# Covert Channels and Countermeasures in Computer Network Protocols

*Sebastian Zander and Grenville Armitage, University of Technology Melbourne*
*Philip Branch, Swinburne University of Technology*

*Editor's Note:* IEEE Communications Surveys and Tutorials *(http://www.comsoc.org/livepubs/ surveys/index.html) is an online sister publication that offers peer reviewed, archival quality surveys and tutorials for the benefit of society members. The editor in chief, Nelson da Fonseca, and his editorial board have chosen an outstanding article, published in third quarter 2007, to be reprinted and featured here (with the original format shortened for the magazine). It is hoped that this featured article will heighten interest in* IEEE Communications Surveys and Tutorials*.*

## ABSTRACT

Covert channels are used for the secret transfer of information. Encryption only protects communication from being decoded by unauthorized parties, whereas covert channels aim to hide the very existence of the communication. Initially, covert channels were identified as a security threat on monolithic systems such as mainframes. More recently, focus has shifted toward covert channels in computer network protocols. The huge amount of data and large number of different protocols in the Internet is ideal as a high-bandwidth vehicle for covert communication. This article provides an overview of the existing techniques for creating covert channels in widely deployed network protocols, and common methods for their detection, elimination, and capacity limitation.

## INTRODUCTION

Often it is thought that the use of encryption is sufficient to secure communication. However, encryption only prevents unauthorized parties from decoding the communication. In many cases the simple existence of communication or changes in communication patterns, such as an increased message frequency, are enough to raise suspicion and reveal the onset of events. Covert channels aim to hide the very existence of the communication. Typically, covert channels use means of communication not normally intended to be used for communication, making them quite elusive.

Lampson introduced covert channels in 1973 in the context of monolithic systems as a mechanism by which a process at a high security level leaks information to a process at a low security level that would otherwise not have access to it [1]. While a serious threat even for single hosts, the potential for covert channels in computer networks is greatly increased. In computer networks overt channels such as network protocols are used as carriers for covert channels [2, 3].

The huge amount of data and large number of different protocols in the Internet makes it ideal as a high-bandwidth vehicle for covert communications. The capacity of covert channels in computer networks has greatly increased because of new high-speed network technologies, and this trend is likely to continue. Even if only one bit per packet can be covertly transmitted, a large Internet site on a high-speed connection could leak a significant amount of data per year.

Covert channels in computer network protocols are similar to steganography, the hiding of information in audio, visual, or textual content. While steganography requires some form of content as cover, covert channels require some network protocol as carrier. The ubiquitous presence of a small number of network protocols suitable as carriers (e.g., Internet Protocol) make covert channels widely available. They are usable even in situations where steganography cannot be applied.

Many applications of covert channels are of a malicious or unwanted nature, and therefore pose a serious threat to network security. Furthermore, it is likely that because of increased measures against *open channels*, such as the free transfer of memory sticks in and out of organizations, the use of covert channels in computer networks will increase. Understanding existing covert channel techniques is crucial in developing countermeasures. The detection, elimination, and capacity limitation of covert channels are challenging, but need to be addressed to secure future computer networks.

This article provides an overview of existing

covert channels in network and application protocols and their countermeasures [4]. For space reasons we do not cover a number of related research areas: covert channels in single hosts, steganographic techniques for hiding information in content, subliminal channels in cryptosystems, and anonymous communications [5].

First we describe potential application scenarios, define the terminology, and explain the basic communication principles of covert channels. Then we present an overview of currently known covert channel techniques and their countermeasures. Finally, we conclude and identify future research directions.

## APPLICATION SCENARIOS

A diverse range of individuals and groups has found reason to utilize covert channels for communication and coordination [4]. Typically this is motivated by the existence of an adversarial relationship between two parties (e.g., government agencies vs. criminal or terrorist organizations, hackers or corporate spies vs. a company IT department, or dissenting citizens vs. their governments).

