Whitelists that Work: Creating Defensible Dynamic Whitelists with Statistical Learning

Renée Burton
Cyber Intelligence
Infoblox
rburton@infoblox.com

Laura Rocha
Cyber Intelligence
Infoblox
ldarocha@infoblox.com

***Abstract*—Blacklists are a fundamental component of many cyber security products. Typically populated from a number of sources including machine learning algorithms and malware sandboxes, large automated blacklists carry the risk of blocking access to necessary, benign Internet resources and negatively impacting users. Whitelists reduce this risk by identifying items that should not be blocked. Unfortunately, whitelists are often poorly constructed and ill-maintained, failing to provide the proper counter-balance and potentially weakening the security ecosystem. This paper introduces the first published algorithm for automated whitelist creation. Based on Bayesian statistical learning methods and incorporating network and threat information, it is both defensible and responsive to the environment. For comparison, we describe two additional algorithms for whitelisting and evaluate all three over a six-week period using multiple data sources. Further, we demonstrate how the proposed algorithm can be used as a quality assurance mechanism for blacklists, a concept not previously described in the literature.**

***Index Terms*—network security, security management, statistical learning, machine learning algorithms**

I. Introduction

Modern cyber security products often attempt to proactively protect their user base from malicious cyber actors by preventing them from taking certain actions, such as visiting a web page, downloading a file, or installing software. They may intervene either dynamically, or via configured **blacklists**, comprised of known threat **indicators** such as domain names or IP addresses.[1] Most Internet users have experienced the effect of blacklists at some point, either via protections they've purposefully installed, such as anti-virus products, or from Internet firewalls and other large network security systems. Many security products now incorporate machine learning algorithms, in addition to utilizing curated feeds of known malicious behavior, to increase the breadth of their ability to identify and prevent compromise of their user's environment. It is common to use a hybrid approach of creating blacklists through automated discovery of new threat indicators along the inclusion of items that subject matter experts discovered through more manual processes.

Regardless of how they are created, blacklists pose the risk of accidentally preventing user access to necessary and benign Internet resources. Users of anti-virus and parental control products know the frustration of being denied access to a website incorrectly categorized by the product's algorithms. In addition to errors produced by machine learning algorithms, blacklists harvested from malware itself can also create false alarms. Whitelists help mitigate the inherent risks associated with blacklists. A **whitelist** is a domain-specific list of indicators that the security product should not blacklist.[2] In theory, a whitelist contains known benign indicators that are important to users and serves to counter mistakes that might exist in the blacklist. In practice, whitelists are often manually crafted and ill-maintained, resulting in stale information which may ultimately prove to increase, rather than decrease, the risk of compromise to users [28] [27] [30]. A common practice observed in the literature is to use a self-determined heuristic to subset lists of popular domains [20] [3] to create a whitelist.

While whitelists play a critical role, they are rarely discussed and lack the attention from the community that malicious behavior receives. This imbalance creates an Achilles heel in security products. Overzealous machine learning algorithms can lead to the need for an increased whitelist size, and the lack of regular review of whitelist indicators creates unwieldy and indefensible lists. The ideal whitelist is balanced between ensuring that users have the best experience possible while still providing for their security. As such, a whitelist should be as small as possible, while covering the majority of legitimate user activity. Most importantly, a whitelist should be up-to-date and contain only indicators believed to be benign with a high degree of confidence.

Machine learning algorithms offer the opportunity to automatically determine whitelists, however there are some restrictions that make this challenging. Because of their role in controlling security products, whitelists are highly scrutinized, both for what they contain and what they do not. The ideal algorithm is easily interpreted, defensible, and able to adjust to changes in the environment. This latter condition implies that the model does not suffer from *model drift*[3], a common problem in machine learning, in which the model's performance deteriorates over time due to changes in the data environment. Many commonly used machine learning approaches require significant volumes of labeled data and require complete

[1]While the term blacklist is widely used, a blacklist might be manifested in several ways, not necessarily a literal list, and may contain patterns of activity rather than strict indicators. The term *blocklist* is used interchangeably.

[2]More generally the whitelist can contain patterns of activity.

[3]Model drift is also referred to as concept drift.

retraining on a regular basis.

Our solution addresses these concerns through the use of Bayesian inference models, a method of statistical learning. The algorithm combines user behavior, the majority of which is benign, with known threats to create a classifier that learns over time. The result is a whitelist that continually adjusts to changes in the user and threat environment, is robust to error, and is defensible. As threat levels change within the network, so will the size and content of the whitelist.

In the sections that follow, we consider the use case of a Domain Name System (DNS) Firewall in which the whitelist and blacklist consist of domain names. We'll introduce three different approaches to automated whitelist creation for this use case. All three leverage popularity rankings of Internet domains, and two incorporate threat information. These can be implemented using publicly available data or collected network events. To illustrate the algorithms, and show the advantages of the Bayesian inference approach, we examine three sources of data and the resulting whitelists over a long period of time. To set the stage, we begin with a discussion of whitelisting in computer security products and then introduce the DNS vocabulary necessary for the subsequent examples.

II. Background and Related Work

There is very little in the literature regarding the creation and maintenance of whitelists. The Internet Storm Center (ISC) [14], among others, provide a whitelist for use by the public, but with little insight into how it was generated or is maintained. Many online articles and blogs provide advice to companies on manually curating whitelists for email and user applications [28]. The primary focus of these types of whitelists is to counter issues with email spam filtering, which may cause legitimate emails to be diverted to a user's spam folder. While automated spam identification serves a valuable role in user productivity, early algorithms were particularly likely to falsely identify email as spam, causing users to miss it; hence, spam whitelists were invented.

