Ann. Telecommun. (2014) 69:417–430
DOI 10.1007/s12243-014-0423-x

Hidden and under control

A survey and outlook on covert channel-internal control protocols

Steffen Wendzel · Jörg Keller

Received: 11 June 2013 / Accepted: 7 January 2014 / Published online: 18 March 2014

Abstract Network covert channels are policy-breaking and stealthy communication channels in computer networks. These channels can be used to bypass Internet censorship, to exfiltrate data without raising attention, to allow a safe and stealthy communication for members of political oppositions and for spies, to hide the communication of military units at the battlefield from the enemy, and to provide stealthy communication for today's malware, especially for botnets. To enhance network covert channels, researchers started to add protocol headers, so-called *micro-protocols*, to hidden payload in covert channels. Such protocol headers enable fundamental features such as reliability, dynamic routing, proxy capabilities, simultaneous connections, or session management for network covert channels—features which enrich future botnet communications to become more adaptive and more stealthy than nowadays. In this survey, we provide the first overview and categorization of existing micro-protocols. We compare micro-protocol features and present currently uncovered research directions for these protocols. Afterwards, we discuss the significance and the existing means for micro-protocol engineering. Based on our findings, we propose further research directions for micro-protocols. These features include to introduce multi-layer protocol stacks, peer auto-configuration, and peer group communication based on micro-protocols, as well as to develop protocol translation in order to achieve inter-connectivity for currently separated overlay networks.

Keywords Network covert channel · Covert channel-internal control protocols · Micro-protocols · Information hiding

S. Wendzel (✉)
Fraunhofer FKIE, Bonn, Germany
e-mail: steffen@wendzel.de

J. Keller
Faculty of Mathematics and Computer Science,
FernUniversität in Hagen, Hagen, Germany
e-mail: joerg.keller@FernUni-Hagen.de

1 Introduction

In 1973, covert channels were introduced as policy-breaking communication channels not foreseen in a system design [24]. Later, the term *network covert channel* was defined as a policy-breaking and *stealthy* communication channel [7, 32]. Murdoch mentions the fact that network covert channels must not be stealthy in any case but can instead be placed in steganographic channels to add stealthiness to the channels [33]. In the remainder, we use the term network covert channel as a stealthy and potentially policy-breaking communication channel.

In comparison to a cryptographically secured communication channel, a network covert channel is not used to prevent that a third party can read the transferred secret information but to prevent that the communication itself will being detected. Network covert channels are a dual-use good, i.e., they can be used for good and bad, peaceful, and military purposes. For instance, botnets can improve the stealthiness of their command and control channels with network covert channels [15]. On the other hand, journalists can apply network covert channels to secretly transfer illicit information in networks with censorship. Thus, network covert channels can contribute to the free expression of opinions [54].

Two different classes of network covert channels were previously identified [54]:

1. *Network covert storage channels* hide data in currently unused fields of network protocols.
2. *Network covert timing channels* hide data in timing information, such as timing variations between network packets or by manipulating packet order.

In the sequel, we focus on network covert storage channels since only storage channels provide enough space to transfer control protocols. Besides, *hybrid* covert channels exist which can combine storage and timing channels. As for all storage channels, control protocols can be embedded into the storage channels of hybrid covert channels as well.

A large number of publications deal with the problem of placing hidden data in network packets, e.g., [1, 9, 14, 16, 23, 27, 35, 37, 38, 41, 50], and many techniques were developed to detect and prevent network covert channels, e.g., [5, 12, 17, 26, 39, 40, 54]. We do not focus on such network protocol-specific hiding techniques as others already cover these aspects in detail [54].

Network covert channels only provide a communication channel. To extend their capabilities, covert channel-internal control protocols, so called *micro-protocols* were introduced. A communication protocol is *required to regulate the communication between distributed processes in a computer network* [34]. A micro-protocol is a communication protocol, but unlike other communication protocols, the header of a micro-protocol is placed within the hidden data transferred by a covert channel. The name *micro*-protocol comes from the constraint that network covert channels provide a very limited space for hidden data. In order to fit into such a limited space, a micro-protocol can contain only a few bits.

With micro-protocols, features can be realized which would otherwise *not be feasible* for covert channels. The list of features provided by micro-protocols comprises (but is not limited to):

1. Reliable data transfer
2. Session management for covert transactions
3. Covert overlay network addressing schemes
4. Dynamic routing for covert channel overlays
5. Upgrades of a covert channel overlay infrastructure
6. Peer discovery within a covert channel overlay
7. Switching of utilized network protocols
8. Adaptiveness to network configuration changes

In general, users of covert channels do not reveal the fact that they use a covert channel. Although covert channels are subject to research since decades, it is thus not possible to provide the reader with information about the extent of practical covert channel application or the dimension of micro-protocol usage in practice. However, various use of cases for network covert channels with micro-protocols are possible. We want to highlight the selected use of cases to show the significance of the topic:

1. *Stealthy botnet command and control channels*: If a botnet implements a network covert channel with a micro-protocol, it can signal commands and configuration messages through the micro-protocol (e.g., the command to send a spam message). The rest of the hidden data (i.e., the actual hidden payload) is interpreted according to the micro-protocol information. If the micro-protocol command is to send a spam message, the hidden payload could comprise the spam message to be sent. The micro-protocol can additionally optimize the stealthiness of the botnet's covert channel by optimizing the *covertness* of a routing path [4, 13].
2. *Journalists in networks with Internet censorship*: To transfer illicit information, a journalist can either send data through a simple covert channel or can instead use a micro-protocol-based covert channel. The micro-protocol is capable of hiding the transfer in an improved manner by splitting the payload over multiple simultaneous channels (e.g., different utilized network protocols instead of only one). At the same time, a micro-protocol provides reliability over all channels used and can additionally support dynamic overlay routing in order to allow the journalist to bypass critical censorship infrastructure on a global routing path.
3. *Military communication, secret agencies, and political opposition*: Like journalists, members of other organizations can profit from micro-protocols in covert channels. A military communication at a battlefield should not be detectable by enemy units but should provide a reliable communication. Micro-protocols realize these requirements. Spies may dynamically define optimal routing paths for stealthy communications based on micro-protocols and members of the political opposition can exchange secret information in covert chats with session management and automatic peer discovery of other oppositional members.

A problem is that the existing micro-protocols are *not optimized for a covert channel environment* because they were designed like all other network protocols. As a result, the protocols headers are *too large* and additionally are *not placed within the covert data in the optimal way* [4, 48]: If micro-protocol headers are too large, more bits are required to be manipulated in the utilized network packet. Thus, more anomalies are caused due to the manipulation which can lead to easier detection of the covert channel. If the micro-protocol header is not placed within the hidden data in the optimal way, the operation of the micro-protocol can result in anomalies (e.g., uncommon bit/flag combinations in network packets) which can also lead to the easier detection of the channel.

