Privacy-Conscious Threat Intelligence Using DNSBloom

✉ Roland van Rijswijk-Deij[*][†], Gijs Rijnders[‡][§], Matthijs Bomhoff[§] and Luca Allodi[‡]
[*]University of Twente, [†]NLnet Labs, [§]Tesorion, [‡]Eindhoven University of Technology
r.m.vanrijswijk@utwente.nl, gijs.rijnders@tesorion.nl, matthijs.bomhoff@tesorion.nl, l.allodi@tue.nl

***Abstract*—The Domain Name System (DNS) is an essential component of every interaction on the Internet. DNS translates human-readable names into machine readable IP addresses. Conversely, DNS requests provide a wealth of information about what goes on in the network. Malicious activity – such as phishing, malware and botnets – also makes use of the DNS. Thus, monitoring DNS traffic is essential for the security team's toolbox. Yet because DNS is so essential to Internet services, tracking DNS is also highly privacy-invasive, as what domain names a user requests reveals their Internet use. Therefore, in an age of comprehensive privacy legislation, such as Europe's GDPR, simply logging every DNS request is not acceptable.**

In this paper we present DNSBloom, a system that uses Bloom Filters as a privacy-enhancing technology to store DNS requests. Bloom Filters act as a probabilistic set, where a membership test either returns probable membership (with a small false positive probability), or certain non-membership. Because Bloom Filters do not store original information, and because DNSBloom aggregates queries from multiple users over fixed time periods, the system offers strong privacy guarantees while enabling security professionals to check with a high degree of confidence whether certain DNS queries associated with malicious activity have occurred. We validate DNSBloom through three case studies performed on the production DNS infrastructure of a major global research network, and release a working prototype, that integrates with popular DNS resolvers, in open source.

***Index Terms*—DNS; privacy; measurement; GDPR; threat detection; indicator-of-compromise**

I. Introduction

In modern networks, there is a constant arms race between network managers and miscreants that want to infiltrate the network, to deploy botnets, to infect users with malware or to phish their credentials. Consequently, network security professionals need to have a well-stocked toolbox to combat such adversaries. A well-known approach to threat detection is to monitor Domain Name System (DNS) queries. The DNS fulfills a key role for Internet services: it maps human-readable names to machine-readable IP addresses. Because DNS is so essential, malicious activity on a network oftentimes relies on the DNS in some way. This can be either just to map names to addresses, e.g., for URLs included in phishing e-mails, or more active abuse of the DNS, for instance as a command-and-control (C&C) channel for botnets.

A major problem with monitoring DNS queries on a network is that this is also extremely privacy-invasive [1], [2]. Because almost all network services rely on the DNS in some way, recording what DNS queries a user performs is highly revealing of their Internet use. In the age of ever stricter privacy legislation – think, for example, of Europe's General Data Protection Regulation (GDPR) [3] – simply recording all DNS traffic on a network is not considered *proportional* to the goal of safeguarding network security. Given, however, how valuable DNS query logs can be for network security, the following question is worth asking: *Can we track information about DNS queries without compromising on user privacy?*

In this paper we present DNSBloom, a system that uses Bloom Filters [4] as a privacy enhancing technology to track DNS queries. Bloom Filters were invented in the 1970s as a time- and space-efficient means to index databases. They act as a probabilistic set, where a membership test either confirms *certain* non-membership, or indicates *probable* membership with a low probability of false positives. Bloom Filters rely on hash functions to store information; as such, they never store the original information. DNSBloom leverages this property to protect user privacy while retaining useful detection properties. In essence, when using DNSBloom, a security professional can ask *if* a specific query for a known (malicious) domain name has occurred, but cannot obtain a set of *all* queries that occurred in the network. While this does not allow for real-time monitoring of threats, it does allow for tactical and strategic assessment of threats on a network: upon observation of threats (known as *indicators-of-compromise* – IoCs) using DNSBloom, security professionals can decide to deploy targeted monitoring for specific threats, thus achieving a proportional (e.g., in the sense of the GDPR) collection of data. Moreover, DNSBloom allows operators to keep track of DNS queries over time – in a privacy-conscious manner – and to look back in time to see if emerging threats have already occurred in their network.

To demonstrate its practical value, we validate the use of DNSBloom in three real-world scenarios at a major global research network. Furthermore, we implement a working prototype that seamlessly integrates with all major open source DNS resolver implementations. This prototype is released in open source, to foster reproducibility and future research.

Paper organization — the remainder of this paper is organised as follows. Section II provides background information on Bloom Filters and IoCs. Section III introduces the approach behind DNSBloom. In Section IV, we report on the evaluation of the DNSBloom prototype. Section V reflects on the results of our validation, and Section VI, discusses conclusions and provides an outlook on future research.

978-3-903176-15-7 © 2019 IFIP

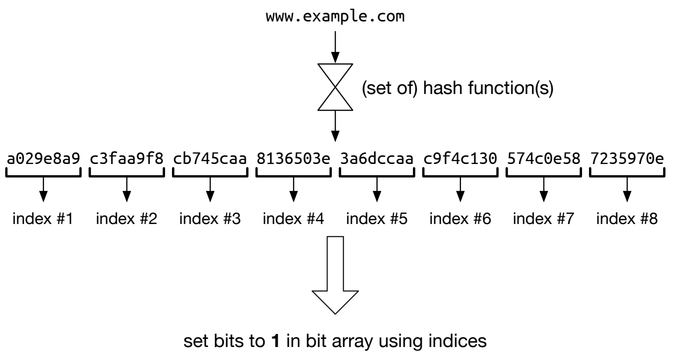

Fig. 1. Mapping an element to a Bloom Filter

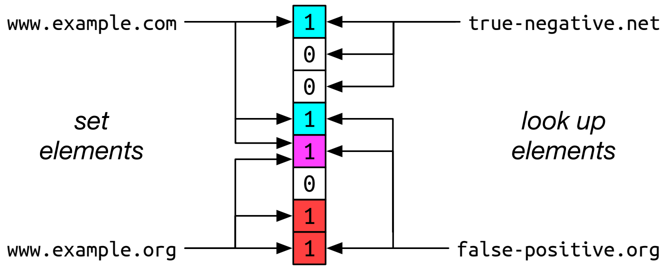

Fig. 2. Setting and looking up elements in a Bloom Filter

II. Background

A. Bloom Filters

Bloom Filters – named after their inventor, B.H. Bloom – were devised in the 1970s as a space-saving way to speed up lookups in databases [4]. The essential goal of a Bloom Filter is to function as a fail-fast lookup function that can quickly determine that an element is *not* in a set. This is useful, for example, in database indexes, where a Bloom Filter can quickly determine whether an element is in an index.

