Network Traffic Obfuscation: An Adversarial Machine Learning Approach

Gunjan Verma[1], Ertugrul Ciftcioglu[2], Ryan Sheatsley[1], Kevin Chan[1], Lisa Scott[1]
[1]Army Research Laboratory; {gunjan.verma.civ, ryan.m.sheatsley.civ, kevin.s.chan.civ, lisa.m.scott92.civ}@mail.mil
[2]Oak Ridge Associated Universities; enciftcioglu@gmail.com

***Abstract*—An agent (D) aims to defend a network's traffic (T) from inference (classification) of applications or protocols (P) traversing that nework by an attacker (A). D aims to confuse A as to the nature of T by altering T to T′ so that A cannot easily ascertain the class of T′. If D is successful, A concludes that T′ belongs to class Q different from the true class P. A variety of approaches have been advanced to this general problem in the primary literature; however, research shows that even if the *data* contents of T are altered (e.g., through encryption), the *meta-data* aspects of T and T′ are similar (e.g., similar packet statistics like size and inter-arrival time). Thus, inference of P is still possible from observing the statistical properties of T′; D must thus further obfuscate these features as well. However, heavy-handed obfuscation could break the protocol or incur substantial overhead; hence minimal perturbations are desired. In this paper, we assume that A is able to observe statistical properties of T. We study the question: how can D optimally create T′ so that A infers T′ belongs to a class other than the true class P, with the additional constraint that T′ is close to T? Insights from the emerging area of adversarial machine learning (AML) provide unique perspectives in answering this question.**

***Index Terms*—network traffic analysis, traffic obfuscation, adversarial machine learning**

I. Introduction

An agent D aims to defend a network from an adversary A. A seeks to infer something about the traffic (T) flowing over D's network, while D aims to sabotage this inference. There are countless motivations for A; for example, oppressive governments may seek to curtail access to certain Internet resources, while Internet service providers (ISP) may aim to shape traffic to meet certain performance benchmarks (e.g., Quality of Service (QoS)). Fundamentally, D seeks to alter T to a modified flow T′ so that A misidentifies T′ as belonging to a different application or protocol class (Q) than the true one (P). We assume that this modification can be inverted by the intended recipient.

We begin by considering the techniques A may employ to infer P. A's most popular historical approach is examination of port numbers, which typically map uniquely to protocols [1]. However, nothing compels network traffic to to use particular port numbers, and it is common for peer-to-peer applications, among others, to intentionally avoid this practice. A popular approach going beyond port numbers is deep packet inspection (DPI). This method pattern matches known signatures [2] in the (unencrypted) portions of packets that are characteristic of a given protocol; various open [3] and propietary implementations exist. DPI challenges are twofold; one, it requires a continuously updated list of signatures (as protocols evolve) and two, it requires access to raw payload data, which may be illegal and will not survive encryption. Beyond payload analysis, A may use statistical signatures of traffic flows such as packet size and inter-arrival time [4]. It is this type of approach that we assume A employs in this paper.

Next, we review techniques that D may use to obfuscate A's inference.

A. Data obfuscation strategies

These are two basic strategies that D can use to obfuscate the data of T.

1) Encryption: Encryption removes fingerprints from T's data; however, fingerprintable meta-data features remain, such as packet size and interarrival times.

2) Mimicry: D shapes T so that it mimics, in some way, a protocol other than the true one. Mimicry can occur at various levels; for example, one can inject regular expressions into packets that make them resemble other protocols. Various implementations of mimicry exist such as SkypeMorph [5]. Another approach to mimicry is *tunneling*; D embeds T within a cover protocol, i.e., the underlying data stream's protocol is infused into an overlay protocol. However, previous research [6] has shown that protocols can be successfully identified even when tunnelled inside other protocols (such as HTTP or SSH) since statistical fingerprints such as packets sizes and inter-arrival times are still preserved.

