Exploring Simple Detection Techniques for DNS-over-HTTPS Tunnels

Carmen Kwan[*]
University of Waterloo
Waterloo, Canada
c27kwan@uwaterloo.ca

Paul Janiszewski[*]
University of Waterloo
Waterloo, Canada
pjanisze@uwaterloo.ca

Shela Qiu[*]
University of Waterloo
Waterloo, Canada
s38qiu@uwaterloo.ca

Cathy Wang[*]
University of Waterloo
Waterloo, Canada
c452wang@uwaterloo.ca

Cecylia Bocovich
The Tor Project
Waterloo, Canada
cecylia@torproject.org

ABSTRACT

While DNS tunneling has shown promise as a censorship circumvention technique, it is limited by the plaintext nature of the DNS protocol, which renders it easily detectable to censors. DNS-over-HTTPS (DoH) [16] resolves this detectability obstacle, by encrypting the entire DNS protocol inside HTTPS. DoH tunneling shows promise as a medium for circumvention as its adoption increases in everyday usage, but it may still be vulnerable to flow-based attacks. This paper explores the design space of threshold-based attacks and defences on encrypted DNS tunnels. We identify thresholds separating tunnel traffic from browser-generated DoH traffic using packet size, packet rate, and throughput. We further propose modifications for encrypted DNS tunnels to evade flow-based detection and measure the reduction in usability. Notably, throughput is decreased by at least 27x and page load time is increased by at least 23x. However, despite the cutback in usability, we outline the potential for DNS tunnels to work in conjunction with, and obfuscate the registration traffic of, other anti-censorship tools.

CCS CONCEPTS

• **Social and professional topics → Technology and censorship**.

KEYWORDS

anticensorship, network protocols, classification, DoH

ACM Reference Format:
Carmen Kwan, Paul Janiszewski, Shela Qiu, Cathy Wang, and Cecylia Bocovich. 2021. Exploring Simple Detection Techniques for DNS-over-HTTPS Tunnels. In *Workshop on Free and Open Communications on the Internet (FOCI '21), August 27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3473604.3474563

[*]The authors contribute equally to this paper.

1 INTRODUCTION

In the field of censorship circumvention, DNS tunneling has been proposed as a covert communication channel (e.g. DNS-sly [2]) and a pluggable transport for Tor (e.g. DNSCatProxy [26]). Two limitations of DNS tunneling are its low bandwidth due to the theoretical bounds of the DNS protocol and its lack of encryption, which exposes communication endpoints. Current implementations of DNS tunneling using plain UDP DNS, such as Iodine [6] and dnscat2 [5], are prone to DNS filtering if the identity of the tunneling server is discovered by the censor.

Opportunities. By encrypting the entire DNS protocol inside HTTPS, tunneling using DNS-over-HTTPS (DoH) [16] can defend against traditional DNS circumvention detection techniques that take advantage of analysing plaintext DNS queries. This becomes possible when DoH resolvers sit outside of the censored network, leaving encrypted HTTPS packets between the client and a DoH resolver as the only traffic that censors can inspect. DoH tunnels ultimately hide the contents of the query and the location of the tunnel server from censors. DoH has been implemented in Chromium and Firefox. Furthermore, since October 2019, DoH has been enabled by default in Firefox for new users in the United States [15]. As DoH is increasingly adopted for non-circumvention usage, the collateral damage of blocking DoH traffic is also increasing, providing an opportunity to consider DoH as a means of circumvention. Furthermore, despite being low-bandwidth, there are real world instances where DNS tunneling is a valuable solution, such as Iran's Internet blackout in November 2019 where Padmanabhan et al. [20] showed that communication efforts using DNS were possible.

Challenges and Contributions. DoH tunneling for circumvention is promising because it leverages the popularity of DoH servers. This paper evaluates the censorship resistance of a DNS-over-HTTPS tunnel prototype, dnstt [10], and makes the following contributions:

- We generate and profile web-based DoH traffic characteristics.
- We propose threshold-based attacks based on DoH traffic characteristics that require as few as 300 packets to detect DoH tunneling.
- We show that dnstt's throughput and page load time decrease by 27x and increase by 23x, respectively, when dnstt is modified to defend against our attacks.

