


Article

Trends and Challenges in Network Covert Channels Countermeasures

Luca Caviglione 

Institute for Applied Mathematics and Information Technologies, 16149 Genova, Italy;
luca.caviglione@ge.imati.cnr.it

Abstract: Network covert channels are increasingly used to endow malware with stealthy behaviors, for instance to exfiltrate data or to orchestrate nodes of a botnet in a cloaked manner. Unfortunately, the detection of such attacks is difficult as network covert channels are often characterized by low data rates and defenders do not know in advance where the secret information has been hidden. Moreover, neutralization or mitigation are hard tasks, as they require to not disrupt legitimate flows or degrade the quality perceived by users. As a consequence, countermeasures are tightly coupled to specific channel architectures, leading to poorly generalizable and often scarcely scalable approaches. In this perspective, this paper investigates trends and challenges in the development of countermeasures against the most popular network covert channels. To this aim, we reviewed the relevant literature by considering approaches that can be effectively deployed to detect general injection mechanisms or threats observed in the wild. Emphasis has been put on enlightening trajectories that should be considered when engineering mitigation techniques or planning the research to face the increasing wave of information-hiding-capable malware. Results indicate that many works are extremely specialized and an effective strategy for taming security risks caused by network covert channels may benefit from high-level and general approaches. Moreover, mechanisms to prevent the exploitation of ambiguities should be already considered in early design phases of both protocols and services.



Citation: Caviglione, L. Trends and Challenges in Network Covert Channels Countermeasures. *Appl. Sci.* **2021**, *11*, 1641. <https://doi.org/10.3390/app11041641>

Academic Editor: David Megías

Received: 12 January 2021

Accepted: 9 February 2021

Published: 11 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: network covert channels; detection; stegomalware; traffic sanitization; normalization

1. Introduction

Information hiding and steganographic techniques are becoming widely used by attackers to avoid detection and remain unnoticed for a large amount of time [1]. For instance, they have been used to conceal the presence of a malware within innocent looking pictures, to hide malicious code or additional functionalities with the aim of implementing covert multi-stage loading architectures, as well as to exfiltrate secret information in Advanced Persistent Threats (APTs) [2,3]. As a consequence of the versatility of information hiding mechanisms, a new-wave of malware endowed with steganographic capabilities (named *stegomalware*) has been observed in the wild and the trend is expected to grow in the near future [1,4]. One of the most popular and effective uses of information hiding to support insecurity concerns the creation of *covert channels*, i.e., hidden communication paths allowing two peers to exchange data. According to Lampson, covert channels are “not intended for information transfer at all, such as the service program’s effect on the system load” [5]. Literature proposes a wide array of scenarios and mechanisms for creating hidden communication paths between a variety of software and hardware entities. As possible examples of endpoints wanting to secretly exchange data, we mention: cloud-based and softwarized services, containers and virtual machines and multi-core or multi-CPU frameworks. Despite the nature of the specific target, a covert channel can be generally linked to an imperfect isolation, i.e., state information or a specific behavior of a software or hardware element is leaked outside the intended functional perimeter [6].

To create a covert channel, the two secret endpoints must agree on a steganographic/hiding scheme, which acts as the pre-shared secret. The secret information is then embedded into a

carrier. To this aim, different methodologies can be used and various hardware or software artifacts can be exploited for storing arbitrary data. For instance, the secret can be directly embedded in the metadata of a file or properly encoded by manipulating some temporal evolution such as the load offered to the CPU. Even if complex encoding and embedding strategies exist, the totality of covert channels can be reduced to a set of well-defined patterns, such as the modulation of some structural properties (e.g., the size of a file) or the introduction of artificial transmission errors [7]. Concerning carriers that can contain the secret information, network flows are becoming quickly the most preferred one used by attackers. Accordingly, the produced hidden communication paths are named *network covert channels*. As possible examples, channels can be built by injecting data in unused protocol fields as well as by modulating some behaviors of the flow like the throughput or the inter packet time (see, e.g., [8] for a survey on the topic).

As a consequence, the investigation of security risks caused by network covert channels has quickly become of paramount importance. In fact, network covert channels have proven to be an effective tool to support several malicious activities. Specifically, they have been used for exfiltrating stolen confidential data (e.g., login credentials), to orchestrate nodes of a botnet, to implement multi-stage loading architectures to retrieve attack routines (e.g., by embedding malicious code in innocent-looking images), to launch commands for configuring backdoors and to support industrial espionage campaigns [1,2,9]. Unfortunately, modern scenarios offer a wide range of protocols, software entities and hardware devices that can be used to contain secret information [7,8]. This grants the attacker huge advantages over the defender and the deployment of countermeasures against a specific steganographic threat seldom happens a priori. Therefore, covert channels are often addressed as a by-product outcome of the mitigation of apparently unrelated anomalous behaviors, the analysis and reverse engineering of a malware sample or the direct disclosure from the attacker [1–4]. Thus, recognizing that a given deployment is exposed to steganographic communication attempts may be a challenging and poorly generalizable process. Nevertheless, development of countermeasures may lead to inefficient solutions based on the “needle in the haystack” principle possibly accounting for the indiscriminate penalization of traffic flows, the disruption of some protocol functionalities or the introduction of performance bottlenecks in the network.

Despite the number of threats exploiting information hiding principles and covert channels to implement the various phases of an attack, the majority of research efforts focused on the investigation of how protocol ambiguities, poor design choices and flawed implementations of computing and communication frameworks can be used to covertly move information through the Internet. As a consequence, several reviews have been prepared (see, e.g., [1,2,8]) also by considering attacks targeting well-defined technological segments such as modern smartphones [9]. Alas, the specificities of the various embedding methods hindered the development of a coherent literature on countermeasures against network covert channels. In this perspective, the detection and neutralization of this class of threats have been never systematically organized or reviewed, with the notable exception of a partial discussion in [8], which dates back to 2007.

Therefore, this paper aims at investigating trends and challenges characterizing the development of countermeasures against network covert channels, with the wide acceptance of tools, methodologies and algorithms aiming at detecting hidden communication attempts. We also include approaches intended for limiting or completely neutralizing the transmission capability of hidden communication paths laying within a network flow. The goals of this investigation are multiple. First, we want to understand if it is possible to delineate a clear trajectory in the literature aiming at counteracting network covert channels, or if the topic is addressed in a split and heterogeneous manner. Second, we aim at deriving some general guidelines or gaps to be filled to face challenges arising from the diffusion of stegomalware. Third, we intend to define some tradeoffs to be considered when designing countermeasures. Last, we want to show that, even if a relevant corpus of works on the topic exists, the attacker can easily evade preexistent mitigation approaches

by moving to another protocol or slightly changing the injection method. To answer such research questions, we carried out a systematic review of the literature also by considering attacks and threats observed in the wild. The limit of this investigation is rooted within the very carrier-centric nature of hiding methods and countermeasures. In fact, the majority of works only proposes channel-specific mitigation mechanisms without further deepening generalizability aspects. Another consequence is that the research is balkanized over different areas (e.g., information security, engineering and networking) often having incongruent definitions and thus accounting for overlaps or difficulties in retrieving ideas.

Summing up, the main contributions of the paper are: (i) a review of the most recent approaches for counteracting network covert channels; (ii) the identification of the major research challenges and the most promising development trends; (iii) the discussion of advancements needed for elaborating a more coherent and unified framework for taming network covert channels and stegomalware.