B. Motivation for statistical obfuscation

We have indicated that data obfuscation strategies do not perturb statistical fingerprints (meta-data) of T. Are these statistical signatures sufficient for A to classify T? The answer is generally yes; as alluded to, research [6]–[8] has shown that encrypted traffic can be accurately classified based on statistical features including packet and payload byte counts, packet sizes and packet rates for each direction (i.e., client to server, server to client). It is this fact that justifies our claim

[2]Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-16-2-0014 (ARL ORAU Research Associate Program). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

that to ensure secrecy of network contents, D must obfuscate at the meta-data level.

In this paper, we assume that one or more of the data obfuscation techniques (for example, encryption) have been applied to T. However, the modified traffic (which we also refer to as T) still exhibits statistical signatures characteristic of the true protocol P. We presume that A is using a traffic classifier based on statistical features; hence D should additionally shape T's statistical characteristics so they are not representative of P. In addition, D seeks that T′ is close to T, because large alterations to the traffic stream are costly for D (and may break protocol or application specific standards, e.g. intolerable delays in VoIP applications).

D may statistically obfuscate T in many ways; e.g., she may delay packets [9] or insert dummy packets into the traffic [10]. Implementations adding transport-layer security, such as GnuTLS [11], obfuscate by adding random amounts of extra padding to the plaintext prior to encryption. However, there is no consideration given to doing this optimally.

This paper seeks to answer the question: how should D achieve statistical obfuscation "optimally", i.e., expending the minimal amount of resources? Obfuscation trades off privacy with quality of service and ease of implementation; e.g., higher obfuscation could mean lower throughput and higher protocol complexity, for example. The goal of optimal obfuscation is to minimize unnecessary overhead. Traffic morphing techniques with some notion of optimal design [12], [13] modify the traffic pattern by altering packet sizes and inserting dummy packets while aiming to minimize overhead. In earlier work, we considered the problem of optimally allocating a fixed budget of chaff resources towards altering packet size and inter-arrival times [14]. [12] is a prominent work in optimal statistical obfuscation, but the method presented therein suffers from two major drawbacks: i) it only works for discrete-valued features (like packet sizes, but not interarrival times), ii) it assumes, in effect, that A is using a nearest-neighbor classifier (so that the goal of obfuscation is to make feature vectors of the original class closely resemble those of another class). Modern ML systems use more complex classifiers like neural networks, necessitating new approaches to obfuscation.

C. Statistical obfuscation with adversarial machine learning

In this paper, we propose D achieve its goal of statistical obfuscation by employing ideas from adversarial machine learning (AML). AML is an emerging science studying the phenomena that all machine learning methods misclassify examples that only slightly differ (i.e., have small normed difference) from correctly classified examples [15]. The fact that the same adversarial example is often misclassified across a wide variety of models and training sets with different architectures implies that adversarial examples are a fundamental phenomenon. AML techniques have found widespread application in multimedia (images, audio) settings, but have received far less attention elsewhere, including for network traffic classification.

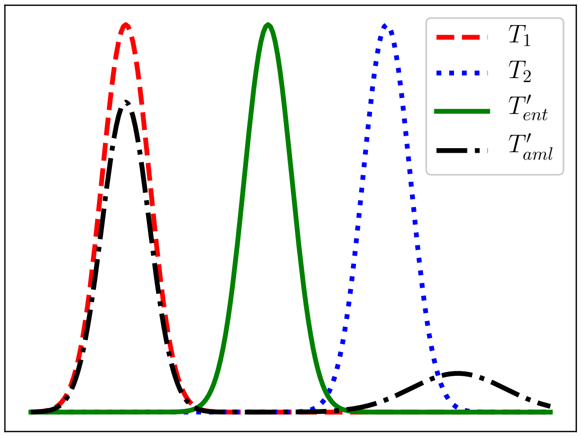

Fig. 1. Schematic illustration of various obfuscation strategies. Details given in text.

