Measuring DoT/DoH Blocking Using OONI Probe: a Preliminary Study

Simone Basso
Open Observatory of Network Interference
simone@openobservatory.org

***Abstract*—We designed DNSCheck, an active network experiment to detect the blocking of DoT/DoH services. We implemented DNSCheck into OONI Probe, the network-interference measurement tool we develop since 2012. We compiled a list of popular DoT/DoH services and ran DNSCheck measurements with help from volunteer OONI Probe users. We present preliminary results from measurements in Kazakhstan (AS48716), Iran (AS197207), and China (AS45090). We tested 123 DoT/DoH services, corresponding to 461 TCP/QUIC endpoints. We found endpoints to fail or succeed consistently. In AS197207 (Iran), 50% of the DoT endpoints seem blocked. Otherwise, we found that more than 80% of the tested endpoints were always reachable. The most frequently blocked services are Cloudflare's and Google's. In most cases, attempting to reach blocked endpoints failed with a timeout. We observed timeouts connecting, during, and after the TLS handshake. TLS blocking depends on either the SNI or the destination endpoint.**

I. Introduction

Since the Snowden revelations in 2013, the IETF community has started working towards a more encrypted and private internet. This spirit is well captured by, for example, RFC 7258 [20] and RFC 7624 [10]. The former states that pervasive monitoring is an attack to be mitigated, where possible, by IETF protocols. The latter provides a threat model for a scenario of widespread, pervasive internet surveillance.

As far as DNS is concerned, RFC 7624 notes that it travels the network in cleartext without confidentiality or integrity guarantees. A global (or country-wide) passive adversary could surveil all the DNS queries sent by users [7]. Alternatively, the adversary could censor the messages by discarding them or intercepting them and forging replies [29].

Two IETF standards address this flaw: DNS over TLS [26] (DoT) and DNS over HTTPS [23] (DoH). In both cases, public DoT/DoH servers typically rely on the existing Public Key Infrastructure (PKI) for authentication.

Providers, operating systems, and applications increasingly implement DoT and DoH. A 2019 study by Lu et al. [30] found that DoT traffic was growing but was "still 2-3 orders of magnitude less" than cleartext DNS (Do53). Cloudflare's DoT, e.g., received 7,318 monthly flows in December 2018 (56% up from July 2018). As regards DoH, they estimate that Google's endpoint received more than 10^7 monthly queries in March 2019 (a 100x increase since March 2018).

The ongoing deployment of these new technologies has interesting policy implications. Tunneling domain name resolutions over HTTPS, in particular, is a significant architectural change, not least because many DoH providers are also content delivery networks (CDN). This fact raises concerns regarding performance [25], competition, and privacy [14].

(While DoH's centralization and the resulting privacy concerns are not the focus of this paper, we note that work is underway to mitigate them [38] [24].)

Simultaneously, the rollout of DoT and DoH does not fully solve the surveillance and censorship issues posed by a cleartext internet. There are at least two remaining fields that could reveal the precise target of otherwise encrypted communications. They are the Server Name Indication [19] (SNI) extension inside the TLS ClientHello and the destination IP address. However, in a landscape increasingly dominated by CDNs [16], the same IP address may host several domains.

Hence, the next milestone to further encrypt communications is the TLS Encrypted ClientHello [37] [17] (ECH–formerly ESNI), which encrypts the ClientHello.

Crucially, ECH encrypts the ClientHello with encryption keys fetched from DNS servers. Fetching such keys over DoT/DoH helps to reduce the scope of local surveillance and censorship efforts. However, with several nation-states busy asserting their cyber-sovereignty [33] [9] [48], the question arises of whether they censor DoT/DoH and ECH.

In this vein, in the summer of 2020, Bock et al. investigated ECH/ESNI blocking in China [13]. They documented that the Great Firewall (GFW) could block ESNI traffic and presented six automatically discovered evasion strategies.

In this paper, instead, we focus on the complementary research question: measuring DoT/DoH services blocking.

Main contributions and findings

We model the blocking of DoT/DoH services (Section III) and design DNSCheck, a measurement methodology to measure such blocking (Section IV). DNSCheck consists of two steps. The first step checks whether the system resolver (and other Do53 resolvers) allow one to resolve a DoT/DoH service's domain name. The second step checks which TCP/QUIC endpoints associated with the service allow domain resolutions using the proper protocol (DoT or DoH).

We discuss measurements collected from Kazakhstan (Section V-C), Iran (Section V-D), and China (Section V-E), covering 123 DoT/DoH services and 461 TCP/QUIC endpoints. Helped by volunteer OONI Probe users, we ran tests between 15th December 2020 and 10th January 2021. We only observed Do53 censorship in AS197207 (Iran). Most DoT/DoH endpoints fail or succeed consistently. In AS197207,

around 50% of the DoT endpoints failed consistently. Apart from this case, for all the other ASN/protocol combinations, no more than 20% of the endpoints failed. The most frequently blocked services are Cloudflare's and Google's.

Timeout is the most frequent cause of failure. We see timeouts connecting, during, and right after the TLS handshake. We investigate timeouts during and after the TLS handshake (Section V-G). They seem to correspond to the discard of either outgoing TCP segments or incoming ACKs.

TLS blocking depends on either the SNI value or the destination endpoint. In particular, in Section V-I we show that in AS197207 (Iran), Google's endpoints blocking is independent of the SNI and only depends on the destination endpoint.

II. Related Work

This Section discusses censorship measurement tools, measuring TLS blocking, and DoT/DoH related research.

1) Censorship Measurement Tools: Niaki et al. [34] describe ICLab, a longitudinal censorship measurement platform using VPNs as distributed vantage points. Since late 2016, ICLab focuses on DNS manipulation, TCP packet injection, and block page detection and discovery.

Sundara Raman et al. [44] describe Censored Planet, a longitudinal internet-wide censorship observatory performing remote detection of TCP/IP blocking, DNS manipulation, and HTTP/HTTPS blocking. They rely on four different remote measurement tools: Augur [35], Satellite/Iris [39], Quack [47], and Hyperquack [45]. The latter detects SNI blocking by initiating a TLS conversation with a remote server inside a specific country. Because censorship is symmetrical in several cases, they can measure SNI censorship remotely.

IODA is a longitudinal internet measurement platform that detects country-wide internet blackouts, some of which have historically been politically motivated [18]. To detect such outages, IODA combines routing information, monitoring of unsolicited network traffic, and active probing.