Clearly, government agencies, criminals, or terrorist organizations have an interest in keeping their communication secret. However, simply using encryption does not prevent adversaries from detecting communication patterns. Often only the evidence that communication takes place is sufficient to detect the onset of activity, discover organizational structures, or justify obtaining police warrants.

Once spies or hackers have compromised computer systems, they usually exfiltrate data or instrument the systems for malicious purposes. Such activities generate network traffic that — if not covert — would immediately alert system administrators, who would then discover the compromised systems.

Computer viruses or worms can use covert channels to spread themselves undetected or covertly exchange information necessary for distributed processing (e.g., executing brute force attacks on cryptosystems).

Recent attempts by some governments to limit the freedom of speech in the Internet have led to proposals for using covert channels to circumvent these measures [4]. In countries that forbid strong encryption of data, covert channels can be used to *secure* the information transport (although this is not strong security in the cryptographic sense).

Network administrators can use covert channels to secure network management related communication by hiding it from hackers. Often even ordinary employees may want to use covert channels to simply bypass their company firewalls in order to access Internet resources.

The field of anonymous communications is concerned with obfuscating sender/receiver identities and their relationships (who is communicating with whom). Imperfections in these techniques are effectively covert channels usable to thwart anonymization [4].

## TERMINOLOGY AND COMMUNICATION MODEL

### TERMINOLOGY

Different terms have been used to describe the process of hiding information in network protocols. Whereas many researchers referred to covert channels, some also used the terms network steganography or information hiding [6]. Throughout this article we use the term *covert channel* when we refer to the hiding of information in network protocols, and refer to the information transmitted across the covert channel as *covert information*. Transmission of information through a system mechanism is an *overt channel*.

Traditionally covert channels were classified into storage and timing channels [7]:
• *Storage channels* involve the direct/indirect writing of object values by the sender and the direct/indirect reading of the object values by the receiver.
• *Timing channels* involve the sender signaling information by modulating the use of resources (e.g., CPU usage) over time such that the receiver can observe it and decode the information.

### THE PRISONER PROBLEM

The prisoner problem was first posed by Simmons and is the de facto model for covert channel communication [8]. Two people, Alice and Bob, are thrown into prison and intend to escape. To agree on an escape plan they need to communicate, but Wendy the warden monitors all their messages. If Wendy finds any signs of suspicious messages, she will place Alice and Bob in solitary confinement — making it impossible for them to escape. Alice and Bob must exchange innocuous messages containing hidden information that (hopefully) Wendy will not notice. There are three types of wardens [9]:
• A *passive* warden can only spy on the channel, but cannot alter any messages.
• An *active* warden is able to slightly modify the messages, but without altering the semantic context.
• A *malicious* warden may alter the messages with impunity. However, malicious wardens are rare.

Extending this scenario to computer networks, Alice and Bob use two networked computers for communication [3]. They run an innocuous overt communication channel between their computers containing a covert channel. Alice and Bob share a secret, which is useful for determining covert channel encoding parameters and encrypting/authenticating covert messages. For practical purposes Alice and Bob may well be the same person, for example, a hacker exfiltrating restricted information. Wendy manages the network and can monitor the passing traffic for covert channels or alter the passing traffic to eliminate or disrupt covert channels. Figure 1 depicts the model (Alice sending to Bob).

### COMMUNICATION SCENARIOS

There are a number of different scenarios for covert communication depending on whether Alice and Bob are the sender and receiver of the

**■ Figure 1.** *The prisoner problem: the de facto model for covert channel communication.*



**■ Figure 2.** *Possible combinations of different covert sender/receiver locations.*

overt channel, or they act as *middlemen* and manipulate an overt channel between innocent users [10].

If the sender of the covert channel is also the sender of the overt channel, it can manipulate the overt channel as desired (e.g., maximize the channel capacity). However, sometimes the covert sender may not be able to create overt channels or may choose not to do so for improved stealth. In this case the sender can act as middleman by embedding a covert channel into an existing overt channel. Obviously, in this situation the covert sender has no control of the overt channel, and the capacity of the covert channel depends on the existing overt channel.

