On Precisely Detecting Censorship Circumvention in Real-World Networks

Ryan Wails*†, George Arnold Sullivan‡, Micah Sherr*, and Rob Jansen†
*Georgetown University
†U.S. Naval Research Laboratory
‡University of California, San Diego

***Abstract*—The understanding of realistic censorship threats enables the development of more resilient censorship circumvention systems, which are vitally important for advancing human rights and fundamental freedoms. We argue that current state-of-the-art methods for detecting circumventing flows in Tor are unrealistic: they are overwhelmed with false positives ($> 94\%$), even when considering conservatively high base rates (10^{-3}). In this paper, we present a new methodology for detecting censorship circumvention in which a deep-learning flow-based classifier is combined with a host-based detection strategy that incorporates information from multiple flows over time. Using over 60,000,000 real-world network flows to over 600,000 destinations, we demonstrate how our detection methods become more precise as they temporally accumulate information, allowing us to detect circumvention servers with perfect recall and no false positives. Our evaluation considers a range of circumventing flow base rates spanning six orders of magnitude and real-world protocol distributions. Our findings suggest that future circumvention system designs need to more carefully consider host-based detection strategies, and we offer suggestions for designs that are more resistant to these attacks.**

I. Introduction

With information access controls in place in over 60 countries [54], Internet censorship continues to be used to restrict human rights and fundamental freedoms of individuals worldwide. Recent protests in China against strict COVID-19 lockdown policies, in Iran against the death of a woman who was arrested for not wearing a hijab, and in Russia against its war with Ukraine have all been met with increased censorship in order to restrict subjects' access to digital services and disrupt online communication [2, 21, 74]. At the same time, the rising trend of digital authoritarianism has compelled over 60 nation-states to commit to a *Declaration for the Future of the Internet* that promotes "a global Internet that advances the free flow of information" and clearly communicates deep support for the study of censorship and strategies for circumventing it [76]. We seek to advance the principles of a free and open Internet and better understand how to design more effective and resilient censorship-resistant communication systems by studying how a sophisticated censor might detect and block the use of such systems in real-world networks.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23394
www.ndss-symposium.org

We consider a censor whose primary objective is to block unwanted network flows created by deployed censorship circumvention software. To facilitate blocking, the censor first uses a classification technique to label network flows as either *benign* or *circumventing*. A major problem for the censor in deploying a classifier is that the prevalence of benign flows in their network is expected to be *much* greater than the prevalence of circumventing flows [44, 75]. The extremely low base rate of circumventing flows implies that the censor may be overwhelmed with classification errors (false positives) even if the classifier has what may appear to be acceptable performance (for example, 99.9% accuracy). Thus, it is commonly assumed that the censor is constrained to classifiers with very high precision (which implies very low false positive rates) so that they can effectively limit the collateral damage caused by erroneously blocking benign flows [44, 58, 75].

Inspired by the base rate challenge, we examine the research question: *how can a censor use machine learning to detect censorship circumvention in real-world networks with high precision?* Previous work inadequately addresses this question due to a limited evaluation of classifier performance with respect to base rates and network scale. We demonstrate throughout §IV that a recent, state-of-the-art method for classifying Tor circumventing flows [79] fails when scaling to realistic base rates: precision decreases from 97% to 3% when the base rate of circumventing flows decreases from an unrealistic 50% to a still conservative 0.1%. Moreover, a common goal of circumvention protocols is to hide among the "long tail" of traffic [84, 86, 88], but the state-of-the-art methods only coarsely analyze false blocking rates in the tail.

In this paper, we present a new approach for detecting censorship circumvention in real-world networks. Our key insight is that circumvention protocols typically include a client that communicates with a limited set of proxies or other long-lived servers (for example, Tor bridges); as the protocols are repeatedly used over time, circumventing flows will tend to concentrate around these long-lived proxy server destinations. From this insight, we examine two primary novel advances over previous work on detecting censorship circumvention protocols. First, we transform the circumventing *flow* detection problem into many circumventing *host* detection problems, each of which is much smaller in scale and thus considerably more manageable. Second, our detection methods incorporate information from *many flows over time* for each destination host; our classifiers become more precise and we gain confidence in them as we accumulate information over time.

Our host-based detection methods accumulate temporal information from flow-based classifiers. We consider the use of deep learning methods to classify each host's flows, including convolutional neural networks (CNNs) and a stacked denoising auto-encoder (SDAE), because these were found to work particularly well at fingerprinting encrypted website traffic in Tor [1, 14, 57, 66, 68]. We find that the deep-learning classifiers improve flow-based classification by orders of magnitude relative to the state-of-the-art techniques [79]. Furthermore, we find that deep-learning methods (which do not require manual feature engineering) generalize better than classical techniques (which are tuned to a specific circumvention protocol), but do not scale well enough to be applied in isolation without host-based augmentation.

Our circumventing host detection methods are designed to predict whether or not a host is involved in a circumvention protocol after a configurable confidence threshold is reached. Our primary method for computing confidence is based on standard statistical analyses, and the *classification with rejection* strategy [15] is used in order to reduce false classifications. By augmenting the deep-learning classifiers with our host-based detection methods, we find that a censor can significantly improve classifier performance—after observing at most 38 flows for each host, false positive classifications are eliminated, but circumventing hosts are still caught with perfect recall.

Our host-based detection methods do require that we store a small amount of additional state, which we argue is minimal and can be easily satisfied using modern hardware. Our analysis in §VI shows that our methods require at most 14 bits of storage per destination (address, port) pair, which means that tracking over 4×10^9 destination services requires no more than 50 GiB of storage.

We argue that better understanding how to defend against a more sophisticated censor that uses machine learning to detect circumventing flows will help circumvention system designers develop more resilient circumvention systems. Indeed, one of the primary takeaways of our work is that circumvention systems that use long-term hosts as ingress points should not behave consistently or else they will be easily detected by host-based methods. Circumvention system designers should consider employing adversarial examples, using ephemeral obfuscation servers, and developing new programmable or polymorphic obfuscation protocols to improve resilience to host-based censorship methods.

We summarize our main contributions as follows:

- We are the first to evaluate deep-learning and host-based censorship circumvention detection methods in real-world networks; our study uses over 60,000,000 flows to over 600,000 destinations observed from a university campus network.
- We present a new methodology for evaluating classifier scalability that for the first time considers classifier performance across a range of circumventing flow base rates spanning six orders of magnitude.
- We present the first evaluation of the extent to which Tor-based circumventing flows can be discovered with deep learning algorithms tuned for encrypted traffic analysis.
- We present a novel application of host-based analysis techniques to the problem of precisely detecting censorship circumvention in real-world networks.
- We present insights for future work, including several pitfalls that future researchers should carefully consider as they evaluate their systems, and strategies for defending against host-based censorship.

II. Background

Internet censorship is the restriction of access to public information or services on the Internet. Nation-states and other organizations use Internet censorship to control the flow of sensitive, offensive, or otherwise damaging information into and out of the networks they operate [54, 64]. Generally, a censor conducts Internet censorship using an apparatus consisting of two primary functionalities [44]: (1) *fingerprinting* classifies network traffic flows into allowed and prohibited classes (using traffic analysis techniques); and (2) *direct censorship* actively blocks the prohibited flows (for instance, by instrumenting firewall rules or forging TCP `RST` packets). Our work focuses on fingerprinting.

Researchers have responded to censorship by developing censorship resistant systems (CRSes) that employ techniques to evade fingerprinting. The goal(s) in these systems is to make it difficult to detect the prohibited communication and/or to increase the collateral damage (false positives) that would be incurred when performing blocking [44]. The approaches used by previously proposed CRSes, which we survey in §VIII, include protocol mimicry, protocol tunneling, and protocol randomization. Though our methods apply broadly, our work focuses on the most popular CRS framework, Tor [23].

Tor is an anonymous communication network that can be used to more safely browse the Internet and resist censorship. Tor routes Internet communication through a series of three volunteer proxy servers (the *entry*, *middle*, and *exit* relays) such that the intended destination cannot be observed at any point between the client and the exit. Because the destination is hidden, censors are unable to effectively control access to Internet content fetched through Tor. Therefore, censors have focused on blocking access to the Tor network by blocking access to its proxy servers.

Real-world censors have used IP filtering, active probing [85], and deep packet inspection (DPI) to fingerprint and block Tor [75]. Since Tor (like most CRSes) relies on proxy servers outside of the censor's control, censors may identify the IP (or IP and port) of the proxy servers and then block any flows on its network with matching destination addresses. Typically, the proxy server IP addresses can be discovered by running instances of the CRS software and inspecting the destination of the connections it creates. Additionally, the censor may actively probe suspicious hosts by sending specially crafted packets to them in an attempt to elicit a non-standard or identifying response [24, 27, 32, 85]. Finally, censors use DPI to match flows based on particular features of the network traffic. For example, a particular byte sequence in the first few packets in a flow has been previously used to identify the Tor protocol's unique TLS configuration [85].

Resisting Censorship with Tor: To combat censorship, Tor has deployed: (1) *bridge* relays whose address information is

not publicly available and to which clients connect in place of entry relays, and (2) a framework called *pluggable transports* that enables the development of unique methods to obfuscate the traffic between the client and bridge (that is, across the censor's observation point). This paper focuses on the two most popular pluggable transports that are currently built into Tor browser and available for users: obfs4 [88] and Snowflake [70].

The obfsproxy family of protocols (which includes obfs4) was designed to evade application-layer fingerprints by producing a data stream that is indistinguishable from random bytes and uses an authenticated Diffie-Hellman key exchange to protect against man-in-the-middle attacks [72, 73, 88]. Because obfs4 produces a data stream that is immediately randomized (unlike most encrypted protocols, such as TLS, which have structured handshakes), previous work has found that the entropy of the initial messages can be used to distinguish it from normal traffic [79]. However, we show that such methods yield poor precision outside of laboratory conditions (see §IV).

Snowflake is a unique pluggable transport that enables anyone with a modern web browser to serve as a proxy between the client and the Tor bridge [70]. Snowflake is sometimes called a *flash-based proxy* technique since its proxies (which are volunteers' web browsers) are not fixed and proxies may be ephemeral. Snowflake *clients* first find available Snowflake *proxies* by contacting a *broker* over a domain fronted [30] connection, and then use public STUN servers to enable connections with proxies through NAT devices. Snowflake clients and proxies then tunnel traffic through direct WebRTC connections.

Adversary Model and Scope: In this paper, we examine the detectability of the widely used obfs4 and Snowflake pluggable transports. We run instances of the pluggable transports in a large university network while collecting and analyzing network traffic from the perspective of a network censor. In addition to applying state-of-the-art detection techniques [79], we also consider methods that are reasonably available to the censor but not yet considered by previous work: (1) the use of *deep learning* to mitigate the reliance on manual, hand-engineered features, and (2) *host-based analysis* to improve classifier performance by combining information observed from a series of flows over time.

