DNS Exfiltration Guided by Generative Adversarial Networks

Abdulrahman Fahim*, Shitong Zhu*, Zhiyun Qian*, Chengyu Song*, Evangelos Papalexakis*,
Supriyo Chakraborty†, Kevin Chan‡, Paul Yu‡, Trent Jaeger*, and Srikanth V. Krishnamurthy*
* *University of California Riverside*
† *AI Foundations, Capital One*
‡ *Army Research Lab*

Abstract—**Today, DNS exfiltration attacks are detected by checking for anomalies present in the traffic, such as unusually high transmission rates to a single domain and/or DNS query patterns that are very different from those in benign queries. While such approaches are seemingly robust, we show in this paper that our carefully designed and novel DNS exfiltration attack, DOLOS, that uses a generative adversarial network (GAN), can guide the encoding of sensitive data in a manner that both evades these detectors and significantly speeds up the exfiltration rate compared to prior methods. At its core, DOLOS divides the exfiltration data into smaller chunks, and projects each chunk into a representation that is very similar to benign queries. In addition, DOLOS adaptively tunes its exfiltration rate to conform with benign DNS traffic from the compromised host, and introduces proper levels of spurious traffic to reduce entropy. Importantly, DOLOS evades machine learning (ML) based detectors with no prior knowledge of their architectures or training sets (i.e., it is a blackbox exfiltration). We perform extensive evaluations using multiple datasets and also have a real implementation of DOLOS. Our evaluations show that DOLOS has a 12% detection probability even if 6 out of the 9 state-of-the-art defenses that we consider, are jointly used to detect exfiltration; if any of today's baseline exfiltration techniques try to achieve the same rate as DOLOS in this setting, they are almost surely detected. If we reduce the rates of the baselines to achieve even a low albeit slightly higher detection probability than DOLOS (0.15), we see that they take 25× longer to achieve the exfiltration. With the other three defenses, we find that baselines are almost surely detected while DOLOS remains relatively unaffected regardless of the rate of exfiltration.**

1. Introduction

Attempts to steal sensitive information of interest (e.g., credit card details) from compromised hosts is an ongoing goal of attackers [44], [83]. One technique for stealing sensitive information is DNS exfiltration [7], wherein adversaries hide and thereby exfiltrate data in DNS queries. Untill a decade ago, DNS exfiltration was not seen as a major threat and thus, enterprises had overlooked inspecting DNS traffic in their intrusion detection systems and firewalls [38]. This seemingly has resulted in an increase in DNS exfiltration incidents [23] and thus, in stolen sensitive data from private networks [44]. In light of this, many enterprises have begun to monitor DNS traffic and have deployed many recent DNS exfiltration defenses [3], [64], [74], bringing about the belief that DNS exfiltration has been effectively curbed. Such defenses mainly rely on recognizing distinctive patterns in existing/previoius exfiltration traffic compared to benign DNS queries (e.g., entropy of the query, number of capital letters) [3]. In addition, exfiltration detectors monitor traffic to unexpected domains and (1) measure the volume of DNS traffic or/and (2) apply sophisticated information-theoretical approaches to estimate the amount of exfiltrated data that might be potentially embedded in the observed stream [73], [74]. Thus, in the absence of a careful tuning of the exfiltration rate or when simplistic encoding schemes (e.g., Iodine [15]) are used to represent the exfiltration data, these detectors can easily catch exfiltration attempts. *In this work we ask: is DNS exfiltration viable in spite of these defenses?*

Today's exfiltration methods. Existing DNS exfiltration attacks leverage general-purpose encoders (e.g., Base-32/Base-64) to create DNS queries from sensitive data. These methods (agnostic to the type of exfiltration data) transform any arbitrary input data into a specific representation space to comply with DNS rules (e.g., the limited character set allowed in DNS queries) [63]. While they have shown success in the past [44], recently proposed machine learning (ML)-based defenses can differentiate these exfiltration attacks from benign queries with high accuracy [3], [64].

Challenges in the presence of today's defenses. Even if an attacker manages to compromise a host (e.g., in an enterprise via a phishing attack), accomplishing a successful DNS exfiltration attack is not easy. First, an attacker has no knowledge of the defenses deployed by the victim; DNS exfiltration detectors can range from signature-based scanners to much more sophisticated ML-based detectors [3], [11]. Second, the encoding of the exfiltration data must allow exfiltration to occur at reasonably high rates to exploit the data in a timely way. To do so, the encoding must be compact. Beyond this, since detectors often consider host-specific volumes to detect anomalies, the attacker's malware must determine the proper exfiltration rate that is as high as possible and yet evades detection, with low runtime complexity.

Our approach. In this paper, we design DOLOS (named after the Greek spirit of trickery), a stealthy and efficient black-box DNS exfiltration attack. At its core, DOLOS has an encoding-decoding framework, which is built atop a generative adversarial network (GAN). In brief, by iteratively trying to fool a discriminator neural network (that continuously learns to distinguish between benign and fake queries), the generator learns to map exfiltrated data to a latent space representation which is almost indistinguishable from that of benign DNS queries (and hence, can elude strong state-of-the-art detectors). Because the discriminator is arguably the best detector, refining queries towards evading the discriminator makes the generated encoding extremely effective in blackbox settings (can fool several of today's ML based detectors). Note that formally, a latent space is defined as an abstract, possibly multi-dimensional space that encodes a meaningful internal representation of externally observed inputs. To aid fast exfiltration, the mapping (encoding) is kept as compact as possible, while ensuring that it is decodable with high accuracy at the attacker's external site. Although DOLOS's training uses benign traffic different from that at a compromised host, it learns the intrinsic patterns of benign DNS queries; thus, its outputs online are very similar to such queries even when it is applied to previously unseen exfiltration data. Note that training a deep-learning-based generator on the host itself encumbers high computation cost and requires a lot of training data which is hard to obtain online in a timely way. DOLOS circumvents this issue by training its models offline and porting them onto the victim (this approach is very effective as shown later).

We account for multiple practical constraints, such as composing the exfiltrated data into small chunks that adhere to the specifications of DNS queries [63]. DOLOS also includes a novel rate-tuning module that adjusts the exfiltration rate, guarantees decodability at the remote the site (the encoding itself only provides decodability with high accuracy but no guarantees on its own) and injects appropriate spurious queries based on observed benign traffic from the victim; this prevents the attack from being detected and maximizes the exfiltration efficiency to the extent possible. Put together, DOLOS achieves stealthy, efficient, reliable and stable DNS exfiltration in the wild.

Contributions. A summary of our contributions are:

- We design and prototype a novel generative encoding-decoding framework for stealthy encoding of arbitrary data, efficiently into DNS queries.
- We include a novel exfiltration-rate-tuning module, that includes online mechanisms to ensure proper spurious query injection and reliability in data extraction in conjunction with the above framework, to design DOLOS, a stealthy and efficient DNS exfiltration tool for secretly collecting data from compromised hosts evading several of today's defenses.
- We evaluate DOLOS (with datasets and to a limited extent with a prototype implementation) against 9 state-of-the-art defenses [3], [9], [11], [34], [39], [53], [64], [73], [74] and compare its performance with traditional exfiltration attack baselines. We find that DOLOS experiences a 12 % detection rate even if 6 of the 9 defenses we consider are jointly used; if the baselines try to achieve the same rate of exfiltration as DOLOS, they are almost surely detected by at least one of the defenses. If their rates are reduced to achieve a 0.15 detection probability (still slightly higher than that with DOLOS), we see that they are 25 × slower than DOLOS. With the other three defenses that require to be trained with malicious examples of the attack method, the baselines are almost always detected while DOLOS is almost never detected regardless of the rate of exfiltration.

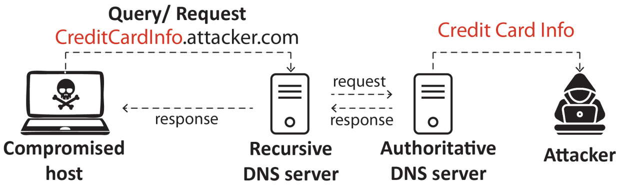

Figure 1: An example of DNS exfiltration. An attacker embeds credit card information (in red) in a DNS query destined for its remote domain, "attacker.com". The query is routed to "attacker.com" to resolve the IP address of CreditCardInfo, which enables the attacker to acquire the information.

2. Background and Threat model

2.1. Background

Malware on a compromised host can exfiltrate stolen data by embedding the same in DNS queries. Since DNS resolvers are recursive, such exfiltration queries are delivered to a primary domain of the attacker (e.g., `attacker.com`). An example of DNS exfiltration is shown in Fig. 1, where credit card information is extracted from a victim. DNS exfiltration allows opportunistically accessed data to be streamed over a long period without interruption or detection. Importantly, being a critical service, DNS cannot be completely blocked by administrators [4]. In contrast, protocols like ftp and HTTP may be blocked/restricted [13], [19]. For example, FramePos, a malware targeting networked Point of Sale (POS) machines, exfiltrated 56M credit card records over six months via DNS queries, after capturing information when cards were processed by the victim POS host [25], [44]. However, state-of-the-art detection methods are effective in detecting and thwarting such attacks [64].

Today's DNS exfiltration attacks. Next, we discuss measures attackers currently take towards successful exfiltration using DNS, while remaining stealthy.

Acquiring aged domains. If there is a large traffic volume to a *new* domain, many defenses (e.g., a popular one from Palo Alto Networks [66]) trigger an alarm suspecting that the domain was created for DNS exfiltration. To counter, attackers either purchase or compromise aged domains [72].

Choosing common DNS lookup types. DNS supports multiple lookup types [63], the most common ones being `A` and `AAAA`, to resolve IP domains. Other DNS lookup types include `TXT`, used for domain ownership verification and email spam prevention; such lookups carry larger volumes of data [63] and are uncommon. Exfiltration using these latter types is faster, but these types often trigger alerts due to their rarity [33], [64]. Thus, attackers typically use `A` and `AAAA`, which limit the rate of exfiltration, but cannot be easily detected.

Bypass caching by choosing small TTLs. Most DNS resolvers cache previously resolved DNS queries to avoid repeated resolutions. DNS responses carry Time-To-Live (TTL) values [63] that dictate how long the resolved query stays valid (in the cache). Attackers' domains typically respond with very small TTL values to force the DNS resolver to repeatedly resolve the malware's requests to increase the volume of exfiltrated data. Since benign domains also commonly use small TTL values ($\leq 60s$ as per a previous study [74]), detecting exfiltration based on small TTL values is error prone. We point out that in [3], 38% of requests in the studied dataset have TTL values of 0s (no caching). Thus, any method that relies on TTL for classification will cause high false positives. To the best of our knowledge, there is no detection method that uses TTL values to make inferences.

Managing transmission rates. Aggressive transmission of exfiltration queries (at high rates) can be detected even by defenses that simply count requests to a remote domain within short time windows [73]. Hence, attackers use grace periods (e.g., $\approx$ minutes) between queries. One detection method counts the number of cache misses to flag attacks; this implicitly limits the number of exfiltration queries that can be sent in the time window [39]. To compensate for this rate reduction, exfiltration queries can be made longer; however, there are limits on the lengths of DNS queries [63]. Moreover, other defenses can more accurately detect long queries than short ones [42], [64], [73]. *Thus a challenge in fast exfiltration is how to generate long queries without being detected.*

Encoding exfiltration data. Attackers typically encode exfiltration data for two reasons. First, encoding ensures that the generated query complies with standard DNS protocols. For instance, common DNS request types (i.e., `A` and `AAAA` [63]) only accept 64 characters as the alphabet for body text (i.e., alphanumerical letters, hyphen and dot). Second, it offers some obfuscation aiding stealth. Sending raw data, even if viable, may trigger defenses that compare embedded DNS traffic with sensitive data (i.e., potential exfiltrated data) from the compromised machine.

To the best of our knowledge, current DNS exfiltration attacks only use general-purpose data encoders (e.g., Base-32/64 and Hex) to map data into a representation space of the characters used in DNS queries [15], [25]. Such encodings however, may differ from benign DNS queries and expose the attack (discussed earlier and in § 6.2).

Defensive efforts to detect exfiltration. Previous works assume full knowledge of DNS traffic content (in plaintext) by the detector/defender [3], [64], [73], [74]. We follow the same assumption. While there is increasing encrypted DNS traffic on the public Internet [56], in enterprise environments where DNS exfiltration attacks constitute a major threat, DNS encryption is uncommon [56]; this is because network operators are motivated to monitor DNS traffic and deploy existing defenses to protect the enterprise network [2], [37] .

DNS exfiltration detection. Many legitimate domain names appear to be randomly generated (e.g., "vwdfusdgdkshjdsd.aws.amazon.com") and have become popular [64], [74]. Thus, naive defenses relying on the readability of domain names are ineffective. This has motivated smarter defenses that check either the rates at which queries are sent to individual domains or apply machine learning to determine if the features in DNS queries are suspicious. While these defenses are effective in thwarting today's exfiltration attacks, as shown in § 6.2, they are ineffective against DOLOS.

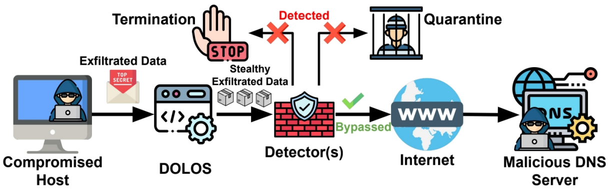

Figure 2: Threat Model: employed policies by the defense.

