DNS over HTTPS (DoH) vs DNS over TLS (DoT): Performance and Privacy

Tejas Lohia
Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
23110335
tejas.lohia@iitgn.ac.in

Umang Shikarvar
Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
23110301
umang.shikarvar@iitgn.ac.in

Tanishq Bhushan Chaudhari
Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
23110329
tanishq.chaudhari@iitgn.ac.in

Mohit
Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
23110307
mohit.23110307@iitgn.ac.in

Zainab Kapadia
Computer Science and Engineering
Indian Institute of Technology, Gandhinagar
23110373
zainab.kapadia@iitgn.ac.in

***Abstract*— One of the most integral parts of Internet is the resolution of names of the Domain Name system which translates the readable domain names into the IP addresses. Usually when a human-readable name is queried to a browser, it is resolved by querying it to the DNS resolver. This query is unencrypted, thus leaving user vulnerable to eavesdropping and tampering with the packet. It has been researched that this can lead to tracking of the user's browsing history thus making the user vulnerable. This vulnerability can be resolved by the introduction of two additional protocols. DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT). Instead of sending queries unencrypted, these protocols establish encrypted connection between users and resolvers providing security and better reliability than traditional DNS resolution which is also called Do53[1]. In this report we compare the performance and privacy of these protocols.**

Keywords—Domain, IP address, query, DNS, encrypted, eavesdropping, tampering, HTTPS, TLS, performance, privacy

I. Introduction

The Domain Name system (DNS) is a distributed, hierarchical naming protocol that maps domain names to corresponding IP addresses, enabling host identification within IP-based networks. DNS functions through a client–server model involving stub resolvers, recursive resolvers, and authoritative name servers. A DNS query typically originates from a client application and is forwarded by the stub resolver to a recursive resolver, which iteratively queries the hierarchy of authoritative servers; root, top-level domain (TLD), and domain-level, to obtain the final resource record. Communication is generally conducted over UDP port 53 for efficiency. DNS messages are encoded using a well-defined binary format consisting of a header, question, answer, authority, and additional sections. Each resource record includes a name, type, class, time-to-live (TTL), and data field. While the protocol supports multiple record types (e.g., A, AAAA, MX, NS, TXT), the resolution process is inherently plaintext, lacking confidentiality and integrity.

Recently there has been research exploring the exploitation using the unencrypted DNS query which could be used to reveal the user identity and track the history of the user. As a result, protocols are being developed and deployed to make the DNS query resolution secure. Two protocols DNS-over-HTTPS (DoT) and DNS-over-TLS (DoT) have been developed. Both protocols ensure an encrypted and reliable connection between the client and DNS resolver over the TCP protocol. This encryption ensures that the requests can't be eavesdropped over a public network by sending the requests after a secure encryption to the resolver.

These protocols use TCP protocol at transport later as well additional handshakes for encryption thus adding performance overhead.

II. Background

Due to the original implementation of the DNS over port 53 and it is using UDP protocol, there were several privacy issues which were not considered which intensified in the recent internet era.

To counter the privacy issues in 2016, the DNS over TLS was introduced [3]. It introduced the query resolution over the TLS protocol which meant that it changed the underlying UDP protocol to TCP protocol. Once the connection has been established all queries are encrypted by the transported using TCP protocol.

In 2018, Hoffman et al. proposed DNS-over-HTTPS to prevent security of DNS responses [4]. This protocol uses HTTP as the transport protocol.

III. DNS-over-TLS (DoT)

DNS-over-TLS (DoT) is a security enhancement to the traditional DNS protocol that encapsulates DNS queries and responses within a Transport Layer Security (TLS) session. Operating over TCP port 853, DoT ensures that the communication between the client (stub resolver) and the recursive resolver is encrypted and authenticated, thereby preventing passive eavesdropping and on-path manipulation of DNS traffic. By leveraging the TLS handshake mechanism, DoT establishes a secure channel that maintains confidentiality and integrity of DNS data while preserving the

original DNS message format. Although DoT adds minimal latency due to the TLS negotiation, it significantly improves privacy by concealing DNS metadata from intermediaries, making it especially relevant in environments where network-level monitoring or censorship is a concern.

IV. DNS-OVER-HTTPS (DOH)

DNS-over-HTTPS (DoH) is an encrypted DNS protocol that transmits DNS queries and responses over the Hypertext Transfer Protocol Secure (HTTPS), typically using TCP port 443. By encapsulating DNS messages within HTTPS traffic, DoH provides confidentiality and integrity through the TLS encryption already inherent to HTTPS. This design not only prevents intermediaries from intercepting or modifying DNS queries but also makes DNS traffic indistinguishable from regular web traffic, thereby reducing the effectiveness of network-level blocking or filtering. Unlike DoT, which operates on a dedicated port, DoH integrates seamlessly with existing web infrastructure and can leverage persistent HTTP/2 or HTTP/3 connections for improved performance. However, this tight integration with application-layer protocols can also complicate traffic management and caching at the network level.

V. MEASURING PERFORMANCE

UDPs tend to fail more on the lossy network, thus we measure the performance for two sets of websites. We got the list of websites based on their popularity from tranco. From this we take the top 50 websites and 50 websites from rank 3000 till 3049 (Referred as bottom 50). In the second set of websites, we assume that the network is relatively choppy, thus we expect the TCP based protocols to perform better.

