Leveraging eBPF to preserve user privacy for DNS, DoT, and DoH queries

Sean Rivera
University of Luxembourg
sean.rivera@uni.lu

Vijay K. Gurbani
Illinois Institute of Technology
Vail Systems, Inc.
vkg@{iit.edu,vailsys.com}

Sofiane Lagraa
University of Luxembourg
sofiane.lagraa@uni.lu

Antonio Ken Iannillo
University of Luxembourg
antonio.iannillo@uni.lu

Radu State
University of Luxembourg
radu.state@uni.lu

ABSTRACT

The Domain Name System (DNS), a fundamental protocol that controls how users interact with the Internet, inadequately provides protection for user privacy. Recently, there have been advancements in the field of DNS privacy and security in the form of the DNS over TLS (DoT) and DNS over HTTPS (DoH) protocols. The advent of these protocols and recent advancements in large-scale data processing have drastically altered the threat model for DNS privacy. Users can no longer rely on traditional methods, and must instead take active steps to ensure their privacy. In this paper, we demonstrate how the extended Berkeley Packet Filter (eBPF) can assist users in maintaining their privacy by leveraging eBPF to provide privacy across standard DNS, DoH, and DoT communications. Further, we develop a method that allows users to enforce application-specific DNS servers. Our method provides users with control over their DNS network traffic and privacy without requiring changes to their applications while adding low overhead.

ACM Reference Format:
Sean Rivera, Vijay K. Gurbani, Sofiane Lagraa, Antonio Ken Iannillo, and Radu State. 2020. Leveraging eBPF to preserve user privacy for DNS, DoT, and DoH queries. In *The 15th International Conference on Availability, Reliability and Security (ARES 2020), August 25–28, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3407023.3407041

1 INTRODUCTION

Motivation Since the publication of RFC 7626 [5] in August 2015, Domain Name System (DNS) privacy has been a growing area of research in the community. DNS is a fundamental protocol on the Internet, and as all traffic originates as a DNS request, users reasonably expect that these queries remain private between the stub resolver and the DNS nameserver (even though these queries are normally sent in cleartext). DNS privacy is a fundamental tenet of browser privacy as all traffic originates as a DNS request and anything embarrassing or potentially compromising from a user is visible to the browser, and vulnerable at the DNS layer. Attackers who successfully intercept DNS communication gain unique insights into user behavior. Despite research in areas of resolver privacy and encrypted traffic [11][19][25][38][5], most current solutions in protecting DNS traffic are at the provider level, outside the control of users. This allows providers to access any information from a DNS request, potentially decreasing the privacy of a user. RFC 7626 recommends that one of the best options for privacy-conscious users is to use different DNS providers for different applications(e.g. Firefox, Skype), rendering them more difficult to track. The current structure of most user-based systems makes using multiple DNS servers difficult, prompting the need for a more efficient way to utilize this privacy method.

Problem statement Currently, most applications for DNS privacy are designed to protect against third party eavesdroppers, but there is not a similar system for users to control their privacy against pervasive monitoring at the resolver level. Privacy conscious users who wish to avoid being fingerprinted by their DNS provider must manually change their system resolver, a very coarse solution. In this paper, our goal is to propose an alternative system to route user DNS queries to a variety of different providers. This addresses the current need for an effective method for privacy-conscious users to control their DNS traffic and increase privacy. This system should also be compatible with other advancements in DNS user privacy such as encrypting user data over HTTPS or TLS.

Objective Our goal is to provide users with more control over their privacy by allowing them to use application-specific DNS filtering or monitoring for any from of DNS that cannot be directly filtered. Additionally, we aim to ensure that this interface works in conjunction with other alternative DNS systems such as DoH (DNS over HTTPS) and DoT (DNS over TLS), providing a helpful addition to ensure a user's privacy. We aim to provide users with far greater control of their DNS queries.

Solution The extended Berkeley Packet Filter (eBPF) is a new technology for kernel-space monitoring and network traffic shaping. In this paper, we demonstrate how eBPF can be leveraged to provide better user privacy. We show how eBPF can monitor DoH systems, allowing users to make informed decisions about their privacy even though it cannot directly interact with the underlying communications. We then demonstrate the privacy gains of such a system and measure the performance overheads imposed.

ARES 2020, August 25–28, 2020, Virtual Event

ACM ISBN 978-1-4503-8833-7/20/08…$15.00
https://doi.org/10.1145/3407023.3407041

The rest of the paper is organized as follows. Section 2 covers the necessary background for this research. Section 4 describes the work we did with eBPF and DNS privacy methods. Section 5 describes our experimental analysis of our system. Section 6 contains our conclusions and thoughts on future work.

2 BACKGROUND AND RELATED WORK

The Domain Name System (DNS) is one of the most fundamental protocols on the Internet. It maps human-readable domain names to corresponding IP addresses of Internet endpoints. DNS presents an essential component of our Internet infrastructure.

2.1 Privacy assessment of DNS

There are fundamental concerns about privacy and security in DNS. Users are worried about privacy in DNS-based name resolutions. The critical path occurs between the client and the resolver (server), where several entities such as the Internet service provider (ISP) and domain name server might eavesdrop on communications. While new technologies address the problem of third-party eavesdropping, they do so by instead giving more information to the DNS server operator, thereby requiring privacy-conscious users to regularly change their DNS server to preserve anonymity.

Recently, various proposals have been standardized by the Internet Engineering Task Force (IETF) as Request For Comments (RFC) documents to encrypt the DNS communication between clients and resolvers. In RFC7858 [38], the authors described the use of Transport Layer Security (TLS) to provide privacy for DNS, and in a followup [9], the authors demonstrated how to implement RFC7858 functionality in QUIC to provide transport privacy for DNS. In RFC8484 [19], the authors proposed DNS Queries over HTTPS. In RFC8094 [25], the authors proposed the use of Datagram Transport Layer Security (DTLS) for DNS. The common contribution of all proposals is to encrypt and authenticate the DNS traffic to guarantee both client-to-resolver integrity and security and confidentiality across a variety of different transport protocols. Before that, there were several other implementations such as DNSCrypt[11] a protocol that authenticates communications between a DNS client and a DNS resolver.

In this paper, we focus on two predominantly deployed standards: DNS over TLS (RFC7858) and DNS over HTTPS (RFC848).