2.2. Threat model

Attack scenario and assumptions. In this work, we consider targeted attacks [88] where the malware acquires and exfiltrates a specific type of data (e.g., credit card numbers as in the FramePos attack [25]). We assume the attacker has already controlled one or more victim hosts, e.g., via insiders or compromises (this is how exfiltrations happen in the real world) [47], [48]. Similar to the "solarwinds" attack [22], [83], the initial malware file is very small. Subsequently, the DOLOS malware downloads the ML model and necessary files (each of small size) that are used later for exfiltration. Downloading a set of small files to avoid easy detection is commonly used by many advanced malware [22], [83] (users can easily notice large unexplainable files). The malware is assumed to acquire the data either from that machine or from the private network to which the machine is connected (e.g., accessing sensitive infrastructure logs in the private network). The malware can spread to multiple hosts in an enterprise network and all infected hosts engage in exfiltration; this was seen in previous DNS exfiltration attacks (e.g., [48], where roughly 6K devices belonging to the same company were infected).

Similar to FramePos [25], data is assumed to be acquired opportunistically, and the attacker seeks to exfiltrate the data as soon as viable (i.e., timeliness is considered critical for effective use of the data) while evading detection. Fast exfiltration allows quick remuneration. In other words, we assume that the goal of the attacker is fast but stealthy exfiltration.

Defender. Even though the attacker has infiltrated the network, it does not mean that it can exfiltrate the data undetected, as many industrial [4], [37], [77] and research solutions (e.g., [42], [64]) are targeted to stop exfiltration[1]. In practice, such exfiltration detection mechanisms are unknown to the attacker. Upon detecting suspicious primary domains or queries, operators can choose one of two strategies to handle them (shown in Fig. 2): (1) *quarantine*, which pauses traffic to the suspected domain for a preset period. This strategy suits scenarios that expect higher positive rates from detectors, since it is impractical to manually inspect and verify all suspicious traffic; (2) *termination*, which completely disallows ongoing and future DNS queries to the suspected domain.

Attacker. The malware seeks to steal sensitive data via DNS exfiltration, bypassing an unknown defense using DOLOS. Exfiltration can take place to a single or multiple domains, the later acheiving the full potency of the attack. If the defense uses a *quarantine strategy*, and this is known

[1] Note that exfiltration detection is deployed to catch outbound traffic instead of inbound i.e., our downloaded ML models can still be obtained, hidden as benign HTTP traffic (e.g., with a Trojan Downloader [67]).

to DOLOS, it can probe and estimate the best transmission rate that can maximize exfiltration efficiency while avoiding quarantine. Otherwise, DOLOS observes benign traffic on the compromised host, using a sniffer tool (e.g., [80]) to capture the rate of benign DNS requests; DOLOS then tunes the exfiltration rate to be consistent with this rate to avoid detection. We assume that the attacker can attain high privileges on the hosts and mimic benign DNS traffic rates. This is possible via local privilege escalation exploits, which are common [10], [30], [89].

DOLOS is trained with samples of exfiltration data offline before infecting the victim. These samples are assumed to be similar to data exfiltrated online. Such samples, for example, for credit card records or computer logs, have well-known formats and can be obtained/synthesized. Similarly, a model trained with an English text dataset can be used for e-mails or other text data, or a model trained with specifiic classes of images (e.g., medical images) can exfiltrate similar images in the wild.

We assume that the attacker has purchased/compromised old domain(s), and uses common DNS query types. Thus, defenses cannot use these to discern exfiltration traffic and must detect the attack based on its encoding and rate only.

3. System Overview

We design DOLOS to generate embedded DNS queries akin benign traffic; in addition, DOLOS includes mechanisms that boost exfiltration rate, while ensuring that the *aggregation* of exfiltration queries remains undetected.

DOLOS is based on an efficient encoding method, customized to the data of interest (e.g., credit card records or emails). While prior encoding methods (e.g., Base-64) are generic (no prior knowledge of data is necessary), we argue that using customized encoders for different data types trades off generality (see §7) for stealth and speed.

An overview of DOLOS's encoder-decoder framework. DOLOS's encoder and decoder are trained offline with benign DNS and exfiltration datasets. The encoding ensures that the exfiltrated data representation has high similarity to benign data. It is relatively straightforward to categorize the broad type of networks where exfiltration occurs, e.g., enterprises (Windows environments, user-facing applications) and data centers (Linux environments, server applications). We can then feed the corresponding types of benign DNS datasets in the offline training phase. We leave the possibility of leveraging a victim's DNS traffic as 'supplemental online training data' as future work. Note that the full training cannot be done on the victim host since it may require a long time, large amounts of training data, and high computational power.

After training, both the encoder and decoder are integrated with the malware which infects the compromised host (reasons for including the decoder are discussed below). The decoder is also used at the attacker's remote site, to which the encoded data is exfiltrated.

An overview of DOLOS's online functions. At this point, assume that the malware (equipped with the trained encoder) infects a victim host. Blindly performing exfiltration can still expose the attack because the volume of the aggregated exfiltration queries may not conform with benign volumes generated by the victim. Thus, DOLOS's malware includes a module to sniff the host's benign traffic and tune the exfiltration rate and inject some necessary spurious requests (that are also seen in benign DNS streams), accordingly. A bank of spurious queries is generated offline (consistent with benign traffic) and is shipped with the DOLOS malware, and used during exfiltration. We choose this offline approach since the malware does not have a method to craft spurious queries that are stealthy online; thus online generation may result in anomalies that trigger the detector. In addition, it helps that these offline generated spurious queries can be easily compared with the bank at the external site and discarded. During online operation, the host chooses those queries from the bank that are similar to the exfiltration queries (details in § 5).

Finally, note that the encoding generated by DOLOS is lossy (although we ensure that the loss rates are very small during training). To fix this issue, DOLOS validates the decodability of each exfiltration query with the decoder shipped with the malware. If it is decodable, it is sent as is. If not, DOLOS uses an error recovery module (using a traditional lossless compression method in an exterme case) to ensure its decodability. Upon the receipt of a chunk, the remote site uses a simple method (discussed in § 3) to apply the proper decoding and recover data. Since such cases are rare, DOLOS is still able to evade all considered defenses with very high probability.

4. GAN based encoder-decoder design

Next, we describe DOLOS's encoder/decoder, trained offline.

4.1. Properties of DOLOS's encoder/decoder

In this section, we describe the set of desirable properties that guide the design of DOLOS's encoder-decoder framework.

Stealthy encoding. Traditional encoding (e.g., Base-64) does not account for stealth, and thus, a steady stream of such outputs are easily detectable by current detectors. To achieve stealth, we need to coerce the encoded exfiltration traffic to resemble benign DNS traffic. While this is challenging, we identify an opportunity to use Generative Adversarial Networks or GANs (details on GANs in [27]) in a novel way towards overcoming it. GANs have been shown to generate examples that mimic a given distribution (e.g., images resembling real humans). However, they have not been previously used to morph DNS exfiltration data. Our key idea is to train a generator to encode exfiltration traffic with the aid of an evolving discriminator (trained with benign traffic) that disambiguates such traffic from benign DNS traffic. A well-trained generator then becomes an effective encoder that can transform the exfiltration data into a representation akin to benign DNS traffic. While similar training of a GAN for a single objective (not in the DNS context) has been done in other prior efforts, unfortunately, by itself, this does not suffice. One must also ensure high decoding accuracy at the external site, which is critical for successful exfiltration. Note that fulfilling multiple objectives using the same GAN have been explored to a limited extent in the ML community [5], [12], [90].

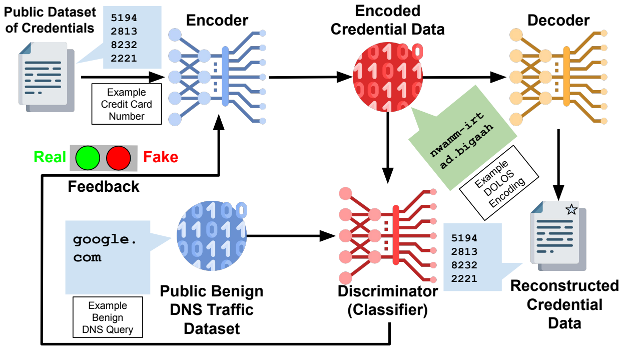

Figure 3: Offline training phase of DOLOS. The data from the encoder is constrained to fool a discriminator, and must be decoded by the decoder with high accuracy.

However, to the best of our knowledge, the first work to apply this approach to realize a DNS exfiltration attack.

Decoding accuracy. To ensure that the generated codes can be correctly decoded with very high probability, in addition to accomplishing stealth, DOLOS includes a *second* discriminator (we abuse the term here) that is, in effect, an evolving decoder. This decoder is trained jointly with the generator and imposes a second objective to be fulfilled by the latter. Specifically, the encoded representation (a) must deceive the first discriminator and (b) must be translatable to its original form by the decoder. To reiterate, to the best of our knowledge, prior GAN efforts do not consider multiple, different objectives during training.

Code compactness. An encoding that is both stealthy and decodable with high probability, could entail high overhead (lower encoding efficiency). Minimizing this overhead is key for efficient exfiltration. Towards this, we model the problem of finding the most compact encoding as a search problem[2]. Specifically, we begin by considering different levels of compactness (corresponds to different encoding overheads). We use a greedy approach where we try the considered compactness levels in an ascending order (most compact to least). For each, we try to generate an encoding (satisfying stealth and decoding constraints) within a predetermined time period. If unsuccessful, we move to the next. The approach iteratively continues until an encoding is found. More details are provided in § 4.4.

Blackbox exfiltration. The discriminator is arguably the best anomaly detector since it learns to discern exfiltration queries as they are iteratively refined to be similar to benign DNS traffic. Thus, if the generation process goes through several rounds of interaction with the discriminator, the encoding is likely to be sufficiently tuned to be similar to benign traffic and can evade blackbox anomaly detectors (as shown in §6).

4.2. Encoder and Decoder design

Before delving into the details of our design, we define some notation used in what follows (summarized in Table 1). We define random variables that represent the benign traffic and exfiltration data as x and z, respectively. These random variables will have their own distributions in terms of characters in the query, the correlations across the characters, etc. The offline phase relates to jointly training three neural network blocks, viz., an encoder (`Enc`), a decoder (`Dec`) and a discriminator (`Dis`). The parameters of these neural networks (weights) are denoted as θ_{Enc}, θ_{Dec} and θ_{Dis}, respectively. The data to be exfiltrated is divided up into chunks, and each chunk is to be encoded and confined to one *fake* DNS query. We denote a set of chunks as $\boldsymbol{M}$, and each chunk is represented by $m_i \in \boldsymbol{M}$. The encoder, thus, takes a chunk of the exfiltration data of size L^{m_i}, consisting of a sequence of characters $\mathbf{c} = (c^1, c^2 ... c^j ... c^{L^{m_i}})$, and tries to map that on to a codeword $\mathbf{y} = (y^1, y^2 ... y^{L^{E_i}})$ of length L^{E_i}. Note that L^{m_i} may not be equal to L^{E_i}. The mapping function of the encoder is represented by $Enc(c) = f_{\theta_{Enc}}(c)$.

TABLE 1: Key notation

Notation	Description
Enc, Dec, Dis	The encoder, decoder and discriminator neural networks, respectively
θ_{Enc}, θ_{Dec}, θ_{Dis}	The parameters of the Enc, Dec and Dis, respectively
L^{m_i}	The length of the exfiltration chunk data
L^{E_i}	The length of the encoding of an exfiltration chunk of data
γ	Ratio of the encoding length of a chunk to the length of an exfiltration chunk
V_D	Validation dataset
$accD$	Validation decoding accuracy
H	Maximum # of batches used for training
B	# of samples in a training batch
α	A weighting hyperparameter to balance the updates from the decoder and discriminator networks.

The decoder takes a codeword ($\mathbf{y}$) from the encoder as its input and estimates the original (raw) exfiltrated chunk as a sequence of characters viz., $\hat{\mathbf{c}} = (\hat{c^1}, ... \hat{c^j} .. \hat{c^{L^{m_i}}})$. Given the input $\mathbf{y}$, the decoder function $Dec(\mathbf{y})$ represents the probability that the output $\hat{\mathbf{c}} = \mathbf{c}$, and is denoted by $Dec(\mathbf{y}) = f_{\theta_{Dec}}(\mathbf{y})$.

The discriminator learns how to differentiate between a benign DNS query and a codeword generated by the encoder. Specifically, the discriminator function, $Dis(s) = f_{\theta_{Dis}}(s)$ yields the probability that the given input $\mathbf{s}$, belongs to the distribution of the benign samples. The offline training is depicted in Fig. 3.

Stealth. Since the discriminator seeks to differentiate between benign and fake exfiltration queries, it tries to minimize the cross entropy loss between the input and the correct output (which is known as ground truth during training). Let us denote the probability of the discriminator's prediction on the generated queries (fake) and the benign queries as $Dis(Enc(z))$ and $Dis(x)$, respectively; here, z is the exfiltration data fed to the encoder, and x is a benign DNS query. To minimize the cross-entropy as alluded to above, the discriminator will seek to minimize the loss function: $\min[-log(Dis(x)) - log(1 - Dis(Enc(z)))]$. This, in turn, is equivalent to $\max[log(Dis(x)) + log(1 - Dis(Enc(z)))]$.

At the same time, the encoder seeks to fool the discriminator by minimizing the discriminator's confidence (probability) with regards to labeling the generated fake queries. In other words, it wants to minimize $log(1 - Dis(Enc(z)))$.

Given the conflicting objectives of the discriminator and the encoder, we can model their interactions as an iterative minimax game with the following loss function

[2] We tried to include a compactness constraint directly in the encoder-decoder formulation, but it increased the time complexity significantly.

