# Adversarial DNS Exfiltration: Framework and Defense Evaluation

Cooper Morgan[0009−0003−7830−3706], Logan Day[0009−0005−0596−8162], Filip Jagodzinski[0000−0001−7783−2241], and Hsiang-Jen Hong[0000−0002−3863−7192]

Western Washington University, Bellingham WA 98225, USA
{gibbonj9, dayl6, jagodzf, hongh6}@wwu.edu

**Abstract.** DNS exfiltration remains a persistent threat vector, yet research suffers from critical gaps: lack of standardized tools and semantically invalid synthetic attacks. We present a modular framework for DNS exfiltration research that maintains RFC compliance, temporal realism, and behavioral coherence with real network traffic. Our framework comprises four components ensuring semantic validity: payload generation, victim selection from real network data, time-based attack injection, and query validation. We validate the framework's modularity through three adversarial generators that each exercise a distinct component: a reinforcement-learning strategy selector, an LSTM benign-mimicry timing model, and Stegatetra. Stegatetra emerged from exploring what the encoding interface permits, evading all evaluated academic detectors by encoding data in query selection rather than query content. The framework also supports systematic defense evaluation by integrating existing state-of-the-art detectors alongside new ones such as ContraDNS, a contrastive learning detector we develop within the framework. Our evaluation reveals that no single detection paradigm catches all generator types, motivating multi-modal defense. We present this framework to enable reproducible DNS exfiltration research.

**Keywords:** DNS Exfiltration · Research Framework · Adversarial Machine Learning · Steganography · Network Intrusion Detection

## 1 Introduction

DNS exfiltration has emerged as a practical attack vector for data theft in restricted network environments. In 2014, attackers exfiltrated 56 million credit card numbers from Home Depot over a six-month period using DNS tunneling, demonstrating the real-world efficacy of this technique [21]. Similar techniques have appeared in APT campaigns [23,15,10,24,34], typically operating at low rates to avoid detection [35]. Attackers leverage the ubiquity of DNS queries to silently extract sensitive data from compromised networks. These channels are often minimally monitored compared to HTTP/HTTPS traffic due to their low throughput and necessary functionality [37]. The DNS protocol's widespread allowance through firewalls and its trust-by-default nature make it an attractive channel for covert communication.

Significant research has focused on defending against DNS exfiltration, spanning rule-based [19], anomaly-based [2,26], and machine learning (ML) methods [11,13]. However, a critical asymmetry exists: while detection methods are well-studied, minimal work examines how sophisticated adversaries develop evasion techniques. This creates a blind spot where defenders may not fully understand the threat landscape. Recent work found that most prior studies tested against default tool configurations, which produces overly optimistic detection rates that fail to generalize [12].

Prior adversarial approaches demonstrate the potential for learned evasion. DOLOS [18] uses Generative Adversarial Networks (GANs) to generate queries mimicking benign traffic, achieving 12% detection against ensemble defenses. However, such approaches rely on operational assumptions, including privileged access and extensive attacker infrastructure, that limit real-world applicability. More fundamentally, researchers lack standardized tools for systematic evaluation. Each study reinvents basic functionality: data sources, encoding schemes, and injection mechanisms. Synthetic attacks often violate protocol constraints or lack temporal realism [20], while circular validation limits generalization [12].

We present a modular framework for DNS exfiltration research that addresses these gaps. Our framework operates under a black-box threat model with user-level privileges only, requiring no knowledge of deployed defenses. Four components ensure semantic validity: a data generator producing realistic payloads, a victim selector identifying plausible compromised hosts from real network data, a time-based attack injector preserving temporal patterns, and a query validator enforcing RFC compliance. Researchers can extend any component while the framework ensures attacks remain semantically valid.

To validate the framework's modularity, we develop three adversarial generators that each exercise a distinct framework component: a reinforcement-learning strategy selector, an LSTM benign-mimicry timing model, and Stegatetra. Stegatetra emerged from exploring what the framework's encoding interface permits: rather than optimizing *how* to encode data in subdomain strings, it encodes data through *which* subdomains are selected from a fixed codebook of benign-looking entries. This design sidesteps per-query feature detection entirely, evading all evaluated academic detectors while maintaining practical throughput (8–54 KB/hr depending on domain count). These results highlight a gap in current detection approaches that focus on query-level features rather than transmission patterns.

In this work we make three contributions:

1. A modular research framework ensuring semantic validity and temporal realism for DNS exfiltration research, supporting both attack generation and systematic defense evaluation through pluggable interfaces (Section 5).[1]
2. Three adversarial generators exercising distinct framework components: an RL strategy selector, an LSTM timing model, and Stegatetra, a novel steganographic encoding (Section 6). We also develop ContraDNS, a contrastive

[1] Framework code: `https://github.com/Cwooper/dns-exf-framework`

learning detector built within the framework, to demonstrate its defensive utility.
3. Comprehensive cross-dataset evaluation against state-of-the-art detectors revealing that no single detection paradigm catches all attack types (Section 8).

## 2 Background

### 2.1 DNS Fundamentals

The Domain Name System (DNS) maps human-readable domain names to IP addresses through a distributed hierarchical database [30]. A client resolves a name such as `www.example.com` by issuing a query to a recursive resolver, which walks the hierarchy from the root zone to the TLD (`.com`) to the authoritative nameserver (ANS) for `example.com`. The ANS returns a resource record matching the requested type: `A` for IPv4 addresses, `AAAA` for IPv6, or `TXT` for arbitrary text. Resolvers cache responses for the record's time-to-live (TTL) to reduce load, so identical queries within that window never reach the authoritative server.

### 2.2 DNS Exfiltration Mechanics

Attackers choose DNS as an exfiltration channel because three conditions hold simultaneously. Outbound UDP/53 is almost universally permitted by firewalls, since blocking DNS disrupts normal network operation. Enterprise monitoring typically logs DNS at the metadata level (timestamp, source, query name, response code) rather than inspecting query contents the way HTTP proxies do. And every host on a network is expected to issue DNS queries constantly, so a compromised machine generating additional queries does not look out of place.

A typical exfiltration proceeds in three steps. *Encoding:* the malware converts payload bytes into DNS-label-safe characters (alphanumerics and hyphens) using a scheme such as base32, base64, or hex, respecting RFC 1035's 63-character label and 253-character FQDN length limits [30]. The string `Hello World!` becomes the base64-encoded label `SGVsbG8gV29ybGQh`. *Transmission:* the malware appends the label to an attacker-owned domain (`SGVsbG8gV29ybGQh.evil.com`) and issues a DNS query. Because the resolver has no cached answer for an unseen subdomain, it walks the hierarchy until it reaches the authoritative nameserver (ANS) for `evil.com`, which the attacker controls. *Decoding:* the ANS parses the subdomain label, reverses the encoding to recover the original bytes, and optionally returns command-and-control (C2) instructions in a `TXT` record or a crafted `A` record response.

Two parameters dominate the attacker's design space: *encoding* controls how many payload bytes fit into each query under the RFC 1035 limits, and *injection rate* controls how often queries are sent. Faster rates move more data but expose the attack to rate- and cardinality-based monitoring [28]. Most prior exfiltration tools embed payload bytes directly in subdomain strings. Iodine [16], one of the oldest and most widely used DNS tunneling tools, encodes IP-over-DNS

traffic using long base32/base64 subdomains at high query rates. It produces the signatures detectors target: high entropy, unusual lengths, and unbounded subdomain cardinality [11,28].

## 3 Threat Model

We specify the capabilities and constraints of the adversary our framework evaluates, building on the exfiltration mechanics introduced in Section 2.

**Attacker Capabilities.** We model a threat actor with user-level privileges on a compromised host. Unlike prior works that assume administrative or root access [18], we specifically constrain the attacker to user-mode execution. Consequently, the malware cannot open raw sockets to sniff the network interface, meaning it cannot observe real-time benign traffic rates to perfectly mimic the host's specific noise floor. The attacker operates "blindly" regarding the host's exact network behavior and must rely on pre-calculated transmission strategies or feedback received via the DNS command-and-control (C2) channel.