2.2 DNS over TLS (DoT)

DNS over TLS (DoT) is specified in RFC 7858 [38]. It takes the existing DNS over TCP protocol and wraps it into a TLS layer. The goal of DoT is to increase user privacy and security by preventing the manipulation of DNS data via *Man-in-the-Middle* attacks. DoT is well-suited for a system-wide DNS service such as the stub resolvers of operating systems.

2.3 DNS over HTTPS (DoH)

DNS over HTTPS (DoH) is a more recent proposal from October 2018 and is specified in RFC 8484 [19]. It contains two different approaches, based on the HTTP GET and POST methods. The GET method uses a URI with a parameter that represents a base64 encoding of a DNS message in DNS wire format and should, therefore, be avoided for privacy-conscious users. In the POST method, the DNS request is part of the HTTP message body in the DNS wire format. However, regardless of the method, the DNS reply is sent in the HTTP message body in the wire format. DoH is being utilized in different browser vendors; Chromium-based browsers already support DoH [34] [10] as does Firefox 62+ [28]. DoH is easier to integrate with application development than DoT because it uses the traditional web infrastructure and ports, whereas DoT uses a new custom port (853). Unfortunately, DoH requires users to place more trust in their browsers and providers for privacy, as all of the information is located in one location.

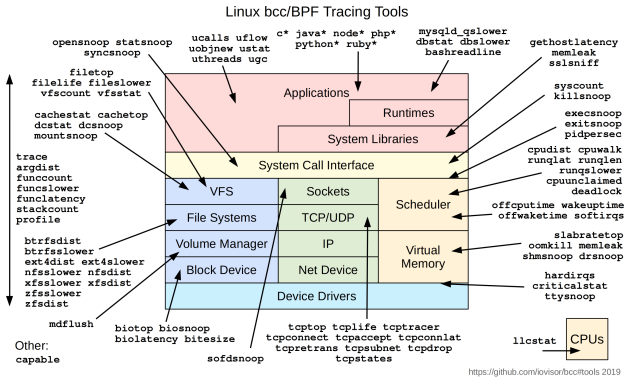


Figure 1: All available BPF tracing and performance tools. Arrows signify the values that can be traced in each Linux block with eBPF. This figure is provided by the developers of bcc, a compiler for eBPF [23].

2.4 BPF/eBPF

The Berkeley Packet Filter (BPF) [27] is a high-performance instruction set for a bytecode virtual machine inside of the Linux kernel. It was created to be a fast programmatic network filtering tool. BPF was extended in 2014 is what is now known as extended BPF (eBPF). eBPF [16] is an extension to the classic BPF framework with hooks to monitor processes beyond network applications. It can be used for generic event processing in the kernel, profiling programs, and libraries. This allows developers additional processing capabilities for more complex applications. One of the main advantages of eBPF is that it allows a user-space application to load code in the kernel at run-time. This means that the new code is executed without recompiling the kernel or installing any optional kernel modules. eBPF programs can use helper functions [29]. Figure 1 shows all available eBPF tracing and performance tools. All components of the Linux system are broken into blocks, with corresponding eBPF tracepoints, corresponding to helper functions. These helper functions are implemented to interact with the system in kernel or user space. The helper functions can be used to print debugging messages, to interact with eBPF maps, or to manipulate network packets.

One of the many helper functions provided by eBPF are user-space probes, or uProbes (shown in red in Figure 1). uProbes allow monitoring of function calls(uProbe), function returns(uRetProbe) and user-specified locations in memory (USDT). To preserve system functionality, uProbes only allow the monitoring program to read the data, but there is no way to change the memory shown with uProbe.

2.5 XDP

eXpress Data Path (XDP) [20] is a new programmable layer in the Linux kernel network stack. It provides a run-time programmable fast packet processing interface inside the kernel. This provides users with access to the lower layers of the networking stack and allows code to be run on packets based only on their headers. Utilizing function hooks, user-defined eBPF programs can access and modify packets in the NIC drivers. This allows XDP applications to reach performances far beyond traditional networking hardware. XDP is already actively used by many different groups and companies. Cloudflare integrated XDP into their DoS mitigation pipeline [2] while Suricata has XDP plugins [13] and Facebook released to use XDP as a high-performance layer 4 load balancer [22].

2.6 Related work

In van Heugten et al. [36] performed a survey analysis of the current methods used for ensuring user privacy in DNS resolution and analyzed the performance of several methods. They compared the overhead and privacy gains for DoT, DoH, DNSCrypt, and other research techniques. The previous analysis was performed by Federrath et al. [14], however, it was performed four years before RFC 7626, and doesn't take many modern solutions into account.

Bushart et al. [8] proposed a method to analyze encrypted and padded DNS traces. They combined size and timing information to infer the websites' user visits. Kim et al. [26] created DNSminer which allows servers and others with collections of large DNS traces to efficiently and reliably extract unique users. Borgolte et al. [4] and Hounsel et al. [21] explored the policy implications of DoH by measuring its performance and how encrypted transports for DNS affect the end-user experience in web browsers. They analyzed and investigated the performance, privacy, competition, and regulatory policy. Hounsel et al. [21] measured how encrypted transports for DNS affect the end-user experience in web browsers. The performance measures are resolution time and web page load time. Brack et al. [6] proposed the use of requests in DNS as an anonymization layer by forwarding packets over public DNS resolvers.

Herrmann et al. [17] proposed EncDNS, an encryption service for DNS that provides user privacy by taking advantage of the privatization provided by the communication between a recursive resolver and an authoritative name server, to encapsulate the DNS query such that no single server knows both the source address and requested lookup address. EncDNS was extended in structure by Oblivious DNS [31]; this work proposed a cleaner way to interface between the encrypted server and the recursive resolver. Additionally, they addressed some issues with the key propagation inherent in EncDNS.

The security of many different DNS privacy techniques has been evaluated as well. The authors of [18] demonstrate that even with minimal knowledge of the DNS queries, traditional browsing carries enough patterns to identify users. Meanwhile, Siby et al. [32] analyzed both DoT and DoH traffic and demonstrated that on their own, encrypting DNS traffic is not sufficient to preserve user privacy. They suggested further extensions to DoH were required.

EXPOSURE[3] is one of the first passive DNS frameworks to be developed. It proposed that a DNS monitor can look at only the responses from the servers to gain information about network traffic without compromising user privacy. Passive DNS doesn't work if used with encrypted DNS methods.

The originality of this paper is in providing a model and analysis of the first use for eBPF and application-specific DNS privacy. Our implementation allows users to have more control over their DNS privacy. It supports both DNS and DoT fully with targeted support for DoH as well. This method differs from other methods by providing the user more control at the application level and through the use of eBPF, which provides it with less run-time overtime.

