Design and Implementation of an Active Warden Addressing Protocol Switching Covert Channels

Steffen Wendzel[1,2], Jörg Keller[1]
[1]*Department of Mathematics and Computer Science, University of Hagen, Germany*
[2]*Department of Computer Science, Augsburg University of Applied Sciences, Germany*
steffen.wendzel1@hs-augsburg.de, joerg.keller@fernuni-hagen.de

***Abstract*—Network covert channels enable a policy-breaking network communication (e.g., within botnets). Within the last years, new covert channel techniques occurred which are based on the capability of protocol switching. There are currently no means available to counter these new techniques. In this paper we present the first approach to effectively limit the bandwidth of such covert channels by introducing a new active warden. We present a calculation method for the bandwidth of these channels in case the active warden is used. Additionally, we discuss implementation details and we evaluate the practical usefulness of our technique.**

***Keywords*-Protocol Switching Covert Channel; Protocol Channel, Active Warden**

I. INTRODUCTION

The term *covert channel* refers to a type of communication channel defined as *not intended for information transfer* by Lampson [1]. While covert channels can occur on local systems, we focus on covert channels within networks. The goal of using covert channels is to transfer information without rising attention while breaking a security policy [2].

Covert channels have been a focus of research for decades. A number of publications (e.g., [3], [4], [5]) describe how to implement covert channels in network packet data and packet timings. Techniques were developed to deal with the problem of covert channels, like the pump [6], which is a device that limits the number of acknowledgement messages from a higher to a lower security level and thus affects covert timing channels based on ACKs. Other well-known techniques are for instance covert flow trees [7], the shared resource matrix (SRM) methodology [8] the extended SRM [9], and steganalysis of covert channels in VoIP traffic [10].

A well-known technology to counter covert channels is the active warden, i.e. a system counteracting a covert channel communication. While passive wardens monitor and report events (e.g., for intrusion detection), active wardens (e.g., traffic normalizers [11]) are capable of modifying network traffic [12] to prevent steganographic information transfer.

Recently, the capability to keep a low profile resulted in a rising importance of network covert channels because of their use cases. For instance, covert channels can be used to control botnets in a hidden way [13]. Covert channel techniques can also be used by journalists to transfer illicit information, i.e., they can generally contribute to the free expression of opinions [14].

A covert channel able to switch a network protocol based on a user's command called *LOKI2* was presented in 1997 [15]. Within the last decade, different new covert channel techniques occurred and not all of them are already addressed by protection means. However, these new techniques enable covert channels to switch their communication protocol automatically and transparently, as well as they were enabled to cooperate in overlay networks by using internal control protocols as presented in [16].

In this paper, we focus on two new covert channel techniques, *protocol switching covert storage channels* (also known as *protocol hopping covert channels*, PHCC) as well as so called *protocol channels* (PCs). PHCC were presented in [17] and were improved in [16]. These channels transfer hidden information using different network protocols to raise as little attention as possible due to peculiar protocol behaviour. For instance, a simple PHCC could use the "User-Agent" field in HTTP as well as the message number in POP3 "RETR" requests to transfer hidden information.

PCs were introduced in [18] and signal information solely by transferring network protocols of a pre-defined set in an order that represents hidden information. Protocols are therefore linked to secret values, e.g., a HTTP packet could represent the value "1" and a POP3 packet could represent the value "0". To transfer the message "110" via this PC, the sender is required to send two HTTP packets followed by a POP3 packet. The bandwidth of a PC is usually limited to a few hundred bit/s, however, for an attacker this is fast enough to transfer passwords, selected records or tweets.

A first detection algorithm for PC (but not for PHCC) was implemented in [19], but there is no work done to limit the bandwidth of PC and PHCC or to prevent them.

We present the first concept as well as an implementation of an active warden able to significantly reduce the bandwidth of both, PHCC and PC. While we do not present a universal solution countering all covert channels, this is the first work discussing means against PHCC and PC. The limitation of these advanced covert channels is more challenging in comparison to single-protocol covert

 ISBN: 978-1-61208-201-1

channels. We evaluate our results by simulation experiments. We demonstrate that our approach is useful for the practical operation in organizations and that the active warden decreases the attractiveness of both channel types for attackers.