In the limited context of spam filtering, the effectiveness of whitelists and the burden of maintaining them has been studied. The results of that 2002 study [9] found spam whitelists both effective and, after initial configuration, relatively easy to maintain. In [16] the author converted a statistical method for identifying spam via content analysis into a method for whitelisting email. With respect to spam within Twitter, [5] used graph theoretic approaches to identify a whitelist of Twitter users.

Most other published references for whitelisting are not truly related to the topic of this paper; we mention them here for completeness. For example, many services exist that allow products or users to interactively query proprietary internal blacklists and whitelists, including SURBL [26] and Google Safe Browsing [11]. In most cases, these systems are designed for blacklist queries, in which software such as a browser or anti-virus product queries a remote database to determine whether the domain or IP address is considered malicious in some fashion. These systems are often implemented via DNS in accordance to RFC 5782 [19]. While entitled "DNS Whitelisting", RFC 5782 addresses the use of the DNS protocol to check whitelists, rather than providing a whitelist for DNS itself. Beyond the title, this RFC has no overlap with the subject of this paper.

As in the case of the RFC 5782 [19], language surrounding the concept of whitelisting can be confusing. Many references use whitelists in the context of allow-only filters, meaning that only items in the whitelist are allowed by default. This approach [29] limits network access for users without explicit consent and was more typical in web browsers in the early 2000s. In this paper, we do not consider this type of whitelist.

A second major area of whitelist literature surrounds application whitelisting, for which the National Institute of Standards and Technology (NIST) has provided guidelines [25]. Application whitelisting is slightly different than the use of whitelisting in this paper, although the algorithm presented here may apply to that use case in certain scenarios. In application whitelisting features of software, either behavior-based or signatures, e.g. SHA-1 hashes, are used to determine whether the software should be allowed to run in a user's environment. Application whitelisting algorithms attempt to identify a set of benign activities within the execution of a piece of software.

It is fairly common in cyber security literature to assume that popular domains are benign. In creating a system for DNS reputation in [3], for example, the authors assume the top 10,000 domains reported in the Alexa Top1M domain list [2] to be "known good domains." According to [18], over one hundred top-tier studies in cyber security between 2014 and 2018 based their experiments and conclusions on Alexa and other public ranking sources. The authors found significant reliability issues with each of the sources they reviewed and formulated their own merged ranking, TRANCO [18]. Similar issues are raised in [24], and they further found that thirty-eight papers published at major security conference venues in 2017 leveraged these lists for their results. Our own data analysis revealed that the Alexa rankings did not conform to established natural properties in rank-frequency data, specifically statistical consequences of Zipf's Law, adding further suspicion to the list creation. In most cases, researchers use the lists as a benign set of domains. In some cases, as in [17], the authors required that domains remain popular over some period of time to be considered benign. The use of top ranked domains for a whitelist is not limited to security research; as one example, the Quad9 DNS service reportedly uses the Majestic Million [22] as the basis for its whitelist [23]. Others [8] have reported using Alexa and Cisco Umbrella [6] for their whitelists. In all cases, the criteria for determining the subset of popular domains for whitelists is arbitrary.

The algorithm presented in this paper is well suited to use cases where there exists a set of potentially benign indicators, the ability to rank the impact of blocking those indicators, and a set of known malicious indicators that can be mapped into the same space. It is particularly applicable to the case of DNS Firewalls, which we introduce in Section III and carry through the paper as an example.

III. DNS Firewalls

The domain name system is a global, hierarchical, distributed database which serves, among other things, to map domain names to Internet Protocol (IP) addresses. While relatively straightforward in concept, in practice the global DNS is complex to the point of being arcane [12]. For the purpose of this paper, we introduce a limited scope and vocabulary; the interested reader can find more depth in the operation and security of DNS in [15] and [21].

The domain name system operates as a query-response protocol, in which a query for a **fully qualified domain name (FQDN)** is made by a client, or *endpoint*, and is answered via an iterative process known as *resolution*. An FQDN is made up of a series of text labels separated by periods. For example, the FQDN `www.google.com` has three labels ['`www`', '`google`', '`com`']. While it is possible for an endpoint to resolve DNS queries themselves, in practice, most devices rely on large **recursive resolvers** to perform resolution on their behalf. For example, Internet Service Providers provide recursive resolvers for their customers.

Much modern malware, and other threats like phishing attacks, require the resolution of domain names to IP addresses. As a result, a significant portion of the cyber threat landscape transits the domain name system in some fashion. DNS Firewalls were designed to provide security to users by enforcing rules based on domain names or IP addresses. In a typical scenario, the DNS Firewall is a component within a recursive resolver. It consults a list of domain names and IP addresses that are known to be malicious while resolving DNS queries: the blacklist. If the DNS Firewall receives a query for a domain on the blacklist, or that resolves to an IP address on the blacklist, it may return an incorrect IP address or respond with a "no such domain" response. The result is that the user is unable to access the malicious resource. In other cases, the blacklist may be the result of online machine learning algorithms. The full breadth of DNS Firewall capabilities is beyond the scope of this paper. In most settings, a DNS Firewall is installed in a relatively large network and its action impacts many users. Even more so than client-based cyber security products, it is particularly important that the DNS Firewall decisions be as accurate as possible. In addition to a blacklist, a DNS Firewall may use a whitelist to ensure that it does not interfere with user's access to legitimate Internet resources. For the purpose of this paper, we consider the whitelist to be a list of DNS domain names. If a domain is on the whitelist, the DNS Firewall does not block access to it; in this context we say that the whitelist **covers** this domain and the associated user activity.