Much research exists on the reasons for the existence of adversarial examples, as well as proposed defenses against them. However, at this time that there is no robust, general defense against adversarial examples [16]. In addition, adversarial examples plague all known machine learning methods. Finally, adversarial examples tend to generalize, meaning that an example that tricks one model is able to trick another [17]. These observations suggest that adversarial examples are prevalent and generalizable phenomena.

The fact that adversarial examples are extremely close to benign examples undermines the notion that in order to fool a machine learning system, one must make the source class resemble the target class (as assumed in [12]). Indeed, AML suggests that far more parsimonious perturbations exist to the inputs which cause misclassification.

Figure 1 schematically contrasts the key ideas between conventional and AML based obfuscation. Consider two traffic classes T_1 and T_2 whose distribution of, say, packet sizes is as shown in the figure. If D wishes to obfuscate T_1, she may elect to alter T_1's distribution to closely match that of T_2; this incurs a high obfuscation cost. Another alternative is that D may choose to obfuscate to T'_{ent}; this can be thought of as the maximum entropy distribution since now A is equally uncertain as to which of the two traffic classes D is employing. The obfuscation espoused by an AML approach is given by T'_{aml}. To the human eye, T'_{aml} is clearly closer to T_1, but the key point is that ML classifiers will classify it as T_2 since it is outside the training data manifold and is at a point in feature space that favors T_2. The science of AML is in discovering such vulnerabilities. Clearly, T'_{aml} incurs the lowest obfuscation cost.

II. Model

In this section we specify models of the adversary and defender.

A. Adversary Model

We assume A observes statistical signatures of network traffic of a particular flow between two nodes, specifically packet sizes and inter-packet arrival times. We assume A is able to observe these signatures in both directions (source to destination and vice-versa). We assume that the model has been trained on a dataset representative of the types of traffic A seeks to classify. A's goal is to infer the application class P in use by D. Concretely, A seeks to classify the observed traffic T into one of N_p possible classes $\{P_1, P_2, \cdots, P_{N_p}\}$. For a given collection of N_f features, $x_1, x_2, \cdots x_{N_f}$, denoted compactly by the feature vector $\mathbf{x}$, A computes $F(\mathbf{x})$, where F denotes the classifier which outputs N_p values, a probability distribution over the possible classes.

B. Defender Model

Recall that D seeks to create a traffic stream T$'$ which statistically obfuscates A. We assume that D knows the features A is using as well as A's model F. This is not as big an assumption as it appears; indeed, the transferrability result of AML states that, qualitatively speaking, the same adversarial example can fool any current machine learning method regardless of design architecture. In other words, D can construct a surrogate F' of A's model, and design T$'$ to obfuscate F'. Transferrability [17] states that T$'$ is likely to obfuscate F as well.

In creating the obfuscation, D must respect constraints, namely:

- Legality constraints: In order to conform to the basic rules governing the actual protocol in use, packets may need to obey certain requirements, e.g. on their minimum or maximum size
- Performance constraints: The task which D's network is supporting (e.g., voice communications) will impose certain performance requirements (e.g., maximum latency).

Both of these constraints imply that D seeks to *minimally* alter T, since large alterations could lead to non-compliance with the protocol or unacceptable performance. AML is focused precisely on the phenomenon of small input perturbations.

C. Generation of Obfuscated Traffic

We briefly summarize the key idea here behind crafting adversarial examples here; more detailed mathematical ideas can be found, e.g., in [18]. One of the seminal approaches to generating adversarial examples is presented in [19]. The authors use the L-BFGS optimization procedure to perturb a given input $\mathbf{x}$ to a different input $\mathbf{x}'$ in order to minimize

$$\min||\mathbf{x} - \mathbf{x}'||_2 + \lambda L(F(\mathbf{x}'), t) \tag{1}$$

where the first term penalizes large perturbations from $\mathbf{x}$ and the second term penalizes departure from the target class t to be perturbed to. L captures the (non-negative) loss between t and the class assigned by the classifier $F(\mathbf{x}')$; $\lambda > 0$ is a tradeoff parameter.