We assume that the censor is willing to be *stateful*, allowing it to store information about a set of flows and a set of hosts over time. As we show in §VI, censors need only to store a minimal amount of such state in order to perform precise host-based classification.

We consider an *on-path* censor, by which a passive tap allows examination but not removal of packets [75]. IP addresses and ports are ignored throughout our analyses as a feature of detection.

We use traditional metrics for understanding classifier performance (e.g., true/false positives/negatives, precision, recall, etc.). Importantly, the censor's goal of minimizing collateral damage corresponds to minimizing false positives.

III. Network Traffic Dataset

To assess the ability of the censor to detect obfuscated protocol traffic from among ordinary network traffic, a source of realistic, benign network traffic is required. Unfortunately, network traffic datasets collected from real users are not publicly available due to the privacy-sensitive nature of Internet communications. Hence, we collected a private dataset of network flow statistics using a university-hosted[1] network observation point which receives a copy of each network packet traversing the university's WiFi network.

This section provides statistics describing the collected data and details for how the data was collected. Note that in §IX we outline the safety measures taken when handling this dataset and further discuss the ethics of this collection.

A. Data Collection

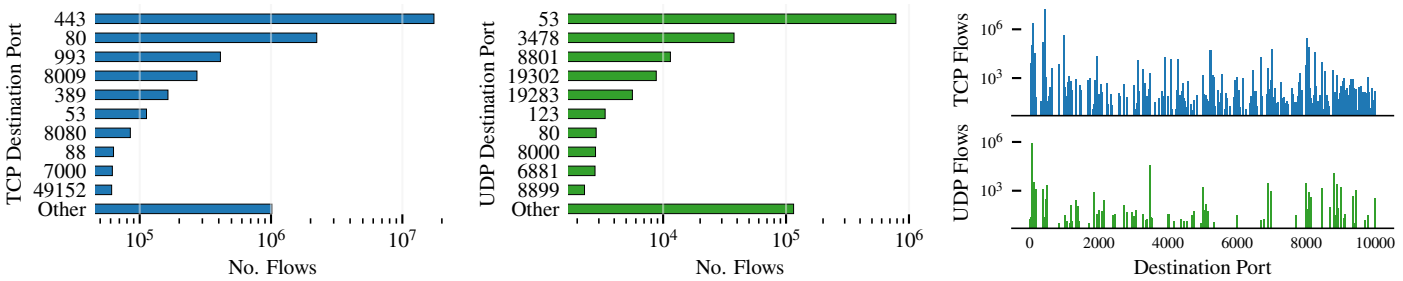
The capture machine we used to collect network data was provisioned with two 10 Gbps network interface cards (NICs) receiving copies of packets traversing the campus WiFi network. The NICs were configured to sort the packets into thirty-six separate receiving-side scaling (RSS) queues according to each packet's five-tuple. An instance of packet capture software running `PF_RING` with zero-copy support was attached to each RSS queue and transferred the packets, in `pcap` format, to instances of a custom packet processing program we wrote in roughly 3,500 lines of C++ code. Only IPv4 TCP and UDP packets were captured during our measurements.

The custom packet processing program sorts the incoming stream of packets into flows according to each packet's five tuple. Up to 5,000 packets are collected for each flow. Once one of three conditions is met, the flow is considered to be ready to be written to disk: (1) the flow contains the limit of 5,000 packets, (2) the flow was transmitted via TCP and contained a valid `SYN/SYN+ACK/ACK` packet handshake and `FIN` packet shutdown, or (3) 60 seconds passed since the first packet of the flow was transmitted. We do not directly write packet data to disk. Instead, when a flow is ready to be written, we compute a number of statistics describing the flow that are recorded. First, we record general information about the flow, such as the time that the first packet was encountered or the flow's five tuple. Note that IPv4 addresses are replaced with hash values derived from a HMAC function using a key that was discarded at the end of the capture. Second, we record summary statistics describing the flow's entropy, timing, and packet size characteristics (these features are further enumerated in §IV-C). And third, we record each packet's payload size.

In addition to collecting *background* network traffic generated by hosts on the university WiFi network, we also produced *obfuscated* protocol flows that were collected by our infrastructure. We operated 8 crawler machines connected to the WiFi network that accessed popular webpages from the Alexa list [10]. For each request, a crawler initiated a new instance of Tor Browser (via Selenium) and randomly selected a site from the Alexa list according to a power law distribution biased in favor of the most popular websites. The crawler then fetched the chosen site via the Tor Browser.

Each crawler was configured to use either obfs4, obfs⋆ (our tweak to obfs4 which we describe in §IV), or Snowflake. The Snowflake crawler used the default configurations as provided in the Tor Browser. We operated our own obfs4 and obfs⋆ bridges

[1]The name of the university is intentionally omitted for privacy.



(a) Number of TCP flows per port when grouped by destination port.

(b) Number of UDP flows per port when grouped by destination port.

(c) No. of flows per port grouped by destination port for the first 10k destination ports.

Figure 1: Distribution of the captured background flows' destination ports. Results are presented in log scale.

to allow fairer comparisons between the classification accuracy of obfs4 and obfs⋆, because obfs⋆ is not a deployed obfuscation protocol. We operated eight obfs4 and seven obfs⋆ bridges, which we hosted on Google Cloud regions in North America, South America, Europe, and Asia. Importantly, unlike prior work [79], the crawlers operated inside of the WiFi network, and their traffic was collected using the same infrastructure as the background traffic produced by the other hosts in the university. Operating obfuscation clients within the network rather than overlaying their traffic on top of a separate background traffic source reduces the risk of introducing subtle artifacts (for example, differences in network effects such as latency or packet sizes) that could lead to artificial distinguishing characteristics in the traffic.

B. Collection Statistics

The data collection ran between March and April 2022 for two weeks. In total, we collected 54,355,226 background flows sent to approximately 600,000 unique (IPv4 address, destination port) pairs. In addition, we collected 83,002 obfs4 flows; 207,975 obfs⋆ flows; and 5,894,149 Snowflake flows.

Fig. 1a and Fig. 1b show the frequency of the ten most popular TCP and UDP flow ports, respectively, when the background flows are grouped by destination port; the remaining ports are bucketed into the bar labeled "other". Unsurprisingly, we find that HTTPS flows (TCP port 443) constitute the vast majority of TCP flows and DNS flows (UDP port 53) constitute the vast majority of UDP flows. HTTP (TCP port 80) and STUN (UDP port 3478) flows occur often, but with roughly $10\times$ less frequency than HTTPS or DNS flows.

The "other" bar in Fig. 1a and Fig. 1b show the frequency of flows not sent to one of the top ten most popular TCP or UDP ports. We observe a significant number of flows on ports outside of the top ten, indicating that the distributions are skewed and contain a long tail. This long tail of destination ports is depicted in Fig. 1c, which shows the number of flows collected for the first 10,000 destination ports that are not among the top ten most popular. (For visual clarity, the y-axis starts at 10^1, but many destination ports had fewer than ten flows.) Many of these tail ports have moderate popularity, suggesting that our capture contains a diverse mix of traffic.

Fig. 2 shows the number of TCP flows captured per hour over the 2 week measurement period. There is a clear diurnal pattern demonstrated in the volume of captured data over time. The pattern is consistent with typical, daily computer network use [63].

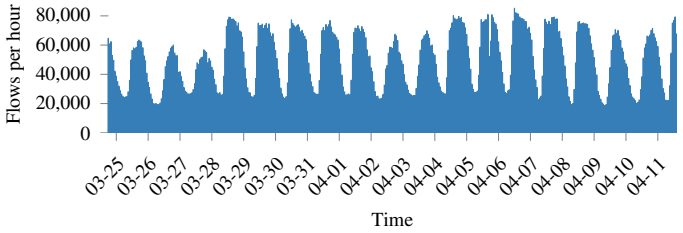


Figure 2: Number of TCP flows captured per hour over the measurement period.

IV. Limitations of the State of the Art

Using the data described in §III, we review state-of-the-art obfuscated-protocol detection methods and assess the degree of realism they achieve.

A. Preliminaries

As suggested by prior work [44, 79], we view the problem of obfuscated protocol detection as a supervised binary classification machine learning task. In the supervised setting, it is assumed that the censor has access to a set of data containing network flows with known ground-truth labels and uses this data to train a machine learning model. In practice, the censor can obtain labeled flows by privately running protocol hosts or perhaps by monitoring a network with known ground-truth behavior (for instance, a private network composed of administratively-controlled machines). In the binary classification setting, flow labels take on one of two values: benign/negative (N) or circumventing/positive (P).

Given this trained model, a censor can predict labels for new flows, outside of the training set, for which ground-truth labels are unknown. This scenario is simulated by evaluating the classifier over a test set of flows, disjoint from the training set, with known ground-truth labels. For each predicted label over the test flows, there are four possible outcomes:

True negative: If the predicted label was negative and the flow was generated by a benign protocol.

False negative: If the predicted label was negative and the flow was generated by a circumvention protocol.

True positive: If the predicted label was positive and the flow was generated by a circumvention protocol.

False positive: If the predicted label was positive and the flow was generated by a benign protocol.

We follow the standard convention of using TN, FN, TP, and FP to refer to the total number of each outcome, respectively.

A classifier's performance may be quantified according to a number of values defined over these outcomes—in this work, we focus on four commonly-used metrics: true positive rate (recall), false positive rate, precision, and F_1-score. These metrics are defined as follows:

True positive rate (Recall or TPR): Defined as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$. Recall indicates what fraction of positive flows were detected and blocked by the censor.

False positive rate (FPR): Defined as $\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$. The false positive rate indicates what fraction of benign flows will be falsely flagged and blocked by the censor.

Precision (Prec): Defined as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$. If it is assumed that the censor takes a blocking action on each flow labeled positive, then precision indicates what fraction of blocked flows were actually generated by circumvention protocols.

F_1-score (F_1): The harmonic mean of precision and recall, $(2\cdot\mathrm{TPR}\cdot\mathrm{Prec})/(\mathrm{TPR}+\mathrm{Prec})$. Powerful classifiers achieve maximal F_1-score, which indicates that they can both identify the positive flows and not falsely classify too many negative flow as positive.

Importantly, the true and false positive rates are both values that are unaffected by the ratio of positive and negative flows—the *base rate*—in the dataset: TPR is defined with respect to only the positive flows, and FPR is defined analogously to only the negative flows. In contrast, precision and F_1-score are both affected by the base rate: FP, the total number of false positive classifications and a term in the denominator of precision, is of course dependent on the total number of negative flows present in the dataset (i.e., high precision is trivial to achieve in sets with few negative examples relative to the positive examples).