A Bloom Filter is implemented as an array of m bits, initially set to zero, which are set based on the output of k independent hash functions. To add an element to the filter, an element x is passed through each of the k hash functions, producing k hash values $h_1(x)\dots h_k(x)$. Each of these hash values is then treated as an index into the bit array m; this means that each value $h_n(x)$ must likely be mapped to the space $[0, m)$ to become a valid index. Every bit at index $h_1(x)\dots h_k(x)$ is then set to 1. Figure 1 shows an example of this process. In the figure, rather than using k independent hash functions, a SHA256 hash is used and split up into 8 separate offsets. This is equivalent to using 8 independent hash functions as SHA256 is a cryptographic hash function with uniform output [5].

To perform a lookup in a Bloom Filter, the same process is repeated. To look up element y, it is passed through the hash functions to produce k hashes $h_1(y)\dots h_k(y)$. Then, the values in the bit array m are checked based on the offsets derived from each hash value $h_n(y)$. If any of the bit values in m at an offset $h_n(y)$ is set to 0, then y is guaranteed not to be in the set described by the Bloom Filter. If all of the bits at offset $h_1(y)\dots h_k(y)$ are set to 1, then y is likely part of the set described by the filter, although there is a probability p_ϵ of a false positive. Figure 2 shows how element setting (left) and lookups (right) work. For the purpose of exposition, the reported example considers values $k = 3$ and $m = 8$; the arrows indicate the indices in m that $h_{1\dots3}$ map an input value to. The figure shows three properties of Bloom Filters. First, independent elements may hit the same bits in the filter during insertion (purple field). Second, if an element is not a member (top-right example), this is a *true negative*. Third, false positives are caused by the element mapping by chance to indices in m that were previously set to 1 by the insertion of one or more other elements.

The values for m and k are selected based on n, the expected number of elements to be inserted into the Bloom Filter, and on the desired false positive probability. In essence, there is a trade-off between the size of the filter m and the false positive probability p_ϵ. For a given k, m, n, the false positive probability has been shown in [6] to correspond to:

$$p_\epsilon \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \tag{1}$$

Furthermore, given a chosen m and an expected n, the optimal number of hash functions that minimises p_ϵ is:

$$k = \frac{m}{n}\ln 2 \tag{2}$$

The *actual* false positive probability depends on the actual number of bits set to 1 in the filter (s), and is computed as:

$$p_a = \left(\frac{s}{m}\right)^k \tag{3}$$

Bloom Filters have a number of attractive properties, two of which are relevant to this work. First, provided that two Bloom Filters A and B use the same parameters (that is: the same hash function(s), and the same values for k and m), then the union of the two filter states $A \cup B$ can trivially be computed using the bit-wise OR operation. Second, update and lookup operations on Bloom Filters execute in constant time, in particular they are constant in k and have complexity $\mathcal{O}(k)$.

B. Collecting DNS queries

Monitoring of the DNS for security purposes has many angles. In this paper, we focus on DNS traffic that is collected near or on DNS resolvers. Figure 3 shows the vantage points surrounding a DNS resolver. For this work, we focus on data collected at vantage points *A* and *B*. Data collected at vantage point *C* is typically of interest for large-scale passive DNS (pDNS) deployments [7], which have far fewer privacy concerns [8]. We consider pDNS as out of scope for this work.

To detect threats on a network, the simplest approach to collecting DNS traffic is to perform packet captures on the link on which incoming traffic from clients reaches the DNS resolver (vantage point *A*). The vast majority of DNS traffic is still plaintext, and such monitoring can be deployed using standard software that is independent of the DNS resolver

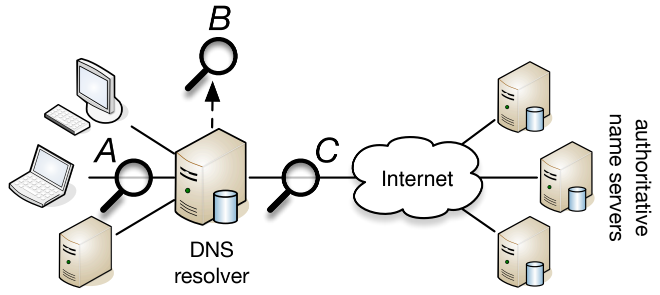

Fig. 3. Monitoring vantage points around a DNS resolver

implementation. This is changing with the standardization of DNS-over-TLS [9], that would make data collection from vantage point *A* impossible[1]. On the other hand, many DNS resolver implementations support direct collection of DNS query data from the running resolver software through the so-called `dnstap` interface[2] (vantage point *B*). This interface always outputs packets in cleartext.

Inspection of DNS queries from clients for security purposes roughly has two purposes. First, the detection of new types of threats, such as new botnets. The second purpose is signature-based detection of pre-identified threats. The work in this paper focuses on the second; based on pre-identified indicators of compromise (IoCs) in the form of domain names, DNSBLOOM identifies past and ongoing threats in a network.

C. Related Work

Numerous commercial solutions exist in the domain of signature-based threat detection from DNS data; academic work in this space almost exclusively focuses on the detection of new types of threats based on DNS traffic [10].

