Covert Channels in the RFC Corpus

A Survey and Measurement Study of Unused, Reserved, and Optional Protocol Fields

Michael J. Bommarito II*
michael.bommarito@gmail.com

August 13, 2026

Abstract

Network protocols are full of forgotten, ignored, or mysterious fields that senders can fill freely, must set to zero, or may omit. Implementations mostly treat them as out of scope, or as arcane edge cases in a test suite. But two parties who agree in advance can turn that zoo of quirks into covert channels that almost never catch the eye.

Prior work picks off one field or one protocol at a time. We screen every RFC from 1 to 9937 and catalog 181 candidate fields across 74 protocols, 176 of them usable in practice. Their capacity is small, but the more useful finding is where it sits. Reserved and must-be-zero bits, the fields most often discussed, are 45.7% of the catalog but only 3.5% of its capacity. A handful of padding, codepoint, and optional fields hold 87.1% of the theoretical capacity. Capacity also concentrates out of a defender's reach. 19.3% of fields ride the wire with cryptographic integrity and hold 42.4% of the total capacity, and no in-path observer can touch those without being noticed.

Yet width on paper is not bytes on the wire, so we implemented all usable channels and tested them with our new open source Python library, Celatim, to confirm byte-for-byte recovery and measure realistic throughput and efficiency between Linux hosts. In addition to Celatim, we release all experimental code and results, so anyone can reproduce the numbers, extend the catalog, or test detection methods.

1 Introduction

How hard is it to hide a message in plain sight on the Internet? The answer, it turns out, is not very.

Take the TCP header. RFC 793 set six bits aside for future use in 1981, under one instruction: must be zero [32]. Explicit Congestion Notification claimed two of them in 2001 [33]. RFC 9293 replaced RFC 793 as the Transmission Control Protocol (TCP) standard in 2022 and carried the remaining four forward unchanged [14]; RFC 9768 spent one more on Accurate ECN in 2026 [6]. Three are left. Forty-five years of revision have cost the field three bits, and the three that survive still carry the instruction they were given in 1981: to this day a sender is supposed to set them to zero, and a receiver is supposed to ignore them unless it has implemented the feature they are reserved for. Encoding data there does violate the sender requirement, yet a cooperating reader recovers the bits with little effort. Three bits is nothing—but they ride in the header of very nearly every TCP segment on the Internet.

*Portions of this work were prepared or revised with assistance from large language models. The author is solely responsible for all content, including any errors or omissions.

This slack has built up, for the most part, deliberately. The Internet's protocols [31, 32] grew over decades from a small community that expected its members to cooperate and its designs to keep changing: be "conservative in what you do" and "liberal in what you accept" [32]. Four decades of backward-compatible, consensus-driven revision have run on that principle. Each round reserves bits for features that mostly never arrive, adds optional fields and extension points, and leaves deprecated ones behind as husks. Unused extension space ossifies (hardens in place) [36]. The countermeasure, GREASE and its kin [3], works by keeping otherwise-unused values in circulation on purpose. No single author designed the result: tolerating the unexpected made the protocols durable, and left them somewhere to hide.

The structure is everywhere. A reserved bit ignored on receipt. A padding region whose contents must be discarded. An opaque token that can carry any value. An optional attribute a router forwards without reading. Simmons named the game the Prisoners' Problem: two prisoners compose messages a warden will forward only if they look innocent [35].

Who the warden is decides whether the channel is a threat or a refuge. To an enterprise defender, a covert channel in a reserved field is how malware moves data past a firewall that inspects only what it understands. To a journalist on a monitored network, the same field is a way out the monitor does not notice. An integrity-protected field a national firewall cannot rewrite without breaking authentication is a blind spot or an obstacle, depending on who you are. The fields are the same either way. Only the adversary's name changes. We measure them once and read what the numbers mean for a defender and for a circumventor.

No one has mapped these fields at corpus scale under a common evidence model. The literature, which we take up in full in §2, either abstracts published techniques into behavioral patterns [40, 42, 27] or dissects one protocol in depth—and one protocol is enough to find dozens: 49 channels in NTP (Network Time Protocol) [20], 22 in IPv6 [25], 20 in QUIC [38]. Attention concentrates on reserved and must-be-zero bits because they make the cleanest illustration. Those bits turn out to be the wrong place to look.

Looking at all 176 usable carriers together shows a pattern that no single-protocol study can see. The total capacity is small, and it is held unevenly. Reserved and must-be-zero bits are the largest class by count, but they hold almost none of the capacity. A few padding, reserved-codepoint, and optional fields hold most of it (Figure 1). A second point is about who can change a field, and here too the news is bad for the defender. The few fields that sit inside integrity protection hold a large share of the capacity, and no observer in the path can rewrite them, because that would break the signature or the authentication. So the fields that are easiest to describe are the worst to use, and the fields that hold the most are the hardest to remove. §3 gives the full distribution and the numbers behind each claim.

A field width is a ceiling, not a measurement, so we do not stop at structure. Every usable catalog row gets a codec round-trip, and 123 of them build a real protocol packet. On executed paths 133 recover their payload byte for byte, 56 of those as parser-visible frames on the wire. Seven run inside their own protocol between two hosts, and production daemons carry two more.

This paper makes four contributions. First, a catalog of 181 covert-channel fields across the RFC series, each anchored to its spec text and sorted into a seven-class taxonomy that cuts across protocol layers (§3, §4).

Second, Celatim, an Apache-2.0 command-line tool and Python package on PyPI and GitHub. It encodes and decodes every usable field, builds real protocol packets for 123, and recovers 133 mechanisms byte for byte on executed paths: 56 over the kernel packet path, seven end to end in their nominal TCP or UDP protocol across a live two-host network, and two through production daemons—real `dnsmasq` DNS and a real OpenSSH key exchange. The same reserved fields survive pass-through, NAT (network address translation), and firewall forwarding across six Linux hosts and

four kernel releases (§5). Readers can rerun any of it or move their own files over the channels.

Third, detection and scrub rules built from the same catalog fields, and a base-rate measurement of where a stateless rule breaks: 0.9974 ROC AUC but 0.023 precision at realistic prevalence (§6).

Fourth, a dual-use reading throughout: each field's defender value and circumventor value come from one set of labels (§7), released under an explicit ethics posture (§7).

Finally, everything is released under permissive license: catalog, measurements, detector and scrub guidance, and the channel code.

2 Related Work

Two lines of work meet here. The survey line organizes published covert-channel techniques into taxonomies. Zander, Armitage, and Branch catalog the techniques known by 2007 across widely deployed network protocols and pair them with detection, elimination, and capacity-limiting countermeasures [42]. Wendzel and colleagues push the abstraction further: 109 techniques published between 1987 and 2013 reduce to 11 hiding patterns, a protocol-independent vocabulary [40]. A 2016 textbook develops that line into a full account of mechanisms and countermeasures [27], and a 2025 taxonomy generalizes it across steganography domains [39]. The unit these surveys organize is the technique, or the pattern behind it.

Our unit is the field, and we sample from the standards corpus rather than the literature. A pattern taxonomy is only as complete as the techniques someone has thought to publish. Sweeping the primary text of RFC 1 through RFC 9937 instead surfaces carrier fields whether or not a channel for them has appeared in print, each anchored to the specification text that creates it. Where the prior surveys sort behaviors into patterns, we measure fields against their specifications: one evidence model across all 181 fields, reporting capacity, survivability, and detectability rather than a category label. Those taxonomies take the reserved bit as their go-to example. The corpus sweep shows that class is the most numerous and the emptiest (§3), visible only when fields are counted over the whole series.

Single-protocol studies show one specification is enough to find dozens: 22 IPv6 channels [25], 49 in NTP [20], 20 in QUIC [38], with focused work on TLS 1.3 [19] and MQTT [28]. Each dissects its protocol by hand. We fold the IPv6, NTP, and QUIC channels into the catalog and check our coverage against them (Appendix A), turning a set of protocol-specific studies into one comparable measurement.

The build-and-run line insists a covert-channel claim be executed, not just asserted [7, 17, 4, 8, 37], and the point that structural capacity is not goodput predates us [30, 42]. What we add is the corpus-scale size of that gap and a channel implementation that measures it. Circumvention systems hide where the traffic goes or what it looks like: domain fronting [16], format-transforming encryption [13], decoy routing [41], and Geneva [4]. We map the carrier fields such systems could use, and stop there: we do not build a circumvention system.

3 Taxonomy

Seven classes cover the catalog. The same specification text that creates a field also fixes how we measure it and how a defender scrubs it (Table 1). The classes cut across protocol layers: a reserved bit is Class A whether it sits in TCP, IPv6, or SCTP (Stream Control Transmission Protocol).

Classes A through E store bits in a field, so their capacity is a width. Class F stores nothing in a field; its capacity is a rate, bounded by the Anantharam–Verdú queue result [2]. Class G hides

Class	Carrier	Example	Scrub
A	Reserved / must-be-zero bits	TCP reserved bits, IPv6 fragment Reserved	zero the field
B	Padding	TLS record padding, ESP TFC, EDNS(0) padding	re-pad
C	Opaque / arbitrary-value	IPv4 ID (atomic), QUIC connection ID	re-randomize
D	Reserved codepoints	TLS/QUIC GREASE, reserved versions	block codepoint
E	Optional / forwarded	BGP optional-transitive attrs, JWT private claims	strip element
F	Timing / count	DNS query timing, QUIC PADDING-frame count	shape timing
G	Subliminal in crypto	RSA-PSS salt, ECDSA nonce	deterministic nonce

Table 1: The seven carrier classes. A–E are storage channels quantified by field width; F is a timing/count channel quantified by a queue/rate model; G is a subliminal channel quantified by signature entropy. The scrub column is the standard-conforming defense, derived from the class.

