Computers & Security 170 (2026) 105043

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

STUN comprehension-optional attributes as a covert channel: Design, implementation, and detection

Gorka Gorrochategui *, Unai Zulaika, Pablo Garaizar

Faculty of Engineering, University of Deusto, Avenida de las Universidades 24, Bilbao, 48007, Spain

ARTICLE INFO

Dataset link: Google Drive Share - PCAPs, Source Code, Scripts, Logs (Original data)

Keywords:
Covert channels
STUN
NAT traversal
Network security
Real-time communication
Offensive security
Intrusion detection
Entropy analysis

ABSTRACT

Real-time communication protocols carry implicit trust at perimeter defenses because enterprise networks must allow them for Voice over IP (VoIP) and Web Real-Time Communication (WEBRTC) operation. Existing covert-channel literature has explored Session Initiation Protocol (SIP) signaling (Mazurczyk and Szczypiorski, 2008), Real-Time Transport Protocol (RTP)/RTCP media (Mazurczyk, 2013; Schmidt et al. 2018), and WEBRTC streams (Barradas et al. 2020; Figueira et al. 2022), but Session Traversal Utilities for NAT (STUN), despite its central role in NAT traversal and its ubiquity in modern networks, has not been investigated as a covert channel carrier.

We design a Layer 2 covert channel that encapsulates arbitrary Ethernet frames within STUN comprehension-optional attributes, which RFC 5389 (Rosenberg et al., 2008) requires compliant implementations to silently ignore. A bidirectional polling mechanism overcomes STUN's client–server asymmetry, producing a full-duplex tunnel indistinguishable from legitimate Interactive Connectivity Establishment (ICE) keep-alive traffic.

A portable proof-of-concept was implemented and deployed on two embedded Linux devices of distinct architectures: an ARM-based Yealink T19P E2 IP phone and a MIPS-based Ubiquiti EdgeRouter X. Empirical evasion testing was conducted against Snort 3 with 47,143 Talos rules, Suricata 8 with the ET Open ruleset, Zeek with the `spicy_STUN` analyzer, and a FortiGate 40F firewall running full Unified Threat Management (UTM) services. Across 3646 data-carrying covert messages, no signature or protocol-aware engine flagged the traffic as anomalous.

To address the detection gap, we propose a lightweight Shannon-entropy detector applied to optional-attribute payloads. The mean per-attribute entropy of the covert channel reaches 6.247 bits/byte, well above the highest entropy observed in 1358 legitimate STUN messages from the ITC-Net-blend-60 dataset (4.392 bits/byte from Telegram). A 4.5 bits/byte threshold flags all 3646 covert messages while producing zero false positives across the legitimate corpus. These results expose a systemic detection gap in current network defense strategies and motivate the integration of entropy-aware inspection into Real-Time Communication (RTC) signaling protocols.

1. Introduction

Network covert channels are a core component of Advanced Persistent Threat (APT) operations, enabling stealthy Command and Control (C2) communications that bypass perimeter defenses. As defensive mechanisms have matured to include Deep Packet Inspection (DPI) and behavioral analysis for traditional carriers like DNS (Žiža et al., 2023), adversaries have shifted towards legitimate, high-trust protocols already present in enterprise infrastructure. Among these, RTC protocols are an attractive target: firewalls must allow their traffic to support VoIP and WEBRTC services, and the parsing complexity of the associated signaling makes thorough inspection expensive. In this paper, we show that STUN, the NAT-traversal substrate beneath these protocols, can itself be repurposed as a bidirectional covert channel that evades current Network Intrusion Detection System (NIDS) and Next-Generation Firewall (NGFW) solutions.

STUN, defined in IETF RFC 5389 (Rosenberg et al., 2008), has not, to the best of our knowledge, been studied as a covert channel carrier in the literature. This is a notable omission given its widespread deployment as a core component of the ICE framework (Keranen et al., 2018), facilitating connectivity for platforms such as Zoom, Microsoft Teams, and hardware IP phones. Existing research on covert channels in real-time communications has concentrated on layers above NAT traversal: SIP signaling fields (Mehić et al., 2014; Tsiatsikas et al., 2015), RTP/RTCP payloads (Schmidt et al., 2018; Saenger et al., 2020),

* Corresponding author.
E-mail addresses: gorka.gorrochategui@deusto.es (G. Gorrochategui), unai.zulaika@deusto.es (U. Zulaika), garaizar@deusto.es (P. Garaizar).

https://doi.org/10.1016/j.cose.2026.105043
Received 16 April 2026; Received in revised form 25 May 2026; Accepted 22 June 2026
Available online 25 June 2026

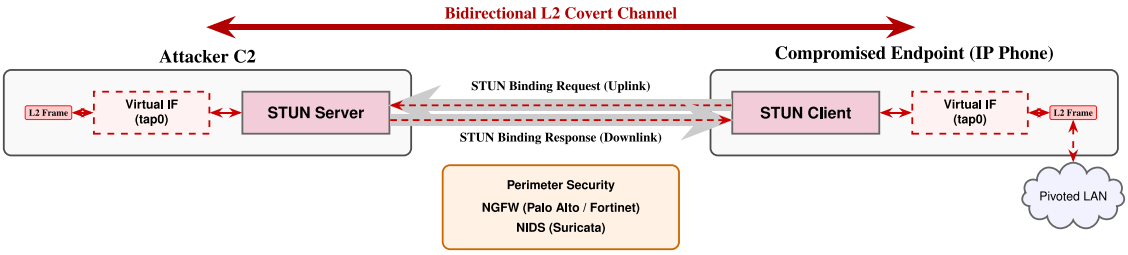

Fig. 1. Architecture of the STUN-based L2 covert channel bypassing perimeter security.

and WebRTC media streams (Barradas et al., 2020; Figueira et al., 2022; Bocovich et al., 2024). These approaches either require an active media session or operate at a layer above the NAT traversal exchange itself. STUN-based channels occupy a distinct position: they exploit the extensibility of the connectivity establishment layer, using comprehension-optional attributes that RFC-compliant implementations are required to silently ignore, independently of any media session.

Realizing this opportunity, however, requires confronting an architectural constraint: STUN's inherent client–server asymmetry, in which the server cannot initiate unsolicited messages, poses a structural challenge for full-duplex C2 that prior work has not addressed. This paper presents the design and implementation of a fully RFC-compliant STUN-based covert channel that overcomes this architectural asymmetry. We introduce a bidirectional polling mechanism that turns STUN's request–response model into a transport for arbitrary IP traffic, and use it to build a complete Layer 2 (L2) tunnel over virtual network interfaces (TAP). The result is a remote bridge usable for Red Team pivoting directly from compromised endpoints.

To validate the practical viability of this approach, we deployed the prototype on two heterogeneous platforms: a production Yealink T19P E2 IP phone (ARM), representative of the commodity VoIP hardware found in enterprise environments, and a Ubiquiti EdgeRouter X (MIPS), representative of the commodity edge-routing infrastructure widely deployed in SME networks. Both targets run resource-constrained embedded Linux and were confirmed to sustain reliable bidirectional tunnel operation. The channel successfully evades detection by Snort 3, Suricata 8 NIDS, and Fortinet NGFW. The results expose a detection gap in current network defense strategies: security appliances often grant implicit trust to these high-frequency NAT traversal exchanges, allowing full-duplex exfiltration tunnels to operate indistinguishably from legitimate signaling traffic. The primary contributions of this paper are summarized as follows:

- We design a bidirectional L2 transport mechanism that exploits the architectural extensibility of the STUN protocol, using comprehension-optional attributes to achieve persistent connectivity without the overhead or visibility of active media sessions.
- We provide a lightweight, self-contained implementation cross-compiled for both ARM and MIPS architectures, demonstrating that commodity embedded hardware (a Yealink T19P E2 IP phone and a Ubiquiti EdgeRouter X) can be transformed into a stealthy network bridge with minimal resource footprint, establishing the technique's applicability across heterogeneous embedded Linux platforms.
- We conduct a comparative evaluation across multiple NIDS (Snort 3, Suricata 8, Zeek) and enterprise firewalls (Fortinet), identifying a widespread detection gap in how security appliances handle protocol-compliant STUN extensions.
- We propose and implement a forensic detection framework based on Shannon entropy analysis, providing a practical blueprint for network defenders to identify semantic protocol misuse in NAT traversal signaling (see Fig. 1).

2. Ethical considerations

This research is conducted for educational and defensive-security purposes within the scope of offensive security methodology. All experiments were performed exclusively in controlled laboratory environments on owned or explicitly authorized equipment and networks. The techniques described are presented to inform security assessment and to support the development of stronger detection and mitigation capabilities; they are not intended for unauthorized use.

No Institutional Review Board (IRB) approval was required for this study, as no human subjects, personal data, or private communications were involved at any stage. All network traffic analyzed was either self-generated within our testbed or obtained from publicly available research datasets. The ITC-Net-blend-60 dataset (Bayat et al., 2024) is published under open-access terms for research purposes, and the MAWI Working Group Archive (MAWI Working Group, 2026) provides anonymized backbone traces with transport-layer payloads removed in accordance with their published privacy guidelines (MAWI Working Group, 1999).

The source code for the covert channel prototype is not publicly released. Following responsible disclosure practices in offensive security research, access is restricted to qualified researchers with verified institutional affiliation and a stated defensive or academic purpose. This policy balances the need for scientific reproducibility with the imperative to minimize immediate weaponization risk. The implementation details provided throughout this manuscript are sufficient to enable independent replication by security professionals without distributing a ready-to-deploy tool.

To further support the defensive community, this work contributes a concrete detection mechanism (Section 10) based on Shannon entropy analysis, along with empirical baselines derived from legitimate traffic. We believe that the publication of both the attack vector and its corresponding countermeasure serves the broader goal of strengthening network security posture against protocol-compliant covert channels.

3. Related work

Network covert channels are traditionally categorized into storage and timing channels (Wendzel et al., 2015). In RTC, research has extensively explored embedding data within signaling protocols or manipulating media streams via RTP (Mazurczyk, 2013). These techniques often use payload-based embedding or header-field modification to establish stealthy communication paths. While NAT traversal is essential for RTC ubiquity (Trautwein et al., 2025), it also exposes an under-researched attack surface.

Within the signaling plane, prior work has demonstrated several embedding vectors. Mazurczyk and Szczypiorski (Mazurczyk and Szczypiorski, 2008) embedded covert data within SIP header fields such as tags, branch tokens, and Call-ID values, while Mehić et al. (2014) proposed an independent SIP-based covert channel, quantified its steganographic bandwidth, and evaluated its detectability against Snort IDS. Schmidt et al. (2018) injected covert data into fabricated RTP packets during Voice Activity Detection silence periods, and Saenger et al.

(2020) injected covert data within fabricated silence RTP packets during VAD-suppressed intervals, tunneling arbitrary IP traffic through a TUN virtual interface. At the session description level, Tsiatsikas et al. (2015) exploited SDP fields to establish botnet C2 communications, confirming that the RTC signaling stack offers multiple underexplored embedding vectors. These approaches generally operate by modifying existing protocol fields or timing patterns, and their detectability through field-level integrity checks and statistical timing analysis has been studied in subsequent literature. Unlike these signaling-layer techniques, which repurpose or modulate fields the protocol already defines, our channel leaves every existing STUN field intact and instead introduces new comprehension-optional attributes, producing no malformed values for field-level integrity checks to flag.