The remainder of this paper is structured as follows. Section II introduces the idea and implementation of our active warden, while Section III discusses results and Section IV focuses on the practical aspects of the presented approach. Section V concludes.

II. Design and Implementation

The idea of an active warden addressing PHCC and PC focuses on one aspect both covert channel types share: the protocol switches. For both channel types, it is a necessity to ensure that network packets using different network protocols reach their destination in the same order as they were sent. Our approach of an active warden for countering PHCC and PC monitors the protocol switching behaviour of network hosts and introduces delays in network packets if a protocol switch occurs.

Like the network pump, we have no explicit detection capabilities in our active warden but aim on limiting the channel's bandwidth nevertheless. In comparison to the pump, we do not limit ACK messages but alter protocol occurrences. Using the presented technique, we can prevent performance decreases for downloads and uploads and minimize practical implications (cf. Section IV) between (autonomous) systems. The system only affects those network packets which change the last occurring network protocol.

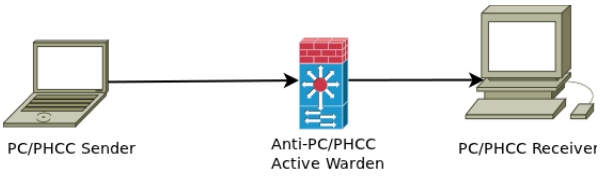

Figure 1. Location of the Anti-PC/PHCC active warden

The active warden must be located between a sender and a receiver (Figure 1). To prevent PC/PHCC-based data leakage for enterprises, a suitable location would be the company's uplink to the internet. The active warden's delay is configurable and thus makes our approach adjustable, i.e. an administrator can choose the individual optimum between maximized protection and maximized throughput. Formally, this creates a multi-criterion optimization problem. For a given delay d, data leakage can occur at a maximum rate $R(d)$, that is decreasing with increasing d. On the other hand, the side-effects for legitimate users will increase with increasing d, i.e. can be modeled by a function $S(d)$. Ideally, one would like to minimize both, R and S, which is however not possible. One can combine the two in a target function

$$\text{penalty}(d) = \epsilon \cdot R(d) + (1 - \epsilon) \cdot S(d) \ ,$$

which is then to be minimized, i.e. after fixing a suitable $\epsilon \in [0; 1]$ the minimization results in an optimal d which represents the administrator's priorities. As both R and S are assumed to be monotonous, a Pareto optimum can be found in the sense that a further reduction of R by increasing d cannot be achieved without increasing S. Typically, instead of using R and S directly, they are normalized to a certain range such as interval $[0; 1]$, and they might be adapted by linear or non-linear functions that reflect e.g. the severeness of an increased leakage.

Imagine a PC using ICMP (1 bit) and UDP (0 bit) and the goal to transfer the message "0110001". In this case, the sender would need to send UDP, ICMP, ICMP, UDP, UDP, UDP, ICMP. If our active warden is located on a gateway between both hosts and can delay packets which probably belong to a PC or PHCC, the successful information transfer will be corrupted. At the beginning, the sender sends a UDP packet which is forwarded by the active warden. Afterwards, the sender sends the ICMP packet, which is delayed for a time d because a protocol switch happened. The next packet is an ICMP packet again and therefore not delayed but forwarded. Afterwards a UDP packet occurs which is delayed for a time d, too. The next two UDP packets do not change the last protocol and are therefore forwarded. The last ICMP packet results in an additional packet switch and is therefore delayed for time d again. If d is 1 second, then all delayed packets will arrive after the non-delayed packets if the sender did not introduce synthetic delays itself. The resulting packet order at the receiver's side will be UDP, ICMP, UDP, UDP, ICMP, UDP, ICMP, or "01**00**101" (containing two incorrect bit values).

The situation is similar for PHCC where the hidden information is not represented through the protocol itself but through alternations of a protocol's attributes (such as the IPv4 TTL or the HTTP "User-Agent"). If the active warden modifies PHCC transmissions sent via different protocols, the reassembled payload will be jumbled.