When examining obfuscated-protocol detection methods, it is necessary to consider a range of realistic base rates because of the relatively low but variable prevalence of circumvention protocol traffic in real-world networks. (The failure to evaluate performance over a range of base rates is a significant weakness of prior work, as we discuss in more detail below.) We let λ be the total number of negative examples to positive examples in the data set. For example, $\lambda = 100$ when there are 100 benign flows per circumventing flow in the dataset. To disambiguate precision and F_1-score calculated at different base rates, we use the notation $\mathrm{Prec}^{\lambda=x}$ and $\mathrm{F}_1^{\lambda=x}$ to make explicit the base rate at which these values are defined. Note that Prec^{λ} (and consequently ${\mathrm{F}_1}^{\lambda}$) can be computed analytically for any base rate λ given only TPR and FPR as $\mathrm{Prec}^{\lambda}(\mathrm{TPR},\mathrm{FPR}) = \mathrm{TPR}/(\mathrm{TPR}+\mathrm{FPR}\cdot\lambda)$. Note that we eschew the use of accuracy as a performance measurement, as it is not meaningful in settings with large class imbalances [35].

Each of these classifier metrics takes on values between 0 and 1. From the censor's perspective, a useful classifier should have the following properties: (1) *high precision* (otherwise, the censor will make the majority of its blocking actions against benign flows); (2) *high recall* (otherwise, the censor will inadvertently permit a large fraction of circumventing flows it would instead prefer to block); and (3) *low false positive rate* (otherwise, the censor will block large fractions of benign flows). Consequently, a useful classifier also achieves high F_1-score values.

B. Basic Empirical Methodology

For all experiments, we created a training dataset containing $\approx 40\%$ of the captured flows and the remaining $\approx 60\%$ were used as the testing dataset. For TCP flows and UDP flows, this partitioning results in a test set consisting of 13,423,887 and 544,963 flows, respectively (TCP- and UDP-based protocols are always evaluated separately in our experiments). The partitioning was performed on a *per-destination-host* basis, meaning that no destination host that appears in the training set also appears in the test set. The test set also contains obfuscated flows—56,239 for obfs4, 207,975 for obfs$\star$, and 11,698 for Snowflake

Classifiers are trained by randomly sampling 1,500 negative flows and 1,500 positive flows from the dataset, which is consistent with prior work [79]. The benign flow random sample is constructed using stratified random sampling according to flow destination port number to ensure that the sample contains a diverse mix of traffic (otherwise, the sample would contain TLS or DNS traffic almost exclusively). In cases when a validation set is required to determine candidate classifiers to evaluate on the test set, we sample 50,000 negative and 50 positive flows from the training set (but disjoint from the 1,500+1,500 training flows).

Classifier performance is calculated over the test set of flows. Every value reported is the average of 10 repeated trials to account for sampling error and randomness in the classifier training process, such as random parameter initialization.

C. Validating Prior Results

Wang et al. [79] detected obfs4 using decision tree classifiers. To be provided as input to the decision tree, each flow is preprocessed into a number of manually-defined features which fall into three categories: (1) entropy-based features, (2) packet-header-based features, and (3) timing-based features. The entropy-based features are summary statistics (minimum, median, mean, and maximum) of per-packet Shannon entropy values computed from the measured frequency distribution of each packet's payload bytes. The packet-header features are statistics related to packet sizes, such as the most frequently-occurring packet size in the flow. Lastly, the timing-based features are normalized histogram bins measuring the time taken between TCP `ACK` packets. These features are computed from the first 30 packets in each flow. Using sample data and code from their implementation, we verified that our custom packet processing software computes these features identically.

Wang et al. evaluate four classifiers: a decision tree using (1) only the entropy features, (2) only the header features, (3) only the timing features, and (4) both entropy and header features. In the evaluation, 1,500 benign web (HTTP/TLS)

Table I: Decision tree classifier performance detecting obfs4. The yellow highlighted columns show poor performance at more realistic values of the base rate λ.

Features	TPR	FPR	$\text{Prec}^{\lambda=1}$	$\text{F}_1^{\lambda=1}$	$\text{Prec}^{\lambda=1\text{k}}$	$\text{F}_1^{\lambda=1\text{k}}$
Entropy	.98	.06	.95	.96	.02	.03
Entropy (Wang et al.)*	.98	.003	.99	.99	.25	.39
Header	.97	.05	.97	.96	.02	.04
Timing	.82	.21	.80	.81	.00	.01
Entropy & Header	.98	.03	.97	.97	.03	.06

*For easier comparison, these values are estimated and reprinted from Wang et al.'s work [79, Figure 7].

Table II: Decision tree false positive rates when detecting obfs4. Blocking rates are higher for benign flows with destination ports not in the training set (open world) and for benign flows with destination ports with low frequency ranks r.

Features	FPR				
	All	Open World	$\text{Tail}_{r>10}$	$\text{Tail}_{r>100}$	$\text{Tail}_{r>1000}$
Entropy	.06	.11	.08	.15	.19
Header	.05	.08	.04	.08	.10
Timing	.21	.24	.19	.30	.36
Entropy & Header	.03	.09	.05	.10	.13

flows and 1,500 obfs4 flows are used in the training set; 3,500 benign web flows and 3,500 obfs4 flows are used in the testing set. They found that the classifiers trained using entropy-based features performed the best, and found that obfs4 could be distinguished from the benign traffic with high TPR ($\geq 98\%$) and low FPR ($\leq 0.3\%$).

We recreated this evaluation on our data using Scikit-learn's default decision tree classifier implementation [61]. In general, we obtained similar results: shown in Table I, the classifier using entropy-based features achieved a 98% TPR and 6% FPR—the higher FPR may be accounted for by the larger diversity of benign traffic in our test set compared to the synthetically collected HTTP/TLS traffic distribution used by in their evaluation.

D. Evaluation Pitfalls of Prior Work

Despite the *prima facie* acceptable performance of these classifiers (using the $\lambda = 1$ base rate, $\text{Prec}^{\lambda=1}$ and $\text{F}_1^{\lambda=1}$ are nearly maximal), we argue that these results *do not* accurately reflect a censor's ability to detect obfs4 under realistic conditions. In what follows, we describe how evaluation pitfalls present in prior work skew performance too favorably toward the censor, and suggest methodological improvements that are better suited for detection in real-world networks.

1) False Positives: We re-examined the results of our validation experiment described in §IV-C. A key insight of our expanded analysis is that *non-uniform subsets of the benign flows experience non-uniform false positive blocking rates.* An important limitation of prior work is that reporting the false positive rate taken only over large classes of traffic paints a misleading picture of the classifier's performance, especially if that classifier is to be used in real-world networks.

Table II demonstrates this finding in two important subsets: the *open world* and the *tail*. During real-world testing, the classifier will encounter traffic from protocols that were not present during training, which is notable because out-of-distribution performance tends to suffer for many classifiers [51]. The "open world" column of Table II shows the false positive blocking rate for benign flows with destination ports not present in the classifier's training set. Our results show that false blocking rates in the open world are 14–200% higher than the overall false positive rate.

Misclassification is even worse in the *long tail of rarely occurring protocols*, where obfs4 is designed to hide [22]. The columns marked "$\text{Tail}_{r>x}$" show the false positive blocking rates for benign flows with destination ports not in the x most frequently occurring ports (for example, $\text{Tail}_{r>10}$ is the set of flows with destination port not in the top 10 most frequently occurring ports, which corresponds to the "other" bar in Fig. 1a). For instance, blocking rates for benign flows in $\text{Tail}_{r>1000}$ are 71–333% higher than the overall false positive rate. It is important to note that frequency rank does not necessarily correspond to importance, and the flows in $\text{Tail}_{r>1000}$ might originate from critical processes with high costs of blocking.

In summary, the false positive rate is considerably worse where it matters most: differentiating obfuscated (and random looking) flows from unknown (but non-obfuscated) protocols.

2) Scalability: It is commonly understood that the base rate of circumventing flows has a significant effect on classifier performance and that the low base rates present in real-world networks pose a challenge for the censor [44, 75]. Although prior work discusses the effects of low base rates, their primary evaluation does not consider that base rates can be highly variable both within the same network or across different networks [79]. Without considering a range of base rates, we are unable to fully understand classifier performance trends and limitations without additional analysis.

We argue that censorship analyses should consider a range of base rates of circumventing flows λ as a primary component of evaluation efforts because it yields more informative results that are more broadly applicable. For example, although we found acceptable performance in terms of the precision and F_1 metrics when using a high $\lambda = 1$ base rate, we found poor classifier performance when considering even just a modest base rate of $\lambda = 1{,}000$. As shown in Table I, $\text{Prec}^{\lambda=1000}$ and $\text{F}_1^{\lambda=1000}$ are approaching 0, suggesting that any blocking actions taken by the censor will be overwhelmingly false positives. Note that we chose $\lambda = 1{,}000$ for illustrative purposes, but in real-world networks, common sense suggests that the base rate of circumventing flows will be even lower than 1 per 1,000 benign flows (in the TCP distribution, for instance, the 9th-most frequently occurring port, 7000, has a rate of approximately 3 per 1,000 flows). Fig. 3 shows the relationship of precision to the base rate for a much larger range of values, $1 \leq \lambda \leq 1 \times 10^6$. Near-zero precision for values of $\lambda > 1 \times 10^3$ suggests that these classifiers will produce too many false positives (relative to true positives) in realistic networks and at scale.

Designing a hand-tuned classifier: This situation naturally raises a question: could the classifier be manually tuned to improve precision, given expert knowledge of the obfuscation

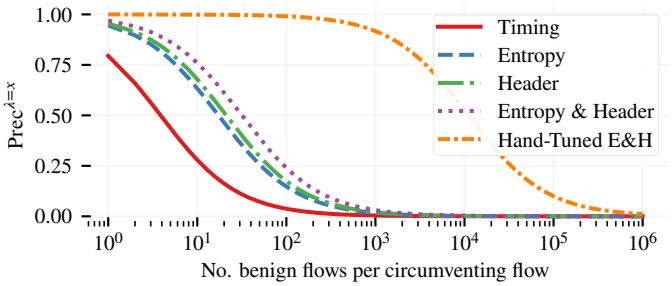


Figure 3: Decision tree obfs4 classification precision at various base rates of circumventing flows.

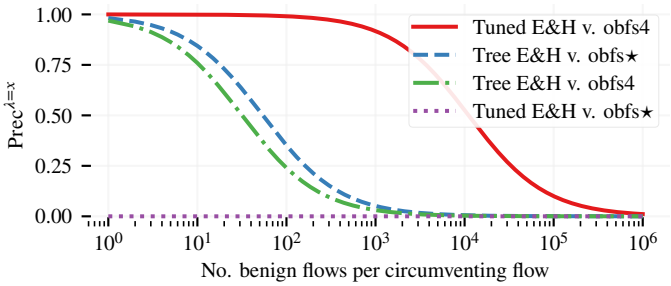


Figure 4: Precision of the entropy-and-header decision tree with and without manual tuning detecting obfs4 and obfs⋆.

protocol? We attempted to modify the entropy-and-header decision tree to use the features more informed by the characteristics of obfs4. Our hand-crafted classifier, for instance, computes the distribution of entropy values that a flow should exhibit if each packet payload is distributed uniformly at random, as is done by the obfs4 protocol, and compares this theoretical distribution with the one measured in the test flow. It performs other checks, too, such as inspecting the most frequently-seen packet size and verifying that it matches the size of one network MTU, which we found is characteristic of obfs4 traffic.

