How DNS undermines Tor's anonymity and state-of-the-art defenses: a comparison

Catarina Gonçalves
University of Porto
up201906638@fc.up.pt

Cristina Pêra
University of Porto
up201907321@fc.up.pt

Deborah Dore
University of Porto
up202202823@fc.up.pt

***Abstract*—An ongoing topic in Tor is threats to anonymity. The modern Tor user needs to exercise caution if they don't want to reveal their identify. DNS and their recursive resolver may be a contributing factor to a problem. In actuality, DNS requests are frequently transmitted in plain text. In this work, we present background information, cutting-edge assaults, and defense strategies like DNS over TLS and DNS over HTTPS. We also investigate DNS requests for websites ending in .onion, which ought to be more secure but aren't.**

***Index Terms*—Tor, DNS, privacy, anonymity, fingerprint**

I. Introduction

The Onion Router project, known as Tor, is the most popular anonymity network today. Tor was developed by US Naval Research Laboratory employees Paul Syverson, Michael Reed, and David Goldschlag with the goal of protecting the identities of US Navy intelligence agents [1]. Later in 2004, it was released under a free license for public use. Due to its nature, Tor is mostly used by anyone who wants to protect their privacy online. Even though it has also been used for malicious purposes, it still represents a significant step forward for democracy. It gained appeal particularly in oppressive nations with restricted freedom of speech. When used for good, it can give to the minority a platform to speak out without worrying about being found out and penalized. It can also be a technique for the majority to generate a constructive argument and be confronted with opposing viewpoints [2]. For all of these reasons, it's critical to early identify system flaws and put defenses in place. To ensure anonymity, Tor users' IP addresses and communication content are masked; the packet is encrypted and goes via a succession of nodes, and each node can only decrypt the packet of one layer in order to determine who the next node will be. This behavior is described in detail in section II.

Even though Tor was carefully designed to protect against threats to user anonymity, it is nevertheless susceptible to attackers who can track incoming and outgoing network data and correlate these flows. The consequences of this attack, also known as *correlation attacks*, are multiple and severe. Despite the fact that TCP connections are an important component of communication and hence the primary data source for an attack, Domain Name System (DNS) traffic is also extremely informative: simply opening a single webpage results in hundreds of DNS queries being sent to several distinct domains [3]. An attacker could exploit this behaviour to its advantage and use this information to craft even more successful attacks as described in section III.

Encrypting DNS is one of the many ways to avoid sending in DNS requests in plain text. Nowadays, most popular DNS providers, such as Google and Cloudflare, have adopted encryption mechanisms for DNS queries. In this paper, we will examine the two most extensively used standard protocols for DNS encryption: DNS-over-TLS (DoT), which was adopted by the IETF in 2016, and DNS-over-HTTPS (DoH), which was standardized in 2018 [4]. Both techniques seek the best way to make an attacker's life difficult while avoiding high latency and amortizing costs.

We start by looking into how DNS and Tor operate. The resolution of DNS names by Tor's exit relays will then be examined. Finally, we'll talk about attacks and defenses. We shall draw attention to unanswered issues about the threat of Tor's anonymity as we approach to a conclusion.

II. Background

All the information required to comprehend the work done in this paper and the section that follows will be provided in this section.

A. Tor

Tor is an anonymity network that protects the privacy of data and communications on the internet by concealing your identity as you browse. Tor takes its name from how it functions and is derived from the project name "The Onion Router." Your data will be first encrypted multiple times before being sent over a network of servers, called relays. The guard is the first relay. This one handles data reception and decryption of the first layer. Although the guard relay is aware of your IP address, it is unaware of the website you are attempting to access. The address of the following relay is the only information it has. The subsequent relays do not know your IP address or the website you are attempting to access. Their only function is to transmit the data to the following relay by removing an encryption layer. As a result, when the data reaches the output node, the last relay, it is decrypted and can then visit the desired site.