IV. Whitelist Creation Algorithms

As manual creation and maintenance of whitelists is fraught with issues, we wish to automate these processes. To begin, we adopt the common assumption that very popular domains, such as `google.com`, are unlikely to present a threat and, moreover, that blocking access to these domains will adversely impact users. This is generically true given the nature of Internet communications: even the most widespread malicious threat will have limited reach in comparison to globally popular resources. Given this assumption, one approach is to order domains according to popularity and add domains that are more popular than a selected threshold to the whitelist. We will see two variations of this approach in Section IV-A.

While easy to implement, the popularity-based algorithms have a number of shortcomings. We address one issue in Section IV-B by incorporating threat data to determine the whitelist threshold. While creating a more informed whitelist, the simple inclusion of threat data will not adjust to changes in the environment and requires performance monitoring over time. We introduce our solution, a Bayesian inference model, in Section IV-C to address these lingering issues and provide a whitelist that is balanced between user experience and user security.

A. Popularity-Based Whitelists

Given a set of domains, we create a **rank ordering** on the set as a way of measuring the potential impact on users if access to the domain is blocked. Throughout this paper, rank has a value in $1..N$, where 1 is the highest rank, meaning the most important. A **popularity measure** is a metric that creates a rank ordering of the data according to some aspect of importance within the environment. For example, the number of DNS queries for a domain over some period of time in a network, or alternatively, a graph-based measure like PageRank.

We'll refer to a set of ranked data based on some popularity measure as **popularity data** and the association of the data with the measure that creates the rank ordering as the **popularity model**. Given one set of data, such as registered domains, one might create a number of associated popularity models using different ranking algorithms. We might consider, for example, aggregate query counts for domains within a network. These counts might be normalized in different ways to create different popularity models, for example, by the time-to-live of the response record, or the average number of queries per subdomain.

If event data is not available, popularity models for Internet domains can also be obtained from publicly available lists, such as Alexa [2], Majestic [22], Cisco Umbrella [6], and TRANCO [18]. These ranked lists are published daily and are commonly used in cyber security research as a source of benign domains. In most public sources, the underlying data that supports the rankings is not released, however the algorithms presented here can still be applied.

Given a popularity model, as an initial approach to whitelist creation, we set a fixed threshold rank M and whitelist all domains more popular, or with higher rank, than M. The result is a simple classifier on domains that identifies elements of the whitelist. By regularly updating the popularity data, this solution addresses concerns about stale or irrelevant items in the whitelist. As seen in Section II, both the cyber security research and product communities have used this approach. In the literature most thresholds appear arbitrarily chosen.

To create a data-driven threshold, in situations where we have underlying frequency information, we can create a fixed threshold using patterns of user behavior over time. Assume we have a popularity model of our DNS data in which we can associate each ranking to the probability of seeing that domain in traffic. In that case, each domain that is added to the whitelist accounts for a certain percentage of expected user traffic and we can choose a threshold M that reflects the amount of coverage we desire in the network. This approach allows the size of the whitelist to change over time, while maintaining a fixed percentage of coverage.

Statistically speaking, we calculate the *cumulative distribution function* (CDF) for the domains in popularity rank order, that is, we compute the cumulative rank-frequency. A whitelist can be determined by a fixed percentage within the cumulative distribution. Specifically, domains that contribute to normal user activity above the fixed percentile are included. An example cumulative distribution function for DNS domain traffic, with a possible threshold of 75%, is shown in Figure 1. Because the cumulative distribution function of DNS queries follows a generalized power law distribution [1], the size of the whitelist grows dramatically for each percentage point of coverage gained above a certain point. In some cases, it may be more important to ensure almost no chance of interruption to the user and a high threshold, e.g, 95%, may be set. In other cases, it might be more important to minimize the size of the whitelist, in which case the 'elbow' of the cumulative distribution function, that is the point at which the slope quickly diminishes, may be a more reasonable threshold. Using a percentile threshold, rather than a fixed rank threshold, allows for the number of domains on the whitelist to change according to changes in the user environment. In Figure 1, the threshold of 75% occurs slightly after the 'elbow' of the distribution. Increasing the whitelist in this example from 75% to 76% coverage would require the addition of 1500 more domains.

Unfortunately, using popularity alone is risky for multiple reasons. Because cyber threats can cause spikes of activity that make malicious domains very popular for some time, it is challenging to determine exactly what constitutes "popular enough" to be part of a whitelist. In addition, collection biases in the data used to determine popularity may render it ineffective for protecting users in a specific, e.g, customer, network. This can easily lead to inadvertently including malicious domains in the whitelist. In the next section, we see how the introduction of threat data can improve our ability to determine an appropriate value for the threshold M.

B. Threat-Informed Whitelists

By including threat data into our algorithm, we can create a more informed threshold. In this scenario, a **threat** is a domain associated with a malicious actor and referred to as a *malicious domain*.[4] As with the popularity data, we consider a set of threats to be **threat data** and the ordered set, using the ranking imposed by the associated popularity model, to be the **threat model**. The rank of the most popular threat in the threat model is called the **threat rank**, also referred to as the **threat level**. Only the highest ranking, rather than the aggregate set of threats is used for a few reasons. First, we know our insight to threat is limited. Cyber threats are constantly emerging, and their presence in global Internet traffic is conditioned on a number of factors that cannot be well measured. Second, the rank ordering is itself an estimate and including all known threats into the model introduces more uncertainty. Finally, the use of a single data point allows for a simpler, more intuitive model. There are a number of publicly available, commercial and non-commercial, threat feeds that can be used for this purpose, such as SURBL [26], Bambenek [4], and DGA Archive [7].