Focusing on the use of Bloom Filters in the network space, early work by Broder and Mitzenmacher [11] provides a survey of the use of Bloom Filters, focusing on performance improvements in network applications. A more recent survey by Geravand and Ahmadi [12] looks at the application of Bloom Filters for network security purposes. Much of the work surveyed in their paper focuses on the use of Bloom Filters to perform efficient data lookups. Some work, though, specifically uses Bloom Filters as a privacy-enhancing method, for example in the context of anonymous routing and packet-based attack attribution. A representative example is work by Zhu and Mutka [13], that uses Bloom Filters to enhance privacy for instant messaging notification.

With respect to attack detection, Bloom Filters have been used as a means to more efficiently detect DNS amplification DDoS attacks. Sun et al. [14] propose a system that uses Bloom Filters to efficiently match DNS queries to responses. Their approach leverages the assumption that valid DNS traffic consists of matching query/response pairs, and that traffic that only contains responses is anomalous and indicative of an attack. Di Paola and Lombardo [15] propose a simplified approach (compared to Sun et al.) for detecting DNS amplification attacks, with a lower false positive rate, due to the use of different filter parameters. Bloom Filters have also been used to enhance the performance and memory footprint of signature-based spam filtering, for example by Yan and Cho [16].

More specific to DNS, the popular open source PowerDNS Recursor resolver implementation uses a Bloom Filter to track newly observed domains[3]. Tracking such domains can, for example, help detect phishing and spam e-mails, as these often use "fresh" domains directly after they have been registered in the DNS (and thus have likely not been observed on resolvers before they are used for a spam run) [17].

Finally, as we will also discuss later on, Bloom Filters are not themselves immune to attacks. Gerbet et al. [5] describe various types of attacks on Bloom Filters that aim, for example, to oversaturate filters to artificially raise the probability of a false positive. Their work also explains how implementers can design their systems to be more resilient against the attacks discussed in the paper.

III. APPROACH

A. Goals and Challenges

With DNSBLOOM we want to operationalise tracking of DNS query information without compromising on user security by infringing the confidentiality of highly privacy-sensitive user information [1], [18]. Before specifying this goal in more detail, we should first consider *why* we wish to track this information over time. As stated above, knowledge of DNS queries that occurred in the past can prove useful if a new indicator of compromise is discovered (in other words: a new domain name is associated with malicious activity). A query log can be used to see if this query name has already been observed on the network, indicating, for example, that some host has been infected, or some users clicked on suspicious links in a phishing e-mail. Consider, for example, the outbreak of the WannaCry ransomware in 2017 [19]. Months after this ransomware was first observed, and only after a massive outbreak, a security researcher discovered that a very specific DNS query that the ransomware performed was actually a so-called 'kill switch' [20]. Using a log of past queries, network administrators could have detected past activity related to this ransomware by looking for the kill-switch domain in their logs.

The question we ask is how can we do this *without* collecting massive amounts of otherwise privacy-sensitive data from the users? Ideally, we would still like to retain useful information to make strategic and tactical decisions such as: should network administrators search for, or start monitoring for a particular new threat? Or: when was an IoC associated with a high profile threat first observed, and in which parts of the network?

More specifically, we identify the following six goals:

G1 Log DNS queries over potentially long periods of time.

G2 Avoid storing personally identifiable information (PII).

[1]For public DNS-over-TLS services, see https://dnsprivacy.org/.
[2]http://dnstap.info/

[3]See https://doc.powerdns.com/recursor/settings.html under `new-domaintracking`.

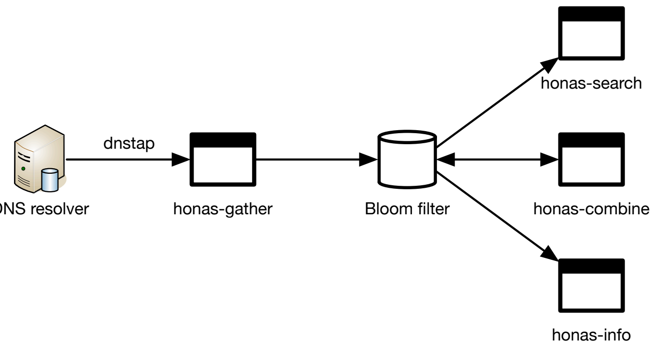

Fig. 4. Overview of DNSBLOOM's architecture

G3 Efficiently store and query logged data.

G4 Determine the approximate time a query was observed.

G5 Distinguish between queries made by clients from different network segments.

G6 Aggregate historic data.

Bloom Filters provide a natural fit to match these goals. In particular, Bloom Filters can help us achieve most of the goals stated above because they:

- Can efficiently store an arbitrary number of items if their parameters are dimensioned correctly (Goals ***G1*** and ***G3***).
- Are not trivially enumerable, and do not store any recoverable version of the original data (Goal ***G2***).
- Can be updated and queried in constant time (Goal ***G3***).
- Can trivially be aggregated provided that the filters to be aggregated share the same parameters (Goal ***G6***).

The other two goals (***G4*** and ***G5***) can trivially be implemented on top of Bloom Filters as discussed in Section III-B.

B. Prototype

To test our approach, we designed and implemented a prototype to meet the goals specified in the previous section. We called this prototype "DNSBLOOM". The prototype is based on an open source application to collect DNS queries in a Bloom Filter, called "HONAS". This application was initially internally developed at Tesorion[4] as a means to gather telemetry on threats for malware research purposes in a privacy-friendly way. The application was converted to open source with funding from SURFnet[5], the National Research and Education Network in the Netherlands [21]. DNSBLOOM is based on HONAS and extends it to make it ready for deployment in an ISP context. In the remainder of this section, we discuss the design and implementation of DNSBLOOM.

Figure 4 shows an overview of the entire prototype. As the figure shows, the system consists of multiple applications. From left-to-right and top-to-bottom, the applications shown perform the following functions: `honas-gather` receives DNS queries from a running DNS resolver using the `dnstap`[2] interface, and records these in a Bloom Filter. A more detailed description of this application is provided below. The `honas-search` application can be used to query stored Bloom Filters. As input, it takes a query formatted as a JSON object, the results of this query are also output as a JSON object. This makes the application simple to integrate in a web service or application. The `honas-combine` application can aggregate two stored Bloom Filters into a new filter that combines the data from both. Before performing the aggregation, it checks whether the filters have the same parameters so they can safely be combined. Finally, the `honas-info` application provides descriptive information about a filter, including the actual false positive rate p_a, based on the fill rate of the filter (see Equation 3).