Using a specific method, such as *Diffie-Hellman*, each layer is encrypted with a unique key that has been traded with the user. Using an unsecured channel, two parties with no prior knowledge of one another can establish a shared secret key together using the Diffie-Hellman key exchange mechanism. The involved parts, *A* and *B*, communicate through an unsecured channel to exchange a value, g. Then, A sends g^a to B and B responds with g^b. $g^(ab)$ will serve as the final key. A hypothetical attacker will never be able to infer $g^(ab)$ from g^a or g^b because the parts know g and their value, g^a and g^b, and can derive $g^(ab)$, while the attacker can't. Even adding the keys together will result in $g^(a + b)$ and not $g^(ab)$. The user can exchange keys using this manner with the guard, the relay (passing through the guard), and the exit node (passing through the relay and the guard). In order to maintain anonymity, a user must never share directly their key with a relay other than the guard; otherwise, the other party will learn their IP address. These keys are used to encrypt each layer of the packet so that the node can decrypt the layer that matches its key.

B. DNS

The DNS, meaning domain name system, is a mechanism that changes website domain names into IP addresses so that you can locate and load the addresses on your web browser. The DNS resolver is the initial step in the DNS lookup process. It is in charge of interacting with the client who submitted the request and starting the chain of queries that results in the conversion of a URL into the necessary IP address. Understanding how DNS functions is crucial in order to comprehend how an attacker can use them to their advantage. A client sends a DNS query to a recursive resolver. A recursive resolver is a server provider by the ISP with resolving and caching capabilities that is able to resolve a domain. If this is the first time that a client requires the resolution of this DNS on this server, the DNS resolution is still not cached and therefore the server contacts a number of authoritative name servers which hold a distributed database of domain names to IP mappings in order to resolve the DNS. In order to find a response to the client's query, the recursive resolver iterates through the hierarchy of authoritative name servers. The client can connect to the target host using the resolved IP address. [4]

Along with website fingerprinting and end-to-end correlation attacks, a traffic correlation attack gives the attacker the ability to spy on both incoming and outgoing Tor TCP traffic. As it tries to connect the two, the attacker can only see inbound, encrypted Tor traffic and use website fingerprinting to try to figure out which address the user is visiting. To identify the websites from which the traffic is coming and where it is going, it uses machine learning. Because DNS queries are typically provided in plain text, DNS resolution is relevant in this scenario. Many pieces of information could be gleaned from these queries to better understand the website the user is viewing.The fact that the size, timing, and sequence of TLS packets reflect the content of a website can be used by an attacker. [4].

C. DNS resolution in Tor

The resolution of DNS queries in Tor guarantees a higher level of anonymity, rather than making the requests directly. Apart from preventing the resolver from seeing the IP address, it also guarantees that the ISP doesn't know that a resolve to a domain name was attempted.

In Tor, exit relays are the ones who perform DNS resolution for Tor clients. The Tor client sends DNS requests over Tor to prevent DNS leakage. The Tor browser establishes a connection to the SOCKS proxy exposed by the local Tor client. The SOCKS protocol is a wraparound that applications can use to instruct the Tor client to establish a circuit to a given domain and port. The Tor client then selects an exit relay whose exit policy supports example.com. Then, the exit relay first resolves example.com and then establishes a connection to the resolved address. If the connection is successful, the client can then exchange data with the destination. Tor exit relays use whatever ISP's resolver is configured on the machine, also a public ISP address such as Google DNS resolver 8.8.8.8. The cache is turned off to prevent tracking attacks [3].