[4]The malicious domains might be thought of as members of the associated blacklist although in a general setting they may not be the same.

Figure 1 demonstrates how the popularity rankings of domains within a DNS environment compare to threats within a blacklist, with respect to the cumulative probability by rank. The threat over two time periods, while mostly below the 75% threshold, varies widely. This volatility make the choice of a static threshold which attempts to both cover use activity and minimize threat difficult. Using a fixed threshold created in this manner is more effective than popularity alone, but ultimately carries the same risks because of the inability to optimize the threshold.

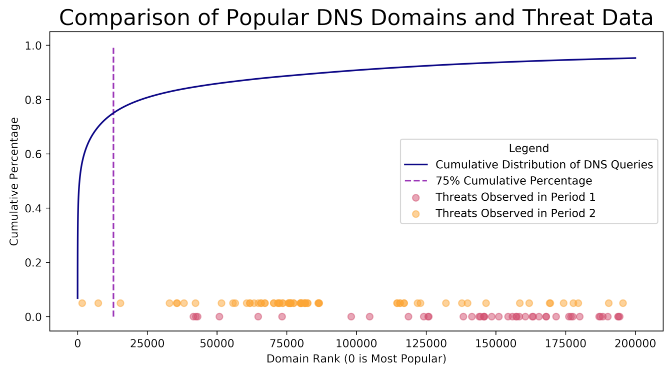

Fig. 1. A comparison of the rank of threat indicators during two time periods to overall popularity rankings for a sample DNS and blacklist data source. The density is measured over the top one million domains; only the top 200k are shown in the figure.

C. Bayesian Inference Model

To address the shortcomings of fixed threshold whitelist algorithms, we use Bayes Theorem (Equation 1) to determine a threshold every time the whitelist is updated. This theorem quantifies a relationship between conditional probabilities, that is, probabilities that are dependent on a set of observations. In the case of whitelist creation, we want to infer the most likely current threat rank from a set of observations over time. The resulting whitelist is comprised of those domains with rank higher than this threat rank and is entirely replaced with every iteration of the algorithm. This determines a one-class classifier for whitelist domains within the space of ranked domains. The resulting classifier differs from the classifier described in Section IV-A in that the threshold is not fixed, either as a percentile or size, and changes over time; it continuously adapts to a changing user and threat environment.

To formalize this method, assume we have a set of domains, D, and an associated ranking on the set valid at time t. Moreover, assume a subset of these domains, T, are known to be malicious. The *observed* threat rank within T, at time t, according to the ranking imposed by D is denoted E. Note that we do not know all threats, and therefore the true highest ranking threat is unknown to us. We wish to infer the *most likely* true threat level, or threat rank, based on the observation, E, and prior information. Each possible threat level is a hypothesis, H, and the set of hypotheses consists of the integer ranks, $1..N$, where $N >> 0$, and $N = 1$ is considered the highest rank.

Let $P(H|E)$ be the conditional probability that H is the true threat rank given that E is the observed threat rank. According to Bayes Theorem,

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}, \tag{1}$$

where $P(E|H)$ is the probability of observing E given that H is the true threat rank, while $P(E)$ and $P(H)$ are the probability of E and H across all possible hypotheses and observations, respectively.

In our application, there is a time series of observations, and Bayes Theorem is used to create a series of equations. The probability of each hypothesis, H, will change over time based on these observations. Let

$$P_t(H) = P(H|E_1, ..., E_t), \tag{2}$$

where $P(H|E_1, ..., E_t)$ is the probability of H conditioned on the sequence of observations $E_1 ... E_t$ and $P_0(H)$ be the initial probability prior to any observations. Considering all values of H, Equation 1 provides an algorithm for updating $P_t(H)$ over time. Since the $P(E)$ is independent of the hypotheses, it does not impact the maximum likelihood and is dropped, leaving

$$P_t(H) \approx P(E_t|H) * P_{t-1}(H). \tag{3}$$

where $t \geq 1$ and E_t is the observation at time t. When considered over all possible observations, $P_t(H)$ is a probability distribution. The threshold for our whitelist is the rank H with the maximum likelihood $P_t(H|E)$.

D. Threat Review

A side effect of this algorithm is the ability to detect problems in a blacklist, creating a quality assurance loop for the security product. To our knowledge, this is the first published algorithm for correcting blacklists. After computing the whitelist threshold, purported threats with a higher rank than the established threshold may exist. While the domains may be actual threats, they may also be false positives in the threat data. For example, malware that leverages domain generation algorithms (DGA) for their command and control may coincidentally generate existing legitimate domains. In other cases, threat feeds may contain legitimate domains that malware uses for ancillary purposes, such as establishing the

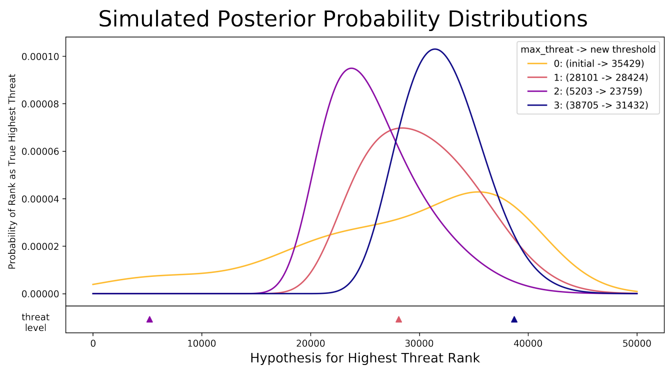

Fig. 2. A simulation of the Bayesian inference Model. The likelihood distribution and threshold is seen to adjust over time.

current time on a compromised host. These domains can be flagged for manual review to ensure that they are indeed active threats. The threat review capability allows us to find faulty entries in the blacklist that may otherwise be hard to detect.