3 THREAT MODEL

There are four groups of potential threats to user privacy that we consider for our threat model: privacy from the ISP, the browser vendor, attackers on a shared network, attackers on a shared system. Each threat can be modeled as an actor with different capabilities and interests. Our threat analysis assumes all four possible threats, although that is not the common case. We also assume that the user is not employing any other DNS privacy tool besides DoH and DoT. With this in mind, we summarize the current protections provided by DoT and DoH for each group.

(1) The ISP: The ISP is the most significant concern to user privacy due to the amount of control it has over a user's data. The ISP can control the stub resolver for the host, monitor any traffic that the host publishes, and even monitor the sites the user visits. Privacy from the ISP is the most complicated to address as it requires a high level of encryption and anonymization on all traffic entering and leaving the network. The ISP utilizing user information to compile a user profile is considered a major threat to user privacy. This threat model assumes that the ISP can determine any DNS transaction to its stub resolver. However, DoH and DoT are assumed to provide protection if an alternative resolver is used than the resolver owned by the ISP. Due to the scope of capabilities held by the ISP, it is considered to be the most significant threat.

(2) The browser vendor (e.g. Google, Mozilla, Apple, Microsoft): The browser vendor has access to all traffic conducted by the user in the browser. If left unsupervised, the browser poses a significant threat to user privacy. For our threat model, the privacy risks of the browser vendor are only considered concerning DNS privacy when users utilize DoH, as this is the sole instance when the browser itself has control over DNS queries. We choose to focus on the browser vendor over other providers of DoH as there is already an existing ecosystem of DoH providers who are also browser vendors. DoH provides users increased security from the ISP while decreasing browser security by providing the browser vendor complete access to user DNS transactions. Since DoH bypasses cleartext DNS and DoT, the browser vendor is in complete control and can gather information freely. Similar to the threats to user privacy discussed with the ISP, the browser vendor compiling a user profile is considered a threat. The threat model focuses on the security risks in the browser vendor caused by the solution provided by DoH.

(3) Network Eavesdroppers: On shared networks, DNS requests may be accessed by other users due to traffic monitoring or flaws in encryption. Network eavesdroppers are perhaps the

most common type of attacker, and therefore their capabilities are the best understood. Network eavesdroppers are included within the threat model but are well handled by DoT and DoH as both provide well-established solutions to many of these problems.

(4) Server Eavesdroppers: Server eavesdropping is more advanced than network eavesdropping, and requires the attacker to be on the same system as the user. This can occur in a cloud environment or through other shared servers. Server eavesdroppers have all of the same capabilities as network eavesdroppers, however, they can also perform new types of attacks relating to user searches and caches. Using only DoT does not address these issues because of shared network caches, but DoH or false DNS queries can mitigate it. Any solution must be conducted in a more privileged ring of escalation to be considered viable for these threats.

For our research, we assume that only the root user (any user with permission to run eBPF programs) can be trusted with all user's DNS queries. Any other users or permissions are assumed to be a potential threat. Our research follows the model provided in Kim et al. [26] that determines the amount of information leaked in a given query. Our goal is to minimize the number of bits of information that anyone DNS server receives, thus limiting opportunities to build profiles of the user.

3.1 Attack Descriptions

This paper considers the following attacks:

- *De-anonymization*: De-anonymization attacks allow leakage of personal information that connects users to queries or creates profiles of users based on their queries. This attack is most frequently observed on the ISP or the browser vendor. Neither DoT nor DoH currently has the ability to address this type of attack. For this paper, we are trying to ensure that a given resolver is unable to construct an accurate profile of the user.
- *DNS Query Leakage*: DNS Query Leakage attacks occur when any part of the DNS query is leaked to any observer aside from the resolver. This can occur due to a lack of encryption, timing attacks, or other similar attacks. DoT provides encryption solutions but cannot address timing attacks, while DoH addresses both encryption and timing attacks.
- *Man-in-the-Middle*: Man-in-the-Middle attacks occur if an attacker can intercept DNS communications. Both DoT and DoH use TLS to address this.
- *Malicious Resolver*: Malicious Resolver attacks are DNS resolvers that return incorrect IP addresses though the error response mechanism. Neither DoT nor DoH has a way to interact with this.
- *DNS spoofing*: DNS spoofing attacks also return incorrect IP addresses, however, they are returned by a server downstream from a DNS resolver. While DoT and DoH do not directly provide security for these attacks, the infrastructure required to enable DoT and DoH provides security. Standard encryption and certification methods address this attack.

4 METHODOLOGY

In this section, we describe various ways that eBPF can be leveraged to aid privacy-conscious users in protecting their privacy.

4.1 Design Goals and Overview

eBPF provides developers with far greater access and control over user-space programs, which we can leverage to provide greater privacy protection of users' DNS requests. We look into the potential benefits provided by eBPF for traditional DNS as well as the newer DoT and DoH. As shown in Figure 1, eBPF provides users with a wide variety of filtering and monitoring hooks across the network stack. We aim to use the functionalities of eBPF to give the user more control over their DNS privacy.

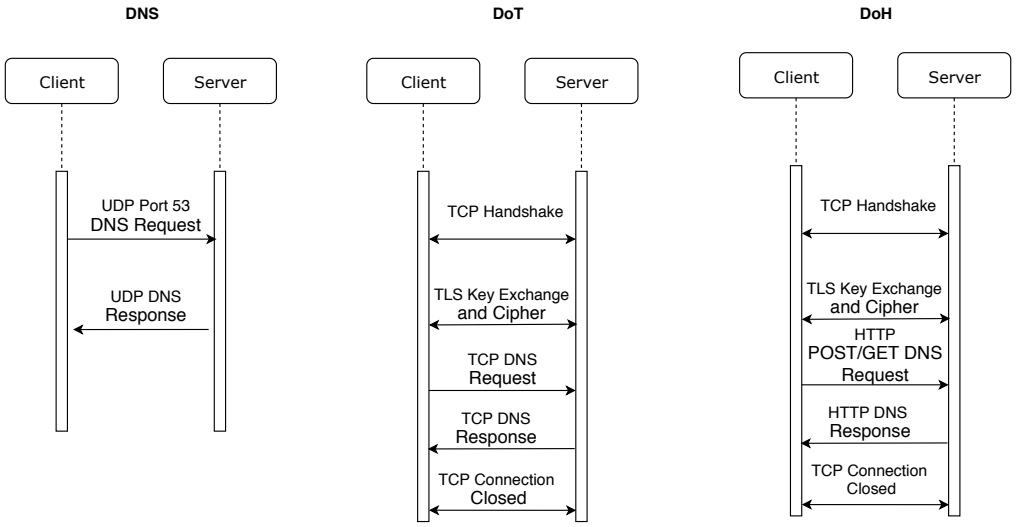
The core of this research is the use of eBPF to provide users with an application-specific DNS implementation, by adding functionality to invisibly edit the requests so that each application can point to an arbitrary server. We also demonstrate a method to capture every single DNS request and forward it to a group of other servers to detect potential prioritization or differences between responses.