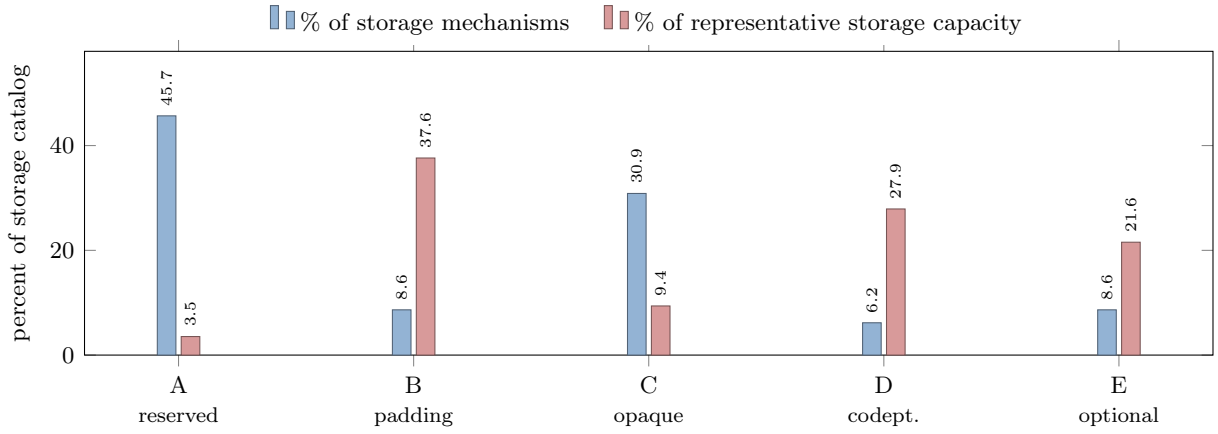

Figure 1: Per carrier class, share of usable mechanisms (count) against share of recorded representative capacity, over the 162 storage mechanisms in Classes A–E. Reserved bits (A) dominate count and hold little capacity; padding, codepoint, and optional fields (B, D, E) are the inverse. Timing and subliminal rows are excluded: different units.

a message in the randomness a signature already needs, bounded by the entropy of that field [35]. These three models use three different units, so we never add them together.

The fields the literature reaches for first are often the poorest. Reserved and must-be-zero bits (Class A), the fields the standard tells a receiver to ignore, are the most numerous class in the catalog: 74 of 162 usable storage fields (45.7%). They hold only 3.5% of the recorded reference capacity, so a defender who zeroes every reserved bit in the catalog clears only that much. The rest sits in three wider classes: padding (Class B, 37.6%), reserved codepoints (Class D, 27.9%), and optional or forwarded elements (Class E, 14 mechanisms holding 21.6%). Together these three hold 87.1% of the recorded capacity, from far fewer mechanisms than Class A alone (Figure 1).

Class D is the sharpest case: 10 mechanisms hold 27.9% of the capacity, almost all of it in three arbitrary-length reserved fields. These are the reserved frame and stream types in HTTP/3 and the reserved transport parameters in QUIC (§4.2). So the split cuts finer than the word "reserved": the reserved *bit* of Class A is nearly empty, while the reserved *codepoint* of Class D is among the fullest carriers. Several of these fields have no fixed width, so their shares depend on the ceiling we assume for the unbounded rows. But across every treatment in Table 11, Classes B, D, and E never fall

below 67.0%.

In absolute terms the capacity is small and right-skewed: most fields hold very little, and a few hold most of it. Half of the 162 usable storage fields carry 24 bits or fewer, and the middle half span 8 to 88 bits. Only 9.3% have no fixed width at all: padding, opaque blobs, and optional elements that carry an arbitrary-length payload. A field is a sliver of the packet. The median one is 0.20% of the full packet on the wire—about 125× smaller than it looks if you count only the header (interquartile range 37.5× to 187.5×). Again, these widths are ceilings, not on-wire goodput; we report both densities and pair every empirical claim with a measured path as discussed in §5.

Who can rewrite a field is a third axis, and it decides which defense is available at all. 83.0% of usable fields are cleartext, so any middlebox (a device in the network path, such as a firewall) that parses them can re-randomize or re-pad them in transit. Authentication removes that option: 19.3% of usable fields are integrity-bound and hold 42.4% of storage capacity that no in-path observer can change without breaking a signature or an authenticated-encryption tag. The capacity a defender most wants to scrub is the capacity it cannot. A checksum does not count, because a middlebox recomputes it. Integrity limits rewriting, not reading: a cleartext authenticated field is still visible, and the flow around it can still be blocked.

4 Catalog

We froze an RFC 1–9937 snapshot (9,738 issued documents) and searched it for the language that usually appears near candidate fields: *reserved*, *must be zero*, *ignored on receipt*, *padding*, *opaque*, *nonce*, *GREASE*, and their variants. That flagged 16,051 candidate passages. The author and GPT 5.4 read the surrounding text before keeping a row, so the unit is a field, not a keyword hit.

A keyword search rarely finds everything, and there is a deeper problem: the definition of the population and its size are open to debate, so the measurement is uncertain from the start.

Where broader protocol-level studies exist (IPv6, NTP, QUIC), we cross-checked our first-pass results against them and added what they had caught and we had missed. The result is 181 catalog entries (176 usable, 5 negative), plus 7 ordinary-payload and 2 non-IETF rows kept for comparison. Every row quotes the spec text that creates the field. The method, the full reconciliation, and a coding-reliability check are in Appendix A.

Table 2 shows the catalog at a glance: 176 usable carriers across 74 protocols and 7 layers. The spread is uneven by design, not by sampling. Application and format specifications contribute the most mechanisms, because there are more of them and they carry more optional structure. Transport contributes the most integrity-bound capacity—almost all QUIC, which authenticates nearly its whole header. Routing and control span the most protocols but hold no integrity-bound carrier at all, so every field there can be rewritten by any speaker on the path. The cryptographic layer is the mirror image: two carriers, both integrity-bound by construction. The per-field rows, with spec quotes and densities, are in Appendix B. The machine-readable catalog is published with the artifact (Appendix D).

4.1 Spec-Acknowledged Channels

Most channels are inferred from a must-ignore clause. Five are not: the RFC names the covert channel in its own text (Table 3). RFC 7837 §9 is the bluntest: it requires the ConEx Destination Option Reserved field to be forwarded unchanged even when non-zero, and calls the result a "4-bit-per-packet covert channel." Its only fix is ESP in tunnel mode, an outer wrapper, because no endpoint or in-path change can close a field the protocol needs forwarded. RFC 7685 §5 zeroes the TLS padding-extension *contents* but admits the padding *length* is a smaller leftover channel—closing the

Layer	Mech.	Prot.	Med. bits	Int-bd	Protocols
Link / encapsulation	18	13	14	1	LISP 3, L2TP 2, MPLS 2, SRv6 2, BIER, GRE, Geneve, IP-TFS, NSH, NVGRE, SMC-R, TRILL, VXLAN
Network	25	4	32	0	IPv6 18, IPv4 4, ICMP 2, ICMPv6
Transport	30	3	24	14	QUIC 22, SCTP 4, TCP 4
Session / security	17	10	31	7	TLS 7, SSH 2, AH, DTLS, ESP, IKEv2, IPComp, ISAKMP, OWAMP, tcpcrypt
Cryptographic	2	2	–	2	ECDSA, RSA-PSS
Routing / control	28	18	16	0	BGP 4, IS-IS 3, OSPF 3, BFD 2, IGMP 2, RIP 2, AMT, BMP, CAPWAP, DLEP, LDP, MLD, MobileIPv6, NHRP, PCEP, PCP, PIM, RSVP
Application / format	56	24	32	10	NTP 18, HTTP/2 5, RTP 5, DNS 4, HTTP/3 4, STUN 2, BinaryHTTP, CoAP, DHCP, Diameter, DoQ, EDHOC, HIP, JWT, Kerberos, OpenPGP, Opus, PPP, PPTP, RTCP, TURN, TZif, UUID, VRRP
Total	176	74		34	

Table 2: Catalog coverage by protocol layer, over the 176 usable primary-population carriers. **Mech.** counts cataloged fields, **Prot.** distinct protocols, **Med. bits** the median structural width over storage classes A–E, and **Int-bd** the carriers whose field rides inside integrity protection. Numbers after a protocol name are its mechanism count; protocols with one are unnumbered. The per-field rows are in Appendix B.

obvious channel leaves a quieter one. The RSA-PSS salt "may provide a covert channel," RFC 8017 Note 5 admits, then cites Simmons and standardizes it anyway. RFC 6437 and RFC 7830 do the same for the IPv6 Flow Label and EDNS(0) padding.

4.2 High-Capacity Carriers

Reserved bits are numerous and thin (§3); the wide, survivable carriers are elsewhere, and one adjective hides the gap. A TCP reserved control bit hands a sender a sliver of one header. An HTTP/3 reserved frame type hands it almost the whole frame: its on-wire density runs near one, because the frame is nearly all covert. Same word, opposite carriers.

HTTP/2 and HTTP/3 reserved frame types, stream types, and settings are the densest fields in the catalog. QUIC adds an opaque connection ID, a NEW_TOKEN blob echoed across connections, PATH_CHALLENGE data, and an unbounded reserved transport parameter that alone holds about a third of the Class D capacity. BGP (Border Gateway Protocol) optional-transitive attributes ride end to end because RFC 4271 §5 makes a speaker that does not recognize an attribute re-advertise it untouched: the default preserves the channel, and scrubbing it is a deliberate departure from the spec. Padding (ESP TFC, TLS records, EDNS(0)) and format carriers (JWT private claims, UUIDv8, Ogg/Opus comment trailers) fill out the wider end.

5 Implementation and Measurement

A matched encoder and decoder can agree on a theoretical encoding that real protocol implementations might reject. To address this issue, we implemented these mechanisms in an open source

RFC	Field	Section	What it says
[1]	IPv6 Flow Label	§6.1	"Covert Channel Risk": the flow label could carry covert data across successive packets; pseudo-random assignment is the stated mitigation.
[24]	TLS ClientHello padding	§5	Padding *contents* could be a covert channel, so they are zeroed; the padding *length* remains a residual, smaller covert channel.
[26]	EDNS(0) padding	§6	Padding octets SHOULD be `0x00` but other values MAY be used, so the padding can be used as a covert channel.
[22]	IPv6 ConEx CDO Reserved	§9	The Reserved bits are forwarded unchanged even when non-zero, giving a "4-bit-per-packet covert channel"; ESP in tunnel mode closes it.
[29]	RSA-PSS salt	§9.1, Note 5	Randomization such as the EMSA-PSS salt "may provide a covert channel"; cites Simmons on subliminal channels.