Our contributions address some of the research gaps highlighted by Tschantz et al. [21]. Namely, we address the usability impacts of modifying a DoH tunnel to be more censorship resistant. We also study attacks that require fewer computing resources to carry out, focusing on simple threshold-based models rather than more sophisticated, but expensive machine-learning based models.

2 RELATED WORKS

Flow-Based Attacks. Ellens et al. [7] developed a methodology to detect the presence of DNS tunneling by combining flow information with statistical methods. Analysis showed an increase in bytes per flow when DNS tunneling was active, as well as an increase in the average number of bytes per packet over a time interval. We build on these observations to develop attacks that identify individual flows presenting circumvention behaviour using dnstt. Note also that while the above observations were made on DNS UDP flows, our experiments are conducted on TLS flows.

Payload and Traffic Analyses. Farnham and Atlasis [8] showed that DNS tunneling can be detected using payload analysis and traffic analysis techniques. The former focuses on attributes of a single DNS request, while the latter analyzes attributes observed over multiple requests. They showcase detection techniques covered in past research and suggest using both payload analysis and traffic analysis techniques as defense in depth to detect DNS tunneling.

Machine Learning Models. Wang et al. [24] found that a decision-tree trained using entropy-based and packet-heading features performs well when detecting flows generated by Meek, a circumvention tool that tunnels traffic over HTTPS connections to popular cloud load balancers.

Vekshin et al. [22] produced two machine learning models. The first model was able to detect DoH flows amongst other HTTPS flows with 99.9% accuracy. The second model was able to identify which DoH client was used between Chrome, Cloudflared, and Firefox. The models used traffic flows and per-packet-information as inputs to generate their classifications. For the DoH classifier, session duration and average inter-packet delay were the two most important parameters. For the client classifier, the most important parameter was the variance of incoming packet sizes.

MontazeriShatoori et al. [19] developed a two-layer machine learning model to detect DoH tunnels. The first layer of the model separates DoH traffic from non-DoH traffic, and the second layer identifies DoH tunneling. Our approach differs in that we adopt a simple threshold-based attack, focusing only on DoH traffic by isolating flows going to and from DoH resolvers. Moreover, while MontazeriShatoori et al. test their model on flows generated by Iodine, DNS2TCP, and DNScat2, we focus on traffic generated by dnstt.

TLS Fingerprinting. Censors can block encrypted censorship circumvention tools such as Tor by identifying their unique TLS fingerprints based on TLS Client Hello message metadata [1]. Despite using HTTPS to encrypt its data, DoH tunnels could still be distinguished from normal browsers using DoH through a TLS fingerprinting attack. Frolov and Wustrow's application https://tlsfingerprint.io/ analyzes TLS fingerprint rarity within the University of Colorado's network. When dnstt was compiled with golang version go1.15.7 darwin/amd64, this application detected that dnstt's Client Hello messages came from the Go-http-client/1.1 package and was seen in less than 0.01% of all the fingerprints it had inspected.[1] Censors could likely detect dnstt by inspecting TLS sessions communicating with DoH resolvers and filtering for this rare fingerprint. To address this vulnerability, we modified dnstt's client [23] to mimic common TLS fingerprints using Frolov and Wustrow's library, uTLS [13]. Although important, further experiments and discussions about this change are beyond the scope of this paper.

3 DNSTT OVERVIEW

DoH vs DoT. DNS-over-HTTPS (DoH) [16] and DNS-over-TLS (DoT) [18] are the two standards for encrypted DNS. Böttger et al. [4] performed an empirical study of the performance of DoH compared to UDP DNS and DoT. Despite more header overhead with HTTP/2, they found that the resolution time of DoH is similar to DoT when factoring in delayed queries. Böttger et al. argued that the increased security of DoH compared to the minimal performance costs may be why DoH has gained more popularity over DoT. Moreover, DoH is harder to block by default as it operates over the same port as general HTTPS traffic, whereas DoT operates on its own port. As such, this paper focuses on circumvention using DoH instead instead of DoT.