4.2 Implementation

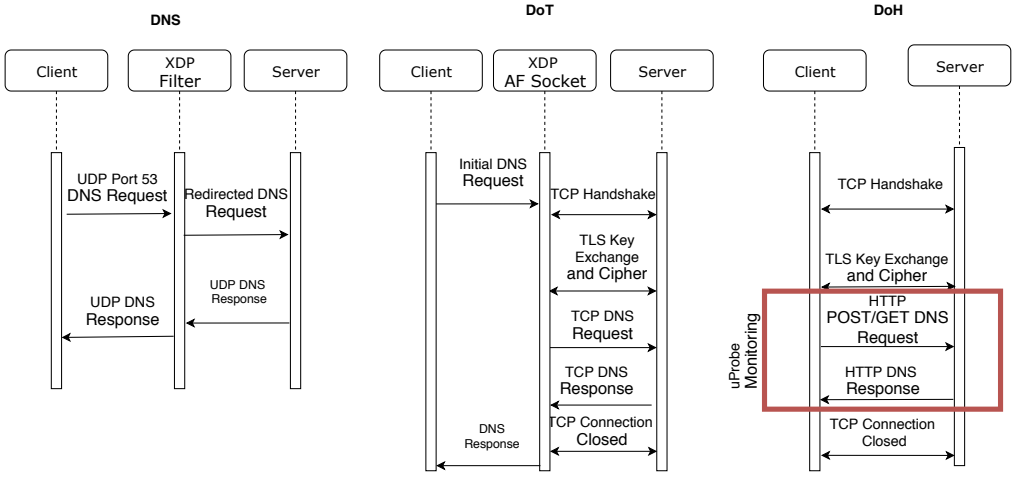
Our DNS system was implemented in Python 3.6 on top of Ubuntu 18.04.3, with kernel 5.0.0.27 using python-bcc[23]. We implemented the system using a hash table of process names and destinations. The system uses XDP to edit the packets before transmission. We hash the name of the application in eBPF and take advantage of the system's ability to determine which program is sending a given packet. From there we determine if the packet is a DNS request and apply the appropriate filtering depending on the hash table rules. By default, we use the system DNS server if there is no specific rule to be found. We chose to implement our method application-specific because it was the most structurally complicated and therefore the best demonstration of eBPF capabilities.

For DoT implementation, we use the eBPF AF socket, a kernel-level socket with instantaneous data communication between kernel and user-space. We intercept the TCP packet before it leaves the system and change the destination address of the DNS server. We complete the negotiation by acting as a trusted MiTM. In this way, the user-space program behaves normally but does not transmit outside of the system. Structurally, we were able to build an application-specific DoT system in the same way we built our DNS system. However, there is more complexity added due to the TLS handshake and encryption requirements.

Additionally, we use eBPF uProbes to hook into the DoH encryption and decryption for monitoring sent requests. In the uProbes, we were able to see the arguments between the encode and decode functions and their respective return values. While we cannot edit any of these, this allows us to reconstruct the message that is sent and display its contents to the user. It is also possible to construct and send fake DoH requests. We don't believe this is valuable since the Browser vendor operates the DoH server and can quickly verify that the fake requests are essentially noise and discard them. This means that any non-privacy conscious actors can quickly verify the real look-ups. Therefore, we do not focus on this functionality.



(a) Current versions



(b) Our eBPF implementation

Figure 2: Normal functionality of DNS, DoT, and DoH vs. our eBPF implementation

For all three versions of our tool (eBPF for DNS, eBPF for DoT, eBPF for DoH), the 1BD-improved and EFF's panopticlick algorithms are utilized [12]. To determine which server to send the next packet of information to, our eBPF tool uses the algorithms to determine how much information is leaked to a given DNS resolver. We compute the number of bits exposed in real-time. To reduce leaking information to multiple resolvers, a list of resolved domain names is saved to prioritize sending information of similar queries to the same DNS resolver. In DNS and DoT, this list is leveraged to minimize the number of bits exposed. For example, if a user were querying the top Alexa 500 site, our eBPF system would choose the resolver for which visiting the site would reveal the least amount of information about the user. Subsequent revisits to the same site use the resolver that satisfied the first query. However, in DoH, as this behavior cannot be manipulated, the amount of bits exposed is calculated and shared with the user. In order to allow users to interact with the system, we provide a simple application that identifies running applications and allows users to specify which DNS or DoT server they should connect to. This application also monitors DoH and displays an estimation of leaked information.

4.3 eBPF for DNS

Traditionally, the best method for concealing DNS activity was to use the most popular DNS server and 'hide' in the noise of the massive amount of requests they received every day. However, with the recent advent of big data processing technology, this strategy is no longer valid. As such, the best strategy has become to shift queries to a wide variety of competing DNS servers, providing an

incomplete or limited picture of the user's browser activity to any single server and making tracking the user more difficult. In order to spread traffic, the current solution is to use a DNS *sink hole* [7] or a similar router-level solution as the system DNS is not designed to support arbitrary server changes. However, these systems require control over the router, and on complex networks using this method is not feasible and does not provide extra privacy.

With the advent of XDP, there is now an efficient low-cost way for users to route DNS traffic to arbitrary servers at the system level. XDP is a component of eBPF that has been extended to be able to not only monitor but to change data as well. As eBPF is automatically aware of which process is sending a packet and can easily find the type and content of a normal packet, users can now filter DNS requests by applications. As shown in Figure 2, all requests, and responses are filtered through the XDP filter before being sent to the server or returning to the user. This masks activity by directing different requests to different servers. eBPF is so versatile that a user can filter the application, the time when it was sent, or even maintain a circular buffer of acceptable servers and send one request to each in turn.

