Novel Approaches for Network Covert Storage Channels

Dissertation

zur Erlangung des akademischen Grades

DOKTOR RER. NAT.

der Fakultät für

Mathematik und Informatik

der FernUniversität

in Hagen

von

Steffen Wendzel
geb. in Aschersleben

Betreuer:
Prof. Dr. Jörg Keller

Hagen, 2013

Erster Gutachter: Herr Prof. Dr. Jörg Keller (FernUniversität in Hagen)

Zweiter Gutachter: Herr Prof. Dr. Felix Freiling (Friedrich-Alexander-Universität Erlangen-Nürnberg)

Vorsitzender der Promotionskomission: Herr Prof. Dr. Wolfram Schiffmann

Protokollant: Herr Dr.-Ing. Hauke Coltzau

Tag der mündlichen Prüfung: 7. Mai 2013

Abstract

Since the late 1980's a large number of techniques to embed covert channels into network protocols were discovered. Covert channels enable a policy-breaking communication while they are additionally hard to detect. While it must be considered non-trivial to counter covert channels in networks, it can be considered trivial to evaluate network protocols in order to find possible ways to embed hidden information in these protocols. This thesis therefore, does not aim on presenting new covert channels in network protocols (except from exemplary channels in BACnet).

Today, covert channels are a useful technique for the development of botnets since these channels can make botnet traffic hard to detect. For this reason, it is an attractive goal for botnet developers to enhance existing covert channel techniques. As this gives leeway for the introduction of additional features into covert channels and enhancement of their invisibility.
Therefore, the research community must also aim on improving covert channels since it would otherwise be unfeasible to find means to counter such novel techniques introduced by botnet developers.
On the other hand, covert channels must be considered as dual-use betterment as they, for instance, can enable journalists to transfer illicit information in networks with censorship without facing detection.

Within the last decade, new covert channels with internal control protocols (so called *micro protocols*) arose. These micro protocols are placed in the hidden data of the channel and can be considered a powerful technique as they introduce new features such as dynamic routing or reliability. In general, micro protocols control a covert channel but their purpose depends on its given utilization. For instance, a micro protocol used within a botnet could signal a botnet command, such as, to send a Spam mail while the actual hidden payload can comprise a fragment of the Spam message to be sent.

This thesis is the first to discuss the need for improved micro protocol designs as the detectability of a covert channel highly depends on the used micro protocol: If a micro protocol causes anomalies, the detection of a covert channel raises.

The **first part (Chapters 3 and 4)** of this thesis introduces two approaches for the design and development of micro protocols. The first approach decreases the size of a micro protocol header to minimize the number of bits to be modified in a network packet — if less bits are required to be modified by the covert channel, the channel will cause fewer anomalies. The second approach ensures the conformity of the micro protocol to the utilized network protocol: If the micro protocol does not violate rules of the utilized protocol, it will also cause less anomalies. Therefore, the existing covert channel terminology is extended.

The initial connection establishment phase (NEL phase) of network covert channels is enhanced by using these micro protocols and it helps overcoming the two-army problem initially discovered in this thesis.

Covert channels (with or without micro protocols) can utilize various network protocols simultaneously. We call the family of such covert channels *protocol switching covert channels*. A problem with these channels is the lack of a means to limit their bitrate. This thesis presents the first approach to limit the maximum error-free bitrate of protocol switching covert channels. The approach has been evaluated and can be considered to be applicable in practice.

The **second part (Chapter 5)** of this thesis discusses the presence of covert and side channels in building automation systems. Their potential for adversaries, which lies in the observation of events and persons in buildings. And finally, in building automation-based data exfiltration to bypass the protection means of a (better protected) enterprise network.

A distinction of such covert channels into high-level covert channels (based on the interaction with the building) and low-level covert channels (based on the utilization of building automation network protocols) is proposed. Furthermore, a prevention means to counter high-level covert (and side) channels in building automation systems as well as a prevention technique for BACnet-based covert channels is also presented and evaluated.

Zusammenfassung

Ende der 80er Jahre wurden die ersten verdeckten Kanäle (covert channels) für Netzwerke vorgestellt. Dabei handelt es sich um Kanäle, die eine Policy-brechende Kommunikation realisieren und deren Detektion und Verhinderung als schwierig zu betrachten ist. Daten verdeckter Kanäle werden dabei meist in ungenutzten Bereichen von Netzwerkpaketen untergebracht. In den beiden folgenden Jahrzehnten erschienen diverse Arbeiten, die weitere verdeckte Kanäle – insbesondere für TCP/IP – aufzeigen konnten. Durch diese vorhergehenden Arbeiten kann es heute als einfach betrachtet werden, neue verdeckte Kanäle in weiteren Netzwerkprotokollen zu finden. Gegenstand dieser Dissertation ist deshalb nicht die Vorstellung neuer verdeckter Kanäle innerhalb von Netzwerkprotokollen (mit exemplarischer Ausnahme von BACnet), sondern die generelle Verbesserung und Unterbindung derselben.

Heute finden verdeckte Kanäle insbesondere bei Botnetzen Anwendung. Mithilfe dieser Technologie ist es Botnetzen möglich, unentdeckt zu kommunizieren und Angriffe zu koordinieren. Die Weiterentwicklung verdeckter Kanäle seitens der Angreifer (etwa Botnetzentwickler) kann primär das Ziel verfolgen, diese Kanäle funktionsreicher und schwieriger detektierbar zu gestalten. Aus diesem Grund muss die Forschung ebenfalls das Ziel verfolgen, verdeckte Kanäle zu verbessern. Nur wenn dies gelingt, können rechtzeitig Gegenmaßnahmen entwickelt werden, um verdeckte Kanäle zu verhindern. Gleichzeitig sind verdeckte Kanäle als Dual-Use-Gut zu betrachten und können etwa Journalisten eine schwer detektierbare Übertragung regimekritischer Informationen in zensierten Netzwerken ermöglichen.

Ein neuerer Ansatz, verdeckte Kanäle zu verbessern, besteht in der Einbettung von Steuerprotokollen (sog. *Mikroprotokollen*). Dabei handelt es sich um Protokolle, deren Headerbereiche in den versteckten Daten des Kanals untergebracht werden. Mikroprotokolle erweitern einen verdeckten Kanal um Features wie Reliability, Kompatibilität zu alten Protokollversionen oder dynamisches Routing. Im angesprochenen Kontext von Botnetzen können Mikroprotokolle nicht nur den verdeckten Kanal steuern, sondern auch Befehle für einen Bot beinhalten (etwa einen Befehl zum Versenden einer Spam-

Nachricht) während der eigentliche (ebenfalls versteckt übertragene) Payload (etwa ein Fragment einer Spam-Mail) abhängig vom jeweiligen Befehl interpretiert wird.

Diese Arbeit zeigt erstmals auf, dass Mikroprotokolle besondere Ansprüche hinsichtlich ihrer Verdecktheit erfüllen müssen – eine Anforderung, die bei anderen Netzwerkprotokollen nicht gegeben ist. Je mehr Anomalien ein Mikroprotokoll im Netzwerkverkehr verursacht, umso wahrscheinlicher ist die Detektion eines verdeckten Kanals.

Im **ersten Teil (Kapitel 3 und 4)** dieser Arbeit werden erstmals Ansätze für die Entwicklung solcher Mikroprotokolle vorgestellt. Zum einen wird dabei die Größe des Mikroprotokolls minimiert: Durch eine geringere Protokollgröße müssen entsprechend weniger Bits im ausgenutzten Netzwerkprotokoll manipuliert werden, wodurch weniger Anomalien im Netzwerkverkehr entstehen. Zum anderen wird ein Verfahren vorgestellt, dass die Konformität eines Mikroprotokolls zum ausgenutzten Netzwerkprotokoll sicherstellt und dadurch ebenfalls Anomalien verhindert. Für die exakte Auseinandersetzung mit der Thematik wird die bestehende Terminologie des Forschungsgebietes verfeinert.

Ebenfalls erst einige Jahre alt ist die Idee, verdeckte Kanäle adaptiv zu gestalten, sie also automatisch an sich ändernde Netzwerkumgebungen anzupassen. In diesem Kontext stellt die vorliegende Arbeit ein Zwei-Armeen-Problem in der Initialisierungsphase von verdeckten Kanälen vor, der so genannten *Network Environment Learning Phase*. Für dieses Problem werden Lösungsvorschläge diskutiert.

Verdeckte Kanäle mit Mikroprotokollen können Verbindungen über mehrere Kanäle hinweg simultan realisieren. Solche Kanäle, die ihr Übertragungsprotokoll transparent wechseln können, nennen sich *Protocol Hopping Covert Channels*. Die Bitraten-Limitierung dieser Protocol Hopping Covert Channels und der verwandten *Protocol Channels* (die versteckte Informationen durch die Verwendung bestimmter Protokolle signalisieren) war bisher nicht möglich. Diese Dissertation stellt erstmals ein Verfahren zur Limitierung beider Kanaltypen vor und evaluiert dessen Effizienz und Nutzen.

Der **zweite Teil (Kapitel 5)** behandelt erstmals verdeckte Kanäle und Seitenkanäle in der Gebäudeautomation. Dabei wird das Schadpotential solcher Kanäle betrachtet. Dieses Schadpotential liegt insbesondere in der Überwachung von Personen und im Umgehen von Sicherheitsmechanismen der TCP/IP-Netze einer Organisation.
Es wird zudem gezeigt, dass zwei verschiedene Arten von verdeckten Kanälen in der Gebäudeautomation existieren: High- und Low-Level-Kanäle. Erstere werden über die Interaktion mit den Sensoren und Aktoren im Gebäude realisiert, letztere sind typische

Netzwerkkanäle, die insbesondere ungenutzte Bereiche in Netzwerkpaketen ausnutzen. Weiterhin werden verschiedene Methoden zur Vermeidung von High- und Low-Level-Kanälen präsentiert und analysiert.

Acknowledgements

I am very grateful for the guidance of my advisor *Jörg Keller* within the last 3.5 years. He always took time for providing a detailed feedback on my paper and thesis drafts and – based on approx. 700 sent and received emails and telephone calls – demonstrated that excellent distance-based guidance is feasible.

I am furthermore thankful for the guidance of *Felix Freiling*. He pointed out various aspects of this thesis that I would probably not have taken into account without him and that helped to make this thesis much more valuable.

I especially thank *Arnulf Deinzer* for believing in my capabilities.

I also thank *Maryna Krotofil* for the time she invested in listening to my defense talk countless times while providing me with feedback of outstanding quality at a time at which she was already very busy with her own research work. Her willingness to create high-quality work is a contribution to my own motivation.

I like to thank *Thomas Rist* for sharing his scientific experience with me and for allowing me to combine the topic of my thesis with the IT4SE research project.

Moreover, I thank *Sebastian Zander* and *Sebastian Schinzel* for the highly interesting discussions on covert and side channels, *Peter Backs* for his great work on the status update-based realization of dynamic routing, *Daniel Stødle* who helped me to understand the details of his tool *Ping Tunnel*, and *Benjamin Kahler* for his work with the BACnet Firewall Router.

I thank *Christian Herdin*, *Andreas Gärtner*, and *Michael Schimmel* for joining my occasional late night work from time to time.

In particular, I would like to thank my parents and family members for their support over this long period and dedicate this thesis to them.

Augsburg, May 2013

Publications and Previous Work

The following publications were already published in the context of this dissertation. Contributions by other authors are mentioned in this section. If a contribution of another author was integrated within the chapters of this Ph.D. thesis, an own paper was explicitly cited and, in case of a solely contribution, the contributor was namely mentioned.

Books:

- S. Wendzel: *Tunnel und verdeckte Kanäle im Netz*, Springer-Vieweg, October 2012. *(in German)*

  This is the first book that provides a detailed introduction to the field of network covert storage channels in combination with network tunneling. Chapter 7 of the book is based on the results of this thesis, and Chapters 4.1-4.5 provide an overview of covert storage channel techniques for network environments that form a part of the related work in Chapter 2 of this thesis.
  The Chapters 4.6-4.7, 5, and 6 are based on the author's diploma and master's theses and it was necessary to explain these basics again in Chapter 2 of this thesis. All other Chapters (1-3) deal with network basics or network protocol tunneling.

Book Chapters:

- S. Wendzel, J. Keller: *Einführung in die Forschungsthematik der verdeckten Kanäle*, In: J. Sambleben and S. Schumacher (Eds.): Informationstechnologie und Sicherheitspolitik – Wird der dritte Weltkrieg im Internet ausgetragen?, Magdeburger Institut für Sicherheitsforschung, BoD, pp. 91-102, October 2012 (in German).

  This chapter provides an introduction to the covert channel research area. It was written by Wendzel and edited by Keller.

Journal Articles:

- S. Wendzel, J. Keller: *Preventing Protocol Switching Covert Channels*, International Journal On Advances in Security, Vol. 5, no. 3&4, pp. 81-93, IARIA, 2012.

  This article enhances the active warden introduced in the paper "Design and Implementation of an Active Warden Addressing Protocol Switching Covert Channels" published in the proceedings of the 7th International Conference on Internet Monitoring and Protection (see below). Keller contributed the discussion of new codings for protocol channels and their use in the context of the active warden and Wendzel contributed the discussion of a randomized delay and formal grammar enhancements of the active warden.

In Proceedings:

- P. Backs, S. Wendzel, J. Keller: *Dynamic Routing in Covert Channel Overlays Based on Control Protocols*, in Proc. International Workshop on Information Security, Theory and Practice (ISTP-2012), pp. 32-39, London, IEEE, 2012.

  This paper introduces the concept of status updates and applies it to dynamic routing in covert channel overlay networks. Wendzel contributed the idea of status updates and Backs developed the status update-based overlay routing with OLSR. Keller contributed aspects to all sections.

- S. Wendzel, B. Kahler, T. Rist: *Covert Channels and their Prevention in Building Automation Protocols – A Prototype Exemplified Using BACnet*, 2nd Workshop on Security of Systems and Software Resiliency, in Proc. 2012 IEEE Int. Conf. on Green Computing and Communications, Conf. on Internet of Things, and Conf. on Cyber, Physical and Social Computing, Besançon, France, pp. 731-736, IEEE, 2012.

  This work presents the first covert storage and timing channels in a building automation protocol suite, namely BACnet. Rist contributed to the related work section of the paper, while Kahler developed the idea and implementation of establishing a MLS BACnet environment using BFR based on Wendzel's idea in the earlier SFCS'12 paper (see below). Wendzel contributed the rest of the paper, especially the section presenting the covert channels in BACnet and also contributed to the section of Kahler through corrections and the evaluation of Kahler's approach against Wendzel's covert channel techniques.

- S. Wendzel, J. Keller: *Systematic Engineering of Control Protocols for Covert Channels*, in Proc. 13th Conference on Communications and Multimedia Security (CMS 2012), International Federation of Information Processing (IFIP), Kent (Great Britain), B. De Decker, D.W. Chadwick (Eds.), LNCS vol. 7394, pp. 131-144, Springer, 2012.

  In this paper, we present the first protocol engineering approach for covert channel-internal control protocols. While Wendzel contributed the majority of the sections, Keller improved the evaluation step for cover protocol areas as well as he provided detail contributions to the whole paper.

- S. Wendzel, J. Keller: *Design and Implementation of an Active Warden Addressing Protocol Switching Covert Channels*, In Proc. The 7th International Conference on Internet Monitoring and Protection (ICIMP 2012), Stuttgart, pp. 1–6, IARIA, 2012.
  This paper received a **best paper award**.

  This paper presents the first protocol switching covert channel prevention means and its evaluation. Wendzel contributed the major aspects of the paper and Keller contributed the optimization aspect for the delay d as well as the idea to use RLL coding (and unary coding for the journal paper version) for covert channels to improve the covert channel bitrate, as well as he contributed details to all remaining sections.

- S. Wendzel: *Covert and Side Channels in Buildings and the Prototype of a Building-aware Active Warden*, in Proc. First IEEE International Workshop on Security and Forensics in Communication Systems (SFCS 2012) held in conjunction with the IEEE ICC 2012, Ottawa, Canada, pp. 6753–6758, 2012.

  This is the first paper that presents covert channels and side channels in building automation systems as well as a first approach to counter such channels.

- T. Rist, S. Wendzel, M. Masoodian, E. André: *Next-Generation Home Automation Systems*, In Proc. Technik für Menschen im nächsten Jahrzehnt – Beiträge zum Usability Day X, Kempter, G. and Weidemann K.-H. (Eds.), pp. 80–87, Pabst Science Publishers, 2012.

This invited paper discusses future trends in building automation systems. Wendzel contributed the outlook section on security aspects in BAS. The rest of the paper was written by the other authors.

- S. Wendzel: *The Problem of Traffic Normalization Within a Covert Channel's Network Environment Learning Phase*, in Proc. Sicherheit 2012 (6. Jahrestagung des Fachbereichs Sicherheit), Darmstadt, N. Suri and M. Waidner (Eds.), LNI vol. 195, pp. 149-161, Gesellschaft für Informatik (GI) / Bonn, 2012.

  This paper discusses problems related to the two-army problem in a covert channel's network environment learning phase (NEL phase) that occur if an active warden is introduced. The paper additionally discusses solutions for this problem.

- S. Wendzel, J. Keller: *Low-attention forwarding for mobile network covert channels*, in Proc. 12th Conference on Communications and Multimedia Security (CMS 2011), International Federation for Information Processing (IFIP), Ghent (Belgium), B. de Decker et. al. (Eds.), LNCS vol. 7025, pp. 122-133, Springer, 2011.

  This paper discusses the mobile usage and optimization of covert channel overlay networks with covert channel-internal control protocols. The idea to utilize multiple areas of a network header for a control protocol as well as implementation-related comments were already developed in Wendzel's diploma thesis. For this thesis, Wendzel contributed the idea of mobile covert channel overlay access, the idea to utilize protocols of multiple layers, calculations for multi-protocol channels and the optimized proxy forwarding. Keller contributed the optimization section which also served as a base for the optimized proxy forwarding section; he also provided detail contributions to the whole paper.

Posters and Technical Reports:

- S. Wendzel: *Control Protocols for Network Covert Channels*, 7th GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit (SPRING), P. Stewin, C. Mulliner (Eds.), SIDAR-Report SR-2012-01, p. 24, July 2012.

  This technical report summarizes the research on covert channel-internal control protocols.

- S. Wendzel: *Mikroprotokolle in verdeckten Netzwerkkanälen*, In: Gernot A. Fink, Jan Vahrenhold (Eds.): Informatik Ruhr Doktorandenkolleg 2011, Oct 6-7, Meinerzhagen Valbert, p. 35, Germany, 2011 (in German).

  This poster provides a short presentation of the idea of using covert channel-internal control protocols.

- S. Wendzel, T. Rist, E. André, M. Masoodian: *A Secure Interoperable Architecture for Building-Automation Applications*, in Proc. 4th Int. Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL), pp. B:1-B:5, Barcelona, Spain, 2011.

  This poster is split into two parts. The first part was primarily written by Wendzel and discusses the secure inter-operability of incompatible hardware components from different manufaturers in smart homes by introducing a secure middleware. The second part was primarily written by Rist and discusses prototypical approaches for the reduction of energy consumption on the basis of the secure middleware to provide privacy for sensitive information.

- T. Rist, S. Wendzel, M. Masoodian, P. Monigatti, E. André: *Creating Awareness for Efficient Energy Use in Smart Homes*, In Proc. Intelligent Wohnen. Zusammenfassung der Beiträge zum Usability Day IX, Dornbirn, Austria, pp. 162-168, 2011.

  While the main focus of this poster is on means to increase the awareness of inhabitant's energy consumption in smart homes, it also discusses various related topics for building automation and represents joint work done under the umbrella of the IT4SE research project. The major part of this paper was written by Rist. Wendzel contributed related work and the inter-operability aspects for a middleware solution.

Professional Articles/In Professional Proceedings:

- S. Wendzel: *Sicherheitsaspekte in der Gebäude-Automation. Perspektiven der Personenüberwachung*, IT-SICHERHEIT, 03/2013, pp. 70-72, 2013.

  This article surveys key aspects of building automation security and highlights selected findings of this thesis to a professional audience.

- B. Kahler, S. Wendzel: *How to own a Building? Wardriving gegen die Gebäude-Automation*, in Proc. 20. DFN Workshop "Sicherheit in vernetzten Systemen", 2013 (in German, to appear).

  This professional paper discusses fingerprinting means to determine the presence of selected building automation systems in local networks and gives an outlook to a wardriving scenario based on ZigBee automation. This paper was joint work with Kahler who additionally implemented the discussed techniques.

- S. Wendzel: *Verdeckte Kommunikation in Gebäuden. Analyse der Gefahren und eine Middleware-basierte Gegenmaßnahme*, BusSysteme Magazine, 03/2012, pp. 182-183, 2012.

  This article is a German translation that summarizes the SFCS'12 paper for a professional audience.

- S. Wendzel, T. Rist, E. André, M. Masoodian, R. Wirth: *Sicherheit beim Energiesparen durch Abstraktion*, BusSysteme Magazine 02/2012, pp. 124-125, 2012.

  The professional article discusses the development of a secure middleware using role-based access control. The middleware served as a base for the building-aware active warden. The article was written by Wendzel, Contributions out of the security scope were made by the other authors.

- S. Wendzel: *Bewusstsein für die Sicherheit im Bereich der Gebäudeautomatisierung*, Hakin9 (de) 03/2011, pp. 11-12, 2011.

  Subject of this professional article is the awareness for security risks in the context of building automation systems.

Previous Work:

The author of this thesis wants to explicitly acknowledge that he already wrote his diploma and master's thesis on covert channel-specific topics. These previous publications were handled as related work for this Ph.D. thesis and were strictly separated from the contributions. Therefore, ideas developed within both theses are only discussed in Chapter 2 (*Background and Related Work*), while the following Chapters 3 (Control Protocols for Storage Channels), 4 (Limitation of Protocol Switching Covert Channels) and 5 (Storage Channels for Building Automation Systems) of this thesis contain the contributions made in the context of the doctoral work.

In his diploma thesis, the author discussed the relevant idea of a *protocol hopping covert channel* and the idea of a *protocol channel* (both channels are types of *protocol switching covert channels*). These channels are required related work and are thus discussed in Chapter 2 of this Ph.D. thesis.

In his master's thesis, the author evaluated the existing means used to limit, detect and prevent covert channels. Chapter 2.5 of this thesis also discusses the detection/limitation/prevention of covert channels since it is required to provide an introduction to this related work to the reader. Therefore, content of the master's thesis was required to be included as re-written and translated content in Chapter 2. New related work in Chapter 2 are parts of the information flow analysis as well as the detailed description of related covert channel establishing techniques, parts of the discussion of active wardens, multi-level security, and the information hiding area, as well as the whole building automation aspect.

Within the author's master's thesis, a detection approach for *protocol switching covert channels* was developed. The algorithm was improved and additional work for machine learning-based traffic classification of protocol switching covert channels was done in joint work with Sebastian Zander. The additional work did not belong to the already finished master's thesis and got published at the 37th LCN conference of the IEEE but was not included in this thesis to provide a clear distinction between previous work and new work for the doctoral thesis. The paper of Wendzel and Zander is the following:

- S. Wendzel, S. Zander: *Detecting Protocol Switching Covert Channels*, in Proc. 37th IEEE Conference on Local Computer Networks (LCN), Clearwater, Florida, pp. 280-283, IEEE, 2012.

The only link to protocol switching covert channels is the previously mentioned paper *Design and Implementation of an Active Warden Addressing Protocol Switching Covert*

Channels with Jörg Keller that does not aim on the detection, but on the limitation of protocol switching covert channels.

The author's diploma thesis is available for download at the website *www.wendzel.de* and the master's thesis (excluding the introduction) became the chapter related to the detection, prevention and limitation of covert channels in the previously mentioned book *Tunnel und verdeckte Kanäle im Netz* (Springer-Vieweg, 2012).

Table 0.1 summarizes the mentioned previous work in the context of this thesis to provide the reader a clear distinction. Only Chapter 2 deals with previous work or re-discusses related work that was already discussed in the master's thesis.

While the diploma and master's thesis dealt with easier topics (basic enhancements of protocol switching and the fundaments of micro protocols, as well as the discussion of related detection, limitation, and prevention work), the more challenging tasks remain to this Ph.D. thesis: The optimization and the engineering of micro protocols, the discussion of a normalized network environment learning phase, and the prevention of protocol switching covert channels. Additionally, this thesis introduces covert channels in building automation systems and presents means to prevent these channels.

Topic	Previous Theses	Ph.D. Thesis
Invention of protocol switching covert channels	X	(related work, Ch.2)
Detection of protocol switching covert channels	X	(related work, Ch.2)
Evaluation of covert channel detection, prevention, and limitation means	X	(related work, Ch.2; additional related work added for Ph.D. thesis)
General idea to utilize a micro protocol in summarized areas of a protocol header	X	(related work, Ch. 2)
Detailed discussion of BLP, IH and CC techniques	-	X (Ch. 2)
Discussion of building automation (related work)	-	X (Ch. 2)
Multi-layer-utilizing micro protocols	-	X (Ch. 3)
Overlay proxies with optimized forwarding	-	X (Ch. 3)
Formal micro protocol engineering approach	-	X (Ch. 3)
Minimized micro protocols with status updates and re-design of an existing research protocol	-	X (Ch. 3)
Evaluation of a normalized NEL phase	-	X (Ch. 3)
Aspect of mobile and upgradable covert overlay infrastructure	-	X (Ch. 3)
Terminological improvements related to MP engineering	-	X (Ch. 3)
Limitation of protocol switching covert hannels	-	X (Ch. 4)
Idea, adversary scenario, and techniques to realize covert/side channels in building automation systems (BAS)	-	X (Ch. 5)
Prevention of high-level covert/side Channels in (BAS)	-	X (Ch. 5)
Prevention of low-level covert/side channels in BACnet	-	X (Ch. 5)

Table 0.1: Summary of previous work done for the diploma and master's thesis in comparison to the work done for the Ph.D. thesis.

Contents

List of Figures

List of Tables

1 Introduction

This chapter introduces the main topics of this thesis. Details, including a state-of-the-art discussion on related work, are provided in Chapter 2. This thesis addresses two areas of today's network covert storage channel research: *control protocols for network covert storage channels* as well as *covert and side channels in building automation systems*.

1.1 Covert Channels

A covert channel is, as defined by Lampson in 1973, a channel that is *not intended for information transfer at all* [Lam73]. In 1985, the U.S. Department of Defense defined a covert channel as *any communication channel that can be exploited by a process to transfer information in a manner that violates the system's security policy* [Dep85]. Murdoch mentions the intentionality of a covert channel: While a side channel leaks information unintentionally, a covert channel intentionally leaks information using a channel not intended for information transfer that additionally breaks a mandatory access control policy [BGNS06, Mur07]. In the context of *multi-level security* (MLS), a covert channel is based on policy breaking communication between different security levels, which can be explained using the Bell-LaPadula (BLP) model [And08, Mur07]: The BLP model contains a set of security levels and the communication within these security levels is restricted by different access control rules. In short, a covert channel exists when a process of a higher level can write confidential data to a lower level *or* if a process of a lower level can read data of a higher level [And08].

However, when related work deals with the topic of *network* covert channels, the MLS context is not always explicitly applied and the term *network covert channel* is used in the meaning of *network steganography*, i.e., a network covert channel's goal is to transfer policy breaking information in a way that the covert communication raises no attention and thus is hidden in other network data [Mil99].

Covert channels can basically be divided into two groups, timing channels as well as storage channels [Dep85]. A *timing channel* transfers information by altering timing

attributes (e.g., time differences between sent network packets) or the order of events (e.g., the sorting of network packets). A *storage channel* transfers information by altering storage attributes of an object, e.g., the "ID" of an IPv4 header or the filename in a given directory. *Behavioral* covert channels have also been proposed as a type of covert channel that can neither be considered as a storage nor as a timing channel and is based on the behavior of the receiver or sender [ALJY12]. Besides, covert channels based on probability distributions, resource exhaustion, and power consumption exist [SM03].

Covert channels are a dual-use good. Petitcolas et al. as well as other authors see covert channels as a security threat that can, for instance, be used by trojan horse communication [PAK99] or by botnets [LGC08, JLY09]. Zander et al. mention additional application scenarios for covert channel communication, e.g., as a technique which can be used by citizens of a country to bypass Internet censorship (e.g., as applied by the Chinese government [Bec11]) and therefore helps to ensure the freedom of speech [ZAB07b]. Since the person/the program that applies covert channel techniques must keep in mind that covert channels are used as a secret, no statistics are available about the actual users and use cases of covert channels in practice. These application scenarios reveal the growing importance of network covert channels in today's Internet and also motivate the growing importance of research in the field of covert channels techniques as well as on covert channel prevention, limitation, and detection means.

1.1.1 Control Protocols and Autonomous Covert Channels

In a computer network, covert storage channels can transfer a different amount of hidden information per packet (dependent on the technique). If the amount of information that can be transferred per packet is big enough, the channel can split the hidden information of a packet into two pieces: a part containing control information as well as one part containing the payload. The control part can be seen as an usual protocol header and can contain a number of different information or instructions, i.e., a sequence number, an acknowledgement flag, or a reset flag.