Despite this restriction, the attacker possesses significant capabilities:

- **Infrastructure Control:** The attacker maintains an Authoritative Nameserver (ANS) capable of parsing custom query structures and responding with C2 instructions (e.g., adjusting transmission rates via TXT records or specific IP responses).
- **Domain Reputation Management:** To bypass reputation-based filtering, we assume the attacker utilizes "aged" domains. Rather than registering new domains immediately prior to the attack, which triggers "newly observed domain" (NOD) alerts in enterprise Network Intrusion Detection Systems (NIDS) [36], the attacker leverages domains with established registration histories or compromised legitimate domains to blend in with trusted traffic.
- **Protocol Compliance:** The attacker can craft queries that strictly adhere to RFC standards [30](A, AAAA, TXT) to avoid parsing errors, while dynamically varying query types to evade simple frequency analysis defenses.

These capabilities are nonetheless bounded by detection pressure: beyond the RFC 1035 length limits noted in Section 2, anomalous query patterns such as high entropy, unusual lengths, or excessive query rates can expose the covert channel to monitoring systems.

## 4 Related Work

### 4.1 Adversarial Attack Generation

Prior work on adversarial DNS exfiltration remains limited. The most prominent recent example, DOLOS, utilizes GANs to automate the evasion of network classifiers [18]. Instead of using static encoding schemes like Base64, DOLOS

trains a generator to map sensitive data bits into domain strings that mimic the character distribution of benign traffic. This approach proved highly effective in simulations, achieving a detection rate of only 12% even when defenders deployed an ensemble of 6 out of 9 detection techniques.

However, DOLOS relies on operational assumptions that limit its real-world applicability. First, it requires root or administrator privileges to sniff the network interface and observe real-time benign traffic rates for dynamic rate tuning, a capability unavailable to standard user-mode exploits. Second, their evaluation assumes 120 distinct authoritative nameservers to dilute signal volume, contrasting with typical APT campaigns that maintain 5–15 aged domains to minimize forensic exposure [32]; one extensive operation employed 45 active domains [27].

Žiža et al. [40] demonstrated that classifiers trained on default tool configurations are highly brittle to parameter variation. They introduced an adversarial encoding scheme called `Base32map`, which maps Base32 characters to three-letter English words to mimic natural language distribution. For example: `1.carcutchiyoupen.webledshefin.dnsresearch.ml`. While this successfully evaded single-request entropy detectors, dropping detection rates to ≈1% for short queries, they found that aggregate statistical features (e.g., inter-request timing consistency) remained effective against these static modifications.

### 4.2 DNS Exfiltration Detection

**Rule-Based and Signature-Driven Detection** Rule-based intrusion detection systems (IDS) such as Zeek rely on handcrafted heuristics and protocol semantics to identify suspicious DNS behavior. We refer to one such extension as ZeekQ, following the terminology of Fahim et al. [18,19]. Common techniques include character frequency analysis, query length thresholds, and entropy-based rules that capture the structured randomness typical of tunneled payloads [8]. Early systems such as NgViz demonstrated the effectiveness of n-gram DNS character frequency analysis for identifying tunneling activity [9]. These approaches are lightweight and interpretable but often require careful tuning to balance false positives against evasion resistance.

**Anomaly Detection Approaches** Anomaly-based methods model benign DNS behavior and flag deviations that indicate covert channels or exfiltration. Flow-based techniques leverage network-level statistics and hypothesis testing to identify anomalous DNS usage patterns [17]. Statistical anomaly detection frameworks, including methods proposed by Ahmed and Nadler [2,26], emphasize unsupervised or semi-supervised learning to detect rare or evolving threats without requiring labeled attack data. Such approaches can be well-suited for uncovering novel tunneling techniques but may be sensitive to shifts in legitimate DNS usage.

**Cardinality- and Distribution-Based Methods** Cardinality-based techniques focus on properties such as the number of unique subdomains, query

diversity, and label reuse over time. Methods such as information-based Heavy Hitters (ibHH) [28] exploit deviations in expected DNS cardinality distributions to identify data exfiltration at scale. These techniques are particularly attractive for deployment at recursive DNS resolvers, where per-domain aggregation enables efficient real-time detection.

**ML-Based Classification** ML approaches model DNS tunneling as a classification problem using statistical fingerprints or learned representations of query behavior. Early ML-based systems demonstrated robust generalization across tunneling tools using carefully engineered features [3]. More recent work has explored supervised and deep learning methods, including LSTM-based sequence models for covert channel detection [13], deep neural classifiers for DNS tunneling discrimination [38], and hybrid feature-based systems designed for real-time operation [1].

**Reputation and Contextual Analysis** Complementary to traffic-focused techniques, reputation-based systems such as Notos and passive DNS analysis frameworks like EXPOSURE incorporate historical and contextual information to detect malicious domains [4,7]. While not specific to DNS exfiltration, these systems provide valuable domain-level context that can augment tunnel detection pipelines.

Despite progress in both detection and adversarial generation, researchers lack standardized tools for systematic evaluation. Each study reinvents basic functionality: data sources, encoding schemes, and injection mechanisms. More critically, synthetic attacks often lack semantic validity, violate RFC constraints, rely on simulated datasets, or use timing patterns inconsistent with real malware [20]. Cross-dataset evaluations reveal that models trained on one dataset frequently fail to generalize [12], suggesting that dataset artifacts rather than attack characteristics drive detection. Our framework addresses these gaps by providing modular, validated components for reproducible DNS exfiltration research.

## 5 Framework Design

We present a modular framework for DNS exfiltration research comprising four components (Figure 1). Each component is designed for extensibility: researchers can plug in custom generators, timing strategies, validators, or detectors while maintaining semantic validity. The framework supports both attack generation and systematic defense evaluation: it integrates external network intrusion detection systems through a thin adapter interface, which we use in Section 8 to evaluate ZeekQ, Buczak RF, ibHH, and Chen et al.'s LSTM detector against the generated attacks alongside our ContraDNS detector. The framework focuses on the query-side exfiltration channel. C2 response channels can be modeled through custom generator extensions.

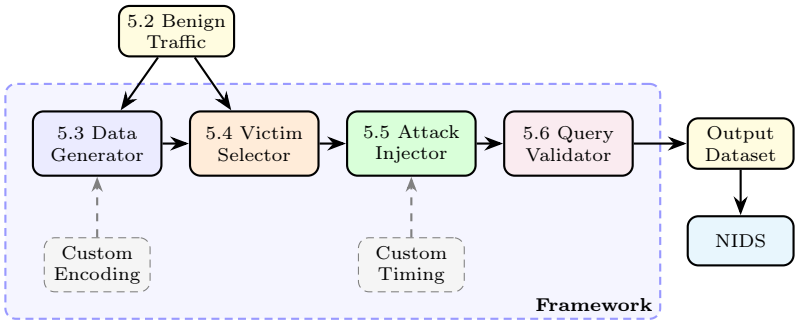


**Fig. 1.** Framework architecture. Solid arrows show data flow; dashed arrows indicate extensibility points for custom generators and timing strategies.

### 5.1 Design Principles

**Semantic Validity by Construction.** Every generated query passes RFC compliance checks and timing constraints before inclusion in the output dataset.

**Temporal Realism.** Attacks are injected into real network data at realistic timestamps, preserving the temporal patterns that time-aware detectors exploit and producing datasets that simulate realistic attack scenarios.

**Reproducibility.** All random operations use seeded generators. Given the same seed, victim selection, timing jitter, and attack sequences are deterministic.

**Extensibility.** Abstract interfaces allow new generators, timing controllers, and validators to integrate without modifying core framework code.

### 5.2 Data Sources

Our framework relies on a stream of benign DNS traffic to provide realistic temporal context for attack injection and ground truth for detector training. Any dataset of timestamped DNS queries can serve this role. We demonstrate the framework using two complementary datasets with distinct characteristics that together provide broader coverage of real-world network conditions.

**CIC-Bell-DNS-2021.** CIC-Bell-DNS-EXF-2021 [14,25] (hereafter CIC-Bell) provides labeled benign and malicious DNS traffic generated in a controlled lab environment. The benign portion spans 312 hours across 13 days, comprising traffic from a single client performing simulated browsing activity. The traffic patterns are simple with minimal CDN/cloud infrastructure, typical of older DNS datasets. The single-client nature makes CIC-Bell suitable for per-query feature analysis but unsuitable for studying per-client rate anomalies.