In this article, we contribute to the existing knowledge by:

1. Providing the first survey of micro-protocols for network covert channels,
2. categorizing all existing micro-protocols,
3. providing the first survey of existing micro-protocol engineering approaches and propose an improvement for one of these approaches,
4. proposing a multi-layer micro-protocol architecture, and
5. presenting new research goals for micro-protocols. We propose to design multi-layer micro-protocols and research that enables the inter-connectivity of currently separated covert channel overlay networks by introducing protocol translation for micro-protocols.

The remainder of this article is structured as follows. Section 2 provides a summary of existing micro-protocols, compares their features, and applies a categorization to these protocols. We discuss the drawbacks of existing micro-protocols and protocol engineering solutions for these problems in Section 3. Based on our findings, we propose novel directions for micro-protocol research in Section 4 and conclude in Section 5.

2 Existing protocols and their history

In this section, we present and discuss existing micro-protocols for network covert channels. These micro-protocols do not only comprise different header sizes but also differ in the number and type of the provided features.

2.1 Ping tunnel

The first micro-protocol presented was developed by Stødle for the tool *ping tunnel* (PT) in 2004 [41]. PT is one of the most feature-rich micro-protocols, but the price for it is the requirement for more space. The header of PT is shown in Fig. 1.

PT utilizes the ICMP "Echo Request" and "Echo Reply" payload to cover its micro-protocol and its payload. The first 4 bytes of the ICMP Echo payload contains a magic byte used to identify PT packets, which makes the tool easy to detect. However, its main purpose is to pass firewalls and not to provide a stealthy communication. The magic byte is followed by the actual micro-protocol which contains the fields shown in Fig. 1: a 4-byte destination IP address, a 4-byte destination port (actually, a port number only requires 2 bytes), a 4-byte state information used to indicate the message type as well as the connection state, a 4-byte acknowledgement number (of the last received packet), the 4-byte length of the payload following the micro-protocol header, a 2-byte sequence number,[1] and a 2-byte identifier field to handle multiple simultaneous connections.

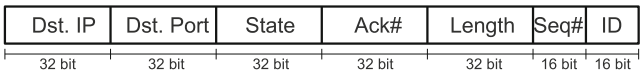

Fig. 1 The header of ping tunnel's internal micro-protocol as presented in [41]

2.2 Micro-protocol by deGraaf et al.

deGraaf et al. designed a micro-protocol in 2005 which we will denote *dG* protocol. The dG protocol prevents packet re-ordering in a port knocking-based covert channel. Therefore, sequence numbers are placed in the UDP destination port field [9]. The 16-bit destination port field is split into a data part and a sequence number. This simple micro-protocol does not prevent packet loss but was the first micro-protocol for covert channels discussed in the research community.

2.3 Micro-protocol by Ray and Mishra

Another micro-protocol by Ray and Mishra (which we will denote RM protocol) was released in 2008 [36]. The protocol is designed for getting embedded into the ICMP Echo payload like the PT protocol.

The header of RM is shown in Fig. 2 and contains a sequence number and an expected sequence number which is the acknowledgement number (2 bits each). One flag indicates that payload is attached and another flag indicates that the packet contains an acknowledgement.

The field *expected sequence number* contains the sequence number of the received packet, e.g., if the field contains the number *1*, the packet with the sequence number *1* was successfully received, but the sequence number *10* (2) is expected for the next packet.

The 2-bit sequence number (i.e., four states are possible) is sufficient for a covert channel communication to prevent the re-usage of a sequence number [36]. In comparison to PT's 16-bit sequence number, the 2-bit sequence number of Ray and Mishra's protocol is small. Since the RM protocol only sends out new packets after the latest packet got received and acknowledged, i.e., a *stop-and-wait automatic repeat request* (ARQ) protocol is used; sequence numbers cannot be used for two packets at the same time. However, waiting for the acknowledgement of a packet before sending out the next packet is a slow process. Since ARQ only requires a 1-bit sequence number, but the protocol provides a 2-bit sequence number, an improved variant of ARQ could

[1] The size difference of the acknowledgment and sequence numbers is known but was not modified so far.

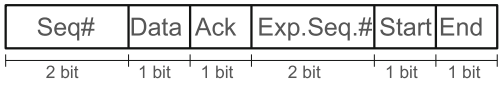

Fig. 2 The header of the protocol presented by Ray and Mishra [36]

be used where multiple packets can be sent sequentially before an acknowledgement is received. Therefore, the RM authors propose the improved versions *Go-back-n ARQ* and *selective repeat ARQ* [36]. Ray and Mishra motivate their choice with the fact that the improved algorithms can lead to more re-transmissions of packets in case of ICMP rate limiting,[2] which can raise more attention.

The last two bits are used to specify whether a covert communication starts or ends with the current packet.

2.4 Covert file transfer protocol

In 2010, Trabelsi and Jawhar published a *covert file transfer protocol* (CFTP) hidden within the IP record route option [43]. CFTP allows to transfer covert messages but additionally allows to upload, download, and list hidden files. It therefore provides a file hosting service–the CFTP server–that is accessible by the CFTP clients. CFTP additionally implements session management and reliability, but in comparison to the other protocols, it also enables authorization. For instance, a client can be allowed to upload files to the CFTP server, but the server could forbid the same client to list the files located on the server. These permissions are indicated by four flags (privilege to list files on the server, download files, upload files, and to send short messages). Besides, additional flags indicate whether a packet is a short message, a file, or a part of a file, and whether a packet is a retransmitted packet or not.

2.5 Hybrid approach using digital watermarking

Mazurczyk and Kotulski developed a control protocol based on a hybrid hiding (HyH) approach in 2006 [31]. The protocol combines a covert channel with digital watermarking. Therefore, a 6-bit control protocol header is embedded into unused bits of the IP, UDP, and RTP header. The header describes parameters of a watermark embedded into the payload of a VoIP message.

The first 4 bits of the control protocol describe the information contained in the watermark, which are analogous to the RTCP protocol (authentication or integrity parameters, inter-arrival jitter, NTP timestamp, or RTP timestamp, just to mention a few). The fifth bit signals the side of the bi-directional communication (sender or receiver). The sixth bit indicates if the packet represents a beginning parameter indicated by the first 4 bytes or whether the watermark contains continuing data of a parameter.

2.6 Smart covert channel tool

In 2012, we presented an optimized control protocol that allows the configuration of dynamic routing in covert channel overlay networks [4]. The so-called *smart covert channel tool* (SCCT) implemented a routing algorithm based on OLSR to achieve a high stealthiness for the propagation of routing information. The header of the control protocol does not comprise a standard header as the header design is highly dynamical by only comprising the components required for a particular routing message. Therefore, four routing messages (requesting peer tables, sending peer tables, transferring topology graphs, and performing routing updates) were implemented. Moreover, the protocol was designed to be integrated into various network protocols which can be utilized by SCCT's software architecture.