V. Results

To illustrate the algorithm, we implemented both threat-informed static threshold and Bayesian inference model algorithms for whitelisting using multiple sources of popularity data and evaluated them over a six-week period. For the popularity model, we compared the publicly available sources Alexa [2] and Majestic Million [22], as well a large source of DNS query-response logs obtained from Farsight Security [10], included in their Channel 202 feed.

These sources were chosen based on their adoption in the literature and to demonstrate the algorithms across three different methods for creating a rank ordering. Alexa rankings, according to the company [2], are based on a combined measure of reach and page views computed over a trailing 3 month period. In contrast, Majestic rankings are determined by the number of referring IP subnets for a given domain name website [22]. Both of the publicly available lists are restricted to website traffic, while the DNS source contains a much broader range of domains, e.g, those related to content delivery services and ad networks. The Farsight records are collected from a globally distributed set of recursive resolvers and contain query-response events between the recursive resolvers and authoritative name servers. The volume exceeds 200,000 observations per second across the sensor array [10]. The rank ordering of Farsight events was computed by totaling the number of queries per second level domain over a week and sorting by total count. These sources produce very different ranked lists, as exemplified in Table I.

The associated threat model was derived from the Infoblox Threat Intelligence Data Exchange (TIDE) [13]. The TIDE data set contains threat indicators for malware command and control domains, malware domain generation algorithms (DGA), malware download sites, and others, in seventeen aggregate feeds. The TIDE data includes both internally curated threat feeds, as well as some available from widely-known

TABLE I
A COMPARISON OF THE TOP FIVE RANKED DOMAINS FOR ALEXA, MAJESTIC, AND DNS QUERY DATA FOR OCTOBER 5, 2019.

Rank	Farsight	Alexa	Majestic
1	akamaiedge.net	google.com	google.com
2	akamaidns.net	youtube.com	facebook.com
3	akamai.net	tmall.com	youtube.com
4	trafficmanager.net	baidu.com	twitter.com
5	fbcdn.net	sohu.com	linkedin.com

third party partners, such as SURBL [26]. All domains rated by the vendor as a "high" threat were included and were not further evaluated.

The popularity for any given indicator may vary widely across sources. We don't expect, for example, either Alexa or Majestic to contain traffic from malware such as Necurs or Pykspa, which create command-and-control domains using a domain generation algorithm (DGA) and are not observed as web pages. Table V shows how the popularity of sampled malicious domains varies across different data sources. This kind of variation is consistent with the findings of other authors, including [18] and [24]. For the application of whitelisting, these differences emphasize the importance of treating each popularity source independently.

For these demonstrations, whitelists were created by using the three algorithms introduced in Section IV. The first is the simple static threshold, which is based on the assumption that popular domains are benign and ignores threat data. In the first model we simulate whielisting a large majority of the expected user traffic. We call this the *max-coverage threshold*. In the second implementation, we reviewed several months of historic threat data and chose a static threshold that minimized the threat. We refer to this static threshold as the *min-threat threshold*. Lastly, we implemented the Bayesian inference model over the same data.

Recall from Section IV-A that domains are ranked from $1 \dots N$, where the most popular, or highest, rank is 1. We say that a domain is above the threshold if it has a higher rank than the threshold. The whitelist size is defined by the number of domains above the threshold that are not identified as malicious. For each of the popularity sources (i.e., Alexa, Majestic Million, and Farsight), we set thresholds as follows:

- min-threat static threshold using either the top $12,000$ domains for Alexa and Majestic, or the top 75% with respect to density for Farsight.
- max-coverage static threshold using the top $200,000$ domains for Alexa and Majestic or the top 95% with respect to density for Farsight.
- Bayesian threshold using the observed threat rank.

This produces a total of nine whitelists weekly for comparison. For each of these, we calculated the following metrics:

- the threshold of the whitelist,
- the threat rank, i.e. the rank of the most popular threat observed, and
- the number of domains in the threat data that were above the threshold, i.e, the number of domains needing review, also referred to as the number of threat "hits".

The key takeaways, detailed in the following sections, are for the goal of balancing coverage and threat,

- the popularity of threat varies so widely that a fixed static threshold can not be picked with confidence,
- even thresholds set to minimize threats can contain a significant number of indicators requiring review,
- the Bayesian model significantly increases coverage, without a tremendous increase in threat,
- the Bayesian model adapts to changes in the environment, and the threshold moves, but does not change dramatically with outliers in the threat data

A. Threat Rank Variance

One might hope to use classical statistics techniques to identify a suitable static threshold based on a history of threat data. To this end, we attempted to find a confidence interval for the most popular threats. The variance for each source, however, was too high. This problem can be illustrated using the coefficient of variation, a normalized measure of variance that allows for comparison across data sets. Table IV shows that the coefficient of variation was approximately 140%, 46%, and 32% for Farsight, Alexa, and Majestic, respectively. This means that the popularity of threats is very unstable and implies that a fixed threshold that accurately represents the threat can not be reliably determined.