The RIPE Atlas [1] [42] is a globally distributed platform consisting of hardware nodes hosted by volunteers. A credit system allows researchers to run experiments on the platform by scheduling tasks on the nodes. The Atlas has historically been used to detect the blocking of infrastructure and services. In 2014, for example, Anderson et al. used the Atlas to investigate blocking events in Russia and Turkey [8].

OONI Probe [21], the tool we use in this paper, is also a longitudinal censorship measurement tool. OONI Probe performs measurements from within a country as ICLab and RIPE Atlas do. Censored Planet, conversely, runs remote measurements. In this work, we focus on TLS blocking (similarly to Hyperquack) and on plaintext DNS (similar to ICLab and Satellite). IODA focuses on large-scale disruptions that make the internet unusable. In contrast, in this work, we focus on the selective censorship of DoT/DoH services.

2) TLS blocking: Chai et al. [17] measure (E)SNI blocking in China. Using both in-country and remote vantage points, they show that ESNI helps to circumvent SNI blocking. They use unrelated TLS servers as control servers, configure SNIs of interest, and check whether there are anomalies. Besides, they also check for DNS based blocking.

Singh et al. [41] measure SNI blocking while studying internet censorship in India. Their methodology consists of using TLSv1.3 and checking whether an offending SNI causes anomalies with a TLS-enabled control server. Besides, they also check for DNS based blocking.

The DNSCheck methodology we introduce here also has a DNS blocking step and a TLS blocking step. However, we use the possibly-censored target endpoints rather than unrelated control servers. Moreover, we not only check for TLS blocking, but we also check whether all the target endpoints are usable as DoT/DoH endpoints.

3) DoT/DoH: Lu et al. [30] describe large-scale measurements of DoT/DoH services. They document the growth of DoT/DoH services in terms of volume between 2018 and 2019. They find Google's DoH services blocked in China. Our findings confirm Google's DoH blocking in China.

Bushart et al. [15], Siby et al. [40], and Trevisan et al. [46] study the possibility of detecting the content of DoT/DoH queries notwithstanding the presence of encryption. Because detection is possible, blocking could also be possible. Conversely, our model assumes that a DoT/DoH service works for any query or is blocked. We plan on investigating if censors implement conditional blocking as part of our future work.

III. Model

This Section describes the DoT/DoH blocking model that informs DNSCheck's design and implementation.

A. Problem Statement

DoT/DoH services may use IP addresses, domain names, or both. Quad9's DoT resolver, for example, is available as both `9.9.9.9:853/tcp` and `dns.quad9.net:853/tcp`.

When a service uses a domain name, operating systems and applications use another resolver to look it up (often the system resolver, i.e., `getaddrinfo`).

Thus, to block a DoT/DoH service, a censor could interfere with domain resolution (if applicable) and with the service's endpoints [27]. For example, an adversary could block plaintext DNS queries for `dns.quad9.net`. Moreover, it could block some (or all) of the corresponding TCP endpoints.

As regards the blocking of endpoints, a censor could (1) prevent connecting to an endpoint, (2) block the TLS handshake, or (3) interfere after the TLS handshake.

(Orthogonally, the censor could force users to install a root certificate on their devices [43]. In turn, this certificate allows a censor to "man in the middle" selected endpoints to choose what "encrypted" DNS queries to block.)

To determine whether to block a flow, a censor could inspect a variety of cleartext fields. Previous research suggests that the most relevant ones are the destination endpoint [17] and TLS's SNI extension [41]. Censors can also possibly use the ALPN extension. Before TLS 1.3, they could also inspect the server's certificate, which the server sent in cleartext.

B. Measuring Blocking

Therefore, measuring the blocking of a DoT/DoH service is a two-step process. The first step should check whether resolving its domain name with cleartext DNS is possible. The second step should check whether each of its endpoints allows us to resolve a domain name.

C. Assumptions

We assume that blocking does not depend on the query's question as long as there is EDNS(0) padding [31]. Considering recent traffic analysis results [15] [40] [46], we plan to relax this assumption as part of our future work.

After receiving a DNS reply over a TLS/QUIC connection, we consider the corresponding endpoint accessible. OONI Probe's testing model does not currently allow us to measure the blocking of long-lived connections.

IV. Methodology

This Section describes OONI's architecture and the DNSCheck measurement methodology.

A. OONI Overview

The Open Observatory of Network Interference (OONI) is a free software distributed network-interference measurement system [21] [2]. Since 2012, we collected 379M measurements from 21.4k ASNs in 239 countries.

Volunteer users install OONI Probe, an open-source client available for Windows (desktop and CLI), macOS (desktop and CLI), Linux (CLI), Android, and iOS.

OONI Probe users can run censorship and network performance tests. Most censorship observations performed by OONI rely on a test that checks for website blocking.

B. OONI Measurements Lifecycle

After running a measurement, OONI Probe annotates it with the country code and the autonomous system number (ASN)[1]. Then, it submits the measurement to the OONI backend.

Upon receiving a measurement, the backend publishes it on Amazon S3. Then, the backend triggers a data processing pipeline. The pipeline's job is to classify/normalize the measurement and store meta information into a database.

In turn, the OONI API [3] allows anyone to find specific measurements by querying such a database. The primary consumer of the OONI API is OONI Explorer [4], a website allowing anyone to explore OONI measurements.

C. OONI Probe Flavors

There are two OONI Probe flavors: stable clients and the research client (also called miniooni). Ordinary users download and run stable clients. Miniooni is for researchers.

Both the stable client and miniooni use the same underlying Go library called Probe Engine [5]. The source code of the miniooni client is also part of the Probe Engine repository.

DNSCheck is still experimental. Therefore it is only accessible to users running miniooni.

D. DNSCheck Overview

DNSCheck first checks whether resolving the domain name of a DoT/DoH service is possible. Then, it checks whether all the related TCP/QUIC endpoints work as intended.

DNSCheck takes in input a test list describing the DoT/DoH services to test. Each service is identified by a "input" URL where the URL scheme (`https`, `dot`, or `udp`) indicates the protocol with which to access the service. The test list is further described in the Appendix.

E. DNSCheck Options

The `default_addrs` option specifies valid IP addresses for the URL's domain. DNSCheck will use them regardless of the result of resolving the URL's domain.

The `http3_enabled` option forces DNSCheck to use HTTP3 when performing DoH resolutions. The default is to negotiate one of HTTP/2 and HTTP/1.1 using ALPN [22].

The `domain` option specifies the domain to resolve. If not specified, we default to `example.org`. As stated in Section III-C, we assume that the resolved domain name does not influence DoT/DoH censorship. Thus, we have chosen to resolve a clearly-non-controversial name.

The `tls_server_name` option forces a specific Server Name Indication (SNI) value in the ClientHello.

