The Problem of Traffic Normalization Within a Covert Channel's Network Environment Learning Phase

Steffen Wendzel
Faculty of Mathematics and Computer Science,
University of Hagen, 58084 Hagen, Germany

Abstract: Network covert channels can build cooperating environments within overlay networks. Peers in such an overlay network can initiate connections with new peers and can build up new paths within the overlay network. To communicate with new peers, it is required to determine protocols which can be used between the peers – this process is called the "Network Environment Learning Phase" (NEL Phase). We show that the NEL phase itself as well as two similar approaches are affected by traffic normalization, which leads to a two-army problem. Solutions to overcome this not completely solvable problem are presented and analyzed. A proof of concept code was implemented to verify the approach.

Keywords: network covert storage channel, traffic normalization, active warden

1 Introduction

A covert channel is a communication channel that is *not intended for information transfer at all* [Lam73]. Using network covert channels it is possible to send information in a way prohibited by a security policy [And08, WAM09]. Network covert channels can be used by both, attackers (e.g. for hidden botnet communication [GP+08, LGC08]) as well as normal users (e.g. journalists, if they want to keep a low profile when transferring illicit information [ZAB07]). Network *storage* channels utilize storage attributes of network protocols while network *timing* channels modify the timing behavior of network data [ZAB07]. We focus on covert storage channels since they (in comparison to covert timing channels) provide enough space to place a covert channel-internal protocol into the hidden data.

Ways to implement covert channels in TCP/IP network packets and their timings were described by different authors (e.g. [Gir87, Wol89, HS96, Row97, CBS04, Rut04, LLC06, JMS10]). However, such channels occur outside of TCP/IP networks as well, such as in smart grids [LG10], electronic identity cards [Sch10] and business processes [WAM09]. A number of means to protect systems against covert channels were also developed (e.g. [Kem83, PK91, Hu91, Fad96, FFPN03, CBS04, BGC05, KMC05]).

Covert channels can contain internal communication protocols [RM08, Stø09], so called *micro protocols* [WK11]. These protocols are located within the hidden data, which is used for both: the micro protocols as well as payload. Micro protocols are used to extend

the capabilities of covert channels by implementing features such as reliability and runtime protocol switching.

To enable two communicators to communicate using a covert channel, they need to find a common cover protocol. A cover protocol is the network protocol (the bits used within a network protocol) which is used to place hidden data into. The discovery of information about network protocols to use can either be done using a micro protocol as presented in [WK11] or by passive monitoring of traffic [YD+08].

A traffic normalizer is a network gateway with the capaibility to affect the forwarded traffic in a way that prevents malicious communication (e.g. malformed traffic as well as any kind of network-based attacks such as botnet traffic or exploit traffic). Traffic normalizers are also known as *packet scrubbers* [BP09]. Usually, a traffic normalizer is part of a firewall system [MWJH00] and can decide to drop or forward network traffic. A popular example of such a combination of a firewall with a traffic normalizer is the *pf* firewall of the OpenBSD operating system [Ope11]. A similar implementation for the FreeBSD kernel focuses on the implementation of transport and application layer scrubbing [MWJH00]. Additionally to a plain firewall system, a traffic normalizer is able to *modify* forwarded traffic [MWJH00]. Therefore, the normalizer can apply rules such as clearing bits of a network protocol's header or setting a header flag that was not set by the packet's sender [HPK01]. Because of a normalizer's applied modifications, their implementation in a network can result in side-effects, e.g. blocked ICMP types result in the unavailability of the related network features of these ICMP types [SN+06].

Traffic normalizers are also referred to as a special type of an active warden. While passive wardens in networks monitor and report events (e.g. for intrusion detection), active wardens are capable of modifying network traffic [SN+06] to prevent steganographic information transfer. Active wardens with *active mapping* capability reduce the problem of ambiguities in network traffic (i.e. data that can be interpreted in multiple ways [LLC07]) by mapping a network and its policies [Sha02]. Afterwards, the mapped information is used by a NIDS to provide unambiguity [LLC07]. Based on the idea of active mapping and traffic normalization, Lewandowski et. al. presented another technique called *network-aware active wardens* [LLC07]. Network-aware active wardens have knowledge about the network topology and implement a stateful traffic inspection with a focus on covert channel elimination [LLC06, LLC07].