The threat rank varied markedly not only within a data source over time, but also across data sources. During the evaluation period, we can see a threat rank as high as 64 in the DNS query logs, and as low as $127,330$ in Alexa. Although Alexa and Majestic Million are popularity rankings of domain names, they are curated in very different ways from each other and the DNS query logs. Referring to Table V, we are reminded of how disparate the threat level for a fixed domain can be across sources. The variance across Alexa, Majestic, and Farsight demonstrates the importance of generating whitelists from sources relevant to the use case environment, but also with thresholds relevant to the specific source.

B. Threat Comparison by Model

Reportedly malicious threat indicators exist above every whitelist threshold as seen in Table III. Recall that we've assumed that our knowledge of the full set of threat indicators is imperfect. Thus the quantity of threats above each whitelist threshold should not be interpreted literally. Instead it tells us two things: a rough estimate of the threat remaining in the whitelist, and the human resources required to review the known potential threats. The former indicates the risk to users of the whitelist and the latter is potential gain in removing faulty, but popular, indicators from the blacklist.

Most evident from the results in Table III and Figure 3 is the risk involved with the max-coverage static threshold. In all cases, whitelists of around $200,000$ domains carry significant risk of threat inclusion. With median values ranging from 163 to over 1800, regardless of data source, incur both significant threat and costly human review. Even the min-threat threshold

TABLE II
A COMPARISON OF THE POPULARITY OF SEVERAL REPORTEDLY MALICIOUS DOMAINS BY RANKING SOURCE IN JUNE 2019.

Domain	Threat Type	Farsight	Alexa	Majestic
fox.to	ConfickerC	N/A	127,607	N/A
cpagrip.com	UncategorizedThreat	137,015	23,615	184,069
newsfacce.com	MalwareGeneric	14,066	234,565	N/A
altmea.com	Phishing	194,693	331,134	N/A
setforconfigplease.com	MalwareDownload	579,461	N/A	28737

includes some threat, and the range overlaps significantly in all cases with the Bayesian model. This implies that we can increase coverage for users with limited increase of threat risk. For the Farsight data, we can quantify this increased coverage. Overall, using the whitelist sizes in Table IV, we see that the Bayesian model increases coverage by about 10%.

C. Model Threshold Behavior

Finally, we considered how the whitelist generated by the Bayesian algorithm adapted to changes in the threat landscape over time. In Table IV we observe a trend of a threshold decrease whenever a threat rank is observed to be more popular than in the previous week. This means that if a threat is observed to be popular, our model will slightly decrease the size of the whitelist — once our model is learning from the data and predicting higher threat ranks. The same analogy can be made for the increasing thresholds. In this way, we can see that the Bayesian model adjusts to the threat, and the threshold moves accordingly, but it will not dramatically "swing" based on a single observed threat. For example, within the Farsight data, we observe a threat rank of 64 in week 3. Zipf's Law would suggest that this extremely popular threat might be a false positive, and we see that the Bayesian threshold moves only slightly from $38,761$ the prior week to $38,555$. If this occurs in isolation, it's impact on the whitelist is minimal and temporary. On the other hand, sustained trends in the threat data will cause the threshold to continually move closer to that new range.

Table IV shows the lists threshold have low variation over time. In Section V-B we observed that Alexa has a lower median threat ranks, hence slightly increasing the whitelist threshold in Figure 4. For Majestic we observe the opposite behavior: with greater median threat rank our model has the tendency to decrease the size of the list. Figure 4 shows the posterior probabilities, by source, after each round of iteration. With every new threat rank a new threshold distribution is generated.

An informed prior is used to obtain an optimal threshold more quickly. In Figure 4 the Farsight informed prior (dark blue line) is closer to the predicted values in the following weeks, and varies slightly across time. For Alexa and Majestic, shown in Figure 4, the prior is more distant from the subsequently weeks threshold, taking a longer time to stabilize to an optimal threshold. Hence the threshold variation is higher for these data sources, as observed in IV.

VI. IMPLEMENTATION CONSIDERATIONS

There are a number of implementation considerations for each of the whitelisting algorithms presented in this paper. Most important are the application and data sources. As we saw in Section V, the threat level associated with a given indicator may vary widely depending on the popularity source. Applications in the DNS Firewall space, for example, that leverage only Alexa or Majestic, for example, as a popularity model will miss a large portion of the threat in their environment. Additionally, the rankings will incorrectly represent the popularity of the domains in the application domain. The high numbers in Table III comment both on the risks inherent in whitelists and the potential faults in blacklists. In practice, blacklists used in whitelist creation should be carefully curated.

Implementing the Bayesian algorithm does require tuning that depends on the popularity and threat data sources, as well as the context of the security product. In particular, the algorithm requires a good model for $P(E|H)$, the likelihood of observing E as the maximum threat level, given that hypothesis H is true, as well as an initial prior, $P_0(H)$, to initialize the algorithm.

In terms of whitelists based on popularity ranking, $P(E|H)$ represents the likelihood that E is observed as the highest threat rank given that H is the true highest threat rank. In general, this probability distribution is unknown and must be estimated based on the use case. While tempting to assume, for example,

$$P(E|H) = 1, E = H,$$
$$P(E|H) = 0, E \neq H,$$

this approach does not allow for noise in collection and creates an inflexible model. The use of discrete step functions will quickly eliminate hypotheses from further consideration, and it is recommended to instead use a continuous estimate which assumes some error in observation. A likelihood distribution that is too restrictive, even if continuous, will result in collapsing the space of hypotheses. While the appropriate estimate for $P(E|H)$ is highly dependent on the data source and use case scenario, starting with a Gaussian distribution centered on H and refining it based on the data sources is effective.