($\mathbb{E}_x$ and $\mathbb{E}_z$ are the expectations over benign and exfiltration data):

$$\min_{\theta_{Enc}} \max_{\theta_{Dis}} \quad \mathbb{E}_x[log(Dis(x))] + \mathbb{E}_z[log(1 - Dis(Enc(z))]. \tag{1}$$

Decodability. To ensure the decodability of the generated codes, we jointly train a decoder. Here, both the encoder and the decoder seek to maximize the probability of correctly predicting the original characters from the latent space encodings. This translates to a minimization of the average cross entropy between the inputs and the ground truth labels. This cross entropy loss minimization is given by:

$$\min_{\theta_{Enc},\theta_{Dec}} \quad \mathbb{E}_z[-log(Dec(Enc(z)))]. \tag{2}$$

4.3. Practicalities

Neural network architecture. We need a neural network architecture that captures semantic relationships as well as short- and long-term dependencies across the characters in a benign DNS query. If the learnt embeddings reproduce these properties, they can better mimic those queries. There exist many neural network architectures that satisfy the above properties, especially in the NLP space, where capturing semantic relationships is critical. Among those, we choose transformers [86] as our choice since a transformer allows for parallel computations of sequential data, which makes the training fast. One nuance is that, typically, transformers take words as inputs; since we want our approach to work with different types of input data (e.g., credit card numbers, text data), we choose our inputs to be characters instead of words. Note that as discussed in detail later, even more complex data forms (e.g., images, which we consider in this work) can be represented using this method (e.g., with an image, each byte representing pixel intensity can be considered as a character and fed to the model).

Representation of the latent space. DNS queries `A` and `AAAA` permit only 64 characters. Thus, the encoder's output (i.e., $y^1, y^2..y^{L^{E_i}}$) is a sequence of discrete characters from these.

Non-differentiable discrete latent space. Our inputs are discrete characters, and so are our latent space encodings. Back-propagation, used to tune the neural network weights, cannot be directly applied to discrete variable representations that are non-differentiable (i.e., they have zero gradients everywhere) [41]. To overcome this, we use a popular solution for discrete representations, viz., the softmax-Gumbel approximation [41]. The idea is to use discrete variables in the forward pass, but use continuous approximations (i.e, softmax) in the backward pass.

4.4. Training algorithm

We train DOLOS to optimize the objectives in Eqns. (1) and (2) using an iterative algorithm. Iterative methods are often used in GANs [27]; however, as discussed, the novel aspect of our work is that we also seek very high likelihood of decodability and compaction.

Towards iteratively optimizing the objectives in equations (1) and (2), we update the weights of the neural networks after each batch of inputs, until we generate stealthy *and* decodable, fake DNS queries. Specifically, we *first* sample a batch from benign DNS traffic and a batch from the output of `Enc` to update the weights of `Dis`. In the *second* iterative step, the same batch from the `Enc` is fed to both the `Dec` and `Dis`, and feedbacks from both are used to update the weights of `Enc`. Since the updates from both networks may vary in magnitude and effect, the encoder may be forced to favor one objective over the other. We use and tune a hyper-parameter α to balance the two objectives. In the *third* step, we update the `Dec` weights to enhance the decoding accuracy. The three steps are repeated until DOLOS is able to successfully bypass a validation step (discussed below in what follows). The offline training of DOLOS is captured in Algorithm 1.

Algorithm 1 Training DOLOS Encoding

```
Input: exfiltration and benign DNS datasets , $acc_D$
Input: Validation dataset ($V_D$), Validation Model ($V_M$)
Input: Validation model fooling rate threshold ($\beta$)
Input: Training time out threshold ($H$)
for $\gamma$ in range (0.5,1.5,0.1) do
  Initialize $Dec$, $Dis$, $Enc$ with latent space of size $\gamma * L^{m_i}$
  while True do
    (1) Sample batches from exfiltration dataset and benign dataset of size B.
    (2) Update the Discriminator as follows: $\nabla_{\theta_{Dis}} \frac{1}{B} \sum_k^B [log(Dis(x^k)) + log(1 - Dis(Enc(z^k)))$.
    (3) Update the Encoder: $\nabla_{\theta_{Enc}} \alpha \frac{1}{B} \sum_k^B [log(1 - Dis(Enc(z^k))) + (1 - \alpha) * \frac{1}{L^{m_i}} \sum_j log(p_{\hat{cj}})$.
    (4) Update Decoder with $\nabla_{\theta_{Dec}} \frac{1}{L^{m_i}} \sum_j log(p_{\hat{cj}})$.
    if $\frac{1}{|V_D|} \sum_{v \in V_D} \mathbf{1}( \; argmax \quad Dec(Enc(v)) = v \; ) \geq acc_D$ & $V_M(Enc(V_D)) \geq \beta$ then
      return trained DOLOS
    end if
    if $\#$ of batches $\geq H$ then
      break {Need a bigger encoding size}
    end if
  end while
end for
```

Compactness. As discussed in § 4.2, we seek compaction to increase the exfiltration rate. For a given length of a raw chunk, the generator is constrained to output a fixed (to be determined) length encoded query (regardless of the semantic content of the raw chunk). We seek to find a value of $\gamma = \frac{L^{E_i}}{L^{m_i}}$, that allows us to map a raw chunk of length L^{m_i} to the shortest possible encoding length L^{E_i} output by the generator. This would then maximize the efficiency of the encoding (highest amount of information encoded into the smallest number of characters in the latent representation). In other words, we search for the smallest value of γ, such that the encoded query is decodable, and preserves stealth. Specifically, any γ smaller would violate either stealth or decodability or both. For simplicity, we confine the search space of γ between 0.5 and 1.5 with step 0.1. We begin with the smallest γ (which yields the most compaction), and if the model does not meet the the criteria used to stop training (discussed next), we re-initialize the models and train them with the next larger γ value.

Once γ is thus determined, if we know what is the maximum permissible encoding length L^{E_i} (the maximum length of DNS queries sent by the victim host), we can compute the corresponding raw chunk length that can be used as $L^{M_i} = \frac{L^{E_i}}{\gamma}$. We then collect tokens to fill an m_i smaller than this length and generate the encoding during online operations as discussed in § 5.

Validation. Since it is very hard to fool the evolving discriminator (as it continuously learns), we use a validation process to determine when to stop training. After every N batches (1000 in our evaluations), we first test the decodability of the generated codes using the trained decoder to ensure it meets the decoding accuracy constraints. Subsequently, we test the stealthiness of the generated codes against an anomaly detector (not the `Discriminator`) just trained on benign DNS queries. If we fool this detector with a very high probability ($>$ 99 %), we assume that the GAN has been sufficiently trained. We note that the anomaly detector is different from the defenses we test DOLOS against, and thus it does not violate the blackbox assumption. Further implementation details are in Appendix A.

4.5. Composing spurious queries

As discussed in §3, we form a bank of spurious queries offline by sampling the generated traffic from batches in a validation dataset and identifing the most frequent 3-4 characters. We randomly combine these along with natural separators present in DNS queries, viz., 'hyphen' and 'dot', to form spurious queries. We refine these with our discriminator until validation.

5. Tuning the exfiltration online

Next, we describe DOLOS's operations on an infected host. We reiterate that DOLOS's encoder is unaware of the defense, or the policies employed upon flagging a domain as an attack site.

Most of today's defenses make an inference on queries sent to each primary domain (i.e., decide if that domain is an exfiltration site or not) [64], [73], [74]. Such inferences are based on the volume, the rate, the repetition of queries and the entropy associated with the aggregation of queries to that domain. To evade detection, DOLOS must tune these parameters for each domain to which it exfiltrates data (can do so independently), towards achieving evasion with respect to those domains.

The best rate for stealthy exfiltration. DOLOS observes benign traffic over an empircally chosen time window (few hours) to estimate the exfiltration rate. In particular, DOLOS needs to choose the number of requests (N), and the average query size, (L), in each time window. The bigger these values, the more data can be exfiltrated, but if they are too large, detection is very likely. A naive approach is to observe the number of requests and the average length of requests to each domain, and from among these, choose the N and L that would maximize the exfiltration rate (i.e., $N * L$ per time window). However, as discussed benign traffic consists of many repetitions of queries (either partial overlaps or full repetitions). To be consistent with benign queries, the attacker has to transmit unique exfiltration requests *and* repeated requests, and requires an estimation of the rate of requests in each category. Note that determining exactly how many times each query is repeated is not necessary as this value differs across different primary domains and we have not seen it being used in practical defenses. In other words, the percentage of unique and repeated queries transmitted to each domain should be consistent with the repetition rate seen in benign queries sent by the victim host.

Beyond repetitions, many requests are partially similar in benign DNS traffic. Not accounting for partial similarity may expose the attack [74]. To illustrate, the following two unique requests are considered partially similar: `gllto1.glpals.com` and, `gllto2.glpals.com`. To evade the detector, DOLOS includes both repetitive and partially overlapping spurious queries consistent with benign traffic; these are later ignored after exfiltration.

Key idea. To estimate the volume of "unique" or dissimilar queries for each primary domain (obtained from the benign traffic on the compromised host), we cluster the associated queries; those belonging to the same cluster can be deemed similar or repetitions. From these, DOLOS identifies the domain for which the combination of average query length and number of unique queries yields the highest exfiltration rate, and uses these values in tuning its exfiltration process.

Clustering algorithm. Existing clustering methods, including even the simplest of them (i.e., k-means clustering [55]) are expensive. This is because k-means requires multiple iterations of comparisons among the data to converge, and a large space complexity to store all the queries from the host. Importantly, the proper "k" is not known a priori. Because of this, we design a simple algorithm for DOLOS. In brief, for each primary domain, the algorithm processes the streamed queries. With each query, it measures the similarity between the query and the *cluster representatives* of previously formed clusters; if the query is not similar to any representative, a new cluster is created with that query chosen as its representative.

To assess the similarity between two DNS queries, we use the following approach. For each pair of queries, we measure the Jaccard similarity [40] by computing intersection between the characters of the query relative to the length. If this value is greater than a pre-selected threshold, we consider the queries to be partially similar. To select the appropriate threshold, we conduct offline analysis using samples of benign traffic and find that a threshold value ranging from 0.7 to 0.8 effectively groups similar queries.

The algorithm has a $O(n * d)$ run time complexity where n and d refer to the number of DNS queries sent by the host to a primary domain and the number of clusters per primary domain to which queries are sent, within the time window of interest. This process is captured in Algorithm 2.

Online exfiltration. DOLOS's online workflow is shown in Fig. 4. DOLOS computes the number of unique queries that it can transmit in a time window, as well as the number of spurious queries it must insert, based on the clustering it has constructed[3]. DOLOS computes the most frequent characters used in the encoded exfiltration queries; it then chooses the spurious queries that are closest to the cluster members (in terms of Hamming distance) from the pre-stored bank (recall § 4.5). Subsequently, for the given exfiltrated data, DOLOS encodes chunks of the proper size (it computes the size based on the learned

[3] The number of spurious queries is the difference between the total number of queries and the number of unique queries that are determined by our clustering (for each domain).

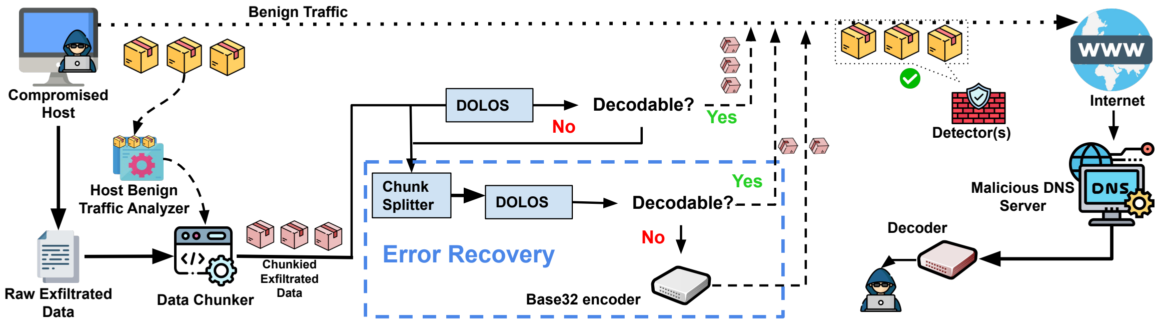

Figure 4: Online attack phase of DOLOS. DOLOS sniffs benign traffic to tune the exfiltration rate in terms of number of queries transmittable in a time window and chunk length. Next, it divides the data into chunks and encodes them as DNS queries. If the encoded query is decodable, it is sent as is; else the error recovery module is used.

value of γ as discussed in § 4.4) to form fake queries using the pre-trained encoder. The exfiltration (fake) and spurious DNS queries are sent in that time window.

Algorithm 2 DOLOS clustering methodology

```
Input : list requests to a domain (Queries), threshold
Initialize clusters set ($ClusSet$)
for newQ in Queries do
    for q in $ClusSet$ do
        if JaccardSimilarity(newQ,q) $\geq$ threshold then
            Break {Found a match}
        end if
    end for
    if no match then
        $ClusSet \leftarrow$ newQ {Create new cluster with query newQ}
    end if
end for
```

Decodability assurance: Since the latent representations generated by DOLOS are inherently lossy in nature, a small subset of chunks may not be decodable (details in § 6.2), despite considering decodability during training. To guarantee decodability, DOLOS checks if the encoded chunk is decodable using the downloaded decoder (a replica of that used at the remote site). If the chunk is not decodable, DOLOS uses an error recovery method to guarantee decodability. Our module for error recovery attempts to use DOLOS's encoder but with shorter chunk sizes, and if it fails, DOLOS replaces the encoded chunk with a new fake query using Base-32 encoding (which is lossless). We use Base-32 because it performs better as compared to other traditional encoding methods in terms of stealth (e.g., Base-64). As shown in § 6.2, the fraction of chunks needing error recovery is very small and the effect on DOLOS's stealth and speed is negligible.