The remainder of the paper is structured as follows. Section 2 briefly introduces the background on network covert channels, while Section 3 discusses aspects to be considered when designing a mitigation approach. Section 4 showcases detection techniques and methodologies, while Section 5 deals with approaches devoted to limiting and eliminating network covert channels. Section 6 discusses trends and challenges and Section 7 concludes the paper and portrays possible future research directions.

2. Background

As hinted, the adoption of some form of steganography to endow malware with stealthy capabilities has started more than a decade ago, mainly with threats using digital media as a diffusion vector or to exchange data containing configuration details (e.g., a list of IP addresses to be inspected or attacked) [1]. Instead, network covert channels have become popular more recently, as they offer to the attacker a wider palette of opportunities. First, a traffic flow is somewhat boundless, thus providing a carrier with few constraints compared to other software artifacts or digital media. Second, it is not uncommon to have network infrastructures with flows that last hours or days, thus allowing the malware to implement permanent hidden communication paths, for instance to support APTs or to orchestrate botnets [10]. In this perspective, Figure 1 depicts the reference scenario when addressing network covert channels and their detection. In more detail, we consider two covert endpoints (named as covert sender and covert receiver, respectively) wanting to remotely communicate in a hidden manner. To this aim, the legitimate, overt network flow is used as the carrier for containing the secret information. We point out that the two covert endpoints can be located in different portions of the network, thus implementing channels with different scopes. They can act in a Man-in-the-Middle manner as depicted in Figure 1 by eavesdropping a preexistent flow, or they can be colocated within the end nodes generating the overt traffic flow. Despite the placement of the communicating endpoints, two core classes of covert channels can be created. They are characterized according to the steganographic approach used to embed the secret information. Specifically:

- if the injection mechanism *directly embeds* data within the carrier, a *storage network covert channel* is created;
- if the injection mechanism manipulates the *timing* of events or the *temporal evolution* of the carrier, a *timing network covert channel* is created.

A general representation of the aforementioned approaches is depicted in Figure 1. A possible example of a storage channel is the one where the secret sender directly writes data in the unused bits available in the TCP header. For the case of a timing channel, the sender could encode information by modulating the delay experienced by two consecutive packets composing the flow, e.g., a delay greater than a given threshold value denotes a 1, otherwise a 0. As discussed in [11], the TCP/IP protocol architecture can be targeted with a variety of methods also by using many techniques or protocols simultaneously.

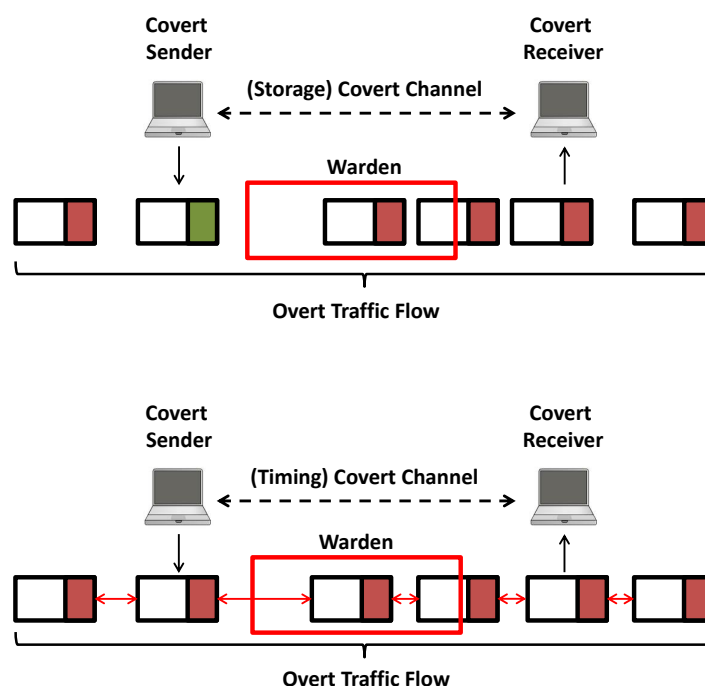


Figure 1. Reference scenario for the usage of network covert channels and their detection.

Similar to other applications of steganography and information hiding, network covert channels are also characterized by the following three core metrics:

- *steganographic bandwidth or capacity*: the volume of secret data that can be sent per time unit by using a given embedding method;
- *undetectability*: the inability to detect the secret data within a certain carrier;
- *robustness*: the number of alterations that the hidden message can withstand before being destroyed.

Obviously, the “best” covert channel should be robust and hard to detect, as well as able to provide the highest bandwidth. Unfortunately, it is not possible to maximize such properties at once, as they are tightly coupled via a *magic triangle* rule [9,12], i.e., it is not feasible to increase a performance index without lowering the remaining two. As an intuitive example, let us consider using some bits of an image as a method to embed secrets. Each alteration will lead to some noise, which must be kept under a suitable threshold to avoid detection or that the visual system can notice that some manipulations happened. In fact, the more bits that are used for the secret (the bandwidth increases), the more would be the noise or visual alterations, thus making the method more evident (detectable) [9]. Concerning network covert channels, embedding secrets directly in the unused bits of the TCP could lead to a channel with a capacity of up to 4 bits/segment. Clearly, the more bits are used, the higher will be the impact on the statistical distribution of bits composing the header, hence making the channel more detectable. Yet, the channel can be easily disrupted as it is sufficient to overwrite unused fields with random values or revert them to a standard configuration. This can be mitigated by using a more sophisticated embedding method, but paying something in terms of complexity and channel capacity. By contrast, a channel modulating the secret information in the inter packet time exploits a less obvious hiding scheme, but it is also slower and weaker since natural delays of the networks (e.g., due to queuing and competing with other flows) can completely destroy the information.

An additional metric, is the *steganographic cost*, which evaluates the degradation experienced by the carrier due to the application of a steganographic method. For instance, for a network covert channel living within a VoIP flow, the steganographic cost can represent the impact on the quality of the conversation experienced by users (e.g., additional delays, reduced SNR performances of the codec or audible artifacts) [13].

3. Development of Countermeasures

The magic triangle relation can be considered as a prime conceptual device for driving the development of countermeasures against the various flavors of covert channels. In this vein, the seminal work of G. J. Simmons introduced the notion of *warden*, i.e., an entity responsible of spotting hidden communication attempts [14]. Specifically, Simmons modeled the threat by considering two prisoners wanting to orchestrate an escape plan by covertly exchanging details. The jail warden monitors such an exchange: if the attempt is spotted, prisoners will be confined. For the case of network covert channels, a warden can be a software or hardware entity able to collect and inspect network packets. Figure 1 depicts an intermediate node (denoted with a red box) acting as a warden.

