You Can't Do That on Protocols Anymore: Analysis of Covert Channels in IETF Standards

Luca Caviglione and Wojciech Mazurczyk

Abstract

Information hiding techniques are used by threat actors to elude countermeasures and prevent reversing the attack chain. Recently, they have been deployed to create covert channels, i.e., parasitic communications paths cloaked in network traffic and digital objects. Unfortunately, their detection and mitigation are not simple tasks, especially when information is hidden in network protocols. For instance, revealing the presence of additional data is context-dependent and sanitization could partially impair the traffic. In this paper, we analyze the work of the IETF to evaluate whether risks arising from the presence of covert channels have been considered during the standardization phase. Our findings indicate that the exposure to hidden communications has been addressed only occasionally. We then provide some guidelines to improve the standardization of new protocols and services, especially to prevent the need of deploying *a-posteriori* fixes.

Introduction

Network protocols are not solely used to remotely exchange information. For instance, they are at the basis of inter-process communication mechanisms to prevent monolithic software applications or the need of complex threading architectures. Such a philosophy culminates in microservices, where several functions cooperate for passing data or invoking APIs [1]. Owing to such a pervasive utilization, efforts for guaranteeing the security and privacy of protocols have been doubled, especially in emerging areas such as the IoT [2].

An aspect often neglected concerns the creation of *covert channels* within network traffic [3]. As an example, unused bits in the headers of packets can be manipulated to contain secret data for implementing parasitic paths to bypass firewalls. The diffusion of network protocols magnified the creation of new attack models, especially to implement advanced persistent threats [4]. Covert channels have been successfully used to leak cryptographic keys or prevent detection by hiding malicious communications in ICMP, HTTP, and DNS traffic, as observed in the Nanolocker, DarkHydrus, and Sunburst malware, respectively [5].

In essence, covert channels can void network-wide counter-measures or elude security frameworks, e.g., sandboxes [4], [5]. Unfortunately, the process of cloaking data is tightly coupled to the hiding scheme and targeted protocol. Therefore, mitigating and detecting covert communications cannot be generalized and require threat-dependent approaches [3], [5]. Even if this was feasible in the past, the growing number of attacks, traffic volumes, data structures, and use-cases imposes to act in the early stages of the design process. A possible approach is to use model checking, but it seems ineffective in identifying the most subtle data hiding opportunities [6].

In this perspective, it is desirable to address the ambiguities, imperfect isolation, or unneeded features of protocols already during the standardization process. For the most popular network technologies (e.g., protocols, codecs, encodings, and metadata), this can happen within the Internet Engineering Task Force (IETF). The IETF operates in an open manner to produce standards published through the Request For Comments (RFC) paradigm.[1] Although its working model is effective for isolating and fixing issues, this is not always possible, as too rigid standards may render a protocol difficult to implement or deploy. Preventing the creation of covert channels should then equalize pros and cons, e.g., the attack surface versus the availability of protocol extensions.

The work of IETF has already been the goal of previous analyses, even if not always focusing on security. For instance, [7] investigates how standardization can steer the innovation process and help to reach an equilibrium among competitors. Instead, [8] models the standardization pipeline via the throughput of RFCs. Concerning technological aspects, [9] identifies the gaps that IETF should fill to develop more privacy-preserving protocols and take advantage of emerging cryptographic techniques. Indeed, protocols, data, and practices developed by the IETF have major security implications. The literature usually addresses them vertically and focuses on specific setups or technological areas. For example, the influence of RFCs on the security posture of IoT ecosystems has been extensively investigated for the Transport Layer Security (TLS) and Constrained Application

[1] RFCs can be searched and retrieved from the RFC Editor at the URL: https://www.rfc-editor.org/retrieve/. They can also be directly downloaded by using an URL with the following scheme: https://www.ietf.org/rfc/rfcXXXX.txt, where XXXX is the number of the desired document.

Luca Caviglione is with the National Research Council of Italy (CNR), Institute for Applied Mathematics and Information Technologies "Enrico Magenes" (IMATI), 16149 Genova, Italy; Wojciech Mazurczyk (corresponding author) is with the Institute of Computer Science, Warsaw University of Technology, 00-665 Warsaw, Poland.

Digital Object Identifier: 10.1109/MNET.2024.3352411
Date of Current Version: 16 September 2024
Date of Publication: 10 January 2024

Protocol [10]. However, to the best of our knowledge, there has been no prior research on the impact of covert channels through the lens of the IETF standardization process. To advance the understanding of Internet security, we investigate how developers and engineers have addressed covert communications over the years. This can complement previous efforts quantifying the susceptibility to data hiding of real traffic [11].

Therefore, this paper is devoted to: *i*) identify the major threat models; *ii*) show how covert channels have been considered in the standardization process also in terms of mitigation; *iii*) quantify the awareness of developers and engineers on information hiding; *iv*) improve the vision of the IETF on covert communications within protocols or software objects; *v*) outline a set of recommendations to make future standards more resistant against data hiding attempts.

Threat Models

As discussed, covert channels enable two or more endpoints to communicate in a hidden manner. They have two basic scopes: *local covert channels* allow the uncontrolled data exchange among processes running on the same hosts, whereas *network covert channels* permit the remote transmission of arbitrary information, e.g., Internet-wide. To this aim, the communicating endpoints, i.e., the *covert sender* and the *covert receiver*, utilize a suitable *carrier* to host the secret data. The literature abounds of methods for hiding information by manipulating protocol fields, unused bits, traffic traits, and timing at which a syscall is invoked [3]. Despite the targeted carrier, two main types of covert channels exist. The first are *storage channels*: in this case, the secret information is directly inserted in the selected carrier, e.g., by overwriting the `Flow Label` of IPv6 datagrams. The second are *timing channels*: in this case, the secret information is modulated according to the evolution of a temporal behavior, e.g., a Morse-like code embedded in suitable values of the `Time To Live` of IPv4 datagrams. Obviously, the hiding process should not reveal the presence of the secret or disrupt the functionalities of the protocol. At the same time, the more data hidden, the smaller the undetectability of the approach, thus a suitable trade-off should be searched for [11]. The outcome of the hiding process is that the covert endpoints can communicate through a path cloaked within a legitimate *overt* flow of software objects or protocol data units.