Table 3: RFCs that acknowledge the covert channel in their own text. These are the strongest specification-grounded cases, and four of the five also name a field-specific mitigation or constraint.

Python package named Celatim. Each field is mapped to a codec, packed into a payload, and sent over a swappable ("pluggable") transport. While some mechanisms do require elevated privileges on some operating systems, the tool requires no root or SYSTEM privileges by default.

The seven carrier classes collapse onto three codec types. A fixed-width value carries Class A reserved bits and small Class C fields. A choose-one-of-k codec carries Class D codepoints and Class F counts. A variable-length byte string carries Class B padding, Class E blobs, and Class G salt. This lookup is keyed to the catalog in Celatim, not 176 hand-written codecs, and every usable field round-trips a payload at exactly its cataloged width.

We test each field at several levels, and we keep the levels apart because they do not prove the same thing. Each level is harder evidence than the one below it. The weakest is structural capacity: the raw width of the field, which is a ceiling, not a measurement. Above it, a codec round-trip encodes a payload and decodes it back in memory (Table 10); all 176 usable fields pass this, and 123 of them go on to build a real protocol packet. Higher still, a packet-path run writes the carrier bytes into a real frame and puts that frame on the wire between two hosts. Higher again, a native run sends the field inside its own protocol, spoken by a real protocol library, between two hosts. The top level, a daemon run, drives production server software.

Across every level we ran, 133 fields recovered their payload exactly. Of those, 56 ran on the packet path, seven reached the native level, and two reached the daemon level. Seven is the top of that count, not the whole of it. We report the highest level each field reached, and we never let one level borrow another's weight. Goodput is a number about our code, not a ceiling on the protocol.

The per-field tables in this section and the appendix show detail for only one carrier per level: the highest-value field, not every field we ran. We ran far more than the tables print. The full record ships with the artifact as machine-readable data (Appendix D). It lists every field that recovered, with its result, its execution path, and a hash of the received payload. The rows printed here are a readable sample of that record, not its full extent.

Seven fields reached the native level, the top of the wire ladder. Here Celatim does not craft a frame by hand. It hands the payload to a real protocol library and lets that library send the

Mechanism	Native protocol path	Pass	Wall	Goodput	CPU	Proto
H2 PING	hyper-h2/TCP	40/40	111.03 ms	18.46 kb/s	22.74 ms	4.9×
QUIC DCID	aioquic/UDP	40/40	155.95 ms	13.14 kb/s	113.08 ms	121.9×
BGP opt-trans	Scapy BGP/TCP	40/40	741.48 ms	2.76 kb/s	379.12 ms	4.5×
EDNS padding	dnspython/UDP	40/40	19.10 ms	111.48 kb/s	16.97 ms	4.0×
RTCP APP	RTCP APP/UDP	40/40	3.48 ms	587.73 kb/s	0.34 ms	2.6×
STUN txid	Scapy STUN/UDP	40/40	759.56 ms	2.70 kb/s	367.35 ms	3.4×
CoAP opt 65000	aiocoap/UDP	40/40	163.30 ms	12.54 kb/s	76.50 ms	1.7×

Table 4: Cross-host native-protocol marquee campaign. Pass counts combine five runs at each of 1, 16, 64, and 256 bytes in both directions (40 per row); metrics are medians for the 256-byte cell. Goodput is recovered application payload per sender endpoint wall time, CPU is sender process time, and Proto is bidirectional serialized application-protocol bytes per payload byte, excluding IP and transport framing. These are implementation-path measurements on one two-host Linux LAN, not protocol limits or a path population.

Payload (B)	Runs	Median time (s)	Median rate (bit/s)
1	18/18	5.5187	1.4
16	18/18	5.2893	24.2
64	18/18	5.1813	98.9
256	18/18	5.2361	391.6

Table 5: Repeated production-daemon EDNS(0) path. Each time and rate covers the paired carrier and separately started control transactions, including daemon and capture orchestration; it is not a network-capacity estimate.

field inside its own protocol, in ordinary TCP or UDP messages, between two x86-64 hosts running Ubuntu 26.04 (Linux 7.0). We ran each field at four payload sizes, five times each, in both directions. The receiving host reads the payload out of the message itself: there is no side channel, and no carrier byte is slipped past the protocol. All 280 runs we kept recovered the payload exactly (Wilson 95% [0.986, 1.000]).

The cost varies widely, and that is the point. At a 256-byte payload, the channel moves between 2.70 and 587.73 kbit/s (median), and every covert byte rides on 1.69 to 121.88 bytes of ordinary protocol traffic—counting the application data only, not the IP and transport headers (Table 4). A few bits hidden in a small header, carried by a large exchange, cost two orders of magnitude in overhead. This is the gap between raw field width and on-wire cost from §3, now measured from end to end.

Two fields reached the very top: they crossed real server software, not just a client library. One drove `dig` against the `dnsmasq` DNS server across six hosts, carrying data in EDNS(0) padding. The other completed a real SSH (Secure Shell) key exchange against `sshd`, hiding data in the KEXINIT cookie. Each passed all 72 of 72 paired trials—one carrier run against one control run, 18 at each of four payload sizes (Tables 5 and 6). The SSH result carries a correction worth recording. An earlier version of our adapter counted a trailing all-zero `uint32` field (RFC 4253) as usable carrier space. A separate review of the protocol caught the mistake. We dropped the 40 affected runs, and the row now counts only the 16-byte cookie.

The subliminal channel (Class G) hides data in randomness a signature already needs. The RSA-PSS salt and the ECDSA nonce are two such fields, so a message hidden there adds no signal an observer can see on the wire, as long as the scheme's assumptions hold. We ran both against real crypto libraries and compared the covert signatures against an honest-random control of 128

Cookies	Payload (B)	Median time (s)	Median rate (bit/s)
1	16	0.1076	1191.7
2	32	0.2222	1152.4
4	64	0.4488	1141.0
8	128	0.8367	1223.8

Table 6: Repeated Paramiko-to-OpenSSH KEXINIT path. Each cookie requires a complete transport key exchange; rate is recovered cookie bits per workflow time, not SSH application goodput.

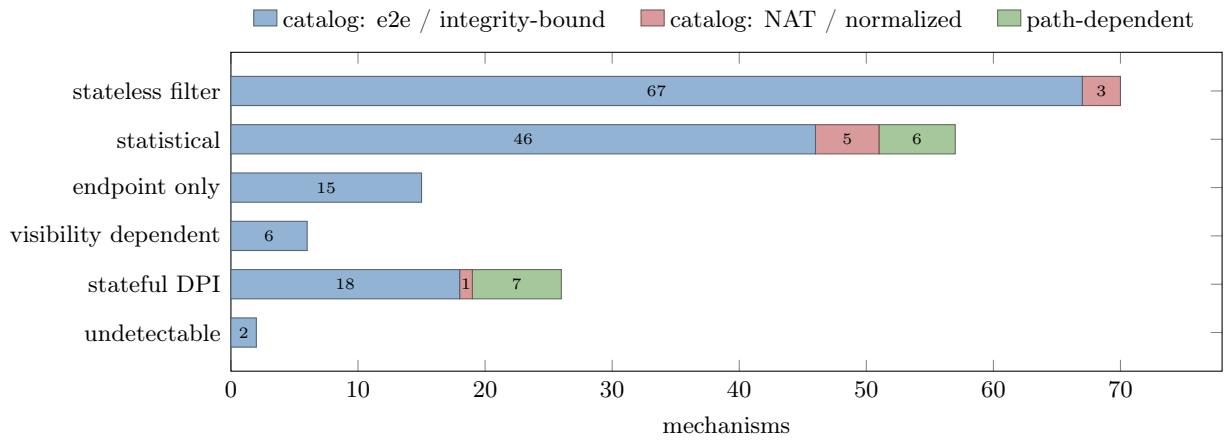

Figure 2: Detectability tier against survivability. The fields most expensive to inspect (encrypted, integrity-bound) tend to survive normalization best: a circumvention advantage that is also the defender's blind spot.

signatures. A two-proportion test on the bit balance of the public signature bytes cannot tell the two apart. This rules out an obvious encoding flaw; it does not prove the two are indistinguishable. We generated only 1 covert signature per mechanism, so the test catches a coarse flaw and nothing finer. The assumption matters: a biased or reused nonce leaks the private key, which is why the fix is a deterministic nonce (RFC 6979), not a scrubber.

The catalog asks whether a field can carry symbols; a usable tool also needs to authenticate its peer, protect the payload, and handle failures. Celatim 0.2.11 ships that layer over a provider contract, and controlled trials establish the transfer contract only—not covertness, performance, or Internet-path survival (Appendix C).

6 Detection and Scrubbing

Detection and scrubbing are the defender's two moves, and the catalog fixes both: how you detect a field, and how you remove it, follow from where its value sits. A reserved bit at a fixed offset can be caught by a stateless packet filter—an exact match on known bits. An opaque field, free by spec to hold any value, has no fixed pattern to match, so telling a covert value from a normal one takes a statistical baseline. An encrypted field is out of reach without the endpoint keys. A subliminal channel, hidden in the randomness a signature already carries, leaves no mark on the wire at all (Figure 2). The same table read from the other side is a circumventor's menu: the fields a defender can least afford to miss are the ones a circumventor most wants to use.