According to [15], different wardens exist and they can be classified by considering the type of information they use to detect the covert channel as well as their structure. Literature still not agree on a unique taxonomy, but three main traits can be used to describe middleboxes and techniques for detecting hidden communication attempts exploiting network artifacts:

- *functionality*: it denotes how the warden counteracts the covert channel. As it happens for other tools enforcing network security (e.g., firewalls or intrusion detection systems), a warden can be *stateless* or *stateful* depending on its ability of considering a packet at time or a burst of traffic, respectively. While the longer correlation of information can lead to improved performances, storing and processing huge traffic footprints may lead to unfeasible storage or computing requirements. If a warden only captures the traffic and acts on a private copy, it is considered *passive*. Otherwise, if it can manipulate, alter or drop packets, it is considered *active*. Lastly, a warden can be *static* or *dynamic*, according to the ability of changing its policies during time or adapting against specific threats or network conditions. We point out that a warden can simultaneously have multiple functionalities, e.g., it can be static and stateless.
- *knowledge*: it defines the type of information used by the warden to spot the presence of the channel. For instance, the warden can be aware of the exploited traffic or service (in this case, it is defined as *network-aware*) or totally *agnostic*. Moreover, it can know a priori the type of used embedding process or steganographic technique, or be engineered to only react against some specific patterns or threats.
- *localization*: it defines where the warden has been placed to intercept the traffic and to monitor the network usage. Due to the evolution of devices and deployments, modern wardens should be capable to protect large network trunks and interact with multiple replicas as to provide scalability and detect the most sophisticated threats. Thus, a warden can act on a *local* basis (e.g., on the device of the end user or his/her local network perimeter) or cover a vast geographical area. An emerging trend concerns the adoption of *distributed* wardens, i.e., nodes able to gather the traffic in different portions of the network and cooperate to spot the covert channel.

In the perspective of developing countermeasures against network cover channels, the ultimate goal of a warden is to neutralize steganographic threats laying within the legitimate traffic. To this aim, the first step is to perform the *detection*, i.e., become aware and identify the hidden communication path. This is a prerequisite for performing some further actions aimed at guaranteeing the security of network and computing infrastructures. Ideally, a warden should completely *eliminate* the channel, but this is not always possible. For instance, this happens when impeding a specific channel will also disrupt some legitimate network services or penalize too much the quality perceived by users. Therefore, it is not uncommon that the covert communication can be only partially impaired or *limited*, e.g., the steganographic bandwidth is reduced leading to a channel not usable or appealing for a specific attack.

A warden could be also considered as a tool for implementing some form of threat intelligence. Specifically, it can not be able to counteract an attack, but can provide suitable knowledge to prevent the creation of a network covert channel. This can be done by

teaming with other tools (e.g., by driving the definition of packet filtering rules to be deployed in firewalls) or to support security-by-design approaches.

4. Covert Channel Detection

In this section, we present the main detection techniques with emphasis on the limits and the challenges to be faced in order to be deployed in real-world scenarios. First we will present mechanisms targeting the two fundamental injection families at the basis of a wide range of hidden communication paths (i.e., those allowing the creation of timing and storage channels). Then, we will address techniques intended for specific protocols. In more detail, we will investigate the detection of covert channels targeting IPv6, which will be expected to become widespread in the near future and thus appealing from the viewpoint of an attacker. In addition, we will also investigate the various methodologies used to reveal hidden communication paths in VoIP conversations as well as DNS and HTTP tunnels. Such protocols have in common a great popularity and the ubiquitous availability, hence their presence in a network is seldom perceived as an anomaly and defenders typically need to deal with non-negligible traffic volumes.

4.1. Timing Channels

The majority of timing channels injects secret information by modulating the inter-packet time of IP datagrams, thus leading to protocol-agnostic methods, i.e., the encoding happens at the network level, despite the used higher protocols and applications responsible of generating and manipulating the stream. Popular transmission schemes are: *on-off switching*, i.e., if the sender transmits a packet within an agreed time frame it signals a 1, otherwise a lack of transmissions represents a 0; *jitterbug-like*, i.e., the information is encoded by changing the inter-packet arrival time of keystrokes within normal keystroke traffic (this algorithm can be generalized to be applied to other traffic sources).

The detection of network timing channels is primarily based on the computation of a statistical indicator or a general performance metric to assess the regularity of the temporal evolution of the traffic flow. Thus, the decision is performed when the timing statistics of the flow deviate too much, e.g., from a threshold [16]. Unfortunately, as discussed in [17], such a class of covert channels can not be easily spotted if the attacker frequently changes the encoding scheme and the protocol, or injects a suitable amount of noise (e.g., via additional delays or by randomly starting and stopping the injection of data within the transmission window). This trait can be mitigated by using more sophisticated statistical tests, but at the price of more complex and memory consuming computations (e.g., over 2000 timing values needed to be processed for up to 400 iterations of an algorithm [18]). As a consequence, when in the presence of large-scale networks or multiple flows characterized by a high throughput the implementation of an effective warden could be not a trivial task. Moreover, traffic composed of small protocol data units can exacerbate performance bottlenecks or the presence of non-optimized implementations.

A key challenge to be faced concerns the ability of the attacker of applying obfuscation or changing the encoding scheme used to map the information on the timing of packets. In this vein, literature showcases several techniques that can be used to avoid the detection from a warden or a firewall (see, e.g., [8,16] and references therein for a detailed discussion). As a workaround, machine learning can be considered for developing more general and efficient classifiers able to reveal relationships difficult to capture via network analysis approaches based on the common sense. For instance, in [19] support vector machines are used to spot different covert communications exploiting a timing scheme. Unfortunately, as it happens for other traffic classification problems, this requires to have enough data for training the decision maker. In addition, the attacker can also adopt more advanced techniques like encrypting the hidden message or apply an obfuscation algorithm to the overt traffic. A promising approach is presented in [17], where authors propose to exploit possible correlation between the content of the memory and the information injected in the traffic flow. Alas, the limit of the idea is that the warden should reside on the node hosting

the covert sender or the covert receiver as well as rethink the general detection principle could be required.

Another class of timing channels targets TCP segments, which offer a more fine array of possibilities for encoding the information. For instance, steganographic schemes may exploit sequences of ACK/SYN, patterns of segments containing information and artificial reordering. Detection can still be done via general network analysis frameworks, but there is also the need of considering specific behaviors of the TCP. For instance, analyzing the size of the various bursts composing a stream can reveal the presence of a secret sender modulating the stream (in general, the size of bursts are solely ruled by the congestion control behavior of the TCP) [20]. A further idea could be developed by considering the timing statistics of segments containing an acknowledgment.

Typically, the elimination of a network timing channel can be considered simple in some network scenarios or under some hypotheses (e.g., the absence of traffic with real-time constraints) and possible optimizations can be borrowed from the literature dealing with timing channels targeting the single node (see [21] for a comprehensive survey on timing channels not limited to the network case). However, an attacker could be able to slow the rate of the covert channel as to make the attack not detectable or the communication impossible to disrupt without impacting legitimate traffic. When designing a warden against timing network covert channels, the detection may be more desirable than limiting or eliminating the threat [16] as identifying compromised hosts or the range of contacted IP addresses could lead to neutralize some “assets” of a cybercriminal, such as a botnet.

Lastly, when in the presence of methods reordering the sequence of packets or messages (e.g., IP datagrams, TCP segments or HTTP requests), a promising trend is to borrow techniques for data mining and apply them to the captured traffic [22]. This can be also done for implementing distributed wardens able to perform detection by considering observations gathered in different portions of the network. Moreover, being able to implement protocol-agnostic schemes is an important requirement, since the attacker can use different layers of the network stack to exfiltrate information. For instance, [23] proposes a protocol-independent approach and demonstrates that the detectability of a channel is proportional to the number of packets altered by the secret sender.

4.2. Storage Channels

As hinted, each network storage covert channel targets a specific part of the protocol data unit [11]. Therefore, the detection is seldom generalizable and requires the development of ad-hoc solutions or methodologies. A complementary approach exploits malware analysis to isolate and reverse engineer the executable to identify the carriers used for establish the channel [24]. To some extent, storage channels represent the perfect synecdoche of the application of steganographic techniques to network traffic, since they highlight that the defender often has to grope in the dark.