A separate line of research targets the media plane for censorship circumvention. Systems like *Facet* (Li et al., 2014) and *DeltaShaper* (Barradas et al., 2017) shape traffic to resemble legitimate multimedia flows, while *Protozoa* (Barradas et al., 2020) and *Stegozoa* (Figueira et al., 2022) instrument WebRTC video pipelines to carry arbitrary data. Although high throughput is achievable, subsequent work has explored Machine Learning (ML)-based classifiers targeting packet timing and length distributions in these systems (Barradas et al., 2018). In parallel, deployment systems like *Snowflake* (Bocovich et al., 2024) demonstrate the resilience of WebRTC-based traffic against censorship due to its ubiquity. A characteristic shared by these media-plane approaches is that they presuppose an established WebRTC session: the covert payload rides within video or data streams that exist only after ICE negotiation has completed. Unlike these media-plane systems, a channel operating at the STUN layer itself requires no media session and presents to the network as ordinary NAT traversal signaling.

A more recent direction targets the NAT traversal relay layer directly. Crosser (2025) demonstrated that TURN relay servers operated by Zoom and Microsoft Teams can be weaponized as C2 channels: short-lived TURN credentials extracted during WebRTC session setup are repurposed to tunnel arbitrary traffic through trusted conferencing domains using SCTP DataChannels over DTLS. The technique bypasses egress filtering by exploiting the implicit trust that organizations grant to conferencing infrastructure, but carries an inherent operational dependency on active meeting credentials and is subject to platform-side mitigation (Zoom subsequently disabled the peer-to-peer TURN capability on which the technique relies). Unlike Crosser's approach, which depends on short-lived third-party conferencing credentials the platform can revoke, our channel requires only an attacker-controlled STUN endpoint that any commodity host can provide.

Across the three lines of work surveyed above, none has examined the Network Address Translation (NAT) traversal control plane itself as a covert-channel carrier. Our channel sits one protocol layer below all of them, on the control plane itself: it remains protocol-compliant throughout, requires no active media session or codec instrumentation, and depends on no third-party conferencing credentials that a vendor can revoke. This keeps it viable even on resource-constrained VoIP endpoints (Petit-Huguenin et al., 2020; Mahy et al., 2010).

Following the unified description framework for network information hiding methods (Wendzel et al., 2016), our technique is classified as a *Network Covert Storage Channel* that combines the *Add Redundancy* pattern (injecting supplementary Type-Length-Value (TLV) attributes into the STUN message structure) with exploitation of the *Reserved/Unused* pattern, as the comprehension-optional attribute range (0x8000–0xFFFF) constitutes protocol-reserved space that compliant implementations must silently discard (Rosenberg et al., 2008). This dual-pattern classification distinguishes our approach from both the header-field modulation techniques used in the SIP/SDP signaling work and the media-stream injection techniques used in the RTP/VAD work reviewed above.

While signature-based NIDS like Snort 3 and Suricata are effective against known attack patterns, they lack the semantic depth to detect protocol-compliant covert channels that introduce no malformed fields. Deep protocol analysis frameworks such as Zeek with the Spicy parser engine (Sommer et al., 2016) provide a stronger foundation by enforcing formal grammars. Our work builds on this by extending entropy-based detection principles, previously applied to covert timing channels (Gianvecchio and Wang, 2007), to the STUN control plane, quantifying statistical anomalies in attribute payloads that signature-based NIDS inherently miss.

Table 1
Comparative analysis of NAT behaviors.

NAT type	Mapping	Filtering policy
Full Cone	Endpoint-Indep.	**None.** Any external host can send packets to the mapped port.
Restricted	Endpoint-Indep.	**IP Restricted.** Only previously contacted external IPs can reply.
Port Restr.	Endpoint-Indep.	**IP & Port Restricted.** Exact source IP and port validation required.
Symmetric	Endpoint-Dep.	**Strict.** Unique mapping per destination; only the target can reply.

4. Technical background

The efficacy of the proposed covert channel is rooted in the structural properties of the STUN protocol and its strategic role in modern network signaling.

4.1. NAT topologies and traversal constraints

NAT behavior, defined by mapping and filtering policies (Table 1), remains the primary determinant of protocol transparency. While restrictive configurations like Symmetric NAT assign unique external ports for every destination, our implementation is designed to be topology-agnostic by ensuring all transactions are initiated from the internal network to maintain stateful consistency.

This ubiquity is reinforced by the ICE framework (Keranen et al., 2018), which utilizes STUN not only for discovery but as a persistent keep-alive mechanism to maintain NAT mappings open. In production environments such as enterprise VoIP and WebRTC (Jennings et al., 2021), these "heartbeats" generate a continuous stream of legitimate signaling traffic, creating a baseline of high-trust "cover" traffic.

Recent large-scale measurements on STUN-derived NAT traversal protocols (Trautwein et al., 2025), involving over 4.4 million traversal attempts across 85,000+ distinct networks, report a hole-punching success rate of approximately 70%, with 97.6% of successful connections established on the first attempt. These results confirm the centrality of NAT traversal signaling in modern P2P architectures.

4.2. STUN protocol structure and extensibility

STUN is a binary, request–response protocol (Rosenberg et al., 2008). Its 20-byte header includes a 96-bit *Transaction ID*, used to correlate requests and responses (Table 2).

The protocol employs a TLV format for attributes. RFC 5389 specifies that implementations must ignore unknown attributes in the "comprehension-optional" range (0x8000–0xFFFF) (Rosenberg et al., 2008). This extensibility provides an RFC-compliant vector for injecting variable-length payloads without disrupting standard protocol parsing or triggering alerts in protocol-aware security appliances. This mechanism was first defined in RFC 5389 (Rosenberg et al., 2008) and is preserved without modification in the current standard, RFC 8489 (Petit-Huguenin et al., 2020) (see Fig. 2).

Fig. 3 illustrates the discovery flow. The server cannot initiate contact (Rosenberg et al., 2008). Our design transforms this perceived limitation into a stealth feature by using polling to drive a full-duplex tunnel through a half-duplex protocol.

Table 2
STUN header field definitions.

Field	Size	Value	Description
Msg. Type	2 B	Variable	Request/Response class (Rosenberg et al., 2008)
Magic Cookie	4 B	0x2112A442	Fixed protocol id (Rosenberg et al., 2008)
Transaction ID	12 B	Random	Request/response correlator (Rosenberg et al., 2008)

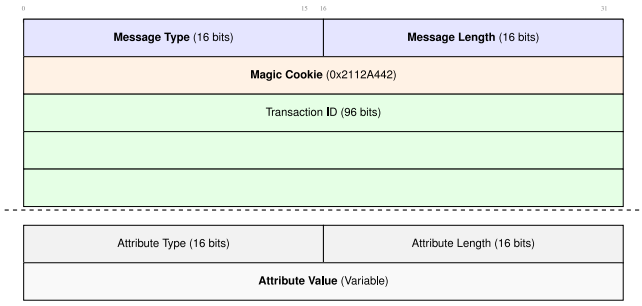

Fig. 2. Structure of a STUN Message (RFC 5389) adapted to single column width.

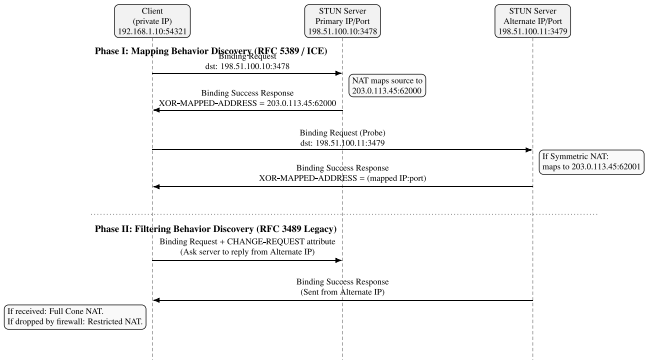

Fig. 3. STUN message flow for modern mapping behavior discovery (Phase I) and legacy filtering behavior probes (Phase II).

5. Methodology

This section sets out our methodology in two parts. We first define the threat model that frames the adversary, the defender, and the network assumptions under which the covert channel operates, and then describe the evaluation pipeline used to measure its detectability against the security appliances under test.

5.1. Threat model

Our threat model assumes a compromised internal endpoint with privileges to create virtual interfaces (TUN/TAP) and initiate outbound UDP traffic to standard STUN ports (3478, 5349) (Rosenberg et al., 2008).

Attacker Goals: The adversary seeks to establish a bidirectional, full-duplex C2 tunnel for data exfiltration and network pivoting (Rosenberg et al., 2008). The primary objective is to maintain a communication channel that is indistinguishable from legitimate RTC traffic (Zander et al., 2007).

Defender Capabilities: The network is protected by NGFW (e.g., Fortinet, Palo Alto) and NIDS (e.g., Suricata) with DPI capabilities for protocol validation. However, we assume the absence of advanced statistical or behavioral analysis capable of detecting high-entropy payloads within TLV attributes (Zander et al., 2007).

Network Assumptions: Outbound STUN traffic is permitted by default to support VoIP and WebRTC applications (Jennings et al., 2021; Keranen et al., 2018). We assume that strict STUN server whitelisting is not enforced due to the massive, decentralized nature of modern RTC infrastructure (Shodan Search Engine, 2026). This environment provides the necessary "cover traffic" to mask the covert channel's operation (see Fig. 4).

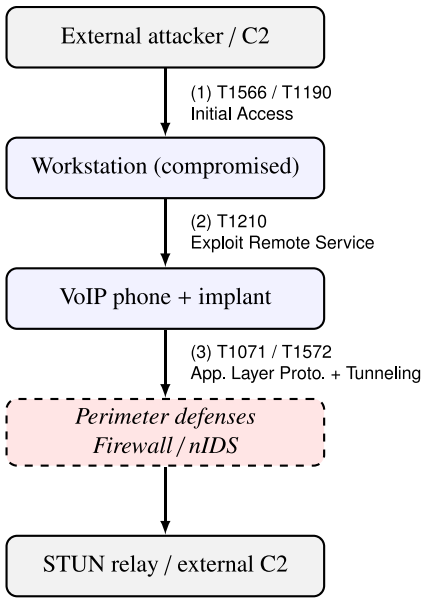

Fig. 4. Threat model with MITRE ATT&CK techniques labeled per stage.

5.2. Evaluation pipeline

The proposed covert channel is evaluated against four detection configurations using two reference inputs: the captured covert-channel traffic and the ITC-Net-blend-60 corpus of legitimate RTC traffic. Each capture is fed in turn to every detector, and the per-platform outcomes are aggregated into the consolidated detection table reported in Section 10.1. Fig. 5 summarizes this workflow.