When a field must be zero at a known offset, the detection rule writes itself. The TCP reserved

bits become `tcp[12] & 0x0e != 0` in BPF (the Berkeley Packet Filter language): a fixed offset-and-mask match that keeps no state. Celatim's public repository ships this match compiled to nftables and iptables. It covers 70 of the 83 located Class A and Class D rows. The other 13 sit behind encryption, integrity protection, or deployment-dependent visibility, where no fixed match can reach them. Neither can a fixed match reach an opaque field, which is free by spec to hold any value: only a baseline separates a covert value from a benign one. Those fields need stateful or statistical detectors instead, which Celatim emits as Zeek and Suricata plans rather than one-line rules.

The exact-match rule looks perfect on a test trace and still fails in deployment, for two separate reasons. First, its false-positive rate is understated. We ran the corrected TCP reserved-bit rule through tcpdump, tshark, and Suricata over ten public CSE-CIC-IDS2018 traces of synthetically generated enterprise-testbed traffic, and it matched none of at least 380,804 eligible packets. Zero false positives sounds ideal, but it is a floor, not a deployment estimate: real stacks emit off-spec bits that this synthetic traffic under-represents, so a live network would raise more false alarms, not fewer.

Second, and worse, is the base-rate problem. A statistical detector can score almost perfectly on calibration traffic and still drown its analyst in false alarms once real prevalence is counted in. Our held-out IPv4-ID detector reaches 0.9974 ROC AUC and 0.9879 average precision against a synthetic covert stream as common and as obvious as such a stream ever gets. Yet at the 0.01% prevalence a real sender would use, its precision falls to 0.023: 42 false alarms for every true detection.

Removing a field runs into the same limit as detecting it (the scrub column of Table 1). An in-path normalizer can zero reserved bits, re-pad padding, and re-randomize opaque fields, but only where it can rewrite the field. It cannot touch a field under integrity protection, because rewriting it would break the signature or authentication tag, so the defense has to move to the endpoint. That is why the 42.4% of capacity that rides inside integrity protection is beyond any in-path scrubber. Timing channels (Class F) are the one exception a normalizer can still beat: it can smooth the packet timing, which works even under encryption. The subliminal channel (Class G) can only be closed at design time.

7 Discussion

We executed and recovered 133 mechanisms byte for byte (§5), 56 of them on the wire between two hosts, so the channels work in practice. Survival against a hostile path is a narrower question, and it is the part we measured least. Recovering a payload on our own path is not the same as surviving a censored one. Direct evidence of survival covers only a small set: the seven native and two daemon campaigns on one LAN, plus the pass-through, NAT, and firewall conditions in Table 12. Every other survivability label is an expectation drawn from the middlebox literature [21, 10, 9], tested against a passive in-path normalizer or NAT that rewrites fields to a standard form per packet, not a stateful deep-packet-inspection (DPI) engine that parses whole flows, and not an actively probing censor [15]. A "survivable" label means the field survives passive normalization. It does not mean the field beats a motivated observer.

The same labels a defender uses also tell a circumventor which carriers to prefer. A sender trying to evade censorship wants fields with three properties. They should reach the far end unchanged, and ideally sit inside integrity protection. They should look like ordinary traffic and raise no alarm. And they should be hard to spot, so that only deep packet inspection could catch them. Padding, optional, and high-entropy opaque fields score best on all three. Reserved bits score worst. This ranking is just the defender's detection priority in reverse. The catalog does not cover everything a real tool needs. It does not hide where the traffic goes, imitate normal timing, resist active probing,

or help two parties find each other. Those remain the job of dedicated circumvention systems.

The network experiments cover a small number of Linux hosts and one LAN; the detector cohort is one public dataset's synthetically generated enterprise testbed, day, and time block. One author coded the catalog, so there is no human inter-rater claim. An independent model re-code of a 40-row sample matched every include/exclude call and 30 of 40 class labels (Appendix A)—a reproducibility check, not a reliability study. Structural capacity is a field-width bound throughout, not goodput.

The work is dual-use by construction: the field a defender scrubs is the field that lets someone under surveillance signal past a warden. We take that as a premise, not a disclaimer, and weigh it under the Menlo Report's principles for information and communication technology research [12]. No human subjects are involved.

Defenders gain a spec-grounded map of which fields carry capacity, plus the detection and scrub rules of §6. The same map aids an exfiltrating insider or a censor tuning a scrubber. We judge that the benefit outweighs the harm, because the channels use fields the RFCs already declare open or ignored. These fields are public, and a defender who cannot see the map is worse off than an adversary who can read the RFCs. Building and measuring the channels is what supports the empirical claims, and we release the code. Withholding it would burden independent researchers, defenders, and less-resourced users far more than the well-resourced actors who can already build from the specifications.

Celatim 0.2.11 is research-grade: no at-risk user should rely on it as a deployed circumvention tool. It has no active-probe resistance, deniability, or rendezvous, and the enclosing flow stays blockable. All experiments ran on author-owned infrastructure with no third-party network, production service, or user traffic; the one external dataset is a licensed public trace, redistributed only as hashes. The public bundle excludes keys, offers, transcripts, and raw captures, enforced by a release-hygiene check in CI. We draw an explicit line against Geneva-style probing of live censors.

We expect to revisit some of these limits in future work. Because Celatim and every research artifact are released openly, other researchers can also rerun the experiments, extend the catalog, and help answer the questions we leave open.

8 Conclusion

Hiding a message in plain sight is not very hard, but the room to do it is small, and it is not where the literature has been looking. Reading the RFC series as a corpus, rather than the covert-channel literature, turns a set of scattered techniques into a measured distribution. The capacity in the 176 usable carriers we catalog is small, right-skewed, and unevenly held. The reserved bit dominates the literature and the count, but it holds almost none of the capacity. A few padding, reserved-codepoint, and optional fields hold most of it instead. And the fields hardest to scrub are the ones that ride inside integrity protection. Structural width is only an upper bound, so we built the channels and ran them: exact recovery on all 280 two-host runs, and 72 of 72 paired trials on each of the EDNS(0) and SSH daemon paths. The same catalog reads as detection and scrub guidance: a held-out statistical detector scores 0.9974 ROC AUC but collapses to 0.023 precision at realistic prevalence. What a defender must watch is exactly what a circumventor can use, from one set of measurements.

The measurement is bounded (§7), and Celatim is research-grade, not a deployable circumvention system. We release the catalog, the channel code, the measurements, and the generated detection rules so others can push on exactly those edges—broader paths and middleboxes, adversarial detectors, and per-mechanism survivability measured rather than inferred.

References

[1] Shane Amante, Brian E. Carpenter, Sheng Jiang, and Jarno Rajahalme. IPv6 flow label specification. RFC 6437, IETF, November 2011.

[2] Venkat Anantharam and Sergio Verdú. Bits through queues. *IEEE Transactions on Information Theory*, 42(1):4–18, 1996.

[3] David Benjamin. Applying generate random extensions and sustain extensibility (GREASE) to TLS extensibility. RFC 8701, IETF, January 2020.

[4] Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. Geneva: Evolving censorship evasion strategies. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 2199–2214, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Kenton Born and David Gustafson. Detecting DNS tunnels using character frequency analysis, 2010.

[6] Bob Briscoe, Mirja Kuehlewind, and Richard Scheffenegger. More accurate explicit congestion notification (AccECN) feedback in TCP. RFC 9768, IETF, April 2026. Standards Track.

[7] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: Design and detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 178–187, New York, NY, USA, 2004. Association for Computing Machinery.

[8] Luca Caviglione, Andreas Schaffhauser, Marco Zuppelli, and Wojciech Mazurczyk. IPv6CC: IPv6 covert channels for testing networks against stegomalware and data exfiltration. *SoftwareX*, 17:100975, 2022.

[9] Ana Custura, Raffaello Secchi, and Gorry Fairhurst. Exploring DSCP modification pathologies in the internet. *Computer Communications*, 127:86–94, 2018.

[10] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoit Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 ACM Internet Measurement Conference (IMC)*, pages 1–8, New York, NY, USA, 2013. Association for Computing Machinery.

[11] Biljana Dimitrova and Aleksandra Mileva. Steganography of hypertext transfer protocol version 2 (HTTP/2). *Journal of Computer and Communications*, 5(5):98–111, 2017.

[12] David Dittrich and Erin Kenneally. The Menlo Report: Ethical principles guiding information and communication technology research. Technical report, U.S. Department of Homeland Security, August 2012.

[13] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 61–72, New York, NY, USA, 2013. Association for Computing Machinery.

[14] Wesley M. Eddy. Transmission control protocol (TCP). RFC 9293, IETF, August 2022. STD 7, Internet Standard; obsoletes RFC 793.

[15] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. Examining how the Great Firewall discovers hidden circumvention servers. In *Proceedings of the 2015 Internet Measurement Conference (IMC)*, pages 445–458, New York, NY, USA, 2015. Association for Computing Machinery.

[16] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2015(2):46–64, 2015.

[17] Steven Gianvecchio and Haining Wang. Detecting covert timing channels: An entropy-based approach. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, pages 307–316, New York, NY, USA, 2007. Association for Computing Machinery.

[18] Thomas Grübl, Weijie Niu, Jan von der Assen, and Burkhard Stiller. QUIC-Exfil: Exploiting QUIC's server preferred address feature to perform data exfiltration attacks. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, pages 682–695, New York, NY, USA, 2025. Association for Computing Machinery.

[19] Corinna Heinz, Marco Zuppelli, and Luca Caviglione. Covert channels in Transport Layer Security: Performance and security assessment. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JOWUA)*, 12(4):22–36, 2021.

[20] Jonas Hielscher, Kevin Lamshöft, Christian Krätzer, and Jana Dittmann. A systematic analysis of covert channels in the network time protocol. In *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES)*, pages 1–11, New York, NY, USA, 2021. Association for Computing Machinery.

[21] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 181–194, New York, NY, USA, 2011. Association for Computing Machinery.

[22] Suresh Krishnan, Mirja Kuehlewind, Bob Briscoe, and Carlos Ralli Ucendo. IPv6 destination option for congestion exposure (ConEx). RFC 7837, IETF, May 2016.