All these techniques have in common that they drop or modify parts of the network traffic regarding to their configuration. For our purpose, it is required to focus on this common dropping and modification feature. We decided to use only the term *normalizer* in the remainder because we only deal with the normalization aspect that all three mentioned techniques (normalizer, active mapper and network-aware active warden) have in common.

This paper contributes to the existing knowledge by presenting and discussing the problem of traffic normalization in a covert channel's *network environment learning phase* (NEL phase). Within the NEL phase, the covert channel peers try to find out which protocols they can use to communicate with each other (e.g. two journalists want to determine how they can communicate using application layer protocols). Thus, the NEL phase is significant for establishing a network covert channel. We show that traffic normalization

within the NEL phase results in a two-army problem. We evaluate passive monitoring not to be capable to solve this problem and present two means to overcome the two-army problem. The drawbacks of these results are discussed as well.

The remainder of this paper is organized as follows. Section 2 introduces the problem of traffic normalization within the NEL phase. Section 3 discusses means to overcome this problem and their possible drawbacks while Section 4 highlights the effect of four existing normalizers on the NEL phase. Section 5 concludes.

2 Related Work and the Problem of NEL-Normalization

Yarochkin et. al. presented the idea of adaptive covert channels, capable of automatically determining the network protocols which can be used between two covert channel communicators [YD+08]. Their approach filtered out blocked and non-routed network protocols within a two-phase communication protocol containing a "network environment learning" (NEL) phase as well as a "communication phase". Within the network environment learning phase, the covert channel software detects the usable network protocols, while the payload transfer is taking place within the communication phase. Wendzel and Keller extended this approach by introducing *goal-dependent* protocol sets, i.e., usable protocols are chosen with different probabilities to optimize a covert channel for goals such as providing a high throughput or sending as few data packets as possible to transfer a given payload [WK11].

A similar approach by Li and He is based on the idea of autonomic computing [LH11]. The authors try to detect *survival values* for embedded steganographic content in network packets, i.e. they evaluate whether hidden data was corrupted within the transmission, or not, and therefore use checksums, transmission rates, and sequence numbers [LH11]. However, Li's and He's approach requires a previously established connection to evaluate the results, what is not sufficient if a normalizer is located between sender and receiver, since the two-army problem cannot be solved under such circumstances.

Yarochkin et. al. are focusing on *whole* network protocols [YD+08]. They detect usable network protocols exchanged between two hosts by monitoring network data. Wendzel and Keller distinguish protocols on a finer scale, e.g. one "protocol" can be to set the "IP ID" as well as the "Don't fragment flag" in IPv4 while another protocol could only use the "Reserved" flag of IPv4 but both "protocols" belong to the same network protocol (IPv4) [WK11]. To prevent confusion, we call the network protocol (or the bits used within a network protocol) the "cover protocol" of a covert channel. Using this term, we can refer to both approaches at the same time.

Figure 1: Communication between two hosts A and B. Neither A nor B know about the existence of a normalizer (and its configuration) between them a priori.

All three mentioned methods, Yarochkin et. al., Wendzel and Keller, and Li and He, aim to gain information about usable cover protocols but do not focus on the problem of traffic normalization. When a traffic normalizer is located between two covert channel systems which want to exchange information about protocols they can send and receive, a normalizer can drop or modify the exchanged information (Fig. 1). It is also possible that more than one normalizer is located on the path between A and B but this does not differ from a single normalizer in our scenario.

Normalizers usually modify bits (such as they unify the TTL value in IPv4) or clear bits (such as the "Don't fragment" flag). Implementations like *norm* [HPK01], the *Snort Normalizer* [Sno11], the netfilter extension *ipt_scrub* [Bar08], or the OpenBSD packet filter scrubbing [Ope11][1] provide more than 80 normalization techniques for IPv4/IPv6, ICMP, TCP, UDP and the most important application layer protocols.