We treat detection as a binary classification problem at the granularity of individual STUN messages. The covert-channel traffic supplies the positive class, formed by the messages that carry an encapsulated payload, while the legitimate corpus supplies the negative class. Because the covert endpoint is under our control, the ground-truth label of every message is known by construction, so detection performance can be measured without manual annotation. We report recall as the fraction of payload-carrying covert messages that a detector flags, and precision as the fraction of flagged messages that are genuinely covert, with every false positive drawn from the legitimate corpus. For the two entropy-based configurations the decision threshold is a free parameter, so the Suricata rule is evaluated across six thresholds from 4.5 to 6.0 bits, whereas the Zeek detector applies a fixed threshold of 4.5 bits to each comprehension-optional attribute in isolation.

The four configurations are chosen to span the range of inspection capability available to a defender. Snort 3 with the Cisco Talos ruleset and Suricata with the Emerging Threats Open ruleset represent the signature-based detection that production NIDS deployments rely on today. The remaining two configurations are the countermeasures we

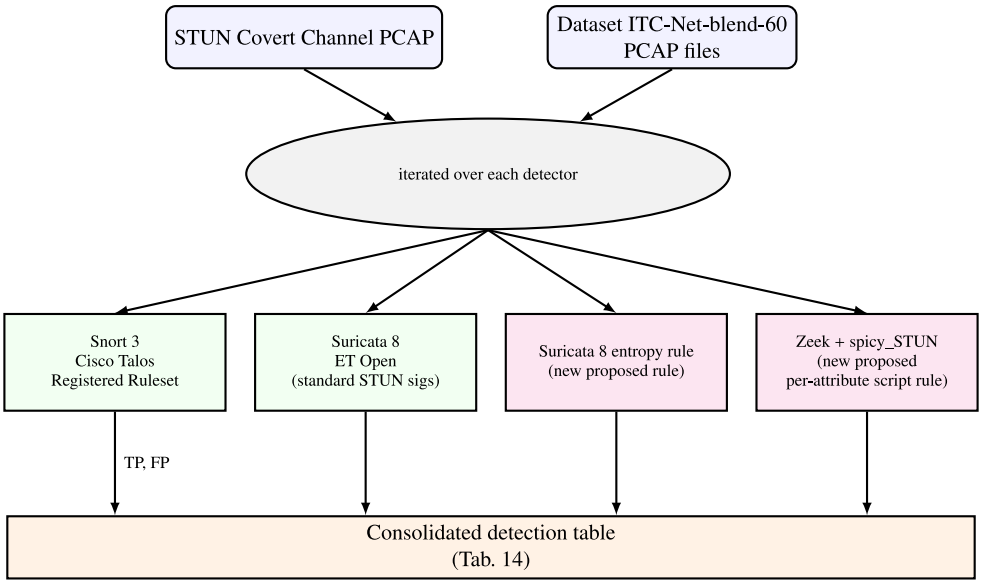

Fig. 5. Experimental evaluation pipeline. The covert channel capture and the ITC-Net-blend-60 corpus of legitimate RTC traffic are iterated through each detector in turn. Boxes in pink denote the detection rules proposed in this work. Per-detector outcomes feed the consolidated detection table reported in Section 10.1.

propose: a Suricata entropy rule that brings information-theoretic inspection to a production signature engine, and a Zeek policy backed by the `spicy_STUN` parser that computes the Shannon entropy of each comprehension-optional attribute. Exercising all four on identical inputs isolates the effect that semantic and statistical depth has on the detectability of the channel.

5.3. Longitudinal analysis of STUN prevalence

To validate STUN as a viable covert carrier, we analyzed its global presence. As of February 2, 2026, Shodan indexed 154,643 publicly reachable STUN servers (Shodan Search Engine, 2026). This infrastructure provides a pervasive baseline for blending malicious signaling with legitimate traffic (Shodan Search Engine, 2026).

5.3.1. Traffic volume and growth trends

We analyzed longitudinal backbone traffic using the MAWI Working Group Archive (MAWI Working Group, 2026). To ensure consistency, 900-second snapshots from the first Monday of February were compared across multiple years.

A technical constraint of the MAWI dataset is the removal of transport-layer payloads for privacy (MAWI Working Group, 1999). Consequently, the STUN Magic Cookie (`0x2112A442`) cannot be verified. We therefore approximate prevalence using a conservative flow-level heuristic based on the IANA-assigned port 3478. This method likely underestimates total activity, as it excludes ICE peer-to-peer checks on ephemeral ports and STUN traffic on non-standard ports, providing a consistent lower bound for longitudinal comparison (see Fig. 6).

The results, illustrated in Fig. 7, show a clear upward trend. Infrastructure remained stable until 2021, followed by a sharp rise coinciding with the global adoption of WebRTC platforms.

Our analysis shows a 239-fold increase in STUN flows from 2019 to 2026. This exponential growth correlates with the proliferation of video conferencing and WebRTC-based platforms (Trautwein et al., 2025; Jennings et al., 2021). The trend confirms STUN's increasing ubiquity and its "allow-by-default" status in modern firewalls, making it an ideal substrate for hiding encrypted L2 payloads within legitimate "heartbeat" traffic (Rosenberg et al., 2008; Keranen et al., 2018).

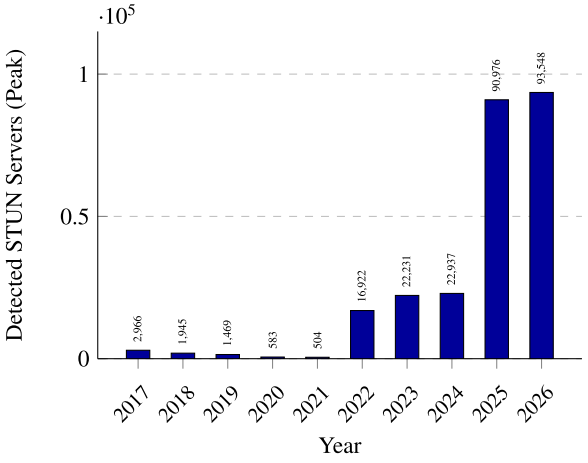

Fig. 6. Evolution of publicly accessible STUN infrastructure (2017–2026). The surge in the last biennium reflects the protocol's expansion.

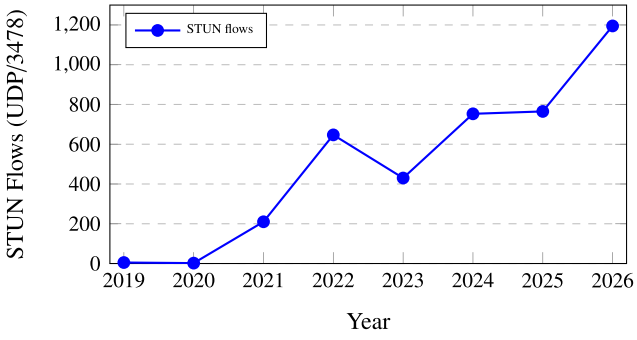

Fig. 7. Evolution of flows to STUN well-known port (3478) in MAWI traces (MAWI Working Group, 2026). Data measured on the first Monday of February at 14:00–14:15 local time.

Table 3
Evaluation of STUN fields for covert data embedding.

Field	Capacity	Detection risk	Status
Transaction ID	12 B (fixed)	Low. High entropy expected.	Discarded
Padding Bytes	0–3 B	High. Deterministic zero-check.	Discarded
Optional Attribute	≈1450 B	Low. Unknown optional attrs. ignored.[a]	Selected

[a] RFC 5389, §7.3.1: "Unknown comprehension-optional attributes MUST be ignored by the agent" (Rosenberg et al., 2008).

6. Proposed covert channel mechanism

This section details the operational mechanics of the covert channel. We first analyze the STUN protocol structure to identify the optimal carrier field that ensures evasion of strict protocol parsers. Subsequently, we describe the implementation of the tunneling infrastructure that relies on standard Linux kernel facilities to achieve transparent network bridging.

Within the taxonomy of hiding patterns established by Wendzel et al. (2015, 2016), the proposed method follows the hierarchy:

`Network Covert Storage Channels`
→ `Modification of Non-Payload`
→ `Structure Modifying`
→ `Add Redundancy`

The channel is a *storage* channel because covert data is embedded in protocol fields rather than encoded in timing variations. It operates through *modification of non-payload* elements: rather than embedding covert data within the value of any data-carrying attribute defined by the protocol (such as `XOR-MAPPED-ADDRESS`, which carries the functional content of a Binding Response), the channel introduces a new comprehension-optional TLV attribute (type `0x8FAD`) using STUN's built-in extensibility mechanism. This approach is analogous to adding IPv4 option fields or extending HTTP headers with new entries: it constitutes a structural addition to the protocol data unit rather than a modification of existing data-carrying content. This constitutes an *Add Redundancy* pattern, injecting supplementary protocol elements that carry the hidden payload. A secondary *Reserved/Unused* pattern also applies, since the comprehension-optional range (`0x8000–0xFFFF`) represents protocol-reserved space that RFC 5389 mandates compliant agents to silently ignore (Rosenberg et al., 2008). This dual-pattern exploitation, structural addition within explicitly ignored space, is what provides the channel its RFC-compliant evasion properties.

6.1. Carrier field selection and analysis

The primary challenge in establishing a high-bandwidth covert channel over STUN is identifying a storage vector that offers sufficient capacity without violating the strict grammatical rules enforced by modern DPI engines. Unlike legacy NIDS that rely on simple signature matching, modern appliances validate protocol adherence. Therefore, the embedding mechanism must utilize fields that allow for high entropy and variable length while remaining syntactically compliant with RFC 5389 (Rosenberg et al., 2008).

To select the optimal encapsulation location, we conducted a structural analysis of the STUN packet, evaluating three potential candidates based on capacity, entropy, and risk of heuristic detection. The comparative analysis is summarized in Table 3.

6.1.1. Candidate A: Transaction ID (Header)

The STUN header contains a 12-byte Transaction ID designed to be cryptographically random (Rosenberg et al., 2008). While this field offers excellent stealth due to its high entropy, its fixed size presents a bottleneck. Achieving usable L2 throughput would require an anomalously high packet rate (PPS), likely triggering volumetric anomaly detection. This field is discarded for data transport but remains viable for low-bandwidth C2 signaling.

6.1.2. Candidate B: Padding bytes

STUN attributes are aligned to 32-bit boundaries (Rosenberg et al., 2008). When length is not a multiple of 4, padding bytes are appended. This space is extremely scarce (0–3 bytes) and often validated as zero by strict parsers. Non-zero data here constitutes a trivial signature for detection, leading to the rejection of this candidate.

6.1.3. Candidate C: Comprehension-optional attributes (Selected)

RFC 5389 mandates that if an agent encounters an unknown attribute in the optional range ($0x8000 - 0xFFFF$), it *must* ignore it and proceed (Rosenberg et al., 2008). This allows the injection of variable-length payloads. By defining a custom attribute (e.g., `0x8FAD`), the system utilizes the remaining Maximum Transmission Unit (MTU) for payload storage. Firewalls identify an "unknown optional" attribute and permit it per the standard, offering the optimal balance of bandwidth and compliance.

6.2. Tunneling infrastructure

To achieve transparent bridging, the architecture utilizes Linux TUN/TAP virtual drivers, allowing user-space applications to interact with the kernel network stack.