F. DNSCheck Algorithm

The DNSCheck algorithm is composed of two steps called *bootstrap* and *lookups*. The *bootstrap* step discovers the IP addresses associated with the input URL's hostname. To this end, it uses `getaddrinfo`. If there is an error, *bootstrap* immediately returns this error. Otherwise, *bootstrap* returns the list of IPs returned by `getaddrinfo`[2].

The *lookups* step first merges the addresses returned by *bootstrap* with the ones in `default_addrs`.

Then, *lookups* computes the related TCP/UDP endpoint for each address, thus producing a list of endpoints to test.

Then, *lookups* computes whether to use an SNI. There are three cases. If the URL's hostname is an IP address, we do not use any SNI[3]. Otherwise, we use `tls_server_name`, if not empty. Otherwise, we use the URL's hostname.

For each endpoint, *lookups* will establish an encrypted connection. Then it will use the connection to send a DNS query for `example.org` and receive the response.

When using TCP, *lookups* will first create a TCP connection and then perform a TLS handshake. If connecting fails, *lookups* records that the failed operation is `connect`. If handshaking fails, it will record `tls_handshake`. When the failed operation is `resolve`, it means the related failure occurred *after* the TLS handshake.

[1] We use various STUN and HTTP services to discover our IP address. Then, we derive country code and ASN using MaxMind compatible databases. We use db-ip.com's free country geolocation database and generate our own ASN database from RIPE's public BGP data dumps.

[2] When passed an IP address as the domain, `getaddrinfo` returns such an address. Therefore, if the input URL hostname is an IP address, the *bootstrap* step will return a list containing such an IP address.

[3] Go enforces this behavior, consistently with RFC 6066 [19].

When using QUIC, *lookups* will create an encrypted QUIC connection in a single operation. (The current QUIC code also does not correctly record the failed operation.)

G. DNSCheck Implementation Details

Users in censored locations often configure alternative resolvers, so there is no guarantee that `getaddrinfo` uses the ISP's resolver. For this reason, we discover and store the system resolver's IP in each measurement[4].

Probe Engine uses its own CA bundle extracted from Mozilla's certificate store. Thus, the system's certificate store's content does not influence the measurement results.

With DoT and DoH, we always include padding according to the recommendations of RFC8467 [32]. We set the ALPN extension to `dot` for DoT. We use `h2, http/1.1` for DoH over TCP (like Firefox), and `h3-29` for DoH over QUIC.

Probe Engine is written in Go and uses the miekg/dns library [6] to parse and serialize DNS messages.

V. Measurements Results

This Section describes the measurement results. We compiled a test list consisting of widely used DoT/DoH services, including, of course, Google's, Cloudflare's, and Quad9's.

When generating the test list, we used Google's DoH resolver to include valid `default_addrs` addresses for each input URL containing a domain name. (See the Appendix for more details on how we did that.)

Starting from 15th December 2020, we and some volunteers ran this test list using the most recent version of miniooni.

A. DoT/DoH Services IP Addresses

In the remainder of Section V we will refer to the following DoT/DoH services' IP addresses:

```
1.0.0.1                 Cloudflare
1.1.1.1                 Cloudflare
2606:4700::6810:f8f9    Cloudflare
2606:4700:4700::1001    Cloudflare
2606:4700:4700::1111    Cloudflare
104.16.248.249          Cloudflare
8.8.4.4                 Google
8.8.8.8                 Google
9.9.9.9                 Quad9
9.9.9.10                Quad9
```

We list them here for clarity.

B. Data Quality

This Section describes data quality issues that we identified and fixed during measurements analysis.

1) Missing IPv6 support: Some systems did not support IPv6, but our test list included plenty of IPv6 addresses in the `default_addrs` field. Therefore, we needed to filter out errors indicating that IPv6 was not supported.

[4]To this end, we query `whoami.akamai.net`. The response includes the source IP address that queried akamai's authoritative name server.

address	sni	result	freq.	perc.
1.0.0.1	null[5]	success	88	100%
1.0.0.1	one.one.one.one	success	88	100%
1.0.0.1	1dot1dot1dot1.cl...	resolve[6] timeout	73	99%
1.0.0.1	1dot1dot1dot1.cl...	connect timeout	1	1%
1.1.1.1	null	tls_handshake timeout	88	100%
1.1.1.1	one.one.one.one	success	88	100%
1.1.1.1	1dot1dot1dot1.cl...	resolve timeout	77	100%
...4700::1001	one.one.one.one	success	88	100%
...4700::1001	1dot1dot1dot1.cl...	resolve timeout	85	100%
...4700::1111	one.one.one.one	success	88	100%
...4700::1111	1dot1dot1dot1.cl...	resolve timeout	84	100%

TABLE I
Frequency of results for Cloudflare DoT measurements in KZ (AS48716)

2) Misconfigured Server: `40.76.112.230` (AT&T's experimental DoT/DoH endpoint) frequently served us an expired certificate. This error happened for vantage points in Cambodia (AS139779; Cambo.Host LTD), Costa Rica (AS52423; Data Miners SA), India (AS14061; Digital Ocean, LLC), and Italy (AS30722; Vodafone Italia SpA).

Therefore, we did not consider results associated with such an endpoint when performing data analysis.

3) False Positives: A specific version of miniooni had the following bug. It scheduled endpoints measurements after a timeout was expired. This bug created false positives where it appeared that `connect` timed out immediately.

We fixed the bug on 26th December 2020 at 10:44 UTC and updated all miniooni instances. We omit false positives caused by this bug from the results.

C. Kazakhstan

We ran DNSCheck 88 times between 15th December 2020 and 10th January 2021 from a VPS located in AS48716 (PS Internet Company LLC). Most runs measured the whole test list; some runs were interrupted. We did not find any issue during the bootstrap phase. Some data points are missing because of false positives; see Section V-B3.

1) DoT: We ran 8,603 DoT lookups, 95% of which succeeded. More than 70% of the failures are timeouts after the TLS handshake. More than 90% of the failures occur with Cloudflare's endpoints. Table I shows the measurement results for some endpoints. Results seem to depend on the SNI.

2) DoH: We ran 19,988 DoH lookups, 82% of which succeeded. More than 75% of the failures are timeouts after the TLS handshake. More than 85% of the failures occur with Cloudflare's endpoints.

Table II shows the measurement results for some of these endpoints. The endpoints fail consistently with the SNI. When there is no SNI, the behavior is inconsistent.

Figure 1 shows the results of the `1.1.1.1` endpoint without any SNI over time. The endpoint worked for some time; then, it started failing again. (The `1.0.0.1` endpoint

[5]A null SNI indicates that the URL's hostname is an IP address. In such cases, there should be no SNI extension according to RFC 6066 [19].

