 8.8.8.8, and the SDNS DNS resolver to resolve hostnames. Error bars show the interquartile range. The vertical dashed blue bar for SmartyDNS and CactusVPN separate sites that are not advertised as being proxied (left of the line) and those for which the SDNS advertises support.

To examine the effects of SDNS usage on web page load times, we instrumented Selenium using the Chrome 76.0.3809.126 driver for Linux and configured it to use one of the three DNS configurations: our local institution's resolver, Google's open resolver, and the SDNS resolver. Note that for channels supported by the SDNS provider, using the provider's DNS resolver also meant that at least some web content will be relayed via one or more of the provider's proxy servers, since Chrome will fetch content from whatever IP addresses are resolved via DNS. We include Google's resolver to consider cases in which resolution is not performed locally, but the returned results are correct (i.e., the content is not proxied). In all cases, the headless Chrome browser fully renders the requested page.

Figure D.1 shows the median page load times for various websites, over 50 fetches performed on

September 7th, 2019. Error bars indicate the interquartile ranges. We focus on two particular SDNS providers for brevity; we obtained similar results for other providers (not shown). We include both sites that are proxied by the SDNS provider (right of the dashed horizontal bar) and those that are not proxied (left of the horizontal bar). The latter is included since we posit that most SDNS users will not regularly manually update their DNS settings and will instead rely consistently on their SDNS provider's resolver. As a result, the performance of any communication that depends on DNS resolution will be affected by the latencies incurred by using a remote resolver and potentially due to over-proxying (see §7.2).

The HideMyIP service, whose performance is shown at the top of Figure D.1, is a unique case because it proxies all network communication. Its DNS resolver always returns its own IP; HTTP/S traffic sent by the client is always proxied through this combined resolver/proxy.[5] For the tested destinations, HideMyIP imposes fairly substantial overhead. For example, it increases the median load times by 79% and 76% for rendering Facebook's top page, when compared against retrieval when local DNS and Google DNS resolutions are performed, respectively.

The increases in webpage rendering time were also pronounced for the SmartyDNS service, shown in the bottom plot of Figure D.1. For the non-proxied sites (that appear to the left of the dashed blue line), the differences in performance when using local versus Google resolution were minor. However, SmartyDNS incurs fairly significant overheads, especially in the case of reddit. We note that the top page of reddit requires the retrieval of more than 180 embedded web objects. The overhead of using SmartyDNS to resolve these requests is significant, resulting in a 128% increase in load page time over using the local resolver. Using SmartyDNS also results in longer page load times for sites that are advertised as being supported (i.e., proxied) by SmartyDNS. As an interesting outlier, retrieving the webpage bbc.co.uk was *faster* when using SmartyDNS. We suspect that this may be due to either a triangle-inequality violation in the Internet topology [58] in which routing through the proxy yields a higher performing connection than the default route, or (perhaps more likely) cacheing that is performed on the proxy node.

[5] We note that configuring a computer to use HideMyIP (i.e., by changing its network settings) prevents the use of non-HTTP-based protocols. HideMyIP is able to proxy all HTTP/S traffic by inspecting the HOST HTTP header or the TLS SNI header (see §4). However, repeating the requested hostname in the application-layer payload is generally a rarity in network protocols. The HideMyIP proxy is unable to determine the actual requested destination when other protocols that do not explicitly include the domain name are used (e.g., ssh), and therefore cannot proxy these protocols. Since the DNS resolver is a machine-wide setting, the use of HideMyIP significantly restricts the types of communications the computer can use.

Table 1: Identified SDNS providers. As indicated by their names, many double as purveyors of commercial VPN access.

Provider	Monthly Cost	Location
AceVPN	$5.95 °	USA
Blockless	$3.32 °	Canada
⋆BulletVPN	$10.98°	Estonia
⋆CactusVPN	$4.99	Moldova
⋆DNSFlex	$5.00	Canada
⋆DNSTrick	$4.95	Unknown
⋆GetFlix	$39.00•	Turkey
⋆HideIPVPN	$4.95	Unknown
Hide-my-IP	$4.95	USA
⋆ibVPN	$10.95	Romania
Ironsocket	$4.16	Hong Kong
⋆Keenow	$5.79	Israel
Le-VPN	$9.95°	Hong Kong
⋆Overplay	$4.99	USA
simpletelly	$4.99	Turkey
⋆SmartDNSproxy	$4.90	Turkey
⋆SmartyDNS	$4.90	Moldova
StrongDNS	$5.00	USA
⋆TrickByte	$2.99	Turkey
TVWhenAway	£7.99	UK
⋆Uflix	$4.90°	Turkey
Unblock-us	$4.99	Cyprus
⋆ Unlocator	$4.95	Denmark
VPNSecure	$9.95	Australia
⋆VPNUK	£5.99	UK◇