Figure 1 depicts the threat models leveraging a covert channel that have been observed in the work of the IETF. For the sake of brevity, we will only refer to the network case, but they can be straightforwardly extended to the local case. The first scenario (denoted with 1) deals with two covert endpoints hiding data in network traffic, which is the typical attack template considered

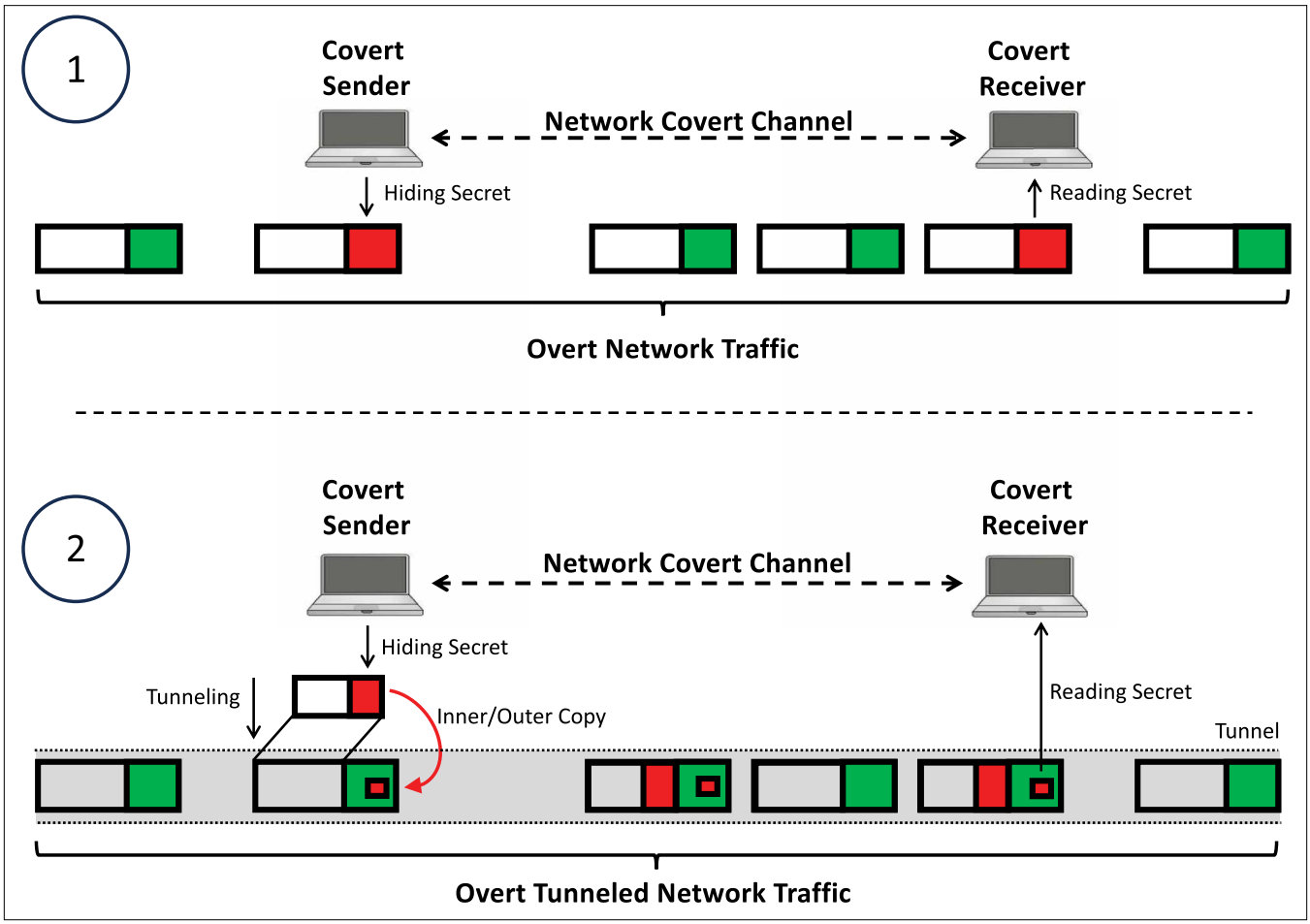

FIGURE 1. Main threat models observed in IETF standards when addressing network covert channels.

in the literature [3]. In this case, two malicious nodes act in a Man-in-the-Middle manner to bypass a middlebox, implement a Command & Control (C&C) path, or exfiltrate information [4]. The second scenario (denoted with 2) recurred several times in documents focused on the "routability" of control data through tunneled traffic, e.g., in RFCs 6040 and 7837. In this case, two malicious endpoints exploit the need of copying status information in ingress/egress nodes to not disrupt network-wide functionalities, for instance, to handle congestion. The covert sender could hide the data in a suitable part of the header of packets to be tunneled (i.e., the inner header). This information is then copied to the header of packets implementing the tunnel (i.e., the outer header). The covert channel is created via the inner/outer copy mechanism, thus allowing the sender to "evade" the encapsulation and leak data towards a receiver, even in the presence of encryption.

The IETF Standardization Process

The IETF does not require any membership or fees, and it largely relies on groups of volunteers participating in the decision-making process. From an organizational viewpoint, the Internet Society interacts with the IETF Administrative Support Activity to guarantee the success of the whole process. The technical management is performed by the Internet Engineering Steering Group (IESG). One of its crucial tasks is to oversee the outputs of all the Working Groups (WGs) to prevent inconsistencies among protocols and technologies. The IETF also interacts with the Internet Architecture Board, which focuses on long-term planning and harmonizes the various standardization activities. The Internet Assigned Numbers Authority supervises the use of global and unique identifiers.

New ideas are submitted through Internet Drafts (I-Ds), which can originate from either a WG or an individual. The various I-Ds typically undergo evaluation and require approval as a WG item, although the final decision must be issued by the Chairs of the WG. Once a draft receives the agreement during the so-called "WG Last Call", it moves to the IESG for further updates. The document is then made available to the entire IETF community for review and for fixing major issues. After receiving the approval, the I-D is published as an RFC by the RFC Editor. Figure 2 summarizes the process starting from the publication of an I-D to the release of the final document.