In order to prevent the covert channel users to take advantage of learning about the value of d, it might also be randomized, cf. Section III.

A. Bandwidth Calculation

Tsai and Gligor introduced the formula $B = b \cdot (T_R + T_S + 2 \cdot T_{CS})^{-1}$ for calculating the bandwidth of covert channels using the values T_R (time to receive a covert message), T_S (time to send a message), T_{CS} (time required for the context switch between processes), and b (amount of transferred information per message) [20]. While the formula addresses local covert storage channels, all parameters are adaptable to a network covert channel as well.

We use a modification of this formula (cf. formula 1) to calculate the bandwidth of a PC in case our active warden is located between sender and receiver. We introduce the active warden's delay d multiplied with the probability p of

 ISBN: 978-1-61208-201-1

a protocol switch per packet. Instead of T_R, T_S and T_{CS} we use T to represent the transmission time (i.e., the time difference between receiving and sending a packet including the processing time required by the active warden).

The amount of transferred data per packet (b) is 1 bit/packet in case two protocols are used for a PC since $b = \log_2 n$, where n is the number of protocols used. Thus, p as well as n will rise if more than two protocols are used (more information can be transferred per packet but the switching rate will also increase). In case of a PHCC, b depends on the amount of storage data per packet and not on the amount of protocols used. Therefore, p will rise if more protocols are used but since the amount of protocols is not linked to the amount of information transferred per packet, b will *not* rise if more protocols are used.

$$B = b \cdot (p \cdot d + T)^{-1} \quad (1)$$

Theoretically, p is 0.5 if randomized input, a uniform coding and a set of two protocols is used since the next packet is either using the same protocol as the last (no protocol switch) or the other protocol (a protocol switch is taking place). In our experiments, the average protocol switching value for a typical protocol switching covert channel using only two protocols was $p = 0.4738806$. Thus, to transfer information without risking a corruption through a delay, a PC/PHCC user is forced to send packets with protocol switches in a way that the delay d cannot corrupt the packet order.

As mentioned earlier, the value p depends on the amount of protocols used as well as on the channel's coding. If a uniform coding was used (as with optimized channels) and if two protocols are used p is approx. 0.5. In case four protocols are used, p is approx. 0.75. In general, for n protocols used p is $(1 - 1/n)$. Thus, formula 1 can be modified to the following version:

$$B = b \cdot ((1 - 1/n) \cdot d + T)^{-1} \quad (2)$$

Protocol Channel: As also already discussed, $B = \log_2 n \cdot ((1 - 1/n) \cdot d + T)^{-1}$ in case of a PC. Taking d as well T into account using formula 1, we calculated the maximum useful bandwidths for an uncorrupted PC transfer using a set of two protocols (Figure 2). According to our calculations, a PC's bandwidth can be reduced to less than 1 bit/s if the active warden introduces a delay of 2.0 sec for protocol switches using realistic values for T.

Protocol Hopping Covert Channel: For a PHCC, the parameter b varies more than parameter T (T is, as in the case of a PC, usually very low). Therefore, we created a different plot where we set T to the static value 0.005 which we measured in experiments. Figure 3 shows the bandwidth of a PHCC dependent on the delay d as well as the amount of transferred bits per packet b. Obviously, the result of the PHCC is equal to the result of a PC if $b = 1$. However, PHCCs can carry more information and are therefore harder to limit, i.e., they require higher delays.

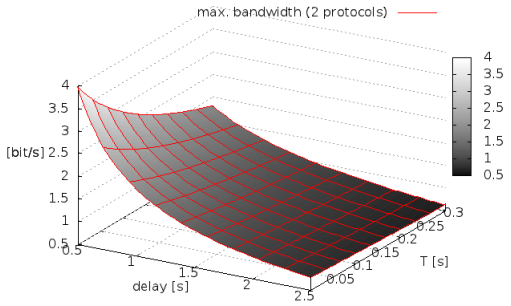

Figure 2. A PC's bandwidth (B) using a set of two protocols dependent on the delay and the transfer value.