**PISCES.** Our primary data source is PISCES [29], a collection of real network flow data from a municipal network spanning March to June 2025. The dataset comprises approximately 2TB of Suricata logs from a multi-client environment. Unlike CIC-Bell, PISCES exhibits modern traffic patterns with heavy CDN and cloud usage (Azure, Office 365, Akamai), producing longer, higher-entropy subdomains.

Table 1 compares dataset query rates in queries per hour (q/hr). The inverted relationship is instructive: CIC-Bell's single client queries many domains infrequently (browsing behavior), while PISCES's multiple clients repeatedly query fewer shared domains (enterprise infrastructure). This makes PISCES more representative of networks where per-client and per-domain rate monitoring would be deployed.

**Table 1.** Dataset characteristics: CIC-Bell vs PISCES query rate profiles

| **Metric** | **CIC-Bell** | **PISCES** |
|---|---|---|
| Unique clients | 1 | 52 |
| Unique domains | 102,307 | 1,950 |
| *Per-client rates (q/hr)* | | |
| Median | 6,410 (single) | 10 |
| 95th percentile | — | 327 |
| *Per-domain rates (q/hr)* | | |
| Median | 2 | 3.4 |
| 99th percentile | 6 | 127 |
| Maximum | 62 | 868 |

PISCES provides a realistic foundation for attack simulation. We extract benign traffic patterns, identify suitable "victim" hosts, and inject attacks while maintaining temporal coherence with actual network behavior. PISCES also enables evaluation of *per-client rate anomalies*. This detection vector is unavailable with single-client datasets like CIC-Bell.

### 5.3 Component 1: Data Generator

The data generator produces realistic exfiltration payloads: credit card data, log files, images, and text. Following prior work [18], these represent common APT objectives spanning structured (credit cards, logs) and unstructured (images, text) formats with varying entropy profiles. This ensures detectors are evaluated against payloads matching real-world exfiltration targets rather than arbitrary test data. Optional compression increases throughput and tests detector sensitivity to high-entropy payloads [31].

**Optional Query Encoding.** For baseline comparisons, the generator can pre-encode payloads using standard schemes (Table 2). These encodings are all se-

mantically valid DNS subdomain formats drawn from patterns real exfiltration tools use in practice, spread intentionally across the feature space that detectors target (entropy, character distribution, subdomain length, and record type) so that adversarial generators have meaningful baselines to improve against. Adversarial generators typically implement custom encoding, validated downstream by the Query Validator.

**Table 2.** Baseline encoding strategies and their detection-relevant properties

| **Strategy** | **Description** | **Target Feature** |
|---|---|---|
| Base64 High Entropy | Standard base64 | Entropy |
| Hex High Numeric | Hexadecimal | Numeric ratio |
| Alphabetic | Base32, letters only | Low entropy |
| Short Subdomain | <12 characters | Length thresholds |
| Long Subdomain | 55+ characters | Length thresholds |
| TXT Record | Uses TXT type | Record type |

### 5.4 Component 2: Victim Selector

The victim selector is a simulation component controlled by the researcher. It identifies source IPs from the benign traffic dataset to use as "compromised" hosts. Using real IPs that already have DNS activity in the dataset, rather than synthetic IPs invented for the evaluation, keeps the simulation faithful to what a detector would see in production: a stateful NIDS that fingerprints clients over time has real behavioral history to reason about, and per-client features behave as they would against a genuine infection. Synthetic IPs would force the detector to handle a client with no history at all, which is an artifact of the simulation rather than a realistic scenario.

**Selection Criteria.** We filter hosts by query count to target realistic workstation infection targets. Very inactive hosts (fewer than 10 queries in the evaluation window) make unrealistic victims because in practice, the hosts that actually get infected are ones active enough to encounter infection vectors in the first place, so a near-silent host is a poor proxy for a real infection scenario. Very active hosts (more than 1000 queries) are typically servers, API gateways, or other infrastructure, which fall outside the user-level threat model of Section 3: user-level privileges are typically scoped to workstations rather than production servers, where access is more restricted. Workstation-range activity gives us victims that plausibly reflect who would actually be infected in the real world.

**Selection Modes.** The four selection modes span a progression from activity-blind to context-aware, with custom scoring as an extension point. Round-robin

is the activity-blind end of the progression: it deterministically cycles through candidates in index order, serving as the default for reproducibility. Weighted selection brings activity into the picture, biasing selection toward hosts with more DNS history to model the intuition that more-active hosts are likelier to encounter infection vectors. Adaptive selection adds temporal context, refining weighted selection by also preferring victims typically active during the attack's time window, reflecting the observed pattern that real exfiltration malware operates during user working hours. Custom selection sits outside the progression entirely and lets researchers plug in arbitrary scoring logic. A commercial research study simulating exfiltration from enterprise infrastructure, for instance, could bypass the default workstation filter to simulate infection of high-activity servers and API gateways, which would otherwise be excluded by the query-count thresholds above.

### 5.5 Component 3: Attack Injector

Prior work has often ignored temporal realism, injecting attacks without considering when queries would realistically occur. The attack injector controls the simulated timestamps at which attack queries appear in the traffic stream, merging them with benign traffic while preserving temporal patterns.

**Design Rationale.** Modern exfiltration malware runs in the background while users continue normal activity [5,32]. The attacker controls exfiltration rate independently of victim behavior. This *time-based* injection model contrasts with activity-proportional approaches that unrealistically couple attack rate to benign traffic [18].

**Timing Modes.** The three timing modes differ in how much they adapt to the benign traffic around them. Fixed intervals inject attack queries every $N$ seconds regardless of context: the simplest mode, useful as a baseline. Jittered intervals add $\pm 10\%$ random variation on top of the fixed rate to break the exact periodicity that simple inter-query-time detectors key on. Adaptive scaling is the only mode that reads the benign stream, scaling the injection rate with the traffic's time-of-day pattern to reflect the observed pattern that real exfiltration malware tracks the host's normal usage rhythm, with higher activity during working hours and lower activity overnight and on weekends.

### 5.6 Component 4: Query Validator

The framework's generators and injector are designed to produce valid queries by construction; the query validator is a defensive backstop that runs after them, catching edge cases and providing a safety net for custom user extensions whose internal invariants may not match the stock components' guarantees. Validation of this kind is often omitted when generating synthetic queries [20], contributing to datasets that contain queries no real resolver would accept.

**RFC 1035 compliance.** Queries must satisfy a total FQDN length of at most 253 characters, each label at most 63 characters, and a character set limited to alphanumeric and hyphen (no leading or trailing hyphen). Queries that violate these constraints fail to resolve in real networks: resolvers reject or truncate them before they reach the authoritative nameserver, which means a detector downstream would never see them either. Flagging them in the validator ensures our evaluation reflects traffic a detector would actually see rather than traffic that died in transit.

## 6 Adversarial Generators

We validate the framework's modularity through three adversarial generators that together exercise two of its component interfaces, query encoding and injection timing, without modification to core framework code. Two generators target the encoding interface at different abstraction levels, demonstrating that a single component can support qualitatively different research directions. The RL Strategy Selector hooks into the *query encoding* component at the strategy-selection level, learning which existing encoding strategy to apply for each payload chunk while delegating injection timing to the framework's default scheduler. The LSTM Benign-Mimicry Encoder hooks into the *injection timing* component, learning when to send queries based on observed traffic, while delegating encoding to a fixed benign-mimicry generator. Stegatetra hooks into the encoding component at the slot-selection level: rather than choosing among existing string encodings, it treats encoding as a combinatorial slot-selection problem, encoding data through which subdomains are queried rather than what they contain.

Neither the RL nor the LSTM generator is novel on its own. Both are standard methods plugged into the framework's interfaces to show that common ML approaches integrate cleanly. Stegatetra, by contrast, emerged as a novel encoding scheme during this exercise: the combinatorial slot-selection interface, once exposed, admitted a design that sidesteps per-query detection entirely.

All generators consume compressed payloads from our data generator. We observed no significant difference in detection rates between high-entropy (compressed) and low-entropy (raw) payloads across tested detectors. Given equivalent detectability, compression maximizes real-world throughput.