To set the baseline, we resolve these two set of queries over these three methods to measure their performance. In this

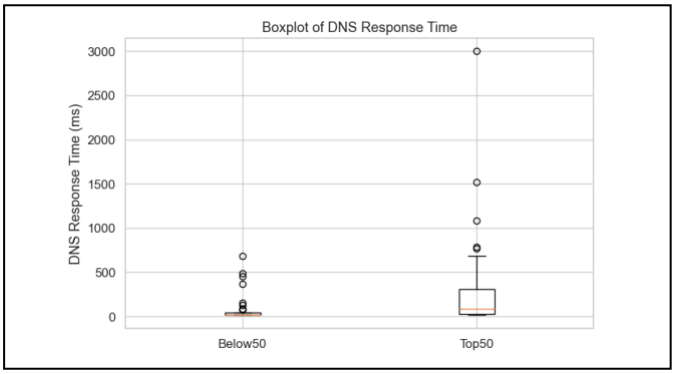


Fig1. Response time for Top50 and Bottom 50 websites for Do53

method we setup a simple testbench. Three codes are initialized for this querying the two set of queries and logging the status, rcode, rtt_ms, bytes_out and bytes_in. For this we query the google server at 8.8.8.8 over all the three protocols and log the details.

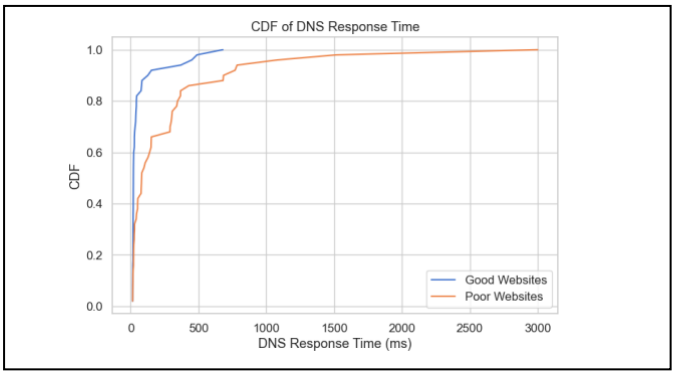


Fig2: CDF of DNS response time for Do53

For the bottom 50 websites, only one of the query failed in case of do53.

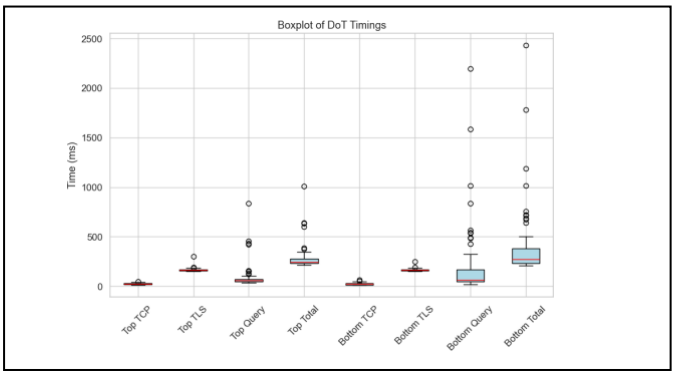


Fig3: Response time for Top50 and Bottom 50 websites for DoT

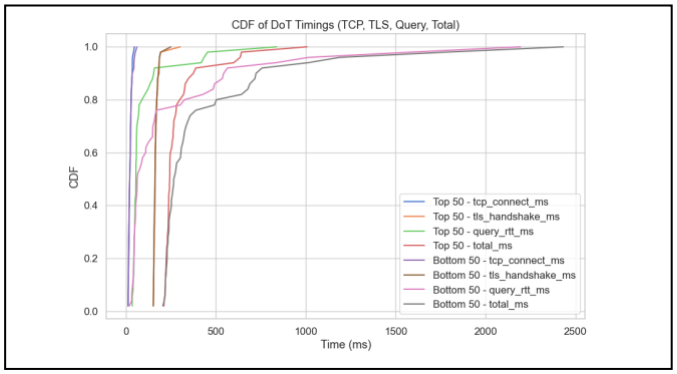


Fig4: CDF of DNS response time for DoT

From the graph it can be seen than the CDF function tends towards right as compared to Do53 for Top50 queries. We use CDF because it captures variability in a better way as compared to simple averages and plots.

Interestingly, for the bottom 50 websites, it could be observed that the CDF has shifted left to Do53 for Bottom 50 queries.

