Home is Where the Hijacking is: Understanding DNS Interception by Residential Routers

Audrey Randall
UC San Diego
aurandal@eng.ucsd.edu

Enze Liu
UC San Diego
e7liu@eng.ucsd.edu

Ramakrishna Padmanabhan
CAIDA/UC San Diego
ramapad@caida.org

Gautam Akiwate
UC San Diego
gakiwate@cs.ucsd.edu

Geoffrey M. Voelker
UC San Diego
voelker@cs.ucsd.edu

Stefan Savage
UC San Diego
savage@cs.ucsd.edu

Aaron Schulman
UC San Diego
schulman@cs.ucsd.edu

ABSTRACT

DNS interception — when a user's DNS queries to a target resolver are intercepted en route and forwarded to a different resolver — is a phenomenon of concern to both researchers and Internet users because of its implications for security and privacy. While the prevalence of DNS interception has received some attention, less is known about *where* in the network interception takes place. We introduce methods to identify where DNS interception occurs and who the interceptors may be. We identify when interception is performed before the query exits the ISP, and even when it is performed by the Customer Premises Equipment (CPE) in the user's own home. We believe that these techniques are vital in the light of the ongoing debate concerning the value of privacy-enhancing DNS transport.

CCS CONCEPTS

• **Networks** → **Home networks**; **Network measurement**.

ACM Reference Format:
Audrey Randall, Enze Liu, Ramakrishna Padmanabhan, Gautam Akiwate, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. 2021. Home is Where the Hijacking is: Understanding DNS Interception by Residential Routers. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3487552.3487817

1 INTRODUCTION

In principle, devices are free to direct their DNS queries to the recursive resolver of their choosing. Indeed, it is this freedom that has enabled the growth of public resolvers such as those offered by Google, Cloudflare, and others. However, a key underlying assumption is that DNS queries are faithfully forwarded as they are addressed. Unfortunately, this is not always so.

DNS queries sent by a user's device can be *intercepted* en route to a target resolver and forwarded to an alternate resolver. Furthermore, this interception can be *transparent*, where the interceptor spoofs responses so they appear to have been sent by the intended resolver. Transparent interception is difficult to detect because the alternate resolver does *not* have to modify the response. Even if the reason for the interception is benign — such as to prevent malware from evading DNS filtering — the interception of requests and misrepresentation of responses raise serious ethical concerns [14, 45] and can also interfere with the correct operation of protocols such as DNSSEC [14, 31].

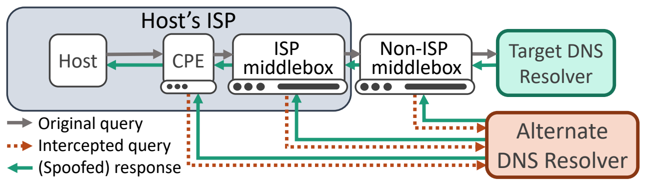

Figure 1: Locations where interception can occur.

While prior work has identified the broad prevalence of transparent interception [24, 31, 49], there are no established techniques for establishing *where* the interception is implemented. Indeed, there are a range of different points in the network where such interception might take place.

DNS redirection, another form of DNS manipulation, has also been found to occur in several parts of the network. DNS redirection occurs when a DNS resolver returns an altered response for specific queries and may occur with or without DNS interception. DNS redirection has been discovered *in Customer Premises Equipment (CPE)* to block resolution of specific domain names [17], *in ISPs* to replace NXDOMAIN responses with advertisements [30, 48] or enhance security and performance [44], and *outside of ISPs* to implement country-level censorship [4, 5, 16, 27]. Transparent interception has been far less extensively studied, although we are aware of anecdotal reports suggesting that, in some cases, the DNS forwarder in CPE [20] may be implicated [11, 13, 19].

In this paper, we develop a technique that can isolate *where* in the network transparent interception is occurring. In particular, we make the following contributions:

- **Identifying Where Transparent Interception Occurs**. We demonstrate how targeted use of standard DNS *debugging queries*—such as `id.server` and `version.bind` [51]—can identify not only the presence of transparent interception but to systematically infer if the source of that interception is the CPE

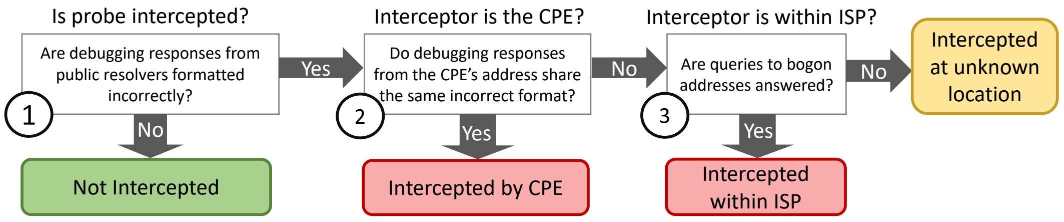
Is probe intercepted?
Are debugging responses from public resolvers formatted incorrectly?
1
Yes
No
Not Intercepted
Interceptor is the CPE?
Do debugging responses from the CPE's address share the same incorrect format?
2
No
Yes
Intercepted by CPE
Interceptor is within ISP?
Are queries to bogon addresses answered?
3
No
Yes
Intercepted within ISP
Intercepted at unknown location

Figure 2: Three-part technique to determine if and where a DNS query is being intercepted.

or middle-boxes in the ISP (Figure 1). Our technique can be implemented on any device that can make DNS queries, without requiring root access or external measurement tools such as an authoritative nameserver.