The covert receiver can be the receiver of the overt channel, but to improve stealth the receiver can also be a middleman extracting the covert information from an overt communication destined for an innocent receiver. Then the covert receiver should (if possible) remove the covert channel, preventing possible detection by the real receiver or any other intermediate nodes.

Being a middleman does not necessarily mean that the covert sender/receiver has to be physically separated from the overt sender/receiver. The covert sender and receiver could be located on routers/gateways between the overt sender and receiver, but they could also be on the same physical device located in lower levels of the network protocol stack. Figure 2 illustrates the possible combinations of covert sender and receiver locations.

## COVERT CHANNELS

In this section we give an overview of existing covert channel techniques. We present only a few examples of different covert channel types. The interested reader is referred to a comprehensive survey in [4].

### UNUSED HEADER BITS

Covert channels can be encoded in unused or reserved bits of protocol headers. This is particularly problematic if protocol standards do not mandate specific values or receivers do not check for the standard values. For example, covert data can be transmitted in unused bits of the IP header's type of service field (Fig. 3).

Even if header fields are only unused in particular situations, they can be used to carry covert data. For example, the Don't Fragment (DF) bit in the IP header's flags field (Fig. 3) can be set to arbitrary values if the sender knows the maximum transfer unit (MTU) size allowed of packets along the path to the receiver and only sends packets of less than MTU size.

Padding bits provide another opportunity to encode covert data. For example, Ethernet frames must be padded to a minimum length of 60 bytes. If the protocol standard does not enforce specific values for the padding, any data can be used [3].
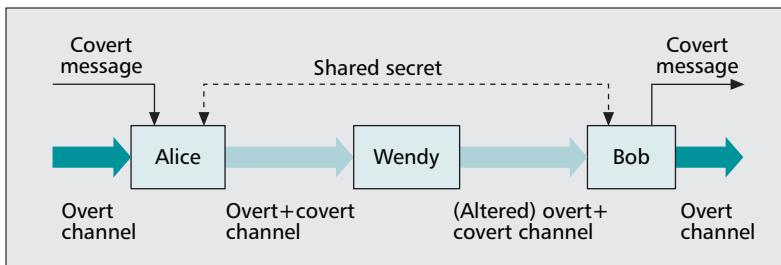
### OPTIONAL HEADER FIELDS

Many protocols support extension of the standard header. Usually there are some predefined header extensions that allow transport of nonmandatory information on demand, but many protocols also allow header extensions to carry data not foreseen in the original specification, thus extending the capabilities of the protocol. Covert information can be encoded in existing or new header extensions. For example, covert data can be masked as IP addresses in IP route record option headers [4]. Another approach is to use the presence or absence of optional header fields as a covert channel.
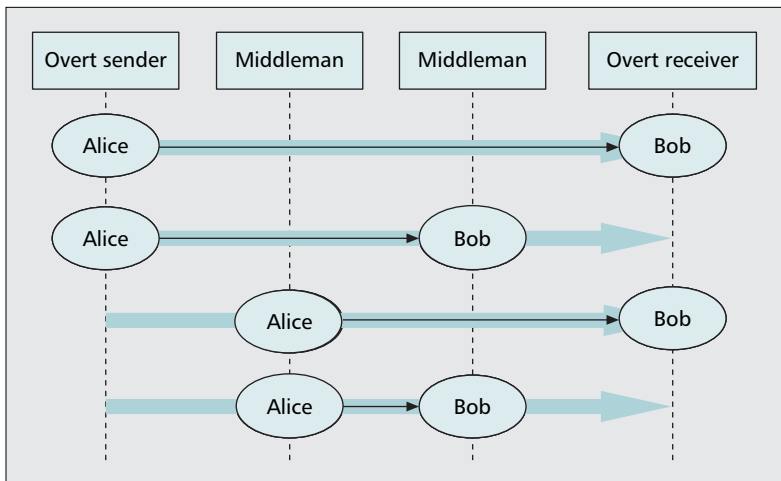
### SEMANTIC OVERLOADING OF HEADER FIELDS