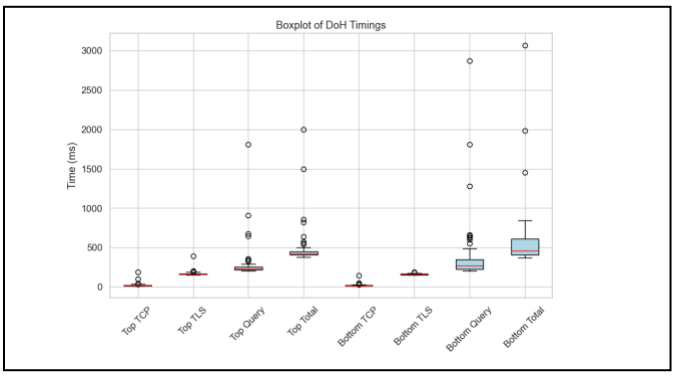


FIG5. RESPONSE TIME FOR TOP50 AND BOTTOM 50 WEBSITES FOR DOH

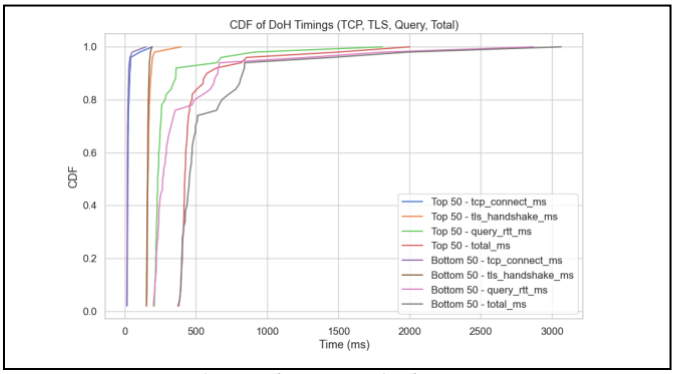


Fig6: CDF of DNS response time for DoH

VI. COMPARISONS

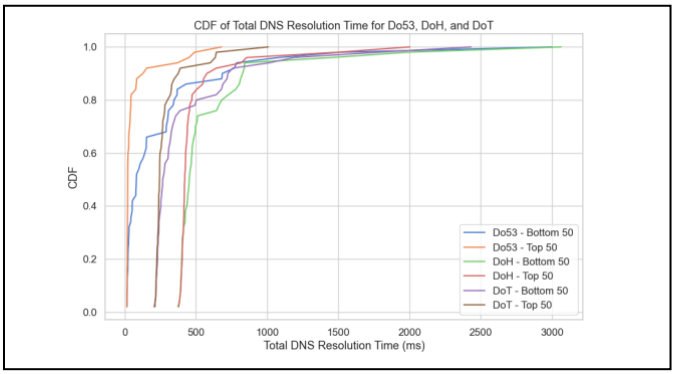


Fig7: CDF of DNS response time for three protocols.

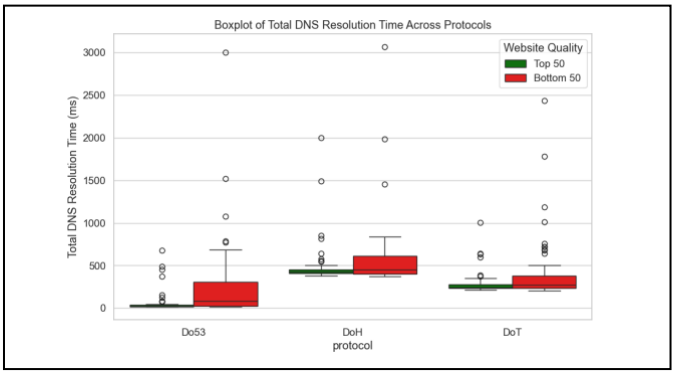


Fig8: Response time for three protocols.

	Do53_T	Do53_B	DoT_T	DoT_B	DoH_T	DoH_B
Average Time(ms)	69.375	220.095	293.077	433.077	503.616	603.894
Success/50	50	49	50	50	50	50

T indicates Top 50 websites, while B are from 3000-3050.The outcomes depict the overhead added because of the various handshakes involved for DoT and DoH. Another important outcome is that in the case of Bottom 50 websites, one of the queries failed to get resolved, which was handled by the other two reliable protocols.

Furthermore, the results clearly show that Do53 remains the fastest option due to its lightweight, connectionless nature, while DoH, though slower, provides maximum security and privacy. The analysis also highlights that website responsiveness plays a crucial role in DNS performance, with less reliable sites exhibiting higher latency across all protocols. These findings emphasize the trade-offs between speed and security when selecting a DNS protocol for practical deployments.

VII. EXPERIMENT SETUP FOR LATENCY

Environment uses two different PCs to test the protocols. Machines: MacOS machines were used throughout the experiment connected through WiFi. All the codes ran python 3/10+ for client and server codes. One of the PC runs the client-side proxy for the browser to encrypt the queries according to the appropriate protocol. Other PC decrypts the DNS messaged wrapped in the protocols and use a custom resolver to query the DNS to the root servers and ultimately solve it iteratively. Iperf tools was also used to simulate realtime simulations in the environment.

Server implements a DNS-over-TLS (DoT) server that handles secure DNS queries over TCP. It listens on a specified IP and port and uses a server certificate, private key, and CA certificate to establish encrypted connections with clients. The server simulates a TLS 1.2 handshake, including ClientHello, ServerHello, Certificate exchange, ClientKeyExchange, ChangeCipherSpec, and Finished messages. Each client connection is processed in a separate thread, allowing concurrent query handling. Encrypted DNS queries are decrypted, resolved using a custom iterative resolver, and the responses are encrypted and sent back to the client. Thus, the server provides a realistic and controlled environment for studying the latency, reliability, and security trade-offs of DNS-over-TLS in comparison with other DNS protocols.

Client implements a DNS-over-TLS (DoT) client compatible with the custom DNS server. It establishes a secure TCP connection to the server, performs a TLS 1.2 handshake, and exchanges encrypted DNS queries and responses. The client constructs a ClientHello message with a random nonce and receives the server's ServerHello and certificate messages. It verifies the server certificate against a trusted CA certificate to ensure authenticity.

Once the handshake is complete, the client derives encryption keys, initialization vectors, and sequence numbers using the TLS pseudo-random function (PRF). It then sends an encrypted Finished message to the server and verifies the server's Finished message to confirm handshake integrity.

Finally, the client sends an encrypted DNS query and receives the response securely over the established TLS channel. The DNS response is decrypted and parsed using dnslib, allowing the client to extract resolution results while maintaining confidentiality and integrity.