The first known implementation of such a covert channel-internal control protocol was provided by the tool "Ping Tunnel" by Stødle in 2004 [Stø09]. While Ping Tunnel is referred to as a "tunneling tool" by the hacking community, the covert channel research community joined the control protocol research topic as well. The first optimized control protocol by the research community was presented by Ray and Mishra in 2008 [RM08a] and is, in contrast to Ping Tunnel's protocol, designed to be adaptable to different protocols and to be space-efficient.

Important work was also published in the area of autonomous covert channels, i.e., channels able to adapt their configuration to different situations. The first approach for autonomous covert channels was presented by Yarochkin et al. in 2008 [YDL+08]. Yarochkin et al. utilize a varying set of network protocols for embedding covert messages. The idea of utilizing different network protocols was first presented in 1997 by a hacker called "daemon9" in the tool *LOKI2* [dae97]. LOKI2 was based on a manual protocol switch issued by a special command inserted into the textual user-interface. The first covert channel with the capability to automatically and transparently switch a protocol was the *protocol hopping covert channel tool* (*PHCCT*) in 2007 [Wen09b]. Later, Li and He applied the concept of *natural selection*, i.e., calculating *survival values* for network protocols, to autonomic covert channels [LH11]. Besides, *protocol channels* exist. Protocol channels operate similar to protocol hopping covert channels but signal hidden information solely by the use of specific network protocols. Both protocol hopping covert channels and protocol channels form the set of so called *protocol switching covert channels*.

Both topics, the control protocols as well as the autonomous covert channels, will be discussed in detail in Chapter 2.

1.1.2 Contributions (pt. 1)

This thesis contributes to the existing knowledge by presenting a two-army problem within the so called *Network Environment Learning* (NEL) phase, in which covert channel peers determine their possible communication options, as well as a solution to overcome the two-army problem. In the context of covert channel overlays and protocol hopping, mobile access to the covert channel overlay as well as infrastructural upgrades and optimizations are proposed.

Additionally, the first protocol engineering approaches for covert channel-internal control protocols are presented: The first protocol engineering method is designed to be incremental, i.e., a protocol designer can switch between most of the method's six steps at any time of the development process. Additionally, the resulting control protocol is optimized for a low-attention operation and the protocol engineering method can be adapted to different network protocols. Besides, a second protocol engineering method is presented that is based on a technique called "status updates" and is used to reduce the size of a covert channel's control protocol.

Moreover, a system is presented and evaluated that is capable of limiting the error-free bitrate of protocol switching covert channels while being of use in practice.

1.2 Building Automation

To automate a building does not only mean to control, but also to monitor it as well as to interact with the building [MHH09, GPK10], i.e., to enable a user to actively control a building instead of providing an autonomous self-controlled building automation. Traditional building automation systems (BAS) were designed for the areas of heating, ventilation, and air conditioning (HVAC), but today, many additional applications for building automation exist [WS97]. For instance, a user's interaction with the building automation interface can be based on the goal to assist the user in his daily life, which is especially used for elders and is part of the *Ambient Assisted Living* (AAL) research [LdILB+09].

While safety was considered as an important goal in the industrial development of BAS (e.g., for smoke detection), the BAS security was not taken into account for a long time. Thus, the field of IT security for BAS is relatively new. More attention to the subject was payed after the millennium change, which is confirmed by the rising number of not only scientific publications but general publications discussing security aspects of BAS. As BAS are complex distributed systems and since they are based on an increasing number of different communication systems and (in some cases closed) protocols, improving the security of BAS can be considered a challenging task. However, a security aspect that has not been taken into account is the one of covert and side channels in BAS.

1.2.1 Covert Channels in Building Automation Systems

At a first sight, building automation and covert channels have nothing in common. However, covert channels as well as side channels in BAS are possible. A BAS is basically a computer network using network protocols and these network protocols can be used to embed covert messages. Much work was done to describe possible covert channels in Internet protocols (e.g., [Row97, AK02, Bau03]), but no work was published to investigate covert channels in building automation protocols. Also protocol-independent covert channels in building automation environments, e.g., opening a window to signal a receiver a hidden message, are possible. In MLS environments, such covert channels in BAS must be seen as a security threat; their techniques and prevention means must be discovered.

1.2.2 Contributions (pt. 2)

This thesis is the first work presenting a linkage between covert channels and building automation systems. Therefore, covert storage channels in BAS as well as a means to prevent them using a middleware are presented. The presented covert channels are divided into two different types: *Low-level* covert channels embedded in a building automation protocol (in this thesis, the BACnet protocol family is used) as well as *high-level* covert channels represented through interactions with a building automation system. While the main aspect of this thesis are covert storage channels, it also covers selected side channels and selected timing channels in building automation systems.

The prevention of high-level channels is achieved by introducing a middleware into the system that applies the Bell-LaPadula model, i.e., it enforces the previously mentioned rules to forbid write-downs from a higher security level and read-ups from a lower security level.

Low-level channels based on the BACnet protocol suite are limited by topological changes in the network and the integration of the *BACnet firewall router* (BFR) to achieve MLS – this last aspect is based on joint work with Benjamin Kahler (cf. publications overview) but was extended for this thesis to discuss the additional problem of write-ups and read-downs that cannot represent high-level but low-level covert channels.

1.3 Thesis Overview

Chapter 2 introduces covert channels, building automation as well as the related topics of this thesis in detail. The contributions for the optimization of network covert storage channels based on micro protocols, terminological contributions, and the discussion of the NEL phase are presented in Chapter 3. Chapter 4 introduces an active warden to limit the maximum error-free bitrate of protocol switching covert channels. Covert and side channels in buildings as well as their countermeasures are discussed in in Chapter 5. Chapter 6 provides a summary on our results and provides an outlook on future work.

The following Figure 1.1 visualizes the assumed difficulty regarding the limitation of the covert channel types discussed in this thesis in comparison to other covert channel techniques. Protocol channels were not part of previous limitation research and protocol hopping covert channels must be considered as hard to limit since they comprise various other covert channels and can adapt to changes in the network environment.

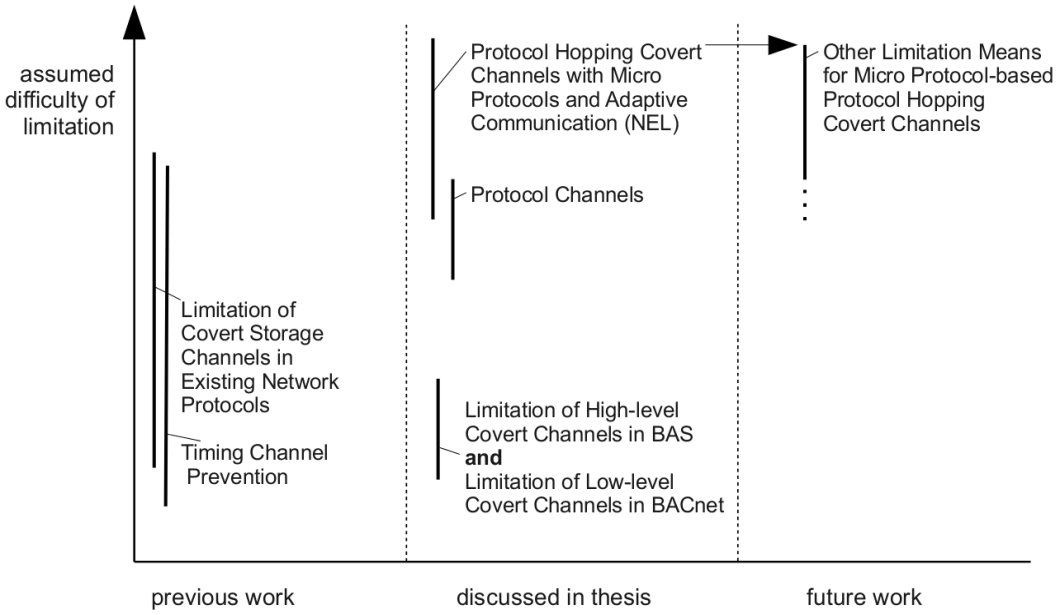

Figure 1.1: Qualitative comparison of considered covert channel limitation difficulties.

2 Background and Related Work

This chapter provides a detailed background for the area of network covert channels and building automation systems, as well as it discusses the fundamentals of related aspects.

2.1 The Bell-LaPadula Model

As mentioned in the previous chapter, covert channels are linked to the topic of *multilevel security* (MLS). Different models for MLS exist but covert channels are usually defined in the context of the so called Bell-LaPadula (BLP) model. The importance of the BLP model for covert channels state a reason for the explanation of the BLP model within this thesis.

The BLP model was presented by D. E. Bell and L. J. LaPadula in 1973 [LB73] and contains a set of security levels (e.g., "top secret", "secret", "confidential"). A higher classification level is linked to higher sensitivity than a lower classification (e.g., the "top secret" classification is more sensitive than the "secret" classification) [Bis03].

Each subject s has a security *clearance* $L(s)$ (e.g., $L(John) = confidential$), what is the maximum security level it can take, and each object o is at a security *classification* $L(o)$ (e.g., $L(file_x) = secret$). While both terms, the security clearance as well as the security classification, refer to elements of the same set of security levels, a distinction exist: A subject can decrease its security level to a lower value than its *clearance*, while an object cannot decrease its security *classification*, i.e., a subject can for instance login to a system using a security level that is lower than the subject's clearance [Eck12]. Additionally, the usage of two different terms eases the discussion and understanding of the BLP model.

The BLP model supports discretionary access control based on an access control matrix as well as it contains two mandatory access control rules [Bis03]. To access an object, a subject needs both mandatory and discretionary access to the object [Bis03].

The first mandatory access rule (the so called *simple security condition* or *no read up* rule (NRU)) ensures that a subject s can only read/execute an object o if $L(s) \geq L(o)$,

while the second rule (the so called *star property* or *no write down* rule (NWD)) ensures that s can only write/append to o, if $L(s) \leq L(o)$ [Bis03]. Additionally, for the NRU and the NWD rule, discretionary read/execute and write/append access, respectively, for s to o must be available [Bis03]. For better readability, "read/execute" is henceforth referred to as "read" and "write/append" is referred to as "write", what is a usual convention in the related literature.

Besides the explained basic version of the BLP model, an extended model containing so called *categories* exists. Categories represent *special access privileges* (e.g., for a specific project or organizational unit) [LB73]. Categories are applied to both subjects and objects. It is possible that a subject or an object are placed in more than one category. For instance, a system could define the categories *accounting*, *development*, and *support*. The subject *John* could be cleared into (*confidential*, {accounting, support})[1], while the object $file_x$ could be at the security level (*secret*, {*accounting*}). Regarding to Bishop, such a combination of a security level with a category is called a *security level* as well (since it can be seen as an enhancement of the existing basic security levels) but is sometimes also called a *compartment* [Bis03].

The two previously mentioned rules (NRU and NWD) are present in the enhanced BLP model as well and are based on the so called *dom* relation (or *domination* relation) [Bis03]. The domination of a security level (the compartment) (L, C) where L is the security level itself and C is the category of the level, over the security level (L', C') is given, if $L' \leq L$ and $C' \subseteq C$ [Bis03]. Thus, read access to an object is denied if the object is at a level with a category the subject is not part of, even if $L(s) > L(o)$ [Bis03]. Based on the *dom* relation, the enhanced versions of the NRU and NWD rules are defined as follows [Bis03]:

- The read access for s to o is only allowed, if s *dom* o and discretionary read access for s to o is given (NRU rule).
- The write access for s to o is only allowed, if o *dom* s and discretionary write access for s to o is given (NWD rule).

Example: Given the subjects *Alice* and *Bob* as well as the goal to read the object $database_x$. Let the associated security levels (visualized in Figure 2.1) be

- (L, C) of Alice: $(secret, \{accounting, research\})$,

[1] While the first parameter represents the security level itself, the second parameter represents the set of security categories.

- (L, C) of Bob: $(confidential, \{accounting, support\})$, and
- (L, C) of $database_x$: $(confidential, \{support\})$.

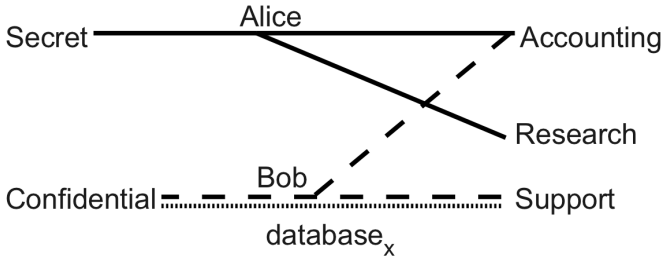

Figure 2.1: BLP example condition. Only *Bob* is allowed to read $database_x$.

In this case, read access to $database_x$ would be denied for *Alice* although $L(Alice) > L(database_x)$ but since *Alice* has no access to the category *support*, i.e., *Alice* $\neg dom$ $database_x$.
On the other hand, *Bob* would gain read access to the database because $L(Bob) = L(database_x)$ and $\{support\} \subset \{accounting, support\}$, i.e., *Bob dom* $database_x$.

If a communication from a higher security level to a lower security level is possible, i.e., the mandatory access rules NRU or NWD are violated, a covert channel exists [And08, Mur07].

2.2 Information Hiding

Covert channels are part of the *information hiding* discipline. While cryptographic research aims on keeping the content of a message or its sender secret, the information hiding research aims on keeping the presence of secret information itself hidden or to ensure embedded information cannot be easily removed from other data (e.g., a watermark in a movie). For instance, an observer should not be able to notice the covert transfer of information within a cover data transfer [PAK99].

Regarding to the first *Information Hiding Workshop*'s informal meeting [Pfi96], information hiding generally *embeds* data of a given datatype[2] (called *Embedded-<datatype>*) that is to be hidden in a other data (called the *Cover-<datatype>*). Optionally, a (steganographic) key can be part of the embedding process. The modified Cover-<datatype> containing the Embedded-<datatype> is called the *Stego-<datatype>*. For

[2]The <datatype> is a placeholder for an object type, e.g., an image, a video or a text [Pfi96].

the *extraction* process (i.e., the re-construction of the embedded data from the Stego-<datatype>), the algorithm used to embed the Embedded-<datatype> in the Cover-<datatype> must be known. If a key was used in the embedding process, it is usually the same key that is required to extract the Embedded-<datatype> from the Cover-<datatype> [Pfi96]. However, public key steganography is feasible and thus, both keys can differ [vAH04].

The party that tries to remove or detect the presence of a/the Stego-<datatype>, or tries to deduce the sender's or receiver's identity, or tries to extract, modify or remove the Embdedded-<datatype> is called the *stegoanalyst*. The party that applies the information hiding technique to hide the Embedded-<datatype> is called the *embeddor* and the non-adversary party that wants to reconstruct the Embedded-<datatype> is called the *extractor* [Pfi96].

The information hiding discipline is split into the following sub-disciplines [PAK99]:

1. *Steganography:* Steganography deals with the problem to hide information within other information, e.g., a cryptographic key hidden within a JPEG image [PAK99].

2. *Anonymity:* Anonymity is defined as *the state of being not identifiable within a set of subjects* [PK01]. This thesis does not focus on anonymity. However, Pfitzmann and Köhntopp provide a detailed terminological discussion of related terms [PK01].

3. *Copyright Marking:* Similar to steganography, copyright marking embeds information into other data (e.g., a company's name in a digital movie). However, it is not necessary, that the embedded information is hidden, but should be hard to remove (*robust*) [PAK99]. In copyright marking, the terminology differs to the mentioned one (e.g., an object is *marked* after the marking algorithm was applied [PAK99]). Copyright marking is not subject of this thesis.

4. *Covert Channels:* Covert channels were already introduced in the first chapter and will be explained in detail within the next section which also includes a discussion of differences between covert channels and similar terms (e.g., side channels).

2.2.1 Adversary Scenario

Simmons introduced the well-known *prisoner's problem* in 1983 [Sim83]. In the prisoner's problem, two prisoners (Alice and Bob) want to cooperate to create a plan for their escape. The direct communication between Alice and Bob is not possible but the warden

Walter allows the exchange of messages between both prisoners. The warden can read, manipulate, drop, and spoof the messages between both prisoner's. Thus, Alice and Bob must apply a technique for information hiding Walter is not aware of [PAK99]; they can apply a steganographic technique such as invisible ink on written messages.

The information hiding literature differs between active and passive wardens [PAK99] which can be seen as the adversaries for the the area of covert channels as well.

Passive wardens only observe information flows and try to detect the presence of steganographic elements, the embedded content (i.e., the hidden message itself), and try to prove that a third party is involved in the communication [Pfi96].

Active wardens have the ability to modify the information flows (e.g., to remove the steganographic content with or without breaking the cover message) [Pfi96]. In a malicious variant of an active warden, the active warden does not only manipulate steganographic content but does also introduce new bogus messages into the covert communication [Cra98].

The adversary's goals (detecting, removing/destroying, manipulating, and understanding the steganographic messages as well as proving third party participation) are linked th the capabilities of the warden.

In Section 2.5, different means to detect and prevent network covert channels will be discussed of which the network covert channel detection means can be seen as variants of a passive warden and the limitation and prevention techniques can be seen as variants of active wardens.

2.3 Covert Channels

The term *covert channel* was briefly introduced in the previous chapter as a channel *not intended for [but used for a] communication* [Lam73] that also breaks a mandatory security policy [Mur07]. Murdoch mentions the variety within the covert channel terminology in the literature and provides a detailed discussion on this terminology in [Mur07], including a clear distinction between covert channels, side channels, steganographic channels, and subliminal channels: In comparison to a covert channel, the sender of a *side channel* leaks information unintentionally, while a *steganographic channel* aims to prevent observation, and a *subliminal channel* is a steganographic channel within a cryptographic algorithm [Mur07]. Murdoch also mentions the fact that covert channels can be placed in steganographic channels but that not all covert channels must be part of steganographic channels.

This thesis is based on the covert channel terminology of Murdoch, but utilizes the term *network* covert channel in the meaning of Millen as covert channel with steganographic attributes that is transferred over the network [Mil99]. For Das, *network* covert channels are equal to *network* steganography [Das12]. Thus, a network covert channel aims on preventing observation and on raising only little attention as also mentioned in [RM08a].

An important aspect of a covert channel is its raised attention. Therefore, Giani et al. introduced the term *covertness* [GBC06]. In general, the covertness of a steganographic communication is linked to the utilized media that provides a different capacity. For instance, a floppy disk provides a smaller capacity than a modern USB memory stick. The higher the transmission rate of the data exfiltration over a media is, the lower is the covertness of the steganographic communication, and thus $Covertness \propto (Capacity - TransmissionRate)$ [GBC06].

2.4 Covert Channel Hiding Techniques

Numerous techniques for embedding covert channels in other data were discovered within the last decades. Thus, a selection of important techniques in the context of this thesis shall be explained. Since our focus is on network covert storage channels, they will be described in more detail while only important timing channel and behavioral channel examples will be given.

In the following, we start with the explanation of low-level covert channels in networks and finish with the highest communication layer of the TCP/IP model.

2.4.1 Local Network Covert Channels

Girling presented a storage channel within LAN frames by manipulating the address field in the frames as well as the length of the frame's data [Gir87]. Girling suggested a communication between a sender and a receiver in the same subnet in which all frames are accessible by all receivers (e.g., 10base2 or Ethernet with hubs but not with switches). The sender alters the destination address within the frames and the passive receiver monitors these frames and extracts information from address values of the frames [Gir87].

Ji extended the idea of frame length-based hidden information transfer by adapting the statistic behavior of the regular traffic in a LAN [JLSN09].

Jankowski et al. presented a similar approach in [JMS10] that allowed group communication via manipulating Ethernet frames. For the group communication, hosts identify each other via ARP requests placed within the manipulated Ethernet frames while payload is transferred in frames with TCP content. In both cases (TCP and ARP), the hidden data is not placed within the ARP or TCP header but in the Ethernet frame's padding. ARP and TCP are used to identify the operation (payload transfer or peer identification). However, the first approach for utilizing frame padding was presented by Wolf [Wol89] but did not comprise the feature of group communication.

Another mentionable approach for a LAN-based covert channel communication is to utilize the DHCP protocol as done by Rios et al. in [ROL12]. The authors evaluated the different fields of the DHCP header for their information hiding suitability and developed a proof of concept-code called *HIDE_DHCP*.

A different idea discussed by Li et al. is to utilize a LAN switch for a covert communication [LZCZ11]. Therefore, three computers A, B and C are connected to the same switch. A sends burst traffic to B (the covert channel receiver). If no other computer sends traffic to C, the throughput between A and B (measured by B) will be high. If the covert channel sender C also sends traffic to B, the throughput between A and B will decrease. C therefore manipulates the data rate at which it sends traffic to B what B understands as covert channel message.

Besides the already mentioned storage channel in LANs, Girling was also the first author to present LAN-based covert timing channels for LAN frames by altering interval timings between two frames [Gir87].

2.4.2 Covert Channels on the Internet Layer

Various techniques to embed covert storage channels in Internet layer protocols were developed since the late 90's. Rowland was the first author to present a number of such covert channels [Row97], new techniques were afterwards presented in [Ahs02], [ML05], [SK06] and [LLC07].

Typical fields to embed hidden information in IPv4 are the Type of Service (ToS) bits, the *Identifier*, the *Reserved* flag, the *Fragment Offset* (by altering fragment sizes [ML05]), the *Time to Live* (TTL) field, and the *Options*. The different fields are linked to different space they provide for hidden data (e.g., the Reserved flag comprises only 1 bit while the Identifier field comprises 16 bit). Since the Internet layer provides routing capabilities, the covert channel also needs to take hop-based modifications into account. For instance, the TTL is decremented by each hop.

IPv6-based covert channels utilize similar fields as IPv4: the *Traffic Class* (similar to IPv4 ToS), the *Flow Label*, the *Hop Limit* and parts of the IPv6 Extension Headers [LLC07].

The ICMP protocol can be considered as a protocol used by many covert channel tools (e.g., it is used by *Ping Tunnel* [Stø09], LOKI2 [dae97] and by a tool developed by Ray and Mishra [RM08a, RM08b]) since it provides much space for hidden data and is easy to utilize. Usually, ICMP *Echo Request* and *Echo Response* messages are used for covert channels since these messages provide much freely utilizable space [SdSNL06, Stø09].

Besides ICMP Echo-based covert channels, other options to embed a covert channel into ICMP were discussed in [BR05] and [SdSNL06].

ICMP-based covert channels are known for being used to coordinate DDoS attacks [SdSNL06].

2.4.3 Covert Channels in TCP and UDP

Rowland initially discovered covert channels in the TCP *Initial Sequence Number* (ISN) field [Row97]. The ISN is only transferred when a new connection is created and has a size of 4 bytes. Rutkowska developed an enhanced ISN-based *passive* covert channel by not directly initiating a TCP connection to transfer hidden information. Instead, a covert channel sender waits for a regular TCP connection and modifies the ISN of the generated traffic by a ISN modification layer in the Linux kernel [Rut04].

Rowland also developed an enhanced version of the ISN-based covert channel with the goal to hide the sender's address [Row97]. Therefore, a *bounce server* is introduced. The bounce server is used by the sender to send messages to the receiver. Therefore, the senders sends a spoofed TCP packet to the bounce server. The packet contains the receivers source address and thus, lets the bounce server respond to the receiver that receives a packet that does not contain the sender's address. The bounce server will increment the ISN that is transferred as acknowledgement number to the receiver that has to decrement the acknowledgement number to get the original ISN value.

Another approach is to use TCP and UDP port numbers to send hidden messages [Bor08] while Giffin et al. used TCP timestamps to embed covert payload [GGLT03] (timestamps are an optional header component of the protocol). The authors applied a minimal delay to create a covert timing channel in this way.

2.4.4 Covert Channels on the Application Layer

Embedding covert channels in application layer protocols can be considered trivial. Especially protocols with plain-text headers (e.g., HTTP, POP3, SMTP, and NNTP) provide numerous ways to hide information. For instance, the *User-Agent* as found in HTTP requests (as well as in SMTP and NNTP message headers) can be used to hide information by choosing one of the many available user agents as well as by choosing one of the available software versions for the selected user agent or by adding information about the user agent's supported features. Various approaches for HTTP-based covert channels were presented in [Bau03].

Zander described a technique to create covert channels in network games like *Quake III* [Zan10]. To send a hidden information, the sending player has to move his character (a 3D person) in a pre-defined way (e.g., rotating around the x/y/z axes) what has to be monitored by the covert channel's receiver.

Such hiding techniques which depend on the application layer payload (e.g., image files) can be considered as *multimedia steganography* and are not considered as a network covert channel within this thesis. For instance, a message can be hidden in an media attachment for an email or in a HTML website [AQDS10].

Timing channels on the application layer exist as well. Eßer developed a covert timing channel for the Apache webserver that delays response messages for a value measured by the receiver [Eße05]. Delays for basically all other application layer protocols are thinkable as well.

2.4.5 A Note on Timing, Behavior-Based and Other Covert Channels

Besides the alreay mentioned timing channels by Girling, Giffin et al. and Eßer, many other covert timing channels for network protocols were developed, such as timing channels based on *inter-arrival times* of network packets (also called *inter packet gaps*, IPGs) [CBS04, Zan10]. Another approach is to modify the order of network packets as discussed by Ahsan and Kundur [AK02].

Anthony et al. proposed a behavior based covert channel using the update mechanism of anti-virus (AV) programs [ALJY12]: Since signature updates in anti-virus databases occur often (up to multiple times a day on an AV client machine), such updates can be considered as normal traffic and thus, raise no attention. The sender of the channel needs access to the AV database and inserts new signatures that the AV client (the

receiver) downloads. The receiver scans a local and unique directory with files using the new signatures. The scan results represent the encoded covert message.

Covert channels in games (e.g., the previously mentioned approach by Zander [Zan10]) can also be considered to be behavior-based covert channels since they do not depend on storage or timing attributes, but on the behavior of players [JLY09].

Sabelfeld and Myers also mention *probabilistic* channels which signal hidden information by *changing the probability distribution of observable data* [SM03]. In a publication by Sabelfeld and Sands, an example for such a channel is given by a conditional coin flipping based on the value of a secret (high) value [SS00].

Resource exhaustion channels are created by entirely allocating a selected finite resource, e.g., local memory or parallel openable file descriptors on a system [SM03]. The last mentioned covert channel type are *power* channels. Power channels require an observer to be able to monitor the power consumption of a computer while the covert channel's sender (or the program that includes the side channel) alters the computer's power consumption [SM03].

2.4.6 Protocol Channels

A technique previously considered as a covert storage channel is the *protocol channel* that was presented by the author in [Wen08a] (including the proof of concept implementation PCT [Wen09a]). A protocol channel communicates by sending a network packet to a receiver using a selected protocol of a previously defined protocol set P. Each element of P is linked to a bit value.

For instance, $P = \{HTTP, DNS\}$ and HTTP="1", DNS="0". To transfer the message "110" the senders needs to send HTTP, HTTP, DNS. Error-correcting codes improve the use of protocol channels. As this thesis will show, it is feasible to counter protocol channels (cf. Chapter 4).

Since the "protocol" field's value (e.g., the *protocol* field in IPv4 or the *Next Header* field in IPv6) specifies the hidden message, this channel can be considered as a covert storage channel. On the other hand, the channel can be seen as a covert timing channel since the sequence (its packet order) must be evaluated by the receiver. However, since the channel's behavior (i.e., sending a packet of the protocol $p_i \in P$) produces the hidden message, the channel can also be seen as a behavioral covert channel.

A covert channel presented by deGraaf et al. in [dAJ05] can be considered as a protocol channel as well. The authors propose to use port knocking for realizing an authentication. The covert channel is therefore embedded into the destination port information

of UDP. The port knocking channel can be seen as a protocol channel since it embeds hidden information by altering the application layer protocol via the destination port.

Swinnen et al. proposed a similar covert channel in [SSP+12]. Their covert channel utilizes reliable TCP-based network protocols (e.g. FTP and HTTP). The authors believe to provide the first approach which enables protocol independence, which is, in the context of the approaches we discussed earlier, incorrect. However, Swinnen et al. added the novel feature of combining different network attributes: A connection could, for instance, not only use the FTP protocol but could be a FTP connection to a selected FTP server to encode a hidden message. Features for adaptive covert channels were shown in [SSP+12] as well but will be discussed in the context of other techniques for adaptive covert channels in Section 2.6.3.

2.5 Detection, Prevention, and Limitation Techniques

Note: The detection, prevention, and limitation of network covert channels was already discussed by the author in his master's thesis of which most parts formed a chapter in his book "Tunnel und verdeckte Kanäle im Netz" (both in German). Therefore, the author did already describe many existing approaches to counter covert channels and evaluated their (dis)advantages. For this Ph.D. thesis, only parts of the subsections on noninterference, traffic normalization and further anti-covert channel means contain a significant amount of new related work. Other work, including the discussion of (dis)advantages, are based on the previous work. Although the author's previous publications exist, the detection, prevention and limitation of covert channels are mandatory topics for a covert channel-related thesis, and thus, must be discussed in this thesis nevertheless.

As most anti-covert channel means are designed to counter only a subset of the possible covert channels, it can be assumed that an effective protection against the majority of network covert channels requires the combination of various available means to implement a good covert channel protection. Therefore, it is necessary to evaluate the protection costs and benefits to conclude an appropriate protection level (cf. [XM04] for details on cost/benefit analysis in the security context).