2.7 Terminology

In [48], we introduced an improved terminology for micro-protocols visualized in Fig. 3. We explain our terms in this section.

Each network covert storage channel uses a network protocol to embed hidden information into. We call this utilized protocol the *underlying protocol*. The area(s), in which the covert data is embedded within the underlying protocol, is called the *cover protocol*. In the cover protocol, the *micro-protocol* and the payload are placed.

For instance, if different bits of the IPv4 header (e.g., selected bits of the TTL, the bits of the IP Identifier, and the DF flag) are used for the covert channel transfer, the combined area of these utilized bits are summarily called the cover protocol, while IPv4 itself is the underlying protocol.

Coming back to the botnet scenario, the micro-protocol could comprise a botnet command (e.g., sending a spam message or installing a software upgrade for a bot) while the actual payload could comprise a spam message fragment or a fragment of a software update for a bot.

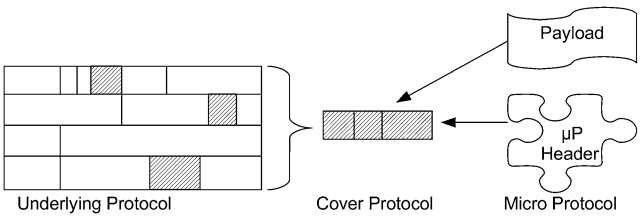

Fig. 3 Underlying, cover, and micro-protocol

[2] With ICMP rate limiting, e.g., provided by modern CISCO devices and Linux, the number of ICMP messages of the same type per time slot can be limited [11].

2.8 Categorization

After introducing the existing micro-protocols and terminology, we will now categorize these protocols. The categorization allows to determine protocol features which were not part of any previous research.

In the first step, we apply the OSI layer model to micro-protocols. The OSI-based categorization allows us to evaluate which kind of layer-associated features were (not) part of any micro-protocol research and could thus lead to new research contributions. Afterwards, we introduce a second categorization of covert channel-specific attributes in order to evaluate the need for micro-protocol feature improvements from an information hiding perspective.

2.8.1 OSI-based categorization

We analyzed the features of all mentioned protocols which we summarize in Table 1. As this table demonstrates, the mostly addressed feature of micro-protocols is a reliable data transfer—a feature of the transport layer. Also, in four of six micro-protocols, a simple session management functionality is implemented. Only PT and SCCT support features of the network layer. None of the protocols but CFTP comprises features of the application layer, and none of the protocols contains features of the presentation (except HyH), data link, or physical layer. However, since covert channels build overlay networks on top of an existing network infrastructure, the data link and physical layer are not considered to be of importance.

A direction for novel micro-protocol research is to introduce features of the presentation layer and additional features of the application layer. For example, overlay networks *can* use own addressing schemes. For a covert channel, space-efficient addresses are optimal as they allow small micro-protocol headers. Hostnames for overlay addresses will increase the usability of the overlay network, and thus, the integration of DNS features of the application layer will improve the usability.

Besides, it is feasible to introduce additional features of layers already taken into account, such as a control message protocol like ICMP for covert channels or quality of service support (*quality of covertness* for dynamic routing in covert channel overlays is already available [4]).

2.8.2 Covert channel-specific categorization

In comparison to normal network protocols (non-micro-protocols), one specific challenge in designing micro-protocols is to embed the bits of the micro-protocol header into the cover protocol without raising attention. A second challenge is to decrease the size of a micro-protocol header. Normal, space-efficient protocols like CSLIP are available as well, but micro-protocols face the problem that their header can be split over different header areas of the underlying protocol for which different encodings of the micro-protocol header bit can be necessary. Advanced micro-protocols like HyH [31] can moreover be split over different network layers [47].

For the second categorization, we therefore introduce the following properties to classify micro-protocols. In contrast

Table 1 Categorization of micro-protocol features in the context of the OSI model

OSI layer	PT	RM	dG	CFTP	HyH	SCCT
7 (Application)	–	–	–	File transfer	–	–
6 (Present)[a]	–	–	–	–	Integrity, authenticity	–
5 (Session)	Multiple parallel sessions (ID field)	Simple sessions based on transactions	–	Simple sessions, authorization	Session information /control	–
4 (Transport)	Reliability	Reliability	Reliability	Reliability	Continuity of transfer	–
3 (Network)	Addressing, proxy functionality	–	–	–	–	Routing, adressing
2 (Data link)	–	–	–	–	–	–
1 (Physical)	–	–	–	–	–	–

[a] Cryptographic features belong to the presentation layer but were not taken into account if they are not directly handled by the micro-protocols although some implementations encrypt payload

to the OSI layer categorization, these properties are specific to covert channels. We introduce a metric called *features per micro-protocol header bit* which can be seen as an efficiency value for the coding of micro-protocol header bits: the more features a micro-protocol implements per utilized bit, the more space efficient its design is and the fewer anomalies are caused per provided feature.

We evaluated the following attributes for the micro-protocols:

1. Hiding technique of the protocol header
2. Size of the micro-protocol header
3. Features per micro-protocol header bit
4. Dependence on TCP/IP attributes
5. Addressing scheme if included
6. Upgradability
7. Handling of incoming packets not belonging to the covert channel but using the same underlying protocol

We discuss all these properties separately and summarize our findings in Table 2.

1. *Optimized hiding* To achieve stealthy communication, hidden information should be placed in the underlying protocol in a way to generate as few anomalies as possible to avoid raising attention. However, an optimized placement of header bits within the underlying protocol was introduced for none of the mentioned protocols.
2. *Size of the micro-protocol header* The RM protocol is more space efficient than PT as it comprises only 8 bits instead of 192 bits. The dG protocol requires 16 bits (the size of the UDP destination port field). CFTP also needs 16 bits (including 4 unused bits). As the HyH header is separated from the parameters (linked to varying sizes), we did not include it in the size evaluation. However, the first 4 bits of the HyH header allow to integrate only required parameters into the watermark in order to save space for currently unrequired parameters. A similar approach to only include required header components is present in SCCT which also leads to a dynamic header size.
3. *Features per micro-protocol header bit* The absolute number of features provided within the 8 bits of the RM protocol is smaller than for PT, but the number of features provided per bit is higher for RM as for PT. However, if large address fields are transferred within a protocol header (such as 32 bit IPv4 addresses used for PT), the number of features per header bit decreases. Within the 16 bits of the dG protocol, only one feature is realized while CFTP supports six features within the same header size. The number of features can be counted in different ways since sub-features can be considered as a feature as well. For instance, in the RM protocol, the start flag and the stop flag are represented by 2 bits and indicate whether a transaction starts or begins. Both features can be seen as a single feature and could be placed in one bit: either the start flag is set (transmission starts or is taking place) or it is not set (transmission ends). Therefore, we only provide approximate numbers for the number of features in Table 2. The feature to transfer a general hidden message was not counted as a micro-protocol feature since payload transfer is a fundamental feature in covert channels that was already realized in the past without micro-protocols. Unused bits which are part of a micro-protocol header definition were counted as header bits without a feature.
4. *TCP/IP dependence* Covert channels can be embedded into various underlying protocols, including protocols of non-TCP/IP stacks, such as BACnet [3]. Micro-protocols should be designed in a way that they are