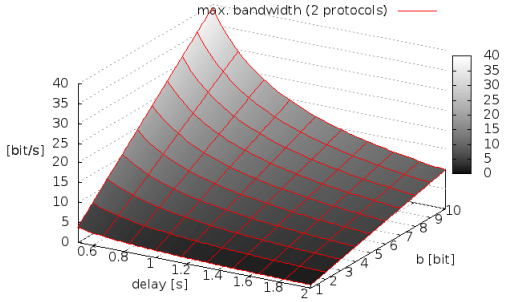

Figure 3. A PHCC's bandwidth using *two* protocols, $T = 0.005$ and delays between 0.5 and 2s as well as the capability to transfer between 1 and 10 bits per packet.

As shown in Figure 4, the bandwidth of a PHCC decreases if the number of protocols used increases, since more protocol switches ($p = 1 - 1/4$) occur, i.e., the active warden's efficiency for PHCCs is positively correlated with p.

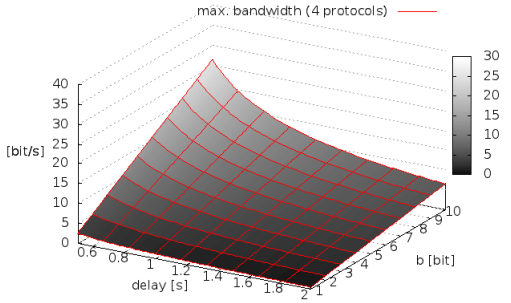

Figure 4. A PHCC's bandwidth using *four* protocols, $T = 0.005$ and delays between 0.5 and 2 as well as the capability to transfer between 1 and 10 bits per packet.

B. Implementation

For our test implementation, we set up a virtual network between two virtualized Linux 3.0 systems (a covert channel sender as well as a receiver with a local active warden

instance) using *VirtualBox* (*www.virtualbox.org*). Both hosts were connected using a virtual Ethernet interface and IPv4. Our proof of concept code focused on layer 4 protocols identifiable by the IPv4 "protocol" field only. Therefore we modified the *protocol channel tool* (pct) [21] that used ARP and ICMP to use UDP and ICMP instead. Additionally, we implemented the functionality to generate randomized input and to adjust the channel's bitrate.

To implement the network delays on the receiver system that is acting as both the active warden gateway and the protocol switching covert channel receiver at the same time, we made use of the firewall system *netfilter/iptables*. Netfilter/iptables provides a "QUEUE" feature which can be used to redirect data packets to a userspace program. Berrange implemented the Perl-based program *delay-net* [22] that enforces configurable network delays using the *IPQueue* module [23] which is based on the iptables QUEUE feature. We modified delay-net in a way that it only delays packets after a protocol switch happened. We also implemented a third program to evaluate the correct transmission at the receiver's side to verify formula 1 (cf. Section III).

Since this test focuses on the protocol switching capability of both the PC and the PHCC at the same time, the testing method is valid for both covert channel types.

III. Results

In our test configuration, the value T is quite small (we measured 0.005 in average) in comparison to the delay time d. As mentioned in the previous section, we were able to determine $p = 0.4738806$ through observing the behaviour of the modified *pct* in our virtual test network. However, p turned out to be slightly higher (0.53) in a real network environment, where protocols occur which do not belong to the PC, and therefore result in few additional protocol switches.

In our test setting we sent PC data using different bitrates and monitored the correctness of the received packet order at the receiver's side. Using this method, we were able to find out the maximum bitrate able to work error-free dependent on the delay introduced by the active warden. Figure 5 shows the results in comparison to our calculation of B (formula 1). The differences between B and our recorded values are small. An active warden with a delay value $d = 2.1\ s$ ($T = 0.005$) reduces the bandwidth limit required for a successful transmission of data to $1bit/s$. If $d = 1.0\ s$, the bandwidth limit is reduced to a maximum of $2.088bit/s$.