2.5.1 Information Flow Analysis and Noninterference

Access control and cryptography are useful techniques to secure data but cannot ensure the confidentiality of the data. *Information flow analysis* aims on verifying, whether high

level data in a program is leaked to a lower level (e.g., verifying whether private data is leaked to the public) [Zda04, Smi07]. The so-called *noninterference* is satisfied if an adversary able to observe all low (public) leveled variables cannot obtain any information about the high (private) data. However, even if a program satisfies noninterference, an information leak can be created by composition of secure systems if a secure system's output is the input to another secure system [McC88, And08].

This thesis focuses on existing environments with already defined protocols (TCP/IP and BAS protocols) and shared resources (existing network infrastructure). While information flow analysis can help to uncover possible covert channels [Bis03], the existence of covert channels in computer networks was already proven by many publications (as discussed in the previous section). Thus, a detection of actually exploitable covert channels as well as their limitation is of more relevance for this thesis than information flow analysis.

However, since information flow analysis is linked to covert channel research, an overview of selected means developed to counter covert channels from an information flow point of view will be explained in this section.

Language-based Security and Covert Channels

Language-based techniques can be used to specify security policies as well as to ensure their enforcement [SM03]. Therefore, information flows in programs are analyzed and security-typed languages can ensure the program's conformity to the security policy using compile-time type checking [SM03].

The presence of covert channels in a program can be analyzed by different language-based techniques such as the *covert flow trees* developed by Kemmerer and Porras [PK91] and can be prevented by other language-based techniques such as Agat's approach that aims on providing observably equivalent program execution times to prevent timing channel-based information leakage [Aga00].

Sabelfeld and Myers mention different covert channel types which can be enabled by a program [SM03]. However, regarding to the previously discussed terminology, these channels must be called *side channels* instead of *covert channels* because they do not rely on an intentional sender.

The first mentioned type are covert channels based on *implicit flows*. In comparison to implicit flows, *direct* flows would *directly* provide secret/private data to a public variable. An example for an implicit flow is given by Smith in [Smi07] where one bit of the variable `secret` is leaked to the variable `leak` using the following code.

```
if ((secret % 2) == 0)
  leak = 0;
else
  leak = 1;
```

The second type of channel mentioned by Sabelfeld and Myers is the *termination-based* channel, i.e., a channel that can leak secret information by observing whether a program terminates. Again, an example for such a program can be found in Smith's work [Smi07]:

```
while (secret != 0)
  ;
```

A similar type of covert channel is the *exception-based* covert channel mentioned by Bishop [Bis03]: If an exception depends on the value of a high leveled variable, the exception leaks information.

Timing channels form the third type of covert channel in [SM03] and language-based approaches aim on analyzing observable differences in the timing behavior of programs to prevent information leakage. Agat presented work for the language-based prevention of timing channels in [Aga00]. He aimed on modifying Java byte code in a way that the runtime of a program does not differ independent whether a condition in the program is true or false. Therefore, additional instructions were introduced into the byte code. For instance, the following code will execute faster, if the condition is false:

```
if (secret == true)
  do_something();
```

Agat introduces instructions for the `else` case which take the same time as if the condition would be true. Based on his timing measurements, Agat discovered the fact that the approach is not sufficient since previously cached data can manipulate the runtime of a program nevertheless – the following example could be executed faster if `secret` is `true` since `method1` was previously cached, even if the execution time of `method1` and `method2` are observable equivalent:

```
method1();
if (secret == true)
  method1();
else
  method2();
```

Shared Resource Matrix Methodology

The *Shared Resource Matrix* (SRM) presented by Kemmerer in 1983 aims to detect covert storage and timing channels. Therefore, it is analyzed, which *operation* (i.e., a program procedure) has access to which *resource* (an attribute) [Kem83]. The access to a resource is categorized in the types *modifying* and *reading*, e.g., the operation *login* has *read access* to the resource *username*. A matrix representation is used to provide an overview on each operation's resource access options (e.g., operation x has *read access* to the shared resource *password*).

To detect possible covert storage channels, it is analyzed, whether two different operations have access to the same resource. One operation is required to have write and the other operation is required to have read access to the shared attribute. For instance, a buying operation for a product database can read whether a shared resource (e.g., a product) is available, while another operation used to keep the product database up to date can write to the shared resource. A covert channel can be established if a high leveled sender writes to a product information while the low leveled receiver reads the product information.

The timing channel detection requires a shared time reference (e.g., sender and receiver process share the same system time reference) and a method for the simultaneous initialization of the sending and the receiving process.

The SRM cannot only be applied to source code but also to other steps in the software development lifecycle. Kemmerer applied the SRM to requirements written in the English language [Mil99]. Bishop mentions the drawback that the SRM cannot handle sequences of procedure calls (e.g., an attribute is not directly accessible to a given function f but indirectly accessible by another function g called by the function f) [Bis03] – a problem solved by the later developed *covert flow trees*. While Wray argued that *all* covert storage channels can be found using the SRM [Wra91], Bishop did not agree because indirect covert storage channels using function call sequences are indeed non-detectable using the SRM.

McHugh developed an improved version of the SRM called the *Extended Shared Resource Matrix* (eSRM) [McH95, McH01]. Therefore, McHugh added 3 features: *user flows* to differ between input and output flows, *operation splitting* to differ between independent information flows within an operation (e.g., the resource x is only modified in one of two flows of an operation), and *guard expansions* to add distinctions for different conditions in the already splitted operation flows (e.g., in the splitted operation Op_1 the resource x is modified only in some cases).

The eSRM enables more detailed analyses and can evaluate covert channels found by the basic version of the SRM as not existing (e.g., a function can read a resource x but the condition required to access the resource can never be satisfied if a high leveled process tries to send secret data). In comparison to the basic SRM, the eSRM results in a more complex matrix.

Covert Flow Trees

As mentioned earlier, *covert flow trees* (CFTs) enable the detection of covert storage channels based on indirect information flows. The idea of CFTs was presented by Kemmerer and Porras in 1991 [PK91]. CFTs can automatically be generated and build a representation in form of a tree of the information flows based on a program's source code [Bis03].

Before a covert flow tree can be generated, a matrix representation of the flows must be created containing the information which operation accesses which resource in which way (referencing, modifying or returning) [Bis03]. The CFT is build using the matrix based on *goals*. A goal is to create a covert storage channel using a specific resource (e.g., a covert storage channel based on the resource *product information*).

Therefore, the goal is represented as a node that requires two child nodes: One child node represents the direct or indirect writing of the resource and the other child node represents *the direct [or indirect] recognition* of the resource [Bis03]. In the final step, lists of operation sequences are created that represent the covert information flow. For each resource, all possible combinations of sending and receiving operations form a single list (e.g., one list could be *(WriteProductInformation, ReadProductInformation)*).

TaintDroid

A recent approach towards a more practical application of information flow security was made by the TaintDroid project [EGC+10]. TaintDroid is an extension for Android-based smartphones *that tracks the flow of privacy sensitive data through third-party applications* [EGC+10] to obtain information about the use of provided private data by the installed applications. Therefore, TaintDroid *taints* (labels) privacy-related data and monitors their use by applications on different levels (messages within inter process communication, native library methods, and files). If tainted information leaves the system, the event (and the application linked to the data leakage) is logged.

The feature of *tainting* is also available in some programming languages (e.g., Perl, PHP, and JAVA [EGC+10]).

Preventing Covert Channels in Business Processes

Information flows are fundamental for business processes. In a business process, a covert channel exists if private data is leaked by the business process into public data (e.g., a public summary of an internal business document containing leaked information). Accorsi and Wonnemann developed a framework to detect covert channels in business processes called *InDico* [AW09, WAM09, AW11a].

InDico comprises different steps and aims on automatically detecting possible covert channels. In the first step, the business process (represented by a typical business process modeling language such as BPMN[3]) is transferred into a colored petri net (CPN) referred to as an *IFnet* [AW11b]. The next step assigns security levels to operations and data objects in the IFnet and aims on *i)* applying access control rights and on *ii)* isolating the different sub-processes within a business process (operations assigned to different security levels shall operate in a separated way) [AW11b]. Finally, a static analysis of the IFnet is done to verify, whether policy-breaking information flows are possible. A business process is afterwards certificated (the certificate contains the possible covert channels of the business process).

The authors already applied InDico in practice where the framework could acknowledge the existence of already known covert channels as well as covert channels in planed business processes [AW11b].

Practical Relevance

However, Zdancewic in 2004 as well as Smith in 2007 mention the problem of practical relevance of information flow security verifications: While the theory behind information flow security was extensively developed within the last decades, the practical application faces the problem of being forced to inter-operate with existing infrastructure and thus, would require code re-writing of the existing software [Zda04]. As a possible path to practical application, Zdancewic proposes to reduce the goal of preventing *all information-flows from secret data to public observers.*

Zdancewic therefore also proposes not to focus only on non-interference but on the application of existing approaches to practice [Zda04] (especially by using policies that allow a *declassification* (*downgrading*) [Zda04]).

Smith continues the discussion of Zdancewic regarding the practical aspects of information flow analysis and acknowledges the *little practical impact* that could be achieved

[3] *Business Process Modeling Notation*

till 2007 [Smi07]. As a problem, Smith also mentions the theoretical focus on *"toy" languages rather than [on] full production languages* [Smi07].

Like Zdancewic, Smith considers non-interference as the wrong goal to aim on since it requires that no information leaks are possible at all. If small information leaks are tolerable, practical functionality can be achieved [Smi07]. Smith mentions a password checker as an example application for such an use case. The password checker should not leak a full password, but should leak the information, whether the password is correct or not, to be able to log a user in [Smi07]. A second practical problem are timing aspects, e.g., a secret information will be made available to a customer after he paid for it [Smi07]. Additionally, Smith proposes to develop *formalism[s] for specifying useful information flow policies that are more flexible than noninterference*, i.e., widely applicable as well as understandable.

However, the Shared Resource Matrix, Covert Flow Trees, tainting, and the inDico framework are examples of indeed practically applicable approaches.

In comparison to the means presented in this thesis, language-based security can lead to covert channel elimination on earlier steps of the software development lifecycle. These approaches can help to determine covert channels in network protocols to be engineered. However, since we deal with existing protocols within the Internet and in building automation systems, approaches are required to *handle* potential covert channels in these environments. The application of data tainting like done for TaintDroid (that represents a dynamic instead of a static analysis) could be a promising extension to prevent covert communication in building automation environments but tainting is not part of the main aspects of this thesis.

2.5.2 Traffic Normalization for Covert Channel Prevention

A traffic normalizer is a network gateway with filtering capabilities that enhances the capabilities of a simple firewall. In the literature, a traffic normalizer is sometimes also referred to as a *packet scrubber* or as a firewall with *scrubbing* functionality [BP09]. While a plain firewall can block and forward traffic, normalizers can be seen as advanced firewalls/intrusion prevention systems capable to modify traffic [MWJH00]. Thus, a traffic normalizer can remove malicious/steganographic elements of network traffic and can forward the remaining data. For instance, a normalizer can clear a specific bit used for information hiding or can unify a field in network protocol headers [HPK01]. As previously explained in Section 2.2.1, a traffic normalizer can be considered as an active warden.

Popular examples of normalizer systems are the *Snort* normalizer [Sno12], the FreeBSD transport layer protocol scrubber [MWJH00], and the OpenBSD *pf* firewall that comprises a *scrubbing* feature [Ope13]. However, since a normalizer alters traffic details, its usage results in side effects [SdSNL06]. For instance, if the QoS values are unified or the fragmentation options are unified for each packet, the available features linked to these fields are reduced.

A special normalizer system called an active warden with *active mapping* capability exists. With active mapping, the network and its policies are mapped [Sha02]. These mapped information is used by a NIDS to reduce ambiguities within the traffic, i.e., data that can be interpreted in multiple ways [LLC07].

Lewandowski et al. enhanced the idea of active mapping to a so-called *network-aware active warden* [LLC07]. These network-aware active wardens comprise knowledge about the network topology and implement a stateful traffic inspection with the goal to eliminate network covert channels in fields of IPv6 protocol headers [LLC06, LLC07].

Another approach for data leakage prevention can also be considered as a type of normalizer: Glavlit is a system presented by Schear et al. that aims on preventing data leakage on the application layer [SKZV06]. In comparison to previous solutions, Glavlit provides a higher performance for the data leakage protection by splitting their system into two components: the *active warden* as well as a *guard*. The active warden decides *whether an object is appropriate for external release* (this process is called *vetting*) [SKZV06] and provides a signature of the object to the guard. The warden can apply *any type of digital and/or human reviews to determine if the object is fit for release* [SKZV06]. When traffic to be send to an external network is sent to the guard, the guard does not decide whether the data is allowed to exit the internal network but ensures that only those objects exit the network that were previously *vetted* by the active warden. Thus, if the same data object is sent multiple times, it does not have to be vetted again by the active warden but does only need to pass the guard.

Algorithms for verifying specific data leakage problems are available as well. For instance, Hall et al. presented an algorithm that is capable of detecting credit card data exfiltration [HKM11]. However, such specific algorithms do not generally aim on preventing covert channels but very specific types of information and are thus not in the focus of this thesis.

Fisk et al. introduced a classification for active wardens that is of importance in the context of traffic normalization. In [FFPN03], they split the media used to transfer the covert channel's data in so called *structured* and *unstructured* carriers. While structured

carriers can be interpreted by a machine and comprise bits with specified interpretation rules, unstructured carriers (e.g., pictures) can be interpreted by humans and comprise bits whose values can be interpreted in different ways [FFPN03]. Headers of network packets can be seen as structured carriers. Thus, traffic normalizers are active wardens which aim on removing covert channels from structured carriers.

In the same publication, Fisk et al. additionally introduce the *Minimal Requisite Fidelity* (MRF). The MRF concept aims on applying normalizations in an acceptable way, i.e., side effects for end users should be minimized but the normalization should be effective nevertheless [FFPN03]. For MRF, a categorization of different normalization cases is presented. Each of these normalization cases can be threaten in a different way to reach MRF [FFPN03]. For instance, *constant* fields in packet headers can always be modified in a way that the field is set to the only possible value without facing end user side effects. Similarly, fields that should be zero'ed in most or any case, can be overwritten with zero values if they contain a value different to zero.

2.5.3 The Pump and similar Concepts

A well-known means to limit covert timing channels in MLS systems is the so called *pump* by Kang and Moskowitz [KM93]. The pump is a system located between a system A of a lower security level (LOW system) and system B of a higher security level (HIGH system). The communication between A and B is only feasible via the pump device. The traffic from the LOW system to the HIGH system is cached and afterwards forwarded to the HIGH system (this process is called *flushing*).

To limit write-downs, the traffic from the HIGH to the LOW system is not forwarded in an unaltered way. The pump only allows acknowledgement messages from the HIGH system to the LOW system, i.e., no other information than acknowledgement information can be sent through the pump from HIGH to LOW. Thus, covert *storage* channels are prevented. A covert *timing* channel is feasible since the HIGH system can try to signal the LOW system confidential information by the number of acknowledgement messages it sends per time slice. Since the pump also limits the number of acknowledgement messages per time slice, the timing channel's capacity can be limited. A pump version called the *network pump* is available and capable to limit covert timing channels in network environments with multiple connected systems.

The theoretical model of the so-called *quantized pump* was later proposed by Ogurtsov et al. in [OOS+96]. The quantized pump allows the configurable covert timing channel limitation as well as a dynamically growing/shrinking buffer space for the pump

that automatically adapts to the performance requirements of the data transmission (increasing/decreasing acknowledgements by the HIGH system per time slice T result in increasing/decreasing transmission rates from LOW to HIGH).

Similar concepts to the pump exist and are mentioned in [OOS+96]: The *One-way Link* (or *Blind Write-Up*) is a simple one-way communication channel from LOW to HIGH. No communication from HIGH to LOW is feasible. The *Upwards Channel* is an improvement of the One-way Link that introduces a buffer between LOW and HIGH since it is thinkable that LOW sends data faster than HIGH can process the received data. However, the Upwards Channel is not designed to handle buffer overflows.

Another technique mentioned in [OOS+96] is called the *ACK Filter* and forwards messages from LOW to HIGH but allows only acknowledgement messages from HIGH to LOW. A limitation as foreseen by the pump is not part of the ACK filter concept. The so-called *Store And Forward Protocol* (SAFP) enhances the ACK filter concept by introducing a buffer between LOW and HIGH. A message from the LOW system to the HIGH system is cached in the buffer. SAFP directly acknowledges the data to LOW before forwarding it to HIGH. Afterwards, the data is forwarded to HIGH. If no acknowledgement is received from HIGH, SAFP forwards the message again.

2.5.4 Further Anti-Covert Channel Means

Besides the already mentioned anti-covert channel means, a number of additional techniques exist of which an overview shall be provided.

Murdoch discovered the detectability of TCP ISN-based covert channels since the *NUSHU* proof of concept-code generates ISN values with another distribution than the original Linux kernel's ISN generator [Mur07].

Tumoian and Anikeev developed an ISN covert channel detection method based on trained neural networks [TA05]. The training phase is host-dependent since different operating systems and kernel versions produce different ISN values. After the training phase is finished, test traffic can be evaluated using the neural network. The false positive rate for the method is less than 0.5% and the false negative rate is between 5% and 10%.

Fadlalla presented the *spurious processes approach* to limit covert channels in MLS systems [Fad96]. The goal is to limit covert channels based on the alternation of shared resources by introducing special processes called *spurious* processes. Each time, a normal (i.e., non-spurious) process accesses a shared resource accessed by another non-spurious process before (e.g., process B tries to create a file X that was previously accessed

(deleted/created/modified) by a process A), a spurious process is introduced to access/modify the resource. The context-switch to the spurious process is made before the context-switch to B will be done. The goal of the spurious process is to provide uncertainty regarding the state of the shared resource to the non-spurious process B that tries to access the resource. If the spurious process modifies (e.g., deletes/creates) the file X, B cannot in any case be sure which process modified the file's status.

In 1991, Hu invented the concept of *fuzzy time* to limit covert timing channels between virtual machines [Hu91]. Between virtual machines, covert timing channels can be realized by measuring the timing of events. Hu separated the host system's timing events from the timing events of the virtual machines, i.e., he made time settings of the virtual machines slightly different to the current time of the host system. If a timer interrupt occurs on the host system, the notification of the virtual machines is slightly delayed and the measurements for the covert timing channel are thus inaccurate. The time at which an actual event is taking place on the host system is called the *event time* while the delayed time at which the virtual machine is informed about the event is called the *notification time*. Since covert channels between *different* virtual machines shall be limited, the notification for each virtual machine is delayed by a randomized value. Thus, to prevent corrupted transmissions, the covert timing channel's data transmission rate has to be reduced in a way that the timing delays do not affect the covert channel's timing measurements in a significant manner. In [Hu91], a configurable fuzzy time as well as an on/off switch for the fuzzy time is proposed.

Besides limitation means (such as the pump), many detection techniques for timing channels in computer networks were also developed. A selection of these techniques shall be described even if covert timing channels are not in the focus of this thesis.[4] Basically, the detection means for network timing channels record the time differences between the occurrence of network packets. These timing differences are called *inter-arrival times* or *inter packet gaps* (IPGs).

Cabuk presented an approach based on the *compressibility of IPGs* in [CBS09]. Therefore, the IPGs of a traffic are recorded and rounded. The rounded values are afterwards converted to a textual string representation S while IPG values > 1 second are deleted from the recording to filter noise. The string representation consists of a letter that specifies the number of zeros after the decimal point (e.g., B25 represents the rounded value of the IPG 0.00249). Afterwards, the compressed length C of the string S in

[4] As mentioned earlier, a detailed description and evaluation of the discussed means can be found in the author's book *Tunnel und verdeckte Kanäle im Netz*, Springer-Vieweg, 2012 as well as in the author's master's thesis.

comparison to the original length of S is calculated. Therefore, the authors utilize an existing compressor $\Im$ (e.g., *gzip*) to get $C = \Im(S)$ and calculate

$$\kappa(S) = \frac{|S|}{|C|}, \tag{2.1}$$

where $|.|$ is the string length operator.
If a covert timing channel is included in the traffic recording, the traffic patterns are more unique due to the synthetic IPG values. Thus, a covert timing channel raises the compressibility $\kappa(S)$ of a traffic recording in comparison to other traffic recordings of the same measurement environment.

Berk et al. found out that most IPG values of regular traffic (i.e., traffic without covert timing channels) can be found around an IPG value x [BGC05]. In other words, x is the IPG value linked to the highest number of packets. The number of packets with an IPG of x is called C_{max}. If, on the other hand, a covert timing channel is present, at least two different timing values are required for the communication and thus, at least two different peaks for timing values are present. The authors calculate the probability that a traffic recording contains a covert channel via

$$P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}, \tag{2.2}$$

where C_{max} is the number of packets around x within a regular traffic recording and C_μ is the number of packets with an IPG of x within the current traffic recording to be tested. Since at least two peaks (instead of one) are present for a timing channel, C_μ is usually low what results in a higher $P_{CovChan}$.

Another approach is to apply machine learning for covert timing channel detection as done by Zander in [Zan10]. Zander therefore applies the C4.5 algorithm that creates a decision tree for the classification of a given traffic recording, i.e., traffic is either classified as traffic containing a covert timing channel or as traffic without a covert timing channel.

The aforementioned behavior-based covert channel within anti-virus programs proposed by Anthony et al. is expected to be easy to prevent. The introduced signature updates *are recognized as unofficial signatures* by the used program ClamAV [ALJY12] and the authors believe that *many* unofficial signature updates can raise attention. As a prevention means, the authors propose to limit the update mechanism in a way that it only utilizes previously approved signature sources [ALJY12].

In joint work with Zander, we developed a detection method for *protocol channels* that utilizes machine learning using C4.5 as well [WZ12].[5] We therefore monitored the number of packets between protocol switches and the time between protocol switches. The accuracy of our approach is 98-99% if the protocol channel sends with a bitrate of at least 4 bits/second.

2.6 Dynamic and Autonomous Covert Channels

In the past, network covert storage channels were relatively simple communication channels which utilized a single network protocol. Within the last years, new techniques arose which enabled advanced functionality for network covert storage channels – a trend that so far did not took place for covert timing channels. These improvements for storage channels came due to the ideas we will discuss in this section.

2.6.1 Protocol Switching/Protocol Hopping

The capability to utilize multiple network protocols for a covert channel enables a covert channel to adapt itself to changing situations. For instance, a protocol X which is available in network A could be blocked in network B. If the covert channel can utilize another protocol Y usable in network B as well as in network A, the communication can still take place [YDL+08].

Another advantage of a protocol switching capability is to counter an a posteriori forensic analysis, i.e., if the hidden traffic is split in n connections using different protocols and each connection uses its own hiding algorithm. Thus, the forensic analyst has to analyze n instead of 1 steganographic hiding techniques [Wen09b] and if m of the n utilized techniques (with $n > m$) are already discovered, a forensic analyst cannot easily conclude that additional $n - m$ techniques were used.

The first covert channel able to utilize two different network protocols (UDP and ICMP) was presented by "daemon9" and called LOKI2 [dae97]. LOKI2's protocol switching is based on a manually issued user command and the feature was considered to be unstable [dae97, Wen09b].

We later presented an improvement of LOKI2 called a *protocol hopping covert channel* (PHCC) of which only a text file was published in 2007 [Wen07b] that was discussed

[5] As mentioned in the preface, the author decided to not include the protocol channel detection in this Ph.D. thesis since it is partly an outcome of his master's thesis. Thus, the detection technique is discussed as related work.

within the professional community within the same year. A PHCC is a network covert storage channel capable of switching the utilized protocol in a way which is transparent for the user [Wen09b]. Additionally, multiple protocols are used simultaneously (not sequentially like in LOKI2) and in a transparent way [Wen09b]. Thus, no user command is required to switch the utilized protocol.

A similar idea within the hacking community (a covert channel with multiplexing capabilities using a separate data and control channel) was proposed earlier by Sławiński and the Gray-World project and did focus on architectural aspects [Tea05] and reverse tunneling [Sł04]. The idea of a PHCC, on the other hand, does not depend on the reverse tunneling scenario and combines the control information and the payload of a channel within all utilized channels at the same time and can also lack a control channel, i.e., a protocol hopping covert channel *can* comprise data channels only.

In the context of network steganography, Frączek et al. later proposed a number of novel techniques of which one technique called *flow-based steganogram scattering* [FMS11] can be considered to be equal to protocol hopping covert channels. Another idea was developed for covert *timing* channels by Luo et al. and utilized multiple TCP flows simultaneously to encode a hidden message [LCC07].

Note: In Section 2.4.6, *protocol channels* were introduced, which signal hidden information by the use of a number of protocols for their message, while each protocol is linked to a bit value. *Protocol hopping covert channels* on the other hand, can utilize multiple covert storage channels and the protocol itself is not of importance for the hidden message since the hidden message is placed in the storage areas (e.g., header bits) of the network packets.

2.6.2 Control Protocols

In general, a communication protocol is *required to regulate the communication between distributed processes in a computer network* [NY85]. A covert channel-internal control protocol enhances the capabilities of a covert channel by adding typical features of regular network protocols (such as reliability or the command to start or stop a communication [Tea05, RM08a]) to the covert channel.

A first internal control protocol for illegitimate network tunnels was presented by Stødle within the tool *Ping Tunnel* [Stø09]. Regarding to the changelog file of Ping Tunnel, the feature was already implemented in the first public version 0.50 in 2004.[6]

[6] Version 0.50 is not available for download anymore but the oldest available version of Ping Tunnel (0.52) did already contain the typical protocol header and was released one month after version 0.50.

To the authors best knowledge, there was no earlier implementation of similar features available in other programs. Ping Tunnel utilizes the ICMP "echo request" and "echo reply" payload to cover its control protocol as well as its payload. The first 4 bytes of the ICMP Echo payload contain a magic byte used to identify Ping Tunnel packages, what makes the tool easy to detect. However, its main purpose is to pass firewalls and not to raise no attention. The magic byte follows the actual control protocol which contains the fields shown in Figure 2.2: A 4 byte destination IP address, a 4 byte destination port (actually, a port number only requires 2 bytes), a 4 byte state information used to indicate the message type as well as the connection state, a 4 byte acknowledgement number (of the last received packet), the 4 byte length of the payload following the control protocol header, a 2 byte sequence number[7], and a 2 byte identifier field to handle multiple connections at the same time.

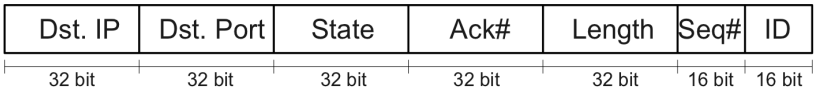

Figure 2.2: The header of Ping Tunnel's internal control protocol as presented in [Stø09].

To prevent packet re-ordering in their port knocking-based covert channel, deGraaf et al. proposed to put sequence numbers in UDP destination port information [dAJ05]. Therefore, the 16 bit destination port is split into a data part and a part for the sequence number. This very simple control protocol does not prevent packet loss but can be considered the first control protocol for covert channels discussed in the research community.

In 2008, Ray and Mishra presented the second control protocol designed for network network covert storage channels which was also implemented in ICMP Echo messages [RM08a, RM08b]. Their protocol is much more space-efficient than the protocol of Ping Tunnel; it only requires 8 bits. As this thesis will show (cf. Sect. 3.5.5), the size of Ray's and Mishra's protocol can be reduced nevertheless without losing functionality.

The protocol of Ray and Mishra is shown in Figure 2.3 and contains a sequence number and an expected sequence number that can be seen as acknowledgement number (two

[7]The difference in the size of the acknowledgement number and the sequence number was initially spotted in the context of the work for this thesis. As a result, the author of Ping Tunnel plans to unify the size of both elements in a future release what will allow 2 bytes of additional payload per packet. Besides, two bugs (one of them leading to a program crash) were also found while working on the thesis and reported to the *Ping Tunnel* developer (fixed in the release 0.72 and in the upcoming release 0.73, respectively).

Figure 2.3: The header of the protocol presented by Ray and Mishra [RM08a].

bits each). One flag can indicate that payload is attached and another flag indicates that the packet contains an acknowledgement.

The field *expected sequence number* does actually contain the sequence number of the received packet, not the *expected* sequence number, i.e., if the field contains the number *01*, the packet with the sequence number *1* was successfully received, but the sequence number *10* (2) is expected for the next packet.

The authors *believe* that a two bit sequence number (i.e., four states are possible) is sufficient for a covert channel communication to prevent the re-usage of a sequence number [RM08a]. In comparison to Ping Tunnel's 16 bit sequence number, the 2 bit sequence number of Ray's and Mishra's protocol is small, but since their protocol only sends out new packets after the latest packet got received and acknowledged, i.e., a *stop-and-wait automatic repeat request* (ARQ) protocol is used, sequence numbers cannot be used for two packets at the same time. However, waiting for the acknowledgement of a packet before sending out the next packet can be considered a slow process. Since ARQ only requires a 1 bit sequence number, but the authors provide a 2 bit sequence number, an improved variant of ARQ could be used where multiple packets can be send sequentially before an acknowledgement is required. However, while the authors mention the possibility to use the improved versions *Go-back-n ARQ* and *selective repeat ARQ*, respectively, they do not implement them. Ray and Mishra motivate their choice for the basic stop-and-wait ARQ algorithm with the fact that the improved algorithms can lead to more re-transmissions of packets in case ICMP rate limiting is used[8], what can raise more attention.