VIII. Testing and Results

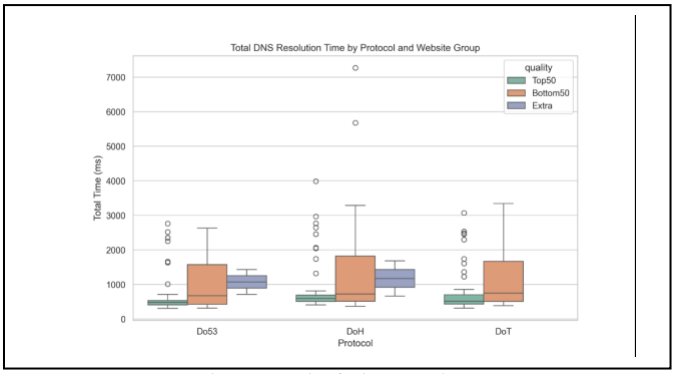
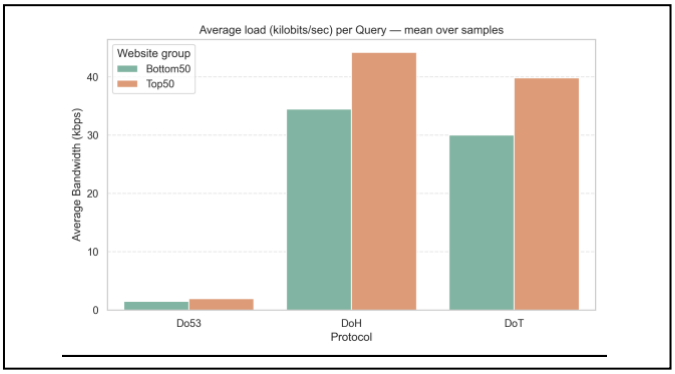


Fig9: Response time for three protocols.

Fig10: Average throughput for all the protocols



	Do53T	Do53B	DoT_T	DoT_B	DoH_T	DoH_B
Tcphandshake	0.00	0.00	55.20	51.01	69.33	25.43
Tls_handshake	0.00	0.00	37.22	41.05	34.41	40.31
Query_time	753.99	997.06	750.25	1071.65	821.87	1300.13
Total_time	753.99	997.06	843.09	1164.10	926.08	1366.27
Bytes_sent	34.76	33.70	881.56	879.10	1236.52	1261.40
Bytes_recv	87.04	89.04	1995.88	2013.29	2188.36	2202.04
Total_bytes	122.24	122.74	2877.44	2892.39	3451.88	3463.44

This table denotes the average value recorded for top 50 and bottom 50 queries. All the temporal values are in milliseconds. T denotes top 50 websites on Tranco while B denotes websites from rank 3000-3050.

The performance analysis reveals lot of details about the overheads and the overall performance of the protocols as well as some of the numbers contradict the expected results.

Time required for TCP handshake is zero for Do53 protocol, while for the DoT is around 55.20 for Top 50 websites and 51.01 for Bottom 50 websites. In the case of DoH protocol, TCP handshake had an average time of 69.33 for top 50 and 25.43 for Bottom 50 websites. These results don't depend upon the websites being popular, these results might just be because of query size difference, or client scheduling.

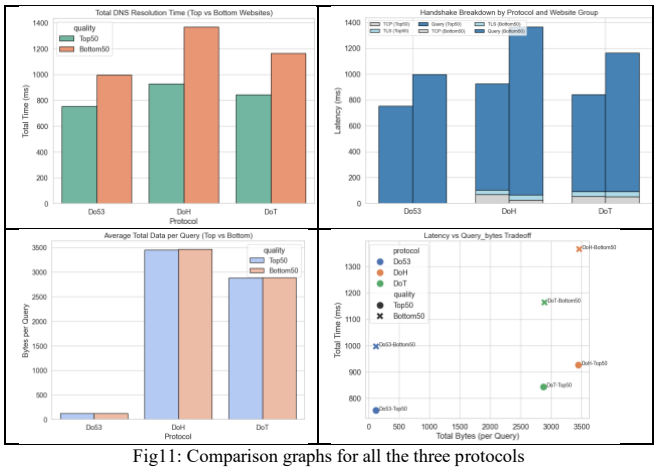


Fig11: Comparison graphs for all the three protocols

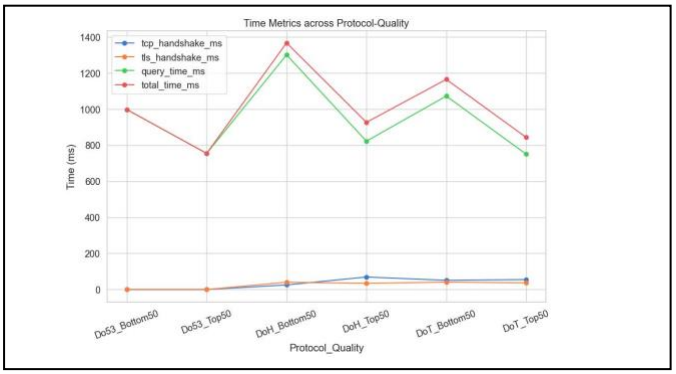


Fig12: Time distribution across protocols and websites

For TLS Handshake the values are zero for Do53, as there is no handshake there, while in case of DoH averaged values are 34.41 and 40.31 for Top50 and Bottom50 respectively. For DoT the values are 37.22 and 41.05 for Top50 and Bottom50 respectively (in ms). These values are quite close and they don't depend on the bottom 50 or top 50 websites as expected, as this handshakes happen only between the client and server.