The error recovery model shown in Fig. 4 works as follows. Upon finding a chunk with decoding errors, DOLOS's online module shrinks the input chunk size by a factor β (set to 0.8 empirically in our experiments) and attempts to encode the new chunk with DOLOS, again. If this fails, the step is repeated a second time. If the two attempts still cause a decoding failure, DOLOS uses Base-32 for encoding. While such queries can be flagged by defenses, because they are rare, the domain to which data is being exfiltrated are not deemed malicious by defenses (they need to observe a sustained stream of such queries to do so) as seen in § 6.2.

Decodability at the remote site: Upon receiving a DNS query, the remote site has to determine whether the received query is encoded using Base-32 or DOLOS's encoder. We use a simple solution for this task. The receiver attempts to decode the query using a Base-32 decoder; if the query is not in the correct format, the Base-32 decoder issues an exception. In that case, the receiver infers that the received query is encoded using DOLOS's encoder. Otherwise (no flag), it deems that the query is encoded using the Base-32 encoder. While it is possible for the Base-32 decoder to decode DOLOS encoded queries in extremely rare occasions, we did not observe those in our experiments. To cope with such cases, one can apply other solutions to make the decoding more robust. For example, the outputs of both decoders can be combined with previously decoded chunks to evaluate which one is more consistent with the received stream. Another possibility is to use a query classifier (similar to the work in [11]) that discriminates between Base-32 and DOLOS encoded queries.

6. Evaluations

We evaluate DOLOS with multiple types of exfiltration data and against multiple defenses, and compare it to prior encoding baselines. We first describe our setup, and then our experiments and results.

6.1. Preliminaries

First, we describe the datasets used, our implementation, the encoding baselines, and the considered defenses. The parameters used in training DOLOS are in Appendix A.

Datasets. We use multiple datasets in our evaluations.

Benign datasets: We use two datasets for training DOLOS (trained on each at a time) and two different others for training the ML based defenses. The use of different datasets for DOLOS and the defenses emulates blackbox settings. DOLOS is trained using:

- Georgia Tech DNS dataset (GT) [84]: This dataset contains DNS traces collected from suspect Windows executables in a sterile, controlled environment; thus, they often do not generate malicious queries. These exceutables often use benign DNS queries to test connectivity [21][4]. We collect PCAP files dating from 2015

[4] To verify that the dataset contains only benign traffic, we use the best defense in our study [3] (discussed later) on the Georgia Tech dataset, and find that the triggering rate is negligible.

until December 2020, and use it to train DOLOS[5]. DOLOS that is trained on this dataset is denoted as DOLOS (GT).

- ISI-reverse DNS queries (ISI-rdns) [69]: This dataset is collected by using a reverse DNS scan over the entire IPv4 space. While a subset of the IP addresses may lead to names that are not associated with real domains, we try our best to filter these out using heuristics based on our domain expertise. Domains with a fraction of numerical and capital characters, larger than 30% are removed. Moreoever, many queries (e.g., to the same primary domain) are similar, and these can create bias in training DOLOS. To remove these, we cluster queries using our method in §5 and only use diverse samples. We denote the version of DOLOS that is trained on ISI-reverse DNS queries as DOLOS (ISI-rdns).

Datasets used for training state-of-the-art defenses are:

- Thapar dataset [79]: This dataset was collected from 4,000 hosts in a university over 10 days. We extract DNS queries from successful DNS responses (e.g., DNS queries associated with NXDOMAIN responses are ignored) [59]. Since the dataset is from an operational setting, our belief is that it represents data used to train real DNS exfiltration defenses.
- ISI host Level dataset [68]: This dataset contains massive raw packets collected at a b-root DNS server with anonymized IP over two days. We group requests by the srcIP field corresponding to the recursive resolver (i.e., each of which is a host). Since this huge dataset is collected from real users, our belief is that a defense trained with dataset should be able to differentiate between benign and exfiltration queries.

Exfiltration Datasets: We consider multiple types of exfiltration datasets to evaluate DOLOS. In all cases, we separate the records into training and testing sets. At most a third of the records (selected randomly) are used for training and validation, while the rest are used in testing.

- *Text dataset*: We use the Amazon Reviews Dataset [60] to represent text data (e.g., emails or documents) that the attacker may compromise. We consider the data in terms of characters (not words as discussed in §4). This dataset contains 168 unique characters (English alphanumerical characters and special characters).
- *Credit Cards*: We mimic real credit card information to create our own synthetic dataset (2M records). Each synthesized credit card record contains a 16 digit credit card number where the digits have to pass the Luhn algorithm test [57], a method that is used to verify synthesized credit card numbers. In addition, the four digit expiry date, the three digit CVV, the first and last names, the address and billing zip codes are also generated as follows. The CVV is just three random digits and the expiry date is randomly chosen between Jan 2024 and Dec 2034. To generate names, we use the dataset in [76] which contains around two million real names. For the address and billing zip codes, we download US west, midwest, northeast and south addresses from *batch.openaddresses.io* [70].
- *Computer Logs*: Computer logs can be useful for subsequent reconnaissance attacks (e.g., [32], [54]). We use two datasets of logs. The first is a Microsoft Windows "Event Logs" dataset (from a public GitHub repository of logs collected over 226 days [1]) of size 27GB. The second is a Linux logs dataset [31], collected from `/var/log/messages` on a Linux server over 264 days. We report the results of Microsoft logs in the main paper, and those of Linux logs with examples in Appendix G.
- *Images*: We use a dataset of x-ray images (in PNG formats) that were used in COVID diagnostics [45]; such data became valuable recently for attackers [78]. We transform the image from the PNG file into a matrix of bytes (each representing pixel intensity). Note that PNG image formats offer lossless compression and the actual values of pixel intensities are retained without change. Thus, this transformation does not result in any loss of data. Further, the matrix is processed to form a sequence of characters/bytes. In other words, we flatten the matrix to a single dimension (sequence of bytes) which is then input to the encoder; the matrix can be reconstructed at the receiver. DOLOS initially transmits the metadata describing the image shape, so that the receiver can reconstruct the same.

Encoding Baselines. We compare DOLOS's encoding against three types of baselines (which we implement):

- *Iodine [15]*: Iodine, the popular DNS exfiltration tool, which compresses data and encodes it with Base-64 (denoted Iodine-64) *or* Base-32 (denoted Iodine-32). While Base-128 is available on Iodine, it does not comply with DNS types `A` and `AAAA`.
- *DNSCat (Compressed HEX) [36]*: DNSCat is a popular DNS exfiltration tool that compresses and encodes the data using Hex-encoding into strings.
- *FramePos encoding* [25]: This encoding was used in the recent attack where credit card information from POS was exfiltrated over DNS. It is essentially a variant of Hex encoding that does not compress data. After encoding the exfiltration data, each byte is XOR-ed with a pre-determined integer value; to decode, the received value is XOR-ed with the same integer at the attack site to retrieve the original value. Details are found in [25].

Defenses. We can categorize defenses into three types: (1) rule based defenses; (2) anomaly detectors (ML based); and (3) classifiers that distinguish between bengin and malicious classes using ML.

Rule based defenses impose empirically derived rules on some properties of outgoing DNS traffic. We summarize rule based defense methods below:

- *Zeek* flags a domain if (i) the length of any transmitted query or (ii) if the number of unique queries within a time window, to the domain exceed preset thresholds.
- *ZeekQ* is similar to Zeek but adds a rule to check if the percentage of numerical characters in a query exceeds a threshold [34].
- *Paxson et al.* [74] collects queries to a domain over a time window and compresses them; if the compressed volume exceeds a certain threshold, the domain is

[5] This dataset was used in [21] but was recently withdrawn. We learned from the authors of [21] that this was due to funding/maintenance issues.

flagged. While Zeek and Paxson el al., issue alerts based on the traffic volume to a domain, they fail if the attacker exfiltrates data to multiple domains with low rates.

- *Ishikura et al.* Unlike the above, [39] builds a shadow least-recently-used (LRU) DNS cache (a copy not interfering with DNS directly), which counts cache misses in a time window. If the number of shadow cache misses for a given client exceeds a threshold (i.e., the maximum number of cache misses across all clients in the prior time window), the defense flags an attack.

Anomaly detectors, listed below, learn features in benign DNS queries or the aggregation of DNS queries to a domain, in a time window. They detect deviations from benign queries and flag existing attacks even if exfiltration is at low rates.

- *Nadler et al.* [64] uses an isolation forest [52] to detect anomalous domains and is adopted by Akamai [4]. The features used are: average length of queries, number of queries, fraction of unique queries, average length of readable subdomains, the aggregated entropy of all transmitted queries and the fraction of DNS types that are `A` and `AAAA`.
- *Jawad et al.* [3] uses a set of hand-crafted features to build an isolation forest anomaly detector. The features are: query length, # of capital letters and numbers, # of subdomains, average and maximum lengths of subdomains, and the entropy of the request.

Classifiers are trained with both benign and malicious samples (assumed to be known to the defender), and perform classification at a DNS query level. Below we list such defenses.

- *Buczak et al.* [9] uses Random Forest to classify benign and malicious queries. A total of 17 features are used including query shape features such as ratio of distinct characters, maximum and average length of subdomains and percentage of numerical characters.
- *Liu et al* [53] uses Support Vector Machine (SVM) to classify benign and malicious queries. It uses the entropy of the uni-gram, bi-gram and tri-gram of characters as features.
- *Chen et al.* [11] trains an LSTM classifier with samples of benign and malicious traffic.

In all cases we follow the directions on training and tuning the defensive models from the original papers. Since the last three defensive models need adversarial samples to train, for each attack method, we provide examples generated by the same method (e.g., the classifier is given Iodine-32 examples, when it is tested against Iodine-32 encoded exfiltration). We also provide the classifier examples generated by DOLOS (for example, we feed malicious queries generated with DOLOS that is trained with GT or the ISI-rdns benign dataset, but test them with a different set of queries that are generated with either the same or the other dataset). One can think of this as providing some form of adversarial training [6] to these defenses, which make them very powerful. We hypothesize that since DOLOS generates different encodings in each training instance (they all look similar to benign but are different), it is effective even with such whitebox defenses. Some results on these are in § 6.2, while additional results and details are in Appendix F.

Attack and defense setups. We assume the worst outcome upon detection because we consider a blackbox setting, i.e., the defense blocks the primary domain that is flagged. Thus, DOLOS monitors and uses the victim host's DNS query patterns to tune the online rates of fake and spurious queries. Some of the defenses we consider seek to detect individual anomalous queries (i.e., ZeekQ and Jawad *et al.*, Chen *et al.*, Buczak *et al.* and Liu *et al.*). We consider queries to the same domain as a flow and incorporate a rule, wherein if the percentage of flagged queries in a flow (denoted as PFQ) exceeds a threshold, the flow (domain) is flagged and the attack is detected. We use the following metrics to evaluate DOLOS.

- *Blackbox detection probability (BDP)*: We compute the probability that the exfiltrating primary domain(s) is detected by "at least" one of the defenses. This probability is given by $1 - \prod_i (1 - p_i)$, where p_i is the probability of being detected by defense method d_i.
- *AUC score*: To measure the stealth of DOLOS, we measure a defense's ability to distinguish benign and malicious traffic by using the receiver operating characteristic (ROC) curve, and we report the area under that curve (AUC) score. A low AUC score means that the defense is poor in performing the distinction, which is good for the attacker (e.g., DOLOS).

We defer the explicit details on the thresholds chosen for the various methods for issuing alerts and rationale for the same to Appendix A because of space issues.

Default settings. Unless otherwise specified, by default we use DOLOS (ISI-rdns) that is trained on the Text dataset, and defenses that are trained on ISI-host datasets. The results are consistent in behavior with the other datasets and with DOLOS (GT), and we showcase samples of several of those. We also use a single exfiltration domain by default.

6.2. Evaluation results

Due to space limits, we present the core results relating to stealth and exfiltration speed of DOLOS in this section. Additional results on speed and stealth relating to the considered datasets and an ablation study are provided in Appendices B, C and D.

Holistic evaluation of DOLOS. We evaluate DOLOS holistically against baselines as they are used today. We apply all of the rule based and anomaly detection based defenses sequentially (together) and compute the blackbox detection probability (BDP). The defenses are trained on the ISI-host dataset. We exclude the classification methods in computing these plots (i.e., [9], [11], [53]) because they detect the baselines almost surely regardless of the rate they use, based on simply the features of the encoding, while DOLOS is unaffected (details are discussed later in Table 2). Our experiments, upon including these defenses, showed that the absolute performance with DOLOS was unchanged from what is discussed below; however, the BDP with the baselines was ≈ 1.0 regardless of the rate, thus, precluding them from exfiltrating almost any data. Our evaluations are with all considered exfiltrated datasets together (i.e., Text, credit cards, logs and Images) and we

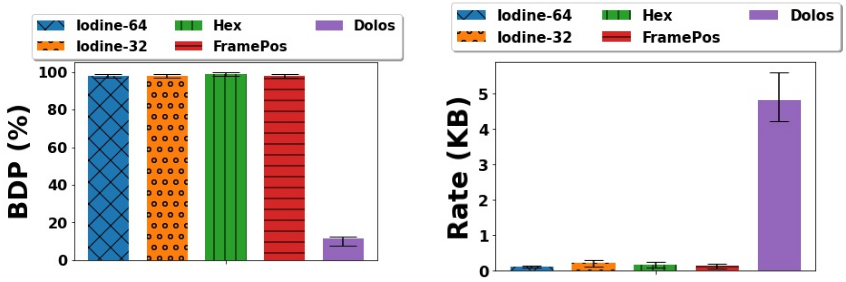

Figure 5: On the left is the blackbox detection probability when baseline methods use a constant exfiltration rate commensurate with the average rate of Dolos with deployed anomaly and rule based detection methods. On the right is the maximum rate that baselines can send with a fixed BDP of 0.15.