### 6.1 RL Strategy Selector

The RL generator hooks into the framework's query encoding component, learning which encoding strategy minimizes detection while maximizing throughput. We frame this as a Markov Decision Process: the agent observes its progress through the payload, queries generated so far, and estimated detection score, then selects from 40 predefined strategies. These strategies vary across encoding type (Base64, hex, Base32, alphabetic), subdomain length (11–80 characters), label count (1–6), and record type (A or TXT).

We train an Actor-Critic network using Proximal Policy Optimization (PPO) [33], rewarding low detection scores and high bytes-per-query throughput. While DOLOS uses adversarial training to generate encodings [18], our agent learns which existing *strategy* (Table 2) to use for each payload chunk. This ensures deterministic encode/decode functions and guaranteed payload recoverability. The RL generator uses fixed-interval injection timing (360 q/hr, or 1 query per 10 seconds) to isolate the encoding optimization problem. This rate is fast enough for efficient evaluation while remaining non-obvious to timing-based detection. After training, the agent converges on a mixture of three strategies: `alpha_25` (45% of queries), `hex_3label` (29%), and `hex_30` (26%). Training is whitebox: the agent receives reward signal from a single in-framework detector, while the four Table 4 detectors are held out and unseen. Per-strategy rates against these external detectors therefore equal the corresponding baseline rows, and the RL row reports the strategy-weighted transfer.

### 6.2 LSTM Benign-Mimicry Encoder

The LSTM generator hooks into the framework's injection component (Section 5.5), learning *when* to send queries by observing the benign traffic stream. Rather than optimizing encoding, this approach uses conventional low-entropy encodings but times injections to blend with legitimate activity. The system comprises two components: a timing model that consumes the framework's benign data stream, and a benign-mimicry encoder for query generation.

**Timing Model.** The timing model learns temporal patterns from a separate benign traffic dataset and predicts "high activity" windows suitable for blending attack queries. Features include query count, inter-query time, hour of day, day of week, query rate, unique domains, and average subdomain length. The model outputs a probability that the current window is suitable for injection. Queries are sent when this probability exceeds 0.6, a conservative margin above the sigmoid's natural 0.5 decision boundary chosen as a reasonable default rather than a tuned hyperparameter, biasing the injector toward higher-confidence windows and away from ambiguous periods.

**Benign-Mimicry Encoder.** Rather than using standard encodings that produce distinctive patterns, the encoder generates subdomains that resemble legitimate CDN and API traffic.

The encoder targets benign-like characteristics: Shannon entropy of 3.0–3.6 bits/character (matching the benign 95th percentile), digit ratio below 20% via digit-to-letter mapping, realistic structure with common prefixes (`cdn`, `api`, `static`) and hyphen-separated patterns, and subdomain length limited to 24 characters. The underlying encoding uses modified Base32 with deterministic digit-to-letter mapping, enabling payload recovery while maintaining benign-like character distributions.

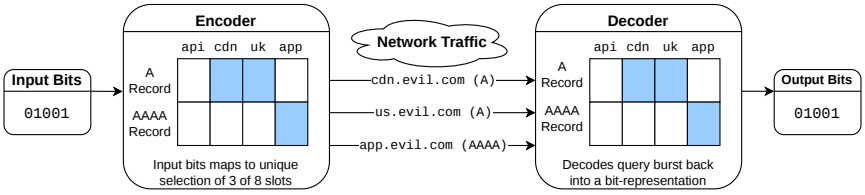


**Fig. 2.** Stegatetra toy example with 4 subdomains and 2 record types (A, AAAA), producing 8 query slots. The encoder maps a 5-bit input to a unique selection of 3 slots ($\binom{8}{3} = 56$ combinations); the decoder reconstructs the input by observing which slots received queries. With the paper's default parameters ($n$=800, $k \in [4, 8]$), each burst encodes 62 bits.

### 6.3 Stegatetra: Combinatorial DNS Steganography

Stegatetra takes a fundamentally different approach: rather than embedding data within subdomain strings, it encodes through *which* subdomains are selected from a fixed codebook. Traditional DNS exfiltration embeds data directly in subdomain strings (e.g., `a1b2c3d4.evil.com`), creating unique subdomains for each data fragment. This produces unbounded cardinality that HyperLogLog-based detectors identify [28], and high-entropy strings that trigger ML classifiers [11]. Stegatetra instead uses a fixed codebook of benign-looking subdomains, encoding information through combinatorial selection.

**Encoding Mechanism.** Consider a DNS query slot as a tuple $(d, u, r)$ where $d$ is an attacker-controlled domain, $u$ is a subdomain from codebook $\mathcal{U}$, and $r$ is the record type (A or AAAA). With $D$ domains, $|\mathcal{U}| = 400$ subdomains, and 2 record types, the total slot space is $n = 800D$ slots. We use 400 subdomains to stay below cardinality detection thresholds; A and AAAA record types double the slot space without triggering NIDS rules that flag unusual record types.

Rather than encoding data in subdomain strings, we encode by selecting *which subset* of slots to query. For variable burst sizes $k \in [k_{\min}, k_{\max}]$, the encoding capacity is:

$$\text{bits} = \left\lfloor \log_2 \sum_{k=k_{\min}}^{k_{\max}} \binom{n}{k} \right\rfloor \tag{1}$$

With defaults $n = 800$ (single domain), $k_{\min} = 4$, $k_{\max} = 8$, balancing encoding capacity against burst visibility: 62 bits per burst. The ANS decodes by observing burst size $k$, which domains received queries, the subdomains queried, and the record types used.

**Low-Entropy Codebook Design.** Our codebook $\mathcal{U}$ is designed to evade FQDN-structure-based detection:

- **2–5 characters**: Matches common infrastructure prefixes
- **Alphabetic only**: No digits, hyphens, or unusual characters
- **Common patterns**: Subdomains and country codes (e.g., `api`, `cdn`, `uk`)

This neutralizes detection based on subdomain length, character distribution, Shannon entropy, and lexical features. The key insight: *any detector using the FQDN as input* sees only benign-looking queries. The exfiltrated data exists solely in the pattern of which subdomains appear.

**TTL Management and Slot Scheduling.** DNS caching can introduce a critical constraint: reusing a slot before its TTL expires causes the resolver to return a cached response, losing data. We conservatively assume $\tau = 360$ seconds TTL.

Stegatetra's scheduler explicitly tracks slot availability to guarantee zero collisions. For each slot $s$, we maintain $t_{\text{last}}(s)$. The encoder selects only from available slots:

$$S_{\text{avail}}(t) = \{s : t - t_{\text{last}}(s) \geq \tau \text{ or } s \text{ unused}\} \tag{2}$$

This guarantees $P(\text{collision}) = 0$ by construction, not probability.

The scheduler faces one edge case and one optimization. With fewer than four domains, the encoder can exhaust the available slot pool before any slots refresh, a condition we call *slot starvation*: the encoder must wait for $t_{\text{last}}(s)$ values to age past $\tau$ before it can assemble the next burst, which degrades throughput sharply at low domain counts. The optimization is to schedule waits deliberately rather than greedily. A naive strategy sends a burst as soon as $|S_{\text{avail}}| \geq k_{\min}$, but waiting for slots to accumulate into a larger pool yields higher-capacity bursts and improves throughput by 2–3× for 1–3 domains. With a wait threshold of 50–100 slots, the scheduler batches refreshed slots into denser bursts. With ≥4 domains, slot starvation does not occur and greedy scheduling is near-optimal.

**Domain Scaling.** Adding domains provides superlinear throughput gains (Table 3) from two multiplicative effects. Linearly, $D$ domains at rate $R$ queries per hour yield $D \times R$ total query capacity. Logarithmically, each domain contributes 800 slots, so with $n = 800D$ total slots and $\binom{n}{k} \approx n^k/k!$ for $k \ll n$, the bits per burst $B(D)$ grow as

$$B(D) \approx k \cdot \log_2(800D) - \log_2(k!). \tag{3}$$

Combined with the linear query capacity, Stegatetra's aggregate throughput scales as $\mathcal{O}(D \log D)$.

**Detection Evasion Design.** Stegatetra is engineered to evade the three major detection paradigms:

*Per-query feature analysis:* Detectors that analyze individual query features (entropy, character distribution, subdomain length, or character-level sequences)

see only benign-looking patterns. Our 2–5 character alphabetic codebook produces low entropy, no digits, and common infrastructure-like patterns. Any detector using the FQDN as input, whether through engineered features or learned sequence models, cannot distinguish Stegatetra queries from legitimate traffic.