Query Time logs the time between the query sent from the client and the response received to the client till IP extraction. This metric shows significant differences between the time required for Top50 and Bottom 50 websites for each of the protocol. In the case of Do53, query time is least as expected. For TLS, time is slightly greater than Do53 which deviates slightly from the expected results which predicted it to be significantly greater than Do53. DoH has recorded maximum time for both Top50 and Bottom 50 websites with significant increase for Bottom 50 websites.

DoT takes slightly more time than Do53 mainly due to the encryption and decryption overhead of TLS framing during query exchange. In contrast, DoH records much higher query times because the DNS messages are encapsulated inside HTTP requests, adding extra layers of processing such as HTTP header creation, parsing, and stream handling. Additionally, DoH resolvers often operate over general-

purpose web servers, which are not optimized for low-latency DNS lookups, further increasing delay—especially for less popular websites where cache misses are more frequent. Similar behavior could be seen at Total time required.

The total bytes exchanged show a stark contrast across protocols. For Do53, the total bytes are around 122–123 bytes for both Top50 and Bottom50 websites, reflecting the minimal overhead of plain DNS over UDP. DoT increases this to approximately 2877–2892 bytes, due to the TLS encryption overhead. DoH, however, exhibits the highest byte usage, reaching around 3450–3463 bytes, which is more than 10× the Do53 size. This extreme increase is caused by the HTTP encapsulation and additional headers in DoH queries and responses, in addition to TLS encryption, highlighting the substantial overhead required for privacy and integrity in DoH compared to traditional DNS and DoT.

IX. Page Load Latency

Another important metric used to evaluate the performance of DNS protocols is page loading latency. This metric measures the total time taken for a webpage to fully load in the browser after the initial DNS resolution. It captures the combined effects of DNS lookup delays, TCP or TLS handshakes, and the establishment of secure connections required to fetch webpage resources. Protocols such as DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) introduce additional encryption overheads during connection setup, which may increase the overall page load time compared to traditional DNS-over-UDP (Do53). However, the impact of these overheads can vary based on network conditions, browser caching, and the efficiency of connection reuse mechanisms such as persistent HTTPS sessions. Therefore, analyzing page loading latency provides a holistic view of user-perceived performance and helps determine the real-world feasibility of deploying encrypted DNS protocols.

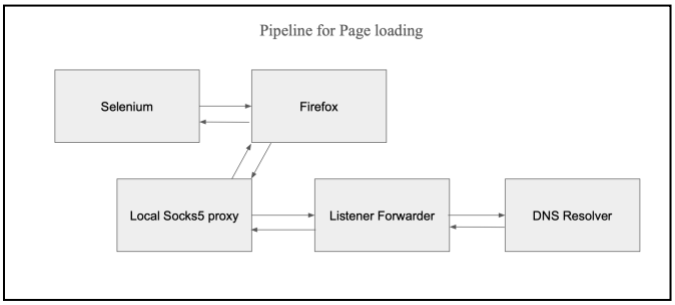


Fig11: Pipeline for Page loading over all the protocols

To accurately measure the page loading latency across different DNS protocols, a controlled browser-based testing architecture was implemented. The setup employed Selenium WebDriver to automate page load operations over ten websites within the Firefox browser. Timing data was collected through the browser's built-in performance measurement interfaces, which capture various navigation and resource loading events. However, when a custom DNS resolver or proxy is used, the browser itself does not perform the DNS resolution and therefore does not record the DNS lookup time in its internal metrics. Consequently, the analysis focuses on total page load time as the primary metric, providing a comprehensive representation of end-to-end user experience across DNS, DoT, and DoH configurations. All DNS queries originating from Firefox were directed through a local SOCKS proxy layer (local_socks5_proxy.py), which served as an intermediary to strictly enforce the routing of DNS traffic through the experimental pipeline and exclude the traffic from MacOS which would have come if we reconfigured the system settings and resolvers.

The proxy layer was designed to forward all outgoing DNS resolution requests exclusively to three custom DNS client modules (listen_forward.py). This forwarder implemented the logic for different DNS transport protocols. Depending on the protocol under test, these files encapsulated and transmitted the DNS queries to the corresponding DNS server implementation (dns_server.py), which handled the actual resolution process and returned the appropriate responses.

This layered design ensured isolation between browser-generated traffic and system-level DNS operations, thereby preventing leakage of unencrypted queries through the default system resolver. It also allowed precise control over the end-to-end query path, enabling consistent measurement of total page loading latency while maintaining protocol-level fidelity and experimental repeatability.

X. Results and Anaysis

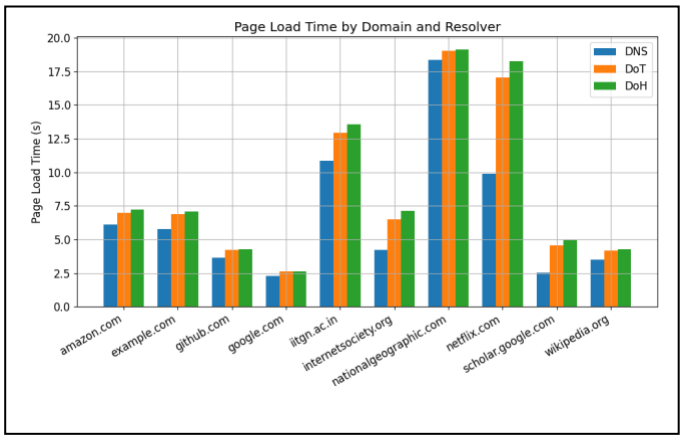


Fig12: Pipeline for Page loading over all the protocols

The above graph illustrates the comparison of page load times across ten domains using DNS, DoT, and DoH resolvers. As expected, the results consistently followed the trend DNS < DoT < DoH, which aligns with our expectations; traditional DNS (UDP-based) introduces the least latency, while encrypted protocols (DoT and DoH) add additional handshake and encryption overhead. Interestingly, this trend was observed uniformly across all tested domains, confirming the reliability of the pattern.