Covert channels may involve syntactic variations of the overt channel that are semantically identical. For example, it is possible to encode covert data in TCP sequence numbers, used to coordinate which data has been transmitted and received (Fig. 4). The first sequence number selected by a client is called the initial sequence number (ISN). The ISN must be chosen so that the sequence numbers of new incarnations of a TCP connection do not overlap with the sequence numbers of earlier incarnations. A very simple covert channel is created by directly encoding covert data in the most significant byte of each ISN and setting the three remaining bytes to zero [4].

However, such simple encoding schemes result in header field value distributions that differ from the distributions of real protocol implementations. To avoid detection more complex encodings have been proposed [11].

Higher layer protocols, especially text-based protocols, offer even more opportunities. For

example, covert data can be encoded into Hypertext Transfer Protocol (HTTP) header fields by varying the use of lower or upper case, or varying the number of white spaces between words.

### MODULATING HEADER FIELDS

Covert data can be encoded in destination address fields or by modulating the order of valid destination addresses in subsequent transmissions. Examples of usable address fields include the destination IP address or UDP/TCP destination port number. Covert information can also be transmitted in source addresses if they can be modulated. This is the case for IP addresses (if spoofing is possible) or source port numbers.

Many communication protocols, such as IP and TCP, have fields to indicate the length of headers, header extensions, or messages. It is possible to convey covert data by modulating the length of messages or headers. Instead of freely modulating the length, a more skilled covert sender could choose from a set of lengths resembling realistic messages [4].

Covert information can be inserted into the low order bits of timestamp fields. For example, covert data can be encoded into TCP timestamp header options [4]. Instead of directly modifying timestamps, the proposed algorithm slows the TCP stream so that the timestamps on packets are valid when they are sent. The algorithm compares the least significant bit (LSB) of every TCP timestamp generated by the system with the current covert bit to be sent. If the LSB matches the covert bit, the TCP segment is sent immediately. Otherwise it is delayed for one timestamp tick.

### PACKET AND MESSAGE SEQUENCE TIMING

Covert information can be transmitted by varying packet rates, which is equivalent to varying interpacket times [4]. The covert sender varies its packet rate each time interval. The receiver measures the rate in each time interval and decodes the covert information. An on/off channel is a special case where one rate is zero and the other rate is chosen based on the situation. Packet timing channels require a mechanism to synchronize time intervals at the sender and receiver.

However, a variant exists that does not require synchronization, because the covert information is encoded in the interpacket delays of consecutive packets [4]. This covert channel makes it possible to leak information from a high-security host to a low-security host in the timing of acknowledgments necessary for reliable communication [12].

Similar to packet timing, the timing of protocol operations can be modulated. For example, a receiving station can acknowledge each message separately or wait until two messages have arrived before acknowledging the first.