⋆ indicates a provider included in our measurement analysis
• indicates a lifetime cost
° indicates a cost that also includes VPN services
◇ contact info in UK, but company registered in Belize

Table 2: Summary of attacks. The adversary, required adversary capabilities, and the target of the attack are listed for each attack.

Vulnerability	Adversary	Required Adversary Cap.	Target
§6.1: Enumerating customers (by IP)	Internet user	reg. domain name; spoof UDP	Customer Threat
§6.2: Real-time SDNS customer identification	Content provider	operate website; view web logs	Customer Threat
§6.2: Real-time proxy server discovery	Content provider	operate website; view web logs	SDNS Provider Threat
§7: Increased risk of traffic analysis	Network eaves.	observe DNS or proxy traffic	Customer Threat
§8: Payment bypassing / free use of pay service	Internet user	send DNS resolution requests	SDNS Provider Threat
§9: Exposure to analytics / business analysis	Internet user	send DNS resolution requests	SDNS Provider Threat

Table 3: Average number of ASes encountered in network paths from various geographic regions to (1) Cloudflare's and Google's DNS resolvers ("Public Resolver") and (2) 108 SDNS resolvers ("SDNS Resolver"). Percentage increases (relative to the public resolvers) are shown in parentheses.

Client Location	Public Resolver	SDNS Resolver
Australia	1.50	2.74 (82.67%↑)
Belgium	1.00	2.60 (160.00%↑)
Brazil	2.00	2.31 (15.50%↑)
Japan	2.00	2.26 (13.00%↑)
United States	2.00	3.10 (55.00%↑)

Table 4: Occurence of open and universal proxies, by SDNS provider, for both HTTP and SNI proxy methods. Uflix, Trickbyte, and SmartDNSProxy shared 10 proxy servers; SmartDNSProxy and Trickbyte shared an additional seven proxies; and CactusVPN and SmartyDNS used the same five proxies. Moreover, among the SDNS providers studied, we observed that, for each protocol supported, an SDNS provider's proxies either were all open/universal, or none of them were.

Provider	Confirmed Proxies	Open (HOST/HTTP)	Universal (Host/HTTP)	Open (SNI/HTTPS)	Universal (SNI/HTTPS)
CactusVPN	5	○	●	●	●
HideIPVPN	3	○	●	●	●
IBVPN	1	○	●	○	●
SmartDNSProxy	36	○	○	○	○
SmartyDNS	5	○	●	●	●
Trickbyte	18	○	○	○	○
Uflix	14	○	○	○	○
VPNUK	17	○	●	○	●

○ none of the service's tested proxies operated in this mode

● all of the service's tested proxies operated in this mode

Table 5: Derived most popular channels and average estimated resolution rate (λ), in requests per hour.

Site	λ (95% CI)
tvland.com	47526 ±65
player.pl	47462 ±131
hbogo.com	47413 ±76
pandora.com	47339 ±93
comedycentral.com	47339 ±158
sonycrackle.com	39475 ±2385
theloop.ca	39099 ±4992
absoluteradio.co.uk	38714 ±217K
amazon.co.uk	36712 ±2224
bleacherreport.com	30703 ±6217

CactusVPN

Site	λ (95% CI)
amazon.com	95694 ±401509
youtube.com	79822 ±79257
foxsportsgo.com	70913 ±5113
bloomberg.com	46726 ±5174
disneylife.com	46498 ±673
funimation.com	46342 ±247
theloop.ca	46122 ±278
travelchannel.com	46036 ±874
amctv.com	45987 ±44
viaplay.se	45948 ±39