DoH Tunneling. dnstt [10] is an encrypted DNS tunnel prototype built by Fifield to demonstrate the use of a separate reliability layer inside a circumvention transport [11]. DNS tunnels use recursive DNS resolvers as proxies to transfer data between a client and a server. For instance, a client that sends the DNS query DATA.example.com is effectively sending the message DATA to the server at example.com. For DoH tunneling, dnstt runs a local proxy on the client's machine that accepts TCP connections and encodes them into DoH requests. A DoH request is handled by a third-party DoH resolver (e.g., CloudFlare's resolver at `1.1.1.1` or Google's resolver at `8.8.8.8`) outside of the censor's region of control, which forwards the DNS request to a dnstt server. The dnstt server decodes the DNS request back to the original TCP request and forwards it to the intended TCP address. A response then travels back through the tunnel as a TXT response to a DNS request, to be decoded by the dnstt client. Since censors can see only the encrypted DoH packets being sent to a DoH resolver, the final destination of the packet, the tunnel server, is hidden from censors.

Turbo Tunnel. Turbo Tunnel [11], an interior session and reliability design pattern, works independently from the obfuscator of censorship circumvention tools. In many systems, the obfuscation layer provides both blocking resistance and user session management. This presents a challenge, as TCP connection-based tools must restart their session from scratch if the underlying TCP connection is attacked. In DNS tunnel-based systems, packets may be dropped or re-ordered while being transported over UDP, so a variety of reliability schemes have been developed to handle this unreliability. DoH faces both of these problems: it uses a TCP/TLS connection up to the first recursive DoH resolver, after which queries are converted to UDP-based DNS. By implementing a Turbo Tunnel

[1]https://tlsfingerprint.io/id/a91c0644c199823d

Table 1: Distinguishing traffic characteristics between normal DoH traffic and dnstt traffic data distribution

Traffic Characteristic	Direction	Threshold (T)	% normal DoH windows > T	% dnstt windows > T
Avg. Payload Length	Bidirection	120 B	2.0	65
Avg. Payload Length	Incoming	176 B	1.0	55
Avg. Payload Length	Outgoing	70 B	1.0	56
Packet Rate	Bidirection	1, 010 packets/s	0.1	69
Packet Rate	Incoming	616 packets/s	0.1	68
Packet Rate	Outgoing	393 packets/s	0.1	70
Throughput	Bidirection	68, 148 B/s	0.1	73
Throughput	Incoming	42, 909 B/s	0.1	73
Throughput	Outgoing	24, 265 B/s	0.1	72

design, any disruptions to these outer transport layers are independent from the persistent end-to-end inner session layer of a connecting client.

4 DOH TRAFFIC CHARACTERISTICS

To determine potential vulnerabilities from using DoH [16] tunneling, we profiled both circumventor and non-circumventor traffic. The non-circumventor dataset represents a user that uses DoH purely for DNS requests while the circumventor dataset represents a user that uses dnstt for DoH tunneling.

Traffic Generation. We were unable to compile a dataset of DoH traffic captured from real users, due to the anonymization of IP addresses in public HTTPS datasets. As such, we generated a non-circumventor dataset for study. As DoH is a low bandwidth protocol, we are interested in profiling a non-circumventor that generates a heavy amount of DoH traffic. Heavy DoH traffic is more likely to resemble circumventor traffic, allowing us to obtain a higher resolution of the threshold between circumventor and non-circumventor traffic. We generated the non-circumventor dataset by visiting the Alexa[2] Global Top 250 websites and performing packet captures. HTTPS packets were captured by applying a filter for port 443 and DoH packets were identified by filtering for IP addresses associated with the Google DoH resolver (8.8.8.8 and 8.8.4.4). We randomly divided the websites into groups of five for each packet capture (pcap) to achieve greater variance in our dataset and to generate longer DoH sessions. Our Selenium script uses geckodriver v0.29.0, and launches Firefox with DoH enabled using the Google DoH resolver. The script visits each website in the group of five sequentially, waiting for the *performanceTiming.responseEnd*[3] signal from the browser between each visit that indicates the end of the network transfer. Since this signal occurs before the website is fully loaded, the rate at which each of the 5 websites is visited is quicker than regular user browsing, leading to a dataset with a heavier amount of DoH traffic. The DNS cache is cleared and a new browser session is started after each pcap is generated. Each pcap contains on average 1-2 DoH sessions, since a Firefox instance reuses the same DoH sessions to service all of its DNS requests. We repeated this script ten times, generating a total of 500 pcaps, for a non-circumventor dataset of size 242MB.