### PACKET AND HEADER FIELD ORDERING

It is possible to implement a covert channel through packet (re)ordering [4]. Because a set of $n$ packets can be arranged in any $n$ ways, a maximum of $\log_2 n!$ bits can be transmitted. This approach requires per-packet sequence numbers
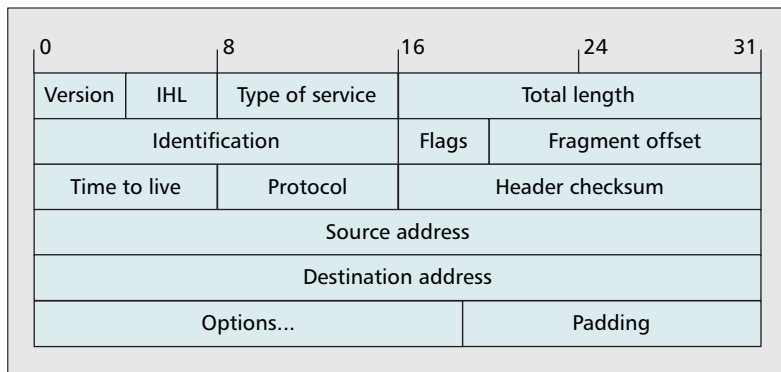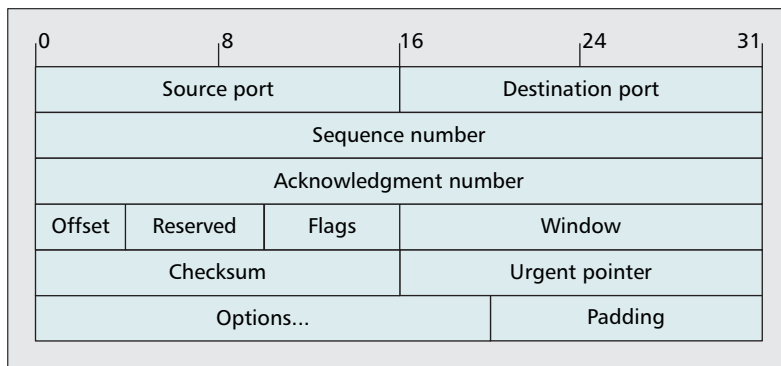


■ **Figure 3.** *IP header structure.*



■ **Figure 4.** *TCP header structure.*

to determine the original order, which are present in a number of protocols, including TCP.

In protocols where the order of header fields is not fixed, the same technique can be applied to (re)order header fields. For example, covert information can be encoded in the order of HTTP header fields.

### COLLISIONS, LOSS, AND RETRANSMISSIONS

A collision-based covert channel can be created in shared medium access protocols (e.g., Ethernet, wireless LAN). The covert sender jams any attempts by another user to transmit, and then backs off of either the minimum or maximum amount of time. Then the covert sender's frame will either lead or lag frames sent by the other user, and the receiver can recover the covert information by analyzing the order of frame arrivals. Some recent shared medium access protocols use splitting algorithms to divide the set of collided senders into smaller subsets, and then these subsets retransmit in order. Covert information can be transmitted by manipulating the collision resolution procedure [4].

If a protocol has per-packet sequence numbers, packet loss can be used as a low-rate covert channel. The covert sender transmits information by artificially losing packets (skipping sequence numbers). Another approach in wireless LANs with variable bit error rate is for a sender to inject synthetic *corrupt* frames (frames with deliberately incorrect checksums). All stations that are part of the covert channel communicate via sending some percentage of their frames with intentionally created bad checksums.
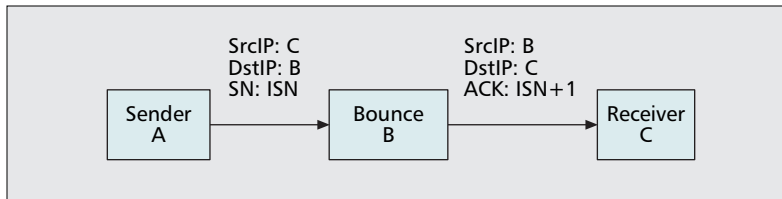
**■ Figure 5.** *TCP initial sequence number (ISN) bounce channel.*

Another option is to transmit covert information through spurious *retransmission* of frames. The covert sender encodes covert data by duplicating specific frames, and the covert receiver decodes the covert information by detecting the duplications.

### INDIRECT CHANNELS

The covert sender does not need to directly communicate with the covert receiver, but can exchange messages through *innocent* intermediate nodes. There are indirect storage and timing channels.

Bounce channels are indirect storage channels that presume the covert sender is able to spoof IP source addresses [4]. For example, the ISN channel described earlier can be realized as a bounce channel (Fig. 5). The covert sender sends a packet with an ISN to a bounce host with a spoofed IP source address set to the intended destination. Upon receipt of the packet, the bounce host sends an acknowledgment or reset packet to the receiver with the acknowledged sequence number equal to the ISN+1. The receiver decodes the covert data from the acknowledged sequence number.