Even with eBPF, users are not protected from Network Eavesdroppers nor ISP monitoring due to the lack of encryption on DNS. However, DoT and DoH are designed to address this issue.

4.4 eBPF for DoT

DoT is effectively the same as traditional DNS but it is encrypted using TLS. Thus, it provides privacy from network eavesdroppers and ISPs concerning the content of browsing.

We use a novel eBPF functionality called an AF socket modeled off of the previous *ktls* work [37] which creates a virtual socket to route all traffic through. In this way, we capture the DoT request, edit it to direct it to the specified server, and then send the request to the virtual socket for handshake and communication. Once we have the final reply, we connect the socket back to the normal application. From the application's perspective, it simply performs a DoT query and receives a result as it normally would, but underneath it, we are capable of changing the DNS server it communicates to. For example, using eBPF in this way, an application may request a query from one server, but it is intercepted before the first TLS request is sent and instead sent to a server chosen by the user. This allows the user to send activity to an arbitrary DoT server much like they would a DNS server, as shown in Figure 2. This method allows us to do the same application-specific filtering as traditional DNS in a DoT environment.

4.5 eBPF for DoH

DoH gives users more privacy from ISPs at the cost of less privacy to browser vendors. Since DoH uses purely browser-internal functionalities (encrypted GET/POST), it makes DNS requests from browsers hard to analyze from the outside. At the moment there is no known way for a user to monitor their DoH requests and see exactly the data being submitted, except for through the browsers debug console. This means a user has no control over the DNS data being generated on their behalf.

While eBPF does not allow developers to directly interfere with DoH requests like with normal DNS requests, we can take advantage of the monitoring and tracking capabilities provided by our uProbes to monitor the creation of the DoH requests and the decoding of the responses. This allows us to verify that the browsers aren't transmitting extra data and alert users about how much a given DoH provider knows.

While, we cannot provide application-specific DNS privacy with DoH systems, by sending fake requests to the server it obfuscates the request the user is interested in. Theoretically, as the browser still knows which request is the truth, it can still track the user from source to destination. Therefore, we are unable to provide a full solution for direct privacy to DoH users.

However, the key advancement of this approach is that it can be used to monitor functionality. The user can use our method to view the information the browser is providing about DNS lookups to monitor all information shared by the browser. This allows the user to know exactly what information is being transmitted by the browser, which can be used to determine when it is time to change DoH providers, or when it is time to disable DoH and take advantage of our application-specific eBPF for DNS or use DoT to preserve privacy. Additionally, we can provide guarantees that the browser is only sending information to the chosen DoH provider, and not leaking information to other systems. This provides the user with direct control over their DoH usage and reduces risks in privacy at both the ISP and browser levels.

5 EXPERIMENTS & RESULTS

We analyze the performance impact of using eBPF to route all DNS traffic to different servers. The system is tested with application-specific and time-based routing. We outline three experiments that evaluate our system from a privacy perspective and quantify the overhead introduced by our system.

5.1 Performance Experiment Setup

The overview of our experimental setup is outlined in Figure 3. To verify the application-specific DNS code was working and not detectable, we set up the system with four applications pointing to four different DNS servers. Applications used included Firefox, Curl, nslookup, and wget. The four DNS servers we used were 8.8.8.8 (Google), 1.1.1.1 (Cloudflare), 9.9.9.9 (Quad9), and 158.64.1.29 (Fondation RESTENA). To verify the packet filtering and simulate both types of adversaries, we used both Wireshark and a virtual switch which recorded all DNS traffic. To verify that the filtering was not detectable by the applications, we recorded their debug output and warnings, and ensured that there were no new errors emitted by the application. For our experiments we only used IPV4.

Additionally, we evaluated our eBPF system while it was monitoring DoH packets. While we are unable to fully redirect the DoH packets with the current limitations of uProbes, we monitored the requests and verified the returns. We exported these returns to a user and raised an alert if there was a deviation between the DoH return the system lookup. While such a system wouldn't be used in practice, we found it helpful to confirm that the DoH providers were not trying to shape traffic based on information from the browser.

Our baseline analysis was conducted by performing a DNS lookup for each site of the Alexa 500 [1]. These websites were chosen because they were the most frequently used. We compared the returned

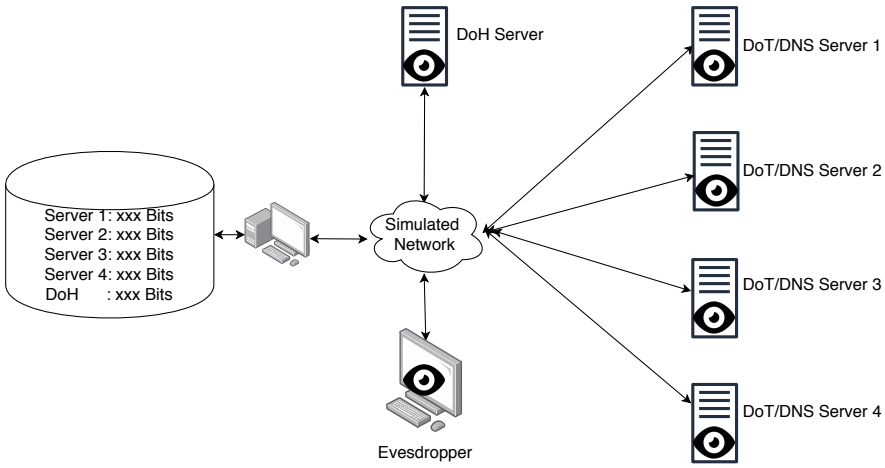


Figure 3: Experimental Network overview

IP addresses of many different DNS providers until we found a group of four that returned the same set of addresses in the same order. This ensured that there would be no latency issues with the IP addresses. We also ensured that the DNS providers supported DNS, DoT, and DoH with the same set of addresses. Next, we leveraged the application-specific DNS framework to determine the overhead provided. We ran four different applications (Firefox, curl, wget, nslookup) and measured the overhead of the application-specific DNS method against the normal system DNS with each application being assigned to one of our chosen DNS providers. We assigned each application to a unique DNS server and had it query each of chosen the Alexa 500 sites, measuring both the time and the CPU overhead of the query.

5.2 Performance Experiment Results

As presented in Figure 4, our eBPF solution does not substantially impact the overhead of a DNS connection, adding only 0.44% to the time it takes for a normal look-up. Meanwhile, DoT has an overhead of an additional 8.15%. We verified that each request is sent to the correct new DNS server by comparing the returned address with that of a DNS query from the virtual switch. We compute the time overhead by comparing the lookup times for all of the Alexa 500 sites both with and without our eBPF solutions. We repeated the experiment 20 times to get a clustered grouping of results for comparison.