The collector application `honas-gather` forms the core of DNSBLOOM. It implements a configurable Bloom Filter implementation, that allows the user to specify the filter parameters k and m. Rather than implementing k independent hash functions, `honas-gather` uses a SHA256 hash, from which it extracts k array indices. As mentioned before, this is functionally equivalent to using k different hash functions [5].

To assist in determining the correct parameters, `honas-gather` implements a dry-run mode, during which it monitors the number of distinct elements n inserted into the filter. To estimate this number, we use the HyperLogLog++ algorithm [22]. The application has been designed to start a new Bloom Filter at regular, configurable intervals. By default, it will start a new filter every hour. The dry-run process determines n on an hourly basis. In addition to this, to facilitate goal ***G6*** (aggregating historic data), it also determines n on a daily basis. Recall that Bloom Filters can only be aggregated into a single filter if all filters use the same parameters. Given that aggregation increases the probability of a false positive, the dry-run also reports the daily count for n. Thus, if we aggregate hourly filters into a single daily filter, we can still guarantee a reasonable false positive probability if we configure k and m based on the aggregate value for n over an entire day. During the dry-run process `honas-gather` outputs the hourly and daily maximum values for n, and provides recommendations for configuring m and k for p_ϵ values of 10^{-3}, 10^{-4} and 10^{-5}. In the prototype, these recommendations are based on the observed n, rounded up to the nearest 100,000 and with an added 10% extra margin.

Once the application is configured, it can start collecting data. Rather than just storing the query names observed in DNS queries, `honas-gather` also stores the individual labels in a DNS query name (except for the top-level domain) and stores the full second-level domain name. The reason we do this is to facilitate detection of more complex indicators of compromise. For example: take malware that uses a so-called Domain Generation Algorithm (DGA), that generates seemingly random domain names for command-and-control servers. The DGA-generated parts of the domain name may occur in different places in a domain name [23]. If we want to be able to detect these, we also need to be able to search for individual labels of a domain name in the Bloom Filter.

Finally, to satisfy goal ***G5*** our prototype implementation has

[4]https://tesorion.nl/en/
[5]https://www.surf.nl/en/about-surf/subsidiaries/surfnet

TABLE I
EXAMPLE OF DATA INSERTED INTO A BLOOM FILTER

Description	Value	Description	Value
FQDN	`evil.domain.com`	FQDN	`Organisation A@evil.domain.com`
2nd-level	`domain.com`	2nd-level	`Organisation A@domain.com`
Label 1	`evil`	Label 1	`Organisation A@evil`
Label 2	`domain`	Label 2	`Organisation A@domain`

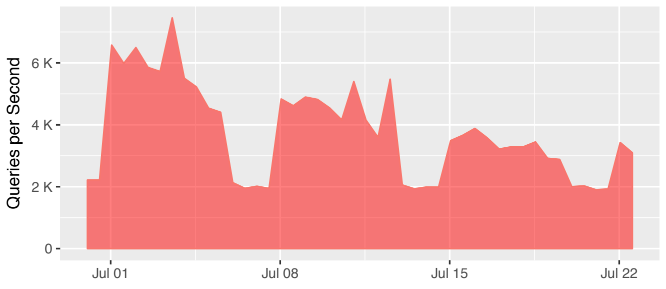

Fig. 5. SURFnet DNS resolver traffic during validation

functionality to map queries from certain IPv4 or IPv6 network prefixes to organisational entities. It does this by matching source IP addresses in incoming queries. The entity that a query is mapped to is then pre-pended to all of the data that is inserted into the filter. For example, Table I shows what information is added to the Bloom Filter if a query for `evil.domain.com` arrives from a prefix that is mapped to 'Organisation A'. As the table shows, the fully qualified domain name (FQDN), second-level domain (SLD) and labels are added directly, and with the entity name prefixed. The direct entry can be used to perform a fail-fast lookup, for instance, if many different entities exist in the network, to prevent having to perform a lookup for every entity all the time. To meet goal ***G4***, our prototype cannot simply link queries with timestamps as this association is lost in the Bloom filter; instead, DNSBLOOM aggregates queries in time slots of predefined length that provide an approximate indication of time of occurrence.

IV. VALIDATION

In order to validate our approach, we deployed the DNSBLOOM prototype at SURFnet, which operates production DNS resolvers for the higher education and research sector in the Netherlands. Our prototype relied on data from all of SURFnet's production resolvers, which on average see in the order of 5,000 to 10,000 queries per second at peak times and roughly 200,000 unique client IPs per day. Thus, our validation environment is likely comparable to a medium-size ISP. We evaluated the performance of the prototype over a period spanning three weeks, from July 1st to July 23rd, 2018. Figure 5 shows the average query load over the validation period. Note that the academic summer holiday starts in early July, hence the slow decline in query load.

As discussed in Section III-B, we first needed to determine suitable parameters k and m to configure our Bloom Filter. In order to do this, we ran the prototype in dry-run mode for one week. We ended up picking $k = 10$ and $m \approx 491$ Mbits to get an estimated p_ϵ of 10^{-3} (in other words: a false positive

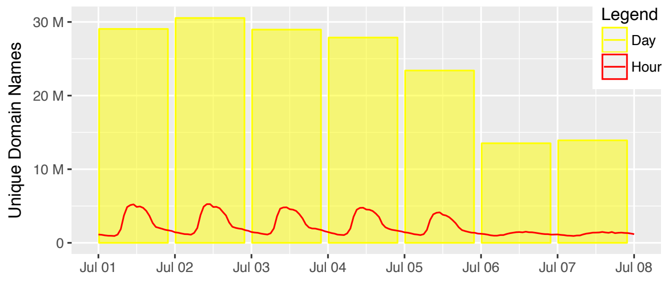

Fig. 6. Observations for n per hour vs. per day

probability of 1 in 1,000). We used the daily estimate for n, because as Figure 6 shows, the number of distinct queries is significantly higher when aggregated over a whole day.