[6]Resolve indicates that the failure occurred after the TLS handshake.

address	sni	result	freq.	perc.
...6810:f8f9	cloudflare-dns.c...	resolve timeout	85	99%
...6810:f8f9	cloudflare-dns.c...	connect timeout	1	1%
...6810:f8f9	mozilla.cloudfla...	success	88	100%
...6810:f8f9	ooni.cloudflare....	resolve timeout	87	100%
1.1.1.1	null	tls_handshake timeout	68	78%
1.1.1.1	null	success	19	22%
1.1.1.1	1dot1dot1dot1.cl...	resolve timeout	87	100%
1.0.0.1	null	tls_handshake timeout	60	69%
1.0.0.1	null	success	27	31%
1.0.0.1	1dot1dot1dot1.cl...	resolve timeout	87	100%

TABLE II
FREQUENCY OF RESULTS FOR CLOUDFLARE DOH MEASUREMENTS IN KZ (AS48716)

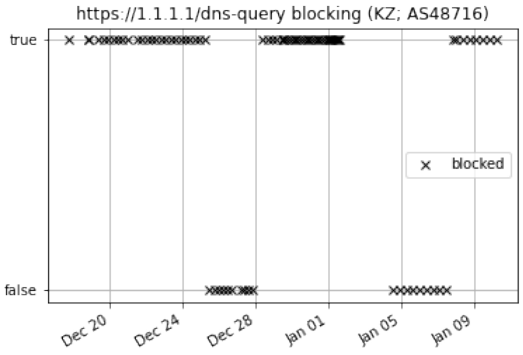

Fig. 1. 1.1.1.1 DoH results in KZ (AS48716)

address	sni	result	freq.	perc.
1.0.0.1	null	tls_handshake timeout	40	100%
1.0.0.1	1dot1dot1dot1.cl...	tls_handshake timeout	40	100%
1.0.0.1	one.one.one.one	tls_handshake timeout	38	97%
1.0.0.1	one.one.one.one	connect refused	1	3%
1.1.1.1	null	success	26	67%
1.1.1.1	null	connect timeout	13	33%
1.1.1.1	1dot1dot1dot1.cl...	success	27	68%
1.1.1.1	1dot1dot1dot1.cl...	connect timeout	13	32%
1.1.1.1	one.one.one.one	success	27	68%
1.1.1.1	one.one.one.one	connect timeout	13	32%
8.8.4.4	null	tls_handshake timeout	40	100%
8.8.4.4	8888.google	tls_handshake timeout	40	100%
8.8.4.4	dns.google	tls_handshake timeout	39	100%
8.8.4.4	dns.google.com	tls_handshake timeout	39	100%
8.8.8.8	null	success	40	100%
8.8.8.8	8888.google	success	40	100%
8.8.8.8	dns.google	success	39	100%
8.8.8.8	dns.google.com	success	39	100%
9.9.9.9	null	tls_handshake timeout	40	100%
9.9.9.9	dns.quad9.net	tls_handshake timeout	39	100%
9.9.9.9	dns9.quad9.net	tls_handshake timeout	39	100%
9.9.9.10	null	tls_handshake timeout	40	100%
9.9.9.10	dns-nosec.quad9.net	tls_handshake timeout	38	100%
9.9.9.10	dns10.quad9.net	tls_handshake timeout	39	100%
10.10.34.36	dns.adguard.com	connect timeout	33	100%
94.140.14.14	dns.adguard.com	tls_handshake reset	37	95%
94.140.14.14	dns.adguard.com	tls_handshake timeout	2	5%
94.140.15.15	dns.adguard.com	tls_handshake reset	37	95%
94.140.15.15	dns.adguard.com	tls_handshake timeout	2	5%

TABLE III
FREQUENCY OF RESULTS FOR SELECTED DOT MEASUREMENTS IN IR (AS197207). (NOTE: ALL THE ELIDED SNIS BELONG TO CLOUDFLARE.COM.)

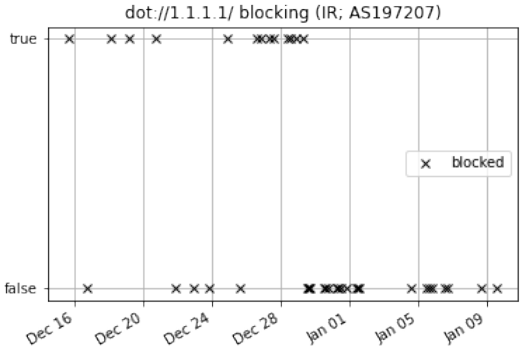

Fig. 2. 1.1.1.1 DoT results in IR (AS197207)

behaves roughly in the same way: its blocking pattern also oscillates, but differently.)

D. Iran

We ran DNSCheck 40 times between 15th December 2020 and 10th January 2020 from AS197207 (Mobile Telecommunication Company of Iran–MCI). Some data points are missing because of false positives; see Section V-B3.

1) Bootstrap Analysis: MCI's system resolver consistently returned to us a bogon [28] IPv4 address (`10.10.34.36`) for every `A` query for `dns.adguard.com`.

We got the same bogon when using `8.8.8.8:53/udp` and `9.9.9.9:53/udp` as alternative resolvers. Interestingly, we did not observe any bogon reply for `AAAA` queries.

The bogon address is in the same `/24` network indicated in the paper on censorship in Iran by Aryan et al. [9].

With manual measurements, we also noticed, as they noticed [9], that the filtered queries never reached a DNS-over-UDP server that we temporarily set up.

This fact strongly suggests that some equipment intercepted the queries and replied on behalf of the real server.

2) DoT: We ran 2,290 DoT lookups, 50% of which succeeded. More than 50% of the failures occur with Cloudflare's, Google's, or Quad9's endpoints. Around 80% of the failures are TLS handshake timeouts.

Table III shows the measurement results for a selection of DoT endpoints. We see that endpoints fail consistently. The Table does not show any correlation between SNI and failure. The differences between these results and our June-2020 investigation of DoT blocking in Iran [11] are: `8.8.8.8` does not fail anymore; `1.1.1.1` now fails inconsistently.