The *Hand-Tuned E&H* line of Fig. 3 shows the result of this manual tuning. The hand-tuned classifier scales to far lower rates of obfuscation, by approximately three orders of magnitude, at the expense of recall, which is reduced from 98% to 47%. (The base rate $\lambda = 37$ is the point at which F_1 suggests that the manually-tuned classifier has higher utility than the original one.) Classifier tuning may be necessary to scale to realistic base rates, but it increases the variance of the classifier and hence reduces generalizability.

3) Generalizability: Because the features used by the hand-crafted classifier very precisely define obfs4, as an evader, it becomes easier to tweak the obfuscation protocol to avoid detection. We play out one iteration of this hypothetical arms race, and design a tweaked obfs protocol which we call obfs⋆. In obfs⋆, the per-packet entropy values can be configured on a per-bridge basis. This is performed by re-encoding each obfs4 packet using an encoding scheme that biases toward a particular bit. For example, consider the toy encoding scheme where each `0` bit produced by obfs4 is replaced with bit triple `111`, and each `1` bit is replaced with `110`. In practice, obfs⋆ re-encodes obfs4 packets on a bytewise basis, and chooses each replacement string unambiguously and in a way that achieves a chosen bias in expectation. Additionally, in obfs⋆, the distribution of packet sizes can also be configured uniquely on a per-bridge basis.

Note that, recently, real-world nation-state censors have been observed to detect and block obfs4, Shadowsocks, and other randomization protocols, apparently by detecting the randomness in the flow [4, 8, 87]. As a response, reduced entropy encodings are actively being considered by The Tor Project and other researchers for future circumvention protocols [22, 87]. Additionally, the Great Firewall Report, an organization that focuses on quantifying technical censorship efforts in China, recently published a patch to Shadowsocks that modifies measured packet entropy values modifying the ratio of `0`s to `1`s in the binary representation of the packet [7]. Our independently-developed experimental protocol obfs⋆ is very similar to this approach of reducing the entropy in each packet by increasing each packet's size.

The corresponding classification results for obfs⋆ are also shown in Fig. 4. Because the hand-tuned classifier was so tightly fit to the signature of obfs4, the obfs⋆ modifications completely degrade classifier performance: it attains 0% precision and 0% recall. And, our results show that the obfs⋆ tweaks do not make the protocol any more detectable by the decision tree classifier with respect to obfs4.

In summary, the state-of-the-art ML-based detection technique fails to perform well under conditions that more accurately reflect the adversary's environment and incidence of obfuscation protocols. While hand-crafting a classifier to detect a targeted obfuscation protocol may yield higher precision than ML-based classifiers, such hand-tuned classifiers are prone to evasion even by trivial protocol modifications.

V. Deep Learning

Many of the most powerful attacks proposed for encrypted network analysis now use techniques adopted from deep learning [3, 14, 57, 66, 68, 69, 90]. Compared to their classical counterparts, deep learning classifiers tend to exhibit better classification performance and do not require complex feature engineering, but require more powerful computational resources to train the classifier. Deep learning techniques remain largely unexplored in the domain of obfuscated protocol detection, and in this section we show that they indeed constitute a more powerful class of attacks than previously considered.

A. Experimental Methodology

For our experiments, we considered three different neural networks considered in prior work for website fingerprinting attacks: a stacked denoising autoencoder (SDAE) and convolutional neural network (CNN) proposed by Rimmer et al. [66], and a CNN proposed by Sirinam et al. [68]. We used the implementations of these models which were openly available on GitHub[2], and made minor modifications, such as changing the last-layer activation function, that were necessary to apply these models in the binary classification setting. For the most

[2]The repositories are located at https://github.com/deep-fingerprinting/df and https://github.com/DistriNet/DLWF.

part, we used the model hyperparameters that were provided by the original authors. However, we do vary the input dimension of the model, discussed further below.

The original models take as input a sequence of packet directions $(d_1, \ldots, d_n)$ for $d_i \in \{-1, 0, 1\}$ associated with a flow—the value 1 is used for a packet sent from the client to the server, -1 for server to client, and the input is 0-padded to length n if the flow contains fewer than n packets. Packet directions are used instead of packet sizes are used because Tor uses fixed-size cells and hence size does not offer additional discriminative power. However, we find that providing sizes to the model does improve performance on the task of detecting circumventing flows. Therefore, we modified the models to take as input a sequence of packet sizes $(p_1, \ldots, p_n)$ where $p_i \in [-1, 1]$ is the real-valued normalized packet size of the ith packet in the flow. As before, the sign associated with each packet size indicates its direction in the flow. Additionally, we found that n, the number of packets per flow provided to the classifier, had a large influence on performance. We experimented with four different values of n in the set $\{100, 500, 1000, 5000\}$.

In contrast to some classical models, neural network classifiers output a real number which can be thought of as a probability or confidence score that the input is a circumventing flow. A neural network classifier's performance can accordingly be tuned by setting a per-classifier threshold for which examples are considered circumventing only if the confidence score is over this threshold. We determined each threshold value by computing the threshold that maximizes the classifier's $\mathrm{F}_1^{\lambda=1\mathrm{k}}$ score during validation.

In addition to training and testing of flows generated by obfs4 and obfs⋆, we also attempt to classify flows generated by the Snowflake censorship circumvention protocol. Snowflake generates a few different types of flows, and we considered two: (1) the domain-fronted request to the broker (made over TCP via TLS), and (2) the data flow communicated with the Snowflake proxy (made over UDP via WebRTC).

B. Validation

For each circumvention protocol, we therefore have 3 models $\times$ 4 input-dimensions $=$ 12 candidate classifiers. To determine the best candidate classifier to evaluate over the test set, we compared the classifiers' performance on the validation set of flows (which is disjoint from the testing data, see §III). Table III reports the average classification performance. For all circumvention protocols, the Sirinam et al.'s CNN model performed the best according to F_1-score; we use this model with the indicated input dimension (500 for obfs4, 5,000 for obfs⋆, and 500 for Snowflake) for classifier testing.

C. Experimental Results

Analogous to the evaluation and results we presented in §IV, we evaluated the CNN's performance on the testing set. In Table IV, the performance of the CNN classifier is summarized when detecting obfs4, obfs⋆, and Snowflake's two flow types (the broker flows and the data flows). For obfs4, the CNN improves classification false-positives by an order of magnitude, while still exhibiting a near-perfect TPR. (In Table II, the lowest decision tree FPR is 3×10^{-2}; the CNN achieves an overall FPR $= 2.9 \times 10^{-3}$.) The performance against obfs⋆ is even better: FPR $= 7 \times 10^{-4}$ with 100% recall. The highest performance is achieved detecting Snowflake's data connections ($\mathrm{F}_1^{\lambda=1\mathrm{k}} = 1.0$), which tend to exhibit much longer flows than typical UDP flows on the network, a property that can be learned by the classifier. Detecting the TLS connections to the broker performed the least-well among the four circumvention protocols (FPR $= 0.18$), which is expected: our network capture contains mostly TLS flows, and the Snowflake broker connections are genuine TLS connections. Furthermore, it is the case that the false positive rate in the tail and open-world is *lower* than the overall false positive rate, which is the opposite of the other classifiers. Again, this is due to the TLS flows, which are less present in the tail or open-world of protocols.

Table III: The best performing classifiers and input dimension used for each circumvention protocol. Performance values reported in this table are for the validation set.

Model	Input Dim.	TPR	FPR	$\mathrm{Prec}^{\lambda=1\mathrm{k}}$	$\mathrm{F}_1^{\lambda=1\mathrm{k}}$
		obfs4			
CNN [68]	100	0.96	5.1×10^{-4}	0.65	0.78
CNN [66]	100	0.92	7.4×10^{-4}	0.55	0.69
SDAE [66]	5,000	0.72	2.4×10^{-3}	0.23	0.35
		obfs⋆			
CNN [68]	5,000	0.99	1.7×10^{-4}	0.85	0.92
CNN [66]	5,000	0.82	1.2×10^{-3}	0.41	0.54
SDAE [66]	5,000	0.79	3.2×10^{-3}	0.20	0.32
		Snowflake (Data)			
CNN [68]	500	1.0	6×10^{-5}	0.94	0.97
CNN [66]	100	0.92	6.6×10^{-4}	0.58	0.71
SDAE [66]	5,000	0.89	5.0×10^{-3}	0.15	0.26
		Snowflake (Broker)			
CNN [68]	500	0.49	6.1×10^{-3}	0.07	0.13
CNN [66]	100	0.32	0.017	0.02	0.03
SDAE [66]	100	0.25	0.016	0.02	0.03

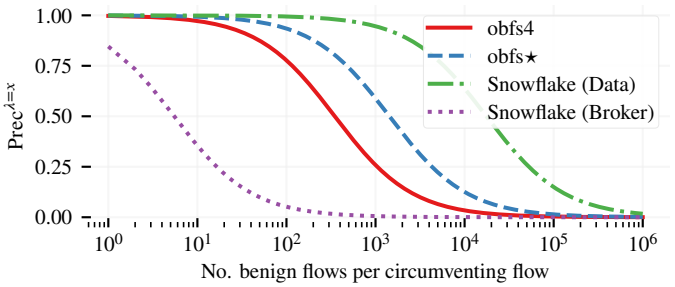


Figure 5: CNN classification precision detecting obfs4, obfs⋆, and Snowflake at various base rates of circumventing flows.

While it is the case that deep learning improves performance, the false positive rates are still prohibitively high to scale to realistic base rates. Fig. 5 plots Prec^{λ} as a function of the base rate λ. For more realistic base rates, such as $\lambda > 1 \times 10^6$, the precision attained by any of the classifiers is near-zero.

D. Computational Performance

While deep learning classifiers offer improved performance with respect to their classical counterparts, they also require

Table IV: Summary of CNN classifier performance against obfs4, obfs⋆, and Snowflake.

Protocol	TPR	FPR					$\text{Prec}^{\lambda=1}$	$\text{F}_1^{\lambda=1}$	$\text{Prec}^{\lambda=1\text{k}}$	$\text{F}_1^{\lambda=1\text{k}}$
		All	Open World	$\text{Tail}_{r>10}$	$\text{Tail}_{r>100}$	$\text{Tail}_{r>1000}$				
obfs4	1.0	2.9×10^{-3}	3.8×10^{-3}	2.4×10^{-3}	4.9×10^{-3}	5.8×10^{-3}	1.0	1.0	0.26	0.41
obfs⋆	1.0	7×10^{-4}	1.6×10^{-3}	1.0×10^{-3}	2.1×10^{-3}	2.5×10^{-3}	1.0	1.0	0.59	0.74
Snowflake (Data)	1.0	5.7×10^{-5}	5.4×10^{-4}	5×10^{-4}	4.1×10^{-4}	3.7×10^{-4}	1.0	1.0	0.95	0.97
Snowflake (Broker)	0.98	0.18	9×10^{-3}	7.1×10^{-3}	9.8×10^{-3}	3.6×10^{-3}	0.85	0.91	0.01	0.01

Table V: Deep learning classifier performance. The train time measures the amount of time required to perform one epoch of training over 100,000 input examples. The "test rate" column shows classifier throughput on a single GPU, and "parallel test rate" shows classifier throughtput using four GPUs.