*Rule-based thresholds:* Short alphabetic subdomains satisfy standard length and character rules. Rate limiting to 500 q/hr/domain (∼8 unique queries per minute) stays below typical cardinality thresholds that flag high-volume tunneling.

*Information-theoretic analysis:* Short alphabetic subdomains carry minimal unique information per query. Even aggregated over time windows, the information rate remains low because the same 400 subdomains are reused across bursts.

We validate these choices in Section 8.

**Table 3.** Stegatetra throughput scales with domain count (500 q/hr/domain, 100KB payload)

| Domains | KB/hr | Time | B/s |
|---|---|---|---|
| 1 | 0.6 | 161.0h | 0.18 |
| 3 | 2.2 | 45.0h | 0.64 |
| 8 | 6.9 | 14.5h | 1.97 |
| 10 | 8.9 | 11.3h | 2.55 |
| 50 | 53.9 | 1.9h | 15.48 |

## 7 Detection Methods

### 7.1 State-of-the-Art Detectors

We evaluate against four detection approaches spanning multiple paradigms. Each detector was selected to represent a distinct detection methodology: rule-based heuristics, supervised feature-based learning, information-theoretic analysis, and supervised sequence learning.

- **ZeekQ** [19]: Rule-based with stateful cardinality tracking. Selected as representative of deployed production systems using heuristic thresholds.
- **Buczak RF** [11]: Supervised Random Forest adapted for stateless per-query detection using 8 non-windowed features (query length, entropy, character ratios, label statistics) from the original 52-feature PCAP-level detector—the same adaptation used by DOLOS [18]. Selected as representative of supervised feature-based classification.
- **ibHH** [28]: Information-theoretic detector using HyperLogLog cardinality estimation. Selected as representative of stateful approaches that track query patterns over time windows.

– **Chen et al. LSTM** [13]: Character-level LSTM operating on raw FQDN strings, trained on benign and malicious DNS queries. Selected as representative of supervised deep sequence models for per-query detection.

### 7.2 ContraDNS: Contrastive DNS Anomaly Detection

To demonstrate this framework's utility for developing and evaluating new exfiltration detection methods, we present ContraDNS. ContraDNS is a new unsupervised anomaly detection method using contrastive learning. The model trains exclusively on benign DNS queries and detects attacks as out-of-distribution samples at inference time. We evaluate ContraDNS on queries from Iodine and the RL, LSTM, and Stegatetra generators.

**Architecture.** A multi-scale character-level CNN encodes DNS queries into 256-dimensional embeddings. Four parallel convolutional layers with kernel sizes $k \in \{2, 3, 4, 5\}$ capture character n-grams at different scales, followed by max-pooling and concatenation. During training, a projection head maps embeddings to a 128-dimensional contrastive space where we apply NT-Xent loss [39].

**DNS-Specific Augmentations.** We generate positive pairs using domain-aware augmentations: case changes, subdomain label shuffling, character dropout, visually-similar character substitution, and TLD swapping.

**Anomaly Detection.** At inference, we compute the centroid of benign training embeddings and score queries by Euclidean distance. Empirically, attacks cluster *closer* to the centroid than benign queries because systematic encoding schemes produce tightly-clustered patterns, while legitimate traffic is more diverse. Queries scoring *below* the threshold are classified as attacks.

## 8 Evaluation

### 8.1 Experimental Setup

**Datasets.** We evaluate on two datasets: (1) PISCES [29] municipal network data (2TB, March to June 2025), and (2) CIC-Bell [14]. Detectors are trained exclusively on benign traffic from each dataset; evaluation injects attacks into a separate benign test stream. We generate attack traffic using each generator with a 100KB test payload.

Small payload sizes are more challenging for detection, producing shorter and less distinctive query patterns. We use 100KB as a compromise between realism and evaluation speed, representative of highly compressible data such as credit card records.

**Metrics.** We report detection rate (percentage of attack queries flagged), false positive rate (FPR) on benign traffic, throughput (KB/hr), and AUC-ROC for ContraDNS.

**Baseline Validation.** To verify detector implementations, we first test against Iodine-style DNS tunneling [16], an obvious attack with long base32-encoded subdomains (100–200 characters) and high entropy.

### 8.2 Detector Validation: Iodine Baseline

We validated detector implementations against Iodine-style tunneling at its native 10 q/s rate (1,707 queries for 100KB). ZeekQ, Buczak RF, and ibHH achieve 100% detection. Chen et al.'s LSTM achieves 84.2%, confirming correct implementation across all four detectors. We exclude Ahmed et al.'s Isolation Forest [2] from PISCES evaluation: when tuned on PISCES traffic, the model could not achieve below 10% FPR while maintaining Iodine detection. However, when trained on CIC-Bell's simpler traffic patterns, Ahmed et al.'s approach achieves acceptable performance (Section 8.6).

### 8.3 Generator vs Detector Matrix

Table 4 shows detection rates for each generator against all detectors, including an ensemble (logical OR of all detectors). We include Iodine as a detectable baseline. Times shown are for exfiltrating a 100KB payload.

**Key Findings.** Both Buczak and ibHH consistently detect obvious exfiltrations. Stegatetra achieves 0% detection against all four evaluated academic detectors while still achieving 0.6–53.9 KB/hr throughput depending on domain count. The ensemble of all detectors fails to catch rate-limited Stegatetra at any configuration. Buczak's feature-based classifier catches high-entropy encodings but misses low-entropy and steganographic approaches due to its stateless design. The critical finding is that attackers can tune Stegatetra's query rate to evade ibHH while maintaining throughput competitive with DOLOS (5–10 KB/hr at 12% detection). None of the evaluated academic detectors model per-client behavioral baselines, which we identify as a plausible but untested detection vector and discuss further in Section 9. All detectors maintain acceptable false positive rates on benign PISCES traffic: ZeekQ 0%, Buczak 1.5%, ibHH 2%, Chen 0.07%.

### 8.4 Throughput Analysis

Throughput and detection risk trade off against each other. Higher query rates move more data but are easier for stateful detectors to flag. Table 5 shows each generator's position on this tradeoff. The subsections then examine the rate-limiting, timing, and per-client factors that determine these positions.

**Table 4.** Detection rates (%) for 100KB payload exfiltration. Ensemble shows the logical OR of all four tested detectors. Domain counts abbreviated as *d*.

| Generator | Queries | KB/hr | ZeekQ | Buczak | ibHH | Chen | Ensemble |
|---|---|---|---|---|---|---|---|
| *Baseline DNS Tunnel* | | | | | | | |
| Iodine | 1,707 | 2110.2 | 100% | 100% | 100% | 84.2% | 100% |
| *Baseline Encoders (360 q/hr)* | | | | | | | |
| Base64 | 3,414 | 10.6 | 0% | 100% | 91.7% | 100% | 100% |
| Hex | 4,096 | 8.8 | 100% | 100% | 91.7% | 1.8% | 100% |
| Alphabetic | 5,120 | 7.0 | 0% | 91.9% | 91.4% | 97.0% | 99.9% |
| Short subdomain | 12,800 | 2.8 | 0% | 24.5% | 74.9% | 23.4% | 100% |
| *Adversarial Generators* | | | | | | | |
| RL Selector | 6,926 | 14.4 | 54.7% | 54.7% | 85.4% | 54.7% | 85.4% |
| LSTM mimicry | 10,240 | 0.6 | 0% | 0.2% | 0% | 0.1% | 0.2% |
| *Stegatetra (500 q/hr/domain)* | | | | | | | |
| Stegatetra (1d) | 80,494 | 0.6 | 0% | 0% | 0% | 0% | 0% |
| Stegatetra (3d) | 67,549 | 2.2 | 0% | 0% | 0% | 0% | 0% |
| Stegatetra (8d) | 57,850 | 6.9 | 0% | 0% | 0% | 0% | 0% |
| Stegatetra (10d) | 56,443 | 8.9 | 0% | 0% | 0% | 0% | 0% |
| Stegatetra (50d) | 46,378 | 53.9 | 0% | 0% | 0% | 0% | 0% |

**Stegatetra vs DOLOS.** Stegatetra achieves comparable or higher throughput than DOLOS (8.9 KB/hr at 10 domains, 53.9 KB/hr at 50 domains) while achieving 0% detection across our evaluated detectors versus DOLOS's reported 12%, without requiring adversarial training or privileged access.