The last two bits are used to specify whether a covert communication starts or ends with the current packet. The authors also describe the application of encryption algorithms for the covert channel. Since the goal of covert channels, as a discipline of the information hiding research area, is only to stay hidden, encryption is not within the focus of this thesis.

[8]With ICMP rate limiting, as for instance provided by modern CISCO devices and by the Linux operating system, the number of ICMP messages of the same type per time slot can be limited [FS11].

In the context of protocol hopping covert channels, we proposed the idea of a minimized control protocol header design [Wen09b, WK11]: The smaller the control protocol header's size is, the more network protocols it can be embedded into.

Given two network protocols X and Y. The protocols comprise areas (A_X and A_Y) usable for covert channel data. Network protocol X's header provides $sizeof(A_X)$ bits of space for covert data and network protocol Y's header provides $sizeof(A_Y)$ bits of space for covert data. A control protocol's header size ($sizeof(A_{ContrProto})$) should not comprise more than

$$sizeof(A_{ContrProto}) = min(sizeof(A_X), sizeof(A_Y)) - n \tag{2.3}$$

bits to fit into the utilized areas of both network protocols while still providing n bits of space for the covert channel's payload [Wen09b]. In case of m used network protocols, it is obvious that $sizeof(ContrProto)$ should be

$$sizeof(A_{ContrProto}) = min(sizeof(A_1), \ldots, sizeof(A_m)) - n \text{ [Wen09b].} \tag{2.4}$$

2.6.3 Adaptive and Autonomous Covert Channels

Besides the development of covert channel-internal control protocols, a second goal was aimed to reach within the last years: *Autonomous* or *adaptive* network covert storage channels.

A first approach for the development of an adaptive network covert channel was presented by Yarochkin et al. in 2008 [YDL+08]. The authors developed an algorithm that is capable of switching the used application layer protocols of a covert channel. Additionally, the algorithm is adaptive, i.e., it can select usable protocols and can monitor whether a protocol is blocked or not, even if a network configuration changes.

Yarochkin et al.'s approach does not cover the problem of traffic normalization in the way as it will be discussed in Section 3.3 of this thesis: The authors filter blocked protocols but do not cover two-way modifications of packets, i.e., non-blocked but altered packets with normalized components in a bi-directional communication.

The authors support redundancy (re-sending the same content again over various channels) to counter payload transmission errors. The idea to utilize only protocols that are usual within a given network to prevent IDS alerts is mentioned but not discussed.

For the communication between two systems (called *agents*), the intersection of supported protocols between both hosts, excluding the set of blocked protocols must be determined: $P_A \cap P_B \setminus P_{blocked}$. Therefore, the communication between two systems is split into two phases, the *network environment learning* phase (NEL phase) and the *communication* phase. Within the NEL phase, the systems try to determine usable protocols for the communication between them, while the communication phase starts after at least one usable protocol was found and transfers the payload. The NEL phase will continue for the whole lifetime of the covert channel processes to adapt the covert channel software configuration to changes in the network.

Within the NEL phase, the sender sends a sequence of packets to the receiver in the hope that the receiver can identify the sequence using passive traffic monitoring. If the sequence is identified, the receiver acknowledges the successful data transmission. As this thesis will show in Section 3.3.2, the approach of Yarochkin et al. results in a two-army problem if a normalizer is present since acknowledgement messages can be modified or dropped.
The authors additionally calculate the round-trip time (RTT) for the response of packets of an *already used* protocol what is done to handle network problems (e.g., packet drops). In addition, a *survival score* is calculated for all already used protocols. The better the acknowledgement rate for a protocol is, the higher the survival score of the protocol. To prevent that protocols, which were considered as blocked, will stay unused after the blocking was administratively uninstalled, a randomized protocol usage is introduced from time to time.

A similar approach to the one developed by Yarochkin et al. was presented by Li and He in 2011 who developed a so called *autonomous covert channel* [LH11]. The idea to calculate survival values for covert channels is similar to the idea of Yarochkin et al. Like Yarochkin et al., Li and He do not describe the details of the acknowledgement channel that informs the sender about the success of transmissions, i.e., the authors did not deal with the two-army problem, nor is a covert channel-internal control protocol mentioned or described. The acknowledgement channel including the feedback information for the success of a transmission was left for future work by the authors [LH11].

A third approach that should be mentioned in the context of adaptive covert channels was presented by Acosta and Medrano. The authors propose a *blending covert method*

that consists of three steps [AM11]: A monitoring phase that analyses received traffic passively (this idea is very similar to the previously discussed idea of Yarochkin et al. but takes payload fields and media streams, such as VoIP traffic, instead of protocol headers into account); a selection phase that is used to identify areas in network streams that could be used to embed hidden information into; and a transfer phase in which the actual embedding, transfer, and extraction of the hidden information is taking place. The third phase does also include a control protocol using synchronization information and a checksum. In [AM12], the authors additionally discusses the optimal placement of hidden information into network data and conclude that traffic streams can be considered as most suitable (*due to high update rates and high complexity* [AM12]).

A fourth approach that did not discuss the mentioned related work of Yarochkin et al. or of Li and He was published in December 2012: Swinnen et al. proposed to monitor network traffic to adapt the traffic generation behavior of their covert channel to the traffic statistics of a given network [SSP+12]. Therefore, passive traffic monitoring was used and daily traffic observations were made since traffic patterns can change on a daily basis [SSP+12]. As we will discuss in Chapter 3, passive monitoring cannot solve the two-army problem within the NEL phase.

Another trend that can be seen as part of autonomous or adaptive covert channels, that so far received only limited attention, is the dynamic routing in covert channel overlays. The problem was initially discussed by Szczypiorski et al. [SMM+08] and was based on the random-walk algorithm. Later, Backs introduced the concept of *Quality of Covertness*, a distinction between covert channel-aware and covert channel-unaware systems (*Agents* and *Drones* — an idea based on [Mem07]), and optimized link-state routing (OLSR) into covert channel overlay networks [Bac12] that was published as joint work with control protocol aspects of this thesis (cf. Chapter 3, especially Section 3.5.7) in [BWK12].

2.7 Building Automation Systems

Building automation systems (BAS) take care about the control, measurement, and management of hardware systems inside a building as well as they handle the communication between these hardware devices [MHH09, GPK10]. The concrete purpose of a BAS is linked to its category (e.g., a factory BAS does not require ambient lighting but such ambient lighting can be of use for a BAS within a private home). Soucek and Zucker mention four BAS categories [SZ12]: commercial buildings (e.g., offices),

institutional buildings (e.g., schools), industrial buildings (e.g., factories), and residential buildings/apartments. Smaller building automation systems used in homes are also called *home* automation systems.

BAS arose in the 1950's and were based on pneumatic elements at the beginning [WS97, KNSN05]. The first BAS comprising its own computer component as well as the capability to log BAS events (e.g., the current temperature in a room) came up in the 1960's [WS97]. Besides the traditional tasks (heating, ventilation, and air-conditioning, or *HVAC*), various new use cases were defined for BAS, such as ambient lighting, safety systems (fire detection, alarm systems), security systems (access control), and elevator control. Newer approaches comprise the use of weather predictions for the optimal use of renewable energy sources to operate a building as well as the use of intelligent algorithms to adapt the BAS to the demands of the inhabitants [SC99, SZ12], the work on user-tailored and ambient interfaces (as we discussed in [RWMA12]), the research and realization on flexible and multi-agent architectures [SC99, KRS+11] and the shift from wired to wireless BAS [Ega05].

In general, the current tasks of BAS comprise the goals of **cost reduction**, **comfort improvement**, and the **improvement of the energy efficiency** [SZ12]. These three aspects will be discussed separately:

The integration of a BAS in an existing or new building is costly, especially, if BAS wires have to be installed in walls. Another reason for higher costs comes from the need for a reliable hardware since hardware components are required to work error-free over decades inside the building. Wireless installations (e.g., professional systems based on ZigBee [Ega05] or smaller systems like HomeMatic (*http://homematic.com*, [Rie10]) and RWE SmartHome (*http://www.rwe-smarthome.de*)) can be comparable cheap, especially due to the reason that no specialist is required for their installation and configuration [Ega05]. The overall costs of a BAS integration can also be considered lower if the BAS contributes to a reduced energy consumption of the building [MHH09].

The improvement of comfort based on BAS is approached from various viewpoints – especially from researchers in the field of medicine, human computer interaction as well as architecture and design. Comfort in a building comprises different aspects, such as an optimal lighting (visual comfort) [KNSN05], BAS-operated elevators [SR09], thermal comfort, or remote controllable hardware components. An important field in the context of comfort is the so-called *ambient assisted living* (*AAL*) [KBR+07]. AAL aims on providing accessibility and assistance for handicapped and elder persons and allows them to stay longer in their own homes before being forced to move to a nursing

home. BAS play a central role for the realization of AAL environments [LdILB+09].

A motivation to utilize BAS to reduce the energy consumption within buildings was initiated by the oil crisis in 1973 [WS97, KNSN05]. In the "20-20-20 targets" published by the European Commission, significant reductions in the energy consumption of buildings are claimed what additionally motivates the use of BAS since approx. 40% of the EU's energy consumption is caused by buildings [SZ12].

As mentioned by Wong and So as well as by Simpson and Riesberg (Eds.), lighting is one of the largest reasons for a high energy consumption [WS97, SR09] and thus, an energy-optimized lighting controlled by a BAS can lead to a significant energy reduction. Therefore, a BAS can try to utilize as much daylighting as possible, i.e., to only add the required synthetic lighting on demand [SR09].[9] Work to increase the awareness of inhabitant's energy consumption in combination with a BAS was also done. For instance, we proposed the integration of an augmented mirror that displays the current energy consumption as well as an augmented calendar to schedule BAS events [RWM+11].

2.7.1 Building Automation Technology

Building automation environments are organized in a hierarchical system with three levels [KNSN05, SZ12] (shown in Figure 2.4):

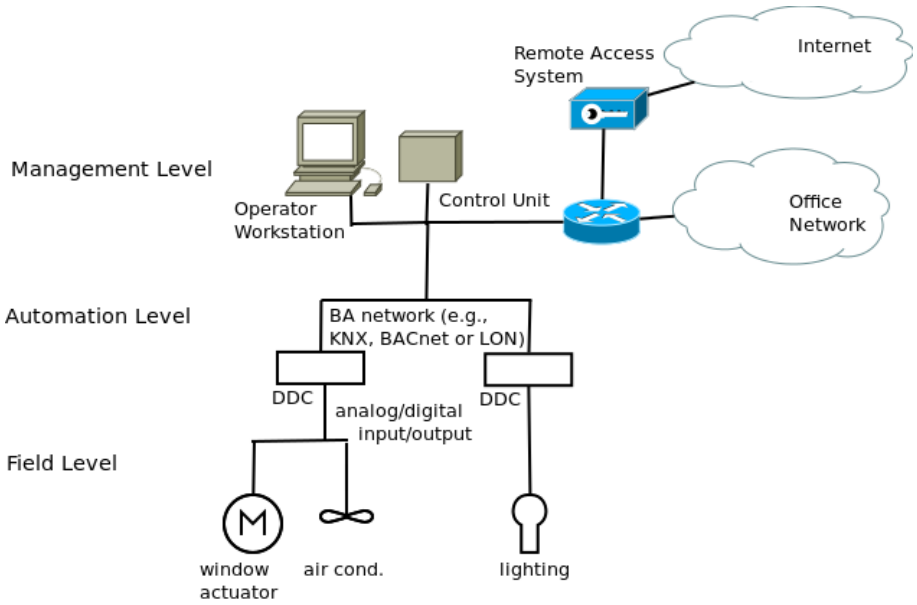

Figure 2.4: Typical hierarchy in a BAS.

[9] The use of more daylight can comprise the side effect to additionally heat up a building what can result in higher cooling costs and thus, can decrease the energy-efficiency again [SR09].

On the *field level*, different hardware devices split into two categories, namely *sensors* and *actuators*, can be found. Sensors measure values and report them while actuators are capable of changing a state (e.g., opening or closing a window or turning on/turning off the lighting in a room). The sensors and actuators are connected to a so called *direct digital control* (DDC) at the *automation level*. A DDC comprises a number of digital and/or analog inputs and outputs to communicate with the sensors and actuators. DDCs are programmable embedded systems (graphic instead of textual programming languages are used in many cases [SZ12]) that are connected to the building automation network to communicate with other DDCs as well as with the management level. In many cases, standardized protocols (such as BACnet or KNX) are used on the automation level but sometimes (especially in older buildings) proprietary protocols are used. On the *management level*, monitoring and (remote-)control of the BAS are accomplished. The management level can be connected to the enterprise network as well as it can be connected to the Internet using gateways for remote administration purposes (therefore, BAS protocols are usually tunneled via UDP or TCP).

In wireless building automation systems, a central control unit (CCU) can be found (e.g., for the AdHoco ZigBee system as well as for the eq-3 HomeMatic). Such a CCU combines the automation and management level since it directly communicates with the sensors and actuators as well as it comprises a management interface (usually web-based).

In today's building automation environments, a number of different protocol suites can be considered popular, especially BACnet, LonWorks, and EIB/KNX. These systems comprise multiple communication layers and various protocols, a different terminology and are not necessarily inter-connectable with other BAS. However, in comparison to old BAS environments, BACnet, LonWorks and EIB/KNX are based on open specifications. While some systems are of higher importance for the management level, other systems are of higher importance for the automation level (e.g., Ethernet could be used at the automation level while BACnet could be used at the management level within the same building).

2.7.2 Inter-operability

The early protocols for building automation networks were proprietary and installed as separate networks [Fis12]. As building automation networks and enterprise intranets as well as previously separated building automation networks became interconnected, these proprietary protocols lead to problems regarding the inter-operability of the building

automation networks and their interconnection caused the need for separate management and bus-systems.

To overcome the problem of incompatible protocols, new protocol standards such as BACnet (which will be discussed later in detail) were developed to serve as a unified communication stack comprising a subset of the OSI layers. Such inter-operable protocol suites are widely accepted and pushed by the industry and are thus used by many vendors. However, a number of proprietary/incompatible protocols are still in the market and products which require these protocols are popular in the home automation field since they are usually comparable cheap.

Besides, middleware solutions as well as proxies and gateways were developed to address the inter-operability problem. *Middleware/architectural* solutions such as developed by [Sno03, MRH+08] provide separate communication interfaces for various connected BAS and provide a unified programming interface for application developers, i.e., they abstract from low-level vendor-specific details. *Proxies* and *Gateways* are used to directly connect incompatible automation networks on the automation level or to connect different incompatible automation level protocols with the management network that uses only a single protocol [KNSN05]. Additionally, gateways can be used to connect the BAS network with the Internet (or with the intranet, if the management level does not use TCP/IP) for remote administration. Therefore, the BAS protocol is usually encapsulated using UDP – as done in the case of BACnet/IP (BACnet encapsulated in UDP) and KNXnet/IP (KNX encapsulated in UDP) [MHH09, SZ12].

2.7.3 Security in Building Automation Systems

As mentioned previously, early BAS were designed as stand-alone systems and no interconnection of these systems was foreseen. Later, BAS networks were interconnected with intranets as well as with the Internet what stressed questions regarding the security of these BAS [SZ12].

For BAS attacks, the adversary can either be in the building or not (i.e., performing a direct physical attack or a remote-attack) [GKNP06]. The attack of BAS can be motivated through different goals such as (but not limited to):

- Getting physical access to a building by attacking the physical access control (PAC, [Hol03, RIMT06]) component of the BAS.
- Getting access to the TCP/IP intranet of an enterprise via the building automation network [SZ12, Fis12].

- Generally attacking other companies (e.g., turning on the lighting in the competing company's basement every night to raise their energy costs or disabling building functionality due to DoS attacks [GKNP06]).
- Terrorist attacks [Fis12] (e.g., disabling fire alarms before placing a fire in the building [Hol03]).
- Obtaining information about inhabitants/employees (not only by external persons but also by inhabitants/employees [Hol03]) within a building by getting access to the monitoring system, direct sensor access (e.g., in [ELP+12] voice recognition, indoor localization, and sensors to detect social interaction were used for a social study in the context of pervasive computing) or via side channels (discussed in this thesis).[10]
- Realizing intentional data exfiltration using building automation (sub)networks via covert channels (discussed in this thesis).

A detailed survey on BAS security was published in [GKNP06] and [GPK10]: Granzer et al. presented a hierarchical attack model for BAS in which they distinguish attacks on the BAS network as well on the devices itself. Network attacks comprise the interception of a communication (using a sniffer), the modification of network data (via man-in-the-middle attacks), the interruption of the communication (e.g., denial of service attacks and the redirection of traffic) and fabrication attacks (i.e., the generation of new malicious frames and replay attacks) [GPK10]. For device attacks, the authors differ between software-side attacks (e.g., code injection), side channel attacks (based on timing, power, and fault behavior analysis), and physical attacks (e.g., replacement of devices) [GPK10]. While Granzer et al. were the first authors to *mention* side channel attacks, this thesis is the first to present such side channels in BAS.

Denial of Service (DoS) attacks on BAS as well as countermeasures were discussed in [Hol03, GRK08]. Other typical attacks such as password guessing, port scanning as well as war dialing to access the BAS must also be considered and since BAS are usually

[10] The study by Efstratiou et al. has shown that privacy plays a significant role for people involved in indoor monitoring as they, for instance, only felt okay being observed (with publicly available results) if they were at a high position in a ranking generated by the observation system, while they otherwise feared the observation results [ELP+12]. For instance, if a presence sensor detects that a person is currently not at his/her desk, the observer could think that the person is currently not working, even if the person is working at home [ELP+12]. 14% of the observed people in the study of Efstratiou et al. did also change their behavior after they were monitored while being aware of the monitoring. In [Gös10], the behavioral change of persons, which are aware that they are observed, is also mentioned.

operated by non-security experts or janitors, misconfigurations that leads to security holes must be considered as well [Hol03].

As mentioned by Holmberg in [Hol03] and discussed by us in [RWMA12], BAS are not hardened with the same quality of protection means as today's operating systems and mature software systems. For instance, BAS web-interfaces could be attacked by XSS since no protection means were taken into account in the past.[11] Another problem related to wireless BAS is that – at least in some cases – no encryption for the communication between central control unit and devices is applied or that the applied encryption algorithms and protocols are not published as well as it is possible that own implementations of cryptographic algorithms contain errors. We discussed the analysis of the broken encryption used for the HomeMatic system that was discovered by the *cirosec GmbH* in [KW13] as well. If thieves can gain information about these protocols and their weaknesses, they can open windows/doors remotely to get access to the building as well as they can monitor the building.

Since BAS networks usually run a different protocol than TCP/IP, the communication from TCP/IP networks such as the Internet is only possible via gateways. The security of a BAS network thus depends on the protection of the connecting network (e.g., whether a firewall or IDS is present) [Hol05].

A problem mentioned by Granzer et al. as well as by Nixon et al. is the performance of the available hardware components: BAS components are operated for decades and cannot be easily modified without working on the building's walls as well as without integrating newer hardware parts (e.g., better performing chips and additional memory). Many existing BAS components are not powerful enough to apply today's cryptographic algorithms and the upgrade of only a selected number of hardware components is problematic since protocol modifications with cryptographic content could cause undefined behavior in the remaining hardware components [NWET05, GKNP06, GPK10]. Granzer et al. were the first to provide a solution for that problem by introducing a backward-compatible IPSec-like protocol for EIB called *EIBsec* [GKNP06]. EIBSec-capable devices can use the security features of the protocol while the other hardware components are not negatively affected by the use of EIBSec. The same workgroup did also propose sandboxing environments for BAS in [GPK10]. Instead of EIBSec, BACnet now provides similar features, as we will discuss in the next section.

[11] The author discovered a XSS vulnerability in the HomeMatic web-interface that got fixed by the vendor but was not made public. Additionally, as we reported in [KW13], it is still feasible to obtain root access to the current firmware by exploiting an old *lighttpd* bug.

Another approach to integrate security in BAS is to provide a secure middleware that prevents direct low-level hardware/protocol access (see previous section). For instance, Maña et al. presented the *Hydra* middleware that comprises role- and attribute-based access control (ABAC/RBAC) support [MRH+08].

Since *safety* was intensively studied in the BAS domain, research was also done to evaluate features with can be used for both safety and security in BAS at the same time. Novak et al. therefore developed an approach that can be applied in the pre-design phase of a BAS to determine requirements which can be used to achieve both goals at the same time [NTP07].

The previously mentioned AAL technologies try to assist elders and handicapped persons in their everyday life within a building. Therefore, building automation systems can contain health information (e.g., motion sensors, dust sensors, pulse and blood oxygenation sensors [WVD+06]). Health information collected in a building must not necessarily be linked to the BAS and can be secured as proposed by [MND12][12], but the linkage of BAS and eHealth monitoring provides the positive side effect of a more comprehensive monitoring environment for elders and is, for instance, used in [KHE11]. Therefore, information stored in BAS components (e.g., a database system within a BAS component such as the CCU) must be additionally secured and cannot be considered secure if the (usually simple) BAS protection means are used exclusively.

As the number of BAS security-related publications indicate, it can be observed that more and more existing security features from other IT areas became adopted to the field of building automation and, as we mentioned in [RWMA12], it can be predicted that this development continues.

BACnet-specific security aspects will be discussed in the next section.

2.7.4 BACnet Overview

BACnet was developed by ASHRAE[13] and aims on allowing the interconnection of different BAS [Hol05]. The current version of the BACnet standard is 135-2010 and was updated by different so called *addenda*. The protocol suite can be used on all three layers of the building automation hierarchy that was introduced in Section 2.7.1, however, its main use lies in the management layer since BACnet can interact with various other BAS

[12] Milutinovic et al. place a base station in the building that is connected to wearable monitoring components of patients connected to an eHealth system but not to the BAS (confirmed by the first author of the paper).

[13] *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.*

protocols (e.g., BACnet can operate based on KNX as well as on LonWorks) [MHH09]. The organizational structure of BACnet is based on so called *objects* and each device comprises such objects (e.g., a temperature measuring device could contain a number of analog temperature sensor objects used to measure the temperature in a number of different rooms) [MHH09].

Objects comprise different *properties* (e.g., the object's type, its value, and its status) [MHH09]. A property can either be readable, writable, optional, or can comprise multiple of these attributes (e.g., it can be optional as well as readable) [MHH09]. To support different kinds of properties, BACnet properties can be of different *data types*, e.g., boolean values, (un)signed integer values, and strings [MHH09]. Figure 2.5 visualizes the concept of BACnet objects.

BACnet messages use these objects to identify the place and type of actions to be taken (e.g., what information is requested or which action is requested to be done at which object). Therefore, BACnet objects comprise a 32 bit *Object_Identifier* property (consisting a 10 bit object type as well as a 22 bit object instance number) [MHH09].[14]

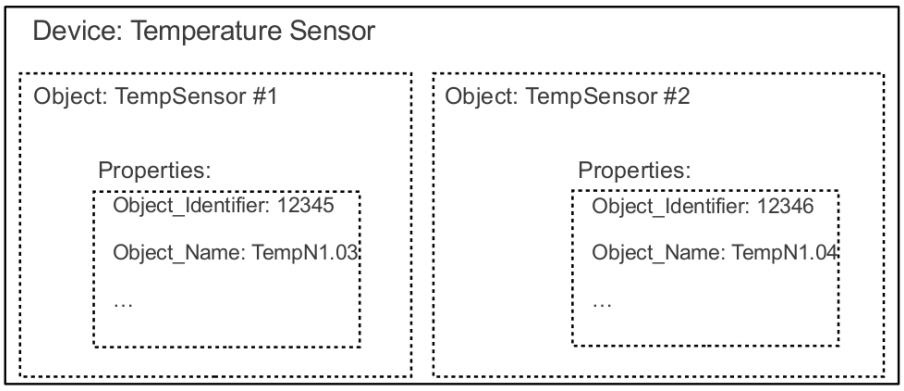

Figure 2.5: A temperature sensor device with two analog input objects comprising different properties.

If a BACnet device interacts with another BACnet device, it uses so called *services*. For instance, the *ReadProperty* service requests the value of an object's property and the *WriteProperty* service issues a change in a property's value of an object. Thus, if the "Object_Type" property of an object shall be read, the ReadProperty service requests the *service parameter* "*Object Type*" that refers to the property "Object_Type". A response with the property's value is sent back as response to the requesting device.

[14]The BACnet design differs between "Object_Identifiers" and "Object Identifiers" (without underscore). Namings with underscores represent object properties (the "Object_Identifier" is a typical object property (cf. Figure 2.5) while the "Object Identifier" is a service parameter [Gro09]).

A response for a service request is not always sent since BACnet supports acknowledged as well as unacknowledged services. Acknowledged services demand an acknowledgment while unacknowledged services do not necessarily require a response.

BACnet differs between *clients* and *servers*. Clients request information from the previously mentioned BACnet services, while servers provide these BACnet services to clients. In addition, BACnet clients can *subscribe* at a server for a value change. In such a case, a client will automatically be informed by the server if a value changes.

BACnet supports routing based on *network layer* information tuples {(destination network ID, destination MAC address), (source network ID, source MAC address)} but these tuple fields are only used for non-local message transfer as they are optional components of the BACnet network layer header [Gro09]. To address a BACnet object at the *application layer* within another network, it is adressed via a tuple (destination network ID, Object ID).

The BACnet layer model comprises only 4 of the 7 OSI layers: The physical, the data link, the network and the application layer [MHH09]. The application layer can provide the functionality of the OSI layers 4-7, if required [MHH09]. The BACnet protocol stack comprises various protocols. For this thesis, only the network and application layer are of importance since Chapter 5 will exemplify selected covert channel techniques on both layers. A detailed discussion of all four layers as well as their protocol headers can be found in [MHH09] as well as in [Gro09].

BACnet Broadcast Management Devices

An aspect that is important to understand BACnet is its broadcasting design: BACnet devices send broadcasts to obtain information about network devices (e.g., to detect routers – similar to ICMP router discovery) and to propagate information to other devices (e.g., event notifications [New10]) [MHH09].

When BACnet is encapsulated in UDP (called BACnet/IP) to connect different buildings over the Internet, IP should not and cannot broadcast the BACnet information to all systems within the IP network or the Internet. Therefore, so called *BACnet/IP Broadcast Management Devices* (BBMDs) were introduced: A BBMD is a device within the local BACnet network that listens for BACnet broadcast messages. If it receives a broadcast message from another device, it embedds the message into UDP and forwards it to the local TCP/IP router [Gro09]. The TCP/IP router sends the packet to other BBMDs using TCP/IP while the receiving BBMDs decapsulate the BACnet messages and broadcast it to the local BACnet network [MHH09]. By using these BBMDs, BAC-

net broadcast messages within one building can be broadcasted in a remote building connected via a TCP/IP tunnel as well.

BACnet Security Aspects

BACnet provides authentication, integrity, confidentiality, and freshness (i.e., random number generation) features since years [ST03, GKNP06]. In the past, different flaws in BACnet and within its environment (i.e., in hardware components) were discovered, including the possibility of man-in-the-middle attacks and the feasibility of replay attacks [GKNP06] as well as buffer overflow attacks, weak/default passwords for hardware components, and a lack of limits for login failures [Fis12].

To increase the security of the BACnet protocol suite, the ASHRAE organization founded a network security working group [Fis12] and later published the addendum 135-2008g to add advanced network security features to the BACnet standard ([ANS10]). The BACnet committee developed a number of goals to improve the security in BACnet (e.g., to counter replay attacks using message IDs, to prevent redirection, spoofing, and denial-of-service attacks, to use advanced signatures and encryption (e.g., AES instead of 56-bit DES), and to consider the key distribution process in BACnet [Hol05, ANS10]). Besides, the committee provided the idea to introduce a user authentication as well as two different network trust levels [Hol05, ANS10]: *Trusted* network environments (they apply physical and/or protocol level security, i.e., encryption or signatures) and *non-trusted* network environments (they apply neither physical nor protocol level security).

Another security-improving development for BACnet is the *BACnet firewall router* (*BFR*) [HBG06]. The BFR provides basic routing and filtering capabilities as well as NAT and acts as a BBMD to connect separated BACnet networks using BACnet/IP.

BACnet addendum 135-2008g additionally added a feature called *data hiding* to BACnet. This feature should not be understood as network steganography and actually means to only provide a subset of the available information if the requesting instance is not allowed to access all available information [ANS10]. For instance, if a property is requested, a *secure device*[15] can return one of different possible error messages (e.g., ACCESS_DENIED or READ_ACCESS_DENIED) if the requester is not allowed to access this information [ANS10]. If an array is requested (or if service results are requested), only those array elements (or information elements of the service) are provided to the requester that are conform to his access permissions [ANS10].

[15] A secure device is not *required* to be located within a *secure network* (previously introduced) but applies its security features in an end-to-end manner together with other secure devices.

2.8 Summary

This chapter introduced the concept of multilevel security (MLS) and as well as the information hiding research area. Covert channels form a discipline within the information hiding area and were introduced in detail. As this thesis focuses on covert storage channels, these channels built the main aspect of this chapter. Therefore, the state of the art regarding storage channels within the different network layers was presented and the detection, prevention, and limitation means for these channels (including timing channels) were discussed.