From a security point of view, RFC 2223 provides detailed instructions for participating in the IETF standardization process. Specifically, each I-D must include a dedicated section called "Security Considerations" outlining potential risks, threats, and vulnerabilities of the proposed ideas, along with any sensitive information that could be exposed or inferred. The section should also include strategies or mechanisms to counteract or minimize all the identified security issues. The importance of such details is evidenced by RFCs 3552 and 9416, which reaffirm requirements and best practices to make the security assessment of standards less weak, as happened in the past. Since security is a prime driver for the success of protocols and techniques, the "Security Consideration" section is reviewed by the experts of the Security Area Directorate.

To evaluate how the IETF addressed the problem of covert communications, we surveyed all RFCs published until August 2023.

Methodology

To evaluate how the IETF addressed the problem of covert communications, we surveyed all RFCs published until August 2023. As a first step, we searched for documents explicitly mentioning `covert channels`. To have an authoritative guideline, we also used RFC 2828 and its updated version RFC 4949, which provide a glossary of Internet security aspects. Due to this investigation, we were able to pinpoint precisely when the IETF refers to the threat models shown in Figure 1. As a result, we refined our analysis by searching for standards containing the following keywords: `storage channel, covert data channel, covert storage channel, covert conversations,` and `covert data`. Using the glossary and further evaluating RFCs devoted to clarifying the internals of the TLS, we found that some documents explicitly mention mechanisms to conceal information in temporal behaviors [12]. This has been observed in RFCs proposing the

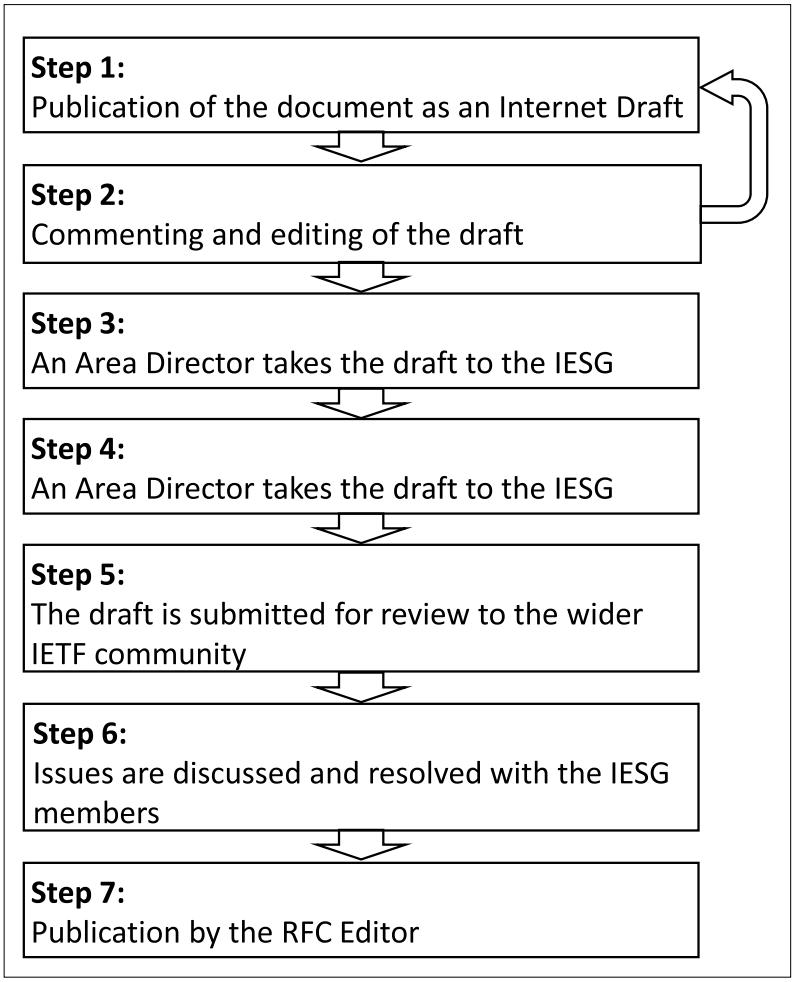

FIGURE 2. Basic steps of the IETF standardization process.

adoption of cryptography to enhance the security of many protocols or settings. Consequently, we searched for RFCs containing the keywords `timing channel` and `covert timing channel`.

For all the queries, we analyzed the threat model taking advantage of the covert channel and the main functionalities used to conceal the information. This served as a consistency check and allowed to gain additional knowledge on the context where the protocol or the software component had been originally imagined. We discovered that some standards suggest abuses vaguely resembling the creation of a covert channel. To trace such use cases, we considered RFCs containing the keywords: `exfiltration`, `timing attack`, `leak information`, and `information leakage`. Additionally, we investigated an expanded set of terms, including those used as a synonym for covert channels or in a misleading manner. Therefore, we also considered RFCs containing keywords like `backchannel`, `out of band`, and `tunneling`. For the case of `tunneling` we further checked whether the security posture of the standard could be brought back to the information hiding use case. Such an investigation did not provide relevant outcomes, apart from some overlaps with RFCs already shortlisted with the previous queries.

As a result, we obtained 76 documents that exhibit many intersections or provide a minimal amount of detail. Some RFCs contained duplicated data, since tied via an obsolete/update relationship. Other documents only mention the possibility of creating a covert channel or hint that such a threat should be assumed as unlikely. For instance, this is the case of RFC 8270 updating RFC 4419, which briefly hints at the feasibility of establishing a covert channel during the Diffie-Hellman group exchange key transmission. When a standard document has been regarded as relevant, we searched for two different contributions. The first is the analysis of how the covert channel can be created, while the second is the presence of some countermeasures or mitigation techniques. For the sake of completeness, Table 1 presents RFCs addressing covert channels in a very general or minimal manner: such documents will not be further considered in the following, unless needed.