In this perspective, development of countermeasures could follow two paths. The first concerns the ability of developing wardens able to inspect in an efficient manner multiple protocols via frameworks that can be easily extended. In this manner, the defender can try to gain ground owing to the inspection of multiple protocol fields without deteriorating performances or having to rewrite relevant portions of the code each time. For instance, [25] proposes to use code augmentation techniques offered by the Linux kernel to efficiently gather data that can be used to spot covert channels. The main advantage of the idea is that multiple “filters” can be stacked without causing too many degradations and very different protocol and software entities can be accessed via a unique programming interface. Data gathering is important not only for performing detection of network covert channels and stegomalware. In fact, being able to implement a warden to efficiently collect information on traffic flows is the prime requirement for populating datasets to be jointly used with machine-learning-capable mechanisms. Unfortunately, the very composite set of features exploited by storage channels highly limits the performances of AI-based methods [26], but a possible improvement can be achieved by searching for well-known, recurrent embedding

patterns [27]. Similarly, application-layer firewalls (e.g., entities able to inspect traffic to perform keyword-based analysis) can spot a hidden communication but paying in terms of capabilities and engineering efforts [28]. To develop successful detection mechanisms, the second path relies upon the design of high-level methods or approaching the detection problem in a more general manner. As possible examples, Ref. [29] showcases the use of visualization tools to ease the development of countermeasures by helping professionals working in threat intelligence with the specific aim of spotting network steganography attacks.

4.3. Packet Length and Inter-Protocol Interdependencies

Modulation of the length of packets is another popular approach used to implement network covert channels and can be applied to different protocols (e.g., IPv6 datagrams [30] or packets containing audio conversations [31]). Additionally in this case, there is not a one-fits-all approach, but the development of some high-level indicators can be the basis to design universal detectors, e.g., by exploiting ubiquitous machine learning classifiers [32] or frameworks able to spot statistical anomalies [33]. In this perspective, the major challenge concerns the ability of engineering an effective dataset for training and tuning the warden.

Additionally, advanced channels exploiting inter-protocol relationships are emerging [34]. Even if the work addresses real-time services, such principles can be also extended to embedding schemes leveraging packet lengths. To perform detection of such a complex method using multiple/different protocols at the same time, a promising mechanism exploits traffic coloring.

4.4. IPv6

The advent of IPv6 opens to many potential covert channels. Specifically, the attacker can exploit new features offered by the protocol or the various transitional and tunneling mechanisms designed to allow IPv4 and IPv6 to coexist. Despite the rich set of features offered by IPv6, real network deployments scale back the different types of covert channels that can be used to launch a real attack [30]. In fact, the amount of IPv6 traffic is still limited (compared to IPv4) and its statistical variability can make some channels highly detectable. For instance, when using the `Traffic Class` field to contain secrets, real traces are characterized only by three different values, thus limiting the bandwidth of the covert channel to few bit/s. A further constraint is due to the Explicit Congestion Notification feature, which is rarely used in real-world nodes, thus causing the alteration of the field as highly suspicious.

As a consequence, the development of detection techniques targeting IPv6 traffic seldom necessitates of additional hardware/software and can be implemented on top of normal traffic inspection frameworks. Similar to many other threats, another emerging approach leverages machine learning. As an example, the work in [35] demonstrates the use of fuzzy logic and genetic algorithms to achieve high detection rates of covert channels targeting the IPv6 header. We point out that both methods require to employ deep packet inspection principles, thus impacting on the scalability of the approach.

Transitional mechanisms can be also abused to create network covert channels or exfiltrate information. According to [36], modern intrusion detection systems are affected by major drawbacks when handling IPv6 traffic. For instance, covert channels targeting transition methods using both IPv4 and IPv6 traffic are difficult to detect compared to other attacks exploiting DNS or SSH tunneling.

4.5. VoIP

Owing to its ubiquitous availability and the potential capacity when used as a carrier, many works concentrated on the detection of covert channels targeting VoIP traffic. Alas, VoIP streams can be exploited in different parts, e.g., in the vocal payload or in the resulting network traffic. Specifically, many techniques use some form of audio steganography or digital watermarking for blending an arbitrary message within the audio of the talker. In general, a common approach for detecting the presence of secret information requires the

warden to compute some audio quality metric to reveal the hidden data [37]. However, this could not be a simple task, as the attacker may take advantage of specific features of the used codec, or transcode the conversation to free space in packets to embed information while maintaining the same level of quality [38]. Therefore, the performance of the detection can be influenced by the original codec both in terms of performance and scalability. A possible improvement could exploit information provided by the network traffic in order to reveal “misuse patterns” [39]. For instance, the warden can identify a covert channel within a VoIP flow by using statistical metrics like the variance of inter-arrival time of packets composing the Real-time Transport Protocol (RTP) stream. The idea of using simple metrics and calculations for spotting the presence of the channel has not been considered only to optimize performances and extendability of a warden. Specifically, computing metrics on the audio signal could be unfeasible due to encryption or to not break the privacy of users. In this vein, lower-level and simpler approaches should be carefully evaluated. As an example, in [40] a scheme exploiting a matrix containing sketches of packets is presented. Another challenge concerns the ability of detecting short communications or channels transferring a restricted amount of information. In this case, the adoption of some form of artificial intelligence looks very promising [41] but at the price of being able to collect suitable data for training the detection framework.

Lastly, if the VoIP stream exploits some form of standard technologies like the Session Initiation Protocol (SIP), the warden can try to detect the hidden communication by analyzing anomalous distributions of values in the headers, bursts of methods or deviation in the traffic statistics of the resulting RTP and SIP flows [31,42]. A major challenge concerns the development of a suitable warden able to perform detection without disrupting the real-time, interactive nature of VoIP communications and SIP signaling. Emerging approaches exploit the use of application-agnostic parameters (e.g., the size of the protocol data units) and deep the detection at the codec or audio level only if packets are considered potentially suspicious. Moreover, algorithms used for traffic coloring can offer an effective foundation to reveal hidden data within RTP and Realtime Transport Control Protocol streams [34].

4.6. DNS and HTTP Tunnels

A class of threats often exploited in the perspective of developing network covert channels is the one relying upon the creation of tunnels. Put briefly, the attacker could try to exfiltrate data or establish abusive communications by injecting malicious traffic within legitimate traffic flows. Even if not strictly related to network steganography or network covert channels, many threats demonstrated the risk of not performing any detection attempts against tunneling approaches. Literature on tunneling often deviates from the one belonging to covert channels but, at the same time, exhibits important overlaps. As a consequence, we also present major ideas and some research questions on DNS and HTTP tunnels, which are the most popular. Observations presented in this paper can be extended straightforwardly to other similar approaches, for instance malware abusing SSH connections.

Concerning the detection of DNS tunnels, the work in [43] presents a method based on uni-bi-tri-grams frequencies of characters of domains resolved via DNS queries. Specifically, since the domain names are mostly characterized by distributions and patterns similar to those describing natural languages, discrepancies or deviations can be used as effective indicators to reveal the presence of hidden data. A different approach, leveraging Markov decision process, is presented in [44]. Tunnels leveraging DNS traffic can be also detected by using a more network-oriented approach. For instance, the throughput of DNS flows can be considered a valid indicator to detect anomalous usages [45]. The importance of being able to reveal the presence of DNS tunnels within a network infrastructure is expected to increase in the future. In fact, many APTs are progressively using domain fronting techniques (i.e., the generation of synthetic domain names and related DNS queries to exfiltrate data or orchestrate a botnet), which can be efficiently spotted via ad-hoc classifiers [46]. Flux/DNS fronting techniques require a sufficient amount of data

to train classifiers and non-negligible computing power to deploy such frameworks in real networks without impacting the quality experienced by end users.