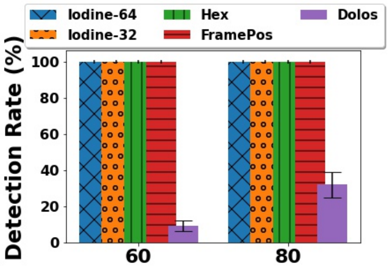

Figure 6: Detection rate with Jawad et al., and Ishikura et al., with multiple exfiltration sites (60 and 80 sites).

report the average performance with confidence intervals in Figs. 5 and 6. Further results relating to individual datasets are in Appendix B.

The BDP of baselines are almost close to 1 while* Dolos *experiences a BDP of only 0.12, when they exfiltrate data at the same rate: We perform multiple experiments using Dolos to exfiltrate a set of files from a different exfiltration dataset (e.g., medical images, credit cards) in each run, with an average data size of $\approx 5KB$ from a single host to one external site; this takes on average, 12 hours. We exfiltrate the same file(s) with the baselines, with the *same average rate* of Dolos. To set this rate, we vary the chunk length and choose the longest one that does not flag any of the methods (making query length a non-factor). We then choose a constant rate given this query length, to commensurate with the average rate of Dolos. We see that, at this rate, the BDP of the baselines are almost close to 1 (> 0.95); the baselines have many disadvantages compared to Dolos (e.g., their encoding, fixed rate, lack of spurious queries) and these tend to trigger at least one defense. In contrast, DOLOS has a BDP of just 0.12.

If the baselines use a low rate to avoid detection,* Dolos *can exfiltrate data 25× faster (with an even lower BDP): Now, in contrast to keeping the rate fixed, we fix the tolerable BDP. We search for the most conservative rate that keeps the BDP to below 0.15 for all the methods (similar to what Dolos achieved in the prior experiment). In such a case, we see in Fig. 5 that Dolos is able to transfer 25 times more data than the baselines in a given fixed time.

Increasing the number of exfiltration sites helps* Dolos *boost its exfiltration rate, but does not help baselines. Next, we examine the exfiltration of files from a single host to multiple exfiltration sites in the control of the attacker. We only use Jawad et al., and Ishikura et al., as our defenses since these allow us to explicitly showcase the impact of increasing the number of remote sites. Jawad et al., imposes a maximum length constraint on the queries sent by the exfiltration methods. Ishikura et al., counts the number of cache misses per host, which can increase as the number of external sites to which the attacker sends queries increases. Again, for the baselines, we choose the maximum length that does not flag Jawad et al., ensuring that rate and the number of external sites are the only factors that influence detection. Dolos uses its rate tuning to be consistent with the hosts's query rates. The results are shown in Fig. 6. The baselines are almost surely detected because unlike Dolos, they send at fixed rate (increased as a consequence of the length limit due to Jawad at al.) and hence, often exceed the cache miss limit. There is an increase in the detection probability of Dolos as well, because Ishikura et al., counts the cache misses across all its connections to the plurality of sites (in fact, as the number of sites increase the detection rate is likely to increase). However, this increase is modest as seen in Fig. 6.

Stealth against individual defenses. We evaluate Dolos 's encoding in evading defenses. We consider exfiltration to a single domain, since most defenses are domain specific. We use the two versions of Dolos (i.e., Dolos (GT) and Dolos (ISI-rdns)) in this experiment. We train the anomaly detectors with the two training datasets exclusively used for defenses (see §6.1). Classifiers have access to the attacker's encoding models. To zoom in on the benefits of DOLOS's encoding, *we empower the baselines with* Dolos*'s rate tuning module (RT)*; in essence they perform markedly better than they would in their native forms (as demonstrated later). In what follows, we only report results associated with the Text dataset in Tables 2 and 3 due to space constraints; results from the remaining datasets are in Appendix C.

Rule based detection methods have low detection rates on all encoding methods (including baselines) if empowered with* Dolos*'s rate tuning. Zeek does not detect any method because the maximum length of the queries and the rate of unique queries, are consistent with those at the host. Ishikura et al [39] fails to detect exfiltration as long as the exfiltration rate is consistent with benign DNS query generation on the host (cache misses rarely exceed threshold in the time window chosen by the method). ZeekQ does not detect any of the encoding methods including Dolos, because the percentage of numerical characters conform to its threshold. Paxson *et al.,* has a slightly higher detection probability on methods that aggressively pursue encoding efficiency (the compressed volume is higher). Thus, both Iodine-64 and Iodine-32 which have higher information per bit compared to the others (regardless of query length), are more likely to be detected.

Classification methods almost always detect baselines but* Dolos *evades them because of its ability of generative diverse code books. Buczak et al. [9], Liu et al. [53], Chen et al. [11] learn and thereby almost surely detect signatures generated by the baseline methods (Table 2); however, DOLOS goes undetected since it creates diverse sets of codes by simply re-training the encoding-decoding framework. Thus, these defenses are unable to build a signatures of traffic from DOLOS. We emphasize that these classification methods are whitebox i.e., they have access to the model architecture and the dataset used to train DOLOS; in spite of it, these classification methods fail to detect DOLOS queries, because DOLOS generates different codes with each fresh training. The locations of the very small fraction of queries using Base-32 encoding with DOLOS are staggered depending on the input, and even if a few of these are detected, these defenses cannot easily categorize a domain as malicious. We provide details with these methods in Appendix F.

Anomaly detection methods are effective in detecting baselines but fail to detect* DOLOS *due to the similarity of the queries it generates, to benign traffic. Nadler *et al.* [64] detects high entropy codes (e.g., Base-64, and Base-32) and thus is effective in detecting the Iodine variants. The other methods including DOLOS are less easily detectable; specifically, the entropy of DOLOS is very similar to that of benign queries. Jawad *et al.* [3] achieves much higher detection rates with the baselines (> 0.5 detection probability), but DOLOS goes undetected. This is because even when considering multiple features, DOLOS's encoding largely resembles benign traffic enabling it evade even this arguably strongest among defenses that do not need to be trained with malicious samples.

AUC scores. We assess the ability of the encoding methods in generating queries indistinguishable from benign traffic, using AUC scores (see §6.1) in Table 4. Since, Jawad *et al.,* considers the length of a query in making an inference, to ensure that length is a non-factor in triggering anomalies (only the encoding matters), we impose that malicious traffic of the other baselines are also consistent with the benign query lengths (DOLOS's GAN based encoding ensures this). We use multiple ROC curves where, in each, fake and benign queries of equal length are plotted. *On average* DOLOS *has ≈2× lower AUC score when compared against the best baseline (i.e., Iodine-32) when tested against Jawad* et al. *[3].* The lower AUC with DOLOS implies that the baseline encoding methods are more likely to be detected than DOLOS even if query length is not a factor, i.e., the encoding with DOLOS can better evade detection compared to baselines.

A case study from previous DNS exfiltration attacks. In a previous attack [35], attackers were able to steal 40M credit cards from US Target stores in 2013 (estimated to be from 1790 stores at that time [87]). The total amount of exfiltrated data was ≈ 4GB. Motivated by this, we ask how fast we can exfiltrate this amount of data compared to the baselines when the attacker compromises the same number of devices. We fix the number of queries per day to 80 and transmit to 120 remote servers, and exfiltrate a 4GB file over varying number of days. We evaluate DOLOS (with the same fixed rate (DOLOS (FT)) along with the baselines with Jawad *et al.*; this is the best performing defense (other defenses are inferior as seen in Table 2) from those that do not need malicious samples for training. We also show how holistic DOLOS performs.

DOLOS *can exfiltrate data more than 2.5 × faster, than the baselines, without being detected, when evaluated with Jawad* et al. *[3].* We consider the fastest time to exfiltrate a 4GB file with different PFQ thresholds (see § 6.1). The results in Fig. 7 show that DOLOS exfiltrates data much faster and yet is undetected. For example, with the logs dataset, DOLOS can potentially go almost undetected even when completing exfiltration in 3 days; in comparison, the closest baseline takes over 8 days if it is to go undetected. In practice, the holistic version of DOLOS (shown by the red stars) uses rate tuning (the best fixed rate is unknown) and so performs slightly worse. Importantly, if any of the baselines was to exfiltrate data within 4.5 days (worst case with full DOLOS), they would almost surely be detected. We also want to emphasize that this is only with the defense by Jawad at al. As discussed earlier, if a plurality of defenses are used, the baselines almost always are detected, unlike DOLOS (e.g., with classifiers).

Queries with decoding errors. Next, we report the percentage of queries that were handled by DOLOS's error recovery module. The percentages of such queries for Text, credit cards, Logs and Images datasets are 8%, 5%, 4% and 18%, respectively. While images have the highest error rates (in comparison with others) the rate is still very small to significantly affect stealth.

6.3. DOLOS Complexity

We had earlier discussed the time for training DOLOS offline (§ 6.1). Here, we examine its runtime complexity.

Clustering. The complexity of DOLOS's clustering depends on the benign query rate of the victim and the number of unique domains. Computations are amortized over a time window and require low CPU workloads. The average number of clusters per domain is 70 against each of which, each query is compared; this takes 1.5ms. With regards to *space overhead*, we store the number of clusters for each unique domain, as well as their representative queries. The maximum needed space for a 24-hour window is 0.6MB (600KB), which is insignificant with respect to today's computers.

Data encoding. We measure the average encoding duration at test time, on a 2.9 GHz laptop CPU with 3 different batch sizes (1, 8 and 16). The average encoding times are 1.4s, 0.452s and 0.28s, respectively. Since the exfiltration queries are sent at time scales of seconds/minutes to avoid detection, this overhead has a negligible influence on the rate. With regards to the *space overhead*, the model sizes for text, credit card and logs models and images are 59M, 22M and 18M, 30M, respectively. On average, the size of the encoder is 35MB, which is very small given the disk and memory of today's machines. The size of the decoder is the same as the encoder associated with a given dataset. The average decoding times for batch sizes (1,8 and 16) are 1.5s, 0.49s and 0.31s, respectively.

We break up the model into multiple files to reduce the storage footprint; the model can be composed from these scattered files in the memory during operation with negligible additional time. Further optimization to smaller and cheaper architectures (e.g., using model pruning or knowledge distillation) is viable but left for future work.

TABLE 2: Detection probability of defensive methods against the considered attacks. DOLOS evade all detection methods because of the similarity of its encoding to benign traffic; the baselines are flagged by at least a sub-set of the defenses.

Method	Anomaly Detection methods				Classification methods					
	Nadler et al		Jawad et al		Buczak et al		Liu et al		Chen et al	
	Thapar	ISI-HOST	Thapar	ISI-HOST	Thapar	ISI-HOST	Thapar	ISI-HOST	Thapar	ISI-HOST
Iodine-64 + (RT)	0.75	0.59	1	0.98	0.89	0.9	0.92	0.94	0.94	0.96
Iodine-32 + (RT)	0.25	0.32	0.51	0.54	0.88	0.88	0	0	0.94	0.96
Hex + (RT)	0.03	0.02	0.53	0.58	0.92	0.92	0	0	0.94	0.96
FramePos + (RT)	0	0.09	0.57	0.87	0.92	0.92	0.89	0.91	0.94	0.96
DOLOS (GT)	0.04	0.01	0.03	0.06	0	0	0	0	0	0
DOLOS (ISI-rdns)	0.04	0.02	0.02	0.04	0	0	0	0	0	0

TABLE 3: Detection probability of rule based defensive methods against attacks (empowered with DOLOS's rate tuning).

Method	Rule Based Methods							
	Zeek		ZeekQ		Ishikura		Paxson et al	
	Thapar	ISI-HOST	Thapar	ISI-HOST	Thapar	ISI-HOST	Thapar	ISI-HOST
Iodine-64 + (RT)	0	0	0	0	0	0	0.17	0.3
Iodine-32 + (RT)	0	0	0	0	0	0	0.1	0.16
Hex + (RT)	0	0	0	0	0	0	0	0.07
FramePos + (RT)	0	0	0	0	0	0	0	0
DOLOS (GT)	0	0	0	0	0	0	0	0.08
DOLOS (ISI-rdns)	0	0	0	0	0	0	0	0.06

TABLE 4: AUC scores of encoding methods with Jawad et al.

Method	Text	Credit Cards	Logs	Images
Iodine-64	0.996	0.996	0.996	0.996
Iodine-32	0.63	0.62	0.68	0.68
Hex	0.995	0.995	0.995	0.995
FramePos	0.998	0.998	0.998	0.998
DOLOS	0.34	0.35	0.34	0.38

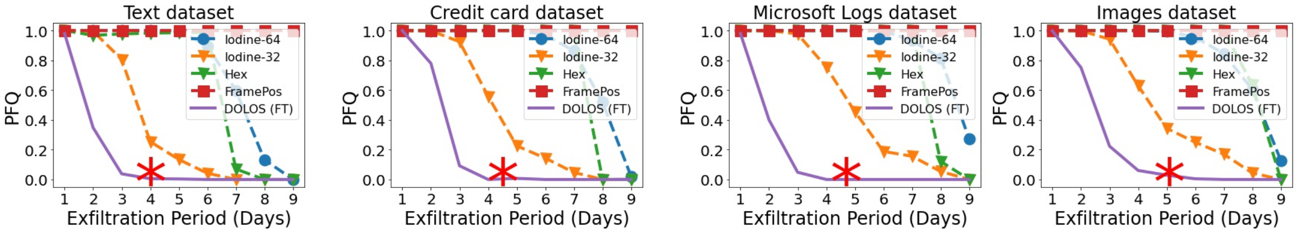

Figure 7: Jawad *et al.* defense [3] PFQ on Text, Credit Card, and Logs and Images datasets, respectively, for varying exfiltration rates. The PFQ when using DOLOS is much lower than encoding baselines. The red stars represent DOLOS's (holistic) performance.