Advanced covert channel techniques (e.g., protocol hopping, the related work on internal control protocols, and autonomous covert channels) formed the end of the covert channel introduction.

In the second part of this chapter, the area of building automation was introduced including its security aspects. An overview of the BACnet protocol was given since a basic understanding of the protocol suite is required for Chapter 5.

3 Control Protocols for Storage Channels

This chapter introduces an extension to the existing covert channel terminology for providing a better distinction between different view-points of the protocol engineering aspect.

An overlay network is a network with its own topology on top of an existing network infrastructure, the *underlay* network [WBZ10]. When an initial covert channel overlay connection is to be established, a two-army problem occurs, which is presented in this chapter as well as solutions to minimize the problem. Thereafter, the idea of mobility and backward-compatibility within covert channel overlay networks is explained.

The attention to be raised by a micro protocol can be decreased in two ways: By shrinking the size of the micro protocol in order to manipulate as few bits of a utilized protocol as possible as well as by ensuring that the micro protocol does not violate the rules of the underlying protocol. Depending on a user's goals, he can apply one of our two approaches to optimize for a low-attention operation via conformity or via a minimized micro protocol. This chapter therefore presents a six step design technique to optimize covert channel-internal protocols in a way that prevents attention-rising protocol designs using formal grammar to ensure that a micro protocol is conform to the utilized protocol. Afterwards, a means called *status updates* to minimize the size of covert channel-internal protocols is presented as well.

Figure 3.1 provides an overview of the topics discussed in this chapter (as well as their relation to each other), excluding the terminology and scenario aspects of the first two sections.

3.1 Micro Protocol Terminology and Motivation

As discussed in Chapter 2, a network covert storage channel is embedded into selected areas of network packets. Like with existing network protocol stacks organized in layers,

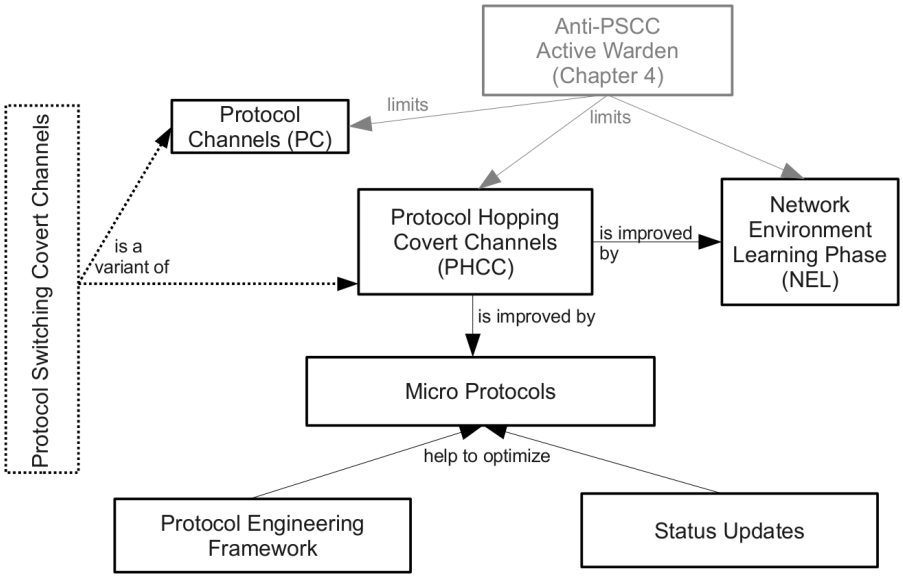

Figure 3.1: Topics and their relations for Chapter 3 (excluding terminological asepects as well as the scenario discussion; grey topics will be explained in the next chapter).

a covert storage channel-internal control protocol can be seen as an additional layer within such a selected area. To ease the understanding of the following sections, we present a new terminology for covert channels based on the following terms.

We use this terminology to help the reader to distinguish between the underlying protocol itself and the area used for the covert operation:

- **Underlying Protocol:** This term refers to the protocol utilized by a network covert storage channel. If, for instance, the covert storage channel is embedded into the ICMPv4 type and code, the underlying protocol is ICMPv4.

- **Cover Protocol:** As already basically proposed in [Wen09b] (but without the terminological context and detail), we combine a network protocol's utilizable areas to a single logic area. In this thesis, the combined area is called the *cover protocol* since it is the subset of the utilized *protocol* used to place the hidden information in. This term is analog to the term *cover-<datatype>* that refers to the data used to place the hidden information into (as defined in [Pfi96]). The cover protocol cannot contain areas which are not part of the underlying protocol. On the other hand, it is possible that the cover protocol contains unused areas, i.e., the hidden information embedded into the cover protocol may not require all cover protocol

space in each packet.[1] We propose the idea to combine cover protocols of protocols from multiple layers to a single virtual cover protocol. To combine multiple layers, all protocols of the different layers must be transmitted within a single packet and all layers must be sent to the receiver. For non-local environments, a user can therefore only use protocols of the Internet layer, the transport layer, and the application layer of the TCP/IP model.

The term cover protocol is introduced to ease the distinction between the underlying protocol and the utilized area within the underlying protocol.

For instance, if an option and the DF flag of IPv4, the TCP Initial Sequence Number (ISN), and the "Host:" field of HTTP/1.1 are used to embed the covert data, they together form the cover protocol. Since multiple layers are combined, all three protocols (IP, TCP, HTTP) together form the underlying protocol. Figure 3.2 visualizes the concept of a cover protocol built from multiple layers.

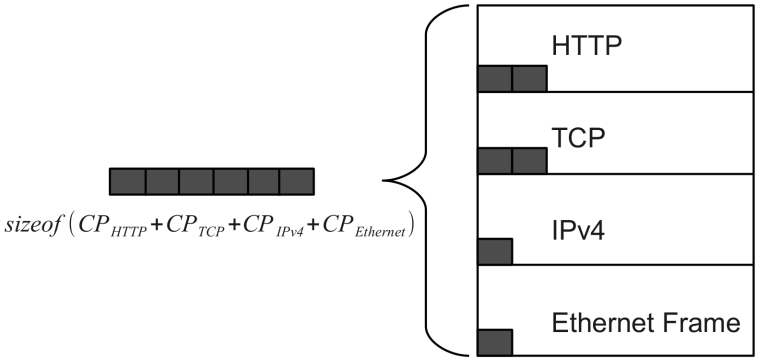

Figure 3.2: Combined cover protocol areas of multiple layers.

If a protocol hopping covert channel (as introduced in Chapter 2.6.1) is used, the available cover protocol space per underlying protocol varies. If we have n protocols $P_1 \dots P_n$ with the cover protocol spaces $sizeof(CP_1), \dots, sizeof(CP_n)$, we can select each protocol P_i with a probability $p_i = prob(P_i)$. We can then calculate the average amount of cover space per packet $sizeof(CP_{avg})$ – a value that can be used to estimate the required number of packets to transfer a message of a given length:

$$sizeof(CP_{avg}) := \sum_{i=1}^{n} p_i \cdot sizeof(CP_i) \tag{3.1}$$

[1] For instance, it is possible that the size of the payload is smaller than the size of the cover protocol, as discussed in Section 2.6.2.

- **Micro Protocol:** As mentioned earlier, we call the covert channel-internal control protocol a *micro protocol*. Micro protocols control the covert channel itself and define payload attributes. Therefore, they can basically contain any feature provided in other network protocols, such as to provide reliability or to allow broadcasting functionality. The micro protocol will be the central topic of this chapter and is placed within the *cover protocol* area.

By using micro protocols, some advances for network covert channels can be realized for which we want to present an overview, before discussing the detail aspects:

- **Mobile Overlay Networks:** By enabling a covert channel peer to use underlying protocols linked to different cover protocols, we can communicate in different environments, i.e., the covert channel can adapt itself to new situations. While for one network, protocols A and B will be suitable, they might be blocked in another network. A micro protocol can be used to overcome this problem and can therefore enable mobile covert channel usage. In a covert channel overlay with protocol hopping covert channels, each overlay hop is able to switch the utilized protocol of a path between itself and another overlay hop.

- **Backward Compatibility:** A micro protocol can contain a version number that allows backward compatible handling of legacy peers in covert channel overlay networks. An upgrade can therefore be realized step-by-step, i.e., it is not necessary to replace all peers within an overlay network at the same time to introduce a new protocol version.

  So far no covert channel protocol has been presented which can operate in overlay networks after a new protocol version including a different structure will be introduced.[2]

- **Robustness and Reliability:** While analyzing the of effects active wardens on steganographic communications, Zawawi et al. motivate robust steganographic techniques. Steganographic elements inserted in a media by using such robust techniques cannot be removed by active wardens in an easy manner [ZMU+12]. Robustness is a problem already addressed in information hiding topics outside of the covert channel research, such as in copyright marking and image steganography (e.g., to survive the re-scaling of images) [PAK99].

[2]The protocol of *Ping Tunnel* contains a version number but no work on its upgradability is available nor were newer protocol versions for *Ping Tunnel* introduced so far [Stø09] (cf. Chapter 2.6.2).

We propose to improve the robustness of network covert channels by the application of micro protocols as already mentioned in Sections 2.6.2 and 2.6.3: Using micro protocols, protocol switches are possible which can adapt to changes in the network configuration (e.g., recently blocked protocols). By introducing sequence numbers into micro protocols, disrupted packet sequences can be re-sorted at the receiver side [RM08a], which also increases the robustness of a network covert channel. In combination to sequence numbers, the integration of acknowledgement numbers, checksums and acknowledgement flags can, as in other protocols too, provide a reliable and error-detecting transfer.

- **Dynamic Routing:** Micro protocols are the foundation required to implement dynamic routing in covert channel overlay networks. Backs presented a novel dynamic routing algorithm and its implementation on the basis of a technique presented in this thesis [Bac12, BWK12]. Therefore, he used the concept of *status updates*, that will be introduced in Section 3.5 as well as *protocol hopping covert channels*.

- **Multicasting:** If IGMP-like features are included into a micro protocol, multicasting functionality can be provided for covert channels as well. Covert channel-internal multicasting can be estimated useful for collaborative communications. Collaborative steganography was already shown to be possible outside of covert channel techniques based on image steganography via *Flickr* [BK07].

3.2 Overlay Networks and Mobile Access Scenario

While research in the area of mobile computing already dealt with the problem of a permanently switching, chaotic network infrastructure, this work is (besides [Bac12] that was technically supervised by the author of this thesis and took dynamic routing into account) the first to address this problem in the context of network covert channels. A network covert channel can utilize the physical connection provided by underlying protocols including the enabled routing capabilities. However, for a covert channel, additional problems in the context of a mobile scenario must be taken into account. If a covert channel peer moves from a physical access point A to another physical access point B, its underlay connection will take care about the physical connection to the Internet (or similar networks). However, access to the covert channel overlay network is not granted since another physical access point (PAP) can change the route to the covert

channel overlay network's access point (CCAP) in a way that other filtering rules can apply. While the user can utilize network A using the IRC protocol, the same protocol could be blocked by a firewall in network B. Figure 3.3 visualizes this scenario for a smart phone user utilizing the IRC protocol to access the CCAP.

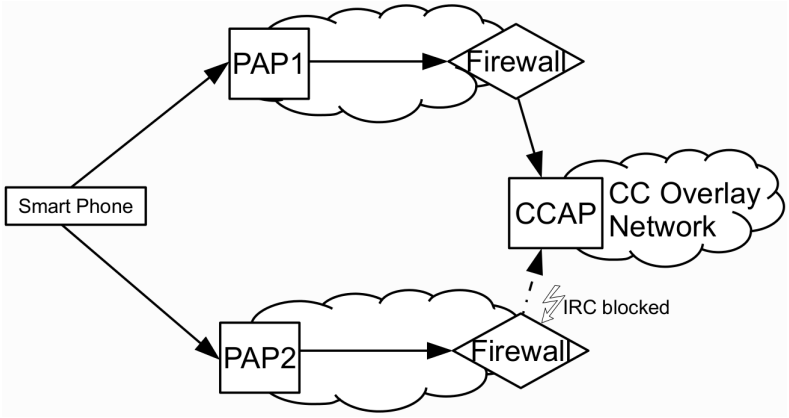

Figure 3.3: A mobile user accesses a covert channel overlay network's access points via different physical access points using IRC as underlying protocol.

Therefore, it is necessary to enable the covert channel to use multiple communication protocols as shown in Figure 3.4. A micro protocol is required to select suitable underlying and cover protocols in a matter that covert channel sender and covert channel receiver can agree on their usage, i.e., both peers must be able to understand the protocol used.

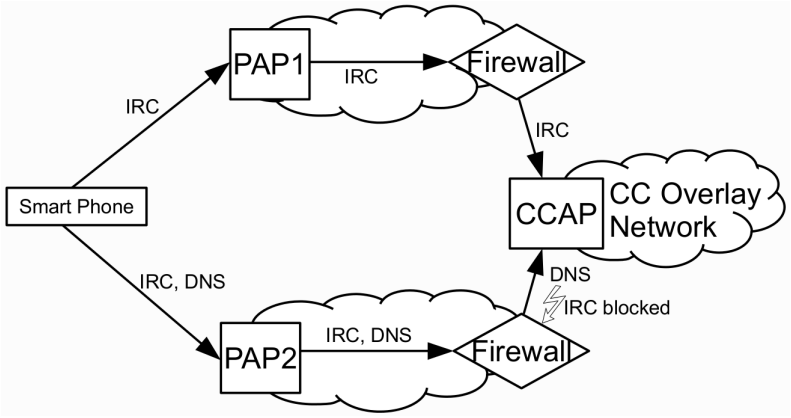

Figure 3.4: A mobile covert channel user utilizes different network protocols to access the covert channel overlay network.

It is important to mention that the problem of dynamically changing underlay infrastructure also appears within the nodes of a covert channel overlay network, and not

just with the peer that initializes a connection to the overlay network's CCAP. Not only can all peers within the network be mobile users as well, they can also face local network configuration switches. Therefore, we apply the view of being an external connector to the overlay network to all peers (each peer is an "external" connector). Thus, a peer not only wants to connect to one or more CCAPs, but also wants to keep an established connection alive and must be capable of providing a proxy and access point functionality to other peers. We will address these tasks in Section 3.3.

3.3 The NEL Phase and Backward Compatible Overlays

To select supported protocols, both peers must discover the supported protocols of each other. Therefore, they go through the NEL phase discussed in the previous chapter. The NEL phase presented by Yarochkin et al. focuses only on underlying protocols with static cover protocols two systems can exchange, i.e., they do not take combined, varying, or split cover protocols into account and do also not distinguish between underlying and cover protocol since their work lacks a terminology as ours. However, Yarochkin et al. focus on varying network configurations, i.e., changes in the set of blocked network protocols.[3]

This section presents that a two-army problem exists in determining the usable underlying protocols and cover protocols. This section also provides solutions to overcome this not completely solvable problem, and additionally takes version-depended support for cover protocols and the optimization for overlay data forwarding into account. Furthermore, we show that this process is direction dependent, i.e., the NEL has to be applied for the communication from A to B as well as for the communication from B to A.

3.3.1 A Normalized NEL Phase

Active Wardens (including traffic normalizers) drop, clear, or modify packets and parts of packets within the network traffic. It depends on their configuration, which network elements they affect. If no active warden would affect the communication between two

[3] Our paper *The Problem of Traffic Normalization Within a Covert Channel's Network Environment Learning Phase* [Wen12] was published with a distinction between *cover* and *micro* protocol but did also lack the term *underlying* protocol, which we initially changed for the paper *Systematic Engineering of Control Protocols for Covert Channels* [WK12b]. We presented the idea of using various cover protocols within a single underlying protocol already in the earlier paper *Low-attention forwarding for mobile network covert channels* [WK11].

peers which have the goal to establish a working connection, the problem would be trivial. We assume an active warden *can* be located between two peers A and B and that, for the case that a warden is present, neither A nor B know about the active warden's existence and configuration in advance. Figure 3.5 visualizes the scenario.

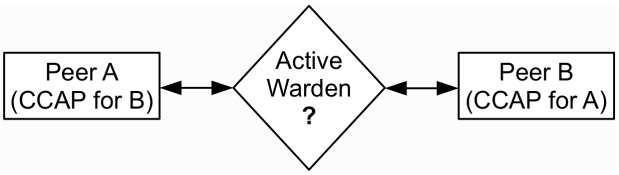

Figure 3.5: A normalizer can be located between A and B and affects the NEL phase.

A and B have to determine all usable cover protocols between them by testing whether a packet of a given cover protocol reaches the other peer. For instance, A could try to send an IPv4 packet with the reserved flag set to B. If an active warden is located on the path between A and B, the normalizer could drop the packet. Alternatively, the normalizer could clear the flag or modify the packet in another way before forwarding it to B.

Since active wardens can contain direction dependent rules (for instance, they could block a specific cover protocol if it reaches an enterprise network but not if it leaves the network), A and B have to test whether a cover protocol reaches the other peer in both directions, i.e., if A can successfully send a cover protocol X to B, it does not imply that B can successfully send the cover protocol X to A.

Active wardens can be configured to affect packets only if selected situations occur. For instance, one rule could be "only drop a packet, if the DF flag is set and the MF flag is set". Therefore, *all* possible bit combinations for all selected areas have to be tested by A and B. If a packet with the DF flag set can be successfully transferred and another packet with the MF flag set can be successfully transferred as well, it does not ensure that a packet containing both bits set can be transferred as well between both peers. Also, each received packet must be acknowledged by the peer, what can result in many packets, if many bit combinations for the cover protocol are possible.

3.3.2 The Two-Army Problem

Regarding to the problem's naming, the scenario of the two-army problem is a military situation with two opposing armies A and B. We will briefly explain the two-army problem as described by Kleinrock [Kle78]:

Army A is split into two groups A_1 and A_2. Between the army groups A_1 and A_2 is the army B. A can only successfully attack B if both groups (A_1 and A_2) attack army B at the same time. Therefore, A_1 and A_2 have to agree to attack B. The situation is visualized in Figure 3.6.

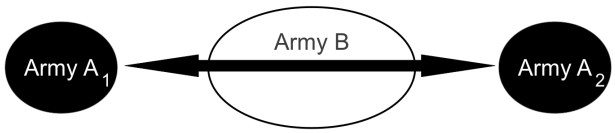

Figure 3.6: The two-army problem

To initiate the attack, A_1 can send an ambassador to A_2. However, the ambassador has to pass army B. If B detects the ambassador on his way to A_2, the message carried to A_2 will be lost. If the ambassador reaches A_2, A_2 can acknowledge to fight together with A_1 against B. However, on his way back, the same situation applies to the ambassador: if the ambassador is detected by B, he will not be able to transfer his secret acknowledgement message to A_1. If the ambassador reaches A_1 with the acknowledgement, A_2 will not attack B since A_2 is not capable of determining whether the acknowledgement reached A_1. Therefore, A_1 can send another ambassador to A_2 and so forth. The problem never ends since it is not possible to ensure that the acknowledgement was received by both A_1 and A_2.

It is not feasible to completely eliminate the two-army problem, however, it is possible to reduce the problem. The TCP protocol therefore uses the 3-way-handshake to provide its full-duplex connection acknowledgements for each direction and re-sends unacknowledged packets [Pos81].

At the beginning of the NEL phase, neither A nor B know how to communicate with each other: Even if a packet was successfully sent to the peer, the peer is not able to determine which cover protocols he can use to acknowledge the received cover protocol to the sender. In other words, the normalized NEL phase results in a two-army problem.

3.3.3 Realizing the NEL Phase in Normalized Environments

This section will present two techniques to realize the NEL phase in normalized environments. At the beginning of the initial NEL phase between two peers, each peer has to know the other peer's address (e.g., an IPv4 address). Every peer contains a set of all its supported cover protocols, e.g., $P = \{x_1, ..., x_n\}$. Each element $x_i \in P$ represents a cover protocol, i.e., each element represents at least one bit of data in a network packet.

x_1 could for instance represent the aforementioned "reserved flag" in the IPv4 header and x_2 could represent the "more fragments" flag. Elements of P can also represent combinations of other elements, e.g., $x_3 = x_1 \cup x_2$="use the 'reserved' flag and the 'more fragments' flag in IPv4". If a combination of elements of P is allowed to be used as a cover protocol, it must be represented by a single element. We enforce this condition to ensure that all possible combinations of cover protocols are part of the NEL phase because active wardens can, as mentioned before, apply rules to such combined conditions as well. For instance, a sample active warden will only drop a packet if x_1 and x_2 are set, i.e., x_3 is present, and the packet will be forwarded if only one element is present.

Our two techniques which help to realize the NEL phase in normalized environments are based on these mentioned conditions (each peer knows the address of the other peer and maintains its own protocol set). For the following descriptions, we say that both peers that want to run the NEL phase are called A and B and their protocol sets are P_A and P_B, respectively. It is not necessary that $P_A = P_B$.

We will first introduce the techniques and will afterwards discuss their optimization.

Simple Solution

Yarochkin et al. as well as Li and He introduced a simple technique that we explained in Section 2.6.3. Their solution is based on the idea to send a packet sequence of the same underlying protocol (without specifying the sequence's details) or to send a test packet to inform the passively listening covert channel peer about the protocols it can use. Since Li and He left details for their approach for future work (cf. Section 2.6.3), it cannot be discussed in this section.

However, in the work of Yarochkin et al. it is assumed that no direction dependent active warden is located on the path between sender and receiver since the receiver must acknowledge received packets through the active warden and the active warden could drop or modify acknowledgement packets. If no response is received, the protocol is considered of little use (it is afterwards only used for testings in the future) for the covert channel, even if it could be used for a one-way communication.
Additionally, Yarochkin et al. focus on whole protocols only. Thus, bit-specific filtering is not taken into account.

Our approach is bit-specific and direction-dependent (i.e., contains no passively listening and acknowledging receiver but two active peers) and thus provides better results than the approach presented by Yarochkin et al. However, this first solution does also not completely overcome the two army problem.

Our solution is based on the idea to send the whole cover protocol set P_A to B and the whole set P_B to A. If the other peer detects the received cover protocol sequence, it knows the unblocked cover protocols. We must assume that only a part of the sequence will be received by the other peer, e.g., x_1, x_2, x_5, x_6 (containing a cleared bit), x_7, x_9. Thus, this solution is error-prone since the receiving peer can never be sure that it currently detects the expected sequence.

Since normalization rules can depend on bit-specific values, all possible combinations of all bit values must be sent to the peer to inform the peer about each possible cover protocol value (means to reduce the number of therefore required packets will be discussed in Section 3.3.3).

Although this approach comprises better (bit-specific instead of protocol-specific) results and although it comprises the advantage of providing both peers awareness about the other peer's transmission options, the two army problem is not solved in any case: If the normalization rules differ in their direction, A and B cannot conclude which cover protocols they can use to communicate with each other since a cover protocol transferred from A to B can probably not be transferred in the opposite direction. For the same reason, passive traffic monitoring as done in [AM11] and [SSP+12] provide no satisfying results.

Therefore, another solution is required to overcome this problem that is discussed in the following section.

Solution with a Third Participant

The second solution is based on the availability of a third participant C that is temporarily available for both A and B for the initial NEL phase. C must therefore be known by A and B. However, it is not required that both peers trust C, as we will show later by introducing a passive C. Both peers must additionally be capable of processing the same micro protocol and the micro protocol must be used for the communication between A, B, and C within the NEL phase.

One might ask, why A and B do not always use C for their communication. C can be a temporary resource and A and B try to establish a direct connection between them to build a new path in the overlay network. In that case, the communication can probably be realized with fewer hops between A and B and raise less attention. Also, the robustness of the link can be increased if fewer other overlay hops are required between A and B.

The scenario of this second solution is shown in Figure 3.7. Both A and B use C to transfer meta information to each other. The meta information contains announcements for cover protocol tests as well as acknowledgements for received cover protocols and will be forwarded by C to the other peer.

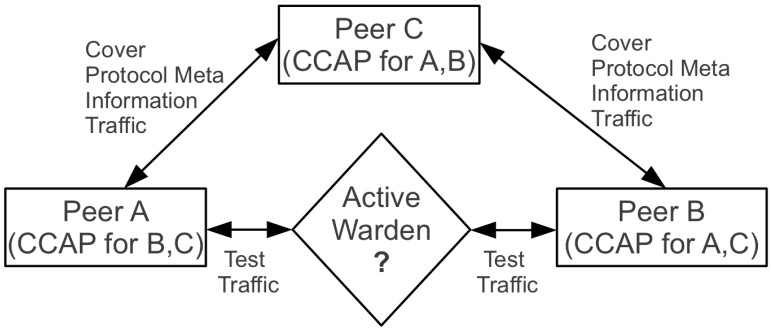

Figure 3.7: Overcoming the NEL phase problem using a third participant C.

For instance, A asks C to forward the information that A will soon send a cover protocol x_1 to B. As mentioned earlier, $P_A = P_B$ need not be true, however, the representations of cover protocols must be unique. Thus, all peers must associate the same element of P with the same bit value. If a bit value (an element of P) is not known to a receiver, he can respond with a micro protocol message indicating that the protocol is not supported.

After C received the packet announcing a cover protocol test from A, it forwards the information to B. After a short waiting time, A will send the test packet directly to B. If B understands the bit value in the packet received from A (via C), it can wait for the expected packet to arrive. If B does not understand the bit value because it does not implement the cover protocol, it responds with a micro protocol error message to indicate that the cover protocol is not supported. If B receives and understands the cover protocol, it sends an acknowledgement information back to C with the command to forward the acknowledgement to A. C receives the packet and forwards it to A.

Afterwards, A is informed that the cover protocol can be used to transfer information to B. If A receives no acknowledgement after a waiting time t, it can mark the cover protocol as blocked. To increase the quality of the resulting information (e.g., in the context of packet loss in networks with high load), A can send meta information packets to C and cover protocol packets to B multiple times. Also B and C can send their packets multiple times. However, the more packets A, B and C transfer, the more attention they attract.

The whole process of evaluating the transfer from A to B has to be done vice versa to ensure A and B both know which cover protocols they can use to communicate with the other peer. After all protocols are tested, the peers can disconnect from C.

Let us assume, C must be trusted, i.e., C can manipulate or drop the packets received from A or B and can also spoof packets. In such a case, A and B could cryptographically sign micro protocol data to prevent modifications (which would not prevent C from dropping packets). To sign messages, key information has to be exchanged in advance. We do not focus on the cryptographic security because the covert channel overlay communication's goal is to prevent that a third party can obtain information about the fact that a covert communication is taking place. If C would be corrupted, C would know about the communication between A and B and the goal would have not been reached. Therefore, A and B *must* trust C.

Using a passive C: There is also a solution available to overcome the problem of having to trust C. If C is not aware of being part of a covert channel overlay network, i.e., it cannot process micro protocol information and is a passive service, A and B can use C nevertheless. As shown by [BK07], steganographic communication can take place in such cases anyhow. The authors developed a wiki for collaborative work based on hiding information in *Flickr* images. The same technique can be applied if A and B would instead place hidden micro protocol information in Flickr images and would announce and acknowledge their cover protocol information in this way. Additionally, a passive C can – although no covert channel software is running on C – cause a protocol switch, since the protocol used between A and C need not be the same protocol as used between C and B.

Improved Protocol Determination Strategies

The number of possible bit combinations to test can become high (2^i if i bits form the cover protocol and if it is possible to use all i bits at the same time). As a result, the NEL phase could require to transfer many packets and thus, could raise much attention. To keep a low profile, it is necessary to reduce the amount of required packets per NEL phase.

Reducing the number of required packets can be reached in three ways:

1. **Independent scanning of different layers:** The scan can be done by only combining cover protocols of a single layer, i.e., each layer is scanned separately from all other layers. For instance, if $P = \{x_1, x_2, x_3\}$, where x_1 belongs to IPv4,

and x_2 and x_3 belong to the TCP protocol, only $2 + 4$ bit combinations have to be taken into account for a scan. However, as active wardens can contain rules that take more than one layer into account (e.g., "do only drop a packet, if the destination address is 10.2.3.4 and the destination port is 6667"), this strategy can result in errors.

2. **Limit tunnel scanning:** If a protocol X is contained in a packet multiple times due to the use of tunneling techniques (e.g., IP is encapsulated in IP, as done by the IPIP and IP-in-IP protocols), the scan could focus on the first occurrence of the protocol. This is error-prone because of the same reason as the first reduction approach (filter routes for multiple levels). However, as mentioned in [Eck12], firewall systems need to be configured to handle multiple encapsulations and this is not automatically the case (many combinations of encapsulated protocols, such as OSPF in IPv4 in IPv6 in IPv4 are thinkable).

3. **Scanning protocols of the same layer independently:** It is thinkable that a protocol X and a protocol Y of the same layer have side effects on each other. For instance, while being unlikely, an active warden could theoretically drop DNS packets with a query for the IP address of a high-secure system from a computer accessing a selected HTTP server's hostname (because the resource could be linked to a different security layer and a write-down must be prevented). However, the probability for such configurations can be considered low.[4] Therefore, side effects for protocols of the same layer should not be taken into account, e.g., a cover protocol state within TCP does not have to be evaluated for a preceding cover protocol state in UDP.