III. Review on attacks and defenses

Attacks on the Tor systems can take in any shape. In so-called *correlation attacks*, an attacker will commonly attempt to correlate two flows—the one from the user to the guard node and the one from the exit node to the website—in order to determine which page the user is visiting. These attacks will ultimately de-anonymize the user and reveal its research. By evaluating the size, timing, and sequence of TLS packets, which constitute the page's content, the attacker may also seek to identify the websites that users are browsing - these attacks are called *fingerprint attacks*. We will discuss suggested attacks from the literature in this part and recommend defenses. In most cases we will distinguish between *closed-world attacks* and *open-world attacks*. The attacks in the first typology are those in which the user is given a list of websites that they can visit. We are most familiar to the second form of attack in everyday lives, when the attacker is unaware of all the websites visited by the users. This section's main objective is to compare the attacks and suggest the best and most effective countermeasure to adopt.

A. DNS fingerprint using a Random Forest Classifier

A local listener can find out which websites a user is browsing through an encrypted or anonymous route by using DNS-based fingerprints. According to studies, these assaults can be successful against Tor, HTTPS, openSSH tunnels, encrypted web proxies, and VPNs.

In their work [4], Siby, Juarez, Diaz, Vallina-Rodriguez, Troncoso, carried on a simple DNS fingerprint attack and compared the results using different defence mechanism.

DNS fingerprint has been treated as a supervised learning problem: a multi-class classifier is trained over a collection of samples. The classifier must discriminate between websites it has already seen during training in *closed-world attacks*, so it is aware of every conceivable page a user might visit. In contrast, there might be sites in the *open-world attacks* that the classifier is unaware of. The features on which the classifier is trained on a novel set of features: since the DNS responses are smaller than web resources, these usually fit inside a simple TLS record therefore common features are inadequate. The authors of the papers proposed the use of n-grams of TLS records lengths in a trace. Packets going from the client to the resolver will have a positive n-gram, and a negative n-gram for packets going from the resolver to the client. An example is (-64,88,33,-33) [4].

A Random Forest [9], which is an aggregation of individual Decision Trees, serves as the classifier. The class that the majority of the trees choose is the output of the random forest (for classification problems). This is why single decision trees often perform worse than random forest classifiers, which are far more resistant to overfitting. After training the classifier, the analysis of the data revealed that the F1-Score, the ability of the classifier to distinguish between classes, in open-world attacks, is equal to 0.7 when the classifier's threshold is set to 0.8, demonstrating a reasonable ability of the classifier to recognize which page the user is visiting. Now we will provide some solution to these types of attacks (with focus on open-world attack):

- *EDNS(0) Padding*: to counteract size-correlation attacks on encrypted DNS, the DNS protocol does indeed have a feature called EDNS Padding that adds padding to both DNS clients and resolvers. The general rule of thumb is to pad DNS request with multiple of 128 bytes and DNS responses with multiple of 468 bytes. We can have variations to this technique:
  - EDNS(0) Padding with 128 padding: using 128 bytes for the padding.
  - EDNS(0) Padding with adBlock: use an adBlocker alters the pattern of the request making an attacker's life more difficult.
  - EDNS(0) Padding with 468 padding: using 468 bytes for the padding.
- *DNS over TLS*: DoT is a protocol for encrypting DNS requests that uses TLS to encrypt and authenticate communications. It works by wrapping DNS queries and responses via the Transport Layer Security (TLS) protocol. Furthermore, it protects against on-path attacks that alter or falsify DNS queries and responses. It has been observed that applying this encryption method causes a 0.3 drop in the classifier's F1-Score ability.
- *DNS over HTTPS*: A relatively new technique called DNS over HTTPS (DoH) encrypts domain name system communication by routing DNS requests through an encrypted Hypertext Transfer Protocol Secure session. DoH wants to increase online privacy by concealing DNS query information. Using this encryption mechanisms reduces the F1-Score of the classifier of 10%.

Table I shows the results of the study.