6.4. Real implementation details and results

We host a domain we purchased from godaddy.com, on two name servers as described in RFC [63]. The two name servers are Windows server 2019 virtual machines deployed on Microsoft Azure. We use the default Microsoft DNS server [61], and we generate *CNAME* DNS records for the generated domain as **.dolos.com* (fake name for anonymity reasons). This makes the server respond to DNS queries with any subdomain under *dolos.com* with the default configured IP address. We use the default DNS logging system to decode queries to their original representations. Client queries (regardless of whether they are benign or malicious) are generated using a Mac pro laptop with `nslookup`. We use three popular public DNS resolvers: Google (`IP:8.8.8.8`), Control D (`IP:76.76.10.0`) and CloudFlare (`IP:1.1.1.1`). In addition we also experiment with the DNS resolver (owned by our institution). We capture DNS queries on the local machine and extract the features required to make an inference using the defenses we considered in § 6.1. In our experiments we exfiltrate a 10KB text file with an average chunk length of 90 characters. We repeat the exfiltration with DOLOS and each baseline, but the benign queries are only those that are naturally generated. These constraints are imposed because we do not want to cause unintentional damage to either the public or our institutional DNS resolvers (e.g., DDoS attacks).

Real Implementation results. Since we are unaware if these real systems deploy defenses, we also pass the traffic through Jawad et al., for safety. Flagged queries are stopped and do not reach the destination. We use all considered name servers in this experiment. Finally, we ended up retrieving all fake queries that were not detected by Jawad et al., which suggests that the public DNS resolvers do not employ any potent defense (although we cannot verify this explicitly). We will investigate this in the future by talking to administrators or other means (e.g., security blogs).

7. Discussion

Defending against DOLOS. One potential defense against DOLOS is to use program analysis to identify programs responsible for creating DNS queries for exfiltration. One can check if a program accesses (i) confidential files and (ii) inputs those to a processing engine. However, running such analyses on all programs on all hosts is expensive. Instead, one can monitor the DNS queries and use a detector or ensemble of detectors with somewhat lower detection thresholds (i.e., higher detection rate of attacks but high false positives). Expensive program analysis can be used as a second filtration layer on only those programs that generated these queries.

Exfiltrating multiple types of data. To exfiltrate multiple types of data, one can train and use multiple encoders. At run time, the malware can include a codeword to indicate the type of data being exfiltrated. The remote server then uses the appropriate decoder for recovery.

Whitelists and their impact on DOLOS. Enterprises could use whitelists to block access to DOLOS's remote server. Today, the deployed whitelists specify the *only* domains that users can access (i.e., all other domains are inaccessible/blocked by default). [65] suggests that the top (most popular) one million domains could be used as such, in a whitelist. However, we argue that such whitelists are highly impractical in the real world, because access patterns to domains/websites could vary significantly over time. This makes it infeasible for network operators to keep such lists up to date without inducing blockages to benign domains. Google reports that about 4% of the 2 billion domains are accessed with between 10 and 1000 visits, in a month [81]. This is about 80 million websites and not all of these can be in the top-1-million whitelist described in [65]. In other words, having whitelists with a million pages causes more problems for defenses in terms of false positives and thus is not useful against DOLOS.

Out-of-order packet delivery. In-order packet delivery from the compromised host to the remote domain is not guaranteed by the DNS protocol. To ensure delivery guarantees, new attributes such as sequence numbers may be needed. This aspect is left for future work. We highlight that DOLOS decodes each query independently. Thus, we believe that using the decoded queries, an ML model can assess the received queries to determine whether they were received in order and rectify the order if necessary.

8. Related Work

Text generation. GPT [75] and BERT [17] are ML methods to synthesize text similar to training text. However, our problem differs since we seek a latent space representation, largely decodable to its raw form. Similarly, GANs have been used for text generation [49], [92] but do not handle multiple goals (mimicking benign traffic and ensuring decodability). Recent work uses multiple generators and discriminators with different loss functions in order to achieve a single objective (e.g., generating better synthetic images [26], [82]). However, DOLOS seeks to fulfil multiple objectives.

Adversarial Perturbations. There is work on perturbing inputs to deceive a neural network (e.g., [85], [93]). Such methods cannot be readily applied to modify exfiltration data. NIDSGAN [93] perturbs packet headers while adhering to domain constraints to deceive ML-based intrusion detection systems. However, they only have one objective (evasion).

Encoder-decoder frameworks. ML-based encoder-decoder frameworks have been studied. For example, [20] builds such a framework to send text data over an erasure channel. This work differs from ours in two ways. First, the latent representations are continuous whereas in our case, they are discrete (they form the fake queries). Beyond this, our approach has constraints on the latent space (it should resemble benign traffic). A work close to ours is [43], where the authors design an end-to-end communication system over an erasure channel but constrain the latent representation to binary values. The problem differs from ours because, in addition to constraining the representation to characters used by DNS, we also have additional requirements (e.g., stealth and decodability).

Detection. As discussed, defenses against DNS exfiltration are either rate based [18], [46] or encoding based [16], [42], [50], [51], [58], [91]. These methods were either considered in our evaluations in § 6.2 or are very similar to those considered and we expect them to fail in detecting DOLOS.

DNS Exfiltration tools. Today's DNS exfiltration tools such as DNSmessanger [8], DNSteal [24] and Iodine [15], use DNS query types that do not conform with benign traffic and/or use the traditional encoding (e.g., Base-64, Base-32 and Hex) that are inefficient and detectable. For example, DNSmessager uses DNS type `TXT`, which, as discussed, is easily detectable.

9. Conclusions

In this work, we show that contrary to the common belief that current defenses have succeeded in curbing DNS exfiltration, our GAN-guided approach, DOLOS, can achieve successful exfiltration. DOLOS encodes the exfiltration data such that it not only mimics benign traffic to fool defenses but also achieves significantly faster exfiltration than what is possible today. In addition, it includes a rate tuning module that is key in preventing the attack from being detected in a blackbox setup. Finally, DOLOS also ensures that exfiltrated data is fully decodable at an external site like with traditional methods. Our evaluations show that DOLOS can exfiltrate data 25 × faster than the baselines, if the detection probability needs to be kept low (< 0.15).

Acknowledgement

This research was sponsored by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). 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 Combat Capabilities Development Command 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. We also thank the anonymous reviewers and our anonymous shepherd for their comments and suggestions that helped significantly improve our paper.

References

[1] Loghub: A collection of system log datasets for intelligent log analysis. https://github.com/logpai/loghub, 2019.

[2] National Security Agency. Adopting encrypted dns in enterprise environments. https://media.defense.gov/2021/Jan/14/2002564889/-1/-1/0/CSI_ADOPTING_ENCRYPTED_DNS_U_OO_102904_21.PDF, 2021.

[3] Jawad Ahmed, Hassan Habibi Gharakheili, Qasim Raza, Craig Russell, and Vijay Sivaraman. Monitoring enterprise dns queries for detecting data exfiltration from internal hosts. *IEEE Transactions on Network and Service Management*, 17(1):265–279, 2019.

[4] Akamai. Dns: The easiest way to exfiltrate data? https://www.akamai.com/blog/security/dns-the-easiest-way-to-exfiltrate-data, 2022.

[5] Isabela Albuquerque, Joao Monteiro, Thang Doan, Breandan Considine, Tiago Falk, and Ioannis Mitliagkas. Multi-objective training of generative adversarial networks with multiple discriminators. In *International Conference on Machine Learning*, pages 202–211. PMLR, 2019.

[6] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[7] Seth Bromberger. Dns as a covert channel within protected networks. *National Electronic Sector Cyber Security Organization (NESCO)(Jan., 2011)*, 2011.

[8] Edmund Brumaghin. Covert Channels and Poor Decisions: The Tale of DNSMessenger. http://blog.talosintelligence.com/2017/03/dnsmessenger.html, 2017. [Online; accessed March-15-2022].

[9] Anna L Buczak, Paul A Hanke, George J Cancro, Michael K Toma, Lanier A Watkins, and Jeffrey S Chavis. Detection of tunnels in pcap data by random forests. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, pages 1–4, 2016.

[10] Joseph Carson. Privilege escalation on linux: When it's good and when it's a disaster (with examples). https://delinea.com/blog/linux-privilege-escalation, 2020.

[11] Shaojie Chen, Bo Lang, Hongyu Liu, Duokun Li, and Chuan Gao. Dns covert channel detection method using the lstm model. *Computers & Security*, 104:102095, 2021.

[12] Jinyoung Choi and Han Bohyung. Mcl-gan: Generative adversarial networks with multiple specialized discriminators. 2019.

[13] CloudFlare. Http policies. https://developers.cloudflare.com/cloudflare-one/policies/filtering/http-policies/, 2020.

[14] Cloudflare. Cloudflare Resource Hub. https://www.cloudflare.com/resource-hub/?resourcetype=Whitepaper, 2022. [Online; accessed Dec-1-2022].

[15] code.kryo.se. Iodine:code.kryo.se. https://github.com/yarrick/iodine, 2021.

[16] Anirban Das, Min-Yi Shen, Madhu Shashanka, and Jisheng Wang. Detection of exfiltration and tunneling over dns. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 737–742. IEEE, 2017.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[18] Wendy Ellens, Piotr Żuraniewski, Anna Sperotto, Harm Schotanus, Michel Mandjes, and Erik Meeuwissen. Flow-based detection of dns tunnels. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 124–135. Springer, 2013.

[19] EnterpriseDT. How to secure your sftp server. https://enterprisedt.com/blogs/completeftp/how-to-secure-sftp-server/, 2020.

[20] Nariman Farsad, Milind Rao, and Andrea Goldsmith. Deep learning for joint source-channel coding of text. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2326–2330. IEEE, 2018.

[21] Aaron Faulkenberry, Athanasios Avgetidis, Zane Ma, Omar Alrawi, Charles Lever, Panagiotis Kintis, Fabian Monrose, Angelos D Keromytis, and Manos Antonakakis. View from above: Exploring the malware ecosystem from the upper dns hierarchy. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 240–250, 2022.

[22] FireEye. Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor. https://www.fireeye.com/blog/threat-research/, 2021. [Online; accessed June-1-2021].

[23] Romain Fouchereau. IDC 2021 Global DNS Threat Report. https://www.efficientip.com/resources/idc-dns-threat-report-2021/, 2021. [Online; accessed April-20-2022].

[24] g0dmode. Dnssteal. https://github.com/m57/dnsteal, 2015.

[25] Gdatasoftware. New FrameworkPOS variant exfiltrates data via DNS requests. https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests, 2014. [Online; accessed Feb-26-2022].

[26] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8513–8521, 2018.

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[28] Google. Google place autocomplete api. https://developers.google.com/maps/documentation/places/web-service/autocomplete#maps_http_places_autocomplete_amoeba-py, 2020.

[29] Google. Google places api. https://developers.google.com/maps/documentation/places/web-service/overview, 2020.

[30] Google. kctf vrp setup. https://google.github.io/kctf/vrp.html, 2022.

[31] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448*, 2020.

[32] J Dinal Herath, Ping Yang, and Guanhua Yan. Real-time evasion attacks against deep learning-based anomaly detection from distributed system logs. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 29–40, 2021.

[33] Dominik Herrmann, Christian Banse, and Hannes Federrath. Behavior-based tracking: Exploiting characteristic patterns in dns traffic. *Computers & Security*, 39:17–33, 2013.

[34] Hhzzk. Dns-tunnels. https://github.com/hhzzk/dns-tunnels, 2017.

[35] Elise Hu. Hackers stole 40 million credit, debit card numbers from target. https://www.npr.org/2013/12/19/255559793/hackers-stole-40-million-credit-debit-card-numbers-from-target, 2013.

[36] iagox86. Dnscat2. https://github.com/iagox86/dnscat2, 2022.

[37] Infoblox. Data exfiltration and dns. https://www.infoblox.com/wp-content/uploads/infoblox-whitepaper-data-exfiltration-and-dns-closing-the-back-door.pdf, 2020.

[38] Infoblox. Preventing dns-based data exfiltration. https://www.infoblox.com/wp-content/uploads/infoblox-solution-note-preventing-dns-based-data-exfiltration.pdf, 2020.

[39] Naotake Ishikura, Daishi Kondo, Vassilis Vassiliades, Iordan Iordanov, and Hideki Tode. Dns tunneling detection by cache-property-aware features. *IEEE Transactions on Network and Service Management*, 18(2):1203–1217, 2021.

[40] GI Ivchenko and SA Honov. On the jaccard similarity test. *Journal of Mathematical Sciences*, 88(6):789–794, 1998.

[41] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[42] Iram Jawad, Jawad Ahmed, Imran Razzak, and Robin Doss. Identifying dns exfiltration based on lexical attributes of query name. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2021.

[43] Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels. *Advances in neural information processing systems*, 32:2758–2768, 2019.

[44] Chris Johnston. Home Depot: 56 million credit cards compromised. https://www.theguardian.com/business/2014/sep/19/home-depot-56m-credit-card-numbers-compromised, 2014. [Online; accessed June-15-2022].