Table 2 Categorization of micro-protocol features in the context of covert channel-specific attributes

Attribute	PT	RM	dG	CFTP	HyH	SCCT
Hiding optimized	–	–	–	–	–	–
Size optimized	–	Comparably small header	–	–	Parameter-based	Yes
Features per header bit	Few (< 0.1)	High (0.75)	Few (< 0.1)	Average (0.375)	Dyn. size	Dyn. size
TCP/IP dependence	Yes (IP addr./ protocol size)	–	–	–	–	Yes (IP addr.)
Addressing scheme	IPv4	None	None	None	None	IPv4
Upgradability	–	–	–	–	–	Proposed
Handling of non-covert input	Magic byte, reliability	Reliability	Reliability	Reliability	Authenticity	-

suitable for various underlying protocols in order to enable switches of the underlying protocol [48].

While the PT and RM protocols were developed for getting placed within the ICMP Echo payload and the dG protocol was designed for getting embedded within the UDP header, CFTP was designed to be hidden in an IP option's header and HyH can be embedded into IP, UDP, and RTP. SCCT supports various protocols by design, but generally, all protocols can be adapted to arbitrary protocols. However, it is easier to place the 8-bit header of the RM protocol into another utilized protocol than it is to place the 192-bit header of PT in another underlying protocol since many protocols only provide a few bits. Another possibility to integrate the PT protocol into another underlying protocol would be to split the PT header over multiple protocol layers—an approach proposed in [47].

5. *Addressing scheme* A characteristic of PT and SCCT are the IPv4 addressing schemes which allow no non-TCP/IP communication. The RM, dG, CFTP, and HyH protocols do not contain any internal addressing schemes and can thus be embedded in any other protocol as well. A suitable addressing scheme for TCP/IP independence would be to introduce an own addressing scheme into a covert channel overlay network.
6. *Upgradability* To allow newer micro-protocol versions, it is an important feature to use version numbering in micro-protocol headers. This enables to create backward compatibility as well as previously unsupported features [46]. None of the micro-protocols comprises such a feature.
7. *Handling of incoming packets which do not belong to the covert channel* Received data might be inadvertently interpreted as belonging to a covert communication. The PT header therefore introduces a *magic byte*: only if the magic byte equals a given value, the packet is treated as a covert channel packet. HyH comprises support to evaluate the authenticity of packets. The other protocols do not comprise such features. However, due to the provided reliable data transfer, sequence numbers which do not match an existing connection could be logically dropped by all other protocols (except SCCT) as long as the implementations foresee such a feature.

2.9 Micro-protocols in the network environment learning phase

Yarochkin et al. introduced a so-called *network environment learning* phase (NEL phase) [52]. In the NEL phase, two covert channel peers (a) try to determine the presence of each other and (b) try to determine the underlying protocols they can use to communicate with the other peers. Non-routed or blocked underlying protocols are discovered and not used for the communication.

The discovery of usable underlying protocols can be achieved in an organized manner as micro-protocols allow to announce test packets in advance [45]: if a test packet does not reach the receiver after a waiting time t, the receiver knows that the underlying protocol cannot be used. The receiver sends the result of the packet transfer to the sender using a micro-protocol message.

A two-army problem can be relieved by using micro-protocols: if two peers send test packets to each other to discover usable underlying protocols and if a traffic normalizer drops packets on the path between both systems, the peers cannot be sure whether the message sent did not reach the other peer or whether the acknowledgment message (that indicates the successfully received packet) was dropped by the normalizer on its way back to the sender [45]. Based on a third participant and micro-protocols, meta information to announce test packets can be exchanged between two peers and they can determine whether a packet reached the peer or not.

2.10 Control protocols for covert timing channels

Although we focus on covert storage channels as only these channels can comprise headers of micro-protocols, control mechanisms are available for timing channels as well. Therefore, timing channels use various encoding mechanisms (e.g., sending parity bits [44]).[3]

Luo et al. showed that multiple TCP flows can be used simultaneously to encode a hidden message via timing channels while additionally providing reliability [29]. A similar approach called *TCPScript* that encodes messages into TCP data bursts was presented by the same authors in [30]. TCPScript is an approach that relies on the sliding-window algorithm, the sequence number, and the capability to acknowledge messages and can be integrated into other protocols (e.g., SCTP) as well [30].

CoCo is a reliable timing channel that modulates hidden information via inter-arrival times of packets and is adjustable to configure undetectability, rate, and robustness of the channel [20]. *ProtoLeaks* is another timing channel using TCP-based protocols to provide reliability and therefore utilizes permutations of multiple distinguishable, sequential network objects (e.g., HTTP requests or FTP commands) [42].

[3] In [44], we speak of a *storage* channel, but as the channel utilizes the order of network objects, it can also be categorized as a *timing* channel.

3 Overcoming existing drawbacks: micro-protocol engineering

Basically, existing protocol engineering means can be applied to micro-protocols as well. However, to address micro-protocol-specific design problems and drawbacks of existing micro-protocols, two engineering approaches were introduced which can be combined with the existing means.

3.1 Drawbacks of existing control protocols

Being linked to the introduced features, micro-protocols can also affect the detectability of a network covert channel in a positive way (e.g., by enabling the adaption of the covert traffic to the patterns of legitimate traffic and by enabling protocol switching) or in a negative way. In this section, we cover the negative effects: First of all, embedding a micro-protocol header requires space of the cover protocol. Each needed header bit leaves less space for the actual payload of the channel that must be placed in the cover protocol as well. Thus, the more bits micro-protocol headers require, the more bits must be manipulated and the more packets must be sent for a given payload transfer and the chance of a detection increases. Therefore, we can conclude that a micro-protocol's size must be as small as possible.

On the other hand, a micro-protocol can negatively affect the detectability of the covert channel by causing anomalies in the underlying protocol. For instance, if a micro-protocol is located in selected bits of the TCP flags, it may cause flag combinations which are not standard conform. Such behavior can cause an alert of the intrusion detection systems. Thus, the behavior of a micro-protocol must conform to the (standard of the) underlying protocol.