A. Scenarios

In order to evaluate the performance of the prototype, we defined three test scenarios. Ethical and privacy implications of the data collection are discussed in Section V. All three scenarios reflect common threat detection situations encountered by security personnel at SURFnet.

Scenario 1: Booters — Our first scenario concerns so-called *Booters*. These are websites that provide DDoS attack capabilities on demand, allegedly to enable their users to 'stress-test' their own network [24]. Since 2012, SURFnet observes an increasing trend of (often young) students using Booters to attack their schools, for example to disrupt online teaching or exams. Because of the severity of this problem, SURFnet is interested in learning more about Booter attacks. As they suspect these attacks to be inside jobs, SURFnet's privacy officer gave permission to monitor for DNS queries specific to Booter websites, based on a blacklist of Booters [25]. Then, if an (attempted) attack is observed, the query logs could be consulted to check if a DNS query for a Booter was observed on the school under attack around the time of said attack.

Scenario 2: Spam filtering — The second scenario looks at spam filtering. SURFnet runs a spam filtering service for its constituency, which processes roughly 10 million mails per day. One of the key functions of this service is to check the IP addresses of mail-sending hosts against IP blacklists. These blacklists contain IPs that in some form have been associated with spam operations. As ground truth, we receive a daily list of blacklist hits (that is: sending IP addresses that appeared on one or more blacklists), and check for DNS queries of the reverse DNS names associated with these IPs. Since SURFnet's mail filtering service uses the same DNS resolvers used by DNSBLOOM, we expect to observe these queries. In addition to this, we also expect to observe queries for these blacklisted IPs from other clients on the network, e.g., from universities or other institutions that run their own mail service and do not rely on SURFnet's spam filtering.

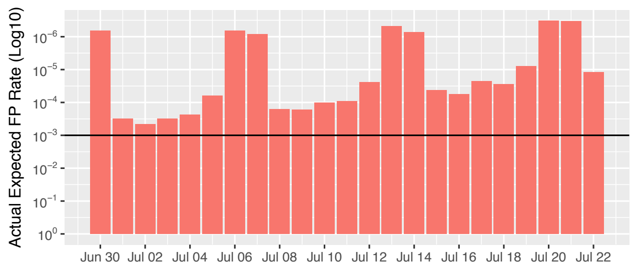

Fig. 7. Actual expected false positive probability (p_a) of 24h Bloom Filters

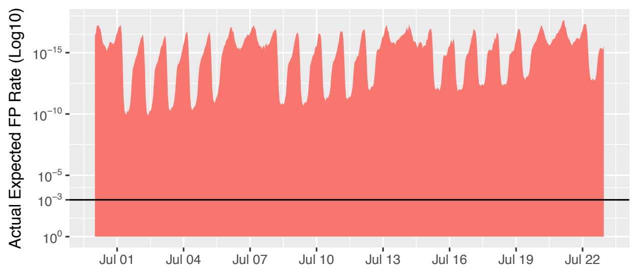

Fig. 8. Actual expected false positive probability (p_a) of hourly Bloom Filters

Scenario 3: National Detection Network — Our third, and final scenario concerns the so-called National Detection Network (NDN) [26]. This is a collaboration between the Dutch National Cyber Security Centre and key players in important sectors of Dutch society. The goal of NDN is sharing of high-profile indicators-of-compromise. In the NDN, these IoCs are shared through a Malware Information Sharing Platform (MISP) [27]. Many of the IoCs in the MISP contain threats that can be identified based on domain names. Up until now, SURFnet has not been able to use this information, though, as recording all DNS traffic to its resolvers simply to detect potential IoCs from the NDN was deemed too privacy-invasive. In this final scenario, we use DNSBLOOM to detect threats shared through the NDN MISP. To also collect ground truth in this scenario, we use a simple process. We first look for threats that re-occur in multiple Bloom Filters (over multiple hours). Then, we check the associated threat in the MISP, to see if the threat is serious. If it is, we then start specific monitoring for queries associated with this threat, as SURFnet's privacy policy allows such monitoring for network security purposes.

B. Results

We first focus on the performance of the Bloom Filters. Starting with CPU workload, during the validation phase our prototype collected data from three operational DNS resolvers and never exceeded a single-core CPU load of 10%. Based on this, we argue that it is perfectly feasible to run DNSBLOOM as a side process directly on the DNS resolver. Recall that – just like insertions – lookups can be performed in constant time $\mathcal{O}(k)$, this means that we expect lookup performance to be similar to insertion performance. Indeed, an unoptimised test run that loads the Bloom filter from disk for every run takes an average of 1.3 seconds to perform 230,000 lookups over 1,000 runs.

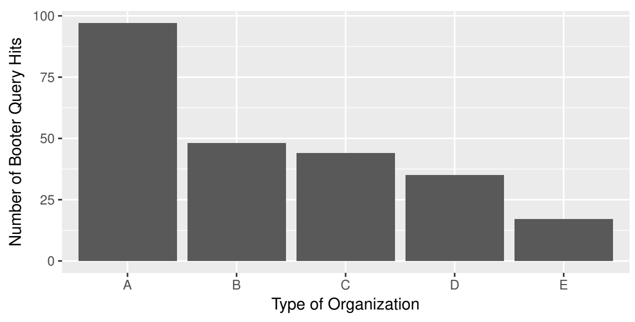

Fig. 9. Booter queries grouped per type of institution on SURFnet's network

We also checked the actual false positive rate for Bloom Filters aggregated per day. Recall our target p_ϵ was 10^{-3} for daily aggregate filters. As Figure 7 shows, during the validation the expected false positive rate for daily aggregate Bloom Filters stayed well below this target. Given that the hourly filters need to use the same parameters in order for aggregation to be possible, the p_a is much lower for the individual hourly filters, as illustrated by Figure 8. This means that the estimation process during the dry-run of the prototype performed well and accurately suggested values for k and m. We do observe a gradual decrease in the actual false positive probability p_a over time. This is likely due to the decrease in query volume, also observed in Figure 5. While this implies a *lower* probability of a false positive, thus increasing the reliability of lookups, a p_a that is far below the target can also lead to wasteful use of storage space, as the filter size m could have been smaller.