An indirect timing channel can be realized by using an innocent server as intermediate. The covert sender sends a large number of requests to the server or stays silent in each time interval, equivalent to one bit per time interval. The receiver periodically probes the server and measures the response time to recover the covert information. A more sophisticated version of this channel does not measure response times, but the servers change in clock skew. Clock skew changes with changing temperature, which in turn changes with varying CPU load [4].

### PAYLOAD TUNNELING

This type of covert channel tunnels one protocol in the payload of another protocol. The main purpose is circumventing firewalls that limit outgoing traffic to a few allowed application protocols (e.g., HTTP).

The first available tools were based on tunneling IP over the Internet Control Message Protocol (ICMP). Today there are also many tools for tunneling data over HTTP [4]. Even the Domain Name System (DNS) protocol can be used as tunnel. The client issues DNS resolution requests for names of the form xxxxx.example.com, where the DNS server for example.com has been modified to participate in the covert channel. The hostname (xxxxx) is encoded covert data. The DNS server sends data back to the client as text records in the DNS responses.

## COVERT CHANNEL COUNTERMEASURES

Before any action can be taken against a covert channel, it needs to be identified. The identification of a covert channel can be ad hoc or based on a formal method. A number of formal methods have been developed for identifying covert channels in single host systems [13]. The few proposed formal techniques for identifying covert channels in network protocols are mostly based on an adaptation of one of these techniques [4].

There are two causes of covert channels: design oversights and weaknesses inherent in the system design. While covert channels caused by oversights may be corrected once discovered, those intrinsic to the system can never be removed without redesigning the system. Therefore, ideally covert channels should be identified and removed during the design phase.

If a covert channel was not removed in the design phase, the next best option is to eliminate its possible use. However, this may lead to very inefficient systems, as covert channels can often only be completely removed by replacing automated procedures with manual procedures. Furthermore, covert channels based on the modulation of visible message parameters are inherent in distributed systems, such as computer networks. Therefore, it seems that covert channels cannot all be completely eliminated [14].

If a channel cannot be eliminated, its capacity should be reduced. What capacity is acceptable depends on the amount of information leakage that is critical. For example, if the channel is so small that classified information cannot be leaked before it is outdated, the channel capacity is tolerable. Limiting the channel capacity is often problematic, because it means slowing down system mechanisms or introducing noise, which both limit the performance of the system.

Any covert channels that cannot be removed should be audited. Auditing acts as deterrence to possible users of the channel. Covert channels with capacities too low to be significant or that cannot be audited should at least be documented (e.g., in the protocol specification) so that everybody is aware of their existence and potential threat.

### ELIMINATING COVERT CHANNELS

Host security cannot remove covert network channels, but it can prevent their exploitation in some scenarios. If hosts are secured from being hacked, hackers cannot exploit covert channels. However, this does not protect against data exfiltration by insiders; nor does it solve the covert channel problem in other scenarios such as censorship circumvention.

One approach to counter tunneling channels is to block protocols that are susceptible to covert channels. For example, ICMP is blocked by many firewalls these days, rendering ICMP tunnels ineffective. Obviously, in the Internet some protocols cannot be blocked because they are vital (e.g., IP, TCP) or their services are too important (e.g., HTTP). However, in a closed

network protocols prone to covert channels could be blocked, or replaced by versions with fewer or limited covert channels.

The leakage of classified information from a high-security system to a low-security system (the classic covert channel) is prevented by a network design where only hosts on the same security level are allowed to communicate. Such an approach may be practical for highly secure networks, but not for large open networks such as the Internet.

Many of the simple storage covert channels are eliminated by normalizing protocol headers, padding, and header extensions [15]. For example, unused or reserved bits and padding are set to zero, and unknown header extensions are removed. Other header fields can be normalized or rewritten under certain assumptions [4]. Preventing the spoofing of IP source addresses effectively eliminates bouncing channels.