We generated a circumventor dataset using the same script, except we visit only one website per pcap, and configure Selenium Firefox to use a HTTP proxy with a local instance of the dnstt client. A dnstt server was hosted on Amazon Lightsail in Montreal, Canada, with 512MB RAM and 1 vCPU. The server acted as an authoritative resolver with a 15 byte domain name. Relative to the long DNS queries produced by dnstt, the length of this domain is quite short, and does not impact results analyzing TCP payload lengths. A dnstt client was separately hosted on Amazon Lightsail in Montreal, Canada, with 4GB RAM and 2 vCPUs. The client used the same Google DoH resolver as the non-circumventor dataset to communicate with the server. We ran our script twice for the Alexa Global Top 250 websites, generating a total of 500 pcaps for the circumventor dataset, totalling 5.7GB in size.

Distinguishing Traffic Characteristics. For our analysis, each pcap file has its packets split per session, defined by a unique combination of source IP address, source port, destination IP address, and destination port. Each session was analyzed by incoming, outgoing, and bidirectional traffic over 0.5 second tumbling windows. We chose tumbling windows because they are a relatively cheap and simple solution for keeping state at a large scale. As Tschantz et al. [21] point out, while some (e.g. [3]) believe that the Great Firewall might be employing sophisticated methods such as machine learning, there has not yet been a rigorous study showing machine learning techniques being used in practice. This suggests that it is still important to study cheap, simple solutions that censors might employ.

As censors want to minimize collateral damage while also limiting the opportunity for DoH tunneling, we searched for traffic characteristics and scores that correspond to a high percentile in normal DoH data distribution, but a low percentile in dnstt DoH data distribution. These scores can be used as an approximation for maximum value thresholds to detect circumvention. We tested three traffic characteristics. *Avg. Payload Length* tracks the average length, in bytes, of TCP payload data for non-empty packets in a window. *Packet Rate* tracks the total number of packets detected, divided by the total time elapsed since the beginning of the current window. *Throughput* tracks the total bytes of payload data, divided by the total time elapsed since the beginning of the current window. For packet rate and throughput, we tested scores that correspond to a percentile between 99 and 100 in normal DoH data distribution, in increments of 0.05%. To select the best threshold, we start

[2]https://www.alexa.com/
[3]https://www.w3.org/TR/navigation-timing/#sec-window.performance-attribute

Table 2: Attack performance on Alexa 251-500 dataset, before dnstt modifications

Technique	Direction	Threshold	Accuracy	Recall	Precision	Packets before Detection
Avg. Payload Length	Bidirection	120 B	0.88	0.99	0.80	$2,000 \pm 5,000$
Avg. Payload Length	Incoming	175 B	0.90	0.99	0.83	$2,000 \pm 3,000$
Avg. Payload Length	Outgoing	70 B	0.86	0.89	0.81	$600 \pm 5,000$
Packet Rate	Bidirection	1,100 packets/s	0.94	1.00	0.88	$4,000 \pm 1,000$
Packet Rate	Incoming	620 packets/s	0.90	1.00	0.82	$2,000 \pm 600$
Packet Rate	Outgoing	400 packets/s	0.85	1.00	0.75	$1,000 \pm 300$
Throughput	Bidirection	69,000 B/s	0.91	1.00	0.84	800 ± 200
Throughput	Incoming	43,000 B/s	0.86	1.00	0.77	400 ± 100
Throughput	Outgoing	25,000 B/s	0.93	1.00	0.87	300 ± 100

Table 3: Attack performance on Alexa 251-500 dataset, after dnstt modifications

Technique	Direction	Threshold	Accuracy	Recall	Precision	Packets before Detection
Avg. Payload Length	Bidirection	120 B	0.44	0.00	0.00	N/A
Avg. Payload Length	Incoming	175 B	0.44	0.00	0.00	N/A
Avg. Payload Length	Outgoing	70 B	0.56	0.25	0.56	10 ± 40
Packet Rate	Bidirection	1,100 packets/s	0.94	1.00	0.88	$2,000 \pm 1,000$
Packet Rate	Incoming	620 packets/s	0.90	1.00	0.83	$1,200 \pm 900$
Packet Rate	Outgoing	400 packets/s	0.85	1.00	0.76	700 ± 400
Throughput	Bidirection	69,000 B/s	0.91	1.00	0.84	$1,100 \pm 300$
Throughput	Incoming	43,000 B/s	0.86	1.00	0.77	600 ± 200
Throughput	Outgoing	25,000 B/s	0.93	1.00	0.88	500 ± 100