3.3.4 A Remaining Problem: Dynamic Routing Environments

A drawback of the presented approaches is the problem of dynamic routing within the underlay network. The overlay network cannot control routing switches in the underlay network. Overlay routing could possibly reduce the problem if covert channel routing software will be placed on the routers but a full control over the underlay network routing is only thinkable if all underlay network routers are modified with covert channel software. Packets containing covert channel information can take different routes in the underlay network and thus, different normalization rules can be applied between two

[4] Scanning multiple protocols of the same layer in the context of their possible side effects is not part of the previously discussed approaches and is mentioned here only due to the theoretical possibility.

peers. The solution introduced by Yarochkin et al. was to implement the NEL phase as a continuing process. Even after the initial NEL phase ends, it will still run as a background service to handle new circumstances (e.g., blocked protocols or routing configuration changes) in the network. The idea can be applied to the presented two approaches as well to solve this problem.

While routes do usually not change for each new packet of the covert channel, *load balancing* was not taken into account by any author. If load balancing is used, route switches in the underlay network can take place more regularly and affect the NEL. Since the NEL process takes some time, i.e., it requires multiple packets to update information about usable cover protocols, load balancing can (in the worst case) handle each following packet in a different way (e.g., round robin redirection to two different DNS servers if the covert channel uses only DNS-based cover protocols and if the receiver is a hop on the path to both servers in any case[5]).

3.3.5 Effects of Traffic Normalizers

When a covert channel has to be designed, its designer must select a set of possible underlying protocols and associated cover protocols. Based on the existing covert channel literature, many network protocols can be selected as underlying protocols, such as TCP, UDP, IPv4, ICMPv4, IPv6, ICMPv6, HTTP, and DNS.

If a NEL phase has to be realized using one of the underlying protocols, the covert channel designer needs to take the features of active wardens into account. Therefore, it is required to analyze existing active wardens. We have chosen to evaluate four well-known traffic normalizers for the NEL phase in this thesis: The OpenBSD *pf scrubbing* feature, the research normalizer *norm*, the Linux netfilter/iptables extension *ipt_scrub*, and the *Snort normalizer*.

These four traffic normalizers were selected due to their different features and their freely available code. Not only are the four systems implemented using different interfaces, they also differ regarding their runtime environment. Snort and norm are userspace tools, pf scrubbing and ipt_scrub are kernelspace components integrated in the operating system's firewalls (OpenBSD and Linux, respectively). For the analysis of the NEL phase, the important difference lies in the fact that the set of integrated normalization rules is highly different:

[5]This idea is similar to the passive ISN-based covert channel presented by Rutkowska in which the receiver is located on a hop between sender and receiver [Rut04].

- **Number of supported protocols:** The different normalizers support different network protocols. From an administrator's point of view, it is recommendable to chain up multiple different normalizers to gain a maximum of supported network protocols. While UDP normalization is only supported by norm, the important HTTP protocol is only supported by Snort. The kernel-space normalization systems do not include support for any application layer protocol and do also only support TCP as a transport layer protocol.

- **Number of supported features per protocol:** The support for a single network protocol (e.g., IPv4) does not imply that all possible normalization rules are integrated in a traffic normalizer. For instance, the IPv4 reserved flag is only cleared by Snort and norm, but not by pf and ipt_scrub. Other normalization rules are configuration-dependent (e.g., pf only clears the DF flag in IPv4 if the "no-df" keyword is used in the configuration). The scale of supported normalization rules reaches from only 8 (ipt_scrub) to more than 70 (norm).

Based on our analysis of the existing documentation and the available source code, we could evaluate that only the following feature rules are supported by at least three of the normalizers:

- **IPv4**: TTL modification (setting the TTL to a default value), clearing the reserved bit and the DF flag, removing IPv4 optional header components (includes the modification of the IHL value), and dropping of packets with a non-standard conform IHL (i.e., IHL < 5). In case of OpenBSD, this is not specified in the manual, but was found in a source code analysis for this thesis.

- **IPv6**: The hop limit value is modified to a default value and extension headers are either modified or removed.

- **ICMP**: The ICMP echo requests and responses (type 8 and 0) are dropped or modified (the Snort normalizer clears the ICMP code values for both ICMP types). Other ICMP types are not normalized.

- **TCP**: The reserved bits are cleared, packets with unusual flag combinations or flag/data combinations are dropped (e.g., SYN and RST set, SYN and FIN set, or SYN set and payload attached), headers with a too small header length are dropped.

By analyzing traffic normalizers, the covert channel can be designed to utilize underlying protocols not supported by the normalizers or by only few normalizers. Thus, it would be a bad idea to utilize the DF flag or the reserved flag of IPv4, or to use ICMP echo request or echo response packets to carry hidden data. It is, for instance, much more unlikely that an ICMP destination unreachable packet containing hidden information will be normalized if used by a covert channel software.

Also UDP and HTTP, as only supported by some of the normalizers are thinkable to be used as underlying protocols for the covert channel. However, HTTP is usually monitored as requests are logged by the webserver software and application layer intrusion detection systems can detect unusual HTTP content.

Network protocols not supported by any of the analyzed normalizers (e.g., some streaming or routing protocols) or protocols not used within the selected target network environment (i.e., the network the covert channel uses to operate) can also be used as underlying protocols but their presence can (since probably not common in the target network) raise attention (e.g., BGP in an RIP routing environment).[6] Thus, a low chance for a traffic normalization does not automatically imply a good quality for the covertness of an underlying protocol. However, streams (e.g., VoIP) were also considered as good underlying protocols by Acosta and Medrano due to the update rate and complexity of such streams, while sequence numbers are considered as unsuitable placement areas due to the risk of generating duplicate messages (with equal sequence numbers) that could raise attention [AM12].

As previously explained in Section 2.6.3, Yarochkin et al. use passive monitoring to detect utilizable network protocols. Another application of passive monitoring can be to detect the occurrence rates of the received network protocols P_{recv} which results in knowledge about rarely used protocols P_{rare}. If the covert channel designer can obtain information about the used normalizer software, he can determine a set P_{norm} of protocols likely to get normalized. Thus, the set

$$P_{useful} \quad = \quad P_{recv} \setminus (P_{norm} \cup P_{rare}) \tag{3.2}$$

will contain the useful underlying protocols since these protocols are likely to occur in the network and have a low chance to get normalized or to raise attention due to their

[6] *norm* verifies the value in the IPv4 *protocol* field and drops packets which are not supported on the encapsulated layer. Application layer protocols based on TCP or UDP are not affected by this policy.

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 3.1: Occurrence rates of different transport layer protocols and ARP from the first three traffic dumps of simpleweb. For each traffic dump, the first 750.000 packets were evaluated.

Source	Type 0	Type 3	Type 8	Type 11
Simpleweb-Loc1	6.60%	85.10%	7.80%	0.50%
Simpleweb-Loc2	2.96%	88.93%	7.97%	0.13%
Simpleweb-Loc3	2.24%	0.08%	97.66%	0.02%

Table 3.2: Different occurrence rates for types of the ICMP protocol in the three selected network environments. Again, the first 750.000 packets were evaluated.

abnormally high occurrence rates (the idea not to use rarely used protocols was already mentioned by Yarochkin et al. in [YDL+09]). However, if instead of the underlying protocol sets, the cover protocol sets would be taken into account, the elements of P_{useful} would be more accurate.

We evaluated different *tcpdump* recordings from the website *simpleweb.org/wiki/Traces* to obtain information about the differences between the occurrence rates of fundamental network protocols. The recordings were all based on the Ethernet protocol and came from bigger organizations (especially universities) and thus, are not representative for other networking environments, such as small businesses. Table 3.1 lists the occurrence rates of the selected transport layer protocols and shows significant differences in their occurrence rates.

As shown in Table 3.2 by using the ICMP protocol, differences do not only appear for the occurrence rates of network protocols but also for the occurrence rates of different types of a single network protocol (ICMP is used as an underlying protocol for different covert channel tools, such as *Ping Tunnel* [Stø09]).

These analyses reveal the network dependence of P_{rare} and thus, P_{useful} depends on each network as well.

3.3.6 Proof of Concept Implementation

We developed a proof of concept implementation to simulate an unidirectional NEL phase (from one peer to the other) with a temporary participant C as explained before.

The simulation contained a receiver (B) that accepted micro protocol information from the temporary participant (C) and made use of the same micro protocol to transfer the result (i.e., whether an announced packet was sucessfully received) information back to C. We implemented B using *libpcap* (*tcpdump.org*) to listen for both micro protocol packets from C as well as for cover protocol testing packets from A.

By using the packet generator *scapy* (*www.secdev.org/projects/scapy/*) we were able to generate all required packets from A to B. The output of C was scripted by hand and the acknowledgement information from B to C was monitored using the packet sniffer *Wireshark*.

The tool *scapy* provided the flexibility to simulate different situations by altering selected attributes of packets: By simply not sending an announced packet, we could simulate packet loss. By sending similar packets as announced (but with slightly different bit combinations), we simulated traffic modifications by active wardens. Non-normalized packets resulted in no problems (they were simply received and acknowledged). If an announced packet from A to B got dropped or lost, it was considered to be blocked (since B received no dropped/lost packets, A will also not receive any acknowledgements). Both peers cannot differentiate between lost and dropped packets.

Traffic modifications resulted in the situation that the same packet (containing the same bit combinations) was received twice. The reason for this observation can be explained using a simple example: If C informs B that A will soon send two packets to test the DF flag of the IPv4 header (one with the DF flag set and one with a cleared DF flag), B will wait for both packets. A will afterwards send both packets. If an active warden modifies the packets (it either always sets or always clears the flag to unify the traffic), B receives the same bit value twice (either two packets with a cleared DF flag or two packets with a set DF flag – in each case, only one packet was actually modified). B will acknowledge the receipt of one of the packets and will not acknowledge the second one. The second occurrence of the same packet will be ignored.

In any case, B must ensure that no noise affects the NEL phase, i.e., a received packet not belonging to the NEL phase but containing the correct bit values/combinations should not be handled as a covert channel packet. Therefore, B can simply configure filter settings as provided by *libpcap*, i.e., to only accept packets with the source address of A (respectively C).

The identification information for announced and acknowledged cover protocol test traffic must be unique for all possible cover protocols (unique IDs for all cover protocols and all bit values must be configured, cf. Section 3.3.7).

Therefore, m bits are required to identify the cover protocol (where $m = \lceil \log_2(|P|) \rceil$). Another n bits indicate the bit value. The number n depends on the number of possible states to be tested. For instance, if the DF flag is to be tested, 2 bit values are possible (DF=0 and DF=1) and therefore, $n = \log_2(2) = 1$ bits are required for the announced/acknowledged value. In other words, the largest combination of m and n bits must be available in the cover protocol space to specify announced tests using a third participant C. Additionally, a micro protocol bit must indicate that the following information is an announcement.[7]

3.3.7 Version-dependent Cover Protocols

As introduced by Yarochkin et al., the NEL phase is a permanent process split into an initial phase that is finished after a subset of usable protocols was found. However, in our model, the *initial* NEL phase is finished when at least one cover protocol supported by *all* covert channel software versions (referred to as a *initiator protocol*) was determined. Our initial NEL phase does only scan for such initiator protocols to ease and speed-up the process.

All peers must be aware of such initiator protocols by marking these protocols in their database. At least one initiator protocol must be shared by *all* covert channel software versions that shall be able to communicate with each other.

After a non-blocked/modified initiator protocol was determined, the initiator protocol is used to transfer a micro protocol packet from host A to B (or from B to A). The packet contains the version number of the covert channel peer. Each peer maintains a list of cover protocols supported by different versions. Tables 3.3 and 3.4 show a sample situation, where A runs software version 1.0 and B runs software version 1.1. As shown, newer versions must be aware of the supported cover protocols of older software versions.

Version	**Protocol**	**Initiator Flag**
1.0	IPv4:TTL	X
1.0	IPv4:Reserved Flag	X
1.0	IPv4:TTL,Reserved Flag	X

Table 3.3: covert channel peer A's cover protocol table (software version 1.0).

While version 1.0 supports to use the IPv4 reserved flag as a cover protocol, version 1.1 discarded this support (for instance because it was detected by a monitoring soft-

[7] This chapter also introduces *status updates* (cf. Section 3.5) — a technique to realize small micro protocol headers.

Version	Protocol	Initiator Flag
1.0	IPv4:TTL	X
1.0	IPv4:Reserved Flag	
1.0	IPv4:TTL,Reserved Flag	
1.1	IPv4:TTL	X
1.1	TCP:ISN	
1.1	IPv4:TTL,TCP:ISN	

Table 3.4: covert channel peer B's cover protocol table (software version 1.1).

Version	Protocol	Initiator Flag
1.0	IPv4:TTL	X

Table 3.5: Intersection of A's and B's cover protocol table (protocols that can be used by both hosts).

ware) and therefore added another cover protocol (the TCP ISN). Table 3.5 shows the intersection of the cover protocol tables for both peers. The only cover protocol supported by both versions is the IPv4 TTL. Peer B now knows that he does not have to check whether he can successfully use the ISN as a cover protocol to communicate with peer A. Peer A, on the other hand, does not know the supported protocols of version 1.1, and will try to reach B by using the reserved flag of IPv4. B denies the usage of the reserved flag to A by acknowledging A's packet using a micro protocol in the TTL-based cover protocol. The micro protocol therefore must include a command that denies the use of the unsupported cover protocol received from the peer. Therefore, each cover protocol in the cover protocol table must be identified by a unique number to prevent race conditions.[8]

In case a cover protocol (or part of it) previously used by another protocol is used with another coding in future protocol versions (e.g., to raise less attention), a new protocol entry must be included in P and the old element must be removed to prevent the simultaneous use of the same cover protocol with two different codings. If changes in codings are foreseen, elements of P must always be associated with a specific coding.

[8]If no cover protocol identifier (and no micro protocol sequence numbers) would be transferred within the micro protocol and if A would probe n different cover protocols before waiting for B's response, A could think that B denies the usage of the wrong cover protocol after receiving an response message that indicates that a received protocol is not supported.

3.3.8 Optimized Post-NEL Communication

After the initial NEL phase is completed, protocols must be selected for the communication. Since different protocols can raise different attention (such values can only be estimated), their usage rates can be configured by the covert channel sender. To minimize the raised attention, a protocol hopping covert channel can, as proposed by Keller [WK11], also minimize the required number of packets for a transaction or the overhead of a transaction. The optimization depends on the user's demands, especially on the use case. Therefore, Keller introduces the value q_i as the number of bits of the underlying protocol P_i which are required to be transferred for one cover protocol bit:

$$q_i := \frac{sizeof(P_i)}{sizeof(CP_i)} \tag{3.3}$$

If linear optimization is applied, the optimal probabilities for $P_1, \ldots, P_n$ can be calculated [WK11]. Therefore, it is useful to assign each protocol at least a small probability of occurrence to make forensic traffic analyses harder [WK11].

To optimize the throughput of the channel, the function f_1 can be maximized [WK11]:

$$f_1 = \sum_{i=1}^{n} p_i \cdot sizeof(CP_i).$$

On the other hand, the function f_2 can be minimized to minimize the overhead required to transfer a hidden message [WK11]:

$$f_2 = \sum_{i=1}^{n} p_i \cdot q_i.$$

Micro protocols can be used to exchange information about the optimization of covert channel traffic between two peers. Therefore, priority bits for cover protocols could be exchanged between peers in the micro protocol header.

3.3.9 Forwarding in Covert Channel Overlays

Coming back to the introduced scenario of covert channel overlays with a forwarding capability, Keller's ideas can be taken into account to not only optimize the direct communication between two peers but also for optimizing the forwarding of covert data over covert channel proxies/routers (i.e., hops that are aware of the covert channel and that forward the hidden information). Placing multiple hops between a covert channel

sender and receiver can improve the anti-traceability of the systems and, e.g., if the communicators are journalists, improve their safety.

Figure 3.8 visualizes a proxy communication between the sender S and the receiver R using the covert channel proxies $Q_1 \dots Q_n$ in an covert overlay network.

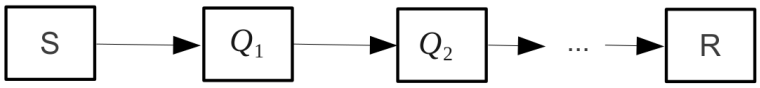

Figure 3.8: The sender S transfers information to the receiver R via the covert channel proxies $Q_1 \dots Q_n$.

We assume that S and each forwarding instance Q_i chooses one protocol to forward data to the next hop of the proxy chain. By applying the NEL phase, each host pair $(S, Q_1), (Q_1, Q_2), \dots,$ and (Q_n, R) can determine the available protocols they can use for a communication in advance.

We assume that SP_i is the set of usable protocols between element i and $i+1$ of the chain and was determined in the NEL phase (S is element 0 and the receiver is element $n+1$ while proxy Q_1 is element 1 and proxy Q_n is element n).

In the simplest case (i.e., no protocols are normalized), SP_i is the intersection of the protocols supported by element i and $i+1$ of the chain, i.e., $SP_i = P(Q_i) \cap P(Q_{i+1})$. Let $s_{max}(i)$ be the maximal available space available per packet of all elements in SP_i and let $q_{min}(i)$ be the minimal overhead of all elements in SP_i. We can now apply Keller's optimization for the forwarding in the proxy chain as follows if a packet is received from element Q_i at element Q_{i+1} and must be forwarded to Q_{i+2} (Q_0 would be the sender and Q_{n+1} the receiver)

Optimize for minimal packet count on hop Q_{i+1} if a packet is received to be forwarded:

1. If $s_{max}(i) = s_{max}(i+1)$, then forward the payload directly to Q_{i+2}.

2. If the received packet is the last packet of the transaction (e.g., indicated by a flag as foreseen by the micro protocol by Ray and Mishra [RM08a, RM08b]), forward the data as well to Q_{i+2} using the protocol with $s_{max}(i+1)$.

3. Otherwise: Forward as many complete covert data packets using the protocol that provides $s_{max}(i+1)$ as feasible. As pointed out by Keller, bursts in case of $s_{max}(i+1) \ll s_{max}(i)$ can be provided by forcing a maximum packet frequency via the leaky bucket method. If there is data left to be forwarded, wait a time t

for new data to arrive and afterwards forward it using the protocol that provides $s_{max}(i+1)$. This step must be repeated until no new data arrives for the time t or the last packet of the transaction was received.

To prevent that only the optimal protocol is used, i.e., to also enable other underlying protocols, each protocol P_i can be used with at least a small occurrence rate, as proposed in the previous section.

Optimize for minimal overhead on hop Q_{i+1} if a packet is received to be forwarded:

To minimize the overhead of the covert channel transaction, the previous algorithm has to use $q_{min}(i)$ instead of $s_{max}(i)$ and $q_{min}(i+1)$ instead of $s_{max}(i+1)$:

1. If the received message size $sizeof(m)$ is of the same size as space is provided by $q_{min}(i+1)$, then directly forward the data using the protocol that provides $q_{min}(i+1)$.

2. If the transaction ends, send all remaining information using the protocol that provides $q_{min}(i+1)$ as long as enough data remains that has the size of the space of the protocol that provides $q_{min}(i+1)$. Afterwards use the protocol $p_j \in SP_{i+1}$ for which the overhead of the remaining k bits is minimal.[9]

3. Otherwise: Forward as many full covert data packets using the protocol that provides $q_{min}(i+1)$ as possible. Wait time t for new traffic to arrive. Afterwards, forward as many full covert data packets using the protocol that provides $q_{min}(i+1)$ as possible. If no new data arrives, use the protocol that provides the smallest overhead for the remaining k bits to forward the data. This step must be repeated until no new data arrives for the time t or if the last packet of the transaction was received.

As will be explained later, additional optimizations for the micro protocol itself are feasible which can be used for more complex scenarios but proxy chaining: Backs realized a micro protocol-based dynamic routing [Bac12] which is based on the concept of status updates which will be introduced in Section 3.5.

[9] It is thinkable that $q_{min}(i+1)$ is only optimal if all provided space of the cover protocol is used but a protocol providing fewer bits could be more suitable for the transfer of the remaining k bits.

3.3.10 Results

Within the NEL phase, a two-army problem exists since a possible traffic normalization between two covert channel systems can take place. The passive monitoring of traffic as done by Swinnen et al. in [SSP+12] and Acosta and Medrano in [AM11] cannot solve the two-army problem, but active traffic tests as done in [LH11] (calculation of survival values with an acknowledgement channel, cf. Chapter 2.6.3) can at least reduce the problem. However, Li and He did left details for their theoretical solution for future work.

We presented two solutions to overcome the two army problem: First, by sending a packet sequence that must be detected by the covert receiver (this is similar to the discussed approach of Yarochkin et al. but is direction-dependent and provides finer grained results) and second, by using a third participant C to announce covert channel test traffic.

We discussed the advantages and disadvantages of both approaches and have shown that only the solution with a third participant is satisfying. We have also shown that the workload for realizing the NEL phase can be reduced and that dynamic underlay routing must be considered problematic.

Additionally, effects of selected existing traffic normalizers were summarized to gain information about potentially unfiltered and low attention raising cover protocols. We conclude that the optimal set of cover protocols depends on the network and normalizer setup. However, a covert channel should utilize regularly occurring underlying protocols and should prevent the use of cover protocols that can be normalized by most of the normalizers.

To support the evolution of a covert channel overlay, i.e., to make it possible to utilize new cover protocols in future software versions, we propose the idea of version numbering.

Both the version numbering as well as the announcement of covert channel test traffic, and the coordination of the communication with other participants can be achieved using micro protocols. The next sections of this chapter deal with the problem of optimizing micro protocols.

After the initial NEL phase is completed, the communication between two peers as well as the forwarding of covert channel data in proxy chains or via overlay routers can be optimized for a minimized packet count or for a minimized overhead.

3.4 Using Formal Grammar to Design Micro Protocols

While few approaches (Ray and Mishra in [RM08a] and Ping Tunnel [Stø09]) already presented concrete micro protocols, no previous work is known regarding the design of micro protocols or regarding their optimization. While the space-efficient design of micro protocols will be discussed in Section 3.5, this section covers the optimization processes containing a protocol engineering approach for micro protocols.

The presented systematic approach comprises the goal to be usable for all network protocols with binary headers (including those which were already evaluated for their covert channel use, like the most well-known TCP/IP protocols, as well as protocols which have not been evaluated for covert channel use). The presented approach results in an implementation-ready micro protocol specified in a formal grammar linked to a mapping of micro protocol bits to bits of the cover protocol. The micro protocol will be (standard-)conform to the cover protocol bit patterns and the mapping of bits between micro and cover protocol is optimized in a way that occurrence rates of bit values are (if possible) similar between both protocols. The approach can (if additional terminal symbols are introduced, as will be shown in Section 3.4.4) also deal with protocols that comprise state-dependent bit patterns (e.g., a header flag x can only be set if the previously received packet contained the flag y set).

Formal grammars have been applied in other areas of information security (such as attack modeling [GK02a] or Spam detection [TF12]), but were also used in the protocol engineering community (e.g., to model events and states for protocol implementations [LM83] or to model the procedure and message-format of network protocols [Har77, Har78]). Harangozó motivates the use of a formal protocol description from two viewpoints [Har77]: i) Misunderstandings of protocol descriptions given in natural languages can lead to wrong implementations while a formal description leads to a better implementation. ii) Harangozó also motivates the theoretical aspect in which deadlocks can be easier detected when a formal protocol description is given.

Besides formal grammars, other techniques such as Petri nets, finite state automata, or SDL are used in the context of protocol engineering [Sun78, BS80, NY85, Kön03]. Since many techniques are already available, there is no need to re-invent existing means or to address already solved problems in the development of communication protocols. In general, formal methods for protocol engineering are less error-prone than informal methods since informal methods are often ambiguous, unclear or incomplete [NY85].

To the author's best knowledge, no protocol engineering approach is available to minimize the attention raised by a protocol. The application of formal grammar was chosen since language-subset tests allow to verify whether a protocol conformity is given. The presented approach does not cover the actual operation rules of the micro protocol nor does it prove that the designed micro protocol is free of deadlocks, that the protocol is complete or free of errors since such protocol engineering means are already available. The goals of our approach will be discussed in detail after an overview of the approach will be given.

3.4.1 Micro Protocol Engineering

In the following, the protocol engineering approach for micro protocols including its goals is explained. The approach comprises six steps and is split in two layers. One layer focuses on the cover protocol and the other layer on the micro protocol.

The approach requires that an underlying protocol was already selected to carry hidden data. Determining useful underlying protocols as well as cover protocols can be achieved by applying the network environment learning phase in a network (previously explained in Section 3.3).

Overview

As the understanding of the whole process will be easier if an overview is given before the steps of the process are described in detail, the most important steps of the approach will be explained in the following.

First, a cover protocol has to be selected based on an underlying protocol and the occurrence rates of the cover protocol's bits must be evaluated. Additionally, a grammar has to be created to represent the rules of the cover protocol. A similar procedure has to be done for the micro protocol: A grammar has to be created and the occurrence rates of the micro protocol's bits must be evaluated and sorted. Afterwards both occurrence rates are mapped and it must be verified, whether the micro protocol's language is a language subset of the cover protocol's language. If the micro protocol's language is a language subset, the micro protocol will only produce (standard-conform) bit patterns of the cover protocol.

Goals

Our approach was designed to provide a means for developing micro protocols for covert storage channels. Besides this major goal, different sub-goals exist:

- The approach should be **easy to understand and easy to apply**. Therefore, simple grammar (regular and context-free grammar) forms the base of the approach.
- A **clear distinction between the different working areas** of the approach should be provided. Therefore, we introduced two layers – one layer represents the cover protocol, the other layer represents the micro protocol.
- After the approach is done, the **resulting protocol should coincide to the bit pattern rules of the cover protocol**. If the cover protocol rules are based on a standard, it should be ensured that the standard-conformity of the cover protocol is still given if the micro protocol operates.

  If, for instance, multiple TCP flags belong to the cover protocol, it must be ensured that no flag combination will be set by the micro protocol that does not conform to the usual protocol behavior (i.e., combining the SYN and the RST flag).
- The **attention raised by the micro protocol should be small**. Therefore, bits of the micro protocol are mapped to the cover protocol in a way that occurrence frequencies for bit values are similar to normal traffic.

  The previously mentioned goal to ensure conformity of the micro protocol to the cover protocol also contributes to a minimized attention since uncommon behavior of the underlying protocol can result in detections by intrusion detection systems.
- The approach must **allow a dynamic re-designing and re-optimization** of selected steps. Thus, it must be able to allow the cover protocol to add/remove selected bits, which also applies for the micro protocol. Also, the mapping between both protocols must be changeable in case the language subset verification will not work with a given mapping or if an a posteriori evaluation results in wrongly evaluated occurrence rates of the cover or micro protocol bits.
- The approach shall be **applicable to all network protocols with binary headers**, i.e., if the underlying protocol is changed to another protocol with a binary header, the approach itself does not have to change. It should only be necessary

to repeat the approach or even only a subset of the six steps. For instance, if the micro protocol stays the same (and a suitable cover protocol can be built), the micro protocol can be kept and does not have to be re-engineered from scratch.

3.4.2 Six-Step Approach

As mentioned previously, our approach comprises six steps. The whole approach is visualized in Figure 3.9. The six steps form an incremental process and the process is finished after the sixth step is finished with a successful result. If problems occur, e.g., if the verification in the sixth step was unsuccessful, previous steps can be repeated. We discuss re-designing paths in Section 3.4.5.

In protocol engineering, the *testing* phase verifies the correct processing of a software and thus depends on the *implementation* phase [Kön03]. The implementation phase is out of the scope of our approach since we focus on the *validation* phase that aims on verifying the logical consistency of a protocol. As mentioned earlier, means to verify aspects such as that a protocol is free of deadlocks and that all of the protocol's states are reachable, are already available since decades. This work does therefore only aim on contributing a means that ensures that a micro protocol message defined by a formal grammar does only generate bit patterns in its message that do not violate the rules for bit patterns of the cover protocol (and would thus raise attention). This aspect is considered as part of the *protocol validation* phase since a formal grammar creates the protocol messages based on a logic of which we want to ensure its consistency.

In the following, we describe each of the steps in the order of their appearance in our protocol engineering process.