PC with improved coding: If a PC uses a coding that requires to send new packets only if a value unequal to the current value is required to be transferred, it can overcome the active warden, if sender and receiver are synchronized. This is possible if the sender only transfers a network packet if a protocol switch occurs, i.e. two packets of the same protocol are never transferred after another. The timing intervals between the protocol switches represent the amount

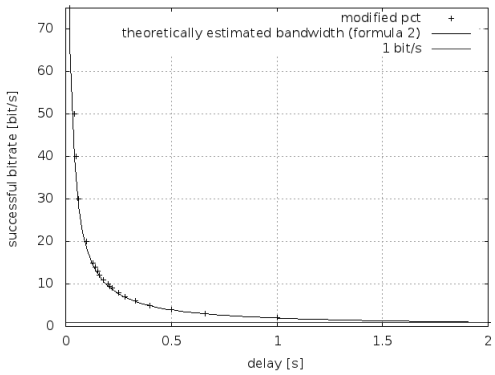

Figure 5. Measured maximal bandwidth of the modified *pct* dependent on the active warden's delay. In comparison, we show formula 1 using the estimated protocol switching value.

of bits to transfer. Thus, such a covert channel would be a hybrid version of a timing channel and a PC.

Example: The sender sends a packet of protocol P_1. The active warden delays it for time d and forwards it. Three more bits as represented by P_1 shall be transferred. Therefore, the sender waits for three time slots. Afterwards, a bit represented by P_2 shall be transferred and the sender sends one such a packet. The active warden delays the packet for d and forwards it. If the sender sends P_1 again, it will also be delayed for d. The receiver will receive P_1, three waiting slots, P_2, P_1, i.e. the same input as was sent by the sender. The only disadvantage introduced by the active warden is the delay of d for all packets but this is a minor consequence.

To overcome this problem, an improved version of the active warden can be developed. In our previous experiments, we focused on an active warden with a constant delay d. If d varies from packet to packet, or in other words, d is randomized (e.g., $d \in [0.1; 2]$ sec), previous packets are likely to overrun newer ones if the timing interval of the sender is too small. Thus, the sender is forced to use big waiting times and thus, will be forced to decrease its bandwidth.

Besides the previously mentioned coding, a PC could also use other codings to improve the amount of bits transferred per protocol switch (b/p). For the default PC coding and two protocols, $p = 0.5$ but when a run length limited (RLL) coding (as used for hard disks [24]) is implemented, p can be decreased.

In case of geometrically distributed symbols, an optimized coding (Huffman coding) can help the covert channel's users to minimize the amount of packets to transfer, but – as usual for covert channel research – we focus on an optimized coding using a uniform distribution (e.g., the covert channel is used to transfer encrypted input).

PHCC: A drawback of our approach is linked to a feature only available for PHCCs but not for PCs. PHCCs provide usually enough covert space to contain sequence numbers in

 ISBN: 978-1-61208-201-1

their payload [16]. Using these sequence numbers, the receiver can reassemble network packets even if their ordering was disturbed [17]. While the active warden is not able to completely solve this problem, it forces a PHCC user to use a sequence number. Such a storage channel-internal sequence number usually consists of only 2-3 bits and thus, the active warden can break the receiver-side sorting nevertheless, if d is large enough. Additionally, since these channels do only provide a few bits per packet, the active warden decreases the available space per packet in this way. Thus, a user is forced to send more packets to transfer the same amount of data than in the case where no active warden would be located on the path between PHCC sender and PHCC receiver.

Receiver-side re-calculation attack for PCs: In case a constant delay is used, it is possible for the attacker recalculate the original sequence of received packets of a PC. However, due to the jitter, it is possible that the attacker is forced to use error-correcting codes. The active warden can implement the previously mentioned randomized d to overcome this problem.

IV. Discussion of Practical Aspects

A goal of the presented active warden approach is to design the system for a practical use. The requirement for only small delays is – even if a user's initial request to a website will be delayed – an acceptable limitation for legitimate traffic in high-security environments since delays of only around 2s can reduce the useful bandwidth of PCs to a maximum of 1 bit/s. For PHCC, the value can differ if the channel provides high values for b. However, to achieve the goal of practical usefulness, it is necessary to implement additional functionality because of the following reasons:

DNS requests: Typically, a user sends DNS requests to a DNS server and, after receiving a response, connects to a system using another protocol. It is required to take care of this typical effect (and similar effects such as using HTTPS right after a user clicked on a link of a HTTP-based website). Protocol switches occur in both cases (DNS→HTTP respectively HTTP→HTTPS). The DNS server and the system a user connects to (usually a web server) will in almost all cases have different addresses, thus it is easy to address this problem if the active warden distinguishes destination hosts.