In order to overcome the outlined problems, we developed solutions: *status updates* [4] optimize the size of the micro-protocol header and the *conformance approach* [48] ensures that the micro-protocol conforms to the underlying protocol.

3.2 Status updates

A first technique to achieve a space-efficient control protocol header was the HyH protocol [31] in which the first four header bits indicate the presence of parameters in the watermark. In this section, we discuss a more dynamic approach: status updates reduce the size of micro-protocols by only transferring header bits if these are required [4]. Hence, this approach optimizes the *features per header bit* metric from Section 2.8.2.

The idea of status updates is based on compressed SLIP (CSLIP) [22] and IPv6 header extensions [8]. CSLIP transfers a bitmask which indicates the presence of header parameters [22] (this approach was applied in the HyH protocol as well [31]). IPv6 extension headers, on the other hand, allow the dynamic combination of extension headers as well as multiple occurrences of extension headers within a single packet.

Status updates combine both ideas and add the idea of states: In the context of status updates, a connection between two covert peers can be linked to many *states*, such as the source address of a connection, the destination address of a connection, or a session's state (active or inactive). These states are only transferred between the peers if they change.

For instance, if a covert sender wants to transfer payload to a covert destination over a proxy, the proxy only needs to receive the status for the destination address once and not with every packet of a connection. If IPv4 addresses are used, the sender has to transfer 32 bits less per packet.

To apply status updates to an existing micro-protocol, its header must comprise parts which are not always needed, i.e., which are only needed to indicate an update of a connection's state. All header components that are not always required are removed from the *default* micro-protocol header and are only transferred if required.

To transfer a header part *not* belonging to the default header, preamble bits must be sent. The value of the preamble bits identifies the type of the header part. For each non-default header part, a unique identifier must be defined and it is possible to combine multiple non-default headers in a single packet.

Using status updates, the size of the RM micro-protocol could be made more efficient: A status update-based version of this protocol transferred fewer bits in comparison to the original header if at least five packets are transferred per transaction [4]. To this end, the bits which indicate the start and end of a transaction are removed from the default header and are only transferred if a transaction begins or ends.

Besides, status updates were also used to implement the efficient transfer of routing updates for covert channel overlays [4].

3.3 Conformance approach

A first approach to ensure that a micro-protocol conforms to the underlying protocol was presented in [48]. To this end, an area of the underlying protocol must be defined to be used for the placement of the micro-protocol–as mentioned–this area is called the cover protocol.

Next, the occurrence rates of micro-protocol bits and the occurrence rates of bits in the cover protocol are evaluated, e.g., based on traffic recordings. Afterwards, micro-protocol bits are mapped to bits of the cover protocol such that occurrence rates match as closely as possible. Thus, the bit occurrence rates do not differ much between an underlying protocol *without* a micro-protocol and an underlying protocol *with* a micro-protocol.

Assume for example that micro-protocol and cover protocol comprise bits ab and cd, respectively, where $Pr(a = 1) = 0.7$, $Pr(b = 1) = 0.5$, $Pr(c = 1) = 0.6$, and $Pr(d = 1) = 0.3$. Then, a should be mapped to c and b should be mapped to d to match the probabilities in the best possible way.

Afterwards, both the micro-protocol as well as the cover protocol are modeled using a formal grammar. Each formal grammar must produce all possible bit combinations for the header of its protocol, and both grammars are required to comprise the same terminal symbols to represent the same bits in the underlying protocol. Therefore, the previously configured bit mapping is used. The sentences produced by both grammars need to contain their respective terminal symbols in exactly the same order, and each terminal symbol must only occur once since a header bit cannot be set twice in the same packet.

In the final step, it is tested whether the micro-protocol's operation can lead to bit patterns in the cover protocol which are not foreseen by the *underlying* protocol. For instance, an unusual TCP flag combination could be created if the micro-protocol utilizes TCP flags. This could raise attention at a typical network IDS. Therefore, a language inclusion test is applied: If the language produced by the micro-protocol's grammar can generate sentences which cannot be created by the language produced by the cover protocol's grammar, the micro-protocol does *not* conform to the cover protocol. This means that the micro-protocol would generate bit combinations in the underlying protocol which should not occur. To apply an automatic language inclusion test, the grammar of the micro-protocol must be regular or context free and the grammar of the cover protocol must be regular. Otherwise, the language inclusion is not decidable.

Typically, the grammars are constructed along the meanings of the packets or packet headers [18, 19]. As an example, assume that the two micro-protocol bits comprise a packet type and a flag according to the rules:

```
M -> Type Flag
Type -> 0|1
Flag -> 0|1
```

The underlying protocol might use the two bits in two modes distinguished as follows:

```
S -> Mode1 | Mode2
Mode1 -> 0 Flag
Flag -> 0|1
Mode2 -> 11
```

Then, the language of the micro-protocol grammar is {00, 01, 10, 11} while the language of the underlying protocol grammar is {00, 01, 11}. As 10 is used in the micro-protocol but not in the underlying protocol, the micro-protocol is not conforming to the underlying protocol and should be revised.

The conformance approach is dynamic as re-designing paths are defined to optimize selected steps of the process. Additionally, the approach can support connection-oriented protocols which have to take bit values of previously sent or received packets into account.

3.3.1 Improvement of the existing approach

In the early phase of the conformance approach, occurrence rates of bits in micro- and cover protocol are evaluated based on traffic recordings or are estimated. Afterwards, bits of the micro-protocol are mapped to bits of the cover protocol in a way that a micro-protocol bit is as likely to occur as a cover protocol bit.

We discovered that different protocol states should be considered in the evaluation since bit occurrence rates depend on protocol states. For instance, some flags of the TCP header are more likely to occur in the connection establishment or connection termination phase than in the phase in which the actual payload transfer is taking place. If the micro-protocol utilizes TCP flags, a state-independent mapping could thus lead to non-optimal results.

For a better mapping, each state of the cover protocol should be evaluated separately with respect to bit occurrence rates. Afterwards, the micro-protocol is mapped to the cover protocol in ways optimized for each state. The drawback of this approach is that for n protocol states, n mappings instead of one must be computed which also results in n language inclusion tests.

Since the covert channel sender always knows the state of the cover protocol when a message is generated, it automatically has knowledge about the bit mapping that must be applied. On the receiver's side, the cover protocol state and with it the correct bit mapping can always be determined, i.e., the micro-protocol bits can always be interpreted in the correct way.