Results

This section discusses how the IETF addresses covert channels through three main dimensions. The first considers the *temporal* evolution to analyze if the awareness evolved in time. The second concentrates on *protocols* to outline the core hiding mechanisms identified by the various WGs. The last addresses *countermeasures* to check whether covert channels have been properly regarded also by proposing some workarounds.

Temporal Evolution

To quantify how the awareness of IETF evolved over time, we reviewed all the RFCs shortlisted through the previously-defined keywords. Figure 3 reports the overall temporal distribution. From 1968 to 1990, no RFCs hinted at covert communications, thus, we omitted such a time frame for the sake of clarity. Covert channels have first been mentioned in 1991 in RFC 1287, which reviewed the desired security properties of the future Internet and explicitly stated that some kind of countermeasure against them would be needed.

A major milestone in terms of awareness occurred in 2000 with the introduction of the IETF glossary. Specifically, RFC 2828 provided basic definitions of covert channels, including *timing* and *storage* variants. Despite its importance, RFC 2828 suggested that covert channels were primarily used to leak confidential information between local processes rather than to implement network-wide attacks. The RFC 3552 released in 2003 is an important landmark, as it entails that

RFC(s)	Prot./Funct.	Mention	Obsolete	Update	Keyword
9411	Benchmark for Secure Devices	Issues due to exfiltration are mentioned only in a very general manner, which cannot be related to covert channels	3511	-	Exfiltration
9325	TLS/DTLS	Exfiltration is primarily used to infer information closer to the concept of side channels rather than covert channels	7525	5288 6066	Exfiltration
9210	DNS	Not very specific, but mention that TCP/DNS should be monitored and logged since used for many exfiltration attempts	-	1123 1536	Exfiltration
8404	Pervasive Encryption	Report that a malware can hide information for C&C and data exfiltration	-	-	Exfiltration
7937	Logging for CDNs	Mention that logging operations can be relevant for exfiltration attempts	-	-	Exfiltration
7624	Pervasive Surveillance	Out of scope, but it defines what "exfiltration" is for part of the crypto community	-	-	Exfiltration
8250	IPv6	Hint at covert channels targeting the `Performance and Diagnostic Metrics Destination` options, but the exfiltration case is not relevant for our investigation	-	-	Timing Attack
5240	Protocol Independent Multicast	Provide interesting details about timing channels, but the considered attack case is not relevant for covert communications	-	-	Timing Attack
6378, 6427, 6435	MPLS	Simply mentioned that the `Generic Associated Channel` can be abused to create covert communications	-	-	Covert Channel
5848	Signed Syslog Messages	The standard ignores covert channels and clearly states that nothing has been done to prevent them. It also reports that every message can be used to conceal data, e.g., `Priority Values` can be altered as Morse-code-like signals	-	-	Covert Channel
5246	TLS	Contains general information on the implementation of timing channels	-	-	Information Leakage
4949	Glossary	Adds a minimal tutorial on covert channels and redefines "storage channels" and "timing channels" as "covert storage channels" and "covert timing channels", respectively. The new definitions are more general and a conceptual separation between network covert channels and tunneling is also provided	2828	-	Information Leakage

TABLE 1. RFCs briefly mentioning covert channels without further specific details.

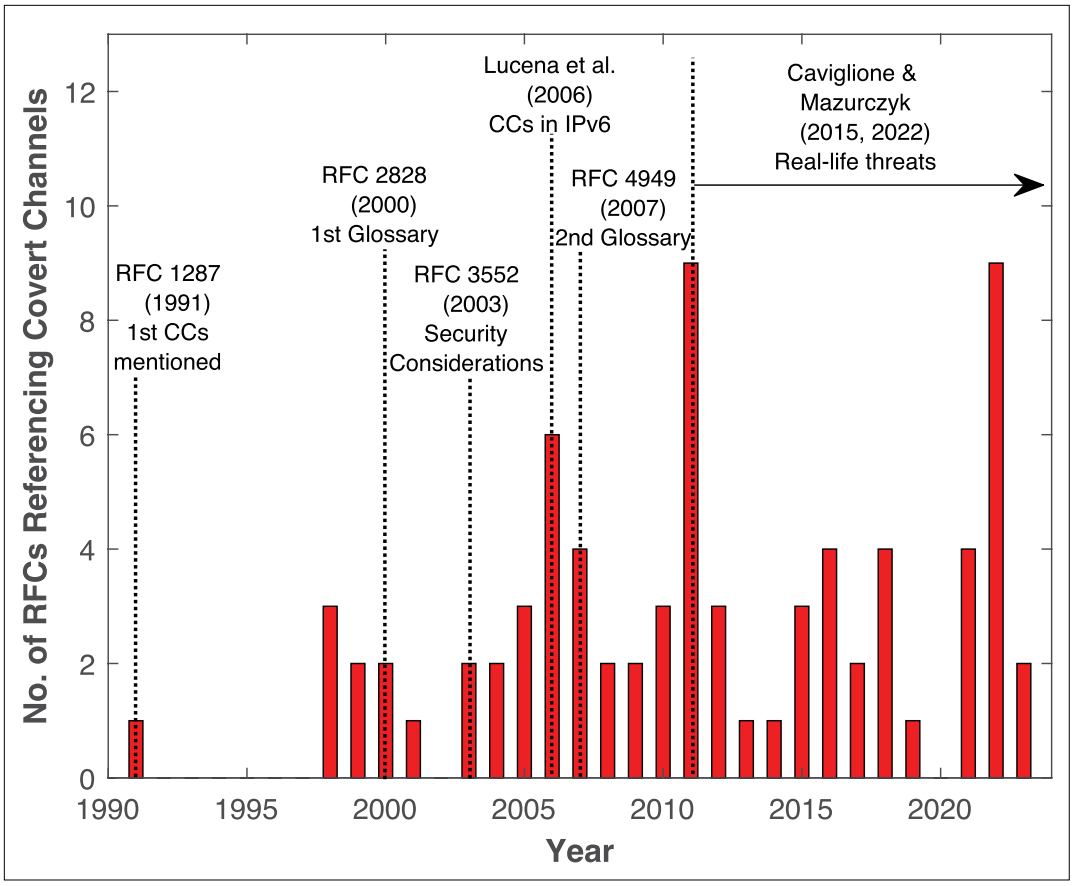

FIGURE 3. Number of RFCs considering covert channels. Years before 1990 have been omitted since no documents mentioned the issue.

future standards will contain a "Security Considerations" section. It also requires that each potential vulnerability is endowed with a countermeasure or a detailed explanation on why it could be neglected. Even if after 2003 documents addressing covert communications increased, they still only represent a fraction.