TABLE I
CLASS COMPOSITION AND COUNTS FOR DATASET ANALYZED IN THIS PAPER

Classification	Flow Types	Number
BULK	FTP	11539
DATABASE	postgres,sqlnet,oracle,ingres	2648
MAIL	imap,pop2/3,smtp	28567
SERVICES	X11,dns,ident,ldap,ntp	2099
P2P	KaZaA,BitTorrent,GnuTella	2094
WWW	www	328091

A myriad of subsequent adversarial example crafting methods have been reported in the literature that exploit the gradient of the loss function [15] or of the target classification [20]. By using norms other than the 2-norm and a variety of loss functions L, [18] extends the approach in (1) to a family of crafting algorithms; we employ this method in this work as it among the strongest currently known methods.

In this work, we enforce the most fundamental constraints on the features, i.e., that packet sizes and inter-arrival times cannot be negative, and that the cumulative distribution of any feature must be monotone non-decreasing. We do this by re-parameterizing the features.

III. Classifier Model Setup

In this section we describe the data and classification model used in this paper.

A. Dataset

The dataset we use has been extensively considered by the traffic classification literature and was originally studied in [21]. A Gigabit Ethernet link was monitored at a university setting hosting more than 1000 researchers and staff. Traffic on both link directions was captured for a 24 hour period over a weekday in 2003. A single day's trace was collected and split into ten blocks of approximately 28 minutes each. From each flow (i.e., a collection of packets between two hosts), features are extracted which are derived only from packet header information. Each flow is manually classified into one of several categories. The class descriptors, the types of flows belonging to each class, and the number of flows per class is given in Table I.

Note that [21] contains 4 more classes than those considered here; we did not consider classes with a small numer of examples (less than 2000), since the sparse number of such training examples adversely impacts classifier performance on those classes.

B. Feature selection

[22] details the full list of 249 features collected on each flow. A key distinction of our work with the majority of previous works on traffic classification is that we only retain features pertaining to packet size and packet interarrival time. By focusing exclusively on these, we liberate the adversary from needing to examine the contents of the payload and header. In particular, we note that [21] mentions that the "most discriminative" features include port numbers, which we do

TABLE II
PER-CLASS ACCURACIES OF NETWORK TRAFFIC CLASSIFIER TRAINED ON PACKET SIZE AND INTER-ARRIVAL TIME FEATURES

Classification	Accuracy
BULK	.99
DATABASE	.97
MAIL	.98
SERVICES	1.0
P2P	.75
WWW	.97

TABLE III
ACCURACY AND CLASS DISTRIBUTION OF ADVERSARIAL ATTACKS

Category	Accuracy	Distribution
BULK	.88	.12/.04/.52/0/.16/.16
DATABASE	.96	0/.04/.04/.2/.08/.64
MAIL	.76	.08/.08/.24/0/.52/.08
SERVICES	.52	.08/.44/0/.48/0/0
P2P	.36	0/0/.04/0/.64/.32
WWW	.84	0/.08/0/0/.76/.16

not use in this work. Thus, of the 249 total features present in this dataset, we retain only 20. These retained features are comprised of the $0, 25, 50, 75, 100$ percentiles of the IP packet sizes, respectively, sent from client to server, as well as from server to client. Also, $0, 25, 50, 75, 100$ percentiles of the packet interarrival times (both from client to server, and server to client) are included. We refer to these percentiles as the "distribution" for short. In particular, we ignore several features involving ACK and SACK counts and features based on counts of various TCP header bits.

C. Model training

We first randomly permute and then partition the data into 5000 validation flows, 5000 test flows, and keep the remaining flows as training data. Because of the large class imbalance, we randomly upsample the training data so that all classes have the same number of training examples. We train a 3 layer neural network with 300, 200, and 100 hidden units. Features are normalized to live in the range $[0, 1]$. Rectified linear activation functions are used for all hidden units. We divide the training data into minibatches of size 1000 and train for 300 epochs. The Adam optimizer [23] is used and the learning rate is decayed from 10^{-4} to 10^{-6} over the entire run.