Step 1: Cover Protocol Header Design

This step requires the previous selection of an underlying protocol. The utilizable areas within the underlying protocol about to be used for the cover protocol can be determined via two ways:

1. As discussed in Chapter 2.4, a number of underlying protocols were already subject to covert channel investigation. If an underlying protocol is to be used for which knowledge regarding the embedding of covert channels is already known, the existing literature can be taken into account. The available research results do usually comprise discussions on the detectability of the different utilizable areas

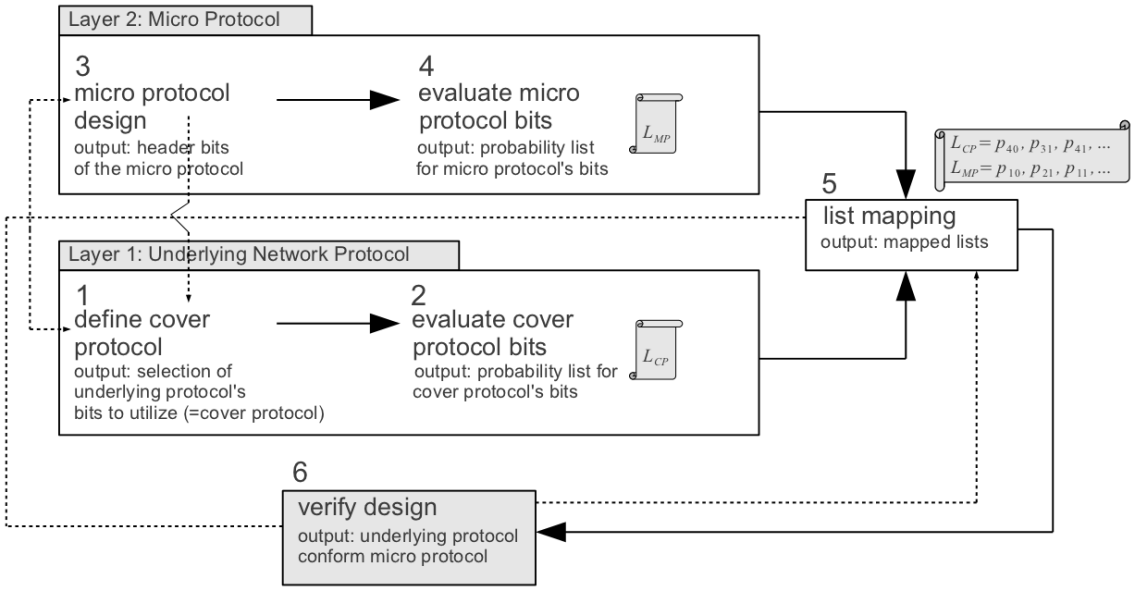

Figure 3.9: The two-layer micro protocol engineering approach from [WK12b]. Dashed arrows represent possible re-engineering paths.

of the underlying protocol and can thus be considered useful for the selection of a cover protocol area.

2. If no previous covert channel research work is available for the favored underlying protocol, the protocol designer has to perform an analysis on his own. Therefore, unused as well as currently unrequired header areas must be evaluated for the placement of covert channels. If the favored underlying protocol is not a self-developed protocol, an official documentation (e.g., a standard specifying document) will be helpful to gain knowledge about the protocol's rules.

Step 2: Probability List Generation

The probability of each bit i in the cover protocol having value j, i.e. $p_{ij} = prob(b_i = j)$ is evaluated (e.g., by estimation or by evaluation of traffic recordings of the given protocol).

Note: If a bit i with $p_{i0} = 1$ or $p_{i1} = 1$ is part of the cover protocol, its use for the micro protocol is limited since it must later be associated with a micro protocol bit value (step 5). An always (un)set bit in the cover protocol can only be used for always (un)set values in the micro protocol since the value does not change. The self-information $I(p_{ix} = 1) = -\log_2(prob(b_i = x) = 1)$ of values which are always x is 0. Thus, such bits are not recommended to become part of the cover protocol and can be removed by switching back to step 1 of the approach. Alternatively, e.g., if used for

changing micro protocol values, these bits can raise high attention. For instance, a very rarely set bit of the micro protocol can later be mapped to the reserved flag of the IPv4 header – it would raise high attention but only in very few packets that will set the cover protocol bit due to a rarely used bit in the micro protocol.

If possible, it should be taken into account that the probabilities for different bit values can depend on the values of other bits within the cover protocol (e.g., the ICMP code bits depend on the ICMP type value). To achieve an easy protocol design, we recommend to only utilize bits which do not depend on other bits (e.g., some flags like the DF flag in IPv4). As pointed out by Keller [WK12b], the number of probability values increases from $2n$ to 2^n if the probabilities of all possible bit patterns of the cover protocol will be taken into account. As we proposed in [WK11], a classification set for probability values can be applied (e.g., $\{low, medium, high\}$) if no exact numeric values can be obtained.

After the probability values for all bits are evaluated, the bits are sorted in a list L_{CP} in increasing order of probability [WK12b]. Each unset and set value of a bit is handled separately, e.g., say a cover protocol comprises two bits b_0 and b_1 with the values $p_{00} = 0.01, p_{01} = 0.99, p_{10} = 0.5, p_{11} = 0.5$, then L_{CP} will be $(p_{00} \rightarrow p_{10} \rightarrow p_{11} \rightarrow p_{01})$ or $(p_{00} \rightarrow p_{11} \rightarrow p_{10} \rightarrow p_{01})$.

Step 3: Micro Protocol Header Design

In the third step, the actual micro protocol M that will be placed in the cover protocol C has to be designed. C must provide enough space for the placement of M, i.e., $sizeof(M) \leq sizeof(C)$, which is similar to previous conditions we already discussed in Section 2.6.2 without distinguishing between cover, underlying, and micro protocol.

If it is not feasible to decrease the size of M to the size of C, the designer needs to increase $sizeof(C)$ by adding additional bits of the underlying protocol (i.e., steps 1 and 2 must be partly repeated).

The micro protocol is designed in the same way as other protocols (e.g., by using UML, Petri nets, SDL, or LOTOS [Kön03]). However, since micro protocols are usually small ($sizeof(M)$ is only a few bits), their design is often straight forward and the use of Petri nets and other tools will not be a necessity in most cases.

By designing a micro protocol, we mean to specify the micro protocol's header and functionality (e.g., a sequence number or ACK flag can be part of the header and is linked to the functionality of reliability). Features from existing protocols can be adopted at this point. Introducing the required functionality is up to the protocol engineer, i.e.,

he must decide whether functionality x and header bit y are necessary for the micro protocol's operation.

Alternatively, an existing micro protocol (e.g., the micro protocol by Ray and Mishra [RM08a]) can be used. Using the micro protocol of *Ping Tunnel* [Stø09] cannot be recommended due to the amount of required space of the protocol.

Step 4: Micro Protocol Evaluation

As previously done in step 2 for the cover protocol, the probability values of the micro protocol's bits must be evaluated. If an existing micro protocol was used in step 3, probability values can be obtained from traffic recordings of micro protocol packets. If no traffic recordings are available, they can be generated by implementing or simulating the micro protocol.

However, if no implementation or traffic recording is available as well as if no previously existing protocol is used, the task can be achieved by estimating the expected bit probability values.

This step finishes with the generation of the bit probability list L_{MP} that is generated in the same way as L_{CP}.

Step 5: List Mapping

Finally, both lists are mapped. Figure 3.10 visualizes a sample situation in which an underlying protocol's bits form the cover protocol. The bits of the cover protocol are mapped to the micro protocol.

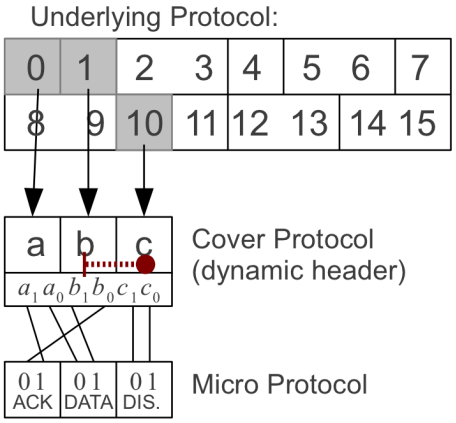

Figure 3.10: A sample underlying protocol with three utilized bits building the cover protocol. A sample mapping of the micro protocol bits to the cover protocol bits is additionally shown. We assume that "c" is only valid if "b" is set.

Keller pointed out that, in case bit patterns are used, *the mapping of lists sorted according to probabilities is rank-preserving and thus optimal in the sense that it minimizes the sum of the squares of probability differences before and after the mapping* [WK12b].

As mentioned earlier, it is necessary that $sizeof(M) \leq sizeof(C)$. Thus, it is possible that $L_{MP} < L_{CP}$, i.e., no 1:1 mapping of both lists is feasible. In [WK12b], we provide the following example: If a cover protocol comprises two bits and the micro protocol requires only one bit (e.g., to signal the beginning and the end of a transaction), both list ends could be mapped:

$L_{CP} = p_{00} \rightarrow p_{11} \rightarrow p_{10} \rightarrow p_{01}$ and $L_{MP} = \emptyset \rightarrow \emptyset \rightarrow p_{01} \rightarrow p_{00}$

We afterwards map p_{00} of the micro protocol to p_{01} of the cover protocol, as well as p_{01} of the micro protocol to p_{10} of the cover protocol while p_{11} and p_{00} of the cover protocol remain unused.

A better solution is to map micro protocol bit values to cover protocol bit values with similar occurrence rates [WK12b]. If, for instance, p_{11} of the cover protocol is more suitable for p_{01} of the micro protocol, the mapping can be optimized.

Step 6: Micro Protocol Header Conformance Verification

Finally, it must be ensured that the micro protocol, when placed inside the cover protocol, does not violate the rules of the cover protocol. Therefore, we need to verify, whether all possible micro protocol states can be reached without causing a rule violation of the cover protocol, i.e., a cover protocol state unforeseen by its design.

To provide an example for rule violating (e.g., standard conformity breaking) behavior caused by a micro protocol, we can imagine a micro protocol that utilizes different bits of the ICMP type and code values. Due to the ICMP standard, the valid ICMP code values depend on the ICMP type. Similar situations apply for the IPv4 options as well as for the link control protocol (LCP) configuration options within PPP. If the micro protocol causes a bit combination that leads to an invalid ICMP type and ICMP code combination, the micro protocol design and/or the bit mapping would be insufficient.

The final step can be split into two sub-steps: First, we define a formal grammar for both the micro protocol and the cover protocol. Afterwards, we test whether the language produced by the micro protocol's grammar is a subset of the language produced by the cover protocol's grammar.

Step 6-a: Grammar Generation:

In this step, both grammars, one for the micro protocol as well as one for the cover protocol, must be defined. Therefore, a context-free or regular grammar (Chomsky type

2 or 3, respectively) can be used for the cover protocol and a regular grammar can be used for the micro protocol. Otherwise, the algorithms for language subset verification cannot be applied.

The grammar generation process is supported by the fact that information about an underlying protocol can usually be obtained by available standard specifications. The grammar generation process is fast since only a few bits of a protocol header become part of a cover protocol (the more bits of a header are utilized, the more attention is risen by the micro protocol). If the same cover protocol was used before, no new grammar has to be created.

To stay conform with the two layer distinction for the first steps of our approach, we continue the layered model for step 6. Therefore, we first define the rules for the cover protocol in the context of the underlying protocol. The protocol designer must be aware of the rules of the underlying protocol, which can, as mentioned before, be extracted from RFCs or other standard documents. The rules related to the bits of the cover protocol are then built into a grammar (for convenience, we do not use bit numbers but letters ($a \ldots z$) to represent cover protocol bits).

In the following, a sample cover protocol containing three bits (a, b, and c) will be used to explain our concept. We assume that bit c is only valid if bit b is set (cf. Figure 3.10). We build a formal grammar G_{CP} that generates the language of our sample cover protocol.

A formal grammar is a tuple $G = (V, \sum, P, S)$. The set V contains all non-terminal symbols while all terminal symbols are part of the set $\sum$. The set P contains the productions for G and $S \in V$ is the starting symbol [GK02a].

For this work, we focus on *regular* and *context-free* grammars. Context-free grammars are formed by production rules which contain a single non-terminal on the left side of a production rule while the right side of a production rule can contain terminals, non-terminals as well as a combination of both. Regular grammars on the other hand, are a subset of the context-free grammars. Regular grammars contain the additional restriction that the right side of a production rule must be the empty string ϵ, a single terminal, or a single terminal and a single non-terminal. If non-terminals are part of production's right sides, they must either always be located before or always be located after the terminal.

For the generation of both grammars it is necessary that the produced sentences contain their terminal symbols in the order of the underlying protocol header's bits they represent. For instance, if we have two bits a and b (representing the DF and MF flags

in IPv4) with the values a_0, a_1, b_0, b_1, a value of bit b must always appear after a value of bit a.

The context-free grammar of our previously mentioned cover protocol is as follows:

$$
\begin{aligned}
G_{CP} &= (V, \sum, P, S) && (3.4)\\
V &= \{S, A, B, C\} && (3.5)\\
\sum &= \{a_0, a_1, b_0, b_1, c_0, c_1\} && (3.6)\\
P &= \{S \rightarrow AC && (3.7)\\
&\quad A \rightarrow a_1 | a_0 && (3.8)\\
&\quad B \rightarrow b_1 | b_0 && (3.9)\\
&\quad C \rightarrow b_1 c_1 | B c_0\} && (3.10)
\end{aligned}
$$

For the regular grammar representation of the cover protocol, the right sides of the production rules must be eased:

$$
\begin{aligned}
G_{CP} &= (V, \sum, P, S) && (3.11)\\
V &= \{S, B, C_A, C_B\} && (3.12)\\
\sum &= \{a_0, a_1, b_0, b_1, c_0, c_1\} && (3.13)\\
P &= \{S \rightarrow a_0 B | a_1 B && (3.14)\\
&\quad B \rightarrow b_0 C_A | b_1 C_B && (3.15)\\
&\quad C_A \rightarrow c_0 && (3.16)\\
&\quad C_B \rightarrow c_0 | c_1\} && (3.17)
\end{aligned}
$$

After the cover protocol's grammar G_{CP} is defined, the grammar for the micro protocol G_{MP} must be defined. Let us assume that our micro protocol comprises an ACK flag, a DATA flag and a DIS (disconnect) flag. To build sentences which are comparable to the sentences of G_{CP} by using the same terminal symbols, we apply the previously generated mapping. For instance, $ACK \equiv a_1, \neg ACK \equiv \neg a_0, DATA \equiv b_1, \neg DATA \equiv \neg b_0, DIS \equiv c_1, \neg DIS \equiv \neg c_0$.

As mentioned earlier, the grammar must produce sentences which contain terminals in the order of their mapped bits in the underlying protocol's header. This condition is necessary to produce comparable sentences by both grammars.

It is possible that G_{CP}'s behavior depends on other bits of the underlying protocol which do not belong to the cover protocol or that G_{CP}'s behavior depends on previous packets. Both problems can be addressed by introducing additional terminal symbols that represent additional circumstances. We discuss this problem in Section 3.4.4.

Step 6-b: Language Inclusion Testing:

We finally must verify whether the language $L(G_{MP})$ produced by the micro protocol grammar G_{MP} is a subset of the language of $L(G_{CP})$ produced by the cover protocol grammar G_{CP}. If this condition is true, G_{MP} cannot produce sentences which do not belong to $L(G_{CP})$. But to break the specification-conform bit patterns of G_{CP}, G_{MP} must be able to produce bit combinations which do not belong to $L(G_{CP})$'s language, which is not possible if the language subset condition is satisfied.

The final step of language-subset verification can be either done by hand or based on an algorithm. In the following, both approaches will be explained.

- *Automatic verification:*
  Baldoni et al. as well as Bouajjani et al. presented an algorithm for the automatic conformance testing, i.e., to test whether a protocol implementation is conform to its abstract specification, and therefore apply a language subset testing [BBM+05, BEF+00]; Baldoni et al. utilize the algorithm presented by Bouajjani et al.

  To apply the algorithm, it is necessary that G_{CP} is a regular language while G_{MP} can either be regular or context-free since it is not possible to decide whether $L(G_{MP}) \subseteq L(G_{CP})$ is true if *both* grammars are context-free [HU94].

  Baldoni et al. generated a formal grammar based on Agent UML (AUML) and on DyLog (a logic programming language based on modal logic). Afterwards, the language-subset must be tested to verify whether the first grammar (representing the implementation) is conform to the abstract specification [BBM+05]. While Baldoni et al. focus on the testing of an implementation, our goal is to ensure the cover protocol-conform behavior of a micro protocol. Baldoni's algorithm can be generally used for language subset tests under the mentioned conditions as long as the formal grammar is available. Since our approach is based on a formal grammar of the required Chomsky-type (context-free or regular), Baldoni's approach is suitable for our problem.

  Baldoni et al. and Bouajjani et al. test the inclusion of the context-free language A within the regular language B, i.e., $A \subseteq B$, by testing whether $A \cap \overline{B} = \emptyset$ [BBM+05, BEF+00]. Since the complement of a language is closed within the

regular languages, $\overline{B}$ will be regular if B is regular. The intersection of a regular language A and a regular language $\overline{B}$ is a regular language. The emptiness for a regular language is also decidable.

However, the intersection of a context-free language A and a regular language $\overline{B}$ ($A \cap \overline{B}$) is a context-free language [BBM+05]. For a context-free language A, $A = \emptyset$ is decidable (but $A \cap B = \emptyset$ is not decidable if B would be context-free as well [RS80, HU94]). The complement operation is also not decidable for context-free languages and it is not feasible to compute whether the complement of a context-free language is still a context-free language [HU94], i.e., $A \cap B = \emptyset$ cannot be decided in this case.

The algorithm would be applicable for two context-free grammars, if for G_{MP} could be proven that $L(G_{MP})$ is regular. However, it is not decidable whether $L(G)$ is regular if G is context-free [HU94].

To apply the automatic verification of $L(G_{MP}) \subseteq L(G_{CP})$, G_{MP} must be available as well as the deterministic automation for $L(G_{CP})$ as input. The algorithm takes $O(p \cdot s^3)$ time (p is the number of productions of G_{MP}, s is the number of states of the deterministic automation of $L(G_{CP})$) [BBM+05, BEF+00].

- *Manual verification:*
  Since a micro protocol only comprises a few bits and can thus only consist of few terminal symbols and is likely to comprise only few non-terminals and productions, a manual verification whether $L(G_{MP}) \subseteq L(G_{CP})$ is feasible.

  As already mentioned, it is not decidable whether $A \subseteq B$ if A and B are context-free languages. However, it is feasible to verify whether a given sentence s is part of a context-free language [HU94]. Thus, if only few sentences are generated, a manual verification of the language inclusion can be done.

  Theoretically, the number of productions can be high nevertheless and thus can increase the required workload for the language inclusion verification. We assume that $|P| \gg |\sum|$ is unlikely for micro protocols and thus, motivate a manual verification.

  For a manual verification, it is necessary to test whether every sentence creatable by G_{MP} can also be created by G_{CP}. For instance, if the micro protocol is capable of setting the "DATA" flag (indicating attached payload) and the "DIS" flag (terminating a covert channel connection) within the same packet, the bit mapping

must be generated. For this example we assume that the mapping is "ACK"="a", "DATA"="b", "DIS"="c", i.e., the sentence to be tested is

$$\{\neg ACK, DATA, DIS\} \equiv a_0 b_1 c_1 \tag{3.18}$$

If $a_0 b_1 c_1$ can be generated using G_{CP}, the tested micro protocol sentence is verified. The same procedure has to be applied for all other possible sentences of $L(G_{MP})$. Regarding to our previous mapping and grammar, $a_0 b_1 c_1$ is indeed feasible.

However, in case the rules do not allow the creation of a required sentence, e.g., only similar results are feasible, a re-engineering of either the cover protocol, of the micro protocol or of the mapping is mandatory. Therefore the re-engineering paths of Figure 3.9 must be used. The re-engineering process is discussed in Section 3.4.5.

3.4.3 Workload Reduction with Two Strategies

We pointed out that the grammar can be simplified if only *independent* bits of the underlying protocol are selected to become part of the cover protocol [WK12b]. Bits are called independent in this chapter, if their value does not depend on another bit's value. A bit that can only be set if another bit is (not) set is thus not independent.

A second simplification arises if we map 1-value bits of the micro protocol to 1-value bits of the cover protocol, i.e., if a micro protocol bit is set to "1", the cover protocol bit is also set to "1". This approach is linked to the drawback of sub-optimal bit mappings between micro and cover protocol but simplifies the grammar. Since only 1-bits are used, the number of required terminal symbols is reduced by 50%. In such a case the terminals $a...n$ represent $a_1...n_1$.

Using this approach, the previously discussed type-2 grammar can be reduced to

$$\begin{aligned}
G_{CP} &= (V, \sum, P, S) && (3.19)\\
V &= \{S, A, B, C\} && (3.20)\\
\sum &= \{a, b, c\} && (3.21)\\
P &= \{S \rightarrow AB|AC && (3.22)\\
&\quad A \rightarrow a|\epsilon && (3.23)\\
&\quad B \rightarrow b|\epsilon && (3.24)\\
&\quad C \rightarrow bc\}. && (3.25)
\end{aligned}$$

Similarly, the type-3 grammar can be reduced:

$$
\begin{aligned}
P &= \{S \rightarrow aB|bC|\epsilon && (3.26)\\
&\quad B \rightarrow bC|\epsilon && (3.27)\\
&\quad C \rightarrow c|\epsilon\} && (3.28)
\end{aligned}
$$

If such a representation is used, the non-inclusion of a terminal n in a sentence means that n_0 instead of n_1 is included – in other words, bit n has the value 0. For instance, ac represents $a_1b_0c_1$ if a terminal b exists and it means $a_1c_1d_0$ if a terminal d but no terminal b exists and so forth.

Not taking the mapping but only the micro protocol bit naming into account, the micro protocol sentences must not contain negated values, e.g., the testing for $\{ACK, \neg DATA, DIS\}$ must be a test for $\{ACK, DIS\}$ (or ac).

The manual as well as automatic verification for language inclusion is done in the same way as explained earlier.

3.4.4 Handling Connection-oriented Protocols

By applying formal grammar, we only took a plain cover protocol header into account. The presented approach did so far not cover previously sent or received packets or their states into account, nor did it focus on cover protocol bits depending on bits of the underlying protocol which were not part of the cover protocol. Therefore, the approach has to be extended to handle connection-oriented underlying protocols and non-cover protocol bits. However, it is recommended to build cover protocols in underlying protocols only in areas which are not connection-dependent to achieve an easier protocol designing process.

To extend our approach, we introduce additional terminal symbols representing states of previous network packets, which allows to stay conform with the existing approach and only directly influences step 6-a (*Grammatical Definition*) and step 6-b (*Language Subset Verification*) indirectly by increasing the workload due to additional (non)terminals.

Our approach of introducing additional terminal symbols with special meanings is not new. While the actual protocol's *operation* is not in our focus since we only focus on states of the micro protocol header instead of state changes, Harangozó, for instance, introduced special symbols for time-outs and the handling of unknown frames to define a protocol's operation already in 1977 [Har77].

In $\sum$, we distinguish between bits of the previous packet (bit i of a latest *received* packet is referred to as i_r) and the following packet (bit j of the packet to be sent is, as previously mentioned, simply referred to as bit j, or j_0/j_1, respectively). The bit i in the previously *sent* packet is referred to as i_s.

If the states of packets received prior to the latest received packet are required to be taken into account as well, additional numbering can be introduced, e.g., bit i of the packet received before the latest packet was received is referred to as $i_{r_{-1}}$. Similar bit naming must be applied if packets *sent* prior to the latest sent packet are required to be taken into account. Table 3.6 summarizes these proposed terminal symbol naming.

Bit	**Meaning**
i	Bit i in the currently assembled packet
i_r (equals i_{r_0})	Bit i in the latest received packet
$i_{r_{-1}}$	Bit i in the packet received before the latest packet
$i_{r_{-n}}$	Bit i in the n'th packet received before the latest received packet
i_s (equals i_{s_0})	Bit i in the latest sent packet
$i_{s_{-1}}$	Bit i in the packet sent before the latest sent packet
$i_{s_{-n}}$	Bit i in the n'th packet sent before the latest sent packet

Table 3.6: Naming of terminal symbols.

Alternatively, alias naming as done previously (e.g., bit i can be referred to as "ACK" or "a"/"A") can be used to identify bits because taking a previously received TCP RST flag into account is – especially for a manual analysis in step 6-b – easier if "RST" instead of i is used. Overall, it is unlikely, that many previous bits must be taken into account. The bits of the packets $p_{-1}...p_{-n}$ are very unlikely to become part of $\sum$ but are included for theoretical reasons since it is thinkable that protocols will be developed which do depend on the states of packets received before the latest packet.

Let us assume that bit b can only be set in the next packet, if the previously *sent* packet contained the bit m_s set *or* the previously *received* packet contained the bit n_r set, then the necessary production rules will be $X \rightarrow m_s|n_r$ and $B \rightarrow Xb|\epsilon$.

Similarly, additional terminal symbols can be introduced to specify the first or last packet of a connection. We use α to indicate the first and ω to indicate the last packet of a connection. If it is necessary to distinguish between received and sent packets, we use the previously introduced indications ω_r and ω_s. For instance, in the TCP handshake, the connection is established using the "SYN" flag from each side. To only allow the SYN flag (s) to be set within the first packet sent, we can create a simple production: $S \rightarrow \alpha s|\epsilon$.

Harangozó proposed to use terminal symbols with indexes to represent sequence numbers [Har77] and also added direction-dependent sequence numbering (i.e., it is possible to acknowledge received messages or to request the re-sending of a received message that contained errors), but his approach makes grammars more difficult. Later Harangozó proposed a multi-layer model using three different grammar levels to represent i) the frame generation process, ii) the generation of fields in the frames, and iii) the sub-field generation [Har78] which can ease the formal definition but requires three instead of one grammar. Thus, it is not recommendable to add underlying header components to a cover protocol which belong to connection-oriented aspects of a protocol.

However, it is necessary to model new terminal symbols in both the micro protocol and the cover protocol grammar, if connection-dependent terminal symbols are required. Otherwise, both grammars would produce different sentences, i.e., the grammars would not be comparable. The same condition applies for the inclusion of additional terminal symbols of the underlying protocol into the grammar to address dependencies: These terminal symbols must be included in both the micro protocol and cover protocol grammar.

3.4.5 Iterative Design

Our approach is designed to be iterative, i.e., steps can be selectively repeated to improve the resulting micro and cover protocol to finally fulfill the requirement of the last step: a micro protocol within a cover protocol that is conform to the rules of the underlying protocol. Re-designing paths are visualized by dashed arrows in Figure 3.9.

If an earlier step is performed again (e.g., to generate more cover protocol space or to reduce micro protocol functionality), the following steps must be repeated. Table 3.7 summarizes the possible modification cases.

The later the step is that is to be performed again, the less work must be done for the re-designing process. For instance, if step 5 is performed again to switch to an alternative mapping, no new cover or micro protocol has to be defined, but step 6 must be performed again to take the new mapping into account. If, on the other hand, step 1 is performed again, all following steps must be performed again as well.[10]

The three steps 2, 4, and 6 cannot be optimized on their own due to their dependence on earlier steps. Thus, repeating these steps can only be motivated to correct mistakes of earlier steps.

[10] If the micro protocol design stays the same and is still tiny enough to be placed into the cover protocol, step 3 and step 4 do not have to be repeated.

Step	**Possible Modification**
1 (Cover Protocol Definition)	increase the number of available cover protocol bits or reduce dependencies for bits to simplify the resulting grammar productions
2 (Probability List Generation)	*depends on step 1; modification not possible*
3 (Micro Protocol Design)	reduce the number of required micro protocol bits or simplify dependencies of micro protocol bits to simplify resulting grammar productions
4 (Micro Protocol Evaluation)	*depends on step 3; modification not possible*
5 (List Mapping)	switch to an alternative mapping to solve conflicts in step 6
6 (Conformance Verification)	*depends on the previous steps; modification not possible*

Table 3.7: Optimization techniques for our micro protocol engineering approach.

Example: The introduced sample grammar does not allow the creation of the message *ac* ($a_1b_0c_1$), i.e., to set the ACK and DIS flag within the same message without setting the DATA flag. For this example, we assume a micro protocol grammar G_{MP} requiring the sentence *ac*.

Different techniques can be applied to overcome this problem:

First solution (functionality exchange): To solve this conflict, the mapping of both lists (step 5) could be relocated in a way that the ACK flag and DATA flag are exchanged in their mapping, i.e., it will be possible to send (ACK,DIS) but therefore, it will not be possible to send (DATA,DIS), as it was possible before. Thus, a probably more important functionality is included and a probably less important functionality is excluded from the protocol to generate an underlying protocol-conform micro protocol. However, a re-mapping can result in a less stealthy communication (if no mapping to equivalent occurrence probabilities is possible).

Second solution (decrease micro protocol functionality): The requirement for the message *ac* without *b* is removed from the micro protocol's requirements (step 3). Therefore, *b* could be interpreted as "invalid" if no payload is attached and *abc* is sent.

Third solution (increase cover protocol space): The cover protocol could be re-designed to comprise an additional (independent) bit *d* (step 1). The additional bit can be used for the micro protocol to map it to the DATA flag. The drawback of this solution is that more cover protocol bits can raise more attention. If the occurrence rate of *d* differs from the occurrence rate of *b*, the replacement can lead to a less-perfect mapping, i.e., to more raised attention as well.

Sub Example: We assume that, like c, d can only be set to 1 if b is set to 1. The mapping between micro and cover protocol is now $a = ACK$, $c = DIS$, $d = DATA$ (bit b is only taken into account due to the dependence of the bits c and d but is not interpreted by the micro protocol) and the cover protocol grammar is now as follows:

$$
\begin{aligned}
G_{CP} &= (V, \sum, P, S) && (3.29)\\
V &= \{S, A, B, X\}, && (3.30)\\
\sum &= \{a, b, c, d\} && (3.31)\\
P &= \{S \rightarrow AB && (3.32)\\
&\quad A \rightarrow a|\epsilon && (3.33)\\
&\quad B \rightarrow bX|\epsilon && (3.34)\\
&\quad X \rightarrow c|d|cd\} && (3.35)
\end{aligned}
$$

DATA can now be mapped to d and b could be always set if c or d are required. Thus, the sentences *bc*, *bd* and *bcd* are feasible without taking b's meaning into account. Afterwards the sentence *abc* is possible and represents the previous sentence *ac*.