In the given scenario, host A and host B want to exchange information about the cover protocols, i.e. network protocols and the protocol specific bits they can use to communicate with each other. For example, host A is sending a network packet (e.g. a DNS request) to host B with the reserved flag set in the IPv4 header: If the normalizer clears the reserved flag (it is zero afterwards), host B cannot determine whether host A did not set the reserved flag or whether a normalizer (B and A are possibly not aware of a normalizer) modified the bit. Alternatively, the normalizer can – like a plain firewall – drop a packet if the reserved flag is set (in that case, host B does not know about the sent packet).

To exchange the information about all usable bits of all cover protocols a covert channel can support between A and B, it is required to test *all* bits and protocols in *both* directions (each received packet must be acknowledged to inform the sender about the successful sending). Since A cannot know which of his packets will reach B, A depends on the acknowledgement of B. If A can successfully send a packet to B, B cannot make sure that the response containing the protocol acknowledge information is not getting dropped by a normalizer (since B does not know which bits or protocols will reach A). Since neither A nor B can be sure whether their packets will reach the other system, they are left in an uninformed situation. Hence, the exchange of protocol information between A and B results in the *two-army problem* [Kle78]. The two-army problem cannot be eliminated but the uncertainty of A and B can be reduced by sending multiple (acknowledgement) messages or acknowledgement messages from each side (A acknowledges B acknowledgement, such as performed by the *three-way-handshake* of TCP [Pos81]). The next section discusses specific means for dealing with the two-army problem within the NEL phase.

3 Realizing the NEL Phase in Normalized Environments

If the covert channel software is able to determine a set of non-normalized cover protocols, it will enable the covert channel to communication without problems. Therefore, we define a set of cover protocols $P = \{x_1, ..., x_n\}$ (P contains all possible bit areas which can be used within a cover protocol) where, for instance, x_1 could represent "sending an IP packet

[1] Regarding to our investigation, *pf* supports a number of undocumented normalization features.

with DF flag set". An element $x_i \in P$ can contain other elements of P (e.g. x_1="set DF flag" and x_2="set the reserved flag", $x_3 = \{x_1, x_2\}$="set the reserved flag as well as the DF flag"). This condition is necessary, since a normalizer can include a rule to drop, modify or clear a packet (respectively bits) if a packet contains both x_1 *and* x_2, but does not apply the same rule (or no rule) in case of x_1 *or* x_2.

Based on this initial specification, we develop two different means for applying the NEL phase in normalized environments as well as in environments where A and B cannot be sure whether a normalizer is located between them.

The **first solution** is to use a pre-defined sequence of the elements of P. In comparison to the passive monitoring approach of Yarochkin et. al., this is an active approach. We describe a disadvantage of the passive approach in Sect. 3.1.

In the first step, A sends an ordered sequence $x_1, ..., x_n$ to B. The data hidden in x_i must contain information B can use to identify these as covert data (such as a "magic byte" as in *ping tunnel* [Stø09] or micro protocol data [WK11]). In case host B receives some of these packets, B can assume that A is currently sending the pre-defined packet sequence. Probably, host B will receive a sequence such as x_3, x_9, x_{10} (with bit i cleared), x_{11}, x_{17} (with bits j and k cleared), x_{19}, x_{20}, i.e. some protocols or bit-combinations were blocked (respectively modified) by a normalizer, or got lost. In this case, host B can never be completely sure that it is currently receiving the packet sequence from A but can use the information received to communicate with A in a limited way nevertheless. The complete process has to be done vice versa since normalizer rules can be direction-dependent. Afterwards, A and B will have a set of protocols they can receive from the other peer. A and B assume that the cover protocols of the packets that were received correctly can be used to send data back to the peer. After this step is completed too, the communication phase can start.

Since B (respectively A) do not know whether their own packets were received by the peer if the normalizer applies direction-dependent rules, and depend on the acknowledgement information from the peer itself, they do not know which cover protocol they can use to send their acknowledgement. Therefore, this solution results in the two-army problem again if the normalizer applies direction-dependent rules and has to be considered error-prone.