A major input to the IETF standardization is probably due to the theoretical work by Lucena et al. [13] published in 2006. Specifically, the authors defined 22 covert channels targeting the IPv6 protocol and also proposed simple mitigation strategies. Despite no RFCs explicitly indicated the paper as a reference, it quickly became popular in the information hiding community and could have played a role in some WGs as well. For instance, the `Flow Label` field of IPv6 has been defined in 2004 in RFC 3697 and did not contain any reference to risks related to covert communications. Instead, its updated specification contained in RFC 6437 and published in 2011 explicitly mentioned such a threat. In parallel, the research community intensified efforts to identify mechanisms for implementing covert channels [3]. This, jointly with the work done to improve the overall IPv6 security, probably contributed to update the IETF glossary through RFC 4949 published in 2007. Although network covert channels are not clearly considered, the new definitions are more general and not only limited to processes.

Lastly, from 2010 malware and advanced persistent threats leveraging covert channels have emerged. To the best of our knowledge, their primary sources of documentation are [4], [5]. On this trail, the most recent RFCs finally view protocol fields and data structures also from the perspective of creating covert communications. As an example, the concise binary object representation for IPv4/IPv6 addresses/prefixes published in 2021 in RFC 9164 warns on zeroed bits that can host parasitic information.

Major Use Cases

As discussed, some WGs explicitly addressed issues caused by covert channels. Table 2 collects the main hiding methods observed in various RFCs grouped according to the protocol or functionality exploited. As shown, about the totality of analyzed mechanisms considers the creation of storage covert channels. The only exception is for standards dealing with TLS/DTLS providing some hints on how to encode information in counters, e.g., to adjust the frequency of discarded packets or the error rate.

In general, the prime technique to set a covert communication relies on (partially) overwriting some fields without disrupting the expected behavior of the protocol. To this aim, both IPv4 and IPv6 offer many features, especially due to the presence of optional fields or extensions for traffic engineering and performance measurements, e.g., the `AltMark` option to identify and track flows under test. Another major source of ambiguity for creating a covert communication is due to encodings. Specifically, least significant

Type/Protocol	Features Used to Create the Channel	RFC(s)
IPv6	The following optional fields can be used to store secret data and create a covert channel: `PadN, Pad1, RPL Target, Destination Option for Performance and Diagnostic Metrics` and `Destination Option for Congestion Exposure`, and the `Nonce Destination`. The additional `optional bits` (e.g., Reserved or made available for private experimentation) can be used to convey secrets. More standard fields like the `Flow Label, Extension Headers`, and `Options/Sensitivity Labels` can serve as the carrier for a covert communication	9343, 9288, 4942, 9010, 8250, 7837, 6744, 9049, 6437, 5570, 5175, 5075
ICMPv6	The payload of `ICMPv6 Error Messages` can be misused to contain secret data	4890
IPv4/IPv6	The `IOAM-Data-Fields, AltMark,` and `OWAMP-Test Session` used for traffic engineering or testing can be modified for covert communications, e.g., a burst of marking packets can implement an encoding. A similar case applies to `metadata-only packets` with a Network Service Header. In the case of NSIS-based flows, when the Router Alert option is used, `additional traffic` can be generated and abused to contain secrets	9263, 9197, 9145, 8393, 8321, 5979, 4656
IPv4	If the `ID` field is ignored, its content can be used for covert communication. Additional bytes in many `Options/Extension Headers` can be appended to contain secret data	9341, 6864, 4302
Tunneled IPv4/IPv6	The `DF` Flag can leak data from the encapsulated packet to the outer traffic flow. Not properly protecting `ECN/DSCP` values in the inner/outer header of encapsulated packets can be used to create a covert channel	3884, 2481, 3168, 4301, 6040, 6627, 9347
DNS	The `Encryption Padding - Type Length Value` and the `EDNS (0)` padding option can be used to store secrets. If an unknown `dnsqueryelement` in URI is accepted, it can serve as the carrier for creating a covert channel. An encoding scheme based on `capital/non-capital letters` can be used to exchange secret data. If random values can be manipulated, the `Digital Signature Algorithm` will lead to a high-bandwidth covert channel	8490, 7830, 4501, 2536
SIP	The `UUI Data` within INVITE messages sent between two User Agents may contain arbitrary information allowing to create a covert communication	7433
Multicast Ping Protocol	Undefined `Echo Request Option` messages can be sent to set a covert channel among a multicast group	6450
Cryptography	Signature schemes based on randomization can be exploited via the `salt` of the EMSA-PSS encoding operation to create a covert channel within RSA-cyphered communications	3447, 8017
TLS/DTLS	The `Padding` of ClientHello messages can contain arbitrary information leading to a storage covert channel within TLS conversations. For both TLS and Datagram TLS traffic, the `Sequence Number, Record Counter, Padding,` and malformed packets (i.e., with a bad `MAC`) can be manipulated to alter the volume/frequency of erratic or discarded datagrams for creating a timing covert channel	7685, 9147, 8446, 7366, 5246
SSH	In general, SSH has not been designed to eliminate covert channels. Examples are the `Padding` and `SSH_MSG_IGNORE` messages that can be used to convey secret data	4251
Encodings	Base16, Base32, Base 45, and Base 64 can be violated by adding `non-alphabet characters` to create a covert channel. In Base16, Base32, and Base64, the padded `non-significant bits` can carry arbitrary data. Values not considered in the standard or "overflowing" a specific threshold can be used for covert communication, especially in `UTF-9` and `UTF-18` encodings. In CBOR Tags, the least significant bits of the `prefix` of v4/v6 addresses can host secrets. In the XMPP protocol, during the `authentication negotiation` phase, Base64 encoded data can convey arbitrary information	9285, 9164, 4648, 4042
Codecs	The `Padding` of Opus codec may convey secret data. The mechanism of AAC for transmitting `Extra Data` can convey secret information	5691, 6716
Presence Information	URIs can point to additional resources that can be used to move secret data. Icons can also be used as carriers. In sieve email filtering, presence data can be modulated to encode bits and implement covert channels	4482, 6133
Pulse-per-Second	APIs for high-precision time-keeping for retrieving `Timestamps` can be used to implement covert channels	2783