### LIMITING COVERT CHANNEL CAPACITY

One way to counter the address modulation channel is limiting the number of possible addresses, effectively limiting the allowed host-to-host connections. Instead of limiting the inter-actions between hosts, sending dummy packets between random hosts inserts noise into the traffic patterns, but reduces efficiency. Padding all packets to a common size eliminates the packet length modulation channel, and a small number of possible packet sizes could be allowed to increase efficiency. While these techniques may be used in closed networks, their wider application in the Internet seems unlikely.

Introducing random noise to mask the covert channel or fixing packet/message rates reduces the capacity of packet and message sequence timing channels. Noise can be introduced by buffering and delaying packets/messages, or making system clocks fuzzy, preventing senders from exactly timing outgoing packets and receivers from accurately measuring the timing [4]. Link padding forces a packet flow to adhere to a specific packet rate and injects dummy packets if necessary (for preventing on/off channels) [16]. Allowing senders to emit a small number of different packet rates increases efficiency while limiting channel capacity to acceptable levels.

A number of techniques have been proposed specifically to limit the capacity of covert channels in the timing of acknowledgments [4, 12].

### AUDITING COVERT CHANNELS

All proposed detection methods are based on the detection of nonstandard or abnormal behavior. The assumption is that the warden knows the normal behavior of protocols and hosts, and can detect abnormal behavior caused by covert channels. However, it is difficult to detect covert channels if there is considerable variation in normal behavior. Furthermore, any covert channel that appears identical to normal use of the protocol will be difficult to detect.

All covert channels based on nonstandard use of the protocol are easy to detect. For example, most protocol standards mandate that unused or reserved bits and padding must be filled with specific values (e.g., zeros); therefore, any covert channels in these bits will be obvious. Other covert channels exploit the fact that header fields can have *arbitrary* values within the requirements of the standard. However, if the fields are naively used and the resulting value distributions are different from the distributions generated by real operating systems, detecting covert channels is not very difficult [11].

Different methods have been proposed to detect packet timing channels [4]. Some researchers have proposed auditing the change of traffic rate over time. If the traffic rate of one host changes by more than a certain threshold, this would indicate a covert channel. Other researchers have proposed detecting these channels based on the packet interarrival time distribution. For covert channels the distribution will have two or more distinct spikes, whereas for normal flows interarrival times are more evenly spread.

A number of solutions have been proposed for detecting payload tunneling channels [4]. In this case the behavior is defined by per-flow metrics such as the average packet size, ratio of small and large packets, number of packets sent/received, and duration of a connection.

## CONCLUSIONS AND FUTURE WORK

Many existing covert channels in network protocols follow the *security through obscurity* approach, and in principle their detection or elimination is straightforward. However, while researchers have developed a number of countermeasures, real world experience shows that industry products still lack methods to deal with covert channels. Also, some proposed countermeasures could significantly reduce overt channel performance; therefore, their applicability in real high-speed networks is questionable. There are many possibilities of creating covert channels in computer network protocols, and the complete elimination of all these channels seems quite challenging.

A number of directions are left for further study. Currently, a comprehensive taxonomy for classifying the different covert channels and countermeasures is missing. Additionally, there is a lack of work on capacity estimation in the presence of channel errors, formal methods for identifying covert channels during protocol design, and more holistic approaches to covert channel elimination/detection and their integration into existing network security management methods. Finally, it seems likely that the arms race of developing new covert channels with improved stealth and capacity, and developing more effective detection and elimination techniques will continue.

> *While researchers have developed a number of countermeasures, real world experience shows that industry products still lack methods to deal with covert channels. Also, some proposed countermeasures could significantly reduce overt channel performance.*

### REFERENCES

[1] B. Lampson, "A Note on the Confinement Problem," *Commun. ACM*, vol. 16, Oct. 1973, pp. 613–15.
[2] C. G. Girling, "Covert Channels in LAN's," *IEEE Trans. Software Eng.*, vol. SE-13, Feb. 1987, pp. 292–96.
[3] T. Handel and M. Sandford, "Hiding Data in the OSI Network Model," *Proc. 1st Int'l. Wksp. Info. Hiding*, 1996, pp. 23–38.
[4] S. Zander, G. Armitage, and P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," accepted for publication, *IEEE Commun. Surveys and Tutorials*, vol. 9, no. 3, Oct. 2007, pp. 44–57.