We also see from Table III how the bogon address for `dns.adguard.com` fails. We also see how the additional IP addresses for this domain fail during the TLS handshake. We also confirmed with manual measurements using Singh et al. [41]'s methodology that there is SNI blocking of `dns.adguard.com` when using port 853.

address	sni	result	freq.	perc.
1.1.1.1	null	success	26	65%
1.1.1.1	null	connect timeout	13	33%
1.1.1.1	null	http failure	1	2%
1.1.1.1	1dot1dot1dot1.cl...	success	27	68%
1.1.1.1	1dot1dot1dot1.cl...	connect timeout	13	32%
8.8.4.4	8888.google	success	40	100%
8.8.4.4	dns.google	resolve timeout	40	100%
9.9.9.9	null	success	40	100%
9.9.9.9	dns.quad9.net	resolve timeout	40	100%
9.9.9.9	dns9.quad9.net	resolve timeout	40	100%
10.10.34.36	dns.adguard.com	connect refused	31	100%
94.140.14.14	dns.adguard.com	tls_handshake reset	38	97%
94.140.14.14	dns.adguard.com	tls_handshake timeout	1	3%
94.140.15.15	dns.adguard.com	tls_handshake reset	39	100%

TABLE IV
FREQUENCY OF RESULTS FOR SELECTED DOH MEASUREMENTS IN IR (USING TCP; AS197207). (NOTE: ALL THE ELIDED SNIS BELONG TO CLOUDFLARE.COM.)

Figure 2 shows the results of the `1.1.1.1` endpoint over time. The blocking status of the endpoint is not consistent.

3) DoH: We ran 5,213 DoH lookups, 93% of which succeeded. Timeout after the handshake (41%) and reset during the handshake (20%) are the most frequent failures. About 53% of the failures occur with Cloudflare, Google, or Quad9 endpoints.

Table IV shows the results for some of these endpoints. Results depend on the SNI for Google and Quad9. As regards `1.1.1.1`, on the contrary, it fails as it does for DoT.

Like for DoT, we see both DNS and TLS blocking for the `dns.adguard.com` endpoints. DNS blocking leads us to use a bogon address. TLS blocking prevents us from using valid IP addresses for the domain. We manually investigated using Singh et al. [41]'s methodology, and found that there is SNI blocking for `dns.adguard.com` on port 443.

E. China

We ran DNSCheck 81 times between 15th December 2020 and 10th January 2020 from a VM running on AS45090 (Tencent cloud computing Co., Ltd.). We did not find any issue during the bootstrap phase. Some data points are missing because of false positives; see Section V-B3.

1) DoT: We ran 4,643 DoT lookups[7], 93% of which succeeded. Around 75% of the failures are `connect` timeouts. More than 90% of the failures occur with Cloudflare's or BlahDNS's Japanese endpoints.

Table V provides more details about the most frequently observed DoT timeout errors. Cloudflare's DoT endpoint always fails when connecting. Conversely, BlahDNS's Japanese endpoint most often fails during the TLS handshake.

Our finding that most blocked endpoints fail on connect is consistent with previous measurements [17].

[7]We did not have IPv6 connectivity in AS45090, hence the significantly-lower number of lookups than in AS48716.

[8]This endpoint timed out consistently until 1st January 2020 at 10:40 UTC, when the connection was refused. After that moment, it was reachable.

address	sni	result	freq.	perc.
1.1.1.1	1dot1dot1dot1.cl...	connect timeout	77	100%
1.1.1.1	one.one.one.one	connect timeout	77	100%
1.1.1.1	null	connect timeout	76	100%
45.32.55.94	dot-jp.blahdns.com	tls_handshake timeout	61	75%
45.32.55.94	dot-jp.blahdns.com	success[8]	19	24%
45.32.55.94	dot-jp.blahdns.com	connect refused	1	1%

TABLE V
FREQUENCY OF RESULTS FOR SELECTED DOT MEASUREMENTS CN (AS45090). (NOTE: ALL THE ELIDED SNIS BELONG TO CLOUDFLARE.COM.)

sni	result	freq.	perc.
cloudflare-dns.com	success	81	100%
mozilla.cloudflare-dns.com	tls_handshake reset	75	93%
mozilla.cloudflare-dns.com	success	6	7%
ooni.cloudflare-dns.com	success	78	96%
ooni.cloudflare-dns.com	connect timeout	2	3%
ooni.cloudflare-dns.com	tls_handshake reset	1	1%

TABLE VI
FREQUENCY OF RESULTS FOR DOH MEASUREMENTS FOR 104.16.248.249 IN CN (AS45090)

	KZ (AS48716)		IR (AS197207)		CN (AS45090)	
transport	count	perc.	count	perc.	count	perc.
DoT	8157	95%	1150	50%	4332	93%
DoH over TCP	16466	82%	4824	92%	9414	89%

TABLE VII
SUCCESSFUL LOOKUPS PER TRANSPORT FOR DOT AND DOH OVER TCP

2) DoH: We ran 10,536 DoT lookups, 89% of which succeeded. Around 72% of the failures are `connect` timeouts. More than 70% of the failures occur with Cloudflare's or Google's endpoints.

Regarding failures, `1.1.1.1`, `1.1.1.2`, `1.1.1.3`, and `8.8.8.8` consistently timeout when connecting. Lu et al. [30] also mention this blocking of Google's DoH in China.

Regarding `104.16.248.249:443/tcp`, instead, failures seem to depend on the SNI (see Table VI).

Follow-up measurements confirm this hypothesis. Using that SNI with unrelated HTTPS servers (e.g., `example.org` and `hbl.fi`) leads to connection-reset failures. This result is consistent with other literature findings mentioning that China uses SNI-based blocking [17].

We also noticed that `8.8.8.8:443/udp` always timed out in the QUIC handshake for any tested SNI. The same occurred for `8.8.4.4:443/udp`.

F. Comparison Between ASNs

Table VII shows the success rate of lookups measurements per transport per ASN. We see that more than 80% of the measurements were successful in most cases. The only exception is DoT measurements in AS197207, where around 50% of the lookups failed.

Table VIII shows the distribution of DoT lookups failures per ASN. We see that timeout after the TLS handshake dominates in AS48716 (KZ). Timeout during the TLS handshake

	KZ (AS48716)		IR (AS197207)		CN (AS45090)	
failure	freq.	perc.	freq.	perc.	freq.	perc.
resolve timeout	323	72%	79	7%	2	∼0%
tls_handshake timeout	88	20%	906	80%	63	20%
connect timeout	1	∼0%	72	6%	233	75%
tls_handshake reset	1	∼0%	74	7%	0	0%
other	33	8%	3	∼0%	13	∼5%
total	446	100%	1134	100%	304	100%

TABLE VIII
DISTRIBUTION OF DOT LOOKUPS FAILURES

	KZ (AS48716)		IR (AS197207)		CN (AS45090)	
failure	freq.	perc.	freq.	perc.	freq.	perc.
resolve timeout	2701	77%	160	41%	3	∼0%
tls_handshake timeout	331	9%	1	∼0%	61	5%
connect timeout	397	11%	72	19%	813	72%
tls_handshake reset	1	∼0%	77	20%	152	14%
other	92	3%	79	20%	93	9%
total	3522	100%	389	100%	1122	100%

TABLE IX
DISTRIBUTION OF DOH-OVER-TCP LOOKUPS FAILURES

is the most frequent failure in AS197207 (IR). In AS45090 (CN), the majority of failures are timeouts when connecting.