D. Classification results

We report our results in term of per class accuracy, or the fraction of all flows of a given class that were correctly classified as belonging to that class, in Table II. The overall (aggregating across all classes) training, validation, and test accuracies are all .97, to two decimal places. Note that the training data has equal class distribution, whereas the validation and test data are drawn from the original data distribution (indicated in Table I).

IV. GENERATION OF OBFUSCATED FEATURES

In this section we discuss the generation of statistically obfuscated traffic features in more detail. We use the software library CleverHans [24], which provides a software framework for generating adversarial examples (using a variety of recently developed algorithms). While there are several methods for creating adversarial examples, we use the so-called "Carlini-Wagner L2 algorithm" (CW for short) [18] as it is among the most effective methods in generating perturbations with small norms. For each of the 6 application classes studied in this paper, we randomly choose 50 inputs from the test set. For each of these inputs, we create a modified input using CW. The modified input is then evaluated by the traffic classifier to determine its label; the proportion of *incorrect* labels is reported in column 2 of Table III as the accuracy of the attack. Higher numbers indicate the modified input was more successful in confusing the classifier. Column 3 shows the distribution of the classification of the 50 adversarial inputs across all 6 categories (where the sequence of numbers mirrors the alphabetical order of column 1). For example, in row 1 of the table, the value of .52 in the third element of the distribution column indicates that a fraction .52 of BULK traffic was misclassified as MAIL traffic. The value of .12 in the first element of row 1, column 3 indicates that a fraction .12 of BULK traffic was still classified as BULK after obfuscation; this is why the accuracy is $.88 (= 1 - .12)$ in column 2.

We can draw several inferences from Table III. For example, most of obfuscated database traffic is classified as WWW. Figure 2 shows boxplots depicting the distributions of packet size and packet time for original and adversarially perturbed inputs. In particular, for a given category and of the 50 samples obfuscated in that category, we select the sample having the median norm perturbation. Corresponding to that sample's original and obfuscated versions, we plot both the packet size and time distribution as boxplots. We repeat this process across all 6 categories. The key takeaway from the figure is that T' is almost indistinguishable from T in terms of packet size, while the median of packet inter-arrival time of T' is within an order of magnitude of that of T (inter-arrival times are plotted on a log scale to facilitate comparison).

V. DISCUSSION

The results from Figure 2 capture the essential finding that reasonably minimalistic strategies are available to D to obfuscate its network traffic. Packet size distributions need to be altered minimally; inter-arrival time distributions require somewhat more alteration. Unlike previous work which assumes that A uses a nearest neighbor classifier with discrete-valued features, our work is more general; we use the well-established field of AML to systematically generate minimalistic perturbations across feature space against complex classifiers. Furthermore, our approach is trivially adaptable to other features (besides packet size and time) that A may employ.

Our results depend, of course, on the adversarial generation mechanism used; we tried other algorithms which produced