**Rate Selection.** Pushing query rates toward the upper percentiles of client activity introduces detection from cardinality-based detectors such as ibHH. For example, Stegatetra's 500 q/hr/domain rate stays within typical CDN traffic patterns, balancing throughput against this risk.

**Client-Level Detection.** While ibHH and similar detectors track information per *domain*, defenders could also monitor per-client query rates. An attacker using 50 domains at 500 q/hr/domain generates 25,000 q/hr from a single client, roughly two orders of magnitude above PISCES's 95th-percentile client rate of 327 q/hr, and well above typical workstation behavior. Client-level rate limits (e.g., 500–1000 q/hr/client) could serve as a complementary detection layer, though attackers can counter them by distributing exfiltration across multiple compromised clients to reduce per-client rates while maintaining aggregate throughput. Stegatetra with 50 domains distributed across 10 clients yields ∼54 KB/hr aggregate while holding each client at 500 q/hr, within PISCES's 95th percentile.

**Table 5.** Throughput vs detection trade-off for 100KB exfiltration. Detection is the ensemble rate (logical OR of all four tested detectors), with DOLOS's value as reported.

| Generator | KB/hr | Time | Detection | Evades? |
|---|---|---|---|---|
| Iodine (10 q/sec) | 2,110 | 3 min | 100% | No |
| Base64 (1 q/min) | 1.8 | 56.9h | 100% | No |
| Alphabetic (1 q/min) | 1.2 | 85.3h | 99.9% | No |
| LSTM encoder (1 q/min) | 0.6 | 170.7h | 0.2% | Partial |
| DOLOS (reported) | 5–10 | 10–20h | 12% | Partial |
| Stegatetra (1 dom.) | 0.6 | 161.0h | **0%** | **Yes** |
| Stegatetra (10 dom.) | 8.9 | 11.3h | **0%** | **Yes** |
| Stegatetra (50 dom.) | 53.9 | 1.9h | **0%** | **Yes** |

**LSTM Encoder.** At the conservative 1 q/min default, the LSTM benign-mimicry encoder achieves 0.2% ensemble detection with 0.6 KB/hr throughput. The low throughput reflects the conservative timing. The injector module supports higher rates if the attacker accepts increased detection risk.

### 8.5 ContraDNS Results

We evaluate ContraDNS on queries from the Iodine, RL, LSTM, and Stegatetra generators. The model trains on 10,000 benign queries with no exposure to attack patterns.

**Table 6.** ContraDNS detection performance across attack types. $N$ is the number of attack queries evaluated; R@$x$% is recall at $x$% false positive rate.

| Attack | $N$ | AUC-ROC | R@1% | R@5% | R@10% |
|---|---|---|---|---|---|
| Iodine | 200 | 0.994 | 80.0% | 100.0% | 100.0% |
| RL (Base32) | 91 | 0.999 | 100.0% | 100.0% | 100.0% |
| RL (Hex) | 62 | 0.988 | 54.8% | 98.4% | 100.0% |
| LSTM | 209,716 | 0.978 | 60.9% | 93.3% | 96.3% |
| Stegatetra | 13,062 | 0.942 | 11.0% | 63.9% | 81.7% |

Detection difficulty correlates with attack sophistication. RL encoding attacks produce distinctive character patterns easily separated from benign traffic (AUC 0.999). LSTM benign-mimicry attacks, despite learning benign character distributions, remain detectable (AUC 0.978). Stegatetra presents the greatest challenge (AUC 0.942, 11% recall at 1% FPR) because it uses real subdomain patterns from legitimate traffic rather than encoding data in subdomain strings. Iodine-style DNS tunneling, characterized by very long base32/base64 encoded subdomains (averaging 180 characters), achieves high detectability (AUC 0.994) with 100% recall at just 5% FPR, demonstrating that traditional DNS tunneling tools remain vulnerable to embedding-based detection.

### 8.6 CIC-Bell Validation

To confirm these findings generalize beyond PISCES, we trained detectors on CIC-Bell [14] benign traffic and evaluated against our attack generators. Table 7 summarizes detection rates. We denote CIC-Bell-trained detectors with the "-CIC" suffix (e.g., Ahmed-CIC). We omit Buczak RF and Chen et al.'s LSTM detector because the CIC-Bell-DNS-EXF malicious portion provides only pre-computed statistical features rather than raw FQDNs, preventing both Buczak's feature extraction and Chen's character-level input.

**Table 7.** Detection rates (%) for CIC-Bell-trained detectors

| **Attack** | **ZeekQ** | **Ahmed-CIC** | **ibHH-CIC** |
|---|---|---|---|
| Iodine | 100% | 100% | 99.9% |
| Base64 | 0% | 100% | 83.3% |
| Hex | 29.2% | 100% | 75.0% |
| Alphabetic | 0% | 0% | 58.3% |
| LSTM Mimicry | 0% | 0% | 0% |
| Stegatetra (1–50 domains) | 0% | 0% | 0% |
| CIC-Bell-DNS-EXF (baseline) | 0% | 0.2% | 94.6% |

Results confirm cross-dataset evasion. Stegatetra achieves 0% detection across all CIC-trained detectors at all tested domain counts (1, 10, and 50). LSTM mimicry similarly evades all CIC-trained detectors, owing to its benign-looking subdomains and conservative throughput. Ahmed-CIC detects high-entropy encodings (Base64, Hex) at 100% but misses low-entropy attacks entirely, suggesting it learned CIC-Bell's characteristically low subdomain entropy as a baseline. Despite ibHH-CIC's threshold being tuned for CIC-Bell's sparse traffic patterns, Stegatetra evades detection because its short alphabetic subdomains (2–5 characters) accumulate insufficient unique information within the 120-second window. FPR on CIC-Bell benign traffic: ZeekQ 1.12%, Ahmed-CIC 2.7%, ibHH-CIC 0%. The degraded performance on the CIC-Bell-DNS-EXF baseline can be explained by the low subdomain lengths within that dataset (mean: 8 chars).

## 9 Discussion

### 9.1 Implications for Defenders

Our results reveal a fundamental tension in DNS exfiltration defense: no single detection paradigm catches all attack types:

**Defense in Depth Is Essential.** Rule-based detectors (ZeekQ) excel at catching high-volume, obvious tunneling but miss low-entropy encodings entirely.

Feature-based ML classifiers (Buczak RF) catch encoding-based attacks with distinctive feature signatures but fail against steganographic approaches that use benign-looking queries. Sequence-based deep models (Chen et al.) detect character-level encoding patterns but similarly fail when attacks reuse benign-looking subdomains. Cardinality-based detectors can be evaded through domain parallelization. Effective defense requires layering multiple detection paradigms.

**Adversarial Awareness Matters.** Detectors trained or tuned against default tool configurations (e.g., Iodine) may provide false confidence. Our framework enables defenders to test against adversarially-optimized attacks before deployment, revealing blind spots that would otherwise remain hidden until exploitation.

**The Attacker's Advantage.** Stegatetra demonstrates that determined adversaries can achieve high throughput (∼54 KB/hr) while evading all tested detectors. This sobering result suggests that purely reactive detection may be insufficient. Complementary strategies such as DNS allowlisting [28], encrypted DNS monitoring, or endpoint-based detection warrant further consideration.

### 9.2 Per-Client Rate Detection: An Underexplored Vector

Our evaluation reveals that per-client query rate to specific domains remains underexplored as a detection vector. Evaluated detectors focus on per-query features (entropy, length, character distribution) or aggregate metrics (total cardinality, network-wide information rate), but none explicitly model per-client behavioral baselines. However, real-world campaigns have distributed exfiltration across many hosts — Home Depot's attackers spread traffic across thousands of individual point-of-sale terminals [21] — reducing per-client anomaly signals.

In PISCES, the median client generates only 10 q/hr total, with 95th percentile at 327 q/hr. A compromised client suddenly generating 100+ q/hr to a single previously-unseen domain would represent a 10× anomaly against its baseline—highly detectable if monitored. Even Stegatetra's distributed approach (100 q/hr across 10 domains) produces 10 q/hr/domain from a single client, which may be anomalous for domains that client has never queried.