The selection of an *initial prior*, $P_0(H)$, probability distribution is also important. While in theory one can derive a good Bayesian model given sufficient data using a uniform prior, that is, the assumption that all hypotheses are equally likely, an *informed prior* is more efficient. We used a historical record of

TABLE III
QUANTITY OF REPORTED THREATS ABOVE WHITELIST THRESHOLDS, BY MODEL TYPE.

Source	min-threat		max-coverage		Bayesian	
	range	median	range	median	range	median
Farsight	[4, 78]	71	[57, 773]	711	[11, 151]	138
Alexa	[5, 49]	40	[11, 179]	163	[6, 66]	56
Majestic	[7, 95]	90	[29, 2073]	1886	[11, 392]	295

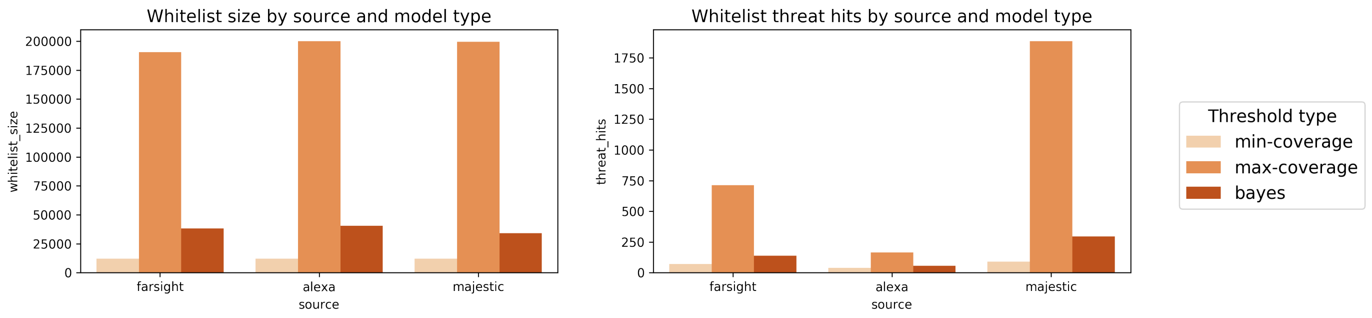

Fig. 3. Median whitelist size and number of threats above the threshold, by model type.

TABLE IV
IMPACT OF THREAT RANK ON THE BAYESIAN WHITELIST THRESHOLD.

Period	Farsight		Alexa		Majestic	
	Threat Rank	Whitelist Threshold	Threat Rank	Whitelist Threshold	Threat Rank	Whitelist Threshold
Week1	39078	36783	75406	35460	39242	38525
Week2	41299	38761	127330	35460	40000	39038
Week3	64	38555	54279	40518	22068	35406
Week4	83	38346	39069	40311	21898	32982
Week5	122	38126	45160	40816	21951	31278
Week6	334	37823	50802	41772	18464	29801
Median	228	38228	52540.50	40414	22009.50	34192.50
Variation	139.92%	1.7%	45.81%	5.21%	32.36%	10.07%

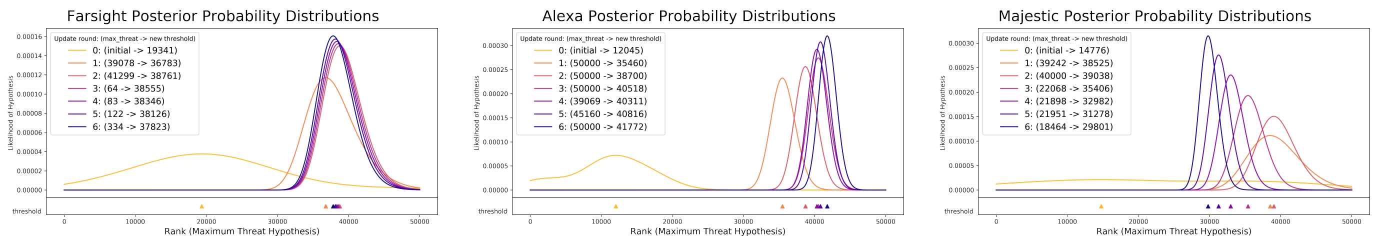

Fig. 4. Prior and posterior probability distributions observed across 6 weeks, and whitelist rank threshold defined by the inferred maximum likelihood.

maximum threat levels to establish the initial prior, converting the histogram to a continuous distribution via a kernel density estimate and normalizing for the range of hypotheses.

One might want to combine multiple sources of popularity data to create whitelists, for example, to take advantage of different perspectives provided by publicly available lists and privately obtained event records. While sharing a dependent variable, independently created rankings are fundamentally different data sets and there is not a mathematically sound means to merge them. Instead, each data source should be processed separately, and a consolidated whitelist created through the union or intersection of the results. As can be seen in Table I, different data sources and ranking measures can produce quite different results even in the most popular domains. Further the popularity of threats can vary dramatically across sources, as seen in Table V.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Sean Tierney for motivating this work. We also appreciate the constructive reviews from Darin Johnson, Emma Lanier, Mike Last, Andrea Urban, and anonymous reviewers that greatly improved the quality of the paper.

References

[1] Lada A. Adamic and Bernardo A. Huberman. "Zipf's law and the Internet". In: *Glottometrics* 3 (2002). URL: https://www.hpl.hp.com/research/idl/papers/ranking/adamicglottometrics.pdf.