with the threshold for the 100th percentile and we decrement the percentile (i.e. increase collateral damage) only if the increase in dnstt windows above this new threshold is $\geq 5\%$. In other words, we assume there are at least 100x more non-circumventors than circumventors and that a censor will not increase collateral damage unless at least an equal number of circumventors is affected. For average payload length, we manually picked thresholds by testing scores that correspond to a percentile between 90 and 100 in normal DoH data distribution, in increments of 1%.

Dataset Results and Analysis. Table 1 summarizes the traffic characteristics tested, the threshold identified for each characteristic, and the percentage of normal DoH and dnstt windows from the distribution that were above this threshold. In general, average payload length, packet rate, and throughput are higher in dnstt windows than normal DoH windows. Our results match Ellens et al. [7]'s observations on DNS flows. This is expected as more data are transferred when tunneling webpages than in genuine DoH requests and dnstt optimizes for maximum throughput by default.

A limitation of our analysis presented in Table 1 is that each window is given equal weight in our data distribution and long sessions that contribute more windows can skew the data distribution. As network traffic is inherently noisy and highly variable, our method still offers an adequate approximation for distinguishing between normal DoH traffic and DoH tunneling traffic. The percentages in Table 1 are an initial exploration into the differences between dnstt and regular DoH traffic. We explore the feasibility of turning these characteristics into realistic attacks in Section 5.

5 CIRCUMVENTION DETECTION ACCURACY

Using the session characteristics from Section 4, we developed threshold-based attacks to detect DoH tunneling.

Attack Descriptions. An attack tracks traffic characteristics for each session, updating them each 0.5 second tumbling window. As before, a session is defined by a unique combination of source IP address, source port, destination IP address, and destination port. As soon as a characteristic of a session exceeds the threshold defined by an attack, the session is labelled as circumvention and can be blocked. The thresholds chosen for each attack are based on Table 1, which seek to minimize the amount of DoH traffic present above the threshold. Performance metrics are obtained by comparing the attack's labelling against the ground truth.

Attack Results and Analysis. In order to prevent over-fitted results, we generated a second dataset using the method described in Section 4, this time visiting the Alexa Global Top 251-500 websites. The non-circumventor DoH dataset totalled 269MB, while the circumventor dnstt dataset totalled 7.2GB. Table 2 shows the performance metrics of each attack and the number of packets analysed before correctly detecting a dnstt session. In total, attacks were carried out over $1,146$ bidirectional sessions, $1,100$ incoming sessions, and $1,113$ outgoing sessions. Accuracy, recall, and precision are calculated using standard statistical classification methods. Recall is a measure of detection effectiveness, while precision is a measure of collateral damage. Other than the average payload length attack, all attacks had a 1.0 recall rate. That is, all dnstt sessions were correctly identified. The throughput attack for outgoing