Indeed, being able to identify HTTP tunnels is also important since HTTP is typically used to develop custom signaling protocols or to retrieve additional malware components [1,2]. Owing to such a popularity, many security appliances operating at the application layer can already inspect or filter HTTP traffic. However, in the perspective of making the detection of covert channels, using more general and technology-independent mechanisms could be beneficial. For instance, in [47] authors show how evaluating the inter arrival time, size and ordering of the packets can be adopted to discriminate between legitimate HTTP traffic and tunneled contents. The aforementioned quantities can be used to compute metrics and train machine learning mechanisms [48], possibly via approaches extracting features in an automated manner [49]. However, the tight coupling of the method creating the covert communication path with the protocol internals still remains the major challenge to face. Specifically, an attacker could attempt to evade detection or reduce the performance of a classifier by altering information like the User-Agent [50]. Another aspect to be considered when engineering a warden to spot HTTP-based covert channels is the ability of an attacker of evading network-based detection deliberately. For instance, it can encode data in the number of hyperlinks and manipulate referrals [51].

5. Covert Channel Limitation, Elimination and Prevention

As discussed, upon detecting the covert channel, further actions to be performed can have different functional goals (e.g., limitation or elimination). Usually, applying a countermeasure to a class of covert channels requires to choose among different tradeoffs. Specifically:

- **security vs. quality:** the limitation or elimination capacity could require reducing the quality perceived by users. For instance, to prevent an attack, the entity in charge of performing the sanitization of the network traffic may delay or drop packets in an unacceptable manner;
- **accuracy vs. performance:** despite the used hardware and software architecture, the processing capacity of the warden (including various middleboxes for enforcing network security) could impact on the behavior of the traffic (i.e., by altering its quality parameters) or require an amount of resources not feasible (e.g., computing and storage capacity). Therefore, the accuracy of the limitation and elimination process should be carefully evaluated especially in terms of overheads;
- **complexity vs. cost:** the degree of sophistication of a covert channel may account for complex mitigation methods. This usually leads to software being developed, hardware resources, or alteration of the network infrastructure, for instance to duplicate traffic flows or to avoid performance bottlenecks. Clearly, such requirements lead to economical expenditures and maintenance costs, which should be carefully evaluated;
- **blockage vs. functionalities:** with the term blockage, we intend the ability of impeding that a given network protocol is used as a carrier. This typically requires to block a protocol for a network or host, or reduce some functionalities. Even if “mutilating” the protocol stack can be admissible in scenarios with high security constraints, this tradeoff should be carefully evaluated, especially for networks operating not in a local manner;
- **security vs. risk:** being able to completely eliminate a channel could not be mandatory or possible in terms of the aforementioned tradeoffs. Thus, according to the considered scenario, it could be more convenient to only limit the steganographic bandwidth of the channel. For instance, a channel of few bit/s could be enough to activate or configure a backdoor, but not sufficient to exfiltrate an industrial secret. Usually, this is at the basis of the “arms race” between defender and attacker, trying to slow the data exfiltration rate or the steganographic bandwidth to remain under the radar [52].

We point out that the aforementioned tradeoffs also apply in the case of detection, but the limitation of a channel and the sanitization of network traffic usually require greater amounts of resources to avoid performance bottlenecks or impair some network functionalities.

5.1. Elimination and Limitation

One of the first ideas to tame covert channels is the Network Pump, theorized and designed by Kang, Moskowitz and Lee in a paper published in 1996 [53,54]. The Pump has been originally designed to prevent the possibilities of using a fraction of the traffic (e.g., ACKs) to build covert channels between processes with different security levels (i.e., high and low in the original work). In essence, it can be used to connect different portions of the networks and data are “pumped” upon proper queuing and ACKs are sent in a probabilistic manner as to reduce the possibility that an attacker can create a covert channel. An important contribution of the Pump is making clear that it is only capable of limiting the risk of setting up a covert channel.

In general, the complete elimination of the chance that a network can be used for setting up a covert channel is an inefficient and unrealistic goal [55]. For instance, guaranteeing that HTTP will not be used to tunnel information can be only achieved by completely blocking its utilization in a given scenario, which is mostly unfeasible. Instead, it would be possible to design a countermeasure to pursue a *tradeoff* between the security risk and the functioning of the network both in terms of performances and functionalities. Always referring to the HTTP case, a possible approach tries to “sanitize” the various flows in order to reduce the carriers that can be used to contain secret information [56]. In this case, the semantic of the protocol is evaluated and restored as close as possible to the one prescribed by the RFCs. Thus, the countermeasure mainly acts by reducing the ambiguities (e.g., the use of capitalized strings within the protocol to encode data) to make the channel less attractive by the attacker or limiting its bandwidth. Alas, objects exchanged via HTTP can still contain hidden data. In this case, ad-hoc proxies can be deployed to check inline objects against steganographic contents and sanitize them [57]. This requires a non-negligible amount of computing resources and also accounts for the presence of specific detectors/sanitizers for each considered inline object (e.g., protocol headers, images, audio, video and additional plugins). Similar consideration can be done for other applications, e.g., for the VoIP. For instance, network-related protocols can be restored, whereas random noise can be added to the voice information as to disrupt or reduce the performances of the covert channel [37]. The examples allow to underline the magnitude of the problem space to be faced for completely eliminating a channel, especially when there is the need of operating at higher levels of the protocol architecture. We point out that, this sort of pseudo-proxying could break the end-to-end semantics of the overt traffic flow, thus leading to additional security and privacy issues.

For the case of storage channels, classical approaches aim at restoring unused fields within header to default values whenever possible or to insert specific padding sequences to overwrite manipulation attempts [9]. An interesting aspect to be considered regards how protocols designed decades ago now behave in the modern Internet. Taking as a paradigmatic example IPv6, the investigation in [30] highlights that the diffuse lack of networks supporting Quality of Service guarantees permits an attacker to use the Traffic Class to embed data. Instead, in some circumstances, the network could protect itself. In fact, the presence of transitional mechanisms or middleboxes (e.g., network address translation) account for manipulations of the header and buffering operations, which can disrupt the most fragile covert channels. A further aspect to consider is the increasing use of encryption and secure variant of protocols. This multiplies the opportunities for an attacker to have additional carriers but also boost the design of “more secure” implementations to mitigate the risks of being abused to set covert channels. To illustrate this concept, we consider the case of IPSec [58], which can be natively endowed with padding for preventing covert channels modulating the packet size, and techniques for adding random errors for disrupting mechanisms embedding data by artificially manipulating packet losses. The limit of such an approach is the need of updating implementations and perform such optimizations on a per-protocol basis. Instead, it could be beneficial centralizing such workarounds in a unique network entity.

Timing channels appears to be more easy to limit or prevent, at least those manipulating inter packet statistics. In this case, buffering and adding random delays can be sufficient. The drawback of this class of countermeasures is rooted in the indiscriminate penalization of all traffic flows. When in the presence of traffic with real-time constraints, buffering can not be too aggressive, thus leaving the attacker with chances of creating a channel. In this case, the goal is to reduce its steganographic bandwidth in order to render the channel unsuitable for the attack. Literature offers different techniques for timing channels. We mention the use entropy-based schemes to make the detection more robust to prevent the attacker from limiting his/her rate to evade the detection [59], and addressing in a more detail timing channels mimicking realistic patterns [60].