3.3.2 Adaptability to similar use cases

In [21], the authors propose to create steganographic connections in a way that implementation errors for network protocols are covered by the steganographic data transfer as well. This complicates the distinction between an original but erroneous implementation and a steganographic implementation of a protocol. However, while the authors mention the requirement for error-conform protocol implementations, they do not provide a solution. Our conformance approach provides a first means to model such error-conform implementations since implementation-known errors can be modeled using the formal grammar. Therefore, the grammar of the steganographic protocol implementation must produce a language subset of the original implementation's grammar.

Lucena et al. earlier presented work on syntax and semantics-preserving steganography for the application layer [28]. Their approach must be considered the first that aimed on ensuring protocol conformity of steganographic protocols to underlying protocols. Besides not taking other layers but the application layer into account, the approach does also differ from our approach as it does not target micro-protocols, but general network steganography and solely focuses on statistical similarity. The authors selected randomized fields in the SSH2 protocol to place their hidden data into. Our conformity approach enables the conformity of complex header structures and, although it is designed for micro-protocols, can be seen as an extension to the work by Lucena et al. as it considers non-randomized header fields.

4 Future challenges for control protocols

In this section, we will discuss goals for further micro-protocol research in order to provide a fundament for upcoming papers in this emerging field.

4.1 Integration of features from other OSI layers

As already pointed out in Section 2.8.1, we see the integration of additional features of the Internet and transport layers as well as integration of features of the presentation and application layers as valuable goals.

In the context of our covert channel-specific categorization from Table 2, we see the development of an own addressing scheme as a fundamental goal. If IP addresses are used in micro-protocols, the micro-protocol size and, thus, the features per header bit value will impair. An own addressing scheme with tiny addresses (e.g., 5-bit addresses for a 32-host network) would be suitable. If larger address ranges, or even subnets, will be introduced, it would also make sense to integrate DNS-like features into network covert channels.

Based on the given use case, it would also be possible to introduce application layer functionality specific to the use case. If, for instance, a video shall be transferred (or even streamed) by a journalist, a streaming functionality would be required,[4] while a general file transfer can be achieved with the existing micro-protocols.

If presentation layer functionality will be included, covert channel peers could agree on encodings or on a compression algorithm for payload (e.g., for the mentioned video streaming).

Other desirable functionalities comprise IGMP-like group communication or DHCP-like peer auto-configuration. Therefore, peers joining the network could be assigned with temporary covert overlay addresses. Such a feature would help to reduce the number of required address bits in micro-protocol headers as temporarily used non-static addresses can be assigned to other peers from time to time.

4.2 Multi-layer micro-protocols

Another promising approach is the creation of an OSI-like layered model for network covert channels. As explained, physical and data link layer functionality are not necessary for covert overlay networks. However, functionality of the network, transport, session, presentation, and application layers must be considered useful for covert channels.

A layered micro-protocol design would allow to include and exclude features depending on the use case. If, for instance, a reliable data transfer is required, a transport layer component providing reliability could be included, otherwise, the space could be saved. This approach is similar to the *status update* approach but would allow an even more dynamic and more feature rich design as status updates only focus on a single protocol and a multi-layer model would lead to a set of protocols for a whole protocol stack.

4.3 Protocol translation for micro-protocols

The discussed micro-protocols raise the question whether an inter-operability of such micro-protocols is feasible. Inter-operability for micro-protocols means that a covert channel peer using micro-protocol P_a can communicate with another peer using micro-protocol P_b with $P_a \neq P_b$.

If inter-operability is feasible, currently separated covert channel peers and even whole overlay networks could be linked together to form larger, more robust distributed systems (e.g., it would be possible to create more alternative routing paths between peers).

The following aspects must be taken into account by further research to enable protocol translation for micro-protocols:

Unsupported features Obviously, two protocols P_a and P_b do not necessarily share the same feature set, and even if same or similar features are implemented, they can be represented in a different or even incompatible way. For instance, the RM protocol comprises a 2-bit sequence number and a 2-bit expected sequence number, while the sequence number field of PT comprises 16 bits and the acknowledgment

[4] While the HyH protocol generally supports VoIP streaming, it does not hide the streaming content itself.

field comprises even 32 bits. A protocol translation can thus not be considered trivial in any case.

Besides, a feature contained in protocol P_a that is not available in protocol P_b cannot be handled in P_b (e.g., the permission flags are only available in CFTP but not within other protocols). Even if two networks using P_a are separated over a network that uses P_b, the transfer over P_b cannot preserve the features which are not supported by the protocol as all those features must be removed as long as P_a is not entirely encapsulated (tunneled) through P_b.

Reliability All micro-protocols implement reliability but use fields of a different size. A protocol translation system could, for instance, cache sequence and acknowledge numbers on the PT side of the connection and apply ARQ on the RM protocol side.

Addressing scheme A completely different problem would be to realize addressing between two PT/SCCT systems that are separated by a network that uses the RM, HyH, or dG protocol: The latter protocols do not comprise any address values since they use the address values in the IPv4 header. One technique to overcome this problem is to tunnel the PT/SCCT protocol as payload through the connection established by the RM/HyH/dG protocol. In any case, the protocol gateway must include the IP address of the peer in the RM/HyH/dG payload if the traffic is forwarded to another gateway.

Magic byte PT's micro-protocol comprises a magic byte that allows to identify whether an ICMP Echo packet belongs to the covert channel or not and the HyH protocol can verify the authenticity of a packet in order to detect whether a packet belongs to a connection or not. In comparison to the magic byte, the authenticity verification can be considered more robust as it is harder to forge. The RM, SCCT, and dG protocols do not support such a feature, and thus, packets that do not belong to the covert channel could be treated as covert channel packets. Such packets might also be forwarded by a protocol translation system to a PT/HyH peer and could possibly corrupt a connection. However, this scenario can be considered unlikely since micro-protocol fields of accidentally sent ICMP Echo packets must comprise suitable values (sequence numbers should fit into the connection). A plausibility check could be used as a solution to this problem (e.g., packets containing sequence numbers out of the scope of the current connection could be ignored).

We consider protocol translation for currently incompatible micro-protocols as a valuable research goal to interconnect currently separated overlay networks.

4.4 Detection and reverse engineering of micro-protocols

Protocol reverse engineering aims to understand undocumented file formats and network protocols and was applied in the past to understand protocols like SMB and ICQ [6]. Recently, protocol reverse engineering became popular in order to understand C&C protocols of botnets.

Micro-protocols were, in comparison to C&C protocols, not subject of reverse engineering in the past. While early botnets used IRC and HTTP as underlying protocols, their architecture became more advanced as P2P architectures, protocol obfuscation, and encryption features were implemented into C&C communications [6, 25].