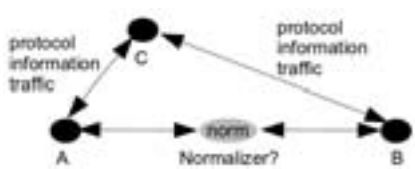

Figure 2: A and B exchanging protocol information using C.

A **second solution** (shown in Fig. 2) we want to present is to include a third (but temporary) participant C known to A and B. We assume that A and C, as well as B and C already exchanged the information about usable protocols, i.e. they are able to exchange

information in a way not dropped or cleared, even if a normalizer is present between them. C is not necessarily required to be aware of a covert communication between itself and A or B, i.e. C could be a shared temporary resource such as a Flickr account [BK07]. C is a temporary solution since A, B (and probably others) can only build a complex covert overlay network if new paths besides already known paths are created. If A and B would transfer all information between them using C, no new path would be created and C would be a single point of failure if no other connection between A and B exists. Additionally, it is more secure to transfer only information about usable protocols between A and B via C than it is to transfer all hidden information itself via C.

In this scenario, C can be used by A to inform B (and vice versa) about packets which A (respectively B) is about to sent. For instance: A sends C the information to tell B that A will now send an IP packet with DF bit set to B; C will forward the information to B. A will sent the described packet either after a short waiting time $< t$ to B or after B responds its waiting state to A (via C). The first method decreases the amount of exchanged data while the latter is less error-prone.

If B received the packet, B knows, that the DF flag can be used and was not cleared or dropped. In any case, B reports the reception/the miss of the announced cover protocol containing packet. The whole process works in both directions, i.e. A and B can exchange all required information about usable cover protocols. The temporary system C is not required to be a single hop within the overlay network, it can also be a chain of proxies $C = P_1 \rightarrow ... \rightarrow P_n$ represented as a single system in this second solution. The only difference between a single hop called C and a proxy chain called C is the internal forwarding within the system. Such forwarding for covert channel proxy chains must be bi-directional and can be achieved and optimized by known micro protocol means [WK11].

Discussion of the two presented means

While the first scenario (sending a pre-defined sequence) is error-prone, the second scenario depends on a third communicator C connected to A and B. A combination of both scenarios will be a promising approach for further implementations.

A drawback in the second solution exists in case C is aware of covert communication (i.e. not a passive system, such as a Flickr account), since C can manipulate the covert communication. For instance, C could drop all protocol information requests from A to B (and vice versa). If A and B are able to exchange cryptographic keys a priori, they can sign messages sent via C to prevent undetected message modifications. However, the use of signatures does not prevent C from actively dropping the messages. Therefore, it is recommended to use a passive C.

3.1 Improved Protocol Determination Strategies

If a covert channel implementation supports a large amount of cover protocols (or some areas containing many bits), the packet count required to transfer all feasible bit-combinations through the possible normalizer can become high, thus the number of required packets for the protocol determination can also be high.

For example, if $P = \{x_1, x_2, x_3\}$ (say each x_i is representing one bit of data in this case), it is possible to send each element alone, in combination with another element, or all three elements at the same time, i.e. there are seven possible combinations. To verify the possible use of these protocols using strategy #2 (as explained before), it would require 14 packets to verify all elements of P and their combinations for the cover protocols between A and B (the packets exchanged between A and C as well as between B and C are not included in this calculation).

If P contains many elements, the number of required packets can result in raising a high attention due to the abnormal traffic pattern [WK11]. Therefore, it is mandatory to reduce the number of required packets.

A reduction of the necessary network packets for determining the usable cover protocols between A and B can result in a loss of quality in the determined protocol information. We present two reduction strategies, where the first strategy results in a loss of quality while the second does not. These strategies are presented since each practical implementation can profit from them to keep a low profile.

Reduction Strategy 1: If each protocol layer (e.g. the TCP/IP layers) can be scanned independent of the other layers, the number of possible combinations decreases. For instance, if x_1 and x_2 refer to bits within the IPv4 header, and x_3 refers to UDP, there are only six (instead of eight) combinations left to try. This scan technique has the drawback that normalizer rules which depend on multiple layers are not taken into account.