While we could not directly alter the DoH packets, we measured the CPU overhead for monitoring the calls and returned values as an additional 3.13%. We show that eBPF is a valid method for securing privacy by spreading DNS server traffic and limiting tracking. To compute the CPU overhead, we measured the number of cycles used by the 4 applications, both with our eBPF uProbes monitoring and without. From there we calculated the number of additional cycles that were consumed by the monitoring. Similar to the above, we repeated the experiment 20 times to get a consistent clustering. While the DoH solution does take less time then the DoT solution, it is important to remember that the DoH monitoring is a passive monitor, not an active redirection. A full DoH redirection would require more overhead then DoT owing to the extra overhead of the HTTP connection.

5.3 Privacy Experiment Scenarios

To evaluate if our eBPF implementation protects the user's privacy, we performed three types of experiments. For the first type, we modeled the attacker as a user on the same system who could send DNS queries of their own, monitor traffic amounts, and even run WireShark in an unprivileged manner. We find that this is an adequate approximation for attacks on shared computing resources, such as cloud computing and similar systems [30][15].

For the second type, we placed an eavesdropper on the network to disrupt DNS queries and monitor all queries from our selected system. We show that our eBPF for DoT and eBPF for DoH systems described in Section IV prohibit information leakage and uphold user privacy.

The third type is based on Herrmann et al. [18], where bits of information exposed to a DNS server based on applications running and sites visited are calculated. It demonstrates how DNS servers are able to build a user profile and what information is most valuable. Using the 1BD-improved described in Section IV, we extend our eBPF implementation to minimize the number of bits sent to any server from a list of viable options. We compute the number of bits exposed in real-time for DNS and DoT. Although we cannot change the packets nor improve privacy, we are able to expose what is being leaked and flag any potential anomaly for the user. We show that the number of bits exposed is minimized based on the number of available servers and that a user running our system has better privacy than a user without it. Further, we compare our system with the test implementation of EncDNS, to compare both information exposure and overhead.

5.4 Privacy Experiment Setup

The same structural components of the virtual environment used to evaluate performance were used to analyze user privacy. A private network was established through the use of Ryu [35] in a simulated environment. The DoH server and four DoT/DNS servers were

Threat	Attack Strategy	DNS	DoT	DoH	EncDNS	eBPF + DoT	eBPF + DoH
ISP	De-anonymization	Vulnerable	Vulnerable	N/A	Partially Protected	**Protected**	N/A
	DNS Query Leakage	Vulnerable	Only if Allowed	Protected	Protected	Only is Allowed	Protected
	Malicious Resolver	Vulnerable	Vulnerable	Protected	Vulnerable	**Partially Protected**	Protected
Browser Vendor	De-anonymization	N/A	N/A	Vulnerable	Vulnerable	N/A	**User Given Warnings**
	Man-in-the-Middle	N/A	N/A	Vulnerable	Protected	N/A	**Protected with User Support**
Network Eavesdroppers	De-anonymization	Vulnerable	Protected	Protected	Protected	Protected	Protected
	DNS Query Leakage	Vulnerable	Protected	Protected	Protected	Protected	Protected
	Man-in-the-Middle	Vulnerable	Protected	Protected	Protected	Protected	Protected
	Malicious Resolver	Vulnerable	Vulnerable	Protected	Partially Protected	**Protected**	Protected
	DNS spoofing	Vulnerable	Vulnerable	Protected	Protected	**Protected**	Protected
Server Eavesdroppers	De-anonymization	Vulnerable	Protected	Protected	Vulnerable	Protected	Protected
	DNS Query Leakage	Vulnerable	Vulnerable	Protected	Vulnerable	**Protected**	Protected
	Man-in-the-Middle	Vulnerable	Protected	Protected	Protected	Protected	Protected
	Malicious Resolver	Vulnerable	Vulnerable	Protected	Partially Protected	**Protected**	Protected
	DNS spoofing	Vulnerable	Vulnerable	Protected	Protected	**Protected**	Protected

Table 1: Evaluation of security measures for DoT and DoH with and without our eBPF solution. Only attacks with vulnerabilities to each threat are included.

simulated, rather than chosen from active servers, and connected to a virtual machine. Within this network, we created hooks for all four groups of potential privacy threats (Section 3). Instead of importing all of the Alexa 500 sites, the 50 most visited sites were loaded on the simulated name servers. Each of our simulated name servers had a cache of 15 addresses set to be served in approximately 0.03 seconds, as compared to an un-cached response of 0.1 simulated seconds [33][24]. Addresses were cycled in-and-out of the cache in a FIFO queue. All three eBPF tools (eBPF for DNS, eBPF for DoT, eBPF for DoH) were run on the test system separately and continuously analyzed the information shared with each attacker. On the virtual machine, we had two users: an experimental user and an attacker user. The experimental user followed a set pattern of queries though the Alexa top 50 sites. The attacking user was set to continuously query the core name servers for the top 50 websites and measures the response time.

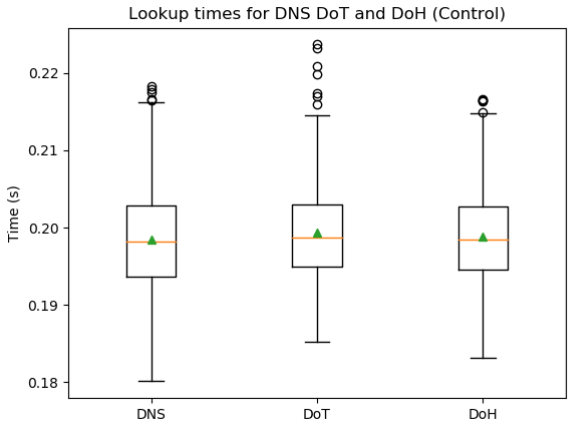
For each potential source of information exposure, we utilized a set of metrics to consider when information would be leaked to a theoretical attacker. For the network eavesdropper, any domain name it is capable of reading is considered exposed. For the server eavesdropper, every time the attacking user was able to predict a site that was present in the cache of the experimental users' queries, it was considered a leaked site. For the DoH server, since we are unable to directly edit the information, we instead use 1BD-improved and the EFF's Panopticlick [12], to measure the bits and compare them with our local calculation derived from Hermann et al. [18]. Finally, for the ISP simulation, we place bit counters on each of our four DNS servers which incremented every time a new URL was requested from the experimental user. Once a counter reaches 30 unique addresses, we consider the user deanonymized [18].