While the overall differences between the three resolvers appear moderate; because total page load time includes other contributing factors such as content download, rendering, and JavaScript execution; it is also possible for these differences to become amplified on pages making multiple DNS lookups. Modern websites typically fetch numerous resources (scripts, ads, images, trackers, etc.) hosted on various domains, which can compound the resolver delay when using DoT or DoH. Therefore, although the encryption overhead is relatively

small per lookup, its cumulative impact can lead to perceptibly higher load times on resource-intensive pages. Overall, the results confirm our expectation of the DNS < DoT < DoH latency hierarchy.

XI. Privacy Metric

The performances tradeoff against the security enhancements have not yet been properly explored. An early research by Firefox found that DNS resolution over HTTPs was only marginally slower than the conventional DNS resolver over port 53[2].

While evaluating the performance of DNS protocols such as Do53, DoT, and DoH, it is equally important to assess their robustness against privacy threats.

One of the most critical threats in DNS communication is the Man-in-the-Middle (MitM) attack, which exploits the lack of encryption or improper authentication between clients and DNS resolvers. This section presents the privacy analysis of the tested DNS protocols under simulated conditions to evaluate the extent of data confidentiality and integrity provided by each protocol.

In this setup, a third person which is substituted by a hop device intercepts the communication between a client and the DNS resolver. This interception allows the attacker to eavesdrop on DNS queries, infer user browsing behavior, or inject spoofed responses to redirect traffic to malicious domains. Traditional DNS (Do53), which relies on plaintext UDP/TCP transport, is inherently vulnerable to such interception and spoofing attacks, as query and response packets can be captured or read with minimal effort.

To analyze this vulnerability, we design an experimental setup where a malicious node is placed between the client (Firefox via proxy) and the DNS resolver. The adversary listens on the communication channel and attempts to extract or modify DNS queries. For encrypted protocols such as DNS-over-TLS (DoT) and DNS-over-HTTPS (DoH), the attacker's ability to observe or tamper with queries is evaluated to determine the effectiveness of their cryptographic protection mechanisms.

XII. TestBed Setup

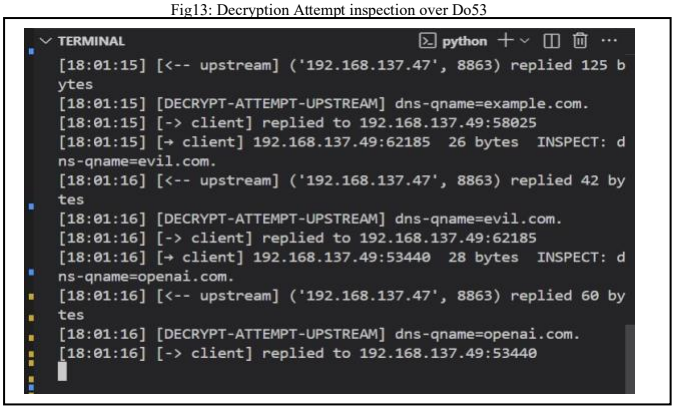
The test environment used three machines on a single local hotspot created by the forwarder (Windows). Network addresses used in the experiments: the *client* was 192.168.137.49, the *forwarder/hotspot* (traffic capture point) was 192.168.137.1, and the *resolver* (custom DNS server running on Mac) was 192.168.137.47. The forwarder acted as the network gateway: both client and resolver connected to the forwarder's hotspot so every client↔resolver packet passed through the forwarder. We captured traffic on the forwarder interface with Wireshark and ran an inspection program on the forwarder that forwarded packets unchanged while attempting to parse and decrypt TLS application records where possible. Pipeline for each query:

1. Client (192.168.137.49) issues a DNS request over the chosen transport.
2. The request travels to the forwarder (192.168.137.1), which forwards it to the resolver (192.168.137.47) and returns the reply along the same path.
3. The forwarder records the traffic with Wireshark and logs inspection results.

XIII. Analysis and Results

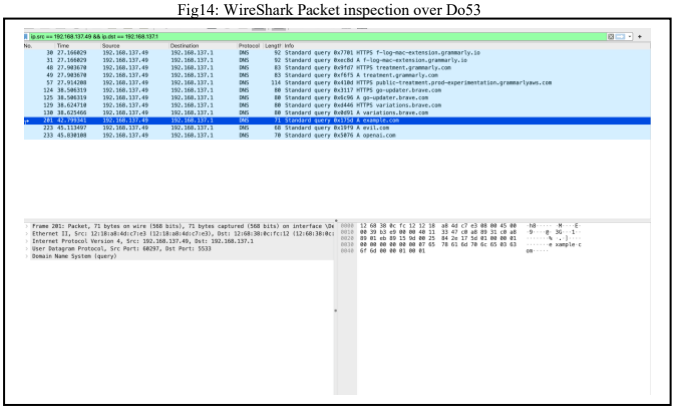
Wireshark was inspected with the following results:
DNS over UDP (Do53, port 53): DNS queries and responses are sent as plaintext UDP payloads. The client QNAME (queried domain) is visible directly in the DNS packet. In Wireshark captured on 192.168.137.1, these packets are decoded as DNS and the QNAME field appears in the packet list. DNS over UDP is trivially readable and loggable by any device on the path.