Reduction Strategy 2: If each different covering network protocol on the same network layer is scanned independent from all other network protocols on the same layer, a loss in the quality of the resulting information is not possible since different protocols usually do not depend on other protocols on the same layer.[2] For instance, if $P = \{x_1, x_2\}$ where x_1 modifies two bits within the TCP header and x_2 modifies one bit within the UDP header, there are only 6 possible bit combinations to verify.

Critique on the passive approach: Another simplification is thinkable but less valuable for real implementations: Regarding to the passive monitoring approach presented by Yarochkin et. al. (as discussed in Sect. 2), A could try to use only protocols for the communication with B which it can receive on its own network interface. However, the occurrence of a network protocol on a network interface of A does not necessarily mean that a normalizer will forward such network packets to A in any case. If, on the other hand, the source address of such a network protocol's packet does not belong to A's subnet, it is the case that such a network protocol is forwarded to the network of A. Such a packet has probably passed a normalizer but since there can be more than one normalizer between A and B, this solution is also error-prone and cannot be recommended for implementations. Of course, one could argue that A could wait for network packets containing the source address of B's subnet, but this is only realistic in the case of regular communication between both subnets. Additionally, filter rules can be address-based instead of interface- or subnet-based, i.e. it is possible that a normalization rule applies for some hosts in B's (A's)

[2] This second rule is mandatory since it is not possible for network packets to contain two different network protocols of the same network layer. An exception is to use network tunneling but in network tunneling, a single network layer also does not really contain two protocols, but a layer can occur multiple times.

subnet but not for all hosts in the subnet. Thus, this approach is only useful for a direct communication between B and A, and due to that limitation provides no improvement in comparison to the already presented means. However, the passive approach is linked to a number of drawbacks: It requires additional waiting time and is is not successfully adaptable to most overlay network situations, since information about the underlay network are not necessarily available. Mentionable as well, is that a *direct* third party network data exchange between A and B, which is required for the monitoring, will usually be rare and will additionally not necessarily include all elements of P.

3.2 The Problem of Dynamic Routing Environments

Problematic as well is the possibility for traffic to take different routes on the path between A and B within the underlay network, where one packet could pass a normalizer and another packet does not (or some packets are transferred over different normalizers). Since neither A nor B are able to control the routing decisions between them and do not necessarily possess knowledge about the underlay network structure, A and B cannot overcome this situation.

3.3 Proof of Concept Implementation

A proof of concept implementation was developed to run a simulated NEL phase. A covert channel receiver program (representing B) was built to accept information from a temporary participant (representing C) as shown in strategy #2. The input from C to B was scripted by hand and the traffic from A to B was generated using the network packet generator scapy (*www.secdev.org/projects/scapy/*), while the covert receiver software is based on *libpcap* (*tcpdump.org*). Using scapy, we were able to simulate packet loss, perfect packet transfer and traffic modifications. If packet loss occurs, B assumes that a packet was filtered. Since uncommon packet occurrences can result in raising a high attention, we cannot recommend to sent such packets multiple times within a small time slice to deal with packet loss. Non-normalized packets obviously result in an uncorrupted NEL phase (excluding lost packets). The effect of traffic modifications turned out to produce multiple packets containing the same flag combinations on system B. For instance, if A sends an IPv4 packet containing the DF flag set as well as one packet with an unset DF flag, B will receive two packets with DF=0 if a normalizer clears the DF flag. B must ensure not to accept packets using the same bits as announced by A from other sources, thus B must apply filter rules (e.g. using libpcap filter settings to accept only packets from A). The covert channel receiver B transfers the information of an useless cover protocol back to A (using C) if such a doubled cover protocol was received. We implemented the acknowledgement transfer to A using a micro protocol as described in [WK11].

4 Effects of Existing Normalizers on the NEL Phase

As discussed in the previous sections, traffic normalizers can drop, modify and clear cover protocols. Thus, for realizing a covert channel's NEL phase, it is important to obtain knowledge about the usable cover protocols a priori. By analyzing four normalizers (*pf scrubbing*, *norm*, the Linux netfilter/iptables extension *ipt_scrub*, and the *Snort normalizer*), we were able to carve out general rules which can be applied to the NEL phase.