5.5 Privacy Experiment Results

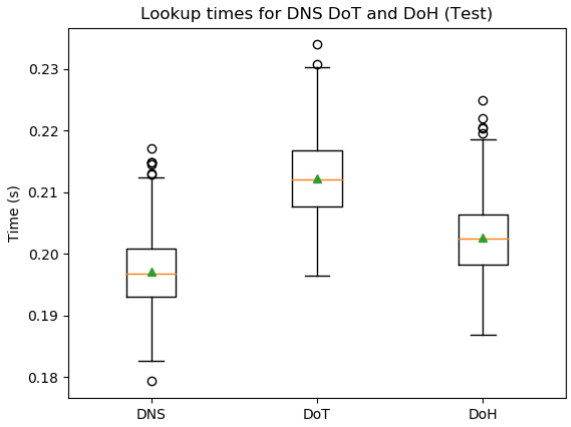
In regards to the first scenario, the network eavesdropper, we found that cleartext was compromised and that both DoT and DoH performed as expected by keeping the DNS transactions private through encryption. EncDNS was easily capable of defeating this as well, which is unsurprising given that this is a simple attack. The second scenario, the shared system attacker, had the same win condition. For every address it determined the user was querying, it was considered successful in breaching a users' privacy. It succeeded at a higher rate than the network eavesdropper and was capable of defeating the DoT implementation due to cache linkage over the shared stub resolver. However, both the EncDNS privacy tool and our method were capable of defeating the shared system attacker by spreading DNS queries over multiple systems.

The third experiment revealed issues with the traditional DoT and DoH approaches. For DoH privacy, a simple running comparison of our 'calculated' bits revealed and 'actual' bits revealed was conducted on the server-side. The 'actual' bits revealed were calculated using a 'truth' reference provided by EFF. We found that EncDNS underestimates the amount of information revealed due to not taking into account the browser user-agent string. In this case, eBPF provided the user with the number of leaked bits as well as updated information on the user-agent string, information not provided by other tools. This provides the user with more information about potential threats to anonymity, a vital concern for many users.

In regards to scenarios addressing the ISP, it was determined that EncDNS leaks considerably more information even if it is set to do a parallel resolution. It does not take into account what information the server already knows and thus, over longer periods, it will eventually leak all information about the user to the ISP. This is a significant threat to user privacy, as allowing the ISP to access all user information is of utmost concern. Using our tool, eBPF is aware of how much information it has given to each server, and is able to



a) No eBPF running



(b) Using eBPF

Figure 4: DNS lookup time measures of the Alexa top 500. Minimal overhead is observed, indicating good performance.

load balance the bits of information revealed to each resolver. The bits revealed were determined to be minimal, protecting the user.

5.6 Discussion

Our results are summarized in Table 1. Our tool provides the user with thorough protection in all cases. Using our solution, we have demonstrated that our eBPF tools allow users to quickly and efficiently protect users from data leakage and privacy attacks. Our tool provides an avenue for users to continuously change which DNS server they are using or to sandbox applications into their unique DNS server. This tool provides users with a greater amount of control over their data and provides a user-friendly method for privacy-conscious users. While DoH appears to provide protection in the majority of cases, it is important to consider the extreme vulnerabilities posed by the browser vendor not addressed by DoH. Using our tool, the user is protected in all cases with DoT and additionally provides support for DoH in regards to the browser vendor by providing users with a complete analysis of their privacy and ensuring that data leakage only occurs to the chosen DoH provider. Additionally, through experimental evaluation, it was determined that the browser can leak an arbitrary amount of information using the user-agent string. Since the browser controls the user-agent string, an arbitrary amount of data may be leaked to the vendor. The most adequate action suggested for this type of attack is to warn the user if their identification changes for any reason. Such warnings are provided to the user by our eBPF implementation. We demonstrate that such a solution does not impact application performance. Given that DNS providers now have the technological capabilities to extract specific user information from DNS traffic, we believe that users will be well served by this method, allowing users to shift their DNS providers regularly. Our system outperformed the DNS privacy tool, EncDNS, in privacy against the ISP and browser vendor, with comparable behavior for more common eavesdropping threats. Ultimately, it comes down to user preference and the level of trust, but this method provides a far greater amount of control to the user with minimal performance overhead.

6 CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this paper, we demonstrate a new way to help provide DNS privacy to users, though the use of eBPF. Using our method, a user can control their DNS traffic at the application level, which is currently a slow and time-consuming process using other methods. We demonstrate that it can be done without heavily impacting the performance of the applications. While eBPF is unable to invisibly interact with encrypted communications such as DoH, we instead provide monitoring and verification of the contents of the lookup. For privacy-conscious users, we recommended that they regularly change their DoH providers to lessen the impact of fingerprinting.

6.2 Future Work

In the future, we plan to expand our eBPF framework across additional DNS options. Of chief interest is in providing users with more control over their DNS queries. We hope to develop new techniques to analyze encrypted DNS traffic before transmission and potentially provide both filtering and alterations. Additionally, we aim to allow network administrators to recreate the functionality currently provided by passive DNS in a DoH and DoT filled world without compromising user privacy.

A question worth considering is at which point conscious privacy monitoring becomes too extensive, as there are many benefits to ISP DNS monitoring, such as network shaping and pre-fetching. While increasing user privacy is of vital importance, a completely private internet would probably not work as well, and may not be as enjoyable for the user. Awareness of the long-term impact should be considered when creating such tools, for both increased privacy and maintaining a coherent system.

ACKNOWLEDGMENTS

This work is partly supported by the project CONCORDIA that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 830927.