- **Pilot Study of Transparent Interception on RIPE Atlas**. We demonstrate our technique on RIPE Atlas infrastructure as a pilot study. We identify over 200 occurrences of such transparent interception. Of these, we further show that CPE-based transparent interception constitutes a significant fraction.
- **Case Study: CPE-based Transparent Interception**. We highlight a particular case that more precisely explains the nature of the DNS interception mechanism. Specifically, we illustrate how the common XB6/XB7 home router (used by many ISPs) use Destination Network Address Translation (DNAT) to transparently intercept queries and forward them to the ISP's resolver.

We believe that techniques such as these, that discover how and where DNS manipulation takes place, are especially important in light of the ongoing public controversy concerning the value and implementation of privacy-enhancing DNS transport [22, 23].

2 BACKGROUND AND TERMINOLOGY

During normal DNS resolution, a user device sends DNS request(s) to a *target resolver* (the user's ISP's resolver, a public resolver such as Google, Cloudflare, etc.). The target resolver recursively resolves the request and returns the response to the user device. Adopting terminology from Liu *et al.* [31], we refer to instances where DNS queries to a target resolver are intercepted and forwarded to an *alternate resolver* as **DNS interception**. DNS responses arriving at the user device from these alternate resolvers arrive with the source address spoofed to be that of the target resolver [31]; if not, the response would be rejected by the user device. When DNS responses are spoofed in this manner, the interception is *transparent* to the user device, and is difficult to detect. We refer to transparent interception as simply "interception" in the rest of the paper.

DNS redirection is a related but different form of DNS manipulation, where DNS responses — often NXDOMAIN responses [12, 30] — are altered, resulting in users getting redirected towards a different resource from the one in the unaltered response. It is often performed by the *target resolver*, rather than by forwarding a query to an alternate resolver. The insight that the response received by the user is *different* from the correct response during DNS redirection has been used to study this phenomenon in detail [30, 44, 48].

DNS interception, in contrast, has received less attention, and is the focus of our paper. In 2018, Liu *et al.* measured the prevalence of DNS interception [31], but did not measure *where* in the network their queries were intercepted. We present a technique that can both determine whether a DNS query is intercepted, as well as if it was intercepted by the client's own CPE or ISP.

3 METHODOLOGY

In this section, we present our methodology for detecting where DNS interception occurs. Figure 2 illustrates the three steps of the technique and the information used to make determinations at each step. We next describe each step in more detail, and present a concrete example of the technique in practice.

3.1 Identifying Query Interception

We show that it is sufficient to use a few select *location queries* — for which it is difficult to spoof the correct response — to detect query interception. Public anycast resolvers implement these queries to aid debugging, by revealing the location of the specific server that answers the query [18]. Our technique issues location queries to four public resolvers (on both primary and secondary IP addresses) and tests for "non-standard" responses to discover query interception. This technique is similar to the one used by Jones *et al.* to detect DNS root manipulation using `hostname.bind` queries [24]. Moreover, since each public resolver has both IPv4 and IPv6 addresses, we can detect interception in both protocols.

Each of the four public resolvers that we study implements its own version of a location query. Table 1 lists the queries and an example expected response. Each resolver uses a different format for its responses, and these formats are consistent around the world. We determined "standard" responses for each resolver by making requests from a network that we knew did not experience interception, and later confirmed that these responses were the expected ones in conversations with public resolver operators. When testing for interception, we compare responses to location queries issued from the device under study against these standard responses; when a response does not match the standard response, we conclude that the query has been intercepted.

We note that if a query is dropped entirely, it will appear to the client as a timeout. While timeouts can potentially reveal interesting behavior, such as censorship, for the purposes of our study we conservatively assume that timeouts are not due to transparent interception. We also note that prior work has observed query *replication*, where two responses are sent to the client: one from the intended recipient, and one from the interceptor's chosen resolver [31]. However, the interceptor's response nearly always

Public Resolver	Type	Location Query	Example Responses
Cloudflare DNS	CHAOS TXT	`id.server`	`IAD`
Google DNS	TXT	`o-o.myaddr.l.google.com`	`172.253.226.35`
Quad9	CHAOS TXT	`id.server`	`res100.iad.rrdns.pch.net`
OpenDNS	TXT	`debug.opendns.com`	`server m84.iad`

Table 1: Location queries and examples of expected responses from each resolver.

arrives first and is accepted by the client, so interception and replication are indistinguishable for our purposes.

3.2 Identifying Query Interception by the CPE

After we determine that interception is occurring using location queries, we then find where the query is first diverted away from its intended destination. We begin by using a novel technique to determine if the client's CPE is responsible for the interception.

First, we issue a `version.bind` query to the CPE's own public IP address. By usual IP routing rules, this query cannot travel beyond the CPE because the CPE is its destination. However, if the CPE is the interceptor, it will switch roles at this point: rather than acting as a packet forwarder following IP rules, the CPE will take on the role of a DNS forwarder instead. This role switch occurs because the most common method of implementing interception is DNAT. DNAT rewrites all query destinations to be the CPE's own private IP address, so that the CPE's DNS forwarder (e.g., Dnsmasq) can send them to its own pre-configured resolver. If the CPE's DNS forwarder supports the `version.bind` request, it will not forward the query any further, and will directly return a response.