Table VIII shows the distribution of DoH-over-TCP lookups failures per ASN. We see that timeout after the TLS handshake dominates in AS48716 (KZ) and AS197207 (IR). In AS45090 (CN), connect timeout is the most frequent failure.

In AS197207 (IR), 60% of the DoH-over-TCP failures are timeouts. In all other cases, this percentage is higher. This fact makes timeout the most frequent failure by far.

AS45090 (CN) most frequently blocks by IP address. Interference during or after the TLS handshake is otherwise most frequent. In the next Section, we will try to better characterize this kind of interference by inspecting packet captures.

G. Investigating TLS timeouts

Probe Engine collects the timing and result of the `connect`, `recv`, and `send` syscalls. In this Section, we compare these results with a few packet captures to understand what could cause TLS timeouts.

Timeouts during or after the TLS handshake always have the following signature. The code writes some data. Then it waits for the socket to become readable. The socket never becomes readable, and, eventually, a timeout expires.

The few packet captures we inspected show that TCP eventually sends a segment that is never acknowledged. Thus, a few retransmissions ensue, then the code times out and closes the socket.

For timeouts during the TLS handshake, the never acknowledged segment contains the ClientHello. For timeouts after the TLS handshake, judging from the packet sizes, the application-level `send` results, and the ACK numbers, the client retransmits a TCP segment containing three pieces of information. That is, a dummy ChangeCipherSpec message, the ClientHandshakeFinished message, and the DNS query.

The application sees the blocking after the end of the TLS handshake for the following reason. As far as the TLS library is concerned, the handshake is complete once it has written the final handshake messages on the socket. There is no way for the TLS library to know that TCP will need to retransmit such messages. Therefore, the TLS library returns successfully, and the miniooni code then sends the DNS query.

After the code closes the socket, TCP transmits the FIN segment without receiving any response.

(Because we did not control any blocked server, we could not capture packets on the server-side. This fact prevented us from determining whether the outgoing segments were lost or the incoming ACKs were lost.)

endpoint	sni	tls_version	result	freq.	perc.
8.8.4.4:853/tcp	8888.google	null	timeout	210	100%
8.8.8.8:853/tcp	8888.google	TLSv1.3	success	80	100%

TABLE X
FREQUENCY OF RESULTS FOR TLS HANDSHAKES FOR DOT://8888.GOOGLE (AS197207; IR)

H. Forcing TLSv1.3

We repeated some failed measurements forcing TLSv1.3, and we did not see any significant change in the results.

I. Destination-Endpoint-Based TLS Blocking

The `dot://8888.google` endpoint inconsistently fails in Iran (see Table III). Let us now investigate all the related TLS handshakes in Table X. (We see more failures than successes because Probe Engine retries failed queries.)

In particular, let us interpret the second row of Table X using Singh et al. [41]'s SNI-blocking methodology. We use TLSv1.3, so the network could not see the server's certificate. All the TLS handshakes succeed. Hence, we can conclude that there is no SNI based blocking for `8888.google`.

We can apply the same reasoning for all the other Google-related SNIs in Table III. We thus conclude that Google's DoT blocking should only depend on the destination endpoint.

J. Impact of the Scheduling Policy

Chai et al. [17] mention that the Great Firewall of China (GFW) implements "residual censorship" along with SNI-based blocking. After the initial SNI-blocking episode, an endpoint remains blocked for 60 seconds.

Bock et al. [13] report up to 180 seconds of residual censorship when characterizing ESNI blocking. They also detected up to 60 seconds of residual censorship in Iran when investigating the Iranian white-list protocol filter [12].

These papers deal with triggering (E)SNI-blocking from random control servers and sending unknown traffic. We do not know whether residual censorship applies to traffic targeting the expected/legitimate servers.

To search for evidence of residual censorship, we modified our scheduling policy. Since 26th December 2020 at 10:44 UTC, we randomized the test list before every run. Since 1st January 2021 at 00:17 UTC, we waited 180 seconds before measuring the same endpoint again.

We noticed no significant impact of changing the scheduling policy on the results. We thus assume that there was no residual censorship for the tested endpoints.

K. Discussion

Ramesh et al. [36] mention that China and Iran have a centralized censorship implementation. This observation gives us confidence that our results could represent typical network censorship in these countries. In particular, our testing in Iran focused on the most widely used ISP (MCI). We also ran sparse testing in other Iranian ISPs with similar results.

Regarding Kazakhstan, Sundara Raman et al. [43] noted that TLS interception only occurred in AS9198 (Kazakhtelecom). Our measurements are from AS48716, which does not appear to use AS9198 as an upstream provider. Our plan to expose DNSCheck to all OONI users gives us confidence that we will soon improve our coverage.

We observed consistent lookup results, in any case. Apart from a few temporary failures, some endpoints failed consistently, most endpoints consistently succeeded. For example, in Kazakhstan (AS48716), `1.0.0.1` is blocked or accessible depending on the SNI. In Iran (AS197207), `8.8.8.8` is always accessible, and `8.8.4.4` is always blocked.

A few endpoints changed their behavior during the observation period (see Figures 1 and 2). We speculate that they may have been blocked and unblocked intermittently. We plan on further investigating these oscillations as part of our future work (provided that they continue to happen).

In AS197207 (Iran), 50% of the DoT lookups failed. Otherwise, at least 80% of the endpoints were usable. The most commonly blocked services were Cloudflare and Google.

Different endpoints failed differently. The most common failure is timeout. We have seen more connect timeouts in AS45090 (China) and more timeouts after the TLS handshake in AS48716 (Kazakhstan). The typical failure mode was different for DoT and DoH in AS197207 (Iran).

VI. Conclusion

We presented DNSCheck, an active network experiment to detect DoT/DoH blocking integrated into OONI Probe. We discussed preliminary results from experiments in Kazakhstan (AS48716), Iran (AS197207), and China (AS45090).

In AS197207, around 50% of the DoT endpoints failed consistently. In the other ASN/protocol combinations, more than 80% of the endpoints were always reachable. Cloudflare's and Google's were the most blocked services.