	Input Dimension	Train Time (s)	Test Rate (per s)	Parallel Test Rate (per s)
CNN [66]	100	7	5,600	23,000
	500	8	5,100	20,000
	1,000	9	4,500	17,000
	5,000	20	2,300	8,800
SDAE [66]	100	16	8,600	34,000
	500	17	8,200	31,000
	1,000	17	7,700	31,000
	5,000	20	6,300	25,000
CNN [68]	100	24	4,100	15,000
	500	23	4,200	16,000
	1,000	25	4,000	15,000
	5,000	47	3,500	13,000

greater computational power. In this section, we briefly evaluate and discuss the performance costs of the deep learning classifiers we considered.

We measured the cost of using these classifiers on a bare-metal server configured with 2 Intel Xeon Platinum 8260 CPUs clocked at 2.40 GHz and four NVIDIA Tesla V100 GPU accelerator cards. Table V gives a summary of classifier performance when each classifier is given 100,000 input examples. We report (1) the time required to train for one epoch (2) the number of examples that can be classified per second on one GPU for a trained classifier, and (3) the number of examples that can be classifier per second using all four available GPUs.

In all cases, the training time is minimal—only a few seconds per epoch, and in practice, classifier training could occur relatively infrequently (but must be performed periodically in order to account for concept drift). Each classifier required fewer than 50 epochs to train, but the number of training epochs could be increased to potentially improve classifier performance.

More importantly, Table V reports *classification throughput*. The best performing CNN classifier (Sirinam et al.) using the largest input dimension (5,000 packets) can achieve a throughput of approximately 3,500 flows per second using a single GPU, and 13,000 flows per second parallelized across four GPUs. To put this rate into context, The Center for Applied Internet Data Analysis (CAIDA) monitors an internet backbone OC129 fiber link for a Tier 1 ISP [71]. On any given link, they never report more than 150,000 flows per second, which would require at most 50 GPUs to classify at line rate. Today, these cards cost approximately 5,000 USD [55], and hence the total cost to monitor such a link is no more than 250,000 USD (and likely even less if custom hardware is used). The most powerful nation state censors are estimated to spend *billions* of dollars to perform internet censorship [28], and so this cost appears to be feasible for moderately-sized to large censors.

E. Discussion

It is perhaps surprising that the CNN classifiers outperform the classical approaches using *only* packet sizes and directions. obfs4 and obfs⋆ do employ unique packet patterns on the wire—by design, they send bursts of roughly MTU-sized payloads followed by a randomly sized chaff packet [88]. Previous work has shown that packet burst sequences may carry a lot of information about the purpose of a flow [66], and deep neural networks can detect these complex patterns. The designers of obfuscated protocols tend to focus on shaping packet payloads. However, this detection method suggests that protocol designers need to also account for payload sizes and burst shapes as well.

It is likely that refinements could further improve performance of the deep learning classifiers. First, although we were not able to evaluate the risk in this work (we did not collect or store packet payload values), it seems likely that packet *payload values* can be provided to these classifiers to further improve performance, which is a strategy recently suggested by Holland et al. [38]. Next, we used a fairly small training set size (1,500 positive and 1,500 negative flows) to keep the results comparable to the classical classifiers of Wang et al. [79]; larger training sizes might yield improved results. Finally, new neural network architectures and deeper models may be good candidates for further improving performance. Recently, transformer networks have been successfully applied to encrypted traffic classification tasks [48, 62, 90], achieving better performance than prior work..

VI. Host-based Analysis

In § IV, we confirmed that classical machine learning methods do not offer enough precision at realistic rates of circumventing flows in the network to be useful to a censor at scale. Deep learning approaches are a natural candidate to improve classifier performance. However, while state-of-the-art CNNs do significantly improve classification performance, they still appear unlikely to succeed at scale and under a realistic base rate of circumvention.

Both of these approaches are myopically focused on classifying and blocking *flows*. However, censors need not employ this strategy: many censors instead focus on identifying *hosts* taking part in a circumvention protocol [75]. Focusing

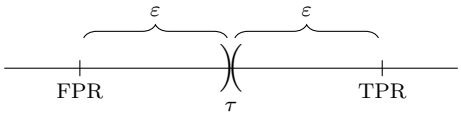
on classifying hosts presents new opportunities for a censor, as hosts may exhibit durable and measurable patterns of behavior over time, at the expense of maintaining some state per host.

In this section, we present a straightforward host-based classification scheme that is bootstrapped from a flow-based classifier. The host classifier works by simply tracking the fraction of positive classifications that are made with respect to each host. If a sufficient number of classifications is made and the fraction of positive classifications exceeds a threshold, then the host is classified as circumventing (and benign otherwise). In §VI-A, we provide a theoretical basis for this approach in ideal conditions. In §VI-B, we show that an implementation of this approach eliminates nearly all false positive classifications while still detecting all obfs4 and obfs$\star$ bridges.

A. Analytical Method

Over some period of time, let p be the total number of positive classifications made with respect to a benign server[3], and m be the total number of classifications made with respect to that server. Consider the fraction p/m for a benign host, which represents the percentage of positive classifications made for flows sent to that host. In our host classification scheme, we will use this fraction to classify hosts: for a threshold $0 \leq \tau \leq 1$, we will label the host as benign if $p/m \leq \tau$ and otherwise as circumventing.

If we assume each classification made with respect to the host is an iid Bernoulli random variable with success probability equal to the FPR, then $\mathbb{E}[p/m] = \mathrm{FPR}$. For some positive real-valued $\varepsilon < 1$, we would like to bound the probability that p/m is more than ε-distance away from the false positive rate. This is depicted as follows:



where $\tau = (\mathrm{TPR} + \mathrm{FPR})/2$ is chosen to be the point that maximally separates FPR ($\mathbb{E}[p/m]$ for a benign host) and TPR ($\mathbb{E}[p/m]$ for a circumventing host).

Hoeffding's inequality [37] provides such a bound: the probability that p/m is more than ε-far away from FPR is bounded loosely by $2e^{-2\varepsilon^2 m}$. Letting $\varepsilon = (\tau - \mathrm{FPR}) = (\mathrm{TPR} - \mathrm{FPR})/2$ and rewriting the expression in terms of m yields the inequality

$$m \geq \frac{\ln(4/\alpha^2)}{(\mathrm{TPR} - \mathrm{FPR})^2} \tag{1}$$

for a given error probability $0 < \alpha \leq 1$. Expressing this bound in terms of m suggests a means to employ this attack: fix an error rate α based on the network size under observation, determine a threshold for the number of observations $\eta = \lceil \frac{\ln(4/\alpha^2)}{(\mathrm{TPR}-\mathrm{FPR})^2} \rceil$, and only classify hosts for which $m \geq \eta$. This strategy—*classification with the reject option*—is a known technique to reduce false classifications [15, 81], and is the strategy we propose for a host-based attack.

In the following parts of this section, we will apply this analysis and construct a intuitive host-based classification technique. Here we conclude with a few remarks about this analysis:

- The observation threshold η scales *logarithmically* in α. The values of η obtained from the classifiers in §V are concretely small. For example, for $\alpha = 1 \times 10^{-6}$, $\eta = 32$ for the CNN detecting obfs4. We provide more precomputed values for η in Table VI.
- The bound produced by Hoeffding's inequality is not tight. A perfect classifier with a true $\mathrm{TPR} = 1$ and $\mathrm{FPR} = 0$ should be able to perfectly classify hosts after only a single observation. However, Equation 1 may require more than one flow even for a perfect classifier. In practice this isn't problematic, as the concrete values of η determined by this equation are small.
- The analysis makes no assumption about the base rate of benign and circumventing hosts in the network—the error probability α bounds the probability of making a false positive or false negative classification. (A union bound may be applied to determine a value of α that will yield an overall desired error rate.) For simplicity, we will use $\alpha = 1 \times 10^{-6}$.
- The assumption that classifications results are iid is unlikely to hold perfectly in practice. However, our empirical results do generally follow the behavior that is predicted by our analysis.
- The denominator of Equation 1 suggests that, given a choice of multiple classifiers, the one that maximizes $\mathrm{TPR} - \mathrm{FPR}$ (which is the same as maximizing Youden's J statistic [89]) produces the classifier requiring the least state.

Table VI: Values of η computed given a value of α for various TPR and FPR values obtained for the deep learning classifiers during validation.

α	η		
	$\mathrm{TPR} = 0.96$ $\mathrm{FPR} = 5.1 \times 10^{-4}$	$\mathrm{TPR} = 0.79$ $\mathrm{FPR} = 3.2 \times 10^{-3}$	$\mathrm{TPR} = 0.25$ $\mathrm{FPR} = 0.016$
1×10^{-3}	17	25	278
1×10^{-6}	32	47	530
1×10^{-9}	47	70	783
1×10^{-12}	62	92	1,035
1×10^{-15}	77	114	1,288

B. Implementation and Results

Algorithm 1 presents a procedure to carry out the host-based attack, which simply tracks the number of total and positive classifications that are associated with any given host (that is, address-port pair). Note that the attack is computationally inexpensive to mount—even for $\alpha = 1 \times 10^{-15}$, the better performing classifiers do not require more than 128 observations. Hence, 2×7 bits of state (i.e., $2 \times \log_2(\eta)$) is required per host. In the *worst* case, if state is maintained for *each* of the 2^{32} IPv4 addresses and 2^{16} TCP ports, this amounts to roughly 500 TiB of storage, which is well within a censor's capabilities (a single consumer-grade hard drive disk may store 20+ TiB today). However, dynamically allocated storage and sketch data structures, such as a counting bloom filter, could be used in practice to significantly reduce this storage cost and scale the

[3] A symmetric argument applies to circumventing servers.

Algorithm 1 Host meta-classification algorithm

In: Let x be a flow, and $x.\{\text{IP, Port}\}$ be the IP address and destination port of the flow.

Let F be a classifier, $F.\{\text{TPR, FPR}\}$ be the classifier's validation true/false-positive rate, and $F(x)$ be F's output label on flow x.

Let α be a desired error rate, $0 < \alpha \leq 1$

State: Let S be an associative array mapping (IP, Port) pairs to pairs of non-negative integers which are default initialized to $(0, 0)$.