However, this result alone is insufficient to demonstrate that the CPE is the interceptor, because there is another circumstance that could allow the CPE to forward the query: if the CPE's port 53 is open, it will act as a DNS forwarder even if it is not an interceptor. To distinguish between these cases, we next issue `version.bind` queries to each of the public resolvers we study. While only one resolver (Quad9) answers `version.bind`, it is immaterial — if the CPE is the interceptor, it will answer the query instead, and produce the same response as the query sent to the CPE's public IP address. We then compare the response strings from the query to the CPE with the responses from the queries to the public resolvers. If they are identical, we may conclude that the CPE is using DNAT to intercept queries to that resolver. (For more details on why we use `version.bind` for this technique, please see Appendix A.)

3.3 Query Interception by the ISP

If the interception is not being performed by the CPE, we next check whether it is occurring within the ISP. We can identify interception within the ISP by using another novel technique: making DNS requests to bogon IP addresses ("bogon queries"). Bogon IPs are unroutable, so bogon queries should not be able to leave the AS in which they originated. We chose one IPv4 and one IPv6 bogon address, confirmed that queries to these IPs were not routable, and directed queries for a generic domain we control to both IP addresses. If we received a response, we concluded that the request must have been intercepted before it could leave the AS. If we did not receive a response, two possibilities exist: either the interceptor was outside the AS, or the interceptor discards queries to unroutable addresses. Thus, if we received no response, we cannot determine where the interceptor was located. We found that most interception in most countries occurs before the query exits the AS (Figure 4).

ProbeID	Cloudflare DNS	Google DNS
1053	SFO	172.253.211.15
11992	*NOTIMP*	62.183.62.69
21823	routing.v2.pw	185.194.112.32

Table 2: Example responses to IPv4 location queries.

ProbeID	Cloudflare DNS	Google DNS	CPE Public IP
1053	-	-	-
11992	*NOTIMP*	*NOTIMP*	*NXDOMAIN*
21823	unbound 1.9.0	unbound 1.9.0	unbound 1.9.0

Table 3: Example responses to IPv4 `version.bind` queries.

3.4 Technique in Practice: Example

Tables 2 and 3 illustrate the technique using three RIPE Atlas probes and their responses. The first step tests if any of the probes are being intercepted. Table 2 shows the location queries to the IPv4 addresses of Cloudflare DNS and Google DNS. Probe 1053 receives an expected response, hence the queries are not intercepted: Cloudflare returns a three letter IATA airport code, and the Google responses map to Google IP addresses. On the other hand, probes 11992 and 21823 have non-standard responses, so their queries were intercepted.

Next, we issue `version.bind` queries for the two probes that were intercepted. Table 3 shows the responses. For probe 21823 the responses from Cloudflare DNS, Google DNS, and the CPE's public IP address are all the same, which indicates that the CPE is the interceptor (Section 3.2). For probe 11992 the responses are a mix of `NOTIMP` and `NXDOMAIN` responses, so the CPE was not the interceptor in this case.

Finally, we determine if probe 11992 was intercepted within the ISP by issuing a query to a bogon IP address. Because bogon IP addresses are not routable, the ISP should drop the queries. However, if the responses are valid and match the responses purportedly from the public resolvers in Table 3, then we can conclude that the interception happened within the ISP. If not, we cannot draw a conclusion about where the interceptor is located within the network. In the case of 11992, we received a NOTIMP response to the bogon query, and thus concluded that 11992's interceptor was within its ISP.

	Resolver IPv4		Resolver IPv6	
	Intercepted	**Total**	**Intercepted**	**Total**
Cloudflare DNS	165	9619	11	3730
Google DNS	160	9655	15	3726
Quad9	156	9616	11	3732
OpenDNS	156	9666	11	3727
All Intercepted	108	9537	0	3691

Table 4: Number of intercepted probes per public resolver.

4 PILOT STUDY ON RIPE ATLAS

We use the RIPE Atlas platform to perform a pilot study that confirms our technique works in the wild. With RIPE Atlas we can launch DNS measurements from roughly 10,000 probes around the world [35]. However, RIPE Atlas is not representative of the Internet as a whole: it has significantly more probes in Europe and North America than anywhere else, and also has a "geek bias" due to its volunteer-driven deployment. These biases should be taken into account before generalizing our findings on DNS interception to ISPs around the world. However, we emphasize that our technique is broadly transferable. With a handful of DNS queries, we can determine not only if queries are being intercepted, but also where the interception is occurring. We therefore believe our technique can be easily deployed on other measurement platforms as well.

4.1 Which probes experience interception?

As described in Section 3.1, the first step of our technique identifies which probes experience interception. We do so by sending *location queries* from every RIPE Atlas probe worldwide that would respond to our measurement requests. Over 9,600 probes responded to at least one experiment (Table 4). We identified 220 RIPE Atlas probes that experience the type of interception we were looking for, which we now break down by their location and behavior.