[5] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding — A Survey," *Proc. IEEE*, Special Issue on Protection of Multimedia Content, vol. 87, July 1999, pp. 1062–78.

[6] J. Millen, "20 Years of Covert Channel Modeling and Analysis," *Proc. IEEE Symp. Sec. and Privacy*, May 1999, pp. 113–14.

[7] U.S. DoD, "Trusted Computer System Evaluation Criteria," Tech. Rep. DOD 5200.28-STD, Dec. 1985.

[8] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," *Proc. Advances in Cryptology*, 1983, pp. 51–67.

[9] S. Craver, "On Public-Key Steganography in the Presence of an Active Warden," *Proc. 2nd Int'l. Wksp. Info. Hiding*, Apr. 1998, pp. 355–68.

[10] N. Lucena *et al.*, "Syntax and Semantics-Preserving Application-Layer Protocol Steganography," *Proc. 6th Info. Hiding Wksp.*, May 2004.

[11] S. J. Murdoch and S. Lewis, "Embedding Covert Channels into TCP/IP," *Proc. 7th Info. Hiding Wksp.*, June 2005.

[12] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The Pump: A Decade of Covert Fun," *21st Annual Comp. Sec. Apps. Conf.*, Dec. 2005, pp. 352–60.

[13] V. Gligor, "A Guide to Understanding Covert Channel Analysis of Trusted Systems," Tech. Rep. NCSC-TG-030, Nat'l. Comp. Sec. Ctr., Nov. 1993.

[14] I. S. Moskowitz, M. H. Kang, "Covert Channels — Here to Stay?," *Proc. 9th Annual Conf. Comp. Assurance*, 1994, pp. 235–44.

[15] M. Handley, C. Kreibich, and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics," *Proc. 10th USENIX Sec. Symp.*, Aug. 2001.

[16] B. R. Venkatraman and R. E. Newman-Wolfe, "Transmission Schedules to Prevent Traffic Analysis," *Proc. 9th Annual Comp. Sec. and Apps. Conf.*, Dec. 1993, pp. 108–15.

## BIOGRAPHIES

SEBASTIAN ZANDER [S'06] (szander@swin.edu.au) received his Dipl.-Ing. in technical informatics from the Technical University of Berlin, Germany, in 1999. Since 2006 he has been a research student at the Centre for Advanced Internet Architectures at Swinburne University of Technology, Melbourne, Australia, and his research topic is covert channels in computer network protocols. Previously he worked as a scientist at Fraunhofer FOKUS, Berlin, and as a research fellow at Swinburne University.

GRENVILLE ARMITAGE [M'03] (garmitage@swin.edu.au) earned a B.Eng. (Elec) (Hons) in 1988 and a Ph.D. in electronic engineering in 1994, both from the University of Melbourne, Australia. Since 2002 he has been an associate professor of telecommunications engineering and director of the Centre for Advanced Internet Architectures at Swinburne University of Technology. He authored *Quality of Service In IP Networks: Foundations for a Multi-Service Internet* (Macmillan, April 2000) and co-authored *Networking and Online Games — Understanding and Engineering Multiplayer Internet Games* (Wiley, April 2006). He is also a member of ACM and ACM SIGCOMM.

PHILIP BRANCH [M'95] (pbranch@swin.edu.au) received a Ph.D. in engineering from Monash University, Victoria, Australia, in 2000. Since 2003 he has been a senior lecturer in telecommunications engineering at Swinburne University of Technology, conducting research within the Centre for Advanced Internet Architectures. His research interests are in game traffic, network security, and lawful interception. He is a co-author of *Networking and Online Games — Understanding and Engineering Multiplayer Internet Games* (Wiley, April 2006).