[45] Kaggle. Covid-19 Image Dataset. https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset, 2021. [Online; accessed Jan-10-2023].

[46] A Mert Kara, Hamad Binsalleeh, Mohammad Mannan, Amr Youssef, and Mourad Debbabi. Detection of malicious payload distribution channels in dns. In *2014 IEEE International Conference on Communications (ICC)*, pages 853–858. IEEE, 2014.

[47] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[48] krebsonsecurity.com. Deconstructing the 2014 sally-beauty breach. https://krebsonsecurity.com/2015/05/deconstructing-the-2014-sally-beauty-breach/, 2015.

[49] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[50] Danielle Lambion, Michael Josten, Femi Olumofin, and Martine De Cock. Malicious dns tunneling detection in real-traffic dns data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5736–5738. IEEE, 2020.

[51] Jianbing Liang, Suxia Wang, Shuang Zhao, and Shuhui Chen. Fecc: Dns tunnel detection model based on cnn and clustering. *Computers & Security*, 128:103132, 2023.

[52] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[53] Jingkun Liu, Shuhao Li, Yongzheng Zhang, Jun Xiao, Peng Chang, and Chengwei Peng. Detecting dns tunnel through binary-classification based on behavior features. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 339–346. IEEE, 2017.

[54] Sheng Liu, Michael K Reiter, and Vyas Sekar. Flow reconnaissance via timing attacks on sdn switches. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pages 196–206. IEEE, 2017.

[55] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[56] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. An end-to-end, large-scale measurement of dns-over-encryption: How far have we come? In *Proceedings of the Internet Measurement Conference*, pages 22–35, 2019.

[57] Hans Peter Luhn. Computer for verifying numbers. *US Patent*, 2(950):048, 1960.

[58] Samaneh Mahdavifar, Amgad Hanafy Salem, Princy Victor, Amir H Razavi, Miguel Garzon, Natasha Hellberg, and Arash Habibi Lashkari. Lightweight hybrid detection of data exfiltration using dns based on machine learning. In *2021 the 11th International Conference on Communication and Network Security*, pages 80–86, 2021.

[59] VASILENA MARKOVA. What is nxdomain? https://www.cloudns.net/blog/what-is-nxdomain/#:~:text=NXDOMAIN%20stands%20for%20a%20non,the%20domain%20does%20not%20exist., 2024.

[60] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908, 2013.

[61] Microsoft. Quickstart: Installing and configure dns server. https://learn.microsoft.com/en-us/windows-server/networking/dns/quickstart-install-configure-dns-server?tabs=powershell, 2022.

[62] Microsoft. White papers on the cloud and Azure. https://azure.microsoft.com/en-us/resources/whitepapers/search/?type=WhitePaperResource&Page=1, 2022. [Online; accessed Dec-1-2022].

[63] P. Mockapetris. *RFC 1035 Domain Names - Implementation and Specification*. Internet Engineering Task Force, November 1987.

[64] Asaf Nadler, Avi Aminov, and Asaf Shabtai. Detection of malicious and low throughput data exfiltration over the dns protocol. *Computers & Security*, 80:36–53, 2019.

[65] NetCraftsmen. USE DNS WHITELISTS TO STOP MALWARE IN ITS TRACKS. https://netcraftsmen.com/use-dns-whitelists-to-stop-malware-in-its-tracks/, 2022. [Online; accessed July-20-2022].

[66] PaloAlto Networks. Detecting DNS tunneling. https://www.paloaltonetworks.com/resources/videos/lightboard-dns-security-service, 2019. [Online; accessed October-01-2021].

[67] norton.com. What is a trojan downloader? https://us.norton.com/blog/malware/what-is-a-trojan-downloader, 2018.

[68] University of Southern California-Information Sciences Institute. Day in the life of the internet (ditl). https://www.impactcybertrust.org/dataset_view?idDataset=884, 2022.

[69] University of Southern California-Information Sciences Institute. Reverse dns. https://www.impactcybertrust.org/dataset_view?idDataset=702, 2022.

[70] openaddresses.io. batch.openaddresses.io/data. https://batch.openaddresses.io/data, 2016. [Online; accessed December-1-2021].

[71] OpenAI. ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/, 2022. [Online; accessed Dec-12-2022].

[72] PaloAlto. Strategically Aged Domain Detection: Capture APT Attacks With DNS Traffic Trends. https://unit42.paloaltonetworks.com/strategically-aged-domain-detection/, 2021. [Online; accessed December-15-2021].

[73] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[74] Vern Paxson, Mihai Christodorescu, Mobin Javed, Josyula Rao, Reiner Sailer, Douglas Lee Schales, Mark Stoecklin, Kurt Thomas, Wietse Venema, and Nicholas Weaver. Practical comprehensive bounds on surreptitious communication over {DNS}. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*, pages 17–32, 2013.

[75] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[76] Philippe Remy. Name dataset. https://github.com/philipperemy/name-dataset, 2021.

[77] Sam Rhea. Announcing cloudflare's data loss prevention platform. https://blog.cloudflare.com/data-loss-prevention/, 2021.

[78] Paul Roberts. What's The Value of a Stolen Chest X-Ray? More Than You'd Think. https://www.digitalguardian.com/blog/whats-value-stolen-chest-x-ray-more-youd-think, 2021.

[79] Manmeet Singh, Maninder Singh, and Sanmeet Kaur. Ti-2016 dns dataset. https://dx.doi.org/10.21227/9ync-vv09, 2019.

[80] Nir Soft. Dns query sniffer. https://www.nirsoft.net/utils/dns_query_sniffer.html, 2013.

[81] Tim Soulo. 90.63% of Content Gets No Traffic From Google. And How to Be in the Other 9.37% [New Research for 2020]. https://ahrefs.com/blog/search-traffic-study/, 2022. [Online; accessed July-20-2022].

[82] Jingwen Su and Hujun Yin. Efficient multi-objective gans for image restoration. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1855–1859. IEEE, 2021.

[83] Symantec. Sunburst: Supply Chain Attack Targets SolarWinds Users. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/sunburst-supply-chain-attack-solarwinds, 2021. [Online; accessed June-15-2021].

[84] Georgia Tech. Gt malware passive dns data daily feed (07/01/2015 to 12/31/2017). https://www.impactcybertrust.org/dataset_view?idDataset=520, 2015.

[85] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems*, 32, 2019.

[86] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[87] www.statista.com. Number of target stores in the united states from financial year 2006 to 2022. https://www.statista.com/statistics/255965/total-number-of-target-stores-in-north-america/, 2013.

[88] www.trendmicro.com. Targeted Attacks. https://www.trendmicro.com/vinfo/us/security/definition/targeted-attacks, 2020. [Online; accessed June-1-2022].

[89] xairy. Linux kernel exploitation. https://github.com/xairy/linux-kernel-exploitation.

TABLE 5: Hyper parameters used in training DOLOS.

Module (Parameter)	Discriminator	Encoder	Decoder
Embed size	400	600	600
hidden size	400	600	600
# of layers	2	2	2
# of heads	2	3	3
Learning rate	0.00004	0.00001	0.00001
# of parameters	195k	440K	440K

[90] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[91] Jiacheng Zhang, Li Yang, Shui Yu, and Jianfeng Ma. A dns tunneling detection method based on deep learning models to prevent data exfiltration. In *Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings 13*, pages 520–535. Springer, 2019.

[92] Guoqiang Zhong, Wei Gao, Yongbin Liu, Youzhao Yang, Da-Han Wang, and Kaizhu Huang. Generative adversarial networks with decoder–encoder output noises. *Neural Networks*, 127:19–28, 2020.

[93] Bolor-Erdene Zolbayar, Ryan Sheatsley, Patrick McDaniel, Michael J Weisman, Sencun Zhu, Shitong Zhu, and Srikanth Krishnamurthy. Generating practical adversarial network traffic flows using nidsgan. *arXiv preprint arXiv:2203.06694*, 2022.

A. Details of various parameters

Model parameters and training. As discussed in § 4, DOLOS is composed of three transformer-based neural networks with hyper-parameters specified in Table 5. We follow Algorithm 1 to train the model, with an average training time $<$ 9 hours per run, on two Tesla P100 GPUs.

Model used for validation during training.. We utilize a validation model (an anomaly detector) to assess the stealth of the generated codes during training. To train this model, we choose some of the commonly used features from existing defenses (specifically, entropy, numbers of capital and numerical letters, numbers of subdomains and maximum and average length of subdomains). We choose an isolation forest anomaly detection method. Specifically, we want to emphasize that our model is different from existing anomaly detection defenses that we compare against in our evaluations (i.e., Nadler et al [64] and Jawad et al [3]). The defenses we test against differ from our home grown model with regards to at least a sub-set of aspects (the architecture is different and/or the features that they consider are different). Importantly, the datasets that these defenses are trained on, are different from that were used to train our validation model.

Defense thresholds. Here, we discuss how we set the thresholds of the various considered methods for issuing an alert.

(a) For Zeek, we set the length and "number of queries" thresholds to the maximum of those observed on the host in a validation time window. This is to ensure that the method does not flag benign domains. We do the same with Paxson et al. [74], but with the compressed volume instead.

(b) With regards to Ishikura et al, we compute the cache misses for all hosts within a time window (24 hours) as suggested by the original paper and choose the maximum as the threshold for the next time window. Note that in the ISI-host dataset there are 3000 plus hosts, in which a subset of them are considered to have the DOLOS malware (one or more). In the original work, the authors suggest using the 99th percentile (instead of the maximum), but this results in a 1% alert rate even if there is no attack activity. We observe that using the 99th percentile does not change the results in Table 2. Apart from the 1% of the hosts that are flagged because of the conservative threshold, no new hosts are flagged.

(c) With respect to Nadler et al., we choose the anomaly thresholds dynamically per host. In particular, the method outputs an anomaly score for each client-domain pair in the data set. We choose the anomalous score that would set the domain false positive rate to 10^{-5} and use this to set the threshold as suggested in the paper [64].

(d) For query-specific defenses (Jawad et al., Buczak et al., Liu et al., Chen et al.), we do not choose thresholds dynamically (since those papers do not discuss how to issue domain or flow specific alerts). We choose a fixed threshold for flagging queries, that corresponds to roughly a 1% PFQ with respect to benign queries. This corresponds to the results obtained with [11], [53]. Then, if PFQ exceeds a threshold (for a domain) that we assume, an alert is issued. Note that usually defenders set the domain false positive to very small percentage (e.g., zero as in the case of Paxson et al [74] or 10^{-5} as the case in Nadler et al [64]). For classification methods, we chose the same false positive rate as Nadler et al., because they are ML models as well; furthermore, these models are expected to provide even better classification results being exposed to the exfiltration samples as well (as discussed in § 6.1). Jawad et al, does not need malicious traffic to train on; when we tried to set the threshold to yield a false positive rate of 10^{-5} to be consistent with the others, the performance of the method dropped significantly (in terms of detection of all attacks including DOLOS and baselines). Thus, we use a more conservative threshold corresponding to a false positive rate of 10^{-3}. This would imply a better attack detection but a higher rate of false positives when no attack exists.

(e) With ZeekQ, two rules are set identical to Zeek. With respect to the additional rule, there are no specifications on what the threshold on the percentage of numerical characters in a query should be set. We check the percentage in benign queries, and set it to the 99^{th} percentile; this ensures that the PFQ is about 1 % in benign settings.

B. Holistic evaluation of DOLOS with various datasets individually

In this part, we provide the holistic evaluation of DOLOS, but individually with each of the considered datasets (as opposed to with all datasets jointly as reported in the main paper). As shown in Figs. 8, 9, 10, 11,12, 13, 14 and 15 the results of individual datasets do not deviate much from the average case, which demonstrates that DOLOS is equally effective in exfiltrating various types of data.

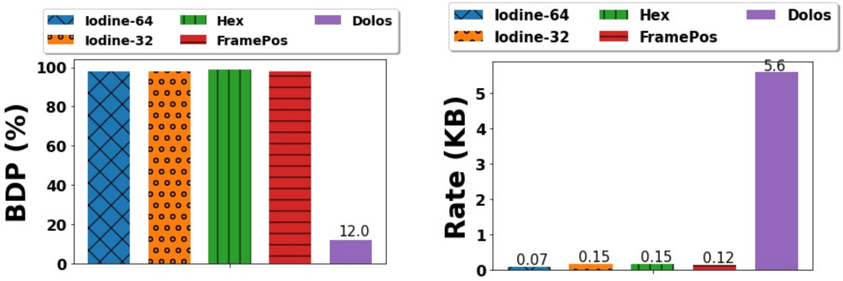

Figure 8: On the left is the blackbox detection probability when baseline methods use a constant exfiltration rate commensurate with the average rate of DOLOS with deployed anomaly and rule based detection methods. On the right is the maximum rate that baselines can send with a fixed BDP of 0.12. These results relate to exfiltration of text data.

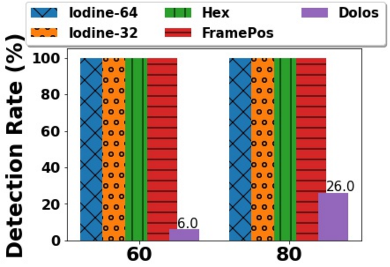

Figure 9: Detection rate with Jawad et al., and Ishikura et al., with multiple exfiltration sites (60 and 80 sites). These results relate to exfiltration of text data.