Out: `Benign` or `Obfuscated` or `Reject`

function HOSTCLASSIFY(x, F, α)
  $\eta \leftarrow \lceil \ln(4/\alpha^2) \cdot (F.\text{TPR} - F.\text{FPR})^{-2} \rceil$
  $\tau \leftarrow (F.\text{TPR} + F.\text{FPR}) \cdot \frac{1}{2}$
    $\triangleright$ *τ and η may be pre-computed if the classifier and α are fixed*
  $m, p \leftarrow S[(x.\text{IP}, x.\text{Port})]$
  $m \leftarrow m + 1$
  if $F(x) = 1$ **then** $\{\ p \leftarrow p + 1\ \}$
  label $\leftarrow$ `Reject`
  if $m \geq \eta$ **then**
    if $p/m \geq \tau$ **then**
      label $\leftarrow$ `Obfuscated`
    else
      label $\leftarrow$ `Benign`
    end if
  end if
  $S[(x.\text{IP}, x.\text{Port})] \leftarrow (m, p)$
  return *label*
end function

attack to larger address spaces, such as IPv6, at the cost of a few additional false positives.

We ran Algorithm 1 to detect obfs4 and obfs$\star$ in our test dataset. Note that the attack requires knowledge of the classifier's TPR and FPR. Although the classifier's "true" test-time TPR and FPR are unknown to the censor, it can use the values obtained during validation, which is how we instantiate the attack.

The results of the experiment are shown in Fig. 6. There are 410,911 destination hosts represented in the test set—importantly, no host in the test set was included in the training set (no example in the testing set influenced any parameter learned by the classifier or used in the host-based attack). First, Fig. 6a shows the number of hosts that will be classified for a particular value of η. The distribution is highly right-skewed, meaning that many destination hosts have just a few flows, but there is a significant tail of popular destinations with many flows in our dataset.

Fig. 6b and Fig. 6c show the *absolute* number of false positive host classifications made by the classifier. (As was the case with our previous empirical evaluations (§IV-B), the results reported are the average of ten independent trial runs.) The red line labeled with η in each plot shows the value computed by the algorithm for the CNN's validation TPR and FPR, and for $\alpha = 1 \times 10^{-6}$. In both figures, we see the expected trend, which is an exponential drop in false-positive classifications versus the number of flows observed. Note that the decrease in total false positives spans three orders of magnitude—e.g., in Fig. 6b at $x = 1$, y is greater than 10^3, but at $x = 20$, y is approximately 1. However, the total number of hosts classified (that is, not rejected) only decreases by a single order of magnitude (Fig. 6a).

For both obfs4 and obfs$\star$, the total number of false positive classifications drops to *less than 1* on average after no more than 30 flows have been observed—the absolute number of false positives reaches 0 after 38 flows have been observed for obfs$\star$. The actual host false positive rate achieved is 2.4×10^{-6} for obfs4 and 1.5×10^{-6} for obfs$\star$, which nearly matches the desired error rate $\alpha = 1 \times 10^{-6}$. This host error rate is two orders of magnitude smaller than the flow error rates achieved for these classifiers, and there are many orders of magnitude fewer hosts in a network than there are flows. Furthermore, a more conservative setting of α can further reduce the error rate at the expense of higher state costs.

In this evaluation, we focused only on the false positive classifications. It was indeed the case that all of our self-hosted bridges in the network were identified, although we did have an overall small number of bridges (15) with which we experimented.

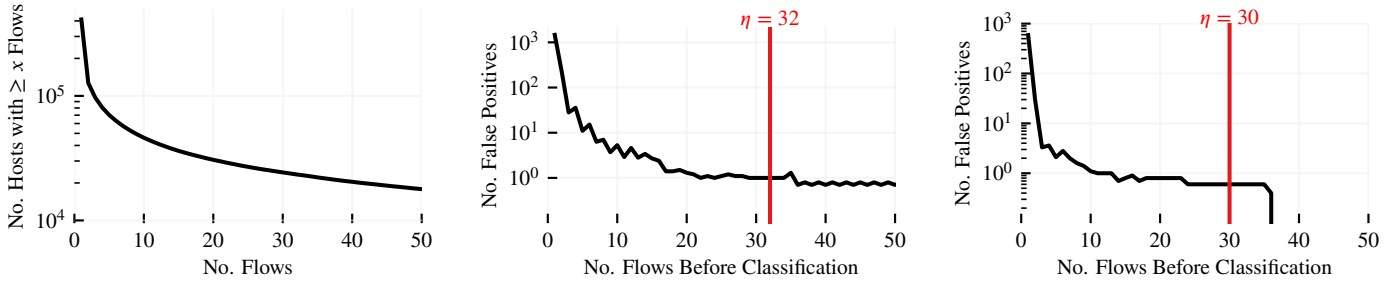
C. Discussion

A few limitations apply to the host-based classification method we proposed. First, the method will detect only servers that have handled sufficiently-many flows in a given period of time (the classifier "refuses" to classify hosts with fewer than η flows). This likely is acceptable for a censor who will still be able to readily block the most popular bridges accessed by users in the network. Second, this host-based classification technique is appropriate for only bridge-based circumvention systems with static bridges; it relies on consistent behavior being exhibited over time with respect to a given destination. Moreover, this technique does not apply to circumventing clients, which may exhibit a wide variety of both benign and circumventing activity.

Some prior work has considered similar host-based techniques. In 2004, Jung et al. proposed the threshold random walk algorithm to detect malicious port scanning [43]. Their approach is based on sequential hypothesis testing [78] and similarly aggregates multiple independent observations into a single host-level classification. Recently, Amich et al. [5] applied the threshold random walk algorithm to the censorship domain, using it to detect probing flows generated by the Geneva circumvention system [16]. This is another noteworthy host-based approach that could be used to leverage multiple observations into improved classifier precision.

In this work, focus on an approach that uses multiple independent observations made a by a flow-based classifier to achieve multiplicative improvements in performance with each additional observation. This is one possible approach, but many other possibilities exist in the broader space of host-based approaches and can be exploited by censors. As an example, a neural network classifier might be given a *collection* of flows to classify as a single input example. Providing the classifier with *multiple* flows might allow it to detect the presence of a fixed handshake for a protocol like TLS or the lack of one for a protocol like obfs4. Similar ideas have been explored in the domain of malware detection, but are under-explored in the domain of censorship and circumvention [6].

Overall, we believe that host-based techniques constitute a significant threat to bridge-based approaches. The host-based paradigm is more aligned with the goal of real-world censors, which tends to be focused on blocking particular hosts rather



(a) Distribution of hosts with at least x flows (b) False positives detecting obfs4 (c) False positives detecting obfs⋆

Figure 6: Results of the host-based attack using a CNN to detect obfs4 and obfs⋆.

than flows. Moreover, real-world networks have orders-of-magnitude fewer hosts than flows, which makes the problem of classification at scale more tractable. In light of these findings, we believe that new directions for circumvention protocols ought to be explored; we give recommendations in §X.

VII. Discussion

Our results highlight the importance of *realism* when evaluating the performance of censorship resistant systems. A censor's ability to distinguish between benign and circumventing flows may be understated or overstated if the set of assumptions made during training or testing are far from reality. Research that assumes unrealistic base rates or evaluates CRSes only in closed-world settings could easily incorrectly conclude that a CRS is secure when it is not, or that a censor is effective when it is not.

In comparison to prior work, we are less convinced that an adversary could effectively use a flow-based classifier at Internet scale to detect obfs4 traffic. For classifiers with even relatively low false positive rates, precision will vanish even at optimistic base rates (e.g., 10^{-3}). This means that a censor will inadvertently block thousands or more benign tail flows for every circumventing flow that is blocked, a level of collateral damage that may be unpalatable for the censor.

On the other hand, detecting *hosts* seems like a more viable strategy for a censor. Under realistic conditions—low base rates in an open-world setting—host-based analysis performs far better than flow-based detection. Detecting hosts is also a more natural approach for a censor, because its likely end goal is to curate a list of IP addresses used for circumvention that it can block wholesale. An important implication of our findings is that CRSes that depend both on not-publicly-advertised static ingress points and a fixed circumvention protocol may be defeated by a censor who performs host-based analysis. Our findings motivate the development of CRSes that avoid dependencies on fixed protocols and network locations.

An appeal for more research: Research in designing elusive, hard-to-detect transports is unfortunately underdeveloped. Deployed CRSes rely on circumvention techniques that appear to (mostly) work [31, 59], but lack rigorous scientific evaluation. In contrast, website fingerprinting (WF) has received considerably more study, with WF attacks now considering a deep-learning adversary and new WF defenses designed to defeat such an adversary. The development of obfuscated protocols might benefit from some of these advances (for example, using adversarial examples to confuse a classifier), but more research is needed.

This paper argues that more faithful and realistic evaluations of CRSes are needed to accurately understand their security and privacy properties. In order to empirically evaluate a system's ability to hide, background data is required. A serious impediment to such evaluation is the dearth of background data; there are unfortunately few datasets available that are appropriate for this task.

The sensitivity of real-world network data make it especially challenging to construct useful background traffic datasets for evaluating CRSes. Collecting the campus dataset used in this paper was a massive, complex, and time-consuming undertaking. We strongly encourage research that explores privacy-preserving methods of collecting and then publishing network flow datasets. Just as the WF literature is beginning to converge on standardized datasets, research in traffic circumvention could significantly benefit from the availability of background traces, both as a means of comparing CRSes and as a method of accelerating the pace of circumvention research.

We believe the research community should more strongly consider host-based analyses. The attacks described in §VI are technically simple and could presumably be implemented by a censor today. We posit that host-based approaches that are even more effective than those described in this paper could be constructed by examining hosts' network activity as a whole—for example, by considering the differences in protocol interactions (e.g., with DNS) between circumventing and benign flows on a suspected host. In general, host-based analysis is both under-explored and a significant threat to CRSes, warranting additional research in this area.

Defenses: Our work suggests several interesting directions for defenses. As mentioned previously, defending circumventing protocol traces with adversarial examples could be an effective strategy against a machine-learning censor.

Our findings show that if an circumvention protocol host has a fixed network location (IP) and a censor can observe some of its flows, it will likely be identified and subsequently blocked. One way to mitigate the effects of host-based analysis is to design circumvention systems to use multiple ephemeral bridges over time rather than a few long-term static bridges, similar to

the design of Snowflake. This would enable client connections to be spread across more bridges, and clients could more quickly switch if an ephemeral bridge is discovered and blocked by a censor. If the ephemeral bridges see low connection rates, the adversary might prefer to focus on identifying the client host rather than the bridge host. However, in this case, the client may have a considerable amount of background traffic not related to the circumvention protocol, and this background traffic could serve to confuse the host-based classification. Further study of host-based analysis to detect clients rather than bridges is an interesting area for future work.