Defense	F1-Score
No defense	0.7
EDNS0-128	0.691 ± 0.004
EDNS0-128-adblock	0.325 ± 0.011
EDNS0-468	0.430 ± 0.007
DNS over TLS	0.395 ± 0.007
DNS Over HTTPS	0.63 ± 0.002

TABLE I
RANDOM FOREST CLASSIFIER'S F1-SCORE ON OPEN-WORLD ATTACKS

DoT traffic present less variability than DoH and therefore performs better: this happens because DoT traffic records are all of the same type while in DoH we can distinguish between two types of traffic records. Therefore the attacker can extract more informations from DoH than DoT. Dot records also have in average less massages than DoH traces.
By looking at Table I we can conclude that the techniques that yields the best results are DoT and EDNS0-128 with an adBlocker. The size of the items shipped and received, however, is the factor on which the authors believe—and their research support this—that the most attention should be paid because it has an impact on every study result. Since an attacker may obtain a lot of information from them, it's crucial to select the optimal method to conceal the true size of the packets.

B. DNS fingerprint using a Wa-kNN classifier

The Wa-kNN algorithm calculates the weighted K-Nearest Neighbor for a given set of data. Wang et al. [7] conducted this website fingerprinting attack. This technique is used to identify a website's specific IP address. By doing this, attackers can gain information about the website's users and their activities. The algorithm makes use of a K-Nearest Neighbor classifier to assign a weight to each observation. This weight is then learned using a custom weight-learning algorithm. From the packet trace between the Tor client and its security, Wa-kNN extracts a set of features to classify each website.

In their paper [3], Greschbach, Pulls, Roberts, Winter and Feamster, have extended Wa-kNN by allowing it to take input from a list of sites that have been derived from observing DNS requests. In particular, they implement two DefecTor attacks:

- *Close the world attack*: This attack consists of a modified Wa-kNN classifier, which when calculating the k-nearest neighbors only considers the distance to observed sites. The classifier still considers the distance for all unmonitored sites. This attack increases the diligence of the conventional website fingerprint attacks, making them more like a "Closed World" configuration. These are

characterized by being digitally known sites and where the world has a limited size.
- *High precision attack*: When there is a tracking of a monitored website, to ensure high precision, it confirms that we observe the same website in DNS traffic. Otherwise, it ends up passing the final classification to unmonitored.

Both of the aforementioned attacks are limited by the percentage of outgoing bandwidth observed by attacker, which makes it impossible to identify a monitored site in DNS traffic that the attacker is not able to see. We can conclude that the first attack benefits from observing increasingly more exit traffic, while the second attack has near-perfect accuracy. However, the close the world attack almost reaches high precision levels at the maxium of the exit bandwidth.

To solve the problem, two types of solutions were considered:

- Short-term solutions: Although there are many aspects to this, the objective is to reduce the exposure of DNS requests. Three extreme design points, in which all exit relays use Google's DNS resolver, local resolver or the resolver provided by their ISP, were taken into consideration instead of significant DNS protocol enhancements.
  - Google's DNS resolver: The corporation would acquire metadata about every Tor user's activities if all exit relays used Google's open resolver, which is in opposition to the distributed trust aim of Tor. Thousands of meek clients chose exit relays while the system was up and running, which allowed Google to see traffic entering and, to a lesser extent, leaving the Tor network and launch DefecTor attacks. These relays use Google's public resolver. This one should therefore be avoided.
  - Local resolver: As a result of each iterative DNS query being exposed to a wide range of ASes in the network, which allows other parties to learn the DNS requests of Tor users, Tor is in this scenario completely independent of third-party resolvers. Therefore, all exit relays might only use the resolver that their ISP provided. Due to resolvers typically being in the same AS as exit relays and the inability of AS-level adversaries to distinguish between DNS queries from exit relays and unrelated ISP customers, this would reduce the network exposure of DNS requests. But this configuration raises the prospect of censored and improperly configured DNS resolvers. Additionally, a small number of ASes, such as OVH, host an excessive number of exit relays, causing them to become the highly centralized data sinks that Tor intends to avoid.