To this end, researchers, on one hand, scrutinize botnet binaries using static and dynamic analysis to reconstruct the message format and behavior of C&C protocols as well as to reverse engineer encryption and decryption methods [6, 25, 51]. Static and dynamic analysis is linked to the drawback that an analyst needs to know about the existence of a hidden communication as well as she needs access to a binary of a bot. The goal of covert channels is to prevent that a third party knows about the existence of a hidden communication, i.e., static and dynamic analysis are not suitable approaches to counter covert channels—with or without micro-protocols. However, after a covert channel is detected, static and dynamic analysis can help to reverse engineer the hiding technique of a covert channel and with it the transferred payload and the micro-protocol design.

On the other hand, protocol reverse engineering means based on network input are available. Antunes et al. reconstruct a protocol state machine and protocol language based on network traces [2], but their approach is not designed to handle micro-protocols as they are split over multiple layers and multiple fields within headers, can utilize various hiding approaches, and can adapt to network traffic in order to remain hidden (e.g., by adapting packet lengths to legitimate network traffic [49]). These covert channel-specific characteristics are valid for other approaches as well: Zhao et al. classify network traffic in order to detect malware activity in real time, which is similar to classification-based covert channel detection approaches [53] but does also not focus on micro-protocols but on the covert channel as a whole. Dietrich et al. determine the presence of botnet C&C traffic by observing the message length, encoding differences and carrier protocols of network communications [10] but require a priori information about the characteristics of particular C&C protocols. Gu et al. cluster similar flows and malicious traffic in order to detect botnets but refer to covert channels as a potential countermeasure to render their approach ineffective [15]. A major challenge for the adaption of these techniques to micro-protocol-based covert channel detection is the high number of potential

hiding techniques which can be applied to implement data hiding in networks.

In comparison to static and dynamic analysis, means based on network input can be implemented on network hops instead of end-user systems and do not require binaries of the bot software.

Although the major goal of a covert channel is to cover the existence of a hidden communication, a future challenge is to verify the adaptability of existing protocol reverse engineering means to micro-protocols and to develop specialized methods capable to (a) deal with cover protocols spread over multiple header fields in different network layers as well as (b) capable to deal with micro-protocols comprising a variable header structure (status update) and (c) capable to deal with micro-protocols which were optimized to provide conformity (conformance approach). Like existing methods (cf. [6]), micro-protocol reverse engineering needs to determine protocol elements (e.g., message format including fixed-length and variable-length fields, length fields, delimiters, protocol keywords, and dynamic fields) and to re-construct the state machine of a micro-protocol. The availability of such an approach would enable adversaries to interact with micro-protocol-based covert channels after detection.

Active adversaries (or *active wardens*) could aim to compromise the covert channel overlay (e.g., by requesting peer tables of the overlay network or by inserting faked routing information in order to redirect all covert channel traffic over their own systems [4]). Moreover, passive adversaries (or *passive wardens*) could aim to determine the involvement of third parties into a covert communication.

5 Conclusion

In this survey, we explained the importance of covert channel-internal control protocols—so-called micro-protocols. Micro-protocols enable feature-rich and stealthy communications for future malware communications on one hand and facilitate secure communications for journalists, governmental organizations, or members of the political opposition in networks with censorship on the other hand. Micro-protocols can be used to introduce features such as reliable data transfer, session management, dynamic routing, peer discovery, protocol switching, and adaptiveness to networkvert channels.

We compared and categorized the existing research protocols to highlight potential paths for further research. A new metric called *features per micro-protocol header bit* was introduced to evaluate the space efficiency of a micro-protocol. Our results demonstrate that most protocols are not space-efficient and do not provide any features of higher OSI layers, protocol upgradability, an optimized placement of the micro-protocol in the hidden data, or an addressing scheme optimized for covert communication.

In the context of existing protocol engineering methods, we proposed an improvement to one of the approaches in order to additionally optimize the stealthiness of micro-protocols.

Finally, we underlined directions for future research in this emerging field: We see valuable goals in the integration of features of the OSI presentation and application layer as well as in adding additional features of the network and transport layer to micro-protocols. We also motivated multi-layer designs for micro-protocols and protocol translation to achieve inter-connectable covert channel overlay networks. Furthermore, we illustrated challenges and chances for the adaption of protocol reverse engineering means from the area of botnet research to micro-protocols.

References

1. Ahsan K, Kundur D (2002) Practical data hiding in TCP/IP. In: Proceedings of workshop on multimedia security at ACM multimedia '02. French Riviera
2. Antunes J, Neves N, Verissimo P (2011) Reverse engineering of protocols from network traces. In: Proceedings of 18th working conference on reverse engineering (WCRE), pp 169–178
3. ASHRAE SSPC 135 (2013) Building automation and control networks (BACnet) (website). http://www.bacnet.org/
4. Backs P, Wendzel S, Keller J (2012) Dynamic routing in covert channel overlays based on control protocols. In: Proceedings of International workshop on information security, theory and practice (ISTP-2012), IEEE pp. 32–39.
5. Berk V, Giani A, Cybenko G (2005) Detection of covert channel encoding in network packet delays. Tech. rep., Department of Computer Science—Dartmouth College
6. Caballero J, Song D (2013) Automatic protocol reverse-engineering: message format extraction and field semantics inference. Comput Netw 57(2):451–474
7. Das A (2012) Steganography: secret data hiding in multimedia, signal conditioning, signals and communication technology, vol 180. Springer, pp 275–295
8. Deering S, Hinden R (1998) Internet protocol, version 6 (IPv6) specification (RFC 2460). http://www.ietf.org/rfc/rfc2460.txt
9. deGraaf R, Aycock J, Jacobson MJ (2005) Improved port knocking with strong authentication. In: Proceedings of 21st annual computer security applications conference, ACSAC '05. IEEE Computer Society, pp 451–462
10. Dietrich CJ, Rossow C, Pohlmann N (2013) CoCoSpot: clustering and recognizing botnet command and control channels using traffic analysis. Comput Netw 57(2):475–486
11. Fall KR, Stevens WR (2011) TCP/IP illustrated: the protocols, vol 1, 2nd edn. Addison-Wesley Professional Computing Series. Addison-Wesley