Scenario 1 — To evaluate how the Bloom Filters performed in the Booter scenario, we systematically checked every Bloom Filter collected over the validation period for queries to domain names on the Booter blacklist [25]. Then, to verify that DNSBLOOM performed as expected, we used the separate query log that directly recorded DNS queries containing names on the Booter blacklist against the detections made using DNSBLOOM. Over the validation period, we observed 103 queries for Booter domains, both in the ground truth and in the Bloom Filters. Moreover, we did not observe any false positives when querying the Bloom Filters. That is: even though we queried the Bloom Filters for the presence of queries for the full Booter blacklist, the hits reported exactly matched the ground truth. This is likely due to the very low false positive probability of the hourly Bloom Filters (Figure 8).

To illustrate what more can be done with the query data from the Bloom Filters, we mapped the Booter queries observed during the validation window to specific types of connected institutions[6] on the SURFnet network. Figure 9 shows the result of this mapping. It is clear that the majority of Booter queries chiefly come from one type (A) of institution on the network. Because of the sensitive nature of this information, we have chosen not to identify specific sectors of SURFnet's constituency in this paper.

[6]E.g. universities, research institutes, vocational education, etc.

Scenario 2 — In the spam filtering case, the number of IP blacklist hits received from the mail filtering service varied between ten and fifteen thousand IPs per day. For each address on this list, we checked if the corresponding reverse DNS name was present in the Bloom Filters. We were surprised to find a slight discrepancy, with up to 4% of IPs not showing up in the Bloom Filters. Further investigation showed that this was likely due to two reasons. First, the mail filtering service uses one additional DNS resolver for which we were not storing data in the Bloom Filters; this means we likely missed some reverse DNS queries. Second, each processing host of the mail filtering service also runs a local DNS resolver with a local cache, thus, some reverse DNS queries may have been answered from the local cache, and did not end up in the Bloom Filters. This limitation is something that needs to be considered especially in larger networks, where a setup with local DNS resolvers forwarding queries to more centralised systems is common.

We used the stored Bloom Filters to check if network segments belonging to connected institutions on the SURFnet network also sent reverse DNS requests for blacklisted IP addresses. This yielded a long list of hits, to which we applied the Pareto principle and examined those network segments responsible for 80% of the total number of observed hits. The institutions on those network segments run their own mail services, rather than relying on SURFnet's mail filtering service. Each of these institutions performed reverse DNS queries for between 2.4% and 13.1% of the IP blacklist hits from the mail filtering service. This suggests that there is a reasonable suspicion that the mail services for these institutions also interacted with the blacklisted IPs on the same day. While this is not necessarily directly useful information, the fact that it was possible to perform this analysis opens up other options that we will discuss in Section V.

Scenario 3 — For the final scenario, monitoring threats from the MISP belonging to the National Detection Network (NDN), we used a two-stage process: we regularly exported the set of domain names associated with threats from the MISP and checked these domains against the Bloom Filter. Then, we inspected the set of hits and selected those that regularly occur in filters for different time periods. For those hits, we looked up the associated threat in the MISP, and if the threat was deemed to be sufficiently serious, we installed specific monitoring to look for actual DNS queries for these threats, to obtain ground truth. Over the validation period, we did this for five different threats. Comparison of the ground truth to the Bloom Filters showed that the detection works as expected and that the queries we observe in the ground truth also show up in the filters. We observed no false positives for the domains for which we had ground truth, again, likely because of the very low false positive rate for the hourly filters (Figure 8).

We want to highlight two particular aspects of this scenario. First, using the DNSBLOOM prototype, SURFnet was able to use the threat information from the NDN MISP for the first time to perform detections. This had previously not been done because inspecting and storing all DNS traffic to monitor for these threats was deemed too large an invasion of user privacy. The fact that this is useful is illustrated by the number of unique threats we detected on a daily basis during the validation period, as shown in Figure 10. Second, during our validation the detection also proved to be directly operationally useful. One of the five threats we detected was the apparent presence of the WannaCry malware [19]. After we had installed specific monitoring for this threat, we found a single host querying the kill-switch domain at regular intervals. We notified SURFnet's incident response team, SURFcert, about this threat.

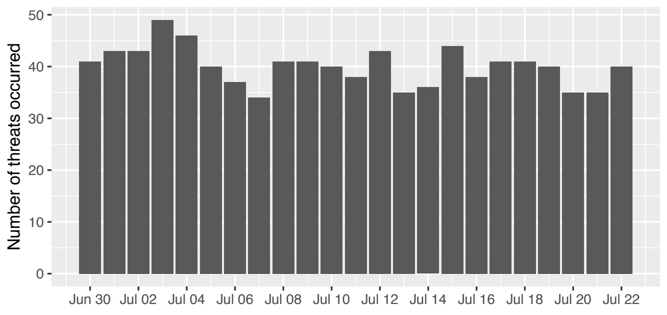

Fig. 10. Daily number of unique threats detected based on NDN IoCs

V. DISCUSSION

The evaluation of the scenarios in the previous section demonstrates that DNSBLOOM performs well in practice. Our prototype was easily able to handle a production workload on a major research network, while requiring only modest resources. While performance was not a primary goal, it certainly makes the system more attractive for operators. The system also reliably performed detections matching the ground truth, without evidence of false positives. This does, of course, not guarantee that false positives will never occur in the future.