4.1.1 Which public resolvers were subject to interception? We test interception with four public resolvers: Google DNS, Cloudflare DNS, Quad9, and OpenDNS. Table 4 shows that the majority of intercepted nodes experienced interception for all four public resolvers. If fewer than four resolvers experienced interception, the most common pattern was either that only one resolver was intercepted, or only one resolver was *allowed*. In the former case, Google DNS and Cloudflare DNS were intercepted more often than Quad9 and OpenDNS, perhaps because of their popularity and market share. In the latter case, we hypothesize that the interceptor is deliberately using a single public resolver, perhaps for malware filtering purposes or ease of implementation.

We note that most interceptors that act on IPv4 queries for a public resolver *do not intercept queries for that resolver's IPv6 addresses*. Table 4 shows that only a handful of probes experience both IPv4 and IPv6 interception. Because IPv6 interception is infrequent, we consider IPv4 and IPv6 jointly for all subsequent analyses.

4.1.2 Is interception transparent? If an interceptor intends to be transparent, we assume it will correctly resolve most DNS queries. If it did not, it would be obvious to the client that something was wrong. To test this hypothesis, we sent a request for

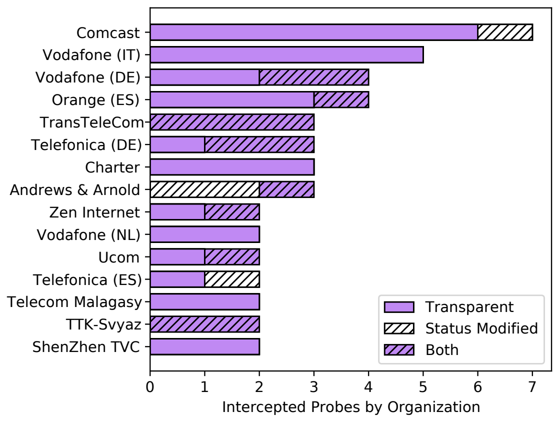

Figure 3: Intercepted probes per top 15 organizations.

`whoami.akamai.com` [29] to all four public resolvers from each intercepted probe. We do not expect this domain to be blocklisted.

The answers to this query let us confirm (a) that interception is indeed occurring (if the returned IP address is *not* one of the egress addresses of the target resolver) and (b) that the interception is transparent (if we do not receive an error in the response).

Figure 3 categorizes the responses. The "Transparent" bar indicates that queries to all intercepted resolvers were unchanged, "Status Modified" indicates that queries to all intercepted resolvers returned DNS error statuses, and "Both" indicates that requests to some resolvers were transparently intercepted while requests to others received modified status codes. The majority of queries across countries and ISPs return a valid response, which indicates that even intercepted queries *are* resolved correctly—just not by the targeted public resolver. However, some queries return a DNS error status for at least one resolver, such as `SERVFAIL` (server failure), `NOTIMP` (not implemented), or `REFUSED`. Because these status codes are not timeouts, they have likely been sent deliberately by the alternate resolver, and are not transient errors. We therefore conclude that some interceptors may block certain public resolvers.

Figure 3 shows that Comcast (AS7922) has the highest number of intercepted probes of any organization. Although RIPE Atlas has a high proportion of probes in the U.S., and in Comcast's AS in particular, this finding is consistent with prior work that showed significant DNS interception occurring in Comcast's networks [31]. Not all probes in Comcast's AS (or any AS) are intercepted: we found that this is because specific models of CPE perform the interception. For details in Comcast's case, see Section 5.

4.2 Is the interception performed by the CPE?

As we described in Section 3.2, once we identify intercepted probes, we use `version.bind` queries to determine whether the interceptor was the probe's CPE. Figure 4 shows the number of probes that were intercepted by their CPE per country and organization. To our surprise, a sizable fraction of the interception we observed was attributable to CPE: the CPE was the interceptor for 49 out of the

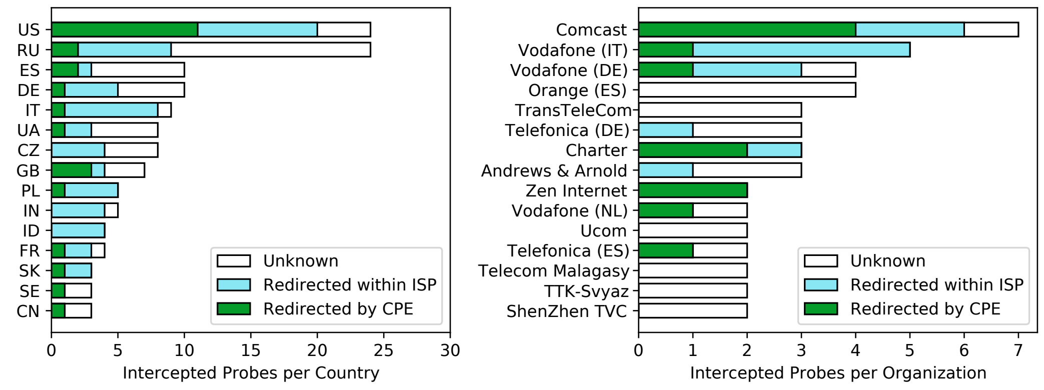

Figure 4: Interception location for the 15 countries and organizations with the most intercepted probes.

`version.bind` Response	# Probes
dnsmasq-*	23
dnsmasq-pi-hole-*	8
unbound*	6
*-RedHat	2
PowerDNS Recursor*, Q9-*, 9.16.15, *-Debian, Windows NS, Microsoft, new, unknown, none, huuh ? ... ,	1 each

* indicates version number

Table 5: Strings sent in response to `version.bind`.