[23] Ike Kunze, Constantin Sander, and Klaus Wehrle. Does it spin? on the adoption and use of QUIC's spin bit. In *Proceedings of the 2023 ACM Internet*

Measurement Conference (IMC), pages 554–560, New York, NY, USA, 2023. Association for Computing Machinery.

[24] Adam Langley. A transport layer security (TLS) clienthello padding extension. RFC 7685, IETF, October 2015.

[25] Norka B. Lucena, Grzegorz Lewandowski, and Steve J. Chapin. Covert channels in IPv6. In *Privacy Enhancing Technologies (PET 2005)*, volume 3856 of *LNCS*, pages 147–166, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[26] Alexander Mayrhofer. The EDNS(0) padding option. RFC 7830, IETF, May 2016.

[27] Wojciech Mazurczyk, Steffen Wendzel, Sebastian Zander, Amir Houmansadr, and Krzysztof Szczypiorski. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. Wiley-IEEE Press, Hoboken, NJ, USA, 2016.

[28] Aleksandra Mileva, Aleksandar Velinov, Laura Hartmann, Steffen Wendzel, and Wojciech Mazurczyk. Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. *Computers & Security*, 104:102207, 2021.

[29] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA cryptography specifications version 2.2. RFC 8017, IETF, November 2016.

[30] Steven J. Murdoch and Stephen Lewis. Embedding covert channels into TCP/IP. In *Information Hiding (IH 2005)*, volume 3727 of *LNCS*, pages 247–261, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[31] Jon Postel. Internet protocol. RFC 791, IETF, September 1981.

[32] Jon Postel. Transmission control protocol. RFC 793, IETF, September 1981.

[33] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, IETF, September 2001. Standards Track; updates RFC 793.

[34] Michael Schneider, Daniel Spiekermann, and Jörg Keller. Network covert channels in routing protocols. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES)*, pages 42:1–42:8, New York, NY, USA, 2023. Association for Computing Machinery. Article 42.

[35] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 51–67, New York, NY, USA, 1984. Plenum Press. Presented at CRYPTO '83; proceedings published 1984.

[36] Martin Thomson and Tommy Pauly. Long-term viability of protocol extension mechanisms. RFC 9170, IAB, December 2021.

[37] Michael Carl Tschantz, Sadia Afroz, Name withheld on request, and Vern Paxson. SoK: Towards grounding censorship circumvention in empiricism. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 914–933, San Jose, CA, USA, 2016. IEEE Computer Society.

[38] Aleksandar Velinov, Aleksandra Mileva, Simon Volpert, Sebastian Zillien, and Steffen Wendzel. Steganography in the QUIC communication protocol. *Journal of Universal Computer Science*, 32(2):181–208, 2026.

[39] Steffen Wendzel, Luca Caviglione, Wojciech Mazurczyk, Aleksandra Mileva, Jana Dittmann, Christian Krätzer, Kevin Lamshöft, Claus Vielhauer, Laura Hartmann, Jörg Keller, Tom Neubert, and Sebastian Zillien. A generic taxonomy for steganography methods. *ACM Computing Surveys*, 57(9):1–37, 2025.

[40] Steffen Wendzel, Sebastian Zander, Bernhard Fechner, and Christian Herdin. Pattern-based survey and categorization of network covert channel techniques. *ACM Computing Surveys*, 47(3):50:1–50:26, 2015.

[41] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the network infrastructure. In *20th USENIX Security Symposium (USENIX Security 11)*, San Francisco, CA, August 2011. USENIX Association.

[42] Sebastian Zander, Grenville Armitage, and Philip Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys & Tutorials*, 9(3):44–57, 2007.

A Method and Reliability

The corpus is a frozen snapshot of RFC 1 through RFC 9937 (9,738 issued documents after removing unassigned numbers). It is pinned by SHA-256, together with the index, cutoff, source URLs, PDF-to-text conversion, and ten regular expressions. The generator recorded every match with its RFC, rule, line range, offset, matched text, surrounding paragraph, and a stable identifier: 16,051 candidate passages across 3,668 RFCs. A candidate enters the catalog when the specification exposes a value, symbol, length, timing choice, or optional element a sender controls and a cooperating receiver can recover without an implementation memory-safety bug. Ordinary payload fields are

Analysis population	Usable	Storage	Timing/count	Subliminal
Primary RFC carrier	176	162	12	2
Ordinary payload comparison	7	7	0	0
Non-IETF comparison	2	2	0	0

Table 7: Analysis populations. Headline catalog statistics and figures use only the primary RFC-carrier population. Ordinary application payloads and non-IETF specifications are retained solely as labeled comparisons.

Study	Reported	Scoped	Full	Partial	Gap
IPv6 [25]	22	22	22	0	0
NTP [20]	49	49	38	11	0
QUIC [38]	20	20	18	2	0

Table 8: Channel-level recall reconciliation against focused single-protocol studies. *Full* counts exact and intentionally bundled field-level matches; *Partial* marks overlap that does not represent the whole reported channel; *Gap* is an in-scope study channel without a catalog row. Scoped excludes only explicitly justified out-of-scope entries. The generated audit report preserves every channel disposition, source relationship, and locator.

excluded from the primary population. Fields whose integrity, signature, or header protection prevents the proposed modification are marked negative results. Successor RFCs that restate a field collapse to one canonical mechanism. Each row is coded with one analysis population, carrier class, layer, reach, survivability, on-path visibility, capacity provenance, detector predicate, and false-positive posture. Each is then compared against 17 prior technical notes and labeled for coverage relative to that baseline: DOC (in the baseline), EXT (adjacent to a baseline field), or NEW (not in the baseline). The legacy NEW label is retained to match the released Celatim 0.2.11 catalog. It records absence from a 17-note baseline, not novelty; many NEW rows are long-known channels, so the label carries no priority claim. Table 7 reports the populations.

A focused single-protocol study can reveal whether the initial sweep missed a carrier. The reconciliations against Lucena (IPv6), Hielscher (NTP), and Velinov (QUIC) initially found full catalog coverage of only 6 of 22, 5 of 49, and 5 of 20 channels. We then added the 62 in-scope gap channels as 44 field-level mechanisms (one, the NTP message digest, as an integrity-bound negative result). After reconciliation, full coverage rises to 22 of 22, 38 of 49, and 18 of 20 (Table 8); the remaining entries are partial matches of already-represented fields. These model-assisted, single-coder rows are carried at the structural and codec-round-trip evidence tier and do not enter the cross-host execution counts. The corpus-wide keyword sweep is a sample of unknown bias, not a census: we do not claim complete recall over the RFC series.

Because one author made the final coding decisions, the catalog carries no human inter-rater claim. As a partial check, an independent model pass blind-re-coded a deterministic 40-row sample from the class definitions and specification text alone. It reproduced the include/exclude decision on 40 of 40 rows (Cohen's $\kappa = 1.0$) and the carrier class on 30 of 40 ($\kappa = 0.67$, substantial), with disagreements confined to the opaque/codepoint/optional boundary. This measures coding-scheme reproducibility under independent application, not human inter-rater reliability, which remains future work (`research/coding-reliability.json`).

Table 9 maps each mechanism to its strongest evidence path. Structural capacity is the controllable field width per carrier unit; header-relative and on-wire densities divide it by the protocol header and a typical full packet. Spec-unbounded rows have no finite width, so Table 11 recomputes the

Evidence path	Mechanisms	Claim boundary
AF_PACKET packet path (primary)	56	Parser-visible PDU bytes in generated Ethernet/IPv4 frames over LAN/VXLAN; not native-daemon traffic.
JSON artifact handoff (primary)	77	Alice serializes carrier units in JSON; the harness hands them to Bob over SSH outside the nominal protocol.
Message control exchange (additional subset)	3	Library-built PDU bytes are hex-wrapped in JSON over TCP and parsed on Bob; not native-protocol delivery.
Total unique exact-recovery mechanisms	133	Primary rows only are additive.

Table 9: Where the exact-recovery evidence comes from. The packet and artifact rows partition the 133 unique mechanisms; their sum is a mixed-path implementation-consistency result, not a native-transport count. The 3 message-control rows are additional evidence for a subset and must not be added to that total. PDU/template capability (123 mechanisms) and the 35 codec-only row are also separate from execution on the AF_PACKET path.

Mechanism	Library/control path	Pass	Loss	1 B	256 B
`http2-ping-opaque`	hyper-h2	20/20	0/20	6.322 ms	6.676 ms
`quic-connection-id`	aioquic QUIC	20/20	0/20	73.725 ms	114.687 ms
`http3-reserved-settings`	aioquic H3	20/20	0/20	67.267 ms	68.592 ms
`coap-tunnel`	aiocoap	20/20	0/20	0.065 ms	0.087 ms
`bgp-optional-transitive`	Scapy BGP	20/20	0/20	0.166 ms	0.160 ms
`ecdsa-nonce`	ECDSA	20/20	0/20	0.042 ms	0.046 ms
`rsa-pss-salt`	RSA-PSS	20/20	0/20	0.046 ms	0.055 ms

Table 10: Payload-size sweep for marquee library/control-path scenarios. Each row contains five runs at 1, 16, 64, and 256 bytes; the table shows endpoint medians and reports reliability loss flags. Every row used a same-process topology; elapsed time is a local diagnostic, not native goodput. The campaign therefore establishes implementation repeatability but not cross-host native-daemon survival.

class shares under common caps from 128 to 8,000 bits. The path-conditioned middlebox matrix is Table 12, and the detector cohort is Table 13. Prior single-protocol work maps to the marquee mechanisms in Table 14.

Unbounded-row treatment	A	B	C	D	E	Int.-bd
Recorded row widths	3.5%	37.6%	9.4%	27.9%	21.6%	42.4%
128-bit cap	9.2%	54.6%	23.8%	3.6%	8.8%	41.1%
1,024-bit cap	6.7%	43.6%	21.0%	8.0%	20.7%	35.4%
8,000-bit cap	2.2%	23.1%	15.8%	16.1%	42.8%	25.0%

Table 11: Sensitivity of representative storage-capacity shares to the reference width assigned to spec-unbounded rows, over all five storage classes A–E (each row sums to 100%). The recorded-width row uses the per-mechanism reference values in the catalog; the other rows replace every unbounded value with one common cap. Reserved bits (A) stay near-empty under every treatment, and padding, codepoint, and optional fields (B, D, E) combined stay a large majority; the ranking within B–E shifts with the cap as the unbounded HTTP/3 codepoint rows (D) are truncated and opaque rows (C) rise. The final column is the share carried in integrity-bound fields under the same treatment: resistance to transparent rewriting, not confidentiality.