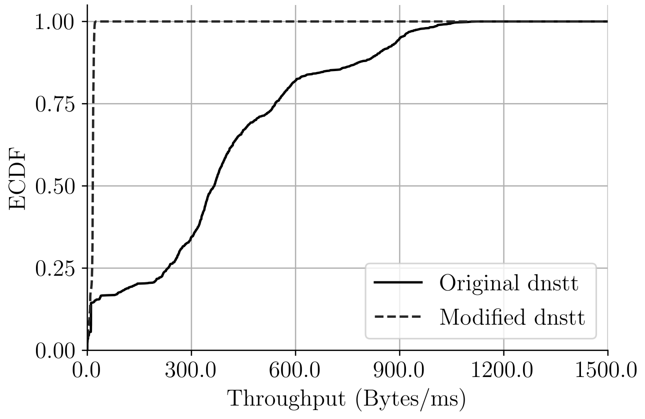

Figure 1: ECDF Throughput

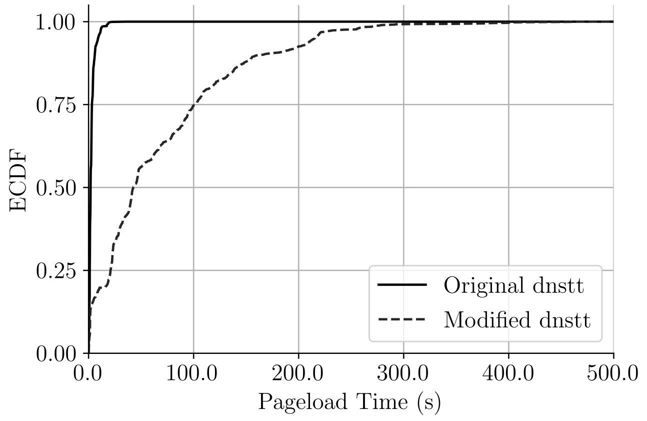

Figure 2: ECDF Page Load

packets needed the fewest number of packets on average before detecting circumvention. In terms of precision, the packet rate attack for bidirectional sessions, as well as the throughput attack for outgoing sessions performed similarly well. In practice, censors might be more interested in maximizing the precision of their attacks than maximizing the recall. Since our normal DoH dataset was generated with a high-traffic user in mind, real-world DoH datasets are expected to have much lighter traffic characteristics than the thresholds studied here. That is, precision is expected to increase with a real-world dataset.

dnstt Traffic Shaping. We modified dnstt to shape its traffic to resemble normal DoH traffic characteristics, to evade the attacks presented above. The dnstt server's effective MTU was set to 100 bytes and the overall packet rate was lowered to 500 packets/s. We changed the server's MTU rather than the client's MTU since the server's packets were often more than 1, 000 bytes larger than the average sizes observed in our DoH dataset. With our modifications, packets can never be sent more frequently or have larger sizes than the limits that we specified. This can open up new attack options as the shape of the generated traffic will have clear features. Houmansadr et al. [17] highlight the importance of eliminating distinguishing patterns in packet sizes and packets rates to properly be unobservable when imitating a protocol. Therefore, when designing packet size limitations and rate limitations into DoH tunneling tools, there should an element of randomness when converging around a desired limit.

Attack Results After Modifications. We generated a circumvention dataset for the above modified version of dnstt, visiting the Alexa 251-500 sites using the same method described in Section 4 to produce 28GB of pcap files. This included 1, 169 bidirectional sessions, 1, 123 incoming sessions, and 1, 136 outgoing sessions. Table 3 shows the performance of our attacks on this dataset. The modified dnstt sessions managed to evade detection on the average payload length attack, for the bidirectional and incoming attacks. Recall and precision were significantly decreased for the outgoing direction as well. For packet rate and throughput attacks, our rate limiting was insufficient to evade detection.

6 USABILITY AFTER MODIFICATIONS

We explore the performance impact of traffic shaping on dnstt [10] to resemble the normal DoH [16] traffic characteristics described in Section 5.

Usability Experiment. To study the performance impact on dnstt when it transmits smaller packets at a lower rate, we modified the MTU and packet rate of dnstt, then compared download performance of the modified dnstt with the original version. Specifically, we compared the page load time and throughput. As mentioned in Section 5, the packet rate was set to 500 packets/s and the server's effective MTU was set to 100 bytes. Both the original and modified versions of dnstt load 100 websites from Cisco's Top 1 Million Umbrella Popularity List[4] using phantomas[5], a performance library that we modified to proxy headless Chromium traffic using dnstt. Each website was loaded 15 times and the throughput and page load time were recorded for each. The dnstt server and client were hosted on the same servers described in Section 4, located in Montreal, Canada. We used Google's DoH resolver, which has the fastest performance, based on Fifield's dnstt download speed tests.[6]

Usability Results and Analysis. Figure 1 shows the ECDF of throughput for unmodified dnstt and modified dnstt. The modified version of dnstt has a maximum throughput of around 25 bytes/ms, while the original version of dnstt can reach throughput of up to 1124 bytes/ms. In fact, with the unmodified version of dnstt, more than 85% of the sites had higher throughput than 25 bytes/ms. Thus, dnstt's throughput is decreased 27x after modifications. Our performance results for unmodified dnstt are faster than those reported by Fifield's performance tests[7], most likely because Fifield's experiments were over a trans-Pacific link. Figure 2 shows the ECDF of page load time for unmodified dnstt and modified dnstt. All the sites could be loaded with the original version of dnstt in under 36 seconds, and on average, a site could be loaded in under 3 ± 3 seconds. However, the modified version of dnstt can take as long as 7 minutes to load a website, with 69 ± 71 seconds being the average page load time. Only about 20% of sites could be loaded with the