220 intercepted probes. Furthermore, we found such probes in countries around the world: these results are not due to an individual network's behavior.

We grouped the `version.bind` responses from these 49 CPE-intercepted probes by the strings they returned, as shown in Table 5. The majority were various versions of Dnsmasq, software that is explicitly designed to run on CPE [15]. We consider Dnsmasq's presence to be confirmation that the interceptor answering the `version.bind` query is the CPE. We also saw Dnsmasq strings on eight probes that indicated the device was a PiHole, suggesting that the owner deliberately intercepted DNS (presumably to avoid advertisements).

4.3 Is the interception within the client's ISP?

If we are unable to confirm that the probe's CPE is the interceptor, we then check whether the interceptor is within the probe's ISP using *bogon queries* (Section 3.3). Figure 4 shows the number of probes that are intercepted by their CPE, intercepted within their ISP, and the probes whose interception location is (potentially) beyond the ISP. For the RIPE Atlas nodes at least, this technique finds that when DNS queries are intercepted, they are intercepted close to the client (at the CPE or ISP) in a majority of cases. Moreover, in many of the countries, interception more often than not happens within the ISP, matching prior work's findings [24, 31] that DNS interception is often a result of ISP policy.

5 CASE STUDY: XB6 ROUTER

We first started to investigate CPE-based interception when it began interfering with our previous DNS experiments. We first experienced interception when we discovered that one author could not contact public resolvers from her residence. Upon investigation, we were able to identify the router model that performed the interception: the Arris/Technicolor XB6 [43].

The XB6 is manufactured by both Arris and Technicolor, but its hardware was designed by Comcast [6]. It uses a firmware package called RDK-B (Reference Design Kit), which is in use by more than 80 million devices around the world [8]. Other ISPs also license RDK-B and rent XB6 routers to their customers, including Shaw Communications, Vodafone, Liberty Global, and many others [8].

Notably, RDK-B includes a DNS resolver called XDNS, which stands for Xfinity DNS [7]. XDNS can redirect DNS queries using DNAT, which Comcast uses to implement malware filtering services [9]. XDNS also implements a response to `version.bind`.

The XDNS filtering service is intended to be opt-in. However, it appears that a bug in some XB6 routers is causing them to direct all queries to the ISP's resolver, without giving users any indication that their choice has been curtailed. This problem is not limited to Comcast: we have observed very similar behavior in other networks where the XB6 is deployed, including Shaw Communications and Vodafone. However, the bug appears to be uncommon.

We have reached out to ISPs about these discoveries. Their responses have been supportive and we are working to identify the source of the bug.

6 LIMITATIONS AND FUTURE WORK

This section describes our work's limitations and identifies directions for future work.

Our method is designed to measure the systematic interception of *all* DNS queries sent to a target resolver. We looked for systematic interception since we had observed DNAT-based transport/network-layer interception in the wild (Section 5). However, if only some queries are intercepted and others (such as our location queries) are not, our method will not determine interception.

Another limitation of our approach is that it relies upon the CPE answering `version.bind queries`. We do not expect this requirement to be a major limitation, however, since the BIND-like interface is now supported by many resolvers—even ones that do not use BIND.

Our approach also assumes that the DNS infrastructure of the client's ISP is located within the client's AS. If the ISP's resolver is located outside the client's AS, our approach will classify the interception's location as "unknown" instead of "within the ISP."

Additionally, our methodology may misclassify a non-intercepting CPE as an interceptor in a specific case: when the CPE has port 53 open, the CPE is a DNS forwarder, *and* the CPE does not respond to `version.bind` but instead forwards the query to a resolver.

We also note that RIPE Atlas is not a representative measurement platform and we therefore do not expect our results on the *prevalence* of DNS interception to generalize; we refer interested readers instead to recent work by Liu *et al.*, who investigated the prevalence of DNS interception [31]. Our goal in using RIPE Atlas is primarily to conduct a pilot study to show that our technique can detect interception and identify where it occurs.

While our approach should theoretically detect DNS interception in DNS over TLS (DoT) [23], we did not evaluate it on RIPE Atlas. DNS over HTTP (DoH) [22] and some configurations of DoT will prevent interception from occurring altogether, but the "opportunistic privacy profile" of DoT disables client certificate validation, so this configuration could allow interception. We leave evaluation of our method for detecting DoT interception for future work.

Techniques based on increasing the TTL of the IP header have the potential to identify which hop intercepted a query. The RIPE Atlas platform does not currently offer the ability to adjust the TTL of DNS requests, but we briefly explored using the VPNGate measurement platform [46] for this purpose. Unfortunately, we found that their VPN rewrites IP TTLs, rendering this experiment impossible. However, changing non-ICMP packet TTLs requires root or SUID root on most systems, whereas our approach only requires the ability to send DNS queries.

7 RELATED WORK

Due to the vital role of recursive DNS resolvers in Internet interactions, many of their properties have been studied, including their proximity to clients [1, 2, 10, 32, 36], their caching behavior [25, 26, 28, 34], and home gateway behavior [20]. Prior work has also studied vulnerabilities affecting DNS resolvers [21, 41, 42] and DNS cache snooping side channels that can reveal the popularity of web domains [33, 38–40, 50]. We focus our discussion on work that has studied DNS interception and DNS redirection.