Mechanism	Path condition	Outcome	Runs	Wilson 95%
`tcp-reserved-bits`	pass-through	preserved	18/18	[0.824, 1.000]
`tcp-reserved-bits`	canonicalize-zero	zeroed	18/18	[0.824, 1.000]
`ipv4-id-atomic`	pass-through	preserved	18/18	[0.824, 1.000]
`ipv4-id-atomic`	canonicalize-zero	zeroed	18/18	[0.824, 1.000]
`ipv4-id-atomic`	MASQUERADE	preserved	18/18	[0.824, 1.000]
`ipv4-id-atomic`	MASQ zero ctl.	zero ctl.	18/18	[0.824, 1.000]
`ipv4-id-atomic`	nft allow	preserved	18/18	[0.824, 1.000]
`ipv4-id-atomic`	nft zero ctl.	zero ctl.	18/18	[0.824, 1.000]

Table 12: Repeated path-conditioned outcomes across six Linux hosts and four kernel releases. Intervals describe execution consistency, not a deployment population. MASQUERADE and nftables preserved the IPv4-ID data and delivered separate field-zero controls at all three taps in every run. Firewall counters and denied probes confirm the default-drop hook was active. Both functions are from Linux Netfilter and do not represent independent vendor products.

Detector	N_-	FPR	N_+	TPR
TCP BPF	380,804	0 [0, .0000101]	1,024	1 [.9963, 1]
TCP tshark	380,851	0 [0, .0000101]	1,024	1 [.9963, 1]
TCP Suricata	380,804	0 [0, .0000101]	1,024	1 [.9963, 1]
IPv4 flag BPF	396,966	0 [0, .0000097]	1,024	1 [.9963, 1]
IPv4-ID window	1,658	.00422 [.00205, .00869]	1,658	.9982 [.9947, .9994]

Table 13: Held-out detector results for one CSE-CIC-IDS2018 pre-attack cohort. Fixed-rule positives contain a nonzero target value in every packet; IPv4-ID positives replace every ID in a 16-packet window. Intervals are Wilson 95%.

Mechanism	Nearest dedicated prior work	Our delta
QUIC fields	QUIC carrier survey and QUIC-Exfil [38, 18]	Extend per field with structural capacity, evidence-tier, survivability, and detector posture; server preferred address remains a recall target.
QUIC spin bit	Known channel [23]	Cited as established, not claimed as new; placed in the catalog for capacity and detectability.
Routing-protocol channels	OSPF channel [34]	BGP optional-transitive attributes are the adjacent gap; we systematize them and execute codec/control paths without claiming multi-AS transit.
DNS tunneling	Saturated detection literature [5]	Novelty is EDNS(0)-padding measurement only, not general DNS tunneling.
HTTP/2	HTTP/2 channels [11]	Field-level catalog with on-wire density and detectability.
BGP optional-transitive	— (gap)	Systematized as a standards-defined forwarding carrier with structural capacity and survivability posture.
EDNS(0) padding	— (gap)	Systematized with codec, daemon-message, and structural capacity evidence.
TLS record padding	— (gap)	Systematized as a spec-permitted carrier.
GREASE as cover	— (gap)	Systematized; reserved-value cover quantified for capacity and detectability.
MQTT comparison row	MQTT 5.0 systematic analysis [28]	Retained as a labeled non-RFC, ordinary-payload comparison rather than counted as evidence of RFC-search coverage.

Table 14: Marquee mechanisms, the nearest dedicated prior work, and our delta. Rows marked "—" identify gaps in the focused literature reviewed here; they are not a claim that no prior implementation exists. Evidence strength remains mechanism-specific.

B Field Catalog

The catalog is published as machine-readable `mechanisms.jsonl` in the public Celatim repository (https://github.com/mjbommar/celatim); the table below is generated from it. It contains all 190 entries: the primary population (176 usable, 5 negative), 7 ordinary-payload, and 2 non-IETF comparisons. The columns are the mechanism name, the governing specification, and the carrier class (A–G). **St.** gives baseline coverage (`DOC` in baseline, `EXT` adjacent, `NEW` not in baseline; the name is a compatibility label, not a novelty claim). **Surv.** gives survivability (`e2e` forwarded, `NAT` rewritten, `path` path-dependent, `norm` normalized, `int-bd` integrity-bound). The last columns are the field width in bits and the header-relative and on-wire densities. Survivability is classified from the literature, not measured per mechanism. Timing and subliminal carriers show - for width-based density.

Primary RFC carrier population

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
IPv4 Identification field (atomic datagrams)	6864, 791	C	NEW	path	16	0.100	0.0013
TCP reserved control bits	9768, 9293, 793	A	EXT	norm	3	0.019	0.0003
QUIC latency spin bit (1-RTT short header)	9000	A	NEW	e2e	1	0.062	0.0001
HTTP/3 reserved frame types (0x1f*N+0x21)	9114	D	NEW	int-bd	9600+	600.000	0.9983
QUIC PADDING frame count/position	9000	F	NEW	int-bd	4	–	–
RSA-PSS signature salt (EMSA-PSS, broadband subliminal)	8017	G	NEW	int-bd	256	–	–
IPv6 Fragment header Reserved + Res	8200, 2460	A	EXT	e2e	10	0.031	0.0008
IPv6 Routing header Type-0 Reserved	8200, 2460	A	EXT	norm	32	0.100	0.0027
IPv6 ConEx Destination Option reserved bits	7837	A	NEW	e2e	4	0.013	0.0003
ICMPv4 unused error-message fields	792	A	EXT	e2e	32	0.500	0.0027
ICMPv6 unused error-message fields	4443, 2463	A	EXT	e2e	32	0.500	0.0027
ICMP Extended Echo Code + Reserved	8335	A	EXT	e2e	16	0.250	0.0013
TCP Urgent Pointer when URG=0	9293, 6093	C	EXT	norm	16	0.100	0.0013
TCP Initial Sequence Number	9293, 6528	C	EXT	NAT	32	0.200	0.0027
TCP Timestamp option low-order TSval bits	7323, 1323	C	EXT	e2e	4	0.025	0.0003
SCTP chunk Flags reserved bits	9260, 4960	A	NEW	e2e	8	0.083	0.0007
SCTP I-DATA Reserved	8260	A	NEW	e2e	16	0.167	0.0013
SCTP Payload Protocol Identifier	9260	C	NEW	e2e	32	0.333	0.0027
SCTP chunk and parameter padding	9260	B	NEW	e2e	24	0.250	0.0020
VXLAN flag and 24+8-bit Reserved	7348	A	NEW	e2e	39	0.609	0.0032
Geneve base-header Rsvd and per-option R	8926	A	NEW	e2e	14	0.219	0.0012
NSH unassigned U-bits and metadata padding	8300	A	NEW	e2e	5	0.078	0.0004
GRE Reserved flag bits	2784, 1701	A	NEW	e2e	8	0.250	0.0007
NVGRE FlowID and Reserved0	7637	A	NEW	e2e	17	0.266	0.0014
L2TPv2 reserved x-bits and offset padding	2661	A	NEW	e2e	6	0.125	0.0005
L2TPv3 control/data x-bits and AVP reserved	3931	A	NEW	e2e	4	0.083	0.0003
LISP data-plane flags, nonce/map-version, LSB	9300, 6830	A	NEW	e2e	35	0.547	0.0029
LISP-GPE N/E/V and reserved bits	9305	A	NEW	e2e	19	0.297	0.0016