5.2. Prevention

An important aspect to consider to fully assess the security of network/computing infrastructures concerns how to completely prevent the possibility of setting up a network covert channel. As hinted, protocols known to be prone to steganographic attacks should be completely blocked [1,8,55]. Unfortunately, this is unfeasible in many scenarios and use cases. Since covert channels are the prime consequence of imperfect isolation, prevention mechanisms should aim at restoring the protocol data units to their basic form, in order to reduce the chance of leaking states or embedding additional information. In this vein, reverting fields to default values or inserting padding (or trimming messages) are the methodologies at the basis of many countermeasures [9].

However, the increasing sophistication of exfiltration attempts as well as the complexity of modern network architectures require to consider covert channels from the early stage of the design phase. Thus, prevention should be considered an “offline” process and handled in a *security-by-design* manner. To this extent, standardization surely plays a major role and security considerations on potential covert channels should be always present. Enforcing a flow to adhere to its standard implementation can be also exploited at runtime to sanitize the behavior of the protocol and disrupt secret messages, if present. Covert channels can be also impeded by selectively blocking some protocol features while pursuing functional tradeoffs acceptable for the specific scenario or application.

6. Trends and Challenges

The growing adoption of covert channels as a paradigm to empower stegomalware and support data exfiltration is expected to persist in the future [1]: new attacks will be more sophisticated, cross-layer and able to leverage the complex interplay of hardware and software components [52]. This will magnify the gap between the attacker and the defender, especially in terms of uncertainty of what to inspect and where to place wardens within the overall network architecture. Reviewed countermeasures as well as tools deployed in the wild seem only partially trying to mitigate such an aspect. Thus, the main trends in the development of countermeasures observed and the challenges to be faced are:

- **generalization:** data gathering as well as detection and sanitization of network covert channels may require to deploy a wide range of collection techniques and to inspect several heterogeneous carriers [2]. As regards the collection phase, literature showcased the use of in-kernel mechanisms to analyze the various carriers exploited to set local covert channels (see, e.g., [25] for the use of the extended Berkeley Packet Filter to spot stealth data exchange via alteration of Unix file permissions). Future wardens should be able to exploit similar techniques to have general filters that can be stacked up when the used carrier is uncertain or impossible to isolate a priori. Moreover, performing kernel-level analysis could help in avoiding bottlenecks or mitigating overheads causing legitimate flows to experience additional delays;
- **abstraction:** future channels are expected to target different parts of modern network ecosystems, such as appliances and IoT nodes. In this vein, network covert channels will be able to use a multitude of carriers, mainly characterized by a very composite set of protocols, sensors and architectural blueprints. In this perspective, an interesting

approach concerns the development of more abstract metrics, which can be used to spot the steganographic attack despite the used carrier. Literature demonstrated that process correlation [61] or anomalous energy consumption and drains [62,63] are excellent candidates to make the detection more threat-independent. To pursue such a vision, the detection could need to be shifted towards the border of the network or close to users. This poses several challenges, for instance in terms of avoiding performance degradation of devices of end users and additional security requirements;

- **cloud:** without doubts, cloud infrastructures will remain a relevant component of the future Internet. There are many works highlighting that covert channels are progressively used to attack virtualized environments mainly to let virtual machines to collude and evade security frameworks (see [64] for a review on the topic). Concerning network covert channels, recent investigations showcased how cloud architectures can be abused to set up Internet-wide communication paths [65]. As a consequence, cloud and virtualized environments should be protected against information-hiding-capable threats [66] and data breaches orchestrated via network covert channels;
- **everything-as-a-service:** as shown, being able to detect covert channels may require a non-negligible amount of storage (e.g., to store traffic dumps or datasets) and computing power (e.g., to compute metrics or run machine-learning-capable frameworks). Being able to deploy wardens for protecting large-scale networks or running sophisticated detection software while delivering satisfactory performances could be unfeasible for small-medium actors. Therefore, a relevant effort should be done to explore Warden-as-a-Service approaches [3,4];
- **reversibility:** in reversible network covert channels, the two secret endpoints communicating are able to “restore” the traffic to its original form (i.e., before the injection of the hidden data) in order to avoid detection [67,68]. To spot this class of emerging threats, the warden should collect the traffic in different portions of the network and do some form of comparison. This poses several challenges, not only limited to technical aspects. For instance, the needed traffic could be collected in environments of different operators leading to technical and legal issues;
- **resistance by design:** the ability of storing arbitrary data or manipulating a protocol are usually consequence of an ambiguous design or an imperfect implementation. An interesting research activity deals with the identification of patterns at the basis of the creation of network covert channels [7] and should be considered when designing the next-generation of secure-by-design protocols. Nevertheless, the application of formal methods to endow a warden with the ability of performing runtime checks or model unknown threats is another promising trend to explore.

7. Conclusions

In this paper, we have presented a thorough review on countermeasures against network covert channels. We have addressed the different steps needed to enforce security when in the presence of hidden communications. In more detail, we have considered works dealing with the detection, limitation and elimination of network covert channels. We have also addressed methodologies to prevent hidden communications. Even if comprehensive, the investigation revealed some imperfections. For instance, the complex continuum of computing and networking platforms leads to a problem space not well delimited. Thus, it has been difficult tracing a clear line between network covert channels and other types of threats. For instance, covert channels targeting cloud scenarios could be quickly become relevant for the network case (and vice versa).

As shown, the majority of frameworks and ideas handles security issues caused by stegomalware and network covert channels in an almost unique threat-dependent flavor, mainly due to the strict dependency of the behavior of the channel on the used carrier or injection mechanism. This magnifies the asymmetry between the attacker and the defender. To partially recover to this, future research should focus on the development of more general indicators and frameworks, as well as methodologies to inspect multiple carriers

at once. Such a trend is partially observable in the literature (see, e.g., energetic indicators to abstract the underlying threat) but, to be effective, should be consolidated. Another interesting path to pursue, which requires long term research efforts, concerns increasing the awareness of developers and engineers of risks caused by network covert channels. In this case, different research areas (e.g., networking and software engineering) could find a common “playground” and do not duplicate efforts or develop overlapped solutions. Moreover, awareness should lead to adopt methodologies for preventing the exploitation of ambiguities already in early design phases both of protocols, services and architectures.