Most prior work has studied *DNS redirection*, where (some) DNS responses received by user devices are altered (Section 2). DNS redirection is often employed to implement country-level or ISP-level policies. Researchers have reported that DNS requests sent from within China to third-party resolvers outside the country face DNS injection [4] (likely to implement censorship measures), and that even DNS requests that originate outside China but transit the country can be redirected due to collateral damage [3]. In a similar vein, studies have leveraged open recursive resolvers to investigate DNS manipulation whose likely purpose is to restrict user access to content [27, 37]. Chung *et al.* reported on NXDOMAIN wildcarding—the practice of rewriting NXDOMAIN errors with A records that point to a web server—and show that this form of DNS redirection may be occurring at the ISP's DNS server, public DNS servers, and ISP middleboxes [12]. Using data from Netalyzr, Kreibich *et al.* showed that NXDOMAIN wildcarding practices were prevalent among several ISPs [30, 47, 48]. Complementing these findings, our study shows that some instances of DNS *interception* occur due to potentially misconfigured CPE infrastructure; replacing these CPE devices sometimes suffices to prevent DNS interception. Our study also differs from previous work on NXDOMAIN wildcarding in that we study transparent interception rather than interception that is detectable by the client.

Also using RIPE Atlas probes, Jones *et al.* and Wei *et al.* measure how frequently DNS debugging queries (specifically `hostname.bind`) towards root servers are redirected [24, 49]. Their discovery of DNS redirection in existing RIPE Atlas datasets encouraged us to use the platform for our pilot study on DNS interception, although our work found `version.bind` to be better suited for our purposes.

Vallina-Rodriguez *et al.* measure the prevalence of DNS proxies in cellular networks by sending queries to their own authoritative nameserver [44]. Our work differs since we focus on where in the network interception is happening instead of its prevalence.

Liu *et al.* measured the prevalence of DNS interception in a recent study [31]. They performed DNS requests over both a commercial proxy network and from several Chinese mobile networks to estimate how common DNS interception is, but did not investigate where in the network interception takes place.

To the best of our knowledge, we are the first to investigate DNS interception over IPv6. Our results on RIPE Atlas suggest that DNS interception occurs far less frequently in IPv6 than in IPv4.

8 CONCLUSION

Our work provides a methodology for identifying the location of a DNS interceptor, whether the interceptor is the host's CPE device, a device in the host's AS, or elsewhere. Being able to empirically determine such information — that would otherwise be invisible to the user — is particularly relevant now when concerns about privacy and integrity have led to multiple proposed standards for embedding DNS traffic within encrypted tunnels: DNS over HTTPS and DNS over TLS. While the complex and many-sided nature of the debate around the deployment of such protocols is beyond the scope of this short paper, it is motivated by precisely the kinds of DNS interception that we observe and that can be more closely monitored by using our work.

9 ACKNOWLEDGEMENTS

We thank our shepherd Matt Calder and the anonymous reviewers for their insightful and constructive suggestions and feedback. We also thank Cindy Moore for her support of the software and hardware infrastructure necessary for this project. Funding for this work was provided in part by National Science Foundation grants CNS-1629973 and CNS-1705050, the Irwin Mark and Joan Klein Jacobs Chair in Information and Computer Science, and generous support from Google.

REFERENCES

[1] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. 2010. Comparing DNS Resolvers in the Wild. In *Proc. ACM Internet Measurement Conference (IMC)*. Melbourne, Australia, 15–21.

[2] Rami Al-Dalky and Kyle Schomp. 2018. Characterization of Collaborative Resolution in Recursive DNS Resolvers. In *Proc. Passive and Active Measurement Conference (PAM)*. Berlin, Germany, 146–157.

[3] Anonymous. 2012. The Collateral Damage of Internet Censorship by DNS Injection. In *Proc. ACM SIGCOMM*. Helsinki, Finland, 21–27.

[4] Anonymous. 2014. Towards a Comprehensive Picture of the Great Firewall's DNS Censorship. In *Proc. USENIX Workshop on Free and Open Communications on the Internet (FOCI)*. San Diego, CA, USA.

[5] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. 2013. Internet Censorship in Iran: A First Look. In *Proc. USENIX Workshop on Free and Open Communications on the Internet (FOCI)*. Washington, D.C., USA.

[6] Jeff Baumgartner. 2021. Comcast Taps Arris, Technicolor for 'XB6' Gateways: Sources. https://www.nexttv.com/news/comcast-taps-arris-technicolor-xb6-gateways-sources-409944.

[7] RDK Central. 2021. CcspXDNS. https://wiki.rdkcentral.com/display/RDK/CcspXDNS

[8] RDK Central. 2021. RDK Surpasses 80 Million Device Deployments Across Leading Video and Broadband Service Providers. https://rdkcentral.com/rdk-surpasses-80-million-device-deployments-across- leading-video-and-broadband-service-providers/.

[9] RDK Central. 2021. Source code for RDK-B. https://code.rdkcentral.com/r/plugins/gitiles/rdkb/components/opensource/ccsp/Utopia/+/7afe5aba7c8e9b89a182cdcffe16159c3b431b16/source/firewall/firewall.c

[10] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. 2015. End-User Mapping: Next Generation Request Routing for Content Delivery. In *Proc. ACM SIGCOMM*. London, United Kingdom.