6.2.1. Packet injection and extraction

The application reads outbound ethernet frames from a virtual interface (e.g., `tap0`). Since STUN is a binary protocol, the captured payload is directly injected into the Value field of the custom attribute without encoding overhead. This approach exploits STUN's flexible attribute structure, which natively supports arbitrary binary data as specified in RFC 5389 (Rosenberg et al., 2008).

6.2.2. Handling asymmetry via polling

In this architecture, the client-side component is a software implant: a lightweight agent deployed on the compromised host, responsible for managing the covert channel lifecycle. To overcome the inherent client–server asymmetry where servers cannot initiate communication, the implant utilizes an adaptive polling mechanism. It periodically sends Binding Requests with uplink data or heartbeats (Rosenberg et al., 2008). The Relay Server piggybacks downlink traffic within the corresponding Binding Response attributes (Rosenberg et al., 2008). This strategy ensures the channel remains operational regardless of NAT type, as every inbound packet is a legitimate response to an internally-generated request.

7. Software architecture and asynchronous I/O design

To ensure high performance on resource-constrained embedded devices across heterogeneous architectures (ARM and MIPS), we implemented a strictly non-blocking architecture. By utilizing the POSIX `select()` system call, the system multiplexes I/O operations between the virtual network interface (`tun_fd`) and the physical UDP socket (`sock_fd`).

7.1. Event loop and packet processing

The core logic acts as a bidirectional bridge. When the TUN interface captures a raw frame, the application encapsulates it into a custom STUN attribute for uplink transmission. Conversely, when a UDP datagram is received, the system validates the STUN header, extracts and decrypts the payload, and injects the reconstructed frame back into the kernel network stack. This event-driven approach avoids busy-waiting and keeps CPU utilization low on legacy hardware.

7.2. Asymmetric buffering strategy

Due to NAT constraints, the relay server cannot initiate unsolicited contact and must wait for a *Binding Request* to transmit downlink

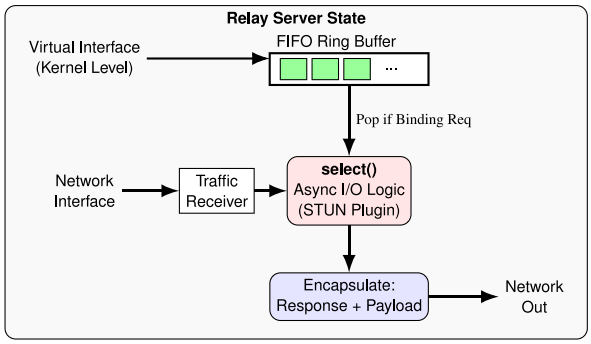

Fig. 8. Server-Side asynchronous buffering and payload encapsulation logic.

data (Rosenberg et al., 2008). To manage this asymmetry, the server implements a per-client FIFO Ring Buffer (see Fig. 8).

Incoming packets destined for the client are pushed into the non-blocking queue. Upon receipt of a client "heartbeat" (Binding Request), the server pops pending frames, aggregates them up to the MTU limit, and piggybacks them within the *Binding Response*. This mechanism enables asynchronous data delivery while adhering to the client-initiated polling rate.

8. Experimental validation

To empirically validate the feasibility, performance, and evasion capabilities of the proposed STUN-based covert channel, we deployed a controlled testbed environment mirroring a hardened enterprise network. This evaluation prioritizes real-world applicability by deploying the prototype on resource-constrained commercial hardware.

8.1. Testbed environment and hardware targets

The experimental topology consists of a compromised branch office endpoint establishing a covert tunnel to an external C2 server.

8.1.1. Hardware implant specification

The implant was validated on two resource-constrained embedded Linux platforms representing distinct hardware architectures and device classes. The Yealink T19P E2 served as the primary evaluation target: all throughput benchmarks, latency measurements, and evasion tests against production security appliances (Scenarios A–C below) were conducted with this device over a controlled LAN path. The Ubiquiti EdgeRouter X was used exclusively as a secondary portability target to confirm operational viability on a MIPS architecture and a different device class; it was not subjected to the full evasion test suite, and all measurements were performed over a production WAN path.

Yealink T19P E2 (ARM). The primary implant target is a Yealink T19P E2 IP phone (Firmware 53.84.0.15, Hardware 53.0.0.224.0.0.0), running an ARM-based embedded Linux environment. This device represents a significant challenge for malware execution due to its minimalist architecture and limited CPU resources, and is representative of entry-level legacy VoIP hardware common in enterprise deployments.

Ubiquiti EdgeRouter X (MIPS). To assess portability beyond VoIP endpoints, the implant was additionally deployed on a Ubiquiti EdgeRouter X running EdgeOS (Debian-based Linux, kernel 4.9). This device is powered by a MediaTek MT7621 SoC (MIPS 1004Kc dual-core, 880 MHz, 256 MB RAM) and is widely deployed as SME edge-routing infrastructure. TUN/TAP kernel support was confirmed present and functional. Operational viability was confirmed via a 60-second `iperf3` session through the active tunnel over a production WAN path, sustaining 41.9 Mbps mean throughput (see Section 9). This platform extends the threat surface of the covert channel from RTC endpoints to network infrastructure equipment, which is typically granted even higher implicit trust by perimeter defenses.

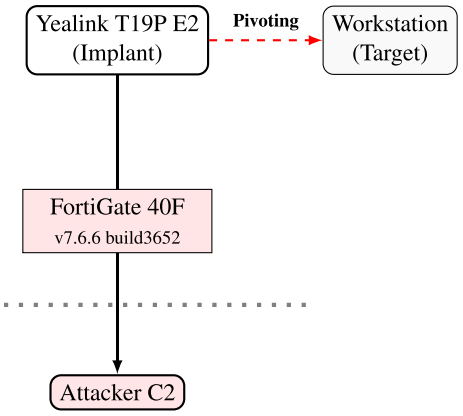

Fig. 9. Experimental setup for Scenario B: Vertical architecture utilizing a FortiGate 40F hardware appliance for UTM inspection during the covert channel evaluation.

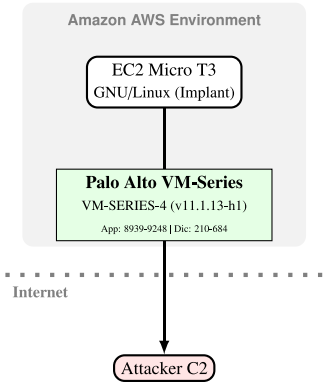

Fig. 10. Experimental setup for Scenario C: Virtualized architecture within Amazon AWS featuring a Palo Alto VM-Series (PAN-OS v11.1.13-h1) and a T3.micro instance.

8.1.2. Evaluation scenarios

The channel's stealth was assessed against three security environments representative of current perimeter defenses.

Scenario A (Suricata NIDS): Network traffic was mirrored via a Switched Port Analyzer (SPAN) port to an analysis node running Suricata NIDS v8.0.4 on Debian Trixie. This setup utilized the Emerging Threats (ET) Open ruleset updated as of February 20, 2026.

Scenario B (Fortinet): Perimeter security was enforced by a FortiGate 40F hardware appliance running firmware v7.6.6 build3652. All UTM services were enabled, including DPI, Intrusion Prevention System (IPS), and granular application control (see Fig. 9).

Scenario C (Palo Alto Networks): To evaluate cloud-native security, a Palo Alto Networks VM-Series instance (PAN-OS v11.1.13-h1) was deployed within an Amazon AWS EC2 environment. The appliance classified the traffic as STUN and generated no security alert. A detailed per-attribute analysis of this platform is left for future work (see Fig. 10).

8.2. Security assessment and detection results

The primary objective of the assessment was to determine if the channel triggered security alerts. Table 4 summarizes the findings.

No tested platform classified the traffic as malicious or blocked the covert channel. While Suricata logged informational STUN protocol events (as it does for any legitimate WebRTC session, see Section 10.2), none of the appliances identified the covert payload within the optional attributes.

Table 4
Detection and evasion results across security vendors.

Platform	Classification	Alert
Snort 3 (Talos)	Generic UDP	None
Suricata 8 (ET Open)	STUN Protocol	ET INFO
FortiGate 40F	STUN/Network.Service (app 16573)	None
Palo Alto VM-Series	STUN	None[a]

[a] Per-attribute payload inspection was not performed on this platform; the evaluation confirmed top-level traffic classification only. A deeper analysis of Palo Alto App-ID and threat-prevention behavior against STUN optional attributes remains future work.

8.2.1. Snort3 Cisco Talos ruleset analysis

Snort3 (v3.11.1.0) was tested with the complete Cisco Talos registered ruleset, comprising 47,143 active detection rules distributed across 66 rule category files. An exhaustive search of this ruleset revealed no rules targeting the STUN protocol: no signatures match STUN keywords or the IANA-assigned UDP port 3478, the magic cookie (`0x2112A442`) defined in RFC 5389 does not appear in any rule, and no heuristic inspects for entropy or attribute-length anomalies in STUN messages.

Consequently, the covert channel produced zero alerts in Snort3, with no traffic classification beyond routine protocol identification. This absence suggests that RFC-compliant protocol abuse remains largely invisible to signature-driven detection systems when the targeted protocol lacks dedicated inspection logic.

8.2.2. Suricata NIDS ruleset analysis

Suricata 8.0.4 was tested with the Emerging Threats (ET) Open ruleset, which includes four active STUN protocol detection rules: `ET INFO STUN Binding Request` (SID: 2016149, port 3478), `ET INFO STUN Binding Response` (SID: 2016150), and two additional rules for non-standard port variants (SID: 2033077, 2033078). All rules employ signature-based detection matching the STUN magic cookie (`0x21 12 A4 42`) and message type indicators (`0x00 01` for requests, `0x01 01` for responses).

Unlike Snort3, Suricata's ET Open ruleset includes signatures that match the STUN magic cookie and message types. Across the covert channel test traffic, the ruleset produced 4574 informational alerts (ET INFO). Under identical evaluation conditions, the same ruleset produced 6459 alerts across the full legitimate corpus from ITC-Netblend-60 (eight STUN-using applications over 16 captures), led by Google Meet (3446) and Telegram (1450). These alerts classify the protocol in use but do not distinguish between legitimate and covert STUN sessions: the same rules fire for any standard WebRTC or VoIP application that uses STUN signaling. No alert indicated malicious activity or payload anomaly.

8.2.3. Fortinet FortiGate appliance results

The Fortinet NGFW appliance identified the underlying STUN protocol via DPI, recording application identifier `STUN` (app ID 16573, category `Network.Service`) in the session logs:

```
"FTNTFGTapp":  "STUN",
"FTNTFGTappcat": "Network.Service",
"FTNTFGTappid": "16573",
"FTNTFGTapprisk":"elevated"
```

Despite this correct protocol classification, no security alert was raised and the channel was not blocked. The appliance treated the session as legitimate STUN signaling; the encrypted payloads within the comprehension-optional attributes were not inspected.