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
LISP LCAF reserved fields	8060	A	NEW	e2e	25	0.391	0.0021
SRv6 SRH Flags and Tag	8754	A	NEW	e2e	24	0.075	0.0020
SRv6 SID Argument bits	8986	C	NEW	e2e	32	0.100	0.0027
TRILL header R-bits	6325, 7176	A	NEW	e2e	2	0.042	0.0002
MPLS label EXP/TC bits	3032, 5462	A	NEW	norm	3	0.094	0.0003
MPLS LSP-Ping/SR/RTM MBZ fields	8029, 8012, 8169	A	NEW	e2e	13	0.406	0.0011
BIER Rsv and OAM bits	8296	A	NEW	e2e	10	0.156	0.0008
IP-TFS/AGGFRAG reserved octet	9347	A	NEW	int-bd	8	0.250	0.0007
SMC-R over RDMA reserved fields (LLC)	7609	A	NEW	e2e	64	1.000	0.0053
QUIC Connection ID (DCID/SCID)	9000, 8999	C	DOC	e2e	160	10.000	0.0133
QUIC Bit after grease_quic_bit	9287	D	NEW	e2e	1	0.062	0.0001
QUIC reserved transport parameters	9000	D	NEW	int-bd	8000+	500.000	0.6667
QUIC NEW_TOKEN token blob	9000	C	NEW	int-bd	1024	64.000	0.0853
QUIC PATH_CHALLENGE Data	9000	C	NEW	int-bd	64	4.000	0.0053
QUIC reserved version numbers (GREASE)	9000	D	NEW	e2e	28	1.750	0.0023
QUIC Stateless Reset unpredictable bits	9000	C	NEW	e2e	128	8.000	0.0107
BGP path-attribute flags low 4 bits	4271	A	NEW	e2e	4	0.026	0.0003
BGP unrecognized optional-transitive attributes	4271	E	NEW	e2e	8000+	52.632	0.6667
BGP opaque extended communities	4360	E	NEW	e2e	48	0.316	0.0040
BGP-LS ASLA Reserved	9294	A	NEW	e2e	16	0.105	0.0013
BMP per-peer header reserved flag bits	7854	A	NEW	e2e	5	0.104	0.0004
OSPFv2 AuType=0 Authentication field	2328	C	NEW	e2e	64	0.333	0.0053
OSPFv3 trailing zero header byte	5340	A	NEW	e2e	8	0.062	0.0007
OSPF RI LSA TLV alignment padding	7770	B	NEW	e2e	24	0.125	0.0020
IS-IS ECO and USER ECO fields	1195	A	NEW	e2e	16	0.250	0.0013
IS-IS Reverse Metric reserved flags	8500	A	NEW	e2e	6	0.094	0.0005
IS-IS SRv6 End-SID Flags octet	9352	A	NEW	e2e	8	0.125	0.0007
RIPv2 unused 2-byte header field	1723	A	NEW	e2e	16	0.500	0.0013
RIPv1 ignored route-entry fields	1058	A	NEW	e2e	2000	62.500	0.1667
PIM-SM common-header Reserved	7761	A	NEW	e2e	13	0.406	0.0011
IGMPv3 Resv/Reserved and Auxiliary Data	3376	A	NEW	e2e	24	0.375	0.0020
IGMPv2 Max-Response-Time in Report/Leave	2236	A	NEW	e2e	8	0.125	0.0007
MLDv1 Code/Reserved and Max-Response-Delay	2710	A	NEW	e2e	40	0.625	0.0033
RSVP/RSVP-TE reserved flag/byte and object reserved	2205, 3209	A	NEW	e2e	32	0.500	0.0027
LDP Reserved fields	5036	A	NEW	e2e	16	0.200	0.0013
PCEP common-header Flags and per-object Res	5440	A	NEW	e2e	7	0.219	0.0006
NHRP Resolution flags unused bits	2332	A	NEW	e2e	11	0.172	0.0009
BFD auth-section Reserved	5880	A	NEW	e2e	8	0.042	0.0007
S-BFD Required-Min-Echo-RX (initiator)	7880	C	NEW	e2e	32	0.167	0.0027
DLEP Hop-Count Reserved and data-item reserved	8629, 8175	A	NEW	e2e	7	0.109	0.0006
CAPWAP header flag reserved bits	5415	A	NEW	e2e	3	0.047	0.0003
Mobile IPv6 Mobility-Header reserved fields	6275	A	NEW	e2e	32	0.500	0.0027
PCP request/response Reserved fields	6887	A	NEW	e2e	144	2.250	0.0120

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
AMT Reserved fields	7450	A	NEW	e2e	24	0.375	0.0020
SSH random padding	4253	B	NEW	int-bd	1024	3.200	0.0853
SSH KEXINIT random cookie	4253	C	NEW	e2e	128	0.400	0.0107
TLS 1.3 legacy_session_id	8446	C	NEW	e2e	256	0.800	0.0213
TLS 1.3 legacy_record_version	8446	A	NEW	e2e	16	0.400	0.0013
TLS 1.3 record padding length (TLSInnerPlaintext.zeros)	8446	B	NEW	int-bd	14	0.350	0.0012
TLS GREASE values	8701	D	NEW	e2e	800	20.000	0.0667
TLS ClientHello padding extension length	7685	B	NEW	e2e	9	0.225	0.0008
TLS Heartbeat padding	6520	B	NEW	int-bd	8192	204.800	0.6827
TLS 1.2 gmt_unix_time in Random	5246	C	NEW	norm	32	0.100	0.0027
DTLS 1.3 legacy_record_version and inner padding	9147	A	NEW	e2e	16	0.154	0.0013
ESP Traffic-Flow-Confidentiality padding	4303	B	NEW	int-bd	2040	31.875	0.1700
AH 16-bit Reserved	4302	A	NEW	int-bd	16	0.167	0.0013
IPComp Flags	3173	A	NEW	e2e	8	0.250	0.0007
IKEv2 generic/ID/transform RESERVED fields	7296	A	NEW	int-bd	31	0.138	0.0026
ISAKMP generic-payload RESERVED and Transform RESERVED2	2408	A	NEW	e2e	24	0.107	0.0020
tcpcrypt cres/fres and trailing ignored bytes	8548	A	NEW	int-bd	13	0.081	0.0011
OWAMP packet padding	4656	B	NEW	e2e	1024	9.143	0.0853
HTTP/3 reserved stream types	9114	D	NEW	int-bd	8000+	500.000	0.6667
HTTP/3 reserved SETTINGS values	9114	D	NEW	int-bd	62	3.875	0.0052
HTTP/3 reserved error codes	9114	D	NEW	int-bd	28	1.750	0.0023
HTTP/2 frame-header reserved R bit	9113, 7540	A	EXT	int-bd	1	0.014	0.0001
HTTP/2 DATA/HEADERS/PUSH_PROMISE padding	9113, 7540	B	EXT	int-bd	2040	28.333	0.1700
HTTP/2 PING opaque data	7540	C	EXT	int-bd	64	0.889	0.0053
HTTP/2 unused flag bits	9113	A	EXT	int-bd	6	0.083	0.0005
EDNS(0) Padding option content	7830	B	EXT	e2e	4096	42.667	0.3413
DNS DSO Z-bits and Encryption-Padding TLV	8490	A	EXT	e2e	16	0.167	0.0013
DNS CAA reserved flag bits	8659	A	NEW	e2e	7	0.073	0.0006
DNS-over-QUIC reserved error code	9250	D	NEW	int-bd	32	2.000	0.0027
RTP header-extension appbits	8285, 5285	A	NEW	e2e	4	0.042	0.0003
RTP/RTCP header-extension and RTCP APP data	3550	E	NEW	e2e	800	8.333	0.0667
RTP VP8 R-bit and RSV	7741	A	NEW	e2e	5	0.052	0.0004
RTP VP9 SS reserved bit and LRR RES	9628	A	NEW	e2e	2	0.021	0.0002
RTP H.265/HEVC FU RES bits	7798	A	NEW	e2e	6	0.062	0.0005
RTCP XR reserved/RSV blocks	7867, 7509	A	NEW	e2e	8	0.083	0.0007
STUN attribute padding and transaction ID	8489	C	NEW	e2e	96	0.600	0.0080
TURN CHANNEL-NUMBER and REQUESTED-TRANSPORT RFFU	8656	A	NEW	e2e	40	0.250	0.0033
STUN TRANSACTION_TRANSMIT_COUNTER reserved	7982	A	NEW	e2e	16	0.100	0.0013
HIP LOCATOR_SET reserved bits	8046	A	NEW	e2e	6	0.019	0.0005
Diameter command/AVP flags and AVP padding	6733	A	NEW	e2e	64	0.400	0.0053

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
VRRPv3 rsvd and v2 auth-data	5798	C	NEW	e2e	64	1.000	0.0053
Kerberos KDCOptions unused bits	4120	A	NEW	e2e	4	0.062	0.0003
PPP Magic-Number when not negotiated	1661	C	NEW	e2e	32	1.000	0.0027
PPTP Vendor-Name field	2637	C	NEW	e2e	512	4.000	0.0427
JWT private / unrecognized claims	7519	E	NEW	e2e	8000+	125.000	0.6667
UUIDv8 custom bits	9562	C	NEW	e2e	122	0.953	0.9531
Binary HTTP trailing padding	9292	B	NEW	e2e	8000+	125.000	0.6667
OpenPGP v6 Padding Packet (Type 21)	9580	B	NEW	e2e	8000+	500.000	0.6667
EDHOC EAD padding (ead_label=0)	9528	B	NEW	e2e	1024	16.000	0.0853
TZif unused header region	8536	A	NEW	e2e	120	0.341	0.0100
Ogg/Opus comment-header trailing binary data	7845	E	NEW	e2e	1024	16.000	0.0853
(EC)DSA per-signature nonce k (subliminal)	6979	G	NEW	int-bd	256	–	–
NTP unknown extension-field tunneling	5905, 7822	C	DOC	e2e	3200	8.333	0.2667
NTP inter-arrival / poll timing channel	5905	F	DOC	e2e	4	–	–
DNS query inter-arrival / sequencing timing	1035	F	DOC	e2e	4	–	–
DHCP sname/file/option tunneling	2131	C	DOC	e2e	1536	6.400	0.1280
CoAP token/message-ID/option tunneling	7252	C	DOC	e2e	256	8.000	0.0213
IPv4 TOS/DSCP field	791, 2474	C	NEW	norm	8	0.050	0.0007
IPv4 reserved (evil) flag bit	791, 3514	A	NEW	e2e	1	0.006	0.0001
IPv4 options field tunneling	791	E	NEW	norm	320	2.000	0.0267
IPv6 Flow Label covert channel	6437	C	NEW	e2e	20	0.062	0.0017
HTTP/2 deprecated PRIORITY frame	9113	A	EXT	int-bd	40	0.556	0.0033
OSCORE reserved flag bits (negative result)	8613	A	NEW	norm	1	0.016	0.0001
BGPsec Secure_Path unused flag bits (external-mutation negative result)	8205	A	NEW	int-bd	8	0.053	0.0007
QUIC reserved bits (external-mutation negative result)	9000	A	NEW	int-bd	2	0.125	0.0002
AH Reserved vs external attacker (negative result)	4302	A	NEW	int-bd	16	0.167	0.0013
IPv6 Traffic Class value modulation	8200	C	NEW	path	8	0.025	0.0007
IPv6 Payload Length trailing data	8200	E	NEW	path	800+	2.500	0.0667
IPv6 extension-header presence/omission	8200	E	NEW	path	1	0.003	0.0001
IPv6 Hop Limit value modulation	8200	C	NEW	path	8	0.025	0.0007
IPv6 sender-selected Source Address	8200	C	NEW	NAT	128	0.400	0.0107
IPv6 PadN option content	8200	B	NEW	e2e	304	0.950	0.0253
IPv6 unknown option content	8200	E	NEW	e2e	304+	0.950	0.0253
IPv6 Jumbo Payload Length field	2675, 8200	C	NEW	path	32	0.100	0.0027
IPv6 Router Alert option value	2711, 8200	C	NEW	path	16	0.050	0.0013
IPv6 Routing Header Type 0 address list	8200, 5095	C	NEW	path	384+	1.200	0.0320
IPv6 later-fragment Next Header	8200	C	NEW	e2e	8	0.025	0.0007
IPv6 receiver-discarded fragment insertion	8200	E	NEW	path	304+	0.950	0.0253
IPv6 fabricated Authentication Header	8200, 4302	E	NEW	path	304+	0.950	0.0253
IPv6 fabricated ESP header	8200, 4303	E	NEW	path	304+	0.950	0.0253
NTP Leap Indicator value	5905	C	NEW	e2e	2	0.005	0.0002
NTP Version Number value	5905	C	NEW	e2e	3	0.008	0.0003
NTP reserved Mode values	5905	C	NEW	e2e	3	0.008	0.0003