[11] Chuck. 2011. "listen on 5353 too?". https://groups.google.com/g/public-dns-discuss/c/MfpyYHcqzjI

[12] Taejoong Chung, David Choffnes, and Alan Mislove. 2016. Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet. In *Proc. ACM Internet Measurement Conference (IMC)*. Santa Monica California USA, 199–213.

[13] Hacker Codex. 2012. How to Stop Your ISP from Hijacking Your DNS Servers. https://hackercodex.com/guide/how-to-stop-isp-dns-server-hijacking/

[14] Tom Creighton, Chris Griffiths, Jason Livingood, and Ralf Weber. 2010. DNS Redirect Use by Service Providers. Internet Draft: draft-livingood-dns-redirect-03.

[15] Dnsmasq. 2021. Dnsmasq - network services for small networks. https://thekelleys.org.uk/dnsmasq/doc.html

[16] Haixin Duan, Nicholas Weaver, Zongxu Zhao, Meng Hu, Jinjin Liang, Jian Jiang, Kang Li, and Vern Paxson. 2012. Hold-On: Protecting Against On-Path DNS Poisoning. In *Proc. Workshop on Securing and Trusting Internet Names (SATIN)*. London, United Kingdom.

[17] The Economist. 2013. France v Google. https://www.economist.com/business/2013/01/12/france-v-google

[18] Xun Fan, John Heidemann, and Ramesh Govindan. 2013. Evaluating Anycast in the Domain Name System. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*.

[19] Xfinity Community Forum. 2019. Changing the DNS on my machine didn't work... https://forums.xfinity.com/t5/Your-Home-Network/Changing-the-DNS-on-my-machine-didn-t-work/m-p/3268054#M309013

[20] Seppo Hätönen, Aki Nyrhinen, Lars Eggert, Stephen Strowes, Pasi Sarolahti, and Markku Kojo. 2010. An Experimental Study of Home Gateway Characteristics. In *Proc. ACM Internet Measurement Conference (IMC)*.

[21] Amir Herzberg and Haya Shulman. 2013. Fragmentation Considered Poisonous, or: one-domain-to-rule-them-all.org. In *IEEE Conference on Communications and Network Security (CNS)*.

[22] Paul Hoffman and Patrick McManus. 2018. *DNS Queries over HTTPS (DoH)*. RFC 9494. https://tools.ietf.org/html/rfc8484

[23] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858. https://tools.ietf.org/html/rfc7858

[24] Ben Jones, Nick Feamster, Vern Paxson, Nicholas Weaver, and Mark Allman. 2016. Detecting DNS Root Manipulation. In *Proc. Passive and Active Measurement Conference (PAM)*.

[25] Jaeyeon Jung, Arthur W. Berger, and Hari Balakrishnan. 2003. Modelling TTL-based Internet Caches. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*.

[26] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. 2002. DNS Performance and the Effectiveness of Caching. In *Proc. IEEE/ACM Transactions on Networking*.

[27] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz. 2015. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *Proc. ACM Internet Measurement Conference (IMC)*.

[28] Amit Klein, Haya Shulman, and Michael Waidner. 2017. Counting in the Dark: DNS Caches Discovery and Enumeration in the Internet. In *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[29] Mario Korf and Barb Strom. 2018. Introducing a New whoami Tool for DNS Resolver Information. https://developer.akamai.com/blog/2018/05/10/introducing-new-whoami-tool-dns-resolver-information

[30] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. 2010. Netalyzr: Illuminating the Edge Network. In *Proc. ACM Internet Measurement Conference (IMC)*.

[31] Liu, Baojun and Lu, Chaoyi and Duan, Haixin and Liu, Ying and Li, Zhou and Hao, Shuang and Yang, Min. 2018. Who is Answering My Queries: Understanding and Characterizing Interception of the DNS Resolution Path. In *Proc. USENIX Security*. Baltimore, MD, USA, 1113–1128.

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

[33] Andrew McGregor, Phillipa Gill, and Nicholas Weaver. 2021. Cache Me Outside: A New Look at DNS Cache Probing. In *Proc. Passive and Active Measurement Conference (PAM)*. Virtual, 427–443.

[34] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker. 2019. Cache Me If You Can: Effects of DNS Time-to-Live. In *Proc. ACM Internet Measurement Conference (IMC)*. Amsterdam, Netherlands, 101–115.

[35] RIPE NCC. 2015. Ripe Atlas: A Global Internet Measurement Network. *Internet Protocol Journal* (2015).

[36] John S. Otto, Mario A. Sánchez, John P. Rula, and Fabián E. Bustamante. 2012. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In *Proc. ACM Internet Measurement Conference (IMC)*. Boston, MA, USA, 523–536.

[37] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. 2017. Global Measurement of DNS Manipulation. In *Proc. USENIX Security*. Vancouver, BC, Canada.

[38] Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos. 2008. Peeking Through the Cloud: DNS-Based Estimation and Its Applications. In *Proc. Applied Cryptography and Network Security Conference (ACNS)*. New York, NY, USA.

[39] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. 2007. My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging. In *Proc. USENIX Workshop on Hot Topics in Understanding Botnets*. Cambridge, MA, USA.