  Considering the above, exit relay operators should avoid public resolvers such as Google and OpenDNS. Instead, if the operator's ISP already hosts a large number of exit relays, they should either run their own resolvers or use those offered by their ISP. To make DefecTor attacks less precise, Tor can also remedy the recently found Tor clipping flaw and think about dramatically raising the minimum TTL for the DNS cache at exit relays. Finding the longest permissible TTL without a pronounced negative impact on user experience is necessary for this modification. Additionally, website administrators of vulnerable websites have the option of extending the TTL of their DNS records as soon as the clipping fault is fixed.
- Long-term solutions: This approach included the implementation of T-DNS, which makes use of a number of TCP improvements to transmit the DNS protocol via TCP and TLS. Confidentiality between exit relays and their resolvers is provided by the TLS layer. Furthermore, website operators should provide an onion service as an alternative for users who are very worried about security. As long as the onion service does not include material from another service, Tor users that connect to it never leave the Tor network and do not require DNS.

C. Yet another threat: the leakage of .onion queries

The .onion namespace consists of a pseudo-top-level domain used to route requests to some hidden services used in Tor. This namespace is not delegated to the global DNS, since all .onion queries are routed within the Tor network. However, multiple .onion requests have been detected in the DNS root, and this number has been increasing over time. In order to resolve this situation, some of the proposed solutions include:

- *Host-Level Remedies*
  - *Browser*: block .onion queries based on further intelligence in the browsers by excluding .onion domains in a prefetching suffix list, or by only allowing the use of .onion when TOR is used.
  - *Legacy Software*: implementing host-level profiler of all DNS traffic generated by hosts;
  - *Configurations*: Tor distribution should provide automatic system-level configuration of .onion resolution.
- *Network-Level Remedies*:
  - *DNS resolvers*: block queries in the DNS hierarchy by using negative caching, and placing public recursive name servers closer to users in order to avoid sending out queries to the root for TLDs that don't exist;
  - *Authorative Name Servers*: these servers shouldn't attempt to resolve .onion strings, instead they should return negative resolution results.

IV. Conclusion

In order to help the reader understand the context, we provided him with all the required background information at the beginning of this paper. Then, three cutting-edge attacks and their main defenses were suggested.
To conclude, although the DNS issue is well known and numerous researchers have attempted to address it, it still poses a serious threat to anonymity, and we must keep dealing with it because the number of attackers is growing daily.

REFERENCES

[1] "The Tor Project: Privacy & Freedom Online," Tor Project — Anonymity Online. [Online]. Available: https://www.torproject.org/. [Accessed: 11-Oct-2022].

[2] E. Jardine, "The dark web dilemma: Tor, anonymity and online policing," SSRN Electronic Journal, 2015.

[3] B. Greschbach, T. Pulls, L. M. Roberts, P. Winter, and N. Feamster, "The effect of DNS on Tor's anonymity," Proceedings 2017 Network and Distributed System Security Symposium, 2017.

[4] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, and C. Troncoso, "Encrypted dns -> privacy? A traffic analysis perspective," Proceedings 2020 Network and Distributed System Security Symposium, 2020.

[5] A. Mohaisen and K. Ren, "Leakage of .onion at the DNS root: Measurements, causes, and countermeasures," IEEE/ACM Transactions on Networking, vol. 25, no. 5, pp. 3059–3072, 2017.

[6] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. K. Jha, "De-anonymisation attacks on Tor: A Survey," IEEE Communications Surveys; Tutorials, vol. 23, no. 4, pp. 2324–2350, 2021.

[7] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in USENIX Security. USENIX, 2014. URL: https://nymity.ch/tor-dns/pdf/Wang2014a.pdf

[8] "Doh! DNS over HTTPS explained," RIPE Labs. [Online]. Available: https://labs.ripe.net/author/gih/doh-dns-over-https-explained/. [Accessed: 19-Oct-2022].

[9] J. Hatwell, M. M. Gaber, and R. M. Azad, "Chirps: Explaining random forest classification," Artificial Intelligence Review, vol. 53, no. 8, pp. 5747–5788, 2020.