We observed that the most frequent cause of failure was a timeout, and we showed that DoT/DoH blocking could depend on the SNI or the destination endpoint.

Acknowledgments

We are in debt to users in the OONI community who helped us run this preliminary measurements campaign. In particular, we are thankful to GreatFire.org, who helped us performing measurements in China.

We would also like to thank Arturo Filastò, Divyank Katira, and Kathrin Elmenhorst. Arturo provided useful comments to improve this paper. Divyank implemented the first version of DNSCheck in Go. Kathrin added QUIC support to OONI Probe and helped us improve the readability of this paper.

References

[1] [Online]. Available: https://atlas.ripe.net/
[2] [Online]. Available: https://ooni.org
[3] [Online]. Available: https://api.ooni.io
[4] [Online]. Available: https://explorer.ooni.org
[5] [Online]. Available: https://github.com/ooni/probe-engine
[6] [Online]. Available: https://github.com/miekg/dns
[7] "The NSA and GCHQ's QUANTUMTHEORY Hacking Tactics," 2014. [Online]. Available: https://theintercept.com/document/2014/03/12/nsa-gchqs-quantumtheory-hacking-tactics/
[8] C. Anderson, P. Winter *et al.*, "Global network interference detection over the RIPE atlas network," in *4th USENIX Workshop on Free and Open Communications on the Internet ({FOCI} 14)*, 2014.
[9] S. Aryan, H. Aryan, and J. A. Halderman, "Internet Censorship in Iran: A First Look," in *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, Washington, D.C., 2013.
[10] R. Barnes, B. Schneier, C. Jennings, T. Hardie, B. Trammel, C. Huitema, and D. Borkman, "Confidentiality in the face of pervasive surveillance: A threat model and problem statement," Internet Requests for Comments, RFC Editor, RFC 7264, 2015. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7264.txt
[11] S. Basso, "DNS over TLS blocked in Iran," Tech. Rep., 2020. [Online]. Available: https://ooni.org/post/2020-iran-dot/
[12] K. Bock, Y. Fax, K. Reese, J. Singh, and D. Levin, "Detecting and Evading Censorship-in-Depth: A Case Study of Iran's Protocol Whitelister," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association, 2020.
[13] K. Bock, iyouport, Anonymous, L.-H. Merino, D. Fifield, A. Houmansadr, and D. Levin, "Exposing and Circumventing China's Censorship of ESNI," Tech. Rep., 2020. [Online]. Available: https://gfw.report/blog/gfw_esni_blocking/en/
[14] K. Borgolte, T. Chattopadhyay, N. Feamster, M. Kshirsagar, J. Holland, A. Hounsel, and P. Schmitt, "How DNS over HTTPS is Reshaping Privacy, Performance, and Policy in the Internet Ecosystem," in *TPRC47: The 47th Research Conference on Communication, Information and Internet Policy 2019*, 2019.
[15] J. Bushart and C. Rossow, "Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association, 2020.
[16] E. Carisimo, C. Selmo, J. Alvarez-Hamelin, and A. Dhamdhere, "Studying the Evolution of Content Providers in IPv4 and IPv6 Internet Cores," *Elsevier Computer Communications Journal*, vol. 145, pp. 54–65, Sep 2019.
[17] Z. Chai, A. Ghafari, and A. Houmansadr, "On the Importance of Encrypted-SNI (ESNI) to Censorship Circumvention," 2019.
[18] A. Dainotti, C. Squarcella, E. Aben, K. C. Claffy, M. Chiesa, M. Russo, and A. Pescapé, "Analysis of country-wide internet outages caused by censorship," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 1–18.
[19] D. Eastlake, "Transport layer security (TLS) extensions: Extension definitions," Internet Requests for Comments, RFC Editor, RFC 6066, 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6066.txt
[20] S. Farrell and H. Tschofenig, "Pervasive Monitoring Is an Attack," Internet Requests for Comments, RFC Editor, RFC 7258, 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7258.txt
[21] A. Filasto and J. Appelbaum, "OONI: Open Observatory of Network Interference," in *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*. USENIX Association, 2012.
[22] S. Friedl, A. Popov, A. Langley, and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension," Internet Requests for Comments, RFC Editor, RFC 7301, 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7301.txt
[23] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," Internet Requests for Comments, RFC Editor, RFC 8484, 2018. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8484.txt
[24] A. Hounsel, K. Borgolte, P. Schmitt, and N. Feamster, "D-DNS: towards re-decentralizing the DNS," *CoRR*, vol. abs/2002.09055, 2020. [Online]. Available: https://arxiv.org/abs/2002.09055
[25] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, "Comparing the Effects of DNS, DoT, and DoH on Web Performance," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 562–572.

[26] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over transport layer security (TLS)," Internet Requests for Comments, RFC Editor, RFC 7858, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7858.txt
[27] Q. Huang, D. Chang, and Z. Li, "A Comprehensive Study of DNS-over-HTTPS Downgrade Attack," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*. USENIX Association, 2020.
[28] G. Jones, "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure," Internet Requests for Comments, RFC Editor, RFC 3871, 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3871.txt
[29] B. Liu, C. Lu, H. Duan, Y. Liu, Z. Li, S. Hao, and M. Yang, "Who is answering my queries: Understanding and characterizing interception of the DNS resolution path," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1113–1128.
[30] C. Lu, B. Liu, Z. Li, S. Hao, H. Duan, M. Zhang, C. Leng, Y. Liu, Z. Zhang, and J. Wu, "An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?" in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 22–35. [Online]. Available: https://doi.org/10.1145/3355369.3355580
[31] A. Mayrhofer, "The EDNS(0) Padding Option," Internet Requests for Comments, RFC Editor, RFC 7830, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc783.txt
[32] ——, "Padding Policies for Extension Mechanisms for DNS (EDNS(0))," Internet Requests for Comments, RFC Editor, RFC 8467, 2018. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8467.txt
[33] M. L. Mueller, "Against Sovereignty in Cyberspace," *International Studies Review*, vol. 22, no. 4, pp. 779–801, 09 2019. [Online]. Available: https://doi.org/10.1093/isr/viz044
[34] A. A. Niaki, S. Cho, Z. Weinberg, N. P. Hoang, A. Razaghpanah, N. Christin, and P. Gill, "ICLab: A Global, Longitudinal Internet Censorship Measurement Platform," in *2020 IEEE Symposium on Security and Privacy (SP)*, ser. Oakland 2020, IEEE. IEEE Computer Society, May 2020, pp. 214–230. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP.2020.00014
[35] P. Pearce, R. Ensafi, F. Li, N. Feamster, and V. Paxson, "Augur: Internet-wide detection of connectivity disruptions," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 427–443.
[36] R. Ramesh, R. Sundara Raman, M. Bernhard, V. Ongkowijaya, L. Evdokimov, A. Edmundson, S. Sprecher, M. Ikram, and R. Ensafi, "Decentralized control: A case study of russia," in *Network and Distributed Systems Security Symposium (NDSS)*, 2020.
[37] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-tls-esni-09.txt, 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tls-esni-09
[38] P. Schmitt, A. Edmundson, A. Mankin, and N. Feamster, "Oblivious DNS: practical privacy for DNS queries," *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 2, pp. 228–244, 2019.
[39] W. Scott, T. Anderson, T. Kohno, and A. Krishnamurthy, "Satellite: Joint analysis of CDNs and network-level interference," in *USENIX Annual Technical Conference*. USENIX, 2016.
[40] S. Siby, M. Juárez, C. Díaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted DNS -> Privacy? A Traffic Analysis Perspective," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.
[41] K. Singh, G. Grover, and V. Bansal, "How India Censors the Web," in *12th ACM Conference on Web Science*, ser. WebSci '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 21–28. [Online]. Available: https://doi.org/10.1145/3394231.3397891
[42] R. N. Staff, "RIPE atlas: A global internet measurement network," *Internet Protocol Journal*, vol. 18, no. 3, 2015.
[43] R. Sundara Raman, L. Evdokimov, E. Wurstrow, J. A. Halderman, and R. Ensafi, "Investigating Large Scale HTTPS Interception in Kazakhstan," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 125–132.
[44] R. Sundara Raman, P. Shenoy, K. Kohls, and R. Ensafi, "Censored Planet: An Internet-Wide, Longitudinal Censorship Observatory," in *In ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