[40] Audrey Randall, Enze Liu, Gautam Akiwate, Ramakrishna Padmanabhan, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. 2020. Trufflehunter: Cache Snooping Rare Domains at Large Public DNS Resolvers. In *Proc. ACM Internet Measurement Conference (IMC)*. Virtual, 50–64.

[41] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2014. Assessing DNS Vulnerability to Record Injection. In *Proc. Passive and Active Measurement Conference (PAM)*. Los Angeles, CA, USA, 214–223.

[42] Sooel Son and Vitaly Shmatikov. 2010. The Hitchhiker's Guide to DNS Cache Poisoning. In *Proc. International Conference on Security and Privacy in Communication Systems (SECURECOMM)*. Singapore, 466–483.

[43] Xfinity Help & Support. 2020. Overview of Xfinity Gateways. https://www.xfinity.com/support/articles/broadband-gateways-userguides

[44] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Nicholas Weaver, and Vern Paxson. 2015. Beyond the Radio: Illuminating the Higher Layers of Mobile Networks. In *Proc. ACM Conference on Mobile Systems, Applications, and Services (MobiSys)*. Florence, Italy, 375–387.

[45] Paul Vixie. 2009. What DNS is not. *Commun. ACM* 52, 12 (2009), 43–47.

[46] SoftEther VPN. 2021. VPNGate: Public VPN Relay Servers. https://vpngate.net

[47] Nicholas Weaver, Christian Kreibich, Boris Nechaev, and Vern Paxson. 2011. Implications of Netalyzr's DNS Measurements. In *Proc. Workshop on Securing and Trusting Internet Names (SATIN)*. Teddington, United Kingdom.

[48] Nicholas Weaver, Christian Kreibich, and Vern Paxson. 2011. Redirecting DNS for Ads and Profit. In *Proc. USENIX Workshop on Free and Open Communications on the Internet (FOCI)*. San Francisco, CA, USA.

[49] Lan Wei and John S. Heidemann. 2020. Whac-A-Mole: Six Years of DNS Spoofing. *arXiv* (2020). https://arxiv.org/abs/2011.12978

[50] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. 2003. Inferring Relative Popularity of Internet Applications by Actively Querying DNS Caches. In *Proc. ACM Internet Measurement Conference (IMC)*. Miami, Florida, USA, 78–90.

[51] Suzanne Woolf and David Conrad. 2007. *Requirements for a Mechanism Identifying a Name Server Instance*. RFC 4892. https://tools.ietf.org/html/rfc4892

A WHY VERSION.BIND IS NECESSARY TO DETECT CPE INTERCEPTION

We identify cases where the CPE is the interceptor by sending a specific CHAOS TXT query for `version.bind` to the public address of the CPE. Under ordinary routing rules, the CPE should not forward this packet to any other destinations, so if we receive a response to this query, we know the CPE has not obeyed usual routing rules and might be the interceptor. A reader might ask why it is necessary to send `version.bind`, which some resolvers are not configured to answer, rather than any DNS request for an ordinary A record: if we receive an answer for an ordinary A record request, does this indicate that the CPE is the interceptor?

Our reasoning is that it does not, and our logic is as follows. When a DNS request for an A record is sent to the CPE's public IP address, if the CPE's port 53 is open, even a non-intercepting CPE will return a response. This behavior is even true for `version.bind` queries. Therefore, the result of a single query to the CPE's public IP address is not sufficient to determine if the CPE is the interceptor. Our method relies on *comparing* the `version.bind` query sent to the CPE's public IP address with the `version.bind` query sent to the public resolver. The answer to a `version.bind` query is a string that is much more unique than an ordinary DNS response, and this property is necessary for determining if the CPE is the interceptor.

Consider the following scenario if an ordinary DNS query were used to determine the interceptor, leading to an incorrect conclusion. Let us assume the CPE is not the interceptor, but it does have port 53 open. If we were to send a query for `example.com` to the CPE's public IP address, the CPE would forward that query to its DNS resolver (for example, Comcast DNS) because its port 53 is open. We would receive the IP address of `example.com`, for example, "1.2.3.4." Next, we send a query for `example.com` to a public DNS resolver like Google DNS. The CPE is not the interceptor, so it sends the query towards Google DNS as intended. The query is intercepted further along the path, but no matter which resolver eventually answers it, the response is "1.2.3.4."

Now consider the case where the CPE *is* the interceptor. Both queries for `example.com` would be forwarded to the CPE's resolver, and both answers would come back as "1.2.3.4." We cannot tell whether the CPE was the interceptor because all answers to our queries are identical. The advantage of using `version.bind` is that it returns a more unique string. If the CPE is not the interceptor, but does have port 53 open, the query to the CPE's public IP address will return its own answer to `version.bind` (e.g., "Dnsmasq 2.7."). The query to a public resolver such as Google DNS will arrive at some non-CPE resolver further along the path, and will return that resolver's answer to `version.bind` (e.g., "PowerDNS"). The CPE's response to a `version.bind` query is unlikely to be identical to the intercepting resolver's response. But if the CPE is the interceptor, both `version.bind` queries will be handled by the CPE's resolver, and they will return identical answers. We can therefore determine with high confidence when the CPE is the interceptor.

B ETHICAL CONSIDERATIONS

Our work does not raise ethical concerns as we issue standard DNS queries towards major public DNS resolvers from a platform with volunteer-consent for such measurements.