8.3. Portability assessment

To demonstrate that the proposed covert channel represents a systemic vulnerability rather than a device-specific flaw, we conducted a portability assessment across enterprise VoIP vendors and edge-routing platforms. Table 5 summarizes the architectural requirements for hosting the STUN implant. All identified platforms utilize standardized Linux kernels with built-in TUN/TAP support and are subject to documented remote code execution (RCE) vulnerabilities that could serve as initial access vectors. The technique therefore applies beyond the RTC endpoint ecosystem to include the network infrastructure layer: edge routers running embedded Linux are equally viable implant hosts, and are often subject to weaker endpoint monitoring than managed workstations or VoIP phones.

9. Performance evaluation

To quantify the operational characteristics of the proposed covert channel, we conducted a series of performance benchmarks measuring throughput, latency, and protocol overhead. These metrics are necessary to assess the practical viability of the channel for real-world C2 operations, data exfiltration, and network pivoting scenarios.

9.1. Experimental setup

The evaluation combines two complementary configurations. The covert channel was first deployed over a production Internet path (client: Debian GNU/Linux 13, Gigabit Ethernet, AS12338; relay: OVH VPS, AS16276; 23 intermediate hops by `traceroute`) to confirm successful operation and evasion under real network conditions. Throughput measurements, including the MTU sweep in Table 6, were conducted over a direct LAN path using the Yealink T19P E2 as the implant endpoint, where results reflect device CPU and MTU limits rather than path congestion.

9.2. Throughput analysis

Throughput was quantified by varying the `tap0` MTU from 500 to 8000 bytes at a request interval of 10 ms, running `iperf3` in 30 runs of 10 s directly on the Yealink T19P E2 over a LAN path. The request–response mechanism issues a new Binding Request as soon as data is available in the `tap0` buffer, so throughput is governed by the device CPU and the MTU of the virtual interface. Table 6 reports the results (mean ± sample standard deviation).

Throughput grows with MTU, reaching 19.61 Mbps at MTU 8000. At MTU 1400 and 1500 the sender rate is nearly identical at 10.25 Mbps: above 1448 bytes the encapsulated STUN message exceeds the 1500-byte physical MTU and is fragmented by the IP layer, adding overhead. At MTUs of 3000 and above, each STUN message is still fragmented at the IP layer (physical MTU of the LAN path is 1500 bytes), but throughput continues to grow because the payload carried per polling cycle grows faster than the additional per-fragment overhead. Standard deviations are below 0.15 Mbps across all 10 points (coefficient of variation $< 1\%$).

9.3. Latency characterization

Round-Trip Time (RTT) measurements were obtained using ICMP Echo Request/Reply packets (`ping`) transmitted continuously for 100 iterations. Baseline latency was measured directly over the Internet path, while tunnel latency was measured end-to-end through the STUN encapsulation layer. Statistical analysis is presented in Table 7.

The additional 16.2 ms latency introduced by the tunnel can be attributed to three primary factors. First, user-space processing overhead occurs as frames transition between the kernel network stack and the user-space application via a TAP (Layer-2) virtual interface, incurring

Table 5
Portability assessment across major IP phone vendors.

Vendor	Arch.	TUN/TAP	RCE Vector
Yealink	ARM	Validated	CVE-2023-43959 (NIST, 2023b)
Ubiquiti	MIPS	Validated	CVE-2023-2374 (NIST, 2023a)
Grandstream	ARM	Documented (Grandstream Networks, 2023)	CVE-2026-2329 (NIST, 2026)
Snom	ARM/MIPS	Documented (Snom Technology, 2024)	Adv. 20150113-0 (SEC Consult Vulnerability Lab, 2015)

Table 6
Throughput vs. tunnel MTU on Yealink T19P E2 over LAN ($n = 30$ runs of 10 s, polling interval 10 ms).

tap0 MTU (B)	Sender (Mbps) mean ± std	Receiver (Mbps) mean ± std
500	4.11 ± 0.04	3.96 ± 0.04
700	4.92 ± 0.00	4.76 ± 0.02
900	7.18 ± 0.06	6.96 ± 0.06
1100	8.47 ± 0.04	8.21 ± 0.04
1300	9.67 ± 0.06	9.36 ± 0.09
1400	10.25 ± 0.07	9.90 ± 0.06
1500	10.25 ± 0.09	9.87 ± 0.10
3000	15.15 ± 0.08	14.64 ± 0.10
4000	16.48 ± 0.08	16.08 ± 0.08
8000	19.61 ± 0.13	19.05 ± 0.15

Table 7
Latency analysis: Baseline vs. STUN tunnel (100 samples).

Configuration	Mean RTT	Std. Dev.	Jitter
Baseline (Direct Internet)	33.2 ms	2.1 ms	1.8 ms
STUN Covert Tunnel	49.4 ms	3.7 ms	3.2 ms
Protocol Overhead	**+16.2 ms**	**+1.6 ms**	**+1.4 ms**

context-switching penalties. Second, encapsulation cost arises from STUN header construction, attribute encoding, and optional encryption, which add computational delay. Third, polling synchronization introduces half-round-trip delays when the relay server must buffer downlink traffic until the next client heartbeat arrives.

At 49.4 ms, the channel supports both traditional C2 command execution and moderately latency-sensitive operations such as interactive shell sessions and remote desktop pivoting. Note that downlink latency is also bounded by the configured polling interval, since the relay buffers inbound traffic until the next Binding Request arrives.

9.4. Operational validation on MIPS hardware (EdgeRouter X)

To confirm that the covert channel operates viably on the MIPS-based Ubiquiti EdgeRouter X, a 60-second `iperf3` session was conducted through the active STUN tunnel over a production WAN path (client: EdgeRouter X tap0 10.200.200.1; relay: tap0 10.200.200.2). The test was performed with a `tap0` MTU of 1400 bytes. Unlike the Yealink throughput characterization, which was conducted over a controlled LAN path to isolate device CPU as the sole bottleneck, this measurement reflects a WAN deployment and is therefore not directly comparable to Table 6; its purpose is to confirm operational viability rather than to characterize device-level performance limits.

The tunnel sustained a mean sender throughput of 41.9 Mbps and a mean receiver throughput of 41.4 Mbps over the full 60-second interval, with 63 TCP retransmissions consistent with normal WAN congestion-control behavior. Per-interval rates ranged from 39.0 to 43.7 Mbps with no stalls or connection drops.

To isolate the overhead attributable to STUN encapsulation, a baseline measurement was performed over the same WAN path and relay endpoint, bypassing the tunnel entirely. The direct connection gave a stable 243 Mbps (sender and receiver), confirming that the available WAN capacity far exceeds the tunnel throughput. The ratio of tunnel to baseline throughput ($41.9/243 \approx 17\%$) therefore quantifies the overhead introduced by STUN encapsulation, user-space I/O multiplexing, and ChaCha20 encryption on the EdgeRouter X CPU, rather than any link-layer constraint. Both raw `iperf3` logs are provided in the supplementary materials (`results/ubiquiti/`).

10. Detection and countermeasures

Although the proposed channel successfully evades current signature-based defenses, its operational mechanics introduce structural anomalies that allow for the definition of behavioral detection strategies, which we validate empirically against a public traffic dataset in Section 10.1. The most direct indicator lies in the analysis of optional attributes within the `0x8000–0xFFFF` range. While legitimate STUN implementations utilize these fields sporadically for structured metadata, our channel injects large payloads. A second indicator arises from the periodic polling architecture, which introduces detectable temporal patterns. Unlike genuine WebRTC sessions, where packet intervals are irregular and contingent upon media negotiation, the tunnel maintains a consistent cadence to ensure C2 responsiveness. This behavior can be identified through autocorrelation analysis of the inter-arrival times (IAT) of *Binding Requests*. A further contextual anomaly exists: the persistence of STUN transactions without the subsequent establishment of RTP or Secure Real-Time Transport Protocol (SRTP) media flows. Given that ICE positions STUN as a precursor to media transport (Keranen et al., 2018), a prolonged control flow that never transitions to a data session constitutes a strong indicator of covert usage. These detection principles align with prior work on statistical methods for storage-based covert channels. Sattolo and Jaskolka (Sattolo and Jaskolka, 2020) evaluated multiple statistical tests for detecting such channels, concluding that complexity-based metrics (entropy, compression ratios, and repetition counts) provide the most reliable discrimination when the embedding mechanism alters the distributional properties of carrier fields. Similarly, Koziak et al. (2021) demonstrated that integrating steganographic awareness into existing IDS architectures significantly improves detection rates for protocol-embedded covert channels, reinforcing the case for extending current NIDS rulesets with entropy analysis capabilities.

From an infrastructure hardening perspective, the most effective mitigation does not rely on deep content inspection but on flow control. Implementing whitelists for known, high-reputation STUN servers would drastically limit an adversary's ability to connect to external C2 infrastructures. Complementarily, enforcing rate-limiting on UDP/3478 per endpoint would degrade the exfiltration bandwidth of the covert channel without disrupting the legitimate signaling required by organizational VoIP or video conferencing services.

10.1. Empirical analysis of legitimate STUN traffic

To provide an empirical baseline for the detection metrics discussed above, we analyzed the publicly available ITC-Net-blend-60 dataset (Bayat et al., 2024), which comprises 36 GB of raw PCAP traffic captured from 60 Android applications across five network scenarios. The dataset was generated through real human interactions on physical devices and is provided in raw PCAP format.

We processed all 59 usable application captures[1] using Zeek 8.1.1 with the `spicy_STUN` analyzer, which fully parses each unencrypted

[1] One application capture in the dataset contained no parseable network traffic and was excluded from analysis.

Table 8
STUN traffic prevalence in the ITC-Net-blend-60 dataset.

Application	STUN Msgs	With Opt. Attrs.	%
Google Meet	1455	665	45.7
Telegram	822	422	51.3
iGap	487	115	23.6
Snapchat	116	10	8.6
Skype	104	101	97.1
Gmail	87	41	47.1
WhatsApp Messenger	61	4	6.6
Soroush Plus Messenger	53	0	0.0
Total	**3185**	**1358**	**42.6**

STUN message and exposes individual attribute types and values. For every parsed message we record (i) the total number of attributes, (ii) the count and types of comprehension-optional attributes (type ≥ 0x8000), (iii) the combined byte length of their values, and (iv) the Shannon entropy of those values. Table 8 summarizes the applications in which STUN traffic was detected.

Of the 59 applications analyzed, only eight (13.6%) generated any STUN traffic that Zeek could fully parse. Google Meet dominates with 45.7% of all observed messages, followed by Telegram (25.8%) and iGap (15.3%). The presence of STUN signaling in Gmail (47.1% optional attribute ratio) is attributable to integrated WebRTC components for Google Meet and legacy Hangouts services within the mobile application. In total, 42.6% of all legitimate STUN messages already carry at least one comprehension-optional attribute, predominantly `FINGERPRINT`, `ICE-CONTROLLING`, and Google-proprietary extensions such as `GOOG-NETWORK-INFO`. Skype, despite contributing only 104 messages, exhibits the highest optional-attribute ratio (97.1%), with several *unknown* attribute types that fall within the comprehension-optional range.