Fig13: Decryption Attempt inspection over Do53



For DoT, the client and resolver perform a TLS handshake first (ClientHello, ServerHello, Certificate, ChangeCipherSpec, Encrypted Handshake), then the client sends the DNS query inside a TLS Application Data record. The DNS query bytes are therefore present inside Application Data records, but they are encrypted. In Wireshark captures on the forwarder these appear as Application Data; the handshake frames do not

Fig14: WireShark Packet inspection over Do53



contain the DNS query. Thus, the forwarder can see TLS metadata (record sizes, timings, IPs, ports) but not the query contents without the session keys. The *first* Application Data record after the handshake is the one that carries the encrypted DNS query. In the following image we open packet

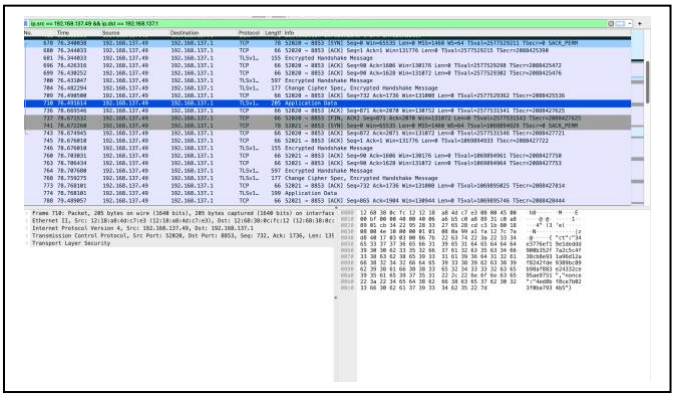


Fig15: WireShark Packet inspection over DoT

During the DoH test, the client (192.168.137.49) sent its DNS queries to the resolver (192.168.137.47) through the forwarder (192.168.137.1) using HTTPS on port 8453. On the forwarder, Wireshark captured these packets but displayed them under the TLS protocol instead of HTTPS.

The capture showed multiple TLS handshake packets followed by Application Data frames. The actual DNS queries and responses were not visible anywhere in the packet details. There was no HTTP POST or any domain name field shown because both the HTTP request and the DNS message were fully encrypted inside these Application Data packets. In short, only the packet direction, size, and timing were visible on the forwarder, the DNS domains and HTTP content could not be read.

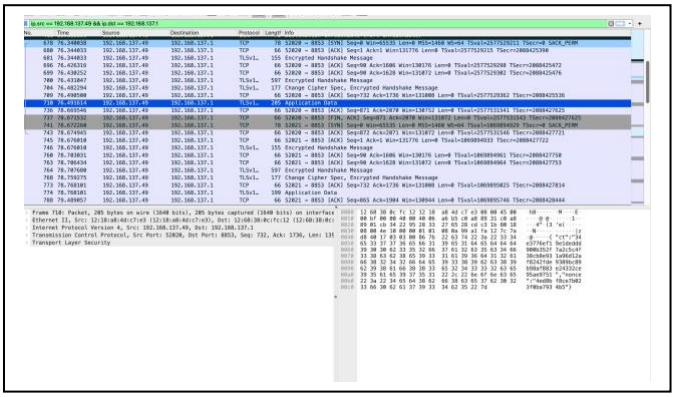


Fig16: WireShark Packet inspection over DoH

To test for the privacy, a decryption was attempted over both the protocols, DoT and DoH.

On the forwarder we implemented passive inspection that: reads TLS record-layer headers, extracts Application Data payloads, attempts to parse those payloads as the JSON framing used in our custom testbed ({"nonce":"...", "ct":"..."}) and tries AES-GCM decryption with a small list of candidate keys.

XIV. Results and Conclusion

In this, we analyzed wiresharks over Do53, DoT and DoH Each worked differently in terms of how the packets appeared in Wireshark and what information could be seen by someone monitoring the network.

In the first case, DNS over UDP, the client sent DNS queries for domains such as example.com, evil.com, and openai.com to the forwarder listening on port 5533. The forwarder then forwarded these queries to the upstream resolver on port 8863. All communication took place in plaintext. In Wireshark every packet was automatically decoded as DNS, showing complete details like the queried domain name, query type, transaction ID, and the response IP address. Anyone capturing packets along this path could easily read which domains were being requested and what IP addresses were returned. There was no privacy or encryption at all; both the content and metadata were fully exposed.

In the second case, DNS over TLS (DoT), the communication started with a TLS handshake between the client and the forwarder on port 8853. The handshake included packets like Client Hello, Server Hello, Certificate, Change Cipher Spec, and encrypted handshake messages. Only after this negotiation was complete did the client send the actual DNS query. This query was placed inside a TLS Application Data record, meaning the DNS bytes were encrypted before being transmitted. In Wireshark, the initial handshake packets were visible and identified as TLS, but the DNS queries themselves appeared only as opaque “Application Data” frames. The forwarder could see that a DNS query was being sent (based on packet size and timing), but it could not read which domain was requested without the TLS session keys. However, the forwarder itself, after decrypting locally, still sent the query upstream to the resolver on UDP port 8863 in plaintext. Therefore, encryption protected the communication only between the client and forwarder; the forwarder-to-upstream hop remained visible and unprotected.