Per-client monitoring requires stateful tracking of client-domain pairs, increasing memory requirements. Additionally, legitimate behavioral changes (new software, changed browsing habits) could trigger false positives. However, combining per-client rate anomalies with domain reputation (age, popularity, TLD) could yield a powerful detection signal that current approaches miss.

**Enterprise Scaling Considerations.** PISCES represents a modest-sized network (∼52 clients). Enterprise deployments with hundreds or thousands of clients would see proportionally higher per-domain rates while per-client baselines remain stable. Per-client rates likely stay consistent across network sizes (median

10–50 q/hr, power users 300–1,000 q/hr), while per-domain rates scale linearly with client count (microsoft.com might see 10,000–100,000 q/hr on large networks).

This scaling behavior means per-domain thresholds must be calibrated per-network, while per-client thresholds could potentially generalize. An exfiltration client generating 500 q/hr to a single C2 domain would be anomalous on *any* network, regardless of size.

### 9.3 Limitations

**Encrypted DNS (DoH/DoT) and DNSSEC.** Our framework and evaluation focus on traditional plaintext DNS without DNSSEC validation. Encrypted DNS protocols (DoH, DoT) prevent network-based inspection of query content but typically route through known resolvers that could implement server-side detection [22]. DNSSEC operates orthogonally: by signing authoritative responses rather than client queries, it does not constrain the query-side covert channels evaluated here, though it expands the response-side surface (e.g., embedding data in DNSKEY or RRSIG records) in ways our query-side detectors cannot observe. As encrypted and signed DNS adoption grows, resolver operators are best positioned to combine the query-side features evaluated here with protocol-aware signals available only at the resolver layer.

**Network Environment Scope.** PISCES is collected from a single municipal network with approximately 52 active clients during our evaluation window. Enterprise and ISP environments differ from this baseline in client population, aggregate query volume, and the diversity of legitimate DNS behavior, any of which can affect per-domain and per-client detection thresholds. We do not claim that PISCES generalizes to every operational network. PISCES is, however, among the first non-synthetic datasets used for DNS exfiltration evaluation and represents a step beyond the synthetic-only status quo of CIC-Bell and similar releases.

**Detector Coverage.** We evaluate four detection approaches spanning rule-based heuristics, supervised feature-based learning, information-theoretic analysis, and supervised sequence learning, but the landscape of DNS security tools is vast. Commercial solutions implement proprietary detection logic that licensing constraints prevent us from evaluating directly; the framework's detector adapter interface accommodates such systems and would support their integration if access were available.

### 9.4 Ethical Considerations

This work develops tools that could facilitate data exfiltration. The benefits to defenders outweigh the risks for several reasons:

**Offensive Techniques Are Already Known.** Stegatetra's core method of encoding in query selection rather than content is not novel in the broader steganography literature [6]. DNS tunneling tools have been publicly available for over a decade. Our contribution is systematizing these techniques to support defensive research.

**Defenders Need Realistic Adversaries.** Security research suffers when defenders only test against naive attacks. By providing a framework for generating sophisticated, semantically-valid attacks, we enable more rigorous detector evaluation before deployment.

### 9.5 Future Work

**Encrypted DNS.** Extending the framework to model exfiltration over DoH or DoT, including server-side detection strategies available to resolver operators. As encrypted DNS adoption grows, traditional network-based inspection becomes impossible. However, resolver operators could implement server-side detection.

**Adaptive Adversaries.** Implementing online learning where generators can adapt in real-time to detector feedback, modeling a white or gray box NIDS. Real-world attackers often probe defenses before deciding on exfiltration strategies. Modeling this adaptive behavior would reveal whether current detectors remain robust under active evasion attempts.

**Endpoint Correlation.** Exploring hybrid detection that combines DNS monitoring with endpoint telemetry. Stegatetra demonstrates that network-only detection has fundamental limits. Correlating queries with process-level data (e.g., which application initiated the query) could address blind spots that network-based approaches miss.

**Large-Scale Evaluation.** Systematic evaluation across a broader range of NIDS implementations, adversarial tools, and datasets. Our evaluation covers major detection paradigms but not exhaustive implementations. Testing across commercial tools and diverse network environments would validate whether our findings generalize beyond academic detectors.

## 10 Conclusion

DNS exfiltration remains a persistent real-world threat: attackers have used DNS tunneling to exfiltrate millions of records over extended periods while evading detection. Yet research progress has been hampered by the lack of standardized tools for generating semantically-valid attacks and systematically evaluating defenses. This paper addresses this gap with three contributions.

First, we present a modular framework for DNS exfiltration research that maintains RFC compliance, temporal realism, and behavioral coherence with real network traffic. The framework provides researchers with building blocks for reproducible experimentation (data generation, victim selection, time-based injection, and query validation) and supports both attack generation and systematic defense evaluation through pluggable interfaces.

Second, we validate the framework's modularity through three adversarial generators that each exercise a distinct framework component. Stegatetra, which emerged from exploring what the framework's encoding interface permits, evades all evaluated academic detectors at competitive throughput (∼54 KB/hr at 50 domains), highlighting gaps in detection approaches that focus on query-level features rather than transmission patterns. We also develop ContraDNS, a contrastive learning detector built within the framework, as a demonstration of its defensive utility; it achieves strong detection on encoding-based attacks while revealing that steganographic approaches remain challenging even for learned representations.

Third, our cross-dataset evaluation reveals that no single detection paradigm catches all attack types. Rule-based detectors miss low-entropy encodings; ML classifiers, whether feature-based or sequence-based, miss steganographic approaches; cardinality-based detectors can be evaded through rate limiting. Detectors trained on one dataset's traffic patterns may fail to generalize, underscoring the importance of diverse evaluation.

These findings have practical implications: organizations relying on a single DNS monitoring solution may have significant blind spots. Defense in depth, combining multiple detection paradigms with complementary coverage, is essential. We present our framework to enable the research community to develop and evaluate the next generation of DNS exfiltration defenses against realistic, sophisticated adversaries.

**Acknowledgments.** This work was supported by the Naval Engineering Education Consortium (NEEC; grant 14290636). We would also like to thank Michael Tsikerdekis for sharing his expertise on DNS exfiltration and for the discussions that helped shape this work.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Abualghanam, O., Alazzam, H., Elshqeirat, B., Qatawneh, M., Almaiah, M.A.: Real-Time Detection System for Data Exfiltration over DNS Tunneling Using Machine Learning. Electronics **12**(6), 1467 (Mar 2023). https://doi.org/10.3390/electronics12061467, https://www.mdpi.com/2079-9292/12/6/1467
2. Ahmed, J., Habibi Gharakheili, H., Raza, Q., Russell, C., Sivaraman, V.: Monitoring Enterprise DNS Queries for Detecting Data Exfiltration From Internal