12. Fisk G, Fisk M, Papadopoulos C, Neil J (2003) Eliminating steganography in Internet traffic with active wardens. In: Revised papers from the 5th international workshop on information hiding, IH '02. Springer, London, pp 18–35
13. Giani A, Berk VH, Cybenko GV (2006) Data exfiltration and covert channels. In: Proceedings of SPIE 6201, sensors, and command, control, communications, and intelligence (c3i) technologies for homeland security and homeland defense V, vol 6201. SPIE, pp 620,103–620,103–11
14. Girling CG (1987) Covert channels in LAN's. IEEE Trans Softw Eng 13:292–296
15. Gu G, Perdisci R et al (2008) BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: van Oorschot PC (ed) USENIX security symposium. USENIX Association, pp 139–154
16. Handel TG, Sandford MT II (1996) Hiding data in the OSI network model. In: Proceedings of 1st international workshop on information hiding. Springer, London, pp 23–38
17. Handley M, Paxson V, Kreibich C (2001) Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics. In: 10th USENIX Security symposium, vol 10, pp 115–131
18. Harangozó J (1977) An approach to describing a data link level protocol with a formal language. In: Proceedings of 5th Data communication symposium, pp 4–37–4–49
19. Harangozó J (1978) Protocol definition with formal grammars. In: Proceedings of Computer network protocols symposium, pp F6–1–F6–10
20. Houmansadr A, Borisov N (2011) CoCo: coding-based covert timing channels for network flows. In: Proceedings of 13th international conference on information hiding, IH'11. Springer, pp 314–328
21. Houmansadr A, Brubaker C, Shmatikov V (2013) The parrot is dead: observing unobservable network communications. In: Proceedings of 34th IEEE symposium on security and privacy. Oakland, (in press)
22. Jacobson V (1990) Compressing TCP/IP headers for low-speed serial links (RFC 1144). http://www.rfc-editor.org/rfc/rfc1144.txt
23. Jankowski B, Mazurczyk W, Szczypiorski K (2013) PadSteg: introducing inter-protocol steganography. Telecommun Syst Model Anal Des Manag 52(2):1101–1111
24. Lampson BW (1973) A note on the confinement problem. Commun ACM 16(10):613–615
25. Leder F, Martini P (2009) NGBPA: next generation botnet protocol analysis. In: Emerging challenges for security, privacy and trust. Springer, pp 307–317
26. Lewandowski G, Lucena NCS (2007) Analyzing network-aware active wardens in IPv6. In: Camenisch J, Collberg C, Johnson N, Sallee P (eds) Information hiding, LNCS, vol 4437. Springer, Berlin, pp 58–77
27. Lucena N, Lewandowski G, Chapin S (2006) Covert channels in IPv6. In: Danezis G, Martin D (eds) Privacy enhancing technologies, LNCS, vol 3856. Springer, Berlin, Heidelberg, pp 147–166
28. Lucena NB, Pease J, Yadollahpour P, Chapin SJ (2005) Syntax and semantics-preserving application-layer protocol steganography. In: Fridrich J (ed) Information hiding, LNCS, vol 3200. Springer, Berlin, pp 164–179
29. Luo X, Chan E, Chang R (2007) Cloak: a ten-fold way for reliable covert communications. In: Computer security - ESORICS 2007, LNCS, vol 4734. Springer, pp 283–298
30. Luo X, Chan E, Chang R (2008) TCP covert timing channels: design and detection. In: Proceedings of international conference on dependable systems and networks (DSN 2008), pp 420–429
31. Mazurczyk W, Kotulski Z (2006) New security and control protocol for VoIP based on steganography and digital watermarking. Annales UMCS, Informatica, AI 5:417–426
32. Millen J (1999) 20 years of covert channel modeling and analysis. In: Proceedings of 1999 IEEE symposium on security and privacy, pp 113–114
33. Murdoch SJ (2007) Covert channel vulnerabilities in anonymity systems. Ph.D. thesis, University of Cambridge (Computer Laboratory
34. Nounou N, Yemini Y (1985) Development tools for communication protocols. Tech. rep., Computer Science Department, Columbia University, New York
35. Nussbaum L, Richard O (2009) On robust covert channels inside DNS. In: 24th IFIP international security conference, IFIP advances in information and communication technology, vol 297, pp 51–62
36. Ray B, Mishra S (2008) A protocol for building secure and reliable covert channel. In: Proceedings of 6th annual conference on privacy, security and trust (PST 2008). IEEE, pp 246–253
37. Rowland CH (1997) Covert channels in the TCP/IP protocol suite. First Monday 2(5). http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/528/449
38. Rutkowska J (2004) The implementation of passive covert channels in the linux kernel. Ftp://ftp.pastoutafait.org/pdf/passive-covert-channels-linux.pdf
39. Shankar U (2002) Active mapping: resisting NIDS evasion without altering traffic. Tech. Rep. UCB//CSD-2-03-1246, Computer Science Division (EECS) (University of California Berkeley)
40. Singh A, Nordström O et al (2006) Stateless model for the prevention of malicious communication channels. Int J Comput Appl 28:285–297
41. Stødle D (2009) Ping tunnel—for those times when everything else is blocked. http://www.cs.uit.no/~daniels/PingTunnel/
42. Swinnen A, Strackx R, Philippaerts P, Piessens F (2012) ProtoLeaks: a reliable and protocol-independent network covert channel. In: Information systems security, LNCS, vol 7671. Springer, pp 119–133
43. Trabelsi Z, Jawhar I (2010) Covert file transfer protocol based on the IP record route option. J Inf Assur Secur (JIAS) 5(1)
44. Wendzel S (2008) Protocol channels as a new design alternative of covert channels. CoRR abs/0809.1949
45. Wendzel S (2012) The problem of traffic normalization within a covert channel's network environment learning phase. In: Proceedings, GI Sicherheit 2012, LNI, vol 195, pp 149–161
46. Wendzel S (2013) Novel approaches for network covert storage channels. Ph.D. thesis, University of Hagen
47. Wendzel S, Keller J (2011) Low-attention forwarding for mobile network covert channels. In: Proceedings 12th conference on communications and multimedia security, LNCS, vol 7025. Springer, pp 122–133
48. Wendzel S, Keller J (2012) Systematic engineering of control protocols for covert channels. In: Proceedings 13th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS 2012), LNCS, vol 7394. Springer, pp 131–144
49. Winter P, Pulls T, Fuss J (2013) ScrambleSuit: a polymorphic network protocol to circumvent censorship. In: Proceedings of workshop on privacy in the electronic society. ACM
50. Wolf M (1989) Covert channels in LAN protocols. In: Berson T, Beth T (eds) Local area network security, LNCS, vol 396. Springer, Berlin, pp 89–101

51. Wondracek G, Comparetti PM, Kruegel C, Kirda E (2008) Automatic network protocol analysis. In: Proceedings of the 15th annual network and distributed system security symposium (NDSS'08)
52. Yarochkin FV, Dai SY et al (2008) Towards adaptive covert communication system. In: Proceedings of 14th IEEE pacific rim international symposium on dependable computing (PRDC 2008). IEEE Computer Society, pp 153–159
53. Zander S (2010) Performance of selected noisy covert channels and their countermeasures in IP networks. Ph.D. thesis, Centre for Advanced Internet Architectures, Swinburne University of Technology
54. Zander S, Armitage G, Branch P (2007) Covert channels and countermeasures in computer network protocols. IEEE Comm Mag 45(12):136–142