SmartDNSProxy

Site	λ (95% CI)
foxsports.com.au	1.7M ±2.7M
zdf.de	395633 ±2482
m6replay.fr	393483 ±2741
tsn.ca	91665 ±1662
rds.ca	88007 ±1163
tennischannel...	72537 ±2066
hbonow.com	66332 ±14379
itv.co	59623 ±15541
itv.com	44405 ±872
foxsoccer2go.com	34031 ±209

IB VPN

Site	λ (95% CI)
amazon.co.uk	754402 ±108K
oxygen.com	497202 ±345K
magine.com	341406 ±468
songza.com	324251 ±4487
vtele.ca	318096 ±4708
abema.tv	285243 ±3549
cbc.ca	285113 ±307K
telemundo.com	271029 ±211K
rte.ie	268518 ±16790
tennischannel.com	261632 ±15117

IronSocket

Site	λ (95% CI)
starzplay.com	90721 ±43758
cartoonnetwork.com	88345 ±216417
ahctv.com	87648 ±91137
cwtv.com	79143 ±74440
cwseed.com	78446 ±51686
indieflix.com	71231 ±219415
theonion.com	35655 ±11610
teamcoco.com	35201 ±137079
tlc.com	35200 ±1.52M
showtime.com	35196 ±118214

Keenow

Site	λ (95% CI)
instagram.com	2708804 ±97K
epix.com	116632 ±388
vh1.com	116564 ±7429
discovery.com	113888 ±21888
cnbc.com	113722 ±9419
history.com	111340 ±2690
amc.com	109985 ±2764
aetv.com	108331 ±1187
beinsports.com	106153 ±1044
cnn.com	101036 ±187K

DNSTrick

Site	λ (95% CI)
player.pl	423944 ±829
hbogo.com	421225 ±683
pandora.com	413256 ±1182
tvland.com	406346 ±4193
comedycentral.com	404203 ±5509
amazon.co.uk	306602 ±209K
cnbc.com	252442 ±593K
sling.com	208366 ±34462
nickjr.com	202391 ±46491
foxsports.com	197627 ±20198

SmartyDNS

Site	λ (95% CI)
docclub.com	79823 ±67529
discovery.com	78740 ±48817
showcase.ca	74609 ±58135
tvplayer.com	66103 ±90796
zattoo.com	59129 ±54614
syfy.com	34851 ±31357
nbc.com	33614 ±6276
history.com	33509 ±2476
rdio.com	33331 ±3398
klowdtv.com	33275 ±67988

TrickByte

Table A.1: Top 10 Countries with the most SDNS resolvers

Country	Num. Resolvers
United States	9
United Kingdom	4
Canada	4
Australia	3
Germany	3
India	3
Netherlands	3
Denmark	2
South Africa	2
Singapore	2

Table A.2: Top ASes with the most SDNS resolvers

AS Name and Number	Num. Resolvers
Amazon (16509)	21
DigitalOcean (14061)	15
SoftLayer Technologies (36351)	10
Choopa LLC (20473)	5
Iomart (20860)	5
Linode LLC (63949)	3
OVH SAS (16276)	3
SiteHost New Zealand (45179)	3
ASERGO Scandinavia ApS (30736)	2
Datacamp Ltd (60068)	2

Table A.3: Top 10 Countries with the most proxies.

Country	Num. Proxies
United States	15
United Kingdom	13
India	6
Australia	3
Denmark	3
Sweden	2
France	2
Canada	2
Germany	2
Norway	1

Table A.4: Top ASes with the most proxies.

AS Name and Number	Num. Proxies
DigitalOcean (14061)	8
QuadraNet (8100)	6
Iomart (20860)	4
Level 3 Parent LLC (3356)	4
ASERGO Scandinavia ApS (30736)	3
OVH SAS (16276)	2
GleSYS AB (42708)	2
Compuweb (51905)	2
Melbikomas UAB (56630)	1

Table C.5: The estimated number of users of a single SDNS resolver for various SDNS providers, and the estimated monthly profit.

Service	**Rate (λ)**	**Est. Users (n)**	**Est. Profit**
CactusVPN	41,119	15,635	$ 76,977
DNStrick	1,794	682	$ 3,330
HideIP VPN	2,127	809	$ 3,952
SmartyDNS	6,389	2,429	$ 11,741
TrickByte	8,269	3,144	$ 9,190
Unlocator	3,565	1,356	$ 6,622