This distribution has direct implications for the detectability of our covert channel. The routine presence of optional attributes means that a simple "has optional attributes" heuristic would flag nearly half of all legitimate STUN traffic as suspicious. Effective detection must therefore rely on deeper metrics, namely, the Shannon entropy and byte length of attribute values.

Table 9 presents the Shannon entropy and combined byte length of comprehension-optional attribute values for the seven applications that exhibited such attributes. Soroush Plus Messenger is excluded as it carried no optional attributes.

The highest entropy observed in any legitimate STUN message is 4.392 bits, produced by Telegram. Across all 1358 messages with optional attributes, the weighted mean entropy is approximately 2.95 bits. In contrast, attribute payloads carrying ChaCha20-encrypted content exhibit near-uniform byte distributions. This gap of over 3 bits above the legitimate maximum provides a wide detection margin. The entropy threshold of 4.5 bits used in our Zeek-based detector (Section 10) would produce zero false positives on this dataset while reliably flagging covert channel traffic.

10.1.1. Empirical validation against covert channel traffic

To validate the effectiveness of entropy-based detection, we captured and analyzed STUN traffic from our covert channel implementation. Across 3646 data-carrying messages, the channel exhibits mean entropy of 6.247 bits (range 5.154–7.856), a 1.86-bit increase over the highest legitimate value (4.392 bits from Telegram). The minimum observed entropy is still 0.76 bits above the legitimate maximum. The average attribute payload of 190.4 bytes (maximum 1596 bytes) is an order of magnitude larger than the legitimate maximum of 24 bytes observed in Skype.

Applying the 4.5-bit entropy threshold yields perfect classification. Table 10 presents the detection performance when the entropy-based detector is applied across all datasets.

The detection results demonstrate perfect classification: all 3646 data-carrying covert messages exceed the 4.5-bit threshold, while zero legitimate messages do. The 0.76-bit gap between the legitimate maximum (4.392 bits) and the covert minimum (5.154 bits) provides a clear detection margin with zero false positives on the entire 1358-message legitimate dataset. These results confirm that entropy-based detection provides a practical and statistically sound mechanism for identifying STUN-based covert channels in operational networks.

10.1.2. Baseline dataset representativeness

The entropy baseline used to validate the detection threshold above was derived from the ITC-Net-blend-60 dataset, which comprises traffic from 60 Android mobile applications. While this dataset provides a reproducible, peer-reviewed baseline of legitimate STUN traffic from real user interactions, it does not include captures from enterprise SIP endpoints or hardware IP phones, the environment targeted by our covert channel implementation. The STUN attributes generated by a Yealink T19P or a PBX-managed VoIP deployment may differ in type, length, and entropy from those produced by mobile WebRTC applications such as Google Meet or Telegram. To our knowledge, no publicly available dataset currently provides full-payload PCAPs of enterprise VoIP/SIP traffic with unencrypted STUN signaling suitable for this analysis.

10.1.3. Adaptive padding evasion

The entropy-based detector proposed above can be evaded by an attacker who interleaves low-entropy filler bytes within each comprehension-optional attribute value, dragging the per-attribute Shannon entropy below the 4.5 bits/byte threshold while still transporting an encrypted payload. The detector is therefore not a complete countermeasure but a low-cost first line of defense, and remains susceptible to adaptive evasion.

10.2. Network monitoring discrepancy: Signatures vs. Protocol grammar

A relevant finding in our experimental validation is the qualitative gap between signature-based NIDS (Suricata) and protocol-aware analyzers (Zeek with Spicy) when inspecting the covert channel traffic. While both tools observe the same packets, they produce fundamentally different levels of insight.

To quantify this effect, we performed a baseline accounting of our covert channel using three different observation methods: raw packet counting (`tcpdump`), signature-based alerting (Suricata with ET Open rules), and deep protocol analysis (Zeek/Spicy). Table 11 presents the results.

Suricata generates one informational alert (`ET INFO`) for every STUN packet, producing 4574 alerts that are identical in severity and classification to those triggered by any legitimate WebRTC session. `SID 2016149` (Binding Request) fires 2287 times and `SID 2016150` (Binding Response) fires 2287 times, matching the raw packet counts exactly. None of these alerts distinguish between the 929 empty NAT keep-alive polls (which carry no covert data) and the 3646 messages that transport encapsulated Ethernet frames. In contrast, the Spicy-based Zeek analyzer parses each STUN message according to its protocol grammar, exposing the presence, type, and content of individual attributes. Of the 4575 messages Zeek processes, 3646 contain comprehension-optional attributes whose Shannon entropy exceeds 4.5 bits, and the entropy-based detector flags every one of them.

This comparison demonstrates that signature-based NIDS lack the semantic depth required to distinguish covert from legitimate STUN traffic. Suricata treats every packet bearing the magic cookie `0x2112A442` identically, producing a flat stream of informational alerts that offers no triage value. Zeek's protocol grammar, combined with the entropy threshold proposed in Section 10.1, enables targeted detection of the 3646 data-carrying messages while ignoring the 929 benign keep-alives. A NIDS that cannot parse STUN attributes cannot distinguish a WebRTC session from a covert tunnel, regardless of how many rules it applies.

Table 9
Shannon entropy and length of comprehension-optional attribute values.

Application	n[a]	$\bar{H}$ (bits)	H_{min}	H_{max}	$\bar{L}$ (B)[b]	L_{max} (B)[b]
Google Meet	665	2.970	1.500	3.906	10.4	18
Telegram	422	3.004	1.500	4.392	13.1	21
iGap	115	3.388	2.000	3.875	11.9	16
Skype	101	2.542	1.497	2.792	22.3	24
Gmail	41	3.061	2.000	3.906	11.5	18
Snapchat	10	1.585	1.585	1.585	12.0	12
WhatsApp Messenger	4	0.811	0.811	0.811	4.0	4

[a] Messages carrying at least one comprehension-optional attribute (type ≥ 0x8000); messages with no optional attributes are excluded.
[b] Combined byte length of all optional attribute values concatenated per message.

Table 10
Entropy-based detection validation: Legitimate vs. covert channel.

Dataset	STUN Msgs	With Opt.	$\bar{H}$	H_{max}	$H > 4.5$	Detection
Legitimate Traffic (ITC-Net-blend-60)						
Google Meet	1455	665	2.970	3.906	0	0.0%
Telegram	822	422	3.004	4.392	0	0.0%
iGap	487	115	3.388	3.875	0	0.0%
Others	421	156	2.144	3.906	0	0.0%
Legitimate Total	**3185**	**1358**	**2.95**	**4.392**	**0**	**0.0%**
Covert Channel Traffic						
STUN Covert Channel	4575	3646	6.247	7.856	3646	100.0%

Table 11
Security event accounting: Signatures vs. protocol grammar.

Method	Pkt count	STUN Req.	STUN Resp.	Total events
Raw UDP (`tcpdump`)	4575	2287	2287	4575 pkts
Suricata (ET Open)	–	2287	2287	4574 alerts
Zeek (`spicy_STUN`)	–	–	–	3646 data msgs

Table 12
Suricata 8 entropy rule: Threshold sweep on STUN covert channel.

Threshold (bits)	TP	FP	Recall	Precision
4.50	3646	5215	100.00%	41.2%
5.00	3646	3210	100.00%	53.2%
5.25	3643	3018	99.92%	54.7%
5.50	3266	537	89.58%	85.9%
5.75	2049	0	56.20%	100.0%
6.00	1644	0	45.09%	100.0%

10.3. Deploying entropy detection in Suricata 8

The Zeek-based detector of Section 10.1 succeeds because Zeek, through the `spicy_STUN` analyzer, parses each message into its constituent attributes and computes entropy on the value of each comprehension-optional attribute in isolation. A question that directly follows is whether the same behavioral signal can be expressed in a production signature-based NIDS without requiring a dedicated policy engine. Suricata 8 introduces a native `entropy` rule keyword (Open Information Security Foundation, 2025) that computes the Shannon entropy of a payload buffer and supports the full range of comparison operators, making it the first time that a widely deployed signature engine exposes information-theoretic metrics as a first-class primitive.

However, Suricata 8 does not ship a STUN application-layer parser. Unlike SIP, HTTP, or DNS, for which the engine exposes protocol-specific sticky buffers (`sip.header`, `http.uri`, `dns.query.name`, etc.), STUN messages are visible to the detection engine only as raw UDP payloads. As a consequence, the `entropy` keyword cannot be scoped to the value of a single comprehension-optional attribute; the smallest granularity available to a rule is the concatenation of the fixed 20-byte STUN header and every attribute present in the message, reachable via a content anchor on the magic cookie and an `offset` directive. The structural asymmetry between what Zeek's grammar can isolate and what a Suricata rule can address is the central obstacle to porting the detector.

To quantify the impact of this limitation, we implemented a prototype rule pair that anchors on the STUN magic cookie, discards short keep-alives via `dsize:>50`, and applies the `entropy` keyword from byte offset 20 onward (i.e. over the concatenation of all attributes):

```
alert udp any any -> any any (
 content:"|00 01|"; depth:2;
 content:"|21 12 A4 42|"; offset:4; depth:4;
 dsize:>50;
 entropy:offset 20, value >T;
 sid:9000010; rev:1;)
```

A symmetric rule with `content:"|01 01|"` covers Binding Responses. The threshold `T` was swept across six values and each configuration was evaluated against the covert pcap and the complete set of STUN-carrying captures in ITC-Net-blend-60. Table 12 reports true positives (out of 3646 data-carrying covert messages) and false positives across the 3185 legitimate STUN messages.

The sweep exhibits the expected trade-off. At the threshold validated for the Zeek detector (4.5 bits/byte), recall is essentially complete but the false-positive rate is high: the entropy of the full attribute concatenation in legitimate STUN is regularly inflated by hashes, cryptographic fingerprints, and random ICE identifiers (`ICE-CONTROLLING`, `MESSAGE-INTEGRITY`, `FINGERPRINT`) that co-exist with the smaller optional fields Zeek inspects one-at-a-time. As the threshold is raised, false positives drop rapidly while recall degrades comparatively slowly, because the covert channel's minimum attribute entropy (5.154 bits) is substantially larger than the legitimate whole-payload entropy for the overwhelming majority of messages.

Table 13 summarizes the gap between the two detectors. Zeek can address the exact buffer on which the covert-versus-legitimate distinction is sharpest (the value of a single optional attribute) and therefore achieves perfect separation with the lowest threshold. The Suricata rule, operating one layer higher, must raise its threshold to compensate for the coarser buffer and loses both precision and a portion of the recall. The gap is neither a flaw in Suricata's entropy implementation nor a flaw in the rule; it is a consequence of the absence of a STUN application-layer parser in the engine, and would close the day Suricata exposes attribute-level sticky buffers for STUN equivalent to those already available for SIP.

Table 13
Entropy-Based Detection: Zeek Script vs. Suricata 8 Rule.

Property	Zeek Script	Suricata 8 Rule
Entropy scope	Single optional attribute value	Concatenated attributes (offset 20)
Granularity	Per attribute	Per message
Threshold	4.5 bits/byte	5.75 bits/byte
Recall	100.0% (3,646/3,646)	56.20% (2,049/3,646)
False positives	0/1358	0/3185
Requires parser	Yes (`spicy_STUN`)	No (raw UDP payload)
Deployment	Zeek policy script	Standard signature file

Table 14
Detection performance summary across all evaluated NIDS.