Different protocols on a single host: However, situations are thinkable in which a user is connected to one host using two different protocols, e.g. to an SMTP server and an IMAP server on the same system. In such cases, whitelisting (e.g., defining trusted hosts) can reduce this problem.

Multiple senders: In an enterprise network, there are usually a number of different computers with Internet access. If the active warden is located on the uplink, it will notice many protocol switches since different systems use different protocols to achieve different tasks. The active warden should distinguish source addresses to solve this problem.

However, some companies run a *network address translation* (NAT) service *within* their network. These systems would appear as a single system to the active warden although the systems as well as the active warden are located inside the company network. Thus, the NAT'ed systems would face delays. A whitelisting is no sufficient solution since these address translated systems are required to be protected from data leakage too. A thinkable limitation for that problem would be to use *remote physical device fingerprinting* [25] to count the number of NAT'ed systems and apply fewer delays per packet switch if the number of hosts behind NAT increases (and vice versa).

Multiple Receivers: If one host sends PC packets to different receivers and each receiver is associated with only a single protocol, no protocol switches occur and no bandwidth is limited on a direct way but in an indirect way: If the receiver is forced to be a distributed system, it has to implement a distributed coordination mechanism (sorting packets and extracting all hidden information on a single system that finally computes the whole hidden message). If the receiver-side network is monitored as well, the coordination itself must be covered too, and thus can probably be detected or at least raise attention. Also, the bandwidth is limited since multiple receivers can receive messages via different performing network access points and the network packets for the coordination can differ in their timings, too. Therefore, the sender must introduce pause intervals (which limit the bandwidth) between new packets to prevent a scrambled result at the receiver.

Redundancy: As all normalizer and firewall-like systems, our prototype of an active warden can result in a single point of failure if not operated on a redundant installation. Modifications of existing redundancy protocols (e.g., CARP) are thinkable to solve this problem. However, as any firewall-like system, the active warden introduces side-effects, i.e., delays, into network traffic.

End-User Limitation: As explained, the effect for end-users is low if the active warden is used. While an extensive end-user study was not part of this work, we measured different HTTP request-response times for 10MByte downloads over the active warden. The standard download time in our network was 0.41-0.57s. After we installed the active warden, we ran a HTTP download as well as a 0.25 bit/s PC to simulate a number of protocol switches as they occur for modern websites (multiple DNS requests for a whole site are normal since they can include script sources from other domains). Our simulation increased the download-time to 0.43-3s. We observed that the basic limitation for connections in the establishment phase where a new protocol (HTTP over TCP) occurred. In the context of the 4s-rule for website rendering [26], we can assume that our active warden is valuable for practical use-cases.

To summarize, all mentioned problems (excepting the network address translation) are solvable by adding the men-

 ISBN: 978-1-61208-201-1

tioned simple features. The configurable delay parameter d provides administrators a way to adjust the efficiency of the active warden to their requirements. Since only protocol switching packets are affected by the active warden, most of a network's traffic is not affected, i.e., download rates and upload rates will not decrease notably.

V. Conclusion

In this paper, we present the first active warden designed to counter both types of protocol switching covert channels: PC as well as PHCC. We limit the useful bandwidth of these covert channels by disturbing the protocol switches through synthetically introduced delays. Therefore, we implemented an active warden and verified its practical usefulness.

Future work will include to find solutions for the problem of network address translation inside a protected network as well as to find solutions for effects of large network environments where load balancing and redundancy protocols are required; the presented prototype was not designed for such environments. Additionally, research must be done to provide an exact bitrate controlling for PHCC using internal sequence numbers since we do only provide a loose bandwidth reduction for this channel type.

References

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[2] S. J. Murdoch, "Covert channel vulnerabilities in anonymity systems," Ph.D. dissertation, University of Cambridge, 2007.