Funding: This research was funded by EU Project SIMARGL—Secure Intelligent Methods for Advanced Recognition of Malware and Stegomalware (Grant Agreement No 833042) and the EU Project ASTRID—AddreSSing ThReats for virtualIzeD services (Grant Agreement No 786922).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mazurczyk, W.; Caviglione, L. Information Hiding as a Challenge for Malware Detection. *IEEE Secur. Priv.* **2015**, *13*, 89–93. [\[CrossRef\]](#)
2. Mazurczyk, W.; Wendzel, S. Information hiding: Challenges for forensic experts. *Commun. ACM* **2017**, *61*, 86–94. [\[CrossRef\]](#)
3. Cabaj, K.; Caviglione, L.; Mazurczyk, W.; Wendzel, S.; Woodward, A.; Zander, S. The new threats of information hiding: The road ahead. *IT Prof.* **2018**, *20*, 31–39. [\[CrossRef\]](#)
4. Caviglione, L.; Wendzel, S.; Mazurczyk, W. The future of digital forensics: Challenges and the road ahead. *IEEE Secur. Priv.* **2017**, *15*, 12–17. [\[CrossRef\]](#)
5. Lampson, B.W. A Note on the Confinement Problem. *Commun. ACM* **1973**, *16*, 613–615. [\[CrossRef\]](#)
6. Falzon, K.; Bodden, E. Towards a Comprehensive Model of Isolation for Mitigating Illicit Channels. In Proceedings of the International Conference on Principles of Security and Trust, Eindhoven, The Netherlands, 4–5 April 2016; pp. 116–138.
7. Wendzel, S.; Zander, S.; Fechner, B.; Herdin, C. Pattern-based survey and categorization of network covert channel techniques. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–26. [\[CrossRef\]](#)
8. Zander, S.; Armitage, G.; Branch, P. A survey of covert channels and countermeasures in computer network protocols. *IEEE Commun. Surv. Tutor.* **2007**, *9*, 44–57. [\[CrossRef\]](#)
9. Mazurczyk, W.; Caviglione, L. Steganography in modern smartphones and mitigation techniques. *IEEE Commun. Surv. Tutorials* **2014**, *17*, 334–357. [\[CrossRef\]](#)
10. Caviglione, L.; Mazurczyk, W.; Wendzel, S. Advanced Information Hiding Techniques for Modern Botnets. In *Botnets: Architectures, Countermeasures, and Challenges*; CRC Press: Boca Raton, FL, USA, 2019; pp. 165–199.
11. Mileva, A.; Panajotov, B. Covert channels in TCP/IP protocol stack - extended version. *Cent. Eur. J. Comput. Sci.* **2014**, *4*, 45–66. [\[CrossRef\]](#)
12. Fridrich, J.; Pevný, T.; Kodovský, J. Statistically undetectable jpeg steganography: Dead ends challenges, and opportunities. In Proceedings of the 9th Workshop on Multimedia & Security, Dallas, TX, USA, 20–21 September 2007; pp. 3–14.
13. Mazurczyk, W. VoIP steganography and its detection a survey. *ACM Comput. Surv. (CSUR)* **2013**, *46*, 1–21. [\[CrossRef\]](#)
14. Simmons, G.J. The prisoners’ problem and the subliminal channel. In *Advances in Cryptology*; Springer: Boston, MA, USA, 1984; pp. 51–67.
15. Mazurczyk, W.; Wendzel, S.; Chourib, M.; Keller, J. Countering adaptive network covert communication with dynamic wardens. *Future Gener. Comput. Syst.* **2019**, *94*, 712–725. [\[CrossRef\]](#)
16. Cabuk, S.; Brodley, C.E.; Shields, C. IP covert timing channels: Design and detection. In Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, DC, USA, 25–29 October 2004; pp. 178–187.
17. Stillman, R.M. Detecting IP covert timing channels by correlating packet timing with memory content. In Proceedings of the IEEE SoutheastCon 2008, Huntsville, AL, USA, 3–6 April 2008; pp. 204–209.
18. Rezaei, F.; Hempel, M.; Sharif, H. Towards a reliable detection of covert timing channels over real-time network traffic. *IEEE Trans. Dependable Secur. Comput.* **2017**, *14*, 249–264. [\[CrossRef\]](#)
19. Shrestha, P.L.; Hempel, M.; Rezaei, F.; Sharif, H. A support vector machine-based framework for detection of covert timing channels. *IEEE Trans. Dependable Secur. Comput.* **2015**, *13*, 274–283. [\[CrossRef\]](#)
20. Luo, X.; Chan, E.W.; Chang, R.K. TCP covert timing channels: Design and detection. In Proceedings of the 2008 IEEE International Conference on Dependable Systems and Networks with FTCS and DCC (DSN), Anchorage, AK, USA, 24–27 June 2008; pp. 420–429.
21. Biswas, A.K.; Ghosal, D.; Nagaraja, S. A survey of timing channels and countermeasures. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–39. [\[CrossRef\]](#)
22. Nowakowski, P.; Zórawski, P.; Cabaj, K.; Mazurczyk, W. Network covert channels detection using data mining and hierarchical organisation of frequent sets: An initial study. In Proceedings of the 15th International Conference on Availability, Reliability and Security, Dublin, Ireland, 17–20 August 2020; pp. 1–10.

23. Wendzel, S. Protocol-independent Detection of “Messaging Ordering” Network Covert Channels. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–8.
24. Chakkaravarthy, S.S.; Sangeetha, D.; Vaidehi, V. A Survey on malware analysis and mitigation techniques. *Comput. Sci. Rev.* **2019**, *32*, 1–23. [\[CrossRef\]](#)
25. Carrega, A.; Caviglione, L.; Repetto, M.; Zuppelli, M. Programmable Data Gathering for Detecting Stegomalware. In Proceedings of the 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 29 June–3 July 2020; pp. 422–429.
26. Cho, D.; Thuong, D.; Dung, N. A Method of Detecting Storage Based Network Steganography Using Machine Learning. *Procedia Comput. Sci.* **2019**, *154*, 543–548. [\[CrossRef\]](#)
27. Ayub, M.A.; Smith, S.; Siraj, A. A Protocol Independent Approach in Network Covert Channel Detection. In Proceedings of the 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, 1–3 August 2019; pp. 165–170.
28. Gilbert, P.A.; Bhattacharya, P. An approach towards anomaly based detection and profiling covert TCP/IP channels. In Proceedings of the 2009 7th International Conference on Information, Communications and Signal Processing (ICICS), Macau, China, 8–10 December 2009; pp. 1–5.
29. Mazurczyk, W.; Szczypiorski, K.; Jankowski, B. Towards steganography detection through network traffic visualisation. In Proceedings of the 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems, St. Petersburg, Russia, 3–5 October 2012; pp. 947–954.
30. Mazurczyk, W.; Powójski, K.; Caviglione, L. IPv6 covert channels in the wild. In Proceedings of the Third Central European Cybersecurity Conference, Munich, Germany, 14–15 November 2019; pp. 1–6.
31. Saenger, J.; Mazurczyk, W.; Keller, J.; Caviglione, L. VoIP network covert channels to enhance privacy and information sharing. *Future Gener. Comput. Syst.* **2020**, *111*, 96–106. [\[CrossRef\]](#)
32. Nair, A.S.; Sur, A.; Nandi, S. Detection of packet length based network steganography. In Proceedings of the 2010 International Conference on Multimedia Information Networking and Security, Nanjing, China, 4–6 November 2010; pp. 574–578.
33. Sur, A.; Nair, A.S.; Kumar, A.; Jain, A.; Nandi, S. Steganalysis of network packet length based data hiding. *Circuits Syst. Signal Process.* **2013**, *32*, 1239–1256. [\[CrossRef\]](#)
34. Lehner, F.; Mazurczyk, W.; Keller, J.; Wendzel, S. Inter-protocol steganography for real-time services and its detection using traffic coloring approach. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, 9–12 October 2017; pp. 78–85.
35. Salih, A.; Ma, X.; Peytchev, E. Implementation of hybrid artificial intelligence technique to detect covert channels attack in new generation internet protocol IPv6. In *Leadership, Innovation and Entrepreneurship as Driving Forces of the Global Economy*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 173–190.
36. Blumbergs, B.; Pihelgas, M.; Kont, M.; Maennel, O.; Vaarandi, R. Creating and detecting IPv6 transition mechanism-based information exfiltration covert channels. In Proceedings of the Nordic Conference on Secure IT Systems, Oulu, Finland, 2–4 November 2016; pp. 85–100.
37. Takahashi, T.; Lee, W. An assessment of VoIP covert channel threats. In Proceedings of the 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007, Nice, France, 17–21 September 2007; pp. 371–380.
38. Janicki, A.; Mazurczyk, W.; Szczypiorski, K. Steganalysis of transcoding steganography. *Ann. Telecommun. Ann. Télécommun.* **2014**, *69*, 449–460. [\[CrossRef\]](#)
39. Pelaez, J.C. Using misuse patterns for voip steganalysis. In Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application, Linz, Austria, 31 August–4 September 2009; pp. 160–164.
40. Garateguy, G.; Arce, G.R.; Pelaez, J. Covert channel detection in VoIP streams. In Proceedings of the 2011 45th Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 23–25 March 2011; pp. 1–6.
41. Lin, Z.; Huang, Y.; Wang, J. RNN-SM: Fast steganalysis of VoIP streams using recurrent neural network. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1854–1868. [\[CrossRef\]](#)
42. Zhao, H.; Zhang, X. Sip steganalysis using chaos theory. In Proceedings of the 2012 International Conference on Computing, Measurement, Control and Sensor Network, Taiyuan, China, 7–9 July 2012; pp. 95–100.
43. Born, K.; Gustafson, D. Detecting dns tunnels using character frequency analysis. *arXiv* **2010**, arXiv:1004.4358.
44. Mc Carthy, S.M.; Sinha, A.; Tambe, M.; Manadhata, P. Data exfiltration detection and prevention: Virtually distributed pomdps for practically safer networks. In Proceedings of the International Conference on Decision and Game Theory for Security, New York, NY, USA, 2–4 November 2016; pp. 39–61.
45. Himbeault, M. A Novel Approach to Detecting Covert dns Tunnels Using Throughput Estimation. Master’s Thesis, University of Manitoba, Department of Electrical and Computer Engineering, Winnipeg, MB, Canada, 2014.
46. Antonakakis, M.; Perdisci, R.; Nadji, Y.; Vasiloglou, N.; Abu-Nimeh, S.; Lee, W.; Dagon, D. From throw-away traffic to bots: detecting the rise of DGA-based malware. In Proceedings of the 21st {USENIX} Security Symposium ({USENIX} Security 12, Bellevue, WA, USA, 11–13 August 2021; pp. 491–506.
47. Crotti, M.; Dusi, M.; Gringoli, F.; Salgarelli, L. Detecting http tunnels with statistical mechanisms. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 6162–6168.