[4]https://s3-us-west-1.amazonaws.com/umbrella-static/index.html
[5]https://www.npmjs.com/package/phantomas
[6]https://www.bamsoftware.com/software/dnstt/performance.html
[7]https://www.bamsoftware.com/software/dnstt/performance.html

modified version of dnstt in 36 seconds, the maximum amount of time it takes any site to load with the original version of dnstt. dnstt's page load time is increased 23x after modifications.

Despite the drastic decrease in throughput for our traffic shaping defences, this highly censorship-resistant version of dnstt has a potential use in combination with other anti-censorship tools. Many tools such as Snowflake [9, 14], TapDance [25], and Conjure [12] require a registration or bootstrapping step to connect clients with circumvention proxies. These registration phases require very little bandwidth (e.g., in the case of Snowflake the exchange of WebRTC session description information), but need to be difficult to detect and block. As such, our modifications to dnstt present a strong candidate for obfuscating the registration traffic of other anti-censorship tools.

7 CONCLUSION

dnstt [10] is a prototype of DoH [16] tunneling for censorship circumvention. We conducted an initial exploration of the traffic characteristics of normal DoH traffic outlined in Table 1, and used these characteristics to explore the effectiveness of threshold-based attacks. We found that dnstt sessions can be detected with up to 100% accuracy using throughput and packet rate attacks, and the collateral damage is as low as 12% on a high-traffic DoH dataset. A significant next step would be to perform these attacks on real DoH users's traffic, to obtain a more accurate picture of the collateral damage that would occur in a censored region. Furthermore, we implemented defences for these threshold-based attacks and found that defending against the average payload length attack received the greatest decrease in performance, while the packet rate and throughput attacks continued to perform well. This suggests that stronger rate limiting is required if dnstt were to mimic non-circumventor traffic. Finally, we measured the usability impact of our defences on dnstt and found that the throughput is decreased by at least 27x and the page load time is increased by at least 23x. While dnstt's usability is reduced to defend against our threshold attacks, this modified version of dnstt can be paired with other circumvention tools that require a censor-resistant bootstrapping step. Repeating our experiments while running dnstt with DoT [18] to compare with the results from running dnstt with DoH is an interesting direction for future work.

ACKNOWLEDGMENTS

This work benefited from the use of the CrySP RIPPLE Facility at the University of Waterloo. The authors would like to thank David Fifield, Matthew Lakier, and Sergey Frolov for supporting this project.

REFERENCES

[1] Sadia Afroz and David Fifield. 2015. 'Timeline of Tor Censorship'. http://www1.icsi.berkeley.edu/~sadia/tor_timeline.pdf. Accessed March 2021.

[2] Qurat-Ul-Ann Danyal Akbar, Marcel Flores, and Aleksandar Kuzmanovic. 2016. DNS-sly: Avoiding Censorship through Network Complexity. In *6th USENIX Workshop on Free and Open Communications on the Internet, FOCI '16, Austin, TX, USA, August 8, 2016*, Amir Houmansadr and Prateek Mittal (Eds.). USENIX Association. https://www.usenix.org/conference/foci16/workshop-program/presentation/akbar

[3] Marc Bevand. 2016. My Experience With the Great Firewall of China. https://blog.zorinaq.com/my-experience-with-the-great-firewall-of-china/ Accessed May 2021.

[4] Timm Böttger, Felix Cuadrado, Gianni Antichi, Eder Leão Fernandes, Gareth Tyson, Ignacio Castro, and Steve Uhlig. 2019. An Empirical Study of the Cost of DNS-over-HTTPS. In *Proceedings of the Internet Measurement Conference*. 15–21.

[5] Ron Bowes. 2013. dnscat2. https://github.com/iagox86/dnscat2.

[6] Erik Ekman and Bjorn Andersson. 2006. 'Iodine'. https://code.kryo.se/iodine/. Accessed March 2021.