Our primary goal was to provide *privacy-friendly* threat intelligence based on DNS query data. We have demonstrated that using Bloom Filters, we can achieve this goal. This can make DNSBLOOM a valuable new tool for security professionals, especially considering increasingly strict privacy legislation, such as the European Union's GDPR [3]. While a full discussion of DNSBLOOM in the context of the GDPR is out of scope for this paper, we note that in informal discussions with SURFnet's legal and privacy department we learned that they believe the system provides sufficient privacy guarantees to be able to say that it does not store personally identifiable information (PII). As a consequence, restrictions on how long data can be stored – stemming from the GDPR – do not apply, and there is, in principle, no time limit to how long data can be retained. For threat intelligence, this can be incredibly valuable, as it allows security professionals to look back in time for recently discovered threats. We note that another benefit may be that the Bloom Filter data can be used for other purposes as well, such as for academic research. In times of the GDPR, it is becoming increasingly hard for network operators to share data with researchers. The Bloom Filters collected by DNSBLOOM, at least *can* be shared.

Limitations — We note that our approach has limitations, that stem from our use of Bloom Filters. First, we have to deal with a small false positive probability when we check whether a query has been performed. Given that a hit in the Bloom Filter implies that an IoC has been triggered, and follow-up measures may be put in place based on such an observation, it is important to consider the possible adverse impact of a false positive. For example: a false positive for a serious IoC may cause an unwarranted wild goose chase that consumes valuable time of security staff. Second, while the use of Bloom Filters provides a significant privacy benefit, there is still a small privacy risk. If an operator has knowledge of a highly specific DNS query that is likely only performed by a specific person (for example, for a personal page that no one else visits), then it is still possible to track presence of this person on the network at certain time periods, based on this query being present in a Bloom Filter. We note though, that – under the condition that the filter contains data from multiple users – it is still impossible to then correlate this identifying query with any other queries performed by the user, as the Bloom Filters are not trivially enumerable. Third, in our current prototype, we use a single filter to store queries from all network segments. This introduces a risk that an attacker in a single network segment can perform a pollution or saturation attack [5] on the filter that then makes the entire filter unusable. If this is a real risk in a network in which DNSBLOOM is deployed, then a possible mitigation is to use separate Bloom Filters for different network segments. Effectively, this trades efficiency and storage space for the risk that a single user in a single network segment can render filters unusable. Finally, a current limitation of the prototype is that while it can confirm that a certain query was observed, it is not possible to say or estimate how many times that query was observed from a specific Bloom Filter. This issue can be overcome, though, for example through the use of a counting Bloom Filter [28].

Ethical Considerations — The goal of the DNSBLOOM system discussed in this paper is, of course, to obtain threat intelligence in a privacy-friendly manner. Yet to validate that our system works as designed, we needed ground truth in the form of direct logs of actual DNS queries. In order to do this in an ethical manner, we: 1) minimised the amount of ground truth we collected, 2) only collected ground truth if the existing privacy policy of the network where we collected data allowed such collection and 3) we deleted the privacy-sensitive query logs as soon as we completed our validation. We note that in actual operation, the standard formula (Equation 3) for estimating the false positive rate of a Bloom filter suffices. Thus, the performance in terms of false positive rate can be estimated in a GDPR-compliant manner when using DNSBLOOM in a production environment.

Open Source — In order for others to experiment with DNSBLOOM and to foster reproducibility of our research, we release our prototype under a permissive open source license[7].

[7]https://github.com/SURFnet/honas

VI. CONCLUSIONS AND FUTURE WORK

Keeping track of the DNS queries performed by clients in a network is a valuable tool for security professionals to gain threat intelligence about the network. Yet simply logging all DNS queries is highly privacy-invasive, and it is highly doubtful if this is an acceptable practice in the age of increasingly strict privacy legislation, such as Europe's GDPR. Therefore, at the start of this work we asked ourselves: *can we track information about DNS queries without compromising on user privacy?* To answer this question, we presented DNSBLOOM, a system that leverages key properties of Bloom Filters, such as the fact the data entered into a Bloom Filter is not trivially enumerable, to provide privacy-friendly DNS-based threat intelligence. In addition to this, the attractive performance characteristics of Bloom Filters make DNSBLOOM fast and lightweight. Using three real-world scenarios that are examples of typical threat intelligence tasks performed by security professionals, we tested DNSBLOOM in practice, in a production setting on data from the DNS resolvers of a major global NREN. These tests show that DNSBLOOM is a promising approach that enables threat detection that could otherwise not be performed, because the alternative – directly logging DNS queries – is considered too great a violation of user privacy.

Future Work — a key next step for DNSBLOOM is to further develop the code to a production quality level, and to deploy it in production. This will allow for a longer period of evaluation, and will likely yield new applications of the technology. Possible future applications are, for example, to use the collected data in filters to study co-occurrence of queries, an approach that has already been used in other work (e.g. [29]) to detect entirely new threats. In terms of improving our prototype, one of the key features we are considering is automated tuning of filter parameters. As query loads change over time, it is hard to choose good filter parameters that guarantee a low false positive rate over long periods (unless a gross over-estimation is applied). While updates to filter parameters limit aggregation options, if they are not too frequent, a good balance can be struck between re-calibrating filter parameters and maintaining a sufficient capability to aggregate filters over hours or days. Finally, we note that while this work and previous work has shown the promising privacy-preserving properties of Bloom Filters, there is a need for a more formal analysis of the privacy guarantees that Bloom Filters can offer. Work by Bianchi et al. [30] provides a good starting point, but further study is warranted for systems such as DNSBLOOM.

ACKNOWLEDGEMENTS

This work was funded by SURF, the Netherlands collaborative organisation for ICT in higher education and research. We thank Wim Biemolt, Bart Geesink and Remco Poortinga - van Wijnen, from SURFnet, for their valuable feedback on our work. We also thank the anonymous reviewers for their feedback.

References

[1] S. Bortzmeyer, "RFC 7626 - DNS Privacy Considerations," https://tools.ietf.org/html/rfc7626, 2015.

[2] D. Herrmann, C. Banse, and H. Federrath, "Behavior-Based Tracking: Exploiting Characteristic Patterns in DNS Traffic," *Computers and Security*, vol. 39, pp. 17–33, 2013.