[3] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday*, vol. 2, no. 5, May 1997, retrieved: Mar, 2012. [Online]. Available: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/528/449

[4] T. G. Handel and M. T. Sandford, II., "Hiding data in the osi network model," in *Proc. First Int. Workshop on Information Hiding*. London, UK: Springer-Verlag, 1996, pp. 23–38.

[5] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *ACM Conference on Computer and Communications Security*, V. Atluri, B. Pfitzmann, and P. D. McDaniel, Eds. ACM, 2004, pp. 178–187.

[6] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *ACSAC*, 2005, pp. 352–360.

[7] 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*, 1991, pp. 36–51.

[8] R. A. Kemmerer, "Shared resource matrix methodology: an approach to identifying storage and timing channels," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 256–277, 1983.

[9] J. McHugh, "An information flow tool for gypsy - an extended abstract revisited," in *Proc. 17th Annual Computer Security Applications Conference*, 2001, pp. 191–201.

[10] C. Krätzer and J. Dittmann, "Früherkennung von verdeckten Kanälen in VoIP-Kommunikation," in *IT-Frühwarnsysteme*, ser. BSI-Workshop. BSI, 2006, pp. 209–214, (In German).

[11] M. Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in *10th USENIX Security Symposium*, vol. 10, 2001, pp. 115–131.

[12] A. Singh, O. Nordström, A. L. M. dos Santos, and C. Lu, "Stateless model for the prevention of malicious communication channels," *Int. Journal of Comp. and Applications*, vol. 28, no. 3, pp. 285–297, 2006.

[13] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *USENIX Security Symp.*, 2008, pp. 139–154.

[14] S. Zander, G. Armitage, and P. Branch, "Covert channels and countermeasures in computer network protocols," *IEEE Comm. Magazine*, vol. 45, no. 12, pp. 136–142, Dec 2007.

[15] Daemon9, "Loki2 (the implementation)," *Phrack Magazine*, vol. 7, no. 5, September 1997, retrieved: Mar, 2012. [Online]. Available: http://www.phrack.org/issues.html?issue=51&id=6

[16] S. Wendzel and J. Keller, "Low-attention forwarding for mobile network covert channels," in *12th IFIP Comm. and Multim. Security*, ser. LNCS, 2011, vol. 7025, pp. 122–133.

[17] S. Wendzel, "Protocol hopping covert channels," *Hakin9*, vol. 08, no. 03, pp. 20–21, 2008, (in German).

[18] ——, "Protocol channels as a new design alternative of covert channels," *CoRR*, vol. abs/0809.1949, pp. 1–2, 2008.

[19] ——, "Analyse der Präventions- und Detektionsmethoden für verdeckte Kanäle," Master's thesis, Augsburg University of Applied Sciences, June 2011, (in German).

[20] C.-R. Tsai and V. D. Gligor, "A bandwidth computation model for covert storage channels and its applications," in *Proc. IEEE Conf. on Security and Privacy*, 1988, pp. 108–121.

[21] S. Wendzel, "pct," 2009, retrieved: Mar, 2012. [Online]. Available: http://www.wendzel.de/dr.org/files/Projects/pct/

[22] D. Berrange, "Simulating WAN network delay," 2005, retrieved: Mar, 2012. [Online]. Available: http://people.redhat.com/berrange/notes/network-delay.html

[23] J. Morris, "IPTables::IPv4::IPQueue module for Perl," 2002, retrieved: Mar, 2012. [Online]. Available: http://search.cpan.org/~jmorris/perlipq-1.25/IPQueue.pm

[24] C. D. Mee and E. D. Daniel, *Magnetic Storage Handbook*, 2nd ed. McGraw Hill, 1996.

[25] T. Kohno, A. Broido, and k. claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, no. 2, pp. 93–108, 2005.

[26] Akamai, "Retail web site performance," 2006, retrieved: Mar, 2012. [Online]. Available: http://www.-akamai.com/dl/reports/Site_Abandonment_Final_Report.pdf

 ISBN: 978-1-61208-201-1