Tool	Configuration	TP	FP	Recall	Precision
Snort 3	Talos ruleset (47,143 rules)	0	0	0%	–
Suricata 8	ET Open (standard)[a]	0	0	0%	–
Suricata 8	Entropy rule, $T = 4.5$ bits	3646	5215	100.0%	41.2%
Suricata 8	Entropy rule, $T = 5.75$ bits	2049	0	56.2%	100.0%
Zeek	`spicy_STUN`, $H > 4.5$ bits	3646	0	100.0%	100.0%

[a] ET Open generates 4574 informational (`ET INFO`) alerts on covert traffic and 6459 on legitimate traffic; these identify the protocol in use and do not indicate malicious activity.

In operational terms, the Suricata rule is therefore best positioned as a lightweight, policy-free front-line filter that can be deployed on any Suricata 8 sensor without additional tooling, with the Zeek detector serving as a high-precision second stage where attribute-aware analysis is available. The two detectors are complementary rather than interchangeable: the Suricata rule provides broad coverage at zero additional tooling cost, while the Zeek detector provides precise attribution where a protocol parser is available.

Table 14 consolidates the detection performance of all evaluated platforms. Standard signature-based tools (Snort 3 and Suricata ET Open) do not identify the covert channel. Both entropy-based detectors achieve full recall at threshold 4.5, but Zeek reaches 100% precision owing to attribute-level buffer granularity, whereas Suricata requires a higher threshold to eliminate false positives at the cost of recall.

11. Limitations and discussion

While the proposed STUN-based covert channel evades current production security appliances, several limitations constrain its scope, discussed below.

11.1. Dependence on permissive STUN policies

The evasion model operates effectively only when perimeter security appliances implement a permissive policy that allows STUN traffic as a complete application protocol, without restriction to specific server endpoints. If an organization enforces a strict whitelist of authorized STUN servers, for example, limiting outbound connections exclusively to servers operated by approved conferencing vendors, the proposed channel would be immediately blocked. However, maintaining such granular whitelisting is fundamentally impractical in modern enterprise environments.

Our longitudinal analysis identified over 154,643 publicly accessible STUN servers as of February 2026, distributed across hundreds of cloud providers and ASNs worldwide. This distributed infrastructure reflects the architectural reality of WebRTC-based applications: each platform (whether a web browser, mobile conferencing app, or IoT device) typically operates its own fleet of STUN servers rather than relying on a shared third-party service (Jennings et al., 2021). For instance, a single organization may use Microsoft Teams (which maintains its own STUN infrastructure), Zoom (operating independent servers), third-party WebRTC tools embedded in SaaS platforms, and browser-based applications that query public STUN servers operated by Google, Cloudflare, or regional ISPs.

A network administrator attempting to whitelist all legitimate STUN endpoints would face an impractical maintenance burden: continuously discovering and authorizing servers across disparate vendors, tracking IP address changes due to cloud auto-scaling and load balancing, and responding to service degradation whenever a new application or update introduces previously unknown servers. The operational cost and risk of breaking critical business communications make such policies rare in practice (Shodan Search Engine, 2026). Consequently, the implicit trust granted to STUN as a protocol, rather than to specific server identities, remains the common security posture. The effectiveness of this covert channel is therefore contingent on the continued prevalence of allow-by-default policies for NAT traversal protocols, a condition unlikely to change given the architectural dependencies of modern real-time communications.

11.2. Detectable polling patterns

The request–response architecture necessitates periodic heartbeat transmissions from the implant to maintain the NAT mapping and enable bidirectional communication. When no data is available for transmission, the client must still send empty Binding Requests to poll the relay server for pending downlink traffic. This creates a predictable temporal signature that could be identified through statistical analysis of inter-arrival times. A fixed polling interval (e.g., exactly every 30 s) would constitute a trivial fingerprint for behavioral detection systems. While our current implementation does not incorporate randomization, future iterations could introduce adaptive polling mechanisms that vary the interval based on network activity or inject random jitter to approximate the irregular timing patterns observed in legitimate keep-alive traffic (Keranen et al., 2018). Such enhancements would increase resilience against time-series anomaly detection while maintaining functional reliability.

11.3. Throughput constraints from polling rate

The achievable bandwidth of the covert channel is fundamentally limited by the polling frequency. Increasing the rate of Binding Requests improves latency and throughput but simultaneously elevates the volumetric profile of the traffic, increasing the risk of detection through packet-per-second (PPS) anomaly detection. Conversely, reducing the polling rate to minimize visibility results in higher latency and lower effective bandwidth. Our experimental validation prioritized stealth over performance, selecting conservative polling intervals that blend with typical VoIP keep-alive behavior. For use cases requiring higher throughput (such as real-time video exfiltration or large file transfers) this design constraint would necessitate either accepting elevated detection risk or implementing multi-stream parallelization across multiple STUN endpoints, further complicating operational security.

To assess the viability of rate-based detection, we measured the maximum number of STUN Binding Requests emitted by a single source IP within any 60-second window across both the covert channel and the legitimate ITC-Net-blend-60 dataset. Table 15 presents the results.

Table 15
Maximum STUN binding request rate per source IP (60 s window).

Dataset	Bind. Req.	Max/src/60 s
STUN Covert Channel	2287	2050
Google Meet	3697	207
Gmail	263	157
Telegram	299	118
Soroush Plus Messenger	79	79
iGap	223	37
Skype	115	36
Snapchat	20	8
WhatsApp Messenger	20	20

During active data transfer the covert channel reaches 2050 requests per 60-second window, well above the legitimate peak of 207 observed in Google Meet. However, several factors render a simple rate-based threshold impractical. First, the polling interval is an operator-configurable parameter: reducing it to one request every two seconds would bring the rate below 30 per minute, indistinguishable from any legitimate keep-alive. Second, even the data transfer rate can be throttled at the cost of bandwidth. Third, in deployments behind NAT, a rate-based rule applied after address translation would aggregate STUN traffic from all internal hosts under a single public IP, requiring the sensor to be positioned before the NAT boundary to preserve per-source granularity. These constraints suggest that volumetric detection of STUN-based covert channels is unreliable without complementary statistical inspection such as the entropy-based approach proposed in Section 10.1.

To validate this assessment empirically, we implemented a prototype Suricata 8 rule using the `detection_filter` keyword, configured to alert when a single source IP exceeds 207 Binding Requests within any 60-second window (one above the legitimate maximum observed in Google Meet):

```
alert udp any any -> any any (
 content:"|00 01|"; depth:2;
 content:"|21 12 A4 42|"; offset:4; depth:4;
 detection_filter:track by_src,count 207,seconds 60;
 sid:9000020; rev:1;)
```

Applied to the covert channel capture, the rule generated 1682 alerts against zero false positives on the entire ITC-Net-blend-60 corpus. The 605 non-alerting requests correspond to the detection_filter warm-up period: the first 207 requests within each 60-second window do not trigger the rule, as the threshold has not yet been exceeded. From an operational standpoint this is sufficient: a single alert flags the source IP and session for investigation; per-packet alert count is irrelevant once the anomaly has been raised. Nevertheless, the fundamental evasion remains trivial—reducing the polling interval to two seconds would drop the rate below 30 requests per minute, well within the legitimate range observed across all eight applications.

11.4. Vulnerability to statistical analysis

Although the proposed channel successfully evades signature-based detection and protocol validators, it remains theoretically vulnerable to behavioral analysis techniques not yet widely deployed in production networks. Machine learning classifiers trained on benign STUN traffic distributions could identify anomalies in several observable features. First, the consistent use of near-maximum-size custom attributes (approximately 1450 bytes) diverges from typical STUN exchanges, where attribute payloads are usually minimal (often fewer than 100 bytes). Second, the high entropy of encrypted payloads within custom attributes differs statistically from the low-entropy padding or structured data found in legitimate STUN extensions. Third, the strict periodicity of request–response pairs, even with jitter, may exhibit autocorrelation properties distinguishable from the more irregular patterns of human-initiated RTC sessions. While current DPI engines lack these capabilities (Zander et al., 2007), the integration of entropy analysis and traffic modeling into next-generation security platforms would substantially reduce the operational lifespan of this technique.

11.5. Predictable attribute size patterns

The current implementation prioritizes maximum bandwidth utilization by consistently filling custom attributes to approach MTU limits. This design choice, while maximizing throughput, creates a detectable pattern: every data-bearing packet exhibits a near-identical size. In contrast, legitimate STUN traffic shows wide variability in message sizes depending on the number and types of attributes included (e.g., XOR-MAPPED-ADDRESS, SOFTWARE, FINGERPRINT). A defender employing packet size distribution analysis would observe an anomalous spike at a specific byte range, potentially triggering manual investigation. Future work could address this limitation by implementing dynamic padding schemes that randomize payload sizes across a distribution that mimics legitimate STUN behavior, trading marginal bandwidth efficiency for improved statistical camouflage.

12. Conclusion

We have shown the practical viability of a covert communication channel that exploits the STUN protocol to encapsulate full Layer 2 traffic. Where earlier steganographic methods rely on header manipulation with limited capacity (Zander et al., 2007), our approach uses the extensibility of STUN itself through comprehension-optional attributes, yielding a high-bandwidth transport that remains strictly compliant with RFC 5389 (Rosenberg et al., 2008) and therefore opaque to current DPI engines.

The experimental validation on two heterogeneous resource-constrained platforms (the ARM-based Yealink T19P E2 IP phone and the MIPS-based Ubiquiti EdgeRouter X) shows that the proposed asynchronous event-loop architecture and adaptive polling mechanism can sustain a stable, full-duplex tunnel even under restrictive NAT topologies (Trautwein et al., 2025), and that portability extends beyond VoIP endpoints to network infrastructure equipment. Fortinet FortiGate and Suricata NIDS did not classify the traffic as malicious or block it, and a Palo Alto Networks VM-Series instance behaved consistently. Suricata logged informational STUN protocol events identical to those generated by any legitimate WebRTC session; no appliance identified the covert payload embedded within the custom attributes.

These findings expose an architectural gap in current network defense. The prevailing allow-by-default policy for NAT traversal protocols, driven by the need to support RTC services such as VoIP and WebRTC (Jennings et al., 2021; GSMA, 2025), creates a permissive environment for stealthy C2 operations. Protocol-specific decoders and signature-based detection alone are insufficient to counter advanced persistent threats (Zander et al., 2007): without statistical traffic analysis covering packet-size distribution, entropy and temporal patterns, STUN-based covert channels remain indistinguishable from keep-alive traffic.

Future work will target defensive countermeasures: detection of high-entropy payloads in optional STUN attributes and characterization of the timing signatures of the polling mechanism. As RTC protocols expand their footprint in enterprise networks, the community needs to move beyond protocol-compliance checks and integrate behavioral and statistical inspection into existing NIDS deployments to counter protocol-compliant exfiltration vectors.