TABLE 2. RFCs addressing covert channels in specific fields and features of protocols, codecs, or software objects.

bits, non-alphabet characters, or minimal violations of the encoding scheme, can be used to append or hide arbitrary bits of data, which can then be remotely transmitted. As an example, v4/v6 `prefixes` in the JSON-like data structure introduced in RFC 9164 can host secrets within the least significant bits. As shown in Table 2, several protocols can embed a storage channel, e.g., by abusing the `UUI Data` of the INVITE method of the SIP protocol.

Despite the multitude of opportunities, the most recurrent use cases observed within IETF standards target *three* main protocol features. The first is the `Flow Label` of IPv6, which can be exploited to conduct a wide-range of attacks. For instance, its ability to uniquely identify an IPv6 flow allows to passively scan endpoints, launch QoS stealing campaigns, and guess the identity of the host (see, e.g., [14] and the references therein). For the case of creating a covert channel, the `Flow Label` offers several opportunities: *i*) it is 20-bit long, thus offering a spacious carrier for embedding secret data; *ii*) it is rarely used in production-quality networks, thus alterations will not impact core functionalities; *iii*) its pseudo-random nature makes it difficult to spot manipulations or outlier values.

DNS is the second protocol that can be exploited to cloak data exchanges, especially by real malware [5]. It provides many additional fields and text-based entries, which can be used to append characters or hide data through linguistic manipulations, for instance, the adoption of synonyms, patterns in punctuation or case-based schemes. Moreover, since some padding data (e.g., the `EDNS (0)`) or optional fields are not protected, DNS traffic can also be altered to contain arbitrary information.[2]

The third feature considered in many IETF standards concerns the "knowledge" that is needed to implement active queue management policies or enforce QoS. In more detail, both the `ECN` bits or the `DSCP` field have been identified as the major vectors to elude tunneling and encryption through an inner/outer copy mechanism between headers. In fact, such fields are mutable, thus difficult to protect without preventing that Internet-wide mechanisms will continue to work properly.

[2] An example observed in the wild is the backdoor implemented by Sunburst discovered in 2020. In essence, it utilizes CNAME records to redirect C&C communications of the infected hosts towards the remote server controlled by the attacker. More details on such threats can be found at: https://github.com/luca-cav/steg-in-the-wild. [Last Accessed: August 2023].

Countermeasures

Only the 59% of analyzed RFCs (i.e., 45 out of 76) proposed some mitigation techniques, which often reduce to naive strategies, especially for rate-limited channels. For the case of tunnels, covert channels are typically not regarded as an issue, unless the traffic performs a transition from protected to unprotected domains.

A prime line of defense against storage channels shared by many WGs is to overwrite susceptible fields with zeros or random bits. For instance, this has been suggested to disrupt channels living within the `Flow Label` of IPv6 or in the `ClientHello Padding Extension` of the TLS, as discussed in RFCs 9049 and 7685. However, to prevent a loss of functionality, zeroing should be done only when a feature is not critical. More strict solutions propose to disable by default unneeded functionalities (e.g., ECN bits to support traffic management operations) or enforce that protocol fields will only accept a limited set of values.

Indeed, neutralizing a covert channel requires recognizing whether a field has been manipulated to carry the secret data. To this aim, several standards (e.g., RFCs 9164 and 4890) suggest to perform sanity checks to report unknown values/configurations, especially for optional fields. Protocol data units can be then dropped or filtered. A similar scheme can be adopted to spot data hidden in software objects. For instance, RFC 4648 proposes to avoid covert channels in Base64-encoded representations by enforcing implementations to reject non-alphabet characters. Other standards (e.g., RFCs 9341, 7837, and 6744) suggest to neutralize covert channels by increasing the overall security level. For instance, evaluating the performance of traffic via markers, such as the `AltMark` option, should be only done in trusted places. Besides, the limited-functionality mode should be preferred when operating in an uncontrolled area to prevent leaking data outside an IPSec tunnel by abusing congestion information. Encryption is another choice to hinder secret endpoints from reading/writing a field to implement a storage channel. As an example, encrypting traffic via IPsec or TLS is effective to avoid that a malicious entity can cloak data within the extensions of the AAC audio codec.

In some cases, the WGs elaborated detailed workarounds. Specifically, to prevent updates used for location-based services from leaking data, RFC 5808 advises periodically sending additional notifications to subscribers, even when no movements occurred. As a result, the covert channel is impaired with a sort of "noisy" signal. The chance of creating a covert communication can be further reduced by acting over the network configuration. If covert channels are a real concern, contact or presence information should not use resources or icons retrieved "outside" a well-defined domain, e.g., to limit risks due to steganography. Another idea is to always check the data returned to the sender(s) to avoid covert communications, e.g., via presence information for sieve email filtering.

As regards timing channels, countermeasures have been proposed only in standards dealing with the TLS, see, e.g., RFCs 9147 and 5246.

Only the 59% of analyzed RFCs (i.e., 45 out of 76) proposed some mitigation techniques, which often reduce to naive strategies, especially for rate-limited channels.

Specifically, to impair encoding mechanisms using malformed records to alter the number of discarded packets, related counters should be reset (e.g., the `Record Counter`) at suitable epochs and erratic packets should be promptly dropped to prevent later processing. To mitigate channels exploiting the time needed to perform specific computations (e.g., by adjusting the `Padding`), there are no known universal approaches besides deploying a constant-time processing scheme. Instead, schemes based on the alteration of the time needed to compute the message authentication codes (e.g., for the `Cipher Block Chaining`) should be considered unfeasible due to the extremely tight temporal behaviors.