[7] Wendy Ellens, Piotr Zuraniewski, Anna Sperotto, Harm Schotanus, Michel Mandjes, and Erik Meeuwissen. 2013. Flow-Based Detection of DNS Tunnels. In *Emerging Management Mechanisms for the Future Internet - 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2013, Barcelona, Spain, June 25-28, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7943)*, Guillaume Doyen, Martin Waldburger, Pavel Celeda, Anna Sperotto, and Burkhard Stiller (Eds.). Springer, 124–135. https://doi.org/10.1007/978-3-642-38998-6_16

[8] Greg Farnham and Antonios Atlasis. 2013. Detecting DNS tunneling. *SANS Institute InfoSec Reading Room* 9 (2013), 1–32.

[9] David Fifield. 2017. *Threat modeling and circumvention of Internet censorship*. Ph.D. Dissertation. University of California, Berkeley.

[10] David Fifield. 2020. dnstt. https://www.bamsoftware.com/software/dnstt/index.html

[11] David Fifield. 2020. Turbo Tunnel, a good way to design censorship circumvention protocols. In *10th USENIX Workshop on Free and Open Communications on the Internet, FOCI 2020, August 11, 2020*, Roya Ensafi and Hans Klein (Eds.). USENIX Association. https://www.usenix.org/conference/foci20/presentation/fifield

[12] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. 2019. Conjure: Summoning Proxies from Unused Address Space. In *Computer and Communications Security*. ACM. https://jhalderm.com/pub/papers/conjure-ccs19.pdf

[13] Sergey Frolov and Eric Wustrow. 2019. The use of TLS in Censorship Circumvention. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/the-use-of-tls-in-censorship-circumvention/

[14] Serene Han. 2011. Snowflake Technical Overview. https://keroserene.net/snowflake/technical. [Online; accessed 8-June-2018].

[15] Firefox Help. 2019. DNS-over-HTTPS (DoH) FAQs. https://support.mozilla.org/en-US/kb/dns-over-https-doh-faqs Accessed April 2021.

[16] Paul Hoffman and Patrick McManus. 2018. *DNS Queries over HTTPS (DoH)*. RFC 8484. IETF Tools. https://tools.ietf.org/html/rfc8484

[17] A. Houmansadr, C. Brubaker, and V. Shmatikov. 2013. The Parrot Is Dead: Observing Unobservable Network Communications. In *2013 IEEE Symposium on Security and Privacy*. 65–79. https://doi.org/10.1109/SP.2013.14

[18] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS)*. RFC 7858. IETF Tools. https://tools.ietf.org/html/rfc7858

[19] Mohammadreza MontazeriShatoori, Logan Davidson, Gurdip Kaur, and Arash Habibi Lashkari. 2020. Detection of DoH Tunnels using Time-series Classification of Encrypted Traffic. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 63–70. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026

[20] Ramakrishna Padmanabhan, Alberto Dainotti, Nima Fatemi, Arturo Filastò, Maria Xynou, and Simone Basso. 2019. Iran's nation-wide Internet blackout: Measurement data and technical observations. https://ooni.org/post/2019-iran-internet-blackout/

[21] Michael Carl Tschantz, Sadia Afroz, anonymous, and Vern Paxson. 2016. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 914–933. https://doi.org/10.1109/SP.2016.59

[22] Dmitrii Vekshin, Karel Hynek, and Tomas Cejka. 2020. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In *Proceedings of the 15th International Conference on Availability, Reliability and Security* (Virtual Event, Ireland) *(ARES '20)*. Association for Computing Machinery, New York, NY, USA, Article 87, 8 pages. https://doi.org/10.1145/3407023.3409192

[23] Cathy Wang, Paul Janiszewski, Shela Qiu, and Carmen Kwan. 2021. dnstt-uTLS Fork. https://github.com/pjanisze/dnstt-uTLS.

[24] Liang Wang, Kevin P. Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. 2015. Seeing through Network-Protocol Obfuscation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 57–69. https://doi.org/10.1145/2810103.2813715

[25] Eric Wustrow, Colleen M. Swanson, and J. Alex Halderman. 2014. TapDance: End-to-middle Anticensorship Without Flow Blocking. In *23rd USENIX Security Symposium* (San Diego, CA). 159–174. http://dl.acm.org/citation.cfm?id=2671225.2671236

[26] Irvan Zhan. 2015. DNSCatProxy. https://github.com/izhan/dnstun_pt.