Different traffic normalizers contain different capabilities. For instance, *Snort* and *norm* are able to clear the IPv4 reserved flag, but *pf* is not. Some capabilities are configuration-dependent, e.g. the handling of the "Don't Fragment" flag in *pf* depends on the usage of the "no-df" rule in the configuration file. Additionally, the amount of features differs significantly: While *norm* supports more than 70 normalization rules, *ipt_scrub* supports only 8 of them. Additional differences apply for the amount of supported network protocols.

For the NEL phase, it is useful to define the elements of P by verifying their support within normalizers: Elements which are not supported by any or by very few normalizers are likely to reach their destination within the NEL phase, while elements which are supported by (almost) any normalizer have a higher probability to get normalized.

We compared the features of the four normalizers and found out that several normalization capabilities are provided by most of the implementations: **IPv4**: TTL modification, removing DF flag, reserved bit and options and drop packets with IHL <5 or > 5[3]. **IPv6**: modify the hop limit, modify or remove optional headers. **ICMP** modify/drop ICMP type 0 and 8. **TCP**: Clear the reserved bits, drop packets with unusual flag or flag/data combinations (e.g., SYN=1 and RST=1, SYN=1 and FIN=1, or SYN=1 with payload), drop in case of a small header length. **UDP** (norm)/**HTTP** (Snort): Only supported by one normalizer.

Additionally, we want to provide a general explanation which cover protocols we can call useful in our scenario. UDP and HTTP are currently only supported by one of the normalizers, which makes them good choices for bypassing normalizers within the NEL phase. On the other hand, there are network protocols which are not in the scope of any of the normalizers, e.g. application layer streaming protocols, frames on the network access layer, or dynamic routing protocols, what makes them good choices for the NEL phase too. If such protocols are used, it is important to understand that the occurrence of rarely used network protocols can raise the attention of a covert channel [WK11] (e.g. the usage of IGRP in an OSPF network). Thus, the usefulness of rarely used protocols for the NEL phase is low. A passive monitoring can be done to determine the occurrence rates of network protocols (but as described in Sect. 3.1, passive monitoring is not recommendable to detect *cover* protocols). Tab. 1 shows occurrence rates of transport layer network protocols within different passive traffic recordings and demonstrates the significant differences within different Ethernet networks. These three traffic recordings where downloaded from *http://www.simpleweb.org/wiki/Traces*, a website providing a number of *tcpdump* recordings (mainly from universities). These different protocol occurrences do not only exist for network protocols but for types of cover protocols as well, as shown in Tab. 2 in case of

[3] In case of OpenBSD, this is not specified in the manual, but found in a source code analysis for this paper.

Source	ARP	TCP	UDP	ICMP	IGMP
Simpleweb-Loc1	0.02%	77.67%	21.94%	0.25%	<0.001%
Simpleweb-Loc2	-	92.46%	0.08%	3.20%	<0.01%
Simpleweb-Loc3	-	68.88%	3.95%	27.12%	<0.01%

Table 1: Occurrence rates of different network protocols in three traffic recordings downloaded from simpleweb (each source contained between 750.000 and 1.200.000 packets).

Source	Type 0	Type 3	Type 8	Type 11
Simpleweb-Loc2	2.96%	88.93%	7.97%	0.13%
Simpleweb-Loc3	2.24%	0.08%	97.66%	0.02%

Table 2: Relative occurrence rates of ICMP types in three networks.

the different ICMP type values (ICMP is a protocol that is used by popular covert channel tools such as *pingtunnel* [Stø09]).

To summarize, the presented strategies of Sect. 3 to overcome the normalization within the NEL will work, even if a combination of all four normalizers would be present. The **first strategy** (sending a pre-defined sequence) can only be affected, if a huge part of the sequence will be normalized (or if the first packet containing the *magic byte* will be cleared). The solution for this problem is to choose only protocols not supported by the normalizers. Since none of the normalizers can affect an already working covert channel via a temporary participant, the **second strategy** can be applied in any case.