Finally, programmable or polymorphic circumvention protocols, such as Marionette [26] or Proteus [77], could also benefit evasion. For example, if the adversary develops a classifier to detect one particular subprotocol in a polymorphic protocol family, then only the endpoints running that particular subprotocol will be blocked. Using a polymorphic protocol would force the adversary to develop a classifier to detect the entire polymorphic protocol family, which is a more difficult task and could be frustrated by frequently changing the polymorphic protocol over time.

VIII. Related Work

Many nation-states engage in censorship across a variety of topics using centralized and decentralized infrastructure. Some censorship techniques seen in real-world studies include host-based filtering, DNS hijacking, throttling, keyword blocking, DNS poisoning and injection, TCP RST injections, forced timeouts, IP and/or port blocking, HTTP filtering, internet blackouts, and SNI-based censorship [9, 65, 67, 75].

Numerous systems have been designed to resist censorship [44], including those based on mimicry [25, 26, 52, 80, 83], tunneling [11, 17, 29, 30, 39, 41, 47, 50, 52, 53, 70, 80, 91], and randomization [84, 86, 88]. Some obfuscators have been deployed as Tor pluggable transports, including Dust [84], Flashproxy [29], StegoTorus [83], FTE [25], and ScrambleSuit [86]. However, currently only obfs4 [88], and Snowflake [70] are officially supported in Tor Browser. Recent work has introduced formalism to help researchers and developers reason about the security of CRSes [42].

There have been numerous works presenting attacks on censorship circumvention systems. Houmansadr et al. argue that unobservability by imitation is fundamentally flawed [40], while Frolov and Wustrow showed that even systems attempting to mimic popular TLS versions (such as those used in web browsers) may still be identified using TLS fingerprints [33]. Censors can distinguish tunneled streams carrying covert traffic from those that do not using basic traffic analysis [40] or more sophisticated machine learning techniques [12] that can even operate at line speed directly on network switches [13]. Common problems are that mismatches between the use of a tunnel and its covert protocol enable identification [34], particularly during protocol initialization as in the case of attacks on obfs4 [79] and Snowflake [49]. Active probing may also be used to identify systems whose response (or lack thereof) is distinguishing [24, 27, 32].

Website fingerprinting is a related area of study in which the goal is to identify specific websites being visited, even when encryption is employed. Early methods use naïve-Bayes and other supervised learning algorithms to identify sites [18, 36, 60, 82]. More recently, deep learning has been applied to increase website fingerprinting accuracy [1, 14, 57, 66, 68], although these methods require large quantities of training data and must be regularly retrained [68]. Sirinam et al. reduce the amount of data needed to perform deep learning for fingerprinting purposes and aim to create a portable classifier using n-shot learning [69]; this approach has been used in a study of fingerprinting attacks on the real-world Tor network [20].

Another related area of study is application-based fingerprinting in which application-specific behaviors produce side channels [19] that can be detected by a network observer and used to learn potentially sensitive information about users [56]. Similar to our host-based approach, the observable behaviors when exercising multiple distinct parts of an application over time can be combined to produce more accurate predictions [46], even in the face of application multiplexing and other challenges [45].

IX. Data Protection and Ethics

Safety Measures: The data was collected on a university wireless network over a two week period during March and April 2022. The captured data included circumventing flows we generated along with normal campus wireless traffic. To ensure the privacy of campus users we took steps to minimize the data we collected, anonymize the data we did collect, and restrict access to that data. (We present the details of these steps below.) We did not collect any personal identifiable information and did our analysis over anonymized statistics that were computed over the data we captured. For our collection we used an existing network capture setup and data protection protocol that were approved by the campus institutional review board (IRB) and network operations staff.

To protect the collected data the capture was done by a machine that only allows network access from a small number of other campus machines and requires multi-factor authentication for access. In addition, the capture machine is located in a secure campus machine room with limited physical access. To protect the privacy of campus users the capture machine cryptographically anonymized all campus-based IP addresses originating or terminating captured connections. The anonymization of the IPs for the background traffic was done with a temporary, distinct key for each collection. §III. The key was immediately discarded at the completion of the collection so IP addresses could not be deanonymized.

On the capture machine the anonymized packets were reconstructed into flows and we extracted summary statistics about each flow including source and destination port, payload entropy, and packet size and direction, and saved only these statistics for analysis. Payload data was held in memory to compute per-flow statistics and then immediately discarded, never being written to persistent storage. Therefore, only flow data and packet headers, both containing anonymized IP addresses, were retained. Only this processed data ever left the secure capture machine. Furthermore, for all but one experiment even these anonymized IP addresses were removed and flows were identified solely by a randomly generated unique identification number. Identifying flows by their anonymized IP address was only used for the host-based network analysis

and for this only one member of the team had access to the mapping of anonymized IP addresses to flow ID numbers to allow grouping by IPs.

The background data we collected on the network was exclusively used to compare against circumvention protocol flows to see whether we could distinguish benign flows from our self-generated, circumventing traffic. Throughout the experiments all data remained on the campus network.

Disclosure: The goal of our work is to move censorship research in new directions that will lead to the development of stronger circumvention systems. We have been working closely with members of the Tor Project's anti-censorship team to improve obfs4 and Snowflake and will continue to share our research findings with them.

X. Conclusion

We explore 3 main areas in which we improve realism with respect to a censor attempting to detect circumvention protocols: (1) the environment in which the censor operates; (2) the precedence of circumvention protocols in the network (i.e., the base rate); and (3) the censors's capabilities. Using live network experiments, we find that the state-of-the-art censorship techniques are overwhelmed with false positives ($>94\%$) at even conservatively high base rates (10^{-3}). We explore how a recent deep learning approach from the website fingerprinting literature can improve precision under high base rates, and find that while precision does improve, it does not improve enough to be practical in realistic environments. We explore the effects of applying host-based analysis methods to the detection of circumvention protocols. We find that considering many flows from a host over time both significantly reduces the base rate under which an adversary must operate while at the same time increasing the precision in identifying the host as running a circumvention protocol. We conclude that host-based analysis is crucial for real world adversaries employing machine learning to detect obfuscated protocols.

Our work has illuminated several new directions for censorship research, as discussed in §VII. Chiefly, host-based analysis methods may be improved beyond our contributions in this paper, where packet payloads could be considered or different deep learning methods could be employed to further improve the classification results. Importantly, we focus on exploring realistic censorship adversaries *in service of understanding how to develop stronger CRSes*. As such, we also outline several promising directions based on our insights for developing defenses against censorship that deserve more attention; these include employing adversarial examples, using ephemeral circumvention servers, and developing new programmable or polymorphic circumvention protocols. Finally, future work should consider applying a privacy-preserving collection methodology to safely collect and share realistic datasets to make censorship research more accessible. Despite the concerning results of our evaluation, we are optimistic that these new directions will lead to stronger CRSes that will be increasingly difficult to block.

Acknowledgements

This work has been partially supported by the Office of Naval Research (ONR), the Defense Advanced Research Projects Agency (DARPA) (including under Contract No. FA8750-19-C-0500), and the Callahan Family Chair Fund.

References

[1] K. Abe and S. Goto, "Fingerprinting attack on Tor anonymity using deep learning," *Proceedings of the Asia Pacific Advanced Network*, vol. 42, 2016.

[2] A. Akbari, "Shutting down the internet is another brutal blow against women by the Iranian regime," *The Guardian*, 2022. [Online]. Available: https://www.theguardian.com/commentisfree/2022/sep/26/elon-musk-iran-women-mahsa-amini-feminists-morality-police.

[3] I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "A look behind the curtain: Traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, no. 1, 2021.

[4] Alice, Bob, Carol, J. Beznazwy, and A. Houmansadr, "How China detects and blocks Shadowsocks," in *Proceedings of the 2020 ACM Internet Measurement Conference*, (Virtual Event, USA), ACM, 2020.

[5] A. Amich, B. Eshete, V. Yegneswaran, and N. P. Hoang, "DeResistor: Toward detection-resistant probing for evasion of Internet censorship," in *Proceedings of the 32nd USENIX Security Symposium*, (Anaheim, CA), USENIX Assn, 2023.

[6] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, (Vienna, AT), ACM, 2016.

[7] Anonymous. "Sharing a modified Shadowsocks as well as our thoughts on the cat-and-mouse game." Net4People BBS Forum Posting. (2022), [Online]. Available: https://github.com/net4people/bbs/issues/136.

[8] Anonymous, K. Bock, J. Sippe, Shelikhoo, D. Fifield, E. Wustrow, D. Levin, and A. Houmansadr, "Exposing the Great Firewall's dynamic blocking of fully encrypted traffic," Open Technology Fund, 2022.

[9] S. Aryan, H. Aryan, and J. A. Halderman, "Internet censorship in Iran: A first look," in *3rd USENIX Workshop on Free and Open Communications on the Internet*, (Washington, DC), USENIX Assn, 2013.

[10] *AWS top 1M sites list*, Amazon. [Online]. Available: https://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[11] D. Barradas, N. Santos, and L. Rodrigues, "DeltaShaper: Enabling unobservable censorship-resistant TCP tunneling over videoconferencing streams," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, 2017.

[12] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," in *Proceedings of the 27th USENIX Security Symposium*, (Baltimore, MD), USENIX Assn, 2018.

[13] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proceedings of the 2021 Network and Distributed System Security Symposium*, (Virtual Event), ISOC, 2021.

[14] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, 2019.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] K. Bock, G. Hughey, X. Qiang, and D. Levin, "Geneva: Evolving censorship evasion strategies," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, (London, UK), ACM, 2019.

[17] C. Brubaker, A. Houmansadr, and V. Shmatikov, "CloudTransport: Using cloud storage for censorship-resistant networking," in *Privacy Enhancing Technologies: 14th International Symposium Proceedings*, (Amsterdam, NL), ser. LNCS, vol. 8555, Springer, 2014.

[18] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (Raleigh, NC), ACM, 2012.

[19] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, (Oakland, CA), IEEE, 2010.

[20] G. Cherubin, R. Jansen, and C. Troncoso, "Online website fingerprinting: Evaluating website fingerprinting attacks on Tor in the real world," in *Proceedings of the 31st USENIX Security Symposium*, (Boston, MA), USENIX Assn, 2022.

[21] H. Davidson, "China brings in 'emergency' level censorship over zero-Covid protests," *The Guardian*, 2022. [Online]. Available: https://www.theguardian.com/world/2022/dec/02/china-brings-in-emergency-level-censorship-over-zero-covid-protests.

[22] R. Dingledine, *Next steps for unclassifiable protocols*, Tor Project mailing list, 2019. [Online]. Available: https://lists.torproject.org/pipermail/anti-censorship-team/2019-May/000015.html.

[23] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, (San Diego, CA), USENIX Assn, 2018.

[24] A. Dunna, C. O'Brien, and P. Gill, "Analyzing China's blocking of unpublished Tor bridges," in *8th USENIX Workshop on Free and Open Communications on the Internet*, (Baltimore, MD), USENIX Assn, 2018.