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
NTP Precision field value	5905	C	NEW	e2e	8	0.021	0.0007
NTP Root Delay field	5905	C	NEW	e2e	32	0.083	0.0027
NTP Root Dispersion field	5905	C	NEW	e2e	32	0.083	0.0027
NTP Reference Identifier value	5905	C	NEW	e2e	32	0.083	0.0027
NTP Kiss-o'-Death code	5905	C	NEW	e2e	32	0.083	0.0027
NTP MAC Key Identifier value	5905	C	NEW	e2e	32	0.083	0.0027
NTP MAC presence/absence	5905	E	NEW	path	1	0.003	0.0001
NTP MAC algorithm selection	5905	D	NEW	path	2	0.005	0.0002
NTP multi-server request ordering	5905	F	NEW	e2e	3	–	–
NTP cross-timestamp redundancy	5905	C	NEW	e2e	16	0.042	0.0013
NTP timestamp low-order fraction bits	5905	C	NEW	e2e	32	0.083	0.0027
NTP unused Receive Timestamp (client mode)	5905	C	NEW	e2e	64	0.167	0.0053
NTP unused Reference Timestamp (client mode)	5905	C	NEW	e2e	64	0.167	0.0053
NTP Message Digest (integrity-bound negative result)	5905	C	NEW	int-bd	128	0.333	0.0107
QUIC Stream ID increment gaps	9000	F	NEW	int-bd	4	–	–
QUIC packet-number increment gaps	9000	F	NEW	int-bd	4	–	–
QUIC non-minimal varint width	9000	C	NEW	int-bd	2	0.125	0.0002
QUIC transport-parameter value modulation	9000	C	NEW	int-bd	16	1.000	0.0013
QUIC Version Negotiation Unused bits	9000	C	NEW	e2e	7	0.438	0.0006
QUIC Stateless Reset Unpredictable Bits	9000	C	NEW	e2e	38+	2.375	0.0032
QUIC intentional packet-loss occurrence	9000	F	NEW	int-bd	2	–	–
QUIC duplicate-frame occurrence	9000	F	NEW	int-bd	2	–	–
QUIC frames-per-packet count	9000	F	NEW	int-bd	3	–	–
QUIC discarded coalesced packet content	9000	E	NEW	e2e	304+	19.000	0.0253
QUIC packet-rate modulation	9000	F	NEW	int-bd	3	–	–
QUIC ACK-frame rate modulation	9000	F	NEW	int-bd	2	–	–
QUIC PING frame occurrence time	9000	F	NEW	int-bd	2	–	–

Hdr. den. is covert bits per header bit and can exceed 1 because capacity accumulates across repeated carrier units; *Wire den.* divides by a full on-wire PDU. Density is defined only for storage classes (A–E); timing/subliminal rows show "–". *St.*: NEW/EXT/DOC prior-art status; *Surv.*: end-to-end (e2e), NAT-rewritten (NAT), path-dependent (path), normalized (norm), or integrity-bound (int-bd).

Ordinary application-payload comparisons

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
DNS TXT record tunneling	1035	E	DOC	e2e	2040+	21.250	0.1700
DNS NULL record tunneling	1035	E	DOC	e2e	8000+	83.333	0.6667
ICMP Echo request/reply data tunneling	792	C	DOC	e2e	8000	125.000	0.6667
DNS-over-HTTPS encapsulation tunneling	8484	E	DOC	int-bd	8000+	500.000	0.6667
HTTP/HTTPS body tunneling	9110	E	DOC	e2e	8000+	111.111	0.6667
WebSocket frame tunneling	6455	E	DOC	int-bd	8000+	500.000	0.6667
WebRTC data-channel tunneling	8831	E	DOC	int-bd	8000+	500.000	0.6667

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.

Hdr. den. is covert bits per header bit and can exceed 1 because capacity accumulates across repeated carrier units; *Wire den.* divides by a full on-wire PDU. Density is defined only for storage classes (A–E); timing/subliminal rows show "–". *St.*: NEW/EXT/DOC prior-art status; *Surv.*: end-to-end (e2e), NAT-rewritten (NAT), path-dependent (path), normalized (norm), or integrity-bound (int-bd).

Non-IETF comparisons

Mechanism	RFC(s)	Class	St.	Surv.	Bits	Hdr. den.	Wire den.
MQTT topic/payload tunneling	OASIS MQTT 5.0	E	DOC	e2e	800	50.000	0.0667
LoRaWAN FPort / frame-payload channel	LoRaWAN L2 1.0.4	C	DOC	e2e	38	0.365	0.0032

Hdr. den. is covert bits per header bit and can exceed 1 because capacity accumulates across repeated carrier units; *Wire den.* divides by a full on-wire PDU. Density is defined only for storage classes (A–E); timing/subliminal rows show "–". *St.*: NEW/EXT/DOC prior-art status; *Surv.*: end-to-end (e2e), NAT-rewritten (NAT), path-dependent (path), normalized (norm), or integrity-bound (int-bd).

C Validation Details

Bob (the receiver) creates a short-lived offer containing a random access token and the fingerprint of a short-lived receiver certificate. Alice (the sender) requires TLS 1.3 and verifies that fingerprint before sending application data. This `offer_bound` mode authenticates the exact offered endpoint, not a person. The receiver acknowledges a chunk only after synchronizing its spool and state. Completion then takes several steps: it checks the whole-file size and SHA-256 hash, places the file at its destination in one atomic move, synchronizes the directory, and sends an authenticated final acknowledgement. Resume reuses the original offer, manifest, provider, chunk plan, and trust mode. Packet-provider file chunks are encrypted before the adapter or privileged helper sees them. The helper retains only `CAP_NET_RAW` and checks peer UID, provider, interface, operation size, and exact flow.

Table 18 summarizes trials originally run with 0.2.0. The same locally built Python package (wheel) was installed on four Linux hosts. Direct TCP/TLS transfers covered two host pairs, both directions, and files from empty through 5 MiB. A separate 32 MiB transfer was stopped after durable acknowledgements and resumed from the original manifest. Four packet providers transferred 1 KiB each between two unprivileged application hosts; direct TLS was the control. Versions 0.2.1 through 0.2.11 change catalog, carrier, and evidence semantics but leave the transfer implementation byte-identical: `src/celatim/transfer` has the same Git tree hash at every one of those tags.

The packet providers place parser-visible carrier packets inside generated outer TCP or UDP frames and are labeled `synthetic_outer_frame`; they do not prove native production stack placement. The campaign does not measure a representative loss distribution, comparative goodput, active-probe resistance, deniability, or Internet-path survival. The identity model authenticates possession of an offer, and the security and privilege boundary still needs independent review before any stable or general-use claim.

Path	Campaign	Supported claim
Direct TCP/TLS	Four transfers; 0 bytes, 64 KiB, 1 MiB, and 5 MiB; two host pairs, both directions	Offer-bound TLS endpoint authentication, durable acknowledgement, final verification, and byte-identical placement.
Interrupted direct	32 MiB; sender process terminated after persisted acknowledgements, then resumed	Original-manifest resume and byte-identical final placement after process interruption.
AF_PACKET carriers	HTTP/2 PING, QUIC connection ID, RTCP APP, and IPv4 ID; 1 KiB each; one host pair	Encrypted provider and capability-bounded packet-service operation over synthetic outer frames; not native-stack placement.

Table 18: Celatim transfer validation, run with 0.2.0; the transfer implementation is unchanged in versions 0.2.1 through 0.2.11. All listed transfers completed with authenticated, acknowledged, and verified receipts and matching source/destination SHA-256 values. These are controlled functional trials, not repeated performance or open-path survivability measurements.

D Artifact Availability

The catalog, channel implementations, measurement code, scenario runners, and the detection/scrub generators are released as Celatim under Apache-2.0 at https://github.com/mjbommar/celatim (this paper describes Celatim 0.2.11). Tagged releases are on GitHub at https://github.com/mjbommar/celatim/releases and on PyPI at https://pypi.org/project/celatim/; Celatim 0.2.11 is pinned to commit `981c53fcf40fbe36b8f99811a86adf0569ae3eb5`, published by the archived trusted-publisher workflow run https://github.com/mjbommar/celatim/actions/runs/29172305929, with wheel SHA-256 `8260d5e690caf2e732b29ee823045880dcf8ac8689b103509033886423a16828` and sdist SHA-256 `b2f162dd46ffd42f426302a504bd9ec098d1485dbcb129204c03bdd5686db494`.

Each headline measurement is backed by an evidence report published with the release that produced it, binding its paired executions to the campaign code, raw manifest, and logs. The EDNS(0) production-daemon path and the corrected SSH KEXINIT path ship with the Celatim daemon campaigns. The detector cohort ships with the detection tooling. Artifacts that hold cookies, raw transcripts, private keys, transfer offers, payloads, and captures that hold environment data are excluded from the public bundle, enforced by a release-hygiene check in CI. The non-privileged reproduction path installs the locked environment, checks lint and types, runs the tests, builds and installs the wheel outside the checkout, verifies the public bundle, and regenerates the paper artifacts from `mechanisms.jsonl`. The privileged AF_PACKET, namespace, middlebox, and daemon experiments are separate manual tiers, documented in the repository.