Acronyms

C2	Command and Control
DPI	Deep Packet Inspection
ICE	Interactive Connectivity Establishment
IPS	Intrusion Prevention System
L2	Layer 2
ML	Machine Learning
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NGFW	Next-Generation Firewall
NIDS	Network Intrusion Detection System
RTC	Real-Time Communication
RTP	Real-Time Transport Protocol
SIP	Session Initiation Protocol
SPAN	Switched Port Analyzer
SRTP	Secure Real-Time Transport Protocol
STUN	Session Traversal Utilities for NAT
TLV	Type-Length-Value
UTM	Unified Threat Management
VoIP	Voice over IP
WEBRTC	Web Real-Time Communication

CRediT authorship contribution statement

Gorka Gorrochategui: Writing – original draft, Data curation, Conceptualization. **Unai Zulaika:** Writing – review & editing, Investigation, Formal analysis. **Pablo Garaizar:** Writing – review & editing, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the support of the Basque Government under the grant for the DEUSTEK research group.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cose.2026.105043.

Data availability

Research Link Provided

Google Drive Share - PCAPs, Source Code, Scripts, Logs (Original data) (Google Drive Share - PCAPs, Source Code, Scripts, Logs)

References

Barradas, D., Santos, N., Rodrigues, L., 2017. DeltaShaper: Enabling unobservable censorship-resistant TCP tunneling over videoconferencing streams. Proc. Priv. Enhancing Technol. (PoPETS) 2017 (4), 5–22. http://dx.doi.org/10.1515/popets-2017-0037.

Barradas, D., Santos, N., Rodrigues, L., 2018. Effective detection of multimedia protocol tunneling using machine learning. In: Proc. 27th USENIX Security Symposium.

Barradas, D., Santos, N., Rodrigues, L., Nunes, V., 2020. Poking a hole in the wall: Efficient censorship-resistant internet communications by parasitizing on webrtc. In: Proc. ACM SIGSAC Conference on Computer and Communications Security. CCS, http://dx.doi.org/10.1145/3372297.3417874.

Bayat, M., Garshasbi, J., Mehdizadeh, M., Nozari, N., Khesal, A. Rezaei, Dokhaei, M., Teimouri, M., 2024. ITC-Net-blend-60: A comprehensive dataset for robust network traffic classification in diverse environments. BMC Res. Notes 17 (165), http://dx.doi.org/10.1186/s13104-024-06817-5.

Bocovich, C., Breault, A., Fifield, D., Serene, Wang, X., 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In: Proc. 33rd USENIX Security Symposium.

Crosser, A., 2025. Ghost calls: Abusing web conferencing for covert command & control. In: Proc. Black Hat USA 2025. Las Vegas, NV, [Online]. Available: https://www.praetorian.com/blog/ghost-calls-abusing-web-conferencing-for-covert-command-control-part-1-of-2/.

Figueira, G., Barradas, D., Santos, N., 2022. Stegozoa: Enhancing WebRTC covert channels with video steganography for internet censorship circumvention. In: Proc. ACM ASIA CCS. http://dx.doi.org/10.1145/3488932.3517419.

Gianvecchio, S., Wang, H., 2007. Detecting covert timing channels: an entropy-based approach. In: Proc. 14th ACM Conf. Computer and Communications Security. CCS, pp. 307–316. http://dx.doi.org/10.1145/1315245.1315284.

Grandstream Networks, 2023. OpenVPN on grandstream IP phones guide. [Online]. Available: https://documentation.grandstream.com/knowledge-base/openvpn-guide/.

GSMA, 2025. IR.92 IMS Profile for Voice and SMS, Version 22.0. GSM Association, Official Document, [Online]. Available: https://www.gsma.com/newsroom/resources/ir-92/.

Jennings, C., Boström, H., Bruaroey, J-I., 2021. WebRTC 1.0: Real-time communication between browsers. W3C Recommendation, [Online]. Available: https://www.w3.org/TR/webrtc/.

Keranen, A., Holmberg, C., Rosenberg, J., 2018. Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal. IETF, RFC 8445.

Koziak, T., Wasielewska, K., Janicki, A., 2021. How to make an intrusion detection system aware of steganographic transmission. In: Proc. European Interdisciplinary Cybersecurity Conference. EICC'21, http://dx.doi.org/10.1145/3487405.3487421.

Li, Shuai, Schliep, Mike, Hopper, Nick, 2014. Facet: Streaming over Videoconferencing for Censorship Circumvention. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES). ACM, pp. 163–172. http://dx.doi.org/10.1145/2665943.2665944.

Mahy, R., Matthews, P., Rosenberg, J., 2010. Traversal using relays around NAT (TURN): Relay extensions to STUN. RFC 5766.

MAWI Working Group, 1999. Guidelines for protecting user privacy in WIDE traffic archive. https://mawi.wide.ad.jp/mawi/guideline.txt.

MAWI Working Group, 2026. MAWI Working Group Traffic Archive. Measurement and Analysis on the WIDE Internet (MAWI), [Online]. Available: https://mawi.wide.ad.jp/. (Accessed 10 February 2026).

Mazurczyk, W., 2013. VoIP steganography and its detection: A survey. ACM Comput. Surv. 46 (2), 20. http://dx.doi.org/10.1145/2543581.2543587.

Mazurczyk, W., Szczypiorski, K., 2008. Covert channels in SIP for VoIP signalling. In: Jahankhani, H., Revett, K., Palmer-Brown, D. (Eds.), Global E-Security. In: Communications in Computer and Information Science, vol. 12, Springer-Verlag, Berlin, Heidelberg, pp. 65–72.

Mehić, M., Mikulec, M., Voznak, M., Kapicak, L., 2014. Creating covert channel using SIP. In: Multimedia Communications, Services and Security. MCSS, In: CCIS, vol. 429, Springer, pp. 182–192.

NIST, 2023a. CVE-2023-2374: Ubiquiti EdgeRouter X web management interface command injection. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-2374.

NIST, 2023b. CVE-2023-43959: Yealink SIP-T19P_E2 authenticated remote code execution. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-43959.

NIST, 2026. CVE-2026-2329: Grandstream GXP1600 series unauthenticated stack buffer overflow in/cgi-bin/api.values.get. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2026-2329.

Open Information Security Foundation, 2025. Payload keywords: entropy. Suricata 8.0 User Guide. [Online]. Available: https://docs.suricata.io/en/suricata-8.0.0/rules/payload-keywords.html#entropy.

Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., Matthews, P., 2020. Session traversal utilities for NAT (STUN). RFC 8489.

Rosenberg, J., Mahy, R., Matthews, P., Wing, D., 2008. Session traversal utilities for NAT (STUN). IETF, RFC 5389.

Saenger, J., Mazurczyk, W., Keller, J., Caviglione, L., 2020. Voip network covert channels to enhance privacy and information sharing. Future Gener. Comput. Syst. 111, 96–106.

Sattolo, T., Jaskolka, J., 2020. Evaluation of statistical tests for detecting storage-based covert channels. In: Proc. 35th IFIP International Conference on ICT Systems Security and Privacy Protection. SEC, Maribor, Slovenia, pp. 17–31. http://dx.doi.org/10.1007/978-3-030-58201-2_2.

Schmidt, S., Mazurczyk, W., Kulesza, R., Keller, J., Caviglione, L., 2018. Exploiting IP telephony with silence suppression for hidden data transfers. Comput. Secur. 79, 17–32. http://dx.doi.org/10.1016/j.cose.2018.08.006.

SEC Consult Vulnerability Lab, 2015. Multiple critical vulnerabilities in snom IP phones. Advisory 20150113-0. [Online]. Available: https://sec-consult.com/vulnerability-lab/advisory/multiple-critical-vulnerabilities-in-snom-ip-phones/.

Shodan Search Engine, 2026. STUN service discovery results. [Online]. Available: https://www.shodan.io. (Accessed 2 February 2026).

Snom Technology, 2024. Configuring VPN on snom deskphones. [Online]. Available: https://service.snom.com/display/wiki/VPN+Support.

Sommer, R., Amann, J., Hall, S., 2016. Spicy: A unified deep packet inspection framework for safely dissecting all your data. In: Proc. 32nd Annual Computer Security Applications Conference. ACSAC, pp. 558–569. http://dx.doi.org/10.1145/2991079.2991100.

Trautwein, D., Ihle, C., Schubotz, M., Gipp, B., 2025. Challenging tribal knowledge: Large scale measurement campaign on decentralized NAT traversal. arXiv preprint arXiv:2510.27500.

Tsiatsikas, Z., Anagnostopoulos, M., Kambourakis, G., Lambrou, S., Geneiatakis, D., 2015. Hidden in plain sight: SDP-based covert channel for botnet communication. In: Proc. 12th Int. Conf. on Trust, Privacy and Security in Digital Business. TrustBus, In: LNCS, vol. 9264, Springer, pp. 48–59. http://dx.doi.org/10.1007/978-3-319-22906-5_4.

Wendzel, S., Mazurczyk, W., Zander, S., 2016. Unified description for network information hiding methods. J. Univers. Comput. Sci. 22 (11), 1456–1486.

Wendzel, S., Zander, S., Fechner, B., Herdin, C., 2015. Pattern-based survey and categorization of network covert channel techniques. ACM Comput. Surv. 47 (3), 1–26. http://dx.doi.org/10.1145/2684195.

Zander, S., Armitage, G., Branch, P., 2007. A survey of covert channels and countermeasures in computer network protocols. IEEE Commun. Surv. Tutor. 9 (3), 44–57.

Žiža, K., Tadić, P., Vuletić, P., 2023. DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour. Int. J. Inf. Secur. 22 (6), 1865–1880. http://dx.doi.org/10.1007/s10207-023-00723-w.

Gorka Gorrochategui is the founder and CEO of Irontec, a technology company he established in 2003 that specializes in telecommunications and cybersecurity solutions with a strong commitment to open source software. Under his leadership, Irontec has developed and released numerous open source projects, contributing actively to the global open source community. He holds a BSc and MSc in Cybersecurity. He has led key departments within his organization, including Real-Time Communications and Operator Services, as well as Offensive Cybersecurity. With over 20 years of experience bridging the gap between industry and academia, he serves as a guest lecturer at the University of Deusto, where he teaches Ethical Hacking. His research and professional work focus on offensive security and ethical hacking, combining practical industry expertise with academic rigor.

Dr. Unai Zulaika Zurimendi is a Research Associate at DeustoTech, University of Deusto, and a member of the DEUSTEK research group. He earned his PhD in 2022, specializing in Explainable Artificial Intelligence (XAI) and Knowledge Graph algorithms. His research focuses on AI and Natural Language Processing (NLP) for healthcare, specifically clinical data extraction and interoperability within European projects like IDEA4RC and PROTECT CHILD. His work explores interpretable link prediction and the development of NLP-powered pipelines for semantic data extraction.

Dr. Pablo Garaizar is an Associate Professor at the University of Deusto and a member of the DEUSTEK research group. He holds a PhD in Computer Engineering and a BSc in Psychology. He has over 20 years of experience lecturing Programming, High-Performance Computing, Ethical Hacking and Cloud System Architectures subjects, and researching on the development of Computational Thinking in novice programmers, Internet-based research experiments and computer networks.