Summing up, countermeasures mainly suggest sanitizing specific fields/values or adopting additional precautions, e.g., encryption. Yet, some RFCs provide two very general guidelines against storage and timing channels. The first is flow analysis, which can help to find possible carriers. The second is flow control, which can be used to confine channels.

Gaps and Recommendations

This section summarizes the major gaps of IETF standards when dealing with covert channels and then presents some possible enhancements.

Analysis of Current Gaps

Although many documents are not relevant for our investigation, RFCs considering covert channels are still a minimal fraction and overemphasize the storage variant. The choice of focusing on realistic threats is sound, even if IETF partially ignores several advanced hiding mechanisms, e.g., cloaking data through packet reordering as well as real malware hiding data in ICMP and MQTT traffic [3]. Unfortunately, the pace at which malicious actors progress is faster than the one of the standardization pipeline. For instance, Linux.Fokirtor demonstrated how to hide arbitrary data in SSH traffic [4], [5]. The IETF should reduce this gap by reviewing its most popular protocols, encodings, and data structures also in terms of recent attacks leveraging information hiding. This can help to design more organic countermeasures, which are currently provided as simple "add-ons" often unable to eliminate the issue. As an example, enforcing that a feature is only adopted in trusted deployments shifts the problem rather than fixing it. Improving awareness on how real offensive campaigns exploit covert communications may steer a WG towards drastic but more secure design choices (e.g., the elimination of a feature).

The lack of an abstract vision of the attack surface characterizing covert communications is another gap, which is confirmed by the absence of protocol-agnostic countermeasures. This choice guarantees that RFCs are always coherent and prevents overlaps or out-of-scope workarounds. Yet, recurring cloaking techniques, such as those based on timing, manipulation of unused bits, and alteration of a sequence of options, are never

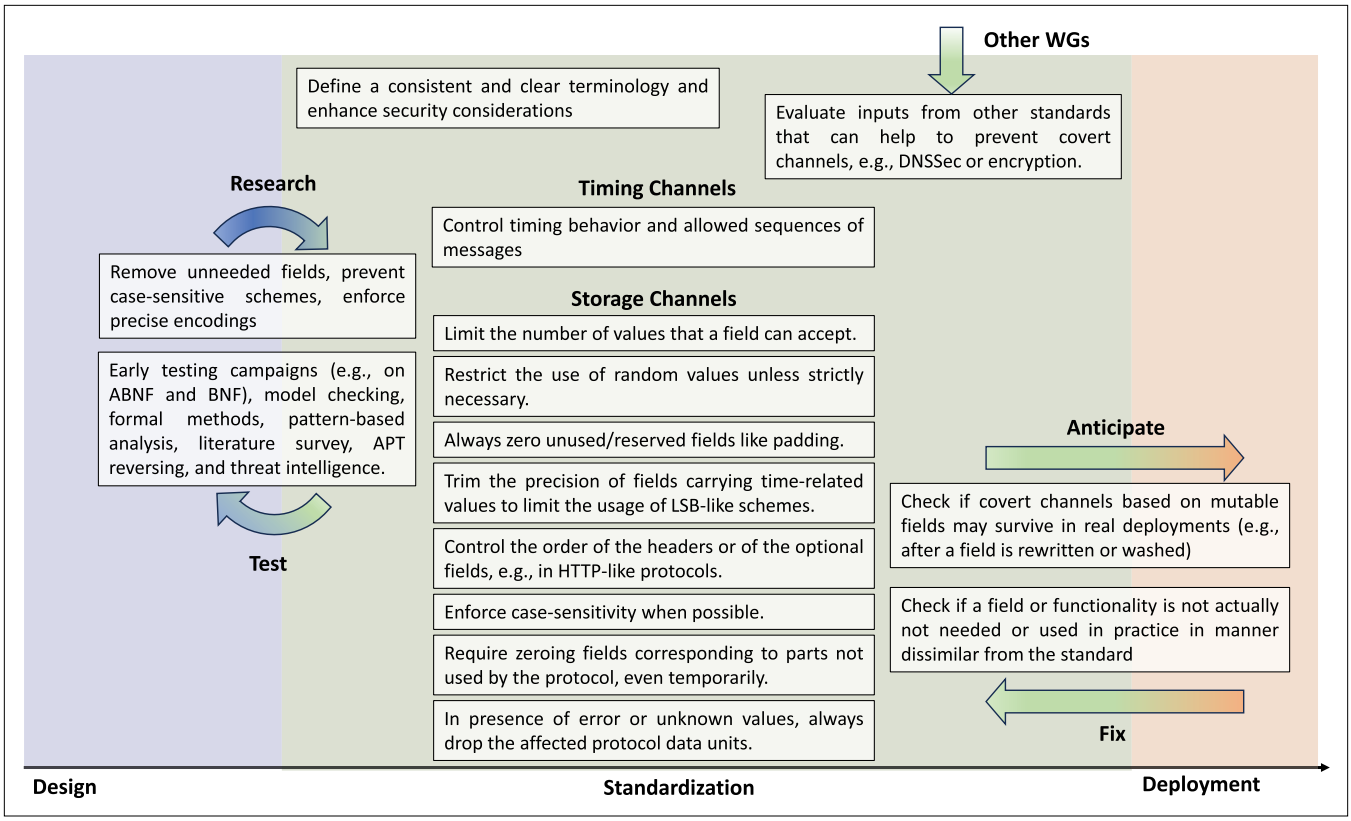

FIGURE 4. Guidelines to improve the overall IETF standardization process when addressing covert channels.

Improving awareness on how real offensive campaigns exploit covert communications may steer a WG towards drastic but more secure design choices (e.g., the elimination of a feature).

addressed, and the discussion is unconsciously delegated to another venue. The RFCs impacting privacy, freedom, and sovereignty should face covert channels in a more holistic manner, possibly orchestrated by the Security Area Directorate.

Lastly, our analysis outlined that definitions have some glitches. Enhancing RFC 4949 is then desirable, especially to prevent that concepts unique to side channels, user profiling, and covert channels will collapse reducing the expressiveness and clarity of many standards.