[2] *Alexa Top Sites*. the daily top1M is available for download at http://s3.amazonaws.com/alexa-static/top-1m.csv.zip. URL: https://aws.amazon.com/alexa-top-sites/.

[3] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. "Building a Dynamic Reputation System for DNS". In: *Proceedings of the 19th USENIX Conference on Security*. USENIX Security"10. Washington, DC: USENIX Association, 2010, pp. 18–18. URL: http://dl.acm.org/citation.cfm?id=1929820.1929844.

[4] *Bambanek OSINT Master Feed Listing*. last accessed May 22, 2019. URL: https://osint.bambenekconsulting.com/feeds/.

[5] David R. Bild, Yue Liu, Robert P. Dick, Z. Morley Mao, and Dan S. Wallach. "Aggregate Characterization of User Behavior in Twitter and Analysis of the Retweet Graph". In: *ACM Trans. Internet Technol.* 15.1 (Mar. 2015), 4:1–4:24. ISSN: 1533-5399. DOI: 10.1145/2700060. URL: http://doi.acm.org/10.1145/2700060.

[6] *Cisco Umbrella Top 1 Million*. Oct. 2016. URL: https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/.

[7] *DGA Archive*. last accessed May 22, 2019. URL: https://dgarchive.caad.fkie.fraunhofer.de/.

[8] *Domain Whitelist Benchmark: Alexas vs Umbrella*. Apr. 2017. URL: https://www.netresec.com/?page=Blog&month=2017-04&post=Domain-Whitelist-Benchmark%5C%3A-Alexa-vs-Umbrella.

[9] David Erickson, Martı̀n Casado, and Nick Mckeown. "The Effectiveness of Whitelisting: a User-Study". In: *Fifth Conference on Email and Anti-Spam*. 2008.

[10] *Farsight Security*. URL: https://farsightsecurity.com.

[11] *Google Safe Browsing*. URL: https://safebrowsing.google.com/.

[12] Bert Hulbert. *Herding the DNS Camel*. Nov. 2018. URL: https://www.ietf.org/blog/herding-dns-camel/.

[13] *Infoblox Threat Intelligence Data Exchange for Active Trust Suite*. SN-0218-03 0318. URL: https://www.infoblox.com/wp-content/uploads/infoblox-solution-note-infoblox-threat-intelligence-data-exchange-for-activetrust.pdf.

[14] *Internet Storm Center Suspicious Domains List*. Accessed May 21, 2019. URL: https://isc.sans.edu/suspicious_domains.html.

[15] Anestis Karasaridis. *DNS Security: In-depth Vulnerability Analysis and Mitigation Solutions*. 2012. ISBN: 978-0387765457.

[16] Eric Kidd. "Bayesian Whitelisting: Finding the Good Mail Among the Spam". In: (Sept. 2002). last accessed May 21, 2019. URL: http://www.randomhacks.net/2002/09/29/bayesian-whitelisting/.

[17] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero Gömez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. "Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse". In: *CoRR* abs/1708.08519 (2017). arXiv: 1708.08519. URL: http://arxiv.org/abs/1708.08519.

[18] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation". In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium*. NDSS 2019. Feb. 2019. DOI: 10.14722/ndss.2019.23386. URL: https://tranco-list.eu.

[19] J. Levine. "DNS Blacklists and Whitelists (RFC-5782)". In: (Feb. 2010). URL: https://tools.ietf.org/html/rfc5782.

[20] Vector Guo Li, Matthew Dunn, Paul Pearce, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. "Reading the Tea leaves: A Comparative Analysis of Threat Intelligence". In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 851–867. ISBN: 978-1-939133-06-9. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/li.

[21] Cricket Liu and Paul Albitz. *DNS and BIND*. O'Reilly, 2006. ISBN: 978-0596100575.

[22] *Majestic Million Now Free for All, Daily*. download at http://downloads.majestic.com/. Oct. 2012. URL: https://blog.majestic.com/development/majestic-million-csv-daily/.

[23] *New "Quad9" DNS service blocks malicious domains for everyone*. last accessed May 22, 2019. Nov. 2017. URL: https://arstechnica.com/information-technology/2017/11/new-quad9-dns-service-blocks-malicious-domains-for-everyone/.

[24] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. "A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists". In: *CoRR* abs/1805.11506 (2018). arXiv: 1805.11506. URL: http://arxiv.org/abs/1805.11506.

[25] Adam Sedgewick, Murugiah Souppaya, and Karen Scarfone. "Guide to Application Whitelisting". In: (2015). URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-167.pdf.

[26] *SURBL: URI Reputation Data*. last accessed May 22, 2019. URL: www.surbl.org.

[27] *The UGLY Truth Behind the Practice of IP Whitelisting*. last accessed October 6, 2019. Nov. 2014. URL: https://https://community.akamai.com/customers/s/article/The-UGLY-Truth-Behind-the-Practice-of-IP-Whitelisting.

[28] *What is Whitelisting and How Should you Implement it?* last accessed May 21, 2019. June 2018. URL: https://www.springboard.com/blog/what-is-whitelisting/.

[29] *Whitelisting Technology: The Solution to Malware Nightmares*. May 2016. URL: https://techtalk.pcpitstop.com/2016/05/26/whitelisting-technology-the-solution-to-malware-nightmares/.

[30] *Whitelists for Cyber Security: The Bad and the Ugly*. Oct. 2018. URL: https://www.digitalimmunity.com/whitelists-for-cyber-security/.