5 Conclusion

In this paper, we have shown that the process of determining possible cover protocols within the so called *Network Environment Learning* (NEL) phase is problematic if a normalizer is located on the path between two covert channel peers since it will result in a two-army problem. We additionally presented two techniques to overcome this not completely solvable problem of one or more possible normalizers which sender and receiver are not aware of, and discussed possible drawbacks of these approaches. We have evaluated three existing approaches to be insufficient for this purpose and implemented a proof of concept application to verify our presented approach. Finally, we analyzed four available normalizers to extract information about cover protocols which can be used for the NEL phase without a high probability of getting dropped or modified.

Acknowledgement

I would like to thank Jörg Keller (University of Hagen) for his valuable contributions and guidance. I would also like to thank the anonymous reviewers for their feedback.

References

[And08] R. Anderson. *Security Engineering - A Guide to Building Dependable Distributed Systems*. Wiley, 2 edition, 2008.

[Bar08] N. Bareil. ipt_scrub: scrubbing for Netfilter, May 2008.

[BGC05] V. Berk, A. Giani, and G. Cybenko. Detection of Covert Channel Encoding in Network Packet Delays. Technical report, Department of Computer Science - Dartmouth College, 2005.

[BK07] A. Baliga and J. Kilian. On covert collaboration. In *Proceedings of the 9th Workshop on Multimedia & Security*, pages 25–34, New York, NY, USA, 2007. ACM.

[BP09] K. Borders and A. Prakash. Quantifying Information Leaks in Outbound Web Traffic. In *30th IEEE Symposium on Security and Privacy*, pages 129–140, 2009.

[CBS04] S. Cabuk, C. E. Brodley, and C. Shields. IP covert timing channels: design and detection. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 178–187. ACM, 2004.

[Fad96] Y. A. H. Fadlalla. *Approaches to Resolving Covert Storage Channels in Multilevel Secure Systems*. PhD thesis, University of Brunswick, 1996.

[FFPN03] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil. Eliminating Steganography in Internet Traffic with Active Wardens. In *Revised Papers from the 5th International Workshop on Information Hiding*, IH '02, pages 18–35, London, UK, UK, 2003. Springer-Verlag.

[Gir87] C. G. Girling. Covert Channels in LAN's. *IEEE Transactions on Software Engineering*, 13:292–296, February 1987.

[GP+08] G. Gu, R. Perdisci, et al. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In P. C. van Oorschot, editor, *USENIX Security Symposium*, pages 139–154. USENIX Association, 2008.

[HPK01] M. Handley, V. Paxson, and C. Kreibich. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *10th USENIX Security Symposium*, volume 10, pages 115–131, 2001.

[HS96] T. G. Handel and M. T. Sandford, II. Hiding Data in the OSI Network Model. In *Proceedings of the First International Workshop on Information Hiding*, pages 23–38, London, UK, 1996. Springer-Verlag.

[Hu91] W.-M. Hu. Reducing Timing Channels with Fuzzy Time. In *1991 Symposium on Security and Privacy, IEEE Computer Society*, pages 8–20, 1991.

[JMS10] B. Jankowski, W. Mazurczyk, and K. Szczypiorski. Information Hiding Using Improper Frame Padding. *eprint arXiv:1005.1925*, 2010.

[Kem83] R. A. Kemmerer. Shared resource matrix methodology: an approach to identifying storage and timing channels. *ACM Transactions on Computer Systems*, 1(3):256–277, 1983.

[Kle78] L. Kleinrock. Principles and Lessons in Packet Communications. In *Proc. of the IEEE*, volume 66, pages 1320–1329, November 1978.

[KMC05] M. H. Kang, I. S. Moskowitz, and S. Chincheck. The Pump: A Decade of Covert Fun. In *ACSAC*, pages 352–360. IEEE Computer Society, 2005.

[Lam73] B. W. Lampson. A Note on the Confinement Problem. *Commun. ACM*, 16(10):613–615, 1973.

[LG10] G. Locke and P. D. Gallagher. Guidelines for Smart Grid Cyber Security: Vol. 3, Supportive Analyses and References (NIST Interagency/Internal Report (NISTIR) - 7628), 2010.