Name	Value
annotations	...
input	dot://dns.quad9.net
probe_asn	AS48716
probe_cc	KZ
resolver_ip	74.125.46.2
resolver_asn	AS15169
default_addrs	"9.9.9.10 2620:fe::10"

TABLE XI
EXAMPLE OF MEASUREMENT METADATA

[45] R. Sundara Raman, A. Stoll, J. Dalek, R. Ramesh, W. Scott, and R. Ensafi, "Measuring the Deployment of Network Censorship Filters at Global Scale," in *Network and Distributed System Security*. The Internet Society, 2020. [Online]. Available: https://censorbib.nymity.ch/pdf/Raman2020a.pdf
[46] M. Trevisan, F. Soro, M. Mellia, I. Drago, and R. Morla, "Does domain name encryption increase users' privacy?" *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 3, pp. 16–22, 2020.
[47] B. VanderSloot, A. McDonald, W. Scott, J. A. Halderman, and R. Ensafi, "Quack: Scalable remote measurement of application-layer censorship," in *USENIX Security Symposium*. USENIX, 2018. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-vandersloot.pdf
[48] S. Yuen, "Becoming a cyber power. china's cybersecurity upgrade and its consequences," *China Perspectives*, vol. 2015, no. 2015/2, pp. 53–58, 2015.

APPENDIX

A. DNSCheck Test List

To run DNSCheck, a researcher needs to prepare a test list consisting of a JSON document entry per line.

Each JSON entry describes a specific DoT/DoH service to measure. The following snippet[9] provides an example:

```
{"annotations":{},"dnscheck":{}, \
  "input":"dot://dns.google"}
```

The `input` key describes what to measure. We use URLs to represent any supported DNS service and protocol.

DoH services are HTTPS services, so we use HTTPS URLs (e.g., `https://doh.powerdns.org`).

We use the `dot` scheme for representing DoT services, e.g., `dot://dns.quad9.net`. We use `853/tcp` when the port is missing. The DNSCheck input parser always ignores the URL path because it has no meaning for DoT.

We use the `udp` scheme for cleartext DNS. We use port `53` when the port is missing. We always ignore the path.

The `annotations` key contains key-value pairs used to annotate measurements. The `dnscheck` key has options changing the behavior of DNSCheck.

B. DNSCheck Data Format

A DNSCheck measurement result contains top-level metadata describing the measurement in general, as well as DNSCheck-specific data. In turn, this consists of data collected during the *bootstrap* and data collected by each *lookup*.

1) Measurement Metadata: Table XI shows a subset of the metadata collected for a typical measurement. We keep track of annotations from the JSON entry. We record the DNS

[9]We use a backslash to indicate that a line continues on the following line.

endpoint	sni	failed_operation	failure	...
9.9.9.10:853/tcp	dns.quad9.net	null	null	...

TABLE XII
EXAMPLE OF LOOKUPS RESULT

endpoint to measure as `input`. We save the country code and the ASN as `probe_cc` and `probe_asn`, respectively. We save the IP used by the system resolver as well as the corresponding ASN. We also track the default addresses specified in the JSON entry as `default_addrs`.

2) Bootstrap Results: We save the input domain and `getaddrinfo`'s return value consisting of either a failure or a list of IPs.

3) Lookups Results: Table XII shows the structure of *lookups* results for a given *endpoint*. In addition to the Table's information, we also save the timing and result of TLS handshakes, `connect`, `read`, and `write` operations.

C. Generating the Test List for DNSCheck

We started from an input list of DoT/DoH services containing 123 DoT/DoH services. The snippet below shows a portion of such a list with three DoT/DoH services:

```
dot://one.one.one.one
dot://1.1.1.1
https://dns.quad9.net/dns-query
...
```

For every input line in such a list, we created one or more JSON entries in the test list. We generated a single JSON entry when the input line contained an IP address. For example, the second line became the following entry:

```
{"input":"dot://1.1.1.1"}
```

Instead, we created three entries when the input line contained a domain name. The first entry describes how to test the specified URL. This entry also contains valid IP addresses for the domain, resolved using Google's public DNS. For example, the first line in the above snippet generated:

```
{"input":"dot://one.one.one.one", \
  "dnscheck":{"default_addrs": \
    "1.1.1.1 1.0.0.1 ..."}}
```

The other two entries resolve the specific domain with Google's and Quad9's public DNS resolvers. These extra entries allow us to check for DNS injection/hijacking. Continuing from the above example, the first line in the snippet generated these extra entries:

```
{"input":"udp://8.8.8.8", \
  "dnscheck":{"domain":"one.one.one.one"}}
{"input":"udp://9.9.9.9", \
  "dnscheck":{"domain":"one.one.one.one"}}
```

Once serialized, the test list consisted of 326 entries for testing 82 domains and 461 TCP/QUIC endpoints.