[3] European Union, "Regulation 2016/679 of the European parliament and the Council of the European Union," *Official Journal of the European Communities*, vol. 59, no. May 2016, pp. 1–88, 2016.

[4] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] T. Gerbet, A. Kumar, and C. Lauradoux, "The Power of Evil Choices in Bloom Filters," in *Proceedings of the 45th International Conference on Dependable Systems and Networks (DSN 2015)*. Rio de Janeiro, Brazil: IEEE Computer Society, 2015, pp. 101–112.

[6] K. Christensen, A. Roginsky, and M. Jimeno, "A New Analysis of the False Positive Rate of a Bloom Filter," *Information Processing Letters*, vol. 110, no. 21, pp. 944–949, 2010.

[7] F. Weimer, "Passive DNS Replication," in *Proceedings of the 17th FIRST Conference (FIRST 2005)*, Singapore, Singapore, 2005.

[8] J. M. Spring and C. L. Huth, "The Impact of Passive DNS Collection on End-user Privacy," in *Proceedings of the SATIN 2012 Workshop*, Teddington, UK, 2012.

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

[10] R. Dodopoulos, "DNS-based Detection of Malicious Activity," Master's thesis, Eindhoven University of Technology, 2015.

[11] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[12] S. Geravand and M. Ahmadi, "Bloom Filter Applications in Network Security: A State-of-the-Art Survey," *Computer Networks*, vol. 57, no. 18, pp. 4047–4064, 2013.

[13] D. Zhu and M. Mutka, "Sharing Presence Information and Message Notification in an Ad Hoc Network," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*. Fort Worth, TX, USA: IEEE Computer Society, 2003, pp. 351–358.

[14] C. Sun, B. Liu, and L. Shi, "Efficient and Low-Cost Hardware Defense against DNS Amplification Attacks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2008)*, 2008, pp. 2062–2066.

[15] S. Di Paola and D. Lombardo, "Protecting against DNS Reflection Attacks with Bloom Filters," in *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2011)*, ser. Lecture Notes in Computer Science, T. Holz and H. Bos, Eds. Springer Berlin Heidelberg, 2011, vol. 6739, pp. 1–16.

[16] J. Yan and P. L. Cho, "Enhancing Collaborative Spam Detection with Bloom Filters," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC 2006)*. Miami Beach, FL, USA: IEEE Computer Society, 2006, pp. 414–425.

[17] Farsight Security, "Newly Observed Domains: threat protection from new domains," 2018. [Online]. Available: https://www.farsightsecurity.com/solutions/threat-intelligence-team/newly-observed-domains/

[18] M. Torrisi, "Personal Data in the DNS," 2017. [Online]. Available: https://dyn.com/blog/personal-data-in-the-dns/

[19] Symantec Security Center, "Wannacry Ransomware Write-up," 2017. [Online]. Available: https://www.symantec.com/security-center/writeup/2017-051310-3522-99

[20] L. H. Newman, "Accidental 'Kill Switch' Slowed Friday's Massive Ransomware Attack," *WIRED*, May 2017. [Online]. Available: https://www.wired.com/2017/05/accidental-kill-switch-slowed-fridays-massive-ransomware-attack/

[21] R. van Rijswijk-Deij, M. Bomhoff, and R. Dolmans, "Let a Thousand Filters Bloom: DNS-Based Threat Monitoring That Respects User Privacy," in *Proceedings of GÉANT TNC18*. Trondheim, Norway: GÉANT, 2018.

[22] S. Heule, M. Nunkesser, and A. Hall, "HyperLogLog in Practice," in *Proceedings of the 16th International Conference on Extending Database Technology - EDBT '13*. Genoa, Italy: ACM Press, 2013, pp. 683–602.

[23] U. Sternfeld, "Dissecting Domain Generation Algorithms Eight Real World DGA Variants," Cyberreason, Tech. Rep., 2016. [Online]. Available: http://go.cybereason.com/rs/996-YZT-709/images/Cybereason-Lab-Analysis-Dissecting-DGAs-Eight-Real-World-DGA-Variants.pdf

[24] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters - An Analysis of DDoS-as-a-Service Attacks," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*. Ottawa, CA: IEEE, 2015, pp. 243–251.

[25] J. J. Santanna, R. D. O. Schmidt, D. Tuncer, J. De Vries, L. Z. Granville, and A. Pras, "Booter Blacklist: Unveiling DDoS-for-Hire Websites," in *Proceedings of the 12th International Conference on Network and Service Management (CNSM 2016)*. Montréal, Canada: IEEE Computer Society, 2016, pp. 144–152.

[26] National Cyber Security Centre, "The National Detection Network." [Online]. Available: https://www.ncsc.nl/english/Cooperation/national-detection-network.html

[27] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform," in *Proceedings of the 3rd ACM Workshop on Information Sharing and Collaborative Security (WISCS 2016)*. ACM, 2016, pp. 49–56.

[28] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "An Improved Construction for Counting Bloom Filters," in *European Symposium on Algorithms (ESA 2006)*, ser. Lecture Notes in Computer Science, Y. Azar and T. Erlebach, Eds. Zürich, Switzerland: Springer Berlin Heidelberg, 2006, pp. 684–695.

[29] K. Sato, K. Ishibashi, T. Toyono, H. Hasegawa, and H. Yoshino, "Extending Black Domain Name List by Using Co-occurrence Relation between DNS Queries," in *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats (LEET 2010)*. San Jose, CA, USA: USENIX Association, 2010.

[30] G. Bianchi, L. Bracciale, and P. Loreti, ""Better Than Nothing" Privacy with Bloom Filters: To What Extent?" in *International Conference on Privacy in Statistical Databases (PSD 2012)*, ser. Lecture Notes in Computer Science, J. Domingo-Ferrer and E. Tinnirello, Eds. Palermo, Italy: Springer Berlin Heidelberg, 2012, vol. 7556, pp. 348–363.