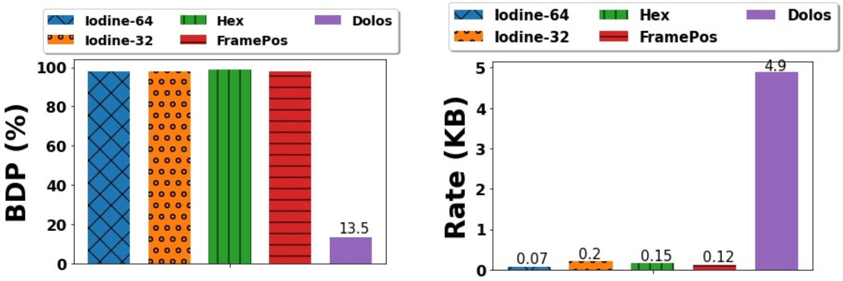

Figure 10: On the left is the blackbox detection probability when baseline methods use a constant exfiltration rate commensurate with the rate of DOLOS with deployed anomaly and rule based detection methods. On the right is the maximum rate that baselines can send with a fixed BDP of 0.14. These results relate to exfiltration of credit card data.

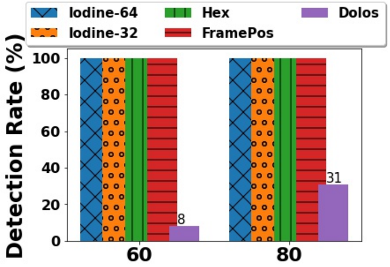

Figure 11: Detection rate with Jawad et al., and Ishikura et al., with multiple exfiltration sites (60 and 80 sites). These results relate to exfiltration of credit card data.

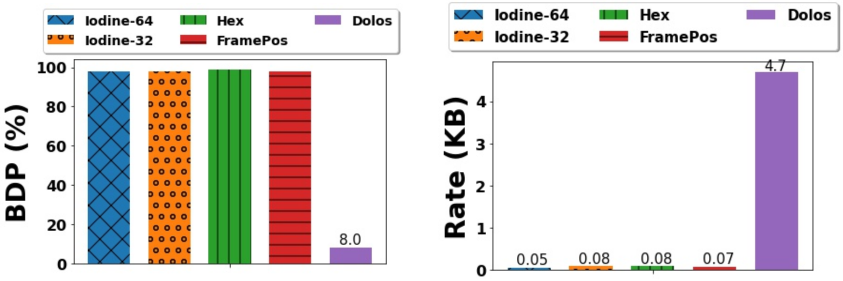

Figure 12: On the left is the blackbox detection probability when baseline methods use a constant exfiltration rate commensurate with the average rate of DOLOS with deployed anomaly and rule based detection methods. On the right is the maximum rate that baselines can send with a fixed BDP of 0.08. These results relate to exfiltration of logs data.

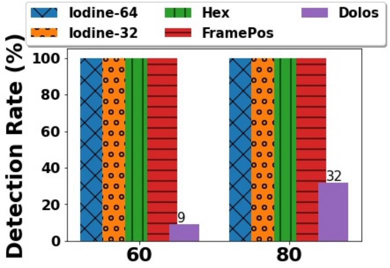

Figure 13: Detection rate with Jawad et al., and Ishikura et al., with multiple exfiltration sites (60 and 80 sites). These results relate to exfiltration of logs data.

C. Stealth with various datasets

In this section, we report the stealth (ability to evade) with respect to individual defenses with different exfiltration datasets. We use the same setup as those corresponding to Table 2. The performance of the classification methods and the rule based methods (i.e., Zeek, ZeekQ and Ishikura et al) are almost identical to the ones in Tables 2 and 3; thus, we do not include them here. We only report the performance of defensive methods that have changed to some extent when a different dataset is used. The results are shown in Table 6.

D. Ablation Study

In this part, we conduct an ablation study to showcase the importance of the different design choices we make in

TABLE 6: Detection probability of defensive methods against the considered attacks with the various considered datasets.

Datasets	Method	Rule based method	Anomaly Detection Threshold	
		Paxson et al	Nadler et al	Jawad et al
Credit Card	Iodine-64 + (RT)	0.3	0.59	0.98
	Iodine-32 + (RT)	0.2	0.32	0.52
	Hex + (RT)	0.12	0.02	0.6
	FramePos + (RT)	0	0.09	0.87
	DOLOS	0.1	0.03	0.01
Microsoft Logs	Iodine-64 + (RT)	0.12	0.59	0.98
	Iodine-32 + (RT)	0.04	0.32	0.51
	Hex + (RT)	0	0.02	0.62
	FramePos + (RT)	0	0.09	0.97
	DOLOS	0.03	0.02	0.03
Images	Iodine-64 + (RT)	0.3	0.59	0.98
	Iodine-32 + (RT)	0.17	0.32	0.54
	Hex + (RT)	0.1	0.02	0.58
	FramePos + (RT)	0	0.09	0.87
	DOLOS	0.05	0.08	0

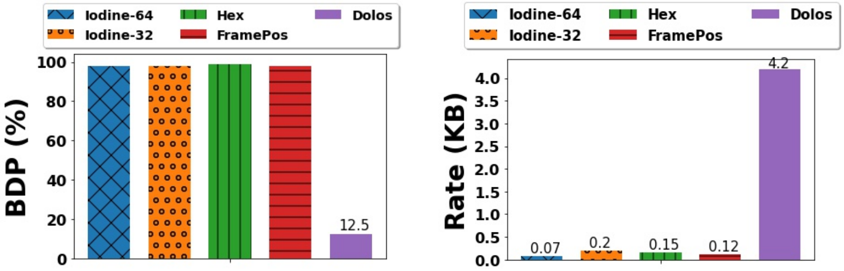

Figure 14: On the left is the blackbox detection probability when baseline methods use a constant exfiltration rate commensurate with the average rate of DOLOS with deployed anomaly and rule based detection methods. On the right is the maximum rate that baselines can send with a fixed BDP of 0.14. These results relate to exfiltration of image data.

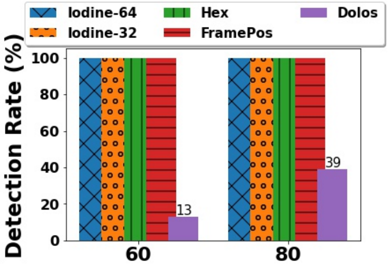

Figure 15: Detection rate with Jawad et al., and Ishikura et al., with multiple exfiltration sites (60 and 80 sites). These results relate to exfiltration of image data.

TABLE 7: Detection probabilities with the different encoding methods on Text dataset when they are equipped with DOLOS rate tuning but with no spurious queries (NQ).

Method	Rule Based Method	Anomaly Detection methods		Classification methods		
	Paxson et al	Nadler et al	Jawad et al	Buczak et al	Liu et al	Chen et al
Iodine-64 + (RT+ NQ)	0.29	0.81	1	1	1	1
Iodine-32 + (RT+ NQ)	0.13	0.47	0.62	1	0	1
Hex +(RT+ NQ)	0.06	0.18	0.6	1	0	1
FramePos + (RT+ NQ)	0	0.14	0.89	1	1	1
DOLOS + NQ	0.05	0.23	0.05	0	0	0

developing DOLOS. In particular, we discuss the performance of (1) DOLOS without our proposed encoding, (2) DOLOS without rate tuning, (3) DOLOS without spurious queries and (4) DOLOS without the error recovery module.

DOLOS without our proposed encoding methods. In Tables 2 and 6, we show multiple baselines when they are equipped with DOLOS rate tuning methods. Such baselines are essentially versions of DOLOS that do not have its GAN based encoding (has only its online rate tuning and spurious query injection components). As seen in these tables, these methods are almost surely detected by at least one of the detection methods. This demonstrates the importance of DOLOS's GAN based encoding.

Deploying DOLOS without rate tuning. We refer to Fig. 7 where a DOLOS with fixed rate is shown. While the best fixed rate can potentially speed up the exfiltration, it is hard to know this rate a priori. Choosing the wrong fixed rate can leave DOLOS vulnerable to easy detection. This is why we use rate tuning, and see (as shown by the red stars) in the figure that this full version of DOLOS only experiences a slight slow down compared to the best fixed rate one can use.

Impact of using spurious queries. we examine the performance of DOLOS without the spurious queries. We show the detection rates of all exfiltration methods in Table 7 when tested with Text dataset. We see that that detection rates of all methods have significantly increased in comparison with the results in Table 2. The omission of spurious queries improves detection (and worsen's DOLOS's stealth) with a significant subset of the defenses because these identify the absence of such queries in the stream. The most significant effect is seen with Nadler et al. This is because this method takes entropy as an explicit feature for classification which increases without such queries.

Deploying DOLOS without its error recovery module. Recall that due to the lossy nature of DOLOS's encoder, queries with decoding errors are fed to an error recovery module that guarantees data decodability at the remote site. In this experiment, we send the faulty queries even if they have errors i.e., use DOLOS without the error recovery module. While the received data has a small number of errors, we can now obtain a slightly higher exfiltration rate. We observe this especially with the image dataset. We discuss in the next section, the decoding error rates and ways to recover from those errors at the remote site. For example, one can possibly use NLP methods that understand the context of the received decoded chunks and correct them accordingly; for images, we observe that the pixel values are only changed slightly, retaining their structure and properties i.e., the changes are imperceptible to a human.

E. Decodability of DOLOS's lossy encoder's outputs

Here, we directly evaluate the decodability of DOLOS's lossy encoding and usefulness if lossy data is exfiltrated (without recovery) with various datasets.

Text datasets: We first look at the decoding accuracy without any external post processing. The attacker obtains 115K records from 130K with no errors (around 87%).

One way of achieving error correction post exfiltration is to use NLP methods, which understand the context of the received text and can perform corrections. For this purpose, we examine the use of the popular ChatGPT tool [71]. We exfiltrate 30 white-papers on different technical topics (e.g., denial of service (DoS), 5G security) from Microsoft and CloudFlare from 2022 [14], [62], and pass the decoded stream to chatGPT. The maximum observed percentage of words in error is less 1.1% across all the documents.

Credit card dataset: The attacker acquires 91% of the records with no errors (i.e., 275K out 300K credit cards were decoded correctly). Most errors are either in the address or the name parts. Fewer than 20 credit cards had errors in the number, CVV or expiry date; thus, if only the credit card number is needed for a transaction, almost all decoded data are usable. When the billing address and names are needed, we use correction methods (see below) that ultimately render 296K out 300K (98.7%) of credit cards usable. We use Google Maps API [28], [29] to validate the address and correct it if in error (i.e., by suggesting the closest match). For the names part, we use a big dataset of names that validates the name as a natural name, or provides the closest suggestion if no match is found.

Logs datasets: 94% of the logs in our test set are decoded with no errors. Records with errors can still be useful to the attacker in reconnaissance of the victim (i.e., the contacted IP addresses, the duration of open sessions). Further, leaking this data can violate the host's privacy [31]. With a small case study on Linux logs dataset, we observe that errors are mostly in the process numbers which all can be corrected using brute force.

Image datasets: We measure the number of correctly decoded chunks per image, and we find that 82% of DNS queries describing an image are received with no errors. In the image space, We observe that the decoder outputs a pixel intensity that is very close to actual pixel value. This makes the errors in the images imperceptible to humans.

Summary: These results suggest that our DOLOS encoder can even be used without our error recovery module in most cases, i.e., even in its absence, the attacker can still receive almost perfect and useful content. This can potentially improve DOLOS's stealth further, since it eliminates all Base-32 encoding, if such errors are tolerable.

F. A closer look at the performance with the classification-based detection methods

Next, we take a closer look at the performance of the various attacks with the classification methods (see § 6.1) used for detection. Specifically, we report the percentage of flagged queries (PFQ) with the considered classification based defenses in Table 8. With regards to DOLOS, we report the results, when classification methods are tested on (i) the same encoding on which it was trained (meaning the exact instance of training is known to the classifier) and (ii) a different encoding instance of DOLOS, denoted as DOLOS (same) and DOLOS (diff), respectively. As seen in the table, the classification methods are extremely effective in detecting signatures that they know about. However, in the case where DOLOS generates codes with a different encoding instance (DOLOS (diff)), classification methods perform very poorly. Liu et al., has a relatively inferior performance in comparison with the other two methods. Specifically, the hand selected features used by the method can detect Iodine-64 and FramePos with high accuracy, but it does not detect the others. Understanding why this is the case falls within the realm of explainability of ML models and is beyond the scope of this work.

G. Linux logs results and case studies

We show the results associated our Linux Logs dataset. 95% of the logs were correctly decoded with no errors. We evaluate the AUC score with Jawad et al., and it is 0.34 (similar to what is observed with Microsoft logs dataset). As shown in Fig. 16, the detectability and exfiltration rate results with this dataset are consistent with the results obtained with the Microsoft logs dataset.

TABLE 8: Percentage of flagged queries (PFQ) when tested against classification methods. We report the results for Text dataset.

Method	Buczak et al	Liu et al	Chen et al
Iodine-64	0.85	0.99	1
Iodine-32	0.62	0.12	0.997
Hex	0.99	0.09	1
FramePos	0.99	0.72	1
DOLOS (diff)	0.07	0.02	0.09
DOLOS (same)	0.55	0.06	0.99

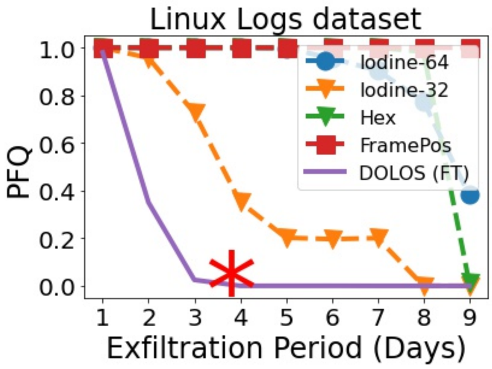

Figure 16: Jawad et al defense [3] PFQ, with the Linux logs dataset.