REFERENCES

[1] alexa. 2019. Top 500 sites on the web. https://www.alexa.com/topsites
[2] Gilberto Bertin. 2013. XDP in practice: integrating XDP into our DDoS mitigation pipeline. In *Technical Conference on Linux Networking, Netdev (vol. 2)*.
[3] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.. In *Ndss*. 1–17.
[4] Kevin Borgolte, Tithi Chattopadhyay, Nick Feamster, Mihir Kshirsagar, Jordan Holland, Austin Hounsel, and Paul Schmitt. 2019. How DNS over HTTPS is Reshaping Privacy, Performance, and Policy in the Internet Ecosystem. *Available at SSRN:https://ssrn.com/abstract=3427563* (2019).
[5] S. Bortzmeyer. 2015. *DNS Privacy Considerations*. RFC 7626. RFC Editor. https://tools.ietf.org/html/rfc7626
[6] S. Brack, R. Muth, S. Dietzel, and B. Scheuermann. 2018. Anonymous Datagrams over DNS Records. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. 536–544.
[7] Guy Bruneau and Rick Wanner. 2010. Dns sinkhole. *SANS Institute InfoSec Reading Room, Aug* 7 (2010).
[8] Jonas Bushart and Christian Rossow. [n. d.]. Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS. *CoRR* abs/1907.01317 ([n. d.]). arXiv:1907.01317 http://arxiv.org/abs/1907.01317
[9] A. Mankin S. Dickinson C. Huitema, M. Shore and J. Iyengar. 2019. Specification of DNS over Dedicated QUIC Connections. https://tools.ietf.org/id/draft-huitema-quic-dnsoquic-06.html [Online; accessed 01-October-2019].
[10] Katharine Daly. 2019. Add DoH UI setting. https://chromium-review.googlesource.com/c/chromium/src/+/1194946 [Online; accessed 01-October-2019].
[11] dnscrypt. 2019. dnscrypt. https://dnscrypt.info/ [Online; accessed 01-October-2019].
[12] EFF. 2019. Panopticlick 3.0. https://panopticlick.eff.org/
[13] Leblond Eri and Manev Peter. 2019. *White paper: Introduction to eBPF and XDP support in Suricata*. Technical Report. Paris, France. https://www.stamus-networks.com/wp-content/uploads/2019/07/suricata-ebpf-xdp-1.pdf
[14] Hannes Federrath, Karl-Peter Fuchs, Dominik Herrmann, and Christopher Piosecny. 2011. Privacy-preserving DNS: analysis of broadcast, range queries and mix-based protection methods. In *European Symposium on Research in Computer Security*. Springer, 665–683.
[15] Edward W Felten and Michael A Schneider. 2000. Timing attacks on web privacy. In *Proceedings of the 7th ACM conference on Computer and communications security*. 25–32.
[16] Brendan Gregg. 2019. eBPF. http://www.brendangregg.com/ebpf.html [Online; accessed 15-October-2019].
[17] Dominik Herrmann, Karl-Peter Fuchs, Jens Lindemann, and Hannes Federrath. 2014. Encdns: A lightweight privacy-preserving name resolution service. In *European Symposium on Research in Computer Security*. Springer, 37–55.
[18] Dominik Herrmann, Max Maaß, and Hannes Federrath. 2014. Evaluating the security of a DNS query obfuscation scheme for private web surfing. In *IFIP International Information Security Conference*. Springer, 205–219.
[19] P. E. Hoffman and P. McManus. 2018. DNS Queries over HTTPS (DoH). https://www.rfc-editor.org/rfc/rfc8484.txt [Online; accessed 01-October-2019].
[20] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. 2018. The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '18)*.
[21] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. 2019. Analyzing the Costs (and Benefits) of DNS, DoT, and DoH for the Modern Web. In *Proceedings of the Applied Networking Research Workshop (ANRW '19)*. 20–22.
[22] Facebook Incubator. 2019. A high performance layer 4 load balancer. https://github.com/facebookincubator/katran [Online; accessed 14-October-2019].
[23] IOVISOR. 2019. BPF Compiler Collection (BCC). https://github.com/iovisor/bcc
[24] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. 2002. DNS performance and the effectiveness of caching. *IEEE/ACM Transactions on networking* 10, 5 (2002), 589–603.
[25] D. Wing K. T. Reddy and P. Patil. 2017. *DNS over Datagram Transport Layer Security (DTLS)*. Technical Report. https://www.rfc-editor.org/rfc/rfc8094.txt [Online; accessed 01-October-2019].
[26] Dae Wook Kim and Junjie Zhang. 2015. You are how you query: Deriving behavioral fingerprints from DNS traffic. In *International Conference on Security and Privacy in Communication Systems*. Springer, 348–366.
[27] Steven McCanne and Van Jacobson. 1993. The BSD Packet Filter: A New Architecture for User-level Packet Capture *(USENIX'93)*. USENIX Association, Berkeley, CA, USA. http://dl.acm.org/citation.cfm?id=1267303.1267305
[28] Patrick McManus. 2018. Improving DNS privacy in Firefox. https://blog.nightly.mozilla.org/2018/06/01/improving-dns-privacy-in-firefox/ [Online; accessed 01-October-2019].
[29] Quentin Monnet. 2019. BPF helpers. https://qmonnet.github.io/bpf-helpers/out/bpf-helpers.html [Online; accessed 15-October-2019].
[30] S Pavithirakini, DDMM Bandara, CN Gunawardhana, KKS Perera, BGMM Abeyrathne, and Dhishan Dhammearatchi. 2016. Improve the Capabilities of Wireshark as a tool for Intrusion Detection in DOS Attacks. *International Journal of Scientific and Research Publications* 6, 4 (2016), 378–384.
[31] Paul Schmitt, Anne Edmundson, Allison Mankin, and Nick Feamster. 2019. Oblivious DNS: Practical privacy for DNS queries. *Proceedings on Privacy Enhancing Technologies* 2019, 2 (2019), 228–244.
[32] Sandra Siby, Marc Juárez, Claudia Díaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2019. Encrypted DNS -> Privacy? A Traffic Analysis Perspective. *CoRR* abs/1906.09682 (2019). arXiv:1906.09682 http://arxiv.org/abs/1906.09682
[33] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2019. Encrypted DNS–> Privacy? A Traffic Analysis Perspective. *arXiv preprint arXiv:1906.09682* (2019).
[34] Bromite Development team. 2019. Bromite. https://www.bromite.org/ [Online; accessed 01-October-2019].
[35] Ryu Project Team. 2017. RYU SDN Framework. https://osrg.github.io/ryu/
[36] JHC van Heugten. 2018. *Privacy analysis of DNS resolver solutions*. Ph.D. Dissertation. Master's thesis, University of Amsterdam.
[37] Dave Watson. [n. d.]. KTLS: Linux Kernel Transport Layer Security. In *netdevconf*. https://netdevconf.info/1.2/papers/ktls.pdf
[38] J. Heidemann A. Mankin D. Wessels Z. Hu, L. Zhu and P. E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). https://www.rfc-editor.org/rfc/rfc7858.txt [Online; accessed 01-October-2019].