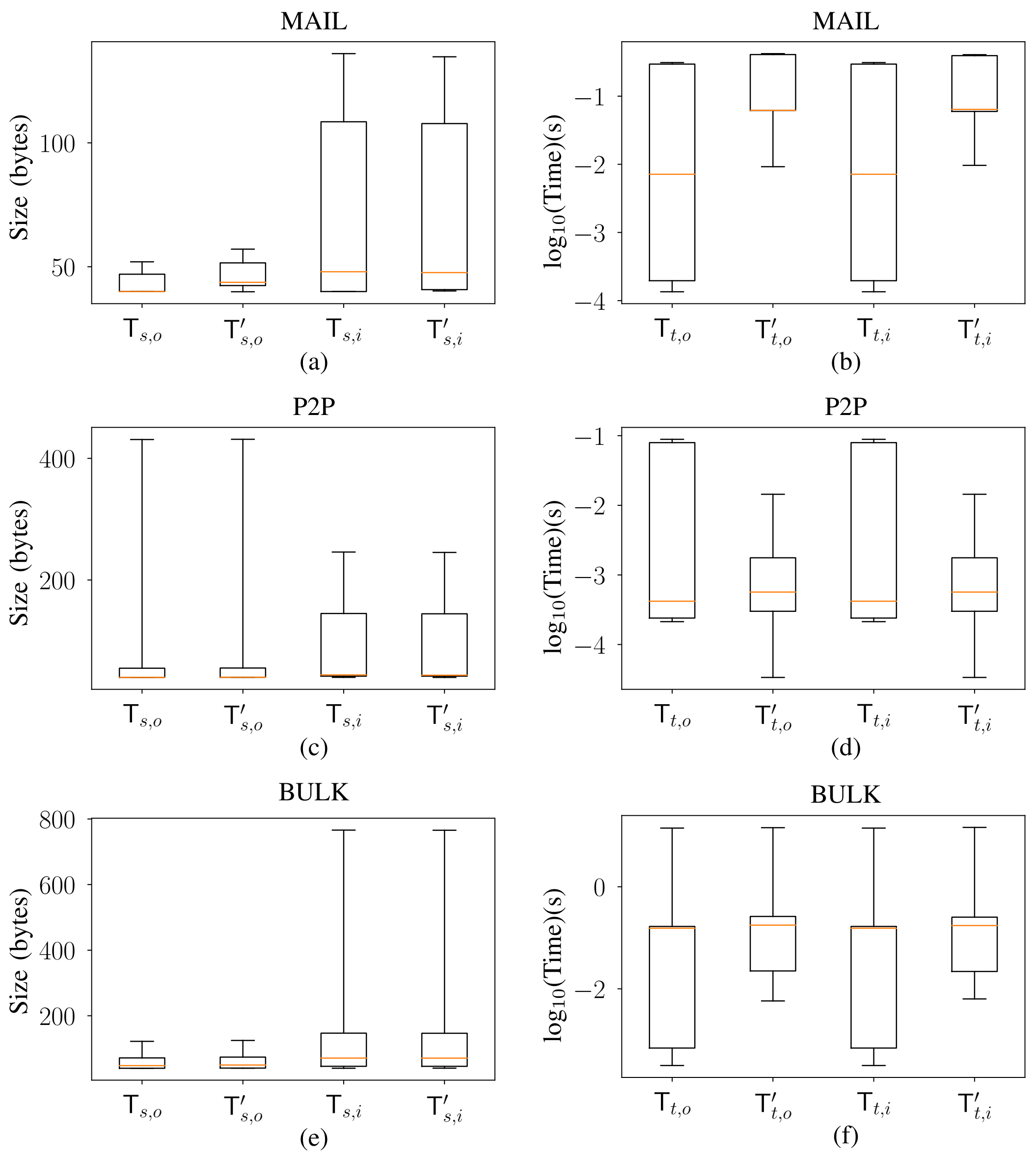

Fig. 2. Boxplots for 3 of the 6 classes studied in this paper (the remaining are qualitatively similar and not shown due to space considerations). The labeling convention within each boxplot is as follows: T denotes benign traffic, while T′ denotes obfuscated traffic (i.e., after applying adversarial perturbations to T); the subscript s denotes size and t denotes time; and the subscript o denotes the outgoing (server to client) and i denotes incoming (client to server) directions. Red horizontal line denotes the median. (a), (c), and (e) show the packet size distributions from the MAIL, P2P, and BULK classes, respectively. (b), (d), and (f) show the packet inter-arrival time distributions from the MAIL, P2P, and BULK classes, respectively. Note that T and T′ are indistinguishable in terms of packet size distributions; inter-arrival times, however, are perturbed within approximately one order of magnitude.

different results (e.g., some methods perturb the packet size distribution more). We chose the attack method in this paper because it seemed to introduce the minimum norm perturbations, but our search was not exhaustive. One can ask whether it is possible to do even "better", i.e. achieve a more minimalistic obfuscation than that presented here. This may be possible as there are regularly new adversarial generation methods being proposed.