Possible Improvements

Undoubtedly, IETF standards demonstrated to have "antibodies" to face many emerging or unexpected security challenges. Figure 4 shows guidelines to improve how to handle threat models leveraging hidden communications.

In general, preventing that data can be cloaked requires to search for suitable trade-offs, primarily among technological constraints, future-proof functionalities, and implementation complexity. A possible general abstraction can condense them in the *flexibility-functionalities* umbrella trade-off. As an example, avoiding to encode data via positional shifts of optional headers requires limitations in the protocol state machine. Yet, this may burden the implementation or render the protocol too rigid and complex.

The stage at which covert channels should be addressed is a very important aspect. Specifically, the literature suggests that detecting and limiting covert channels is difficult to do *a-posteriori* [3], [5], [13]. Hence, their elimination should start from the very early design steps, which needs inputs from the research community to find potential carriers via *tests*. Experts able to dissect new attack campaigns are important as well, since realistic scenarios can help to understand whether protocols/software are experiencing conceptual drifts, i.e., functionalities foreseen in RFCs are employed in a different manner. As a result of this tight interaction, a WG can amend, obsolete, or *fix* a technology. The symmetrical approach is to pursue *anticipation* but understanding if a specific covert channel can survive in the Internet leads to a vast and heterogeneous problem space. To this aim, *liaisons* should be established, especially to borrow effective fixes, e.g., secure versions of complementary technologies. At the same time, searching for pathological behaviors in the Internet, e.g., bleaching or remarking `DSCP` and `ECN` bits [15], could already provide a built-in mitigation without bloating the idea under development.

Figure 4 also contains some technical recommendations for designing and standardizing protocols or software objects more resistant to information hiding attempts. As an example, the availability of resources jointly with the fear that a field could not be large enough to support future scenarios, often triggered a throwing-bits-to-the-problem approach. This led to the creation of timestamps (and related fields) with exceptional precision, which are overkill for many applications. The least significant bit(s) could be then used to host covert channels without causing any noticeable alteration. Similarly, pseudo-random or padded data must be strictly ruled to reduce ambiguities in

the implementation phase. Concerning standardization, having a common set of definitions and threat models is highly desirable, especially since RFCs are intended for a wide audience with different backgrounds. The obsolescence of an RFC should be also driven by mutations, common practices, and "lean" protocol implementations observed in many resource-constrained devices. A covert channel could take advantage of an IoT node unable to enforce additional checks or early discard malformed packets due to computational constraints.

Lastly, along the lines of RFC 3552 and 9416, a major avenue for improvement could be the creation of a dedicated standard document explicitly addressing the security implications of covert channels and data hiding in general.

Conclusion

In this paper, we investigated how covert channels have been addressed in IETF standards through the years. The results showcased that only some RFCs explicitly considered them and provided general countermeasures. Unfortunately, fixing *a-posteriori* functional ambiguities or imperfect implementations is difficult. Targeting early stages of standardization is thus crucial to make the Internet a more secure, robust, and privacy-preserving place.

Acknowledgment

The work of Luca Caviglione was supported by the Project SERICS and Project RAISE through the European Union—NextGenerationEU. The work of Wojciech Mazurczyk was supported by the Excellence Initiative Program-Research University of the Warsaw University of Technology within the Mobility PW Programme.

References

[1] A. Sill, "The design and architecture of microservices," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 76–80, Sep. 2016.
[2] I. Tomic and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1910–1923, Dec. 2017.
[3] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 44–57, Sep. 2007.
[4] W. Mazurczyk and L. Caviglione, "Information hiding as a challenge for malware detection," *IEEE Secur. Privacy*, vol. 13, no. 2, pp. 89–93, Mar. 2015.
[5] L. Caviglione and W. Mazurczyk, "Never mind the malware, here's the stegomalware," *IEEE Secur. Privacy*, vol. 20, no. 5, pp. 101–106, Sep. 2022.
[6] M. Musuvathi and D. R. Engler, "Model checking large network protocol implementations," in *Proc. 1st Conf. Symp. Networked Syst. Design Implement.*, vol. 4, Mar. 2004, p. 12.
[7] W. Wen, C. Forman, and S. L. Jarvenpaa, "The effects of technology standards on complementor innovations: Evidence from the IETF," *Res. Policy*, vol. 51, no. 6, Jul. 2022, Art. no. 104518.
[8] S. McQuistin et al., "Characterising the IETF through the lens of RFC deployment," in *Proc. 21st ACM Internet Meas. Conf.*, Nov. 2021, pp. 137–149.
[9] C. A. Wood, "Not-so-low hanging fruit: Security and privacy research opportunities for IETF protocols," in *Proc. Appl. Netw. Res. Workshop*, Jul. 2023, pp. 41–43.
[10] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 265–275, Jun. 2014.
[11] P. Zórawski, L. Caviglione, and W. Mazurczyk, "A long-term perspective of the Internet susceptibility to covert channels," *IEEE Commun. Mag.*, vol. 61, no. 10, pp. 171–177, Oct. 2023, doi: 10.1109/MCOM.011.2200744.
[12] L. C. Paulson, "Inductive analysis of the Internet protocol TLS," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 332–351, Aug. 1999.
[13] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert Channels in IPv6," in *Proc. Int. Workshop Privacy Enhancing Technol.* Berlin, Germany: Springer, 2005, pp. 147–166.
[14] J. Berger, A. Klein, and B. Pinkas, "Flaw label: Exploiting IPv6 flow label," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1259–1276.
[15] A. Custura, R. Secchi, and G. Fairhurst, "Exploring DSCP modification pathologies in the Internet," *Comput. Commun.*, vol. 127, pp. 86–94, Sep. 2018.

Biographies

Luca Caviglione is a Senior Research Scientist with the National Research Council of Italy, Institute of Applied Mathematics and Information Technologies, Genova, Italy. His research interests include security and networking.

Wojciech Mazurczyk (wojciech.mazurczyk@pw.edu.pl) is a University Professor with the Institute of Computer Science, Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland. His research interests include cybersecurity and communication networks.