Hosts. IEEE Transactions on Network and Service Management **17**(1), 265–279 (Mar 2020). https://doi.org/10.1109/TNSM.2019.2940735, https://ieeexplore.ieee.org/document/8832271/
3. Aiello, M., Mongelli, M., Papaleo, G.: DNS tunneling detection through statistical fingerprints of protocol messages and machine learning. International Journal of Communication Systems **28**(14), 1987–2002 (Sep 2015). https://doi.org/10.1002/dac.2836, https://onlinelibrary.wiley.com/doi/10.1002/dac.2836
4. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for DNS. In: Proceedings of the 19th USENIX Security Symposium. pp. 273–288. USENIX Association, Washington, DC, USA (2010)
5. APT Security: Attack Stages & 6 Ways to Secure Your Network (Jan 2026), https://www.cynet.com/advanced-persistent-threat-apt-attacks/apt-security-warning-signs-and-6-ways-to-secure-your-network/
6. Bellovin, S.M.: Compression, correction, confidentiality, and comprehension: a modern look at telegraph codebooks. Cryptologia pp. 1–37 (Mar 2025). https://doi.org/10.1080/01611194.2025.2457084, https://www.tandfonline.com/doi/full/10.1080/01611194.2025.2457084
7. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. ACM Transactions on Information and System Security **16**(4), 1–28 (Apr 2014). https://doi.org/10.1145/2584679, https://dl.acm.org/doi/10.1145/2584679
8. Born, K., Gustafson, D.: Detecting DNS Tunnels Using Character Frequency Analysis (Apr 2010). https://doi.org/10.48550/arXiv.1004.4358, http://arxiv.org/abs/1004.4358, arXiv:1004.4358
9. Born, K., Gustafson, D.: NgViz: detecting DNS tunnels through n-gram visualization and quantitative analysis. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. pp. 1–4. ACM, Oak Ridge Tennessee USA (Apr 2010). https://doi.org/10.1145/1852666.1852718, https://dl.acm.org/doi/10.1145/1852666.1852718
10. Bromiley, M., Klapprodt, N., Schroeder, N., Rocchio, J.: Hard Pass: Declining APT34's Invite to Join Their Professional Network | Mandiant (Jul 2019), https://cloud.google.com/blog/topics/threat-intelligence/hard-pass-declining-apt34-invite-to-join-their-professional-network
11. Buczak, A.L., Hanke, P.A., Cancro, G.J., Toma, M.K., Watkins, L.A., Chavis, J.S.: Detection of Tunnels in PCAP Data by Random Forests. In: Proceedings of the 11th Annual Cyber and Information Security Research Conference. pp. 1–4. ACM, Oak Ridge TN USA (Apr 2016). https://doi.org/10.1145/2897795.2897804, https://dl.acm.org/doi/10.1145/2897795.2897804
12. Cantone, M., Marrocco, C., Bria, A.: On the Cross-Dataset Generalization of Machine Learning for Network Intrusion Detection. IEEE Access **12**, 144489–144508 (2024). https://doi.org/10.1109/ACCESS.2024.3472907, http://arxiv.org/abs/2402.10974, arXiv:2402.10974 [cs]
13. Chen, S., Lang, B., Liu, H., Li, D., Gao, C.: DNS covert channel detection method using the LSTM model. Computers & Security **104**, 102095 (May 2021). https://doi.org/10.1016/j.cose.2020.102095, https://linkinghub.elsevier.com/retrieve/pii/S0167404820303680
14. CIC-Bell-DNS-EXF 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB, https://www.unb.ca/cic/datasets/dns-exf-2021.html
15. Dahan, A.: Operation Cobalt Kitty: A large-scale APT in Asia carried out by the OceanLotus Group, https://www.cybereason.com/blog/operation-cobalt-kitty-apt

16. Ekman, E.: iodine: Ip over dns tunneling solution. https://github.com/yarrick/iodine (2006), accessed: 2026-01-18
17. Ellens, W., Żuraniewski, P., Sperotto, A., Schotanus, H., Mandjes, M., Meeuwissen, E.: Flow-Based Detection of DNS Tunnels. In: Doyen, G., Waldburger, M., Čeleda, P., Sperotto, A., Stiller, B. (eds.) Emerging Management Mechanisms for the Future Internet. pp. 124–135. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38998-6_16
18. Fahim, A., Zhu, S., Qian, Z., Song, C., Papalexakis, E., Chakraborty, S., Chan, K., Yu, P., Jaeger, T., Krishnamurthy, S.V.: DNS Exfiltration Guided by Generative Adversarial Networks. In: 2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P). pp. 580–599. IEEE, Vienna, Austria (Jul 2024). https://doi.org/10.1109/EuroSP60621.2024.00038, https://ieeexplore.ieee.org/document/10628994/
19. hhzzk: dns-tunnels. https://github.com/hhzzk/dns-tunnels (2017), accessed: 2026-01-18
20. Jin, M., Apostolaki, M.: Robustifying ML-powered Network Classifiers with PANTS (Feb 2025). https://doi.org/10.48550/arXiv.2409.04691, http://arxiv.org/abs/2409.04691, arXiv:2409.04691 [cs]
21. Johnston, C.: Home Depot: 56 million credit cards compromised. The Guardian (Sep 2014), https://www.theguardian.com/business/2014/sep/19/home-depot-56m-credit-card-numbers-compromised
22. Kipp, J.: Using the power of Cloudflare's global network to detect malicious domains using machine learning (Mar 2023), https://blog.cloudflare.com/threat-detection-machine-learning-models/
23. Kremez, V.: FIN6 "FrameworkPOS": Point-of-Sale Malware Analysis & Internals - SentinelLabs (Sep 2019), https://www.sentinelone.com/labs/fin6-frameworkpos-point-of-sale-malware-analysis-internals/
24. Lee, Bryan, R.F.: DarkHydrus delivers new Trojan that can use Google Drive for C2 communications (Jan 2019), https://unit42.paloaltonetworks.com/darkhydrus-delivers-new-trojan-that-can-use-google-drive-for-c2-communications/
25. Mahdavifar, S., Salem, A.H., Victor, P., Garzon, M., Razavi, A.H., Hellberg, N., Habibi Lashkari, A.: Lightweight hybrid detection of data exfiltration using dns based on machine learning. In: Proceedings of the 11th IEEE International Conference on Communication and Network Security (ICCNS). IEEE, Weihai, China (Dec 2021)
26. Nadler, A., Aminov, A., Shabtai, A.: Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol (Jun 2018). https://doi.org/10.48550/arXiv.1709.08395, http://arxiv.org/abs/1709.08395, arXiv:1709.08395 [cs]
27. News, T.H.: 45 Previously Unreported Domains Expose Longstanding Salt Typhoon Cyber Espionage, https://thehackernews.com/2025/09/45-previously-unreported-domains-expose.html, section: Article
28. Ozery, Y., Nadler, A., Shabtai, A.: Information Based Heavy Hitters for Real-Time DNS Data Exfiltration Detection. In: Proceedings 2024 Network and Distributed System Security Symposium. Internet Society, San Diego, CA, USA (2024). https://doi.org/10.14722/ndss.2024.24388, https://www.ndss-symposium.org/wp-content/uploads/2024-388-paper.pdf
29. PISCES: Municipal network flow data. https://pisces-intl.org/ (2025)
30. Domain names - implementation and specification. Request for Comments RFC 1035, Internet Engineering Task Force (Nov 1987). https://doi.org/10.17487/RFC1035, https://datatracker.ietf.org/doc/rfc1035, num Pages: 55

31. Roelofs, G., Adler, M.: zlib Home Site (1995), https://zlib.net/, version 1.3, accessed 2026-01-18
32. Rostovcev, N.: APT41 World Tour 2021 on a tight schedule (Aug 2022), https://www.group-ib.com/blog/apt41-world-tour-2021/
33. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017). https://doi.org/10.48550/arXiv.1707.06347, http://arxiv.org/abs/1707.06347, arXiv:1707.06347 [cs]
34. Symantec, Black, C.: Sunburst: Supply Chain Attack Targets SolarWinds Users, https://www.security.com/threat-intelligence/sunburst-supply-chain-attack-solarwinds
35. Whitmore, W.: Today's Attack Trends — Unit 42 Incident Response Report (Feb 2024), https://www.paloaltonetworks.com/blog/2024/02/unit-42-incident-response-report/
36. Xu, D., Liu, W., Li, J., Chen, Z.: Strategically Aged Domain Detection: Capture APT Attacks With DNS Traffic Trends (Dec 2021), https://unit42.paloaltonetworks.com/strategically-aged-domain-detection/
37. Yassine, S., Khalife, J., Chamoun, M.: A Survey of DNS Tunnelling Detection Techniques Using Machine Learning. In: Proceedings of the 1st International Conference on Big Data and Cyber-Security Intelligence (BDCSIntell). CEUR Workshop Proceedings, vol. 2343, pp. 63–66 (2018)
38. Zhang, J., Yang, L., Yu, S., Ma, J.: A DNS Tunneling Detection Method Based on Deep Learning Models to Prevent Data Exfiltration. In: Liu, J.K., Huang, X. (eds.) Network and System Security. pp. 520–535. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-36938-5_32
39. Ågren, W.: The NT-Xent loss upper bound (May 2022). https://doi.org/10.48550/arXiv.2205.03169, http://arxiv.org/abs/2205.03169, arXiv:2205.03169
40. Žiža, K., Tadić, P., Vuletić, P.: DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour. International Journal of Information Security **22**(6), 1865–1880 (Dec 2023). https://doi.org/10.1007/s10207-023-00723-w, https://link.springer.com/article/10.1007/s10207-023-00723-w