48. Ding, Y.J.; Cai, W.D. A method for HTTP-tunnel detection based on statistical features of traffic. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27–29 May 2011; pp. 247–250.
49. Davis, J.J.; Foo, E. Automated feature engineering for HTTP tunnel detection. *Comput. Secur.* **2016**, *59*, 166–185. [[CrossRef](#)]
50. Schwenk, G.; Rieck, K. Adaptive detection of covert communication in http requests. In Proceedings of the 2011 Seventh European Conference on Computer Network Defense, Gothenburg, Sweden, 6–7 September 2011; pp. 25–32.
51. Luo, X.; Zhou, P.; Chan, E.W.; Chang, R.K.; Lee, W. A combinatorial approach to network covert communications with applications in web leaks. In Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), Hong Kong, China, 27–30 June 2011; pp. 474–485.
52. Caviglione, L.; Choraś, M.; Corona, I.; Janicki, A.; Mazurczyk, W.; Pawlicki, M.; Wasielewska, K. Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection. *IEEE Access* **2020**, *9*, 5371–5396. [[CrossRef](#)]
53. Kang, M.H.; Moskowitz, I.S.; Lee, D.C. A network pump. *IEEE Trans. Softw. Eng.* **1996**, *22*, 329–338. [[CrossRef](#)]
54. Kang, M.H.; Moskowitz, I.S.; Chincheck, S. The pump: A decade of covert fun. In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05), Tucson, AZ, USA, 5–9 December 2005; p. 7.
55. Zander, S.; Armitage, G.; Branch, P. Covert channels and countermeasures in computer network protocols [reprinted from iee communications surveys and tutorials]. *IEEE Commun. Mag.* **2007**, *45*, 136–142. [[CrossRef](#)]
56. Schear, N.; Kintana, C.; Zhang, Q.; Vahdat, A. Glavlit: Preventing exfiltration at wire speed. Irvine Burn. In Proceedings of Fifth Workshop on Hot Topics in Networks (Hot-Nets), Irvine, CA, USA, 29–30 November 2006; Volume 133.
57. Blasco, J.; Hernandez-Castro, J.C.; de Fuentes, J.M.; Ramos, B. A framework for avoiding steganography usage over HTTP. *J. Netw. Comput. Appl.* **2012**, *35*, 491–501. [[CrossRef](#)]
58. Schulz, S.; Varadharajan, V.; Sadeghi, A.R. The silence of the LANs: efficient leakage resilience for IPsec VPNs. *IEEE Trans. Inf. Forensics Secur.* **2013**, *9*, 221–232. [[CrossRef](#)]
59. Walls, R.J.; Kothari, K.; Wright, M. Liquid: A detection-resistant covert timing channel based on IPD shaping. *Comput. Netw.* **2011**, *55*, 1217–1228. [[CrossRef](#)]
60. Gianvecchio, S.; Wang, H.; Wijesekera, D.; Jajodia, S. Model-based covert timing channels: Automated modeling and evasion. In Proceedings of the International Workshop on Recent Advances in Intrusion Detection, Cambridge, MA, USA, 15–17 September 2008; pp. 211–230.
61. Urbanski, M.; Mazurczyk, W.; Lalande, J.F.; Caviglione, L. Detecting local covert channels using process activity correlation on android smartphones. *Int. J. Comput. Syst. Sci. Eng.* **2017**, *32*, 71–80.
62. Caviglione, L.; Gaggero, M.; Lalande, J.F.; Mazurczyk, W.; Urbanski, M. Seeing the unseen: revealing mobile malware hidden communications via energy consumption and artificial intelligence. *IEEE Trans. Inf. Forensics Secur.* **2015**, *11*, 799–810. [[CrossRef](#)]
63. Caviglione, L.; Gaggero, M.; Cambiaso, E.; Aiello, M. Measuring the energy consumption of cyber security. *IEEE Commun. Mag.* **2017**, *55*, 58–63. [[CrossRef](#)]
64. Betz, J.; Westhoff, D.; Müller, G. Survey on covert channels in virtual machines and cloud computing. *Trans. Emerg. Telecommun. Technol.* **2017**, *28*, e3134. [[CrossRef](#)]
65. Caviglione, L.; Podolski, M.; Mazurczyk, W.; Ianigro, M. Covert channels in personal cloud storage services: The case of Dropbox. *IEEE Trans. Ind. Inform.* **2016**, *13*, 1921–1931. [[CrossRef](#)]
66. Liu, A.; Chen, J.; Yang, L. Real-time detection of covert channels in highly virtualized environments. In Proceedings of the International Conference on Critical Infrastructure Protection, Hanover, NH, USA, 23–25 March 2011; pp. 151–164.
67. Mazurczyk, W.; Szary, P.; Wendzel, S.; Caviglione, L. Towards reversible storage network covert channels. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–8.
68. Szary, P.; Mazurczyk, W.; Wendzel, S.; Caviglione, L. Design and performance evaluation of reversible network covert channels. In Proceedings of the 15th International Conference on Availability, Reliability and Security, Dublin, Ireland, 25–28 August 2020; pp. 1–8.