[25] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol misidentification made easy with format-transforming encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, (Berlin, DE), ACM, 2013.

[26] K. P. Dyer, S. E. Coull, and T. Shrimpton, "Marionette: A programmable network traffic obfuscation system," in *Proceedings of the 24th USENIX Security Symposium*, (Washington, DC), USENIX Assn, 2015.

[27] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson, "Examining how the Great Firewall discovers hidden circumvention servers," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, (Tokyo, JP), ACM, 2015.

[28] R. Fedasiuk, "Buying silence: The price of Internet censorship in China," *China Brief*, vol. 21, no. 1, 2020.

[29] D. Fifield, N. Hardison, J. Ellithorpe, E. Stark, D. Boneh, R. Dingledine, and P. Porras, "Evading censorship with browser-based proxies," in *Privacy Enhancing Technologies: 12th International Symposium Proceedings*, (Vigo, ES), ser. LNCS, vol. 7384, Springer, 2012.

[30] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, 2015.

[31] A. Filastó and J. Appelbaum, "OONI: Open observatory of network interference," in *2nd USENIX Workshop on Free and Open Communications on the Internet*, (Bellevue, WA), USENIX Assn, 2012.

[32] S. Frolov, J. Wampler, and E. Wustrow, "Detecting probe-resistant proxies," in *Proceedings of the 2020 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2020.

[33] S. Frolov and E. Wustrow, "The use of TLS in censorship circumvention," in *Proceedings of the 2019 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2019.

[34] J. Geddes, M. Schuchard, and N. Hopper, "Cover your ACKs: Pitfalls of covert channel censorship circumvention," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, (Berlin, DE), ACM, 2013.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). MIT Press, 2016.

[36] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *CCS '09 Workshops*, *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, (Chicago, IL), ACM, 2009.

[37] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, 1963.

[38] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, (Virtual Event, KR), ACM, 2021.

[39] J. Holowczak and A. Houmansadr, "CacheBrowser: Bypassing Chinese censorship without proxies using cached content," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, (Denver, CO), ACM, 2015.

[40] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, (San Francisco, CA), IEEE, 2013.

[41] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer, "I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2013.

[42] J. K. Howes IV, M. Georgiou, A. J. Malozemoff, and T. Shrimpton, "Security foundations for application-based covert communication channels," in *Proceedings of the 43rd IEEE Symposium on Security and Privacy*, (San Francisco, CA), IEEE, 2022.

[43] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, (Oakland, CA), IEEE, 2004.

[44] S. Khattak, T. Elahi, L. Simon, C. M. Swanson, S. J. Murdoch, and I. Goldberg, "SoK: Making sense of censorship resistance systems," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, 2016.

[45] J. Li, S. Wu, H. Zhou, X. Luo, T. Wang, Y. Liu, and X. Ma, "Packet-level open-world App fingerprinting on wireless traffic," in *Proceedings of the 2022 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2022.

[46] J. Li, H. Zhou, S. Wu, X. Luo, T. Wang, X. Zhan, and X. Ma, "FOAP: Fine-grained open-world android app fingerprinting," in *Proceedings of the 31st USENIX Security Symposium*, (Boston, MA), USENIX Assn, 2022.

[47] S. Li, M. Schliep, and N. Hopper, "Facet: Streaming over videoconferencing for censorship circumvention," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, (Scottsdale, AZ), ACM, 2014.

[48] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, (Virtual Event, FR), ACM, 2022.

[49] K. MacMillan, J. Holland, and P. Mittal, "Evaluating Snowflake as an indistinguishable censorship circumvention tool," arXiv preprint, 2020.

[50] R. McPherson, A. Houmansadr, and V. Shmatikov, "CovertCast: Using live streaming to evade Internet censorship," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, 2016.

[51] J. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt, "Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization," in *Proceedings of the 38th International Conference on Machine Learning*, (Virtual Event), ser. PMLR, vol. 139, 2021.

[52] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph: Protocol obfuscation for Tor bridges," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (Raleigh, NC), ACM, 2012.

[53] M. Nasr, H. Zolfaghar, A. Houmansadr, and A. Ghafari, "Massbrowser: Unblocking the censored web for the masses, by the masses," in *Proceedings of the 2020 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2020.

[54] A. A. Niaki, S. Cho, Z. Weinberg, N. P. Hoang, A. Razaghpanah, N. Christin, and P. Gill, "ICLab: A global, longitudinal Internet censorship measurement platform," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy*, (Virtual Event, USA), IEEE, 2020.

[55] "NVIDIA Tesla v100 16gb," Amazon. (2022), [Online]. Available: https://a.co/d/3ExzAjE (visited on 08/16/2023).

[56] S. E. Oh, S. Li, and N. Hopper, "Fingerprinting keywords in search queries over Tor," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, 2017.

[57] S. E. Oh, S. Sunkam, and N. Hopper, "p-FP: Extraction, classification, and prediction of website fingerprints with deep learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, 2019.

[58] Open Internet Tools Project, "Collateral freedom: A snapshot of Chinese internet users circumventing censorship," 2013.

[59] Open Observatory of Network Interference. "Tor bridge reachability." (Ca. 2020), [Online]. Available: https://ooni.org/nettest/tor-bridge-reachability/ (visited on 08/25/2023).

[60] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop Privacy in the Electronic Society*, (Chicago, IL), ACM, 2011.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, 2011.

[62] J. Piet, A. Sharma, V. Paxson, and D. Wagner, "Network detection of interactive SSH impostors using deep learning," in *Proceedings of the 32nd USENIX Security Symposium*, (Anaheim, CA), USENIX Assn, 2023.

[63] L. Quan, J. Heidemann, and Y. Pradkin, "When the Internet sleeps: Correlating diurnal networks with external factors," in *Proceedings of the 2014 ACM Internet Measurement Conference*, (Vancouver, CA), ACM, 2014.

[64] R. S. Raman, A. Stoll, J. Dalek, R. Ramesh, W. Scott, and R. Ensafi, "Measuring the deployment of network censorship filters at global scale," in *Proceedings of the 2020 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2020.

[65] R. Ramesh, R. S. Raman, M. Bernhard, V. Ongkowijaya, L. Evdokimov, A. Edmundson, S. Sprecher, M. Ikram, and R. Ensafi, "Decentralized control: A case study of Russia," in *Proceedings of the 2020 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2020.

[66] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proceedings of the 2018 Network and Distributed System Security Symposium*, (San Diego, CA), ISOC, 2018.

[67] K. Singh, G. Grover, and V. Bansal, "How India censors the web," in *Proceedings of the 12th ACM Conference on Web Science*, (Southampton, UK), ACM, 2020.

[68] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, (Toronto, CA), ACM, 2018.

[69] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, (London, UK), ACM, 2019.

[70] "Snowflake," Tor Project. (Ca. 2018), [Online]. Available: https://snowflake.torproject.org/ (visited on 05/13/2022).

[71] *Statistical information for the CAIDA anonymized internet traces*, Center for Applied Internet Data Analysis, 2019. [Online]. Available: https://www.caida.org/catalog/datasets/passive_trace_statistics/.

[72] The Tor Project, *obfs2 (the Twobfuscator)*, Protocol specification, version 2bf9d096, 2015. [Online]. Available: https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/obfsproxy/-/blob/master/doc/obfs2/obfs2-protocol-spec.txt (visited on 05/13/2022).

[73] The Tor Project, *obfs3 (the Threebfuscator)*, Protocol specification, version 225e420c, 2013. [Online]. Available: https://gitlab.torproject.org/tpo/anti-censorship/pluggable-transports/obfsproxy/-/blob/2bf9d096bb45a4e6c69f1cbdc3d2565f54a44efc/doc/obfs3/obfs3-protocol-spec.txt (visited on 05/13/2022).

[74] A. Troianovski and V. Safronova, "Russia takes censorship to new extremes, stifling war coverage," *The New York Times*, 2022. [Online]. Available: https://www.nytimes.com/2022/03/04/world/europe/russia-censorship-media-crackdown.html.

[75] M. C. Tschantz, S. Afroz, Anonymous, and V. Paxson, "SoK: Towards grounding censorship circumvention in empiricism," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy*, (San Jose, CA), IEEE, 2016.

[76] US Department of State, "A declaration for the future of the internet," 2022. [Online]. Available: https://www.state.gov/declaration-for-the-future-of-the-internet.

[77] R. Wails, R. Jansen, A. Johnson, and M. Sherr, "Proteus: Programmable protocols for censorship circumvention," in *Free and Open Communications on the Internet 2023*, (Lausanne, CH), 2023.

[78] A. Wald, "Sequential tests of statistical hypotheses," *The Annals of Mathematical Statistics*, vol. 16, no. 2, 1945.

[79] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton, "Seeing through network-protocol obfuscation," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, (Denver, CO), ACM, 2015.

[80] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "CensorSpoofer: Asymmetric communication using IP spoofing for censorship-resistant web browsing," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (Raleigh, NC), ACM, 2012.

[81] T. Wang, "High precision open-world website fingerprinting," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy*, (Virtual Event, USA), IEEE, 2020.

[82] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Security Symposium*, (San Diego, CA), USENIX Assn, 2014.

[83] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, "StegoTorus: A camouflage proxy for the Tor anonymity system," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (Raleigh, NC), ACM, 2012.

[84] B. Wiley, "Dust: A blocking-resistant Internet transport protocol," University of Texas at Austin, Tech. Rep., 2011.

[85] P. Winter and S. Lindskog, "How the Great Firewall of China is blocking Tor," in *2nd USENIX Workshop on Free and Open Communications on the Internet*, (Bellevue, WA), USENIX Assn, 2012.

[86] P. Winter, T. Pulls, and J. Fuss, "ScrambleSuit: A polymorph network protocol to circumvent censorship," in *Proceedings of the 12th Annual ACM Workshop Privacy in the Electronic Society*, (Berlin, DE), ACM, 2013.

[87] M. Wu, J. Sippe, D. Sivakumar, J. Burg, P. Anderson, X. Wang, K. Bock, A. Houmansadr, D. Levin, and E. Wustrow, "How the Great Firewall of China detects and blocks fully encrypted traffic," in *Proceedings of the 32nd USENIX Security Symposium*, (Anaheim, CA), USENIX Assn, 2023.

[88] Yawning Angel, *obfs4 (the obfourscator)*, Protocol specification, version c0898c2, 2019. [Online]. Available: https://github.com/Yawning/obfs4/blob/master/doc/obfs4-spec.txt (visited on 05/13/2022).

[89] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, 1950.

[90] R. Zhao, X. Deng, Z. Yan, J. Ma, Z. Xue, and Y. Wang, "MT-FlowFormer: A semi-supervised flow transformer for encrypted traffic classification," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (Washington, DC), ACM, 2022.

[91] H. Zolfaghari and A. Houmansadr, "Practical censorship evasion leveraging content delivery networks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, (Vienna, AT), ACM, 2016.