In the third case, DNS over HTTPS (DoH), the DNS query was wrapped inside an HTTPS request. The client performed an HTTPS handshake identical to any other web connection, establishing a secure TLS session. Once the TLS tunnel was active, the DNS query was placed as the body of an HTTPS POST or GET request and transmitted entirely inside encrypted Application Data. In Wireshark, this appeared the same as normal HTTPS traffic — no DNS decoding, no visible domains, only the TLS handshake followed by encrypted Application Data packets. Here, neither the forwarder nor anyone else on the path could see the actual domain names being resolved unless they decrypted the HTTPS traffic with the session keys.

From these observations, the overall conclusion about privacy is clear. Plain DNS over UDP provides no confidentiality at all; both the domains and responses are visible to anyone capturing packets. DNS over TLS hides the query contents from outsiders on the path but still exposes them to the forwarder and any unencrypted upstream communication. DNS over HTTPS provides the highest privacy among the three, fully encrypting both the DNS content and transport layer so that captured packets reveal nothing about the queried domains. In short, as encryption layers increase from UDP → TLS → HTTPS, the visibility of DNS information decreases, and user privacy improves.

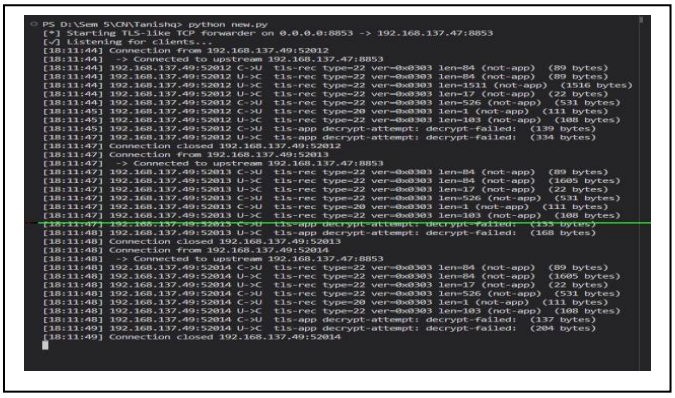


Fig17: Decryption attempt when the underlying protocol is DoT

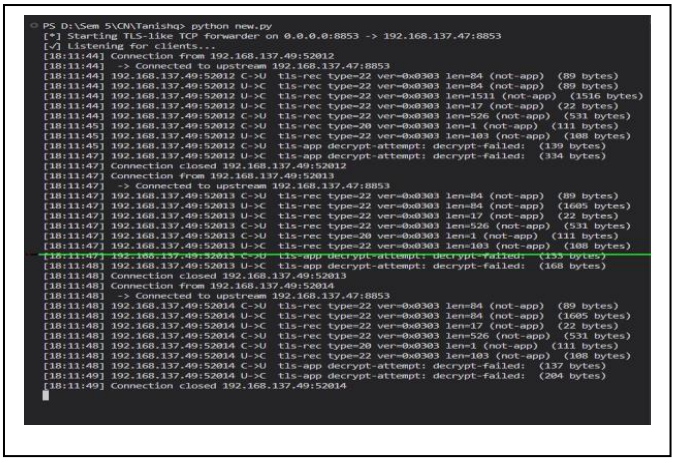


Fig18: Decryption attempt when the underlying protocol is DoH

From the images it could be seen that the decryption has failed in the case of DoT and DoH.

When the forwarder did **not** possess the actual client/server write keys, all decryption attempts failed: the Application Data remained ciphertext and no QNAME was recovered. The forwarder logged record metadata and the decryption failure reasons.

XV. Conclusion

In this paper, we investigated the DNS-over-HTTPS and DNS-over-TLS (DoT) with traditional DNS. We investigated these protocols over various testbed environments and tested them over various metrics to find the tradeoff between performance and security. The protocols are investigated over popular websites (*Top 50*) and not-so-popular websites to check the differences, as it is expected that not-so-popular websites might have lossy network, where the encrypted protocols might outperform Do53 both over performance and privacy due to underlying TCP protocols and encryption. It was discovered that although the DoT and DoH had a significant shift in the CDF of DNS server time over top50 as compared to Do53, this shift was reduced in the case of bottom 50, thus indicating that in the lossy networks DoT and DoH have advantaged of using TCP connection. In Raw performance, Do53 was fast as compared to DoH and DoT in all the testbed environments, because of involvement of several overhead components. DoH and DoT ensured privacy against the instances of eavesdropping as it was tested in the middle router decryption performed. The packets were encrypted thus making them invulnerable. Additional metric used to measure the performance was page loading time which was performed using the Firefox browser and selenium browser.

References

[1] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, "Comparing the Effects of DNS, DoT, and DoH on Web Performance," arXiv:1907.08089 [cs.NI], 2020

[2] M. Hoffman, "Firefox Nightly Secure DNS: Experimental Results," Mozilla Nightly Blog, Aug. 28, 2018. Accessed: May 11, 2019. [Online]. Available: https://blog.nightly.mozilla.org/2018/08/28/firefox-nightly-secure-dns-experimental-results

[3] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, May 2016, doi: 10.17487/RFC7858

[4] P. Hoffman and P. McManus, "DNS Queries over HTTPS (DoH)," RFC 8484, Oct. 2018, doi: 10.17487/RFC8484