[LGC08] Z. Li, A. Goyal, and Y. Chen. Honeynet-based Botnet Scan Traffic Analysis. In W. Lee, C. Wang, and D. Dagon, editors, *Botnet Detection*, volume 36 of *Advances in Information Security*, pages 25–44. Springer, 2008.

[LH11] W. Li and G. He. Towards a Protocol for Autonomic Covert Communication. In *Proc. 8th Conf. on Autonomic and Trusted Computing*, pages 106–117, 2011.

[LLC06] N. Lucena, G. Lewandowski, and S. Chapin. Covert Channels in IPv6. In George Danezis and David Martin, editors, *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 147–166. Springer Berlin / Heidelberg, 2006.

[LLC07] G. Lewandowski, N. Lucena, and Steve C. Analyzing Network-Aware Active Wardens in IPv6. In Jan Camenisch, Christian Collberg, Neil Johnson, and Phil Sallee, editors, *Information Hiding*, volume 4437 of *Lecture Notes in Computer Science*, pages 58–77. Springer Berlin / Heidelberg, 2007.

[MWJH00] G.R. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and application protocol scrubbing. In *Proc. of the INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, pages 1381–1390, 2000.

[Ope11] OpenBSD Project. pf.conf - packet filter configuration file (manual page), July 2011.

[PK91] P. A. Porras and R. A. Kemmerer. Covert Flow Trees: A Technique for Identifying and Analyzing Covert Storage Channels. In *IEEE Symp. on Security and Privacy*, pages 36–51, 1991.

[Pos81] J. Postel. RFC 793: Transmission Control Protocol. DARPA Internet Programm Protocol Speicification, September 1981.

[RM08] B. Ray and S. Mishra. A Protocol for Building Secure and Reliable Covert Channel. In Larry Korba, Steve Marsh, and Reihaneh Safavi-Naini, editors, *PST*, pages 246–253. IEEE, 2008.

[Row97] C. H. Rowland. Covert Channels in the TCP/IP protocol suite. *First Monday*, 2(5), May 1997.

[Rut04] J. Rutkowska. The implementation of passive covert channels in the linux kernel, 2004.

[Sch10] K. Schmeh. Covert Channels in elektronischen Ausweisen. In Christian Paulsen, editor, *Sicherheit in vernetzten Systemen: 17. DFN Workshop*. BoD, 2010. (In German).

[Sha02] U. Shankar. Active Mapping: Resisting NIDS Evasion Without Altering Traffic. Technical Report UCB//CSD-2-03-1246, Computer Science Division (EECS) (University of California Berkeley), December 2002.

[SN+06] A. Singh, Ö. Nordström, et al. Stateless Model for the Prevention of Malicious Communication Channels. *International Journal of Computers and Applications*, 28:285–297, 2006.

[Sno11] Snort Project. Snort Users Manual 2.9.0, March 2011.

[Stø09] D. Stødle. Ping Tunnel – For those times when everything else is blocked, 2009.

[WAM09] C. Wonnemann, R. Accorsi, and G. Müller. On Information Flow Forensics in Business Application Scenarios. *IEEE COMPSAC Workshop on Security, Trust, and Privacy for Software Applications, IEEE*, 2009.

[WK11] S. Wendzel and J. Keller. Low-attention forwarding for mobile network covert channels. In *Proc. of the 12th Conference on Communications and Multimedia Security*, pages 122–133, 2011.

[Wol89] M. Wolf. Covert channels in LAN protocols. In Thomas Berson and Thomas Beth, editors, *Local Area Network Security*, volume 396 of *Lecture Notes in Computer Science*, pages 89–101. Springer Berlin/Heidelberg, 1989.

[YD+08] F. V. Yarochkin, S.-Y. Dai, et al. Towards Adaptive Covert Communication System. In *PRDC*, pages 153–159. IEEE Computer Society, 2008.

[ZAB07] S. Zander, G. Armitage, and P. Branch. Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Comm. Magazine*, 45(12):136–142, Dec 2007.