We have not explicitly studied whether the suggested obfuscation strategies are always practically achievable. For example, Figure 2(a) suggests that obfuscation in the outgoing direction requires increasing the packet size slightly; this is easily done. However, Figure 2(b) shows that the outgoing and incoming packet interarrival time need to be increased by about an order of magnitude. Whether or not this is practically achievable is a separate question we leave for future work. Future work should also consider trading off D's costs of doing obfuscation with the increase in privacy obtained.

VI. Conclusion

The existence of adversarial examples in machine learning is largely considered a major weakness. In this paper, however, we have demonstrated that adversarial examples are a boon for network obfuscation. So long as there remains no definitive solution to the problem of adversarial examples in machine learning, our proposed approach here remains a viable obfuscation mechanism. It is of significant interest to study our approach in the context of wireless networks in tactical scenarios. There are several critical differences between wired networks (from which the dataset drawn in this paper was taken) and wireless networks due to the lossy wireless channel. For example, previous work has shown that the first few packets of a flow are far more informative than remaining ones [25] in terms of discriminative power. However, an adversary may not always have access to the first packets. Indeed, adversaries in wireless systems may have to deal with packet loss and jitter which impacts their packet size and time distributions. It is therefore of great interest to understand the extent to which the wireless medium itself can act as an obfuscator. Another area of interest is to study the impact of so-called targeted obfuscations; obfuscations in which D's traffic belongs to class C_1 but which D desires the adversary infer belongs to C_2 instead (as opposed to C_3, for example). (In this paper, we were content to settle for an adversarial perturbation which altered the true classifier output, regardless of the target class.) If C_2 is viewed as a benign traffic class while C_1 and C_3 are not, then D may prefer a targeted obfuscation towards C_2 so as to minimize further scrutiny of the network by A. There are adversarial example generation mechanisms in the literature which achieve targeted perturbations; it would be of interest to explore these further.

References

[1] M. Cotton, L. Eggert, J. Touch, M. Westerlund, and S. Cheshire, "Internet assigned numbers authority (iana) procedures for the management of the service name and transport protocol port number registry," Tech. Rep., 2011.

[2] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005, pp. 41–54.

[3] "Opendpi," https://github.com/thomasbhatia/OpenDPI, accessed: 2018-04-15.

[4] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[5] H. Mohajeri 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*. ACM, 2012, pp. 97–108.

[6] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.

[7] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," in *International Symposium on Experimental Algorithms*. Springer, 2010, pp. 373–385.

[8] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 135–148.

[9] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go-mixes providing probabilistic anonymity in an open system," in *International Workshop on Information Hiding*. Springer, 1998, pp. 83–98.

[10] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao, "Preventing traffic analysis for real-time communication networks," in *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, vol. 1. IEEE, 1999, pp. 744–750.

[11] "Gnutls," https://www.gnutls.org/, accessed: 2018-08-20.

[12] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis." in *NDSS*, vol. 9. Citeseer, 2009.

[13] A. Iacovazzi and A. Baiocchi, "Optimum packet length masking," in *Teletraffic Congress (ITC), International*. IEEE, 2010, pp. 1–8.

[14] E. N. Ciftcioglu, R. L. Hardy, K. S. Chan, L. M. Scott, D. F. M. Oliveira, and G. Verma, "Chaff allocation and performance for network traffic obfuscation," in *IEEE ICDCS*. IEEE, 2018.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[16] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," *arXiv preprint arXiv:1706.04701*, 2017.

[17] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[20] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[21] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 50–60.

[22] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Tech. Rep., 2013.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] N. Papernot, N. Carlini, I. Goodfellow, and et Al, "cleverhans v2.0.0: an adversarial machine learning library," *arXiv preprint arXiv:1610.00768*, 2017.

[25] Y. Lim, H. Kim, J. Jeong, C. Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 9.