Fourth solution (multi-layer cover protocol): If feasible and if it results in a more suitable mapping, alternative layers of the protocol stack can additionally be taken into account. If the current cover protocol is DNS, it is thinkable to use additional bits of the UDP or TCP header for the cover protocol (both protocols can be used for DNS but in most cases, UDP is used).

These four sample solutions for the discussed problem as well as the mentioned re-designing options of Table 3.7 substantiate the dynamic, iterative designing process of our approach. The arrows in Figure 3.9 visualizes the whole process and the possible re-engineering paths (dashed arrows) again.

3.4.6 Results

Our two-layered approach aims on providing a means for micro protocol engineering with the output of an implementation-ready micro protocol. The approach can be applied to all underlying network protocols with binary headers and allows to freely define own micro protocols. The resulting micro protocol is conform to the behavior of the underlying protocol and mapped to the underlying protocol (through the cover protocol) in a way that it minimizes the raised attention of the covert communication.

Since a micro protocol and the utilized bits within an underlying protocol (i.e., the cover protocol) are usually of a limited size, the grammar verification can be considered a quick process.

Another advantage of our approach lies in the dynamic re-designing options: The protocol designer can switch between the steps of the approach (cf. Figure 3.9) and can optimize the micro protocol as well as the cover protocol and the mapping between both protocols incrementally.

Since we focus on headers with binary bit values, the approach is not designed to work with non-binary (i.e., plaintext) protocol headers such as HTTP or NNTP. However, the approach can be adapted to the circumstance of a plaintext protocol if no bit values are taken into account but plaintext values are used instead. For instance, if a covert storage channel utilizes the "User-Agent" version number in the HTTP header, two terminal symbols a and b could represent two different browser attributes and thus, it would be feasible to create a grammar. In the same way, other plaintext string values can be referenced by terminal symbols (e.g., the name of the "User-Agent" itself or an attribute within a NNTP posting's header). In [TF12], a context-free grammar was used for Spam filtering in email messages and the covert channel messages in mail headers and plain-text protocols can be modeled in the same way for our purpose. However, to ensure an automatic language inclusion testing, context/regular grammars must be applied as well.

A disadvantage of the approach is the raising complexity of the grammar if connection-oriented protocols are used for a covert channel and bits of previously received or sent packets must be integrated into the grammar using additional terminal symbols. There is also only a grammar complexity increasing option available to build sequence and acknowledgement numbers into a formal grammar.

3.5 Status Updates

As mentioned earlier, one goal for a network covert channel is to keep a low profile. For covert storage channels containing micro protocols, this goal can be supported by minimizing the size of the micro protocol. If less micro protocol information is required for a data transfer, fewer overhead will be produced and less attention will be raised.

3.5.1 A Space-efficient and Dynamic Header

While other covert channel-internal control protocols were already presented (they were introduced in Section 2.6.2), we will show that our protocol design is more *space-efficient* than the existing protocols. Another advantage of our protocol is the *dynamic header design* in comparison to the static header design of the existing protocols from other authors.

To achieve a space-efficient and dynamic micro protocol, we introduce a technique called *status update*. A status update is based on the idea to only transfer information if the status of some attribute within the covert channel changes, but not, if a status remains as it is. Thus, if a status is set, it does not change without an explicit command.

For instance, a status update could configure the destination address of a covert channel connection on a peer x used to forward the payload to another peer y. After the destination address is configured, a new status update for the destination address will only be required if the destination address of the communication changes. If no new destination address changing status update will be received by x, all packets will use the latest configured destination address without an explicit specification within the following packets.

If status updates are used, they build the main component of a micro protocol as they provide meta information about the other parts of the micro protocol as well as about the covert channel's payload.

3.5.2 Adoption of Existing Protocol Features

To realize status updates, we apply common techniques from the field of protocol engineering to covert channel-internal control protocols. The IPv4 protocol contains a value (the so-called *Internet Header Length*, IHL) that specifies the size of the IPv4 header. If the IHL is higher than 5 (which is the standard header size for IPv4), the header contains optional components that have to be evaluated. In the optional header, a value indicates the type of the optional header (e.g., a "strict source route" option or a "record route" option). The encapsulated header is specified via the "protocol" field in the IPv4 header. This concept is efficient since header components, which are required in only a minority of cases, are not always part of the IPv4 header and thus, the overhead of a connection is reduced.

In IPv6, the IPv4 fields "protocol", "IHL", and "Options" were unified in the "Next Header" field. The Next Header field specifies either an IPv6 extension header (similar to

the IPv4 options) or an encapsulated protocol (e.g., TCP or a tunneled IPv4 datagram).

Another space-efficient approach was presented for the *Serial Line Interface Protocol* (CSLIP) [Jac90]: CSLIP only transfers the header parts which are required for the packet. The CSLIP protocol therefore contains a bit mask that specifies the included header components.

CSLIP would be a valuable technique for micro protocols but it does not allow to dynamically change the order of header components nor does it allow the multiple occurrence of a header component in a single packet. Our status update approach overcomes both disadvantages.

3.5.3 How Status Updates are Used

We combine the mentioned techniques of the IPv4 Options, the IPv6 Next Header, and CSLIP to the concept of **status updates**. A status update is a value placed in front of each header component. The status update concept can be understood as being similar to an IPv4 header that only contains the IPv4 options without the main header (or as an IPv6 header only containing the field Next Header), and that transfers only the required header components similar to CSLIP.

We assume sender and receiver of a covert channel packet use the same values to identify the same header components. We call the different micro protocol components a **Type of Update** (ToU). For instance, one ToU could identify the micro protocol component that sets the source address of a connection and another ToU could identify the component that sets the destination address of a covert channel. Table 3.8 shows four sample ToU values which would fit in a two-bit field. However, these ToU values are only examples and each protocol designer has to identify own required ToU components.

ToU	**Meaning**
00	SET SOURCE ADDRESS
01	SET DESTINATION ADDRESS
10	END OF UPDATES
11	PAYLOAD FOLLOWS DIRECTLY

Table 3.8: Four sample status update messages.

Another example besides Table 3.8 can be found in [BWK12] that presents ToU values for a whole covert channel-internal routing protocol based on status updates.

The previously mentioned *protocol switching* capability of protocol hopping covert channels could, for instance, be implemented by defining an additional ToU for a protocol

switch. The protocol switching ToU could be followed by a protocol identifier of a static size.

Status updates can be combined to larger updates. The number of possible status updates per message is only limited by the provided cover protocol space. If, for instance, the sample ToUs 00 and 01 of Table 3.8 including their address payload will fit into a single packet's cover protocol space, they can be combined.

Example: We assume that our overlay network uses the IPv4 underlay addresses as overlay addresses as well (i.e., no own covert channel-internal address notation is used) and that our overlay network supports forwarding capabilities (via peers acting as proxies). System A is connected to system B via the covert channel proxy X. A wants to configure proxy X to send all following packets to B with the source address of A. Therefore, A sends the status update data "00" (see Table 3.8) followed by its own IP address, followed by "01", followed by the address of B, followed by "10" to indicate the end of the status updates. Figure 3.11 visualizes the micro protocol data for this example.

Figure 3.11: A simple sequence of micro protocol headers

If no ToU is defined to indicate the "END OF UPDATES", it is necessary to define a static order of ToU values, e.g., by ascending ToU values and another indication for the end of a micro protocol header (e.g., the rule that one ToU of a selected subset of all ToUs is always the last part of the header). Not including an "END OF UPDATES" ToU value provides the advantage of saving one ToU and thus, provides room for additional other functionality.

In comparison to SLIP, our protocol *can* provide slightly less space efficiency since we do not indicate the status updates with a single bit mask but insert the ToU value in front of every update. If we have n different header components, CSLIP would require a bit mask containing n bits to indicate the presence of each header component. Our status update approach, on the other hand, requires $\lceil \log_2 n \rceil$ bits for each of the n ToUs. On the other hand, if only few ($i \ll n$) ToUs are required for a packet, status updates provide a better space efficiency.

However, it is not necessary that a ToU will be defined for each header component, i.e., a single ToU can be used for multiple header components, which decreases the required space for ToU values in the micro protocol header. This problem is described and exemplified in the following Sections 3.5.4 and 3.5.5 where the space efficiency for a status update based micro protocol is verified.

Existing status updates can be extended to allow the *evolution of a micro protocol*. Therefore, at least one possible status update value must be still definable within the existing ToU list (i.e., if n bits are used to define status updates, only $2^n - 1$ ToUs can be defined to allow the integration of an additional ToU). Optionally, the additional ToU can be a *meta ToU*, i.e., a ToU followed by extended ToU values. For instance, if only one possible ToU value is left for definition, this ToU value can be defined as a meta ToU value. A meta ToU value is followed by a n bit ToU value extension that represents new ToU values and optionally leaves space for an additional extension. Other ToUs could be followed by a version number as proposed in Section 3.3.7.

However, the advantages of status updates (also in comparison to the CSLIP approach) are

1. to provide a more dynamic header design since status updates allow to place the same ToU multiple times within a single packet as well as

2. the ToU values allow to specify the order of the status updates within each packet, and

3. status updates *can* decrease the packet count for a transfer.

4. As mentioned before, status updates allow the evolution of micro protocols using *meta ToUs* – a useful feature for upgrading covert channel overlay infrastructure.

5. Status updates provide a better space efficiency for the protocol header than CSLIP if only a small number of the available status updates are to be transferred per packet in average.

Example for the advantages 1, 2, 3 and 5: A wants the previously mentioned proxy X to forward the character "q" to B and to C. If all information fits within a single covert channel packet, A only needs to send one packet instead of two packets to X as would be required for CSLIP (**Adv. 3 and 5**). Therefore, A inserts the ToU 01 ("SET DESTINATION ADDRESS"), followed by the address of B, attaches the payload (ToU 11; if a dynamic payload length is possible, followed by the payload length "1"), followed

by the payload ("q") (**Adv. 2**). Behind that information, A attaches ToU 01 again (**Adv. 1**), followed by the address of C and attaches the payload for C (ToU 11, length "1", data "q"), followed by the ToU 11 to indicate the end of the updates.

3.5.4 Initial Connection Efficiency Problem

We assume that the NEL phase was successful and a new connection between two systems within the overlay network shall be created for a new transaction. Therefore, a sequence of status updates (i.e., a combination of status updates as shown above) is required to configure the connection. For instance, the initial source and destination addresses or the configuration of a covert channel proxy are required to be done.

However, if only few packets are required to be transferred for a transaction, status updates can, as mentioned previously, produce an overhead in comparison to a micro protocol with a static header that does not require ToU values in its header.

Thus, there is *no* advantage of status updates for an *initial* covert connection setup (compared to usual covert channel protocols or CSLIP) because many ToUs of the size $\lceil \log_2 n \rceil$ must be transferred instead of only n bits as in case of CSLIP. On the other hand, status updates *are* an advantage compared to usual static headers (like IPv4) and CSLIP if few changes happen within an already established covert channel connection (as well as if only payload is transferred) since the packet size of all non-initial packets is significantly decreased (**Adv. 5**) because there is no need for an usual and comparatively large static header. The next section will also verify this result.

Therefore the protocol designer has to choose an optimal number of ToUs (if there is a header component that has the size of a ToU, it can probably become part of another ToU to save space in average transactions). Afterwards, a good coding (e.g., Huffman) should be used for the ToU values. The next Section 3.5.5 will also exemplify this aspect.

3.5.5 Design Procedure and Re-design of an Existing Protocol

This section compares the size of an existing micro protocol with a status update-based protocol design to show the advantages and disadvantages of the presented approach. Besides, this example shows the procedure for a status update protocol design.

We already introduced the protocol by Ray and Mishra in Section 2.6.2. To compare the protocol by Ray and Mishra with a status update-based version, we re-designed the protocol.

The protocol by Ray and Mishra is shown in Figure 3.12-a. The header contains a static design and has a size of 8 bits. The header comprises a 2 bit sequence number, a data flag (the flag is set if payload is attached), an acknowledgement flag, a 2 bit expected sequence number, a flag indicating the start of a transaction, and a flag indicating the end of a transaction.

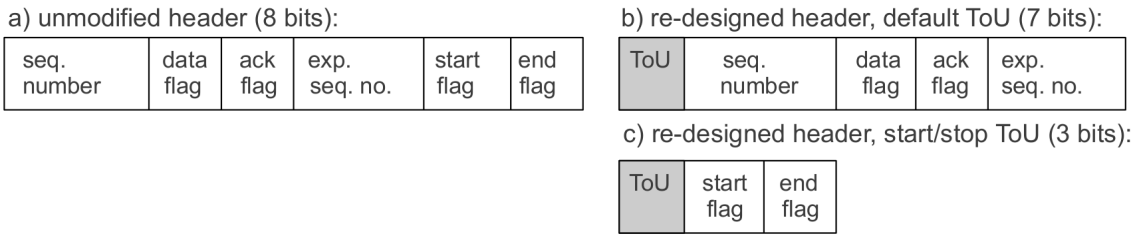

Figure 3.12: a) The header from [RM08a], b/c) Re-designed status update headers.

To create a status update-based version of the micro protocol, we have to select header areas which can be excluded in some cases, i.e., header components which are not required for any packet. In the protocol by Ray and Mishra, only the two last bits (the start and stop flags) are not mandatory header components since only required at the beginning as well as at the end of a transaction. The larger the area to exclude, the better the space efficiency (in this case, it is only 2 bits since the micro protocol by Ray and Mishra is already space-efficient).

In the second step, a list of required ToUs has to be defined. To split the header in two components (default header and start/stop flag header), we need two ToUs that are shown in Figure 3.12-b/c. Theoretically, the second ToU indicating the start/stop of a transaction can occur multiple times while the default ToU cannot occur multiple times since no payload length is specified – a feature that is also not foreseen in the original protocol. Thus, the status update-based design is still advantageous since the start and stop flags can occur multiple times, although such a multiple occurrence is not required, even if no "PAYLOAD FOLLOWS" ToU is provided.

In the final step, the usefulness of the status update-based protocol has to be verified. This can either be done by comparing the traffic recordings of two proof of concept implementations for equal use cases, or by a theoretic calculation as done for this thesis:

Using our status update version of the Ray/Mishra protocol, we were able to reduce the default header size from 8 to 6 bits. However, since the ToU value needs to be added in front of the header (it has the size of $\lceil \log_2 \#ToUs \rceil = 1$ bit), we can maximally save 1 bit per packet. The second header also requires one bit for the ToU value and two bits

for both flags. We defined no "END OF UPDATES" ToU to save one ToU value and since the protocol by Ray and Mishra did also not allow header components to occur multiple times. In other words, we use a status update design with a *static* ToU order (as proposed in Sect. 3.5.3).

In the following, we compared the size of an unidirectional transaction with a varying amount of packets to evaluate the usefulness of our approach in this final step. This step highly depends on the protocol and the separated functionality. For instance, if we would have an IP-like micro protocol and would extract the source and destination addresses, we would have to simulate situations with different address changes to compare both protocols. On the other hand, if we would extract the fragmentation functionality, we would need to simulate network fragmentation scenarios to compare both protocols.

In any case, the goal is to verify whether the new status update-based protocol requires more header bits per transaction than the existing protocol and thus, can be considered useful.

In the case of the new status update-based micro protocol, each packet used to signal the beginning or the end of a transaction is larger than the originally required packet (7+3=10 bits instead of 8 bits). Thus, if we transfer only 2 packets within a whole transaction, the status update headers would require 20 bits instead of 16 bits which makes the status update-based protocol inefficient.[11]

Each packet of a transmission (excluding the packets signaling the start/end of the transaction) is shortened by 1 bit (7 instead of 8 bits) and for each transaction, two packets must contain additional 3 bits for the ToU containing the flags (one for the start and one for the end of the transaction).

If $n \geq 3$ packets are transferred, the status update-based protocol requires 20 bits for the first and the last header as well as $7(n-2)$ bits for the remaining headers instead of $8n$ bits required by the original protocol. Thus, it takes $n > 6$ packets to achieve a better space efficiency than the original protocol. If $n < 6$ packets are transferred, our protocol is less space-efficient. In general, the number of header bits of the status update-based protocol is $n-6$ less than the original header size if n packets are required for a transaction (not taking the payload size into account). Thus, if n is high, the status update-based protocol approaches 7/8 of the originally required size.

[11] Additionally, less payload can be included within the initial and the last packet of a transaction since additional space is required for the status update-based micro protocol. In the following, only the plain header size is taken into account since payload sizes can vary and do additionally depend on the size of the cover protocol. The transferable payload increases for non-initial micro protocol packets due to the additionally available 8th bit. However, since the protocol by Ray/Mishra does not include a length value for the payload, a comparison is not directly feasible.

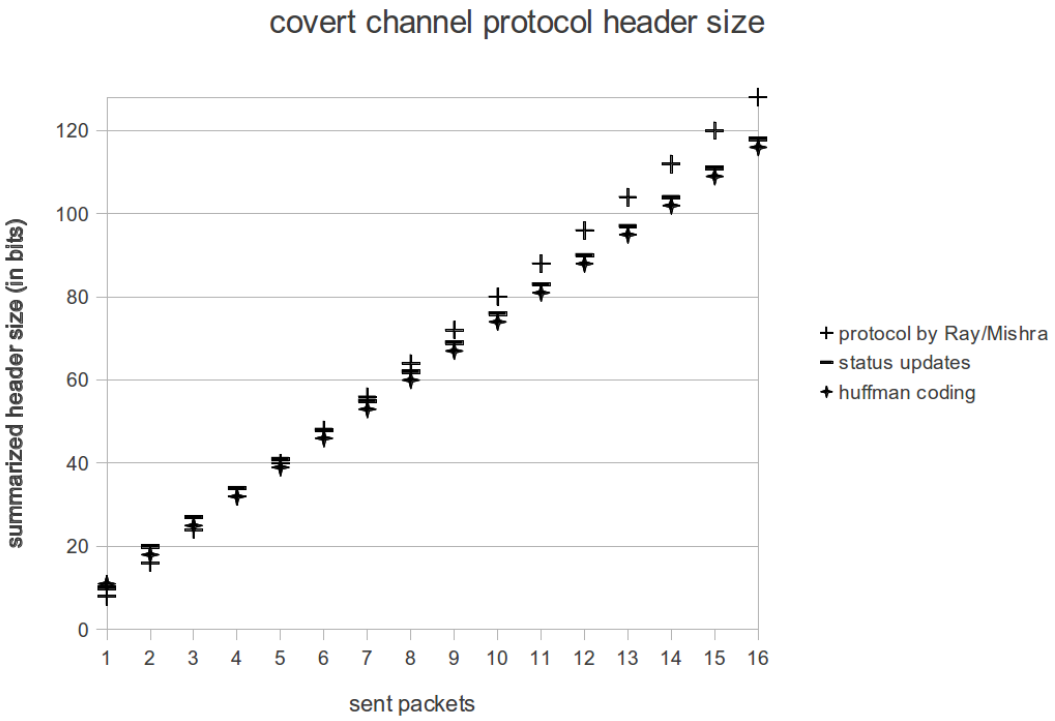

Figure 3.13: Comparison of the summarized header sizes of the original micro protocol and the re-designed micro protocol. The new design is advantageous if a transmission comprises at least 7 packets (two ToUs, no Huffman coding, start/stop ToU header can theoretically occur multiple times) or 5 packets (three ToUs, Huffman coding, static order of header components, no multiple occurrences of ToUs possible).

Figure 3.13 compares the original and the status update-based header design.

An improvement over the presented status update-based re-design is possible if we apply a huffman coding in the preamble of each micro protocol header. This design is similar to the CSLIP approach (but with Huffman coding for the preamble) and linked to the CSLIP disadvantages (no freely definable order of header components and no multiple occurrences of header components). However, a comparison between such a CSLIP-like version of the protocol by Ray and Mishra as well as a status update-based protocol with a single preamble instead of multiple ToUs is interesting since it does not lack any of the original features by Ray and Mishra while still providing a better space efficiency.

Therefore, the following preamble values (visualized in Figure 3.14) can be used:

- '0' indicates the default header,
- '10' is followed by a 1 bit value that either starts (1) or stops (1) a transaction, but not both, followed by the default header,
- '11' is followed by a 2 bit value that comprises a start as well as a stop flag, followed by the default header.

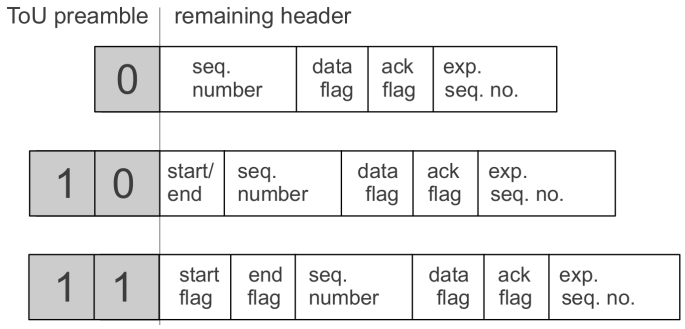

Figure 3.14: CSLIP-like micro protocol re-design of the protocol by Ray and Mishra using Huffman coding with 3 different CSLIP-like preamble values.

For transactions comprising only one packet, '11' is used to signal the start and stop of a transaction within the same packet. Otherwise '10' is send within the first and the last packet to signal the start and afterwards the stop of a transaction while all packets sent in between contain the '0' value in the preamble.

The improved design would only require 5 packets for a transaction to provide better space efficiency than the default header by Ray and Mishra.

The results of this improved approach are visualized in Figure 3.13 as well.

3.5.6 Efficient Re-Design of *Ping Tunnel*

The status update-based version of the protocol by Ray and Mishra results in a slightly improved space efficiency. If a micro protocol containing components with more than a few excludable bits, the efficiency of status update-based designs can improve significantly.

As an example, the previously mentioned *Ping Tunnel* protocol can be optimized: If the sender configures the destination address and destination port using status updates, the address and port only need to be used for the initial part of the channel (and if an address or port change is required) and thus, multiple bytes per packet can be saved.

The Ping Tunnel documentation says that the *IP and port fields are only used in packets from the client to the proxy*, i.e., a status update with a size of one bit can differ between both ToUs, the default message ToU and the address-and-port ToU. For a non-initial message (or a transfer without a proxy) containing only the default header, the header size can therefore be reduced by the $sizeof(IPv4address)+sizeof(portnumber)-sizeof(ToU)$, e.g., the header can be reduced by $32 + 16 - 1 = 47$ bits. However, since the port field in Ping Tunnel uses 32 bits instead of the required 16 bit, even 63 bits can be saved per non-initial packet which is a significant improvement since it allows additional payload per packet and can thus also decrease the overall number of required packets per transaction.

3.5.7 Status Updates for Dynamic CC-Overlay Routing

Backs developed a micro protocol that provides *Optimized Link State Routing*-based dynamic routing for covert channel overlays [Bac12]. His micro protocol is based on the presented status update concept and comprises only 4 additional ToUs for the implementation of his routing algorithm [BWK12]: The first ToU is used to request a complete table of peers (and their properties) as well as a table of the topology; the second ToU is used to respond to such a request; the third ToU is used to propagate updates of the peer table and the fourth ToU is used to propagate topology table updates.

The routing algorithm can optimize the routing path in a way that it provides a maximized covertness (called *Quality of Covertness*) and was adopted from mobile routing environments since the fast changing infrastructure of mobile environments (e.g., a moving mobile smart phone user) is similar to covert channel overlays in which the components of the underlying network can be replaced at any time without informing the covert channel users about the infrastructural change.

Besides, Backs integrated the concept of *agents* and *drones*. While agents are aware of the covert channel existence and micro protocol operation, drones are not aware of the covert channel existence or micro protocol operation [BWK12]. This concept is similar to the use of *Flickr* for collaborative communications [BK07] and can be based on proxy-like web-services (e.g., google website translation services) as proposed by [Mem07].

3.5.8 Identifying Status Updates

Different micro protocol messages could arrive in an order unequal to the order in which they were sent (e.g., one network packet went over a faster route and outstripped an-

other). For instance, let A be a covert channel sender. A sends a destination address (B) status update followed by a payload packet to X. X will forward the payload to the destination B.

If the payload packet arrives earlier (e.g., because it was routed over a faster route or the first packet got lost), X will deliver the payload to the destination address currently configured instead of sending the payload to the destination address specified in the update (since it arrived too late or did not arrive).

Thus, it is significant to reconstruct the original sequence of status updates on the receiver side. A simple and space-efficient solution for this problem could be to use identification bits (e.g., an ID or sequence number incremented for each packet) within the covert data space. This simple technique is used since years in different covert channel tools, e.g., in [Stø09]. The authors of [RM08a] estimate that two bits are enough space for a sequence number within a covert channel.

3.5.9 Results

Status updates can be considered a space-efficient and dynamic means to implement micro protocols. They combine ideas of previous protocols (like CSLIP) but provide a dynamic header design (ToUs can occur multiple times and at different locations within a micro protocol packet).

Since the space efficiency of existing protocols is not provided in any case, we mention different aspects to be taken into account (the ToU coding used, like Huffman; the number of ToUs; the selection of header areas to be split from the original header) and propose to theoretically compare or practically evaluate header sizes of protocols to be re-designed before implementing a status update-based micro protocol.

Additionally, status updates enable the evolution of a micro protocol by extending a protocol design with new ToU values and meta ToU values. Protocol versions can be distinct using the version numbering method described in Section 3.3.7 in the context of the NEL phase.

Since ToUs are expandable, status updates can be used to identify all kinds of possible operations, e.g., underlying protocol switches or commands for dynamic routing processes.

Besides, status updates can decrease the overall number of required packets per transaction and thus, lower the raised attention of a network covert channel similarly to the optimization discussed in the Sections 3.3.8 and 3.3.9.

The presented concept of status updates was practically applied by Backs in [Bac12] and [BWK12] to develop a covert channel routing protocol of a small size. Backs used *Optimized Link State Routing* (OLSR) to achieve a covert channel overlay routing for mobile environments.

3.6 Conclusion and Future Work

This chapter dealt with the topic of covert channels with internal control protocols, so-called *micro protocols*. A terminology fulfilling the requirement of an exact distinction of views (underlying protocol, cover protocol, and micro protocol) was introduced.

Micro protocol-based network covert channels enable covert channels to provide a backward compatibility and the use for mobile environments. Micro protocols also improve the robustness of covert channels and enable dynamic routing in covert channel overlays.

The problem of the so-called *network environment learning phase* (NEL phase) regarding the potential presence of traffic normalizers on the path between sender and receiver was discussed and it was shown that a normalized NEL phase leads to a two-army problem. The most suitable technique to minimize the problem is to use a temporary third participant to set up a new overlay path between two peers.

This chapter additionally presented a protocol engineering framework for micro protocols. The approach must be considered as a covert channel-specific additional approach because protocol engineering approaches for normal network protocols are already available since decades. The reason for creating an additional approach was to minimize the raised attention for micro protocols by optimizing the mapping of micro protocol bits to bits of the cover protocol as well as to ensure that a micro protocol does not violate the rules of the underlying protocol while operating. Therefore, the micro and cover protocol are modeled using a context-free or regular grammar and the conformity of the micro protocol is verified by a language inclusion test. The framework was designed to work for all underlying network protocols with a binary header while a modification of the approach to work with plaintext protocols is thinkable.

Since larger micro protocols raise more attention than smaller micro protocols as they manipulate additional bits in the cover protocol (and with it: the underlying protocol), a technique was developed and evaluated to minimize the size of micro protocol headers which provides a dynamic header design. The concept is called *status updates*. Status updates transfer only required header update information for “states” (e.g., to update the

source or destination address of a covert channel). To validate the usefulness of status updates, two existing protocols were re-engineered using status updates and it could be shown that an already optimized research protocol could be additionally optimized. Besides, status updates can decrease the number of packets required for a transaction and were already applied for an efficient dynamic routing in covert channel overlays [BWK12].

Since status updates are currently not usable in conjunction with the presented protocol engineering approach, a user can either optimize for a small micro protocol or for a micro protocol conform to the behavior and bit occurrence rates of the underlying protocol, but not both. Future work will thus comprise the combination of the formal grammar-based protocol engineering approach with status updates. Due to the dynamic header design, status updates can change the micro protocol's header structure in every packet and thus, require a dynamic mapping of micro protocol bits to cover protocol bits. Besides, the micro protocol engineering framework will be adapted to plaintext protocols.

4 An Active Warden to Counter Protocol Switching Covert Channels

Protocol hopping covert channels (PHCC) were discussed in the previous chapters and can be considered a valuable technique for covert channels with micro protocols. No prevention technique for PHCC has been developed so far. The same problem applies for protocol channels (PC) since no means are available to counter these channels either. The limitation of both channel types is a challenging task since current limitation means focus on covert channels using a special technique. Hence, such existing limitation means can only address few of many possible storage channels used in a PHCC that is capable of switching to another protocol if it is blocked. Thus, there is a need for a technique that counters PHCC as well as PC.

In this chapter, we present the first active warden able to counter both PHCCs and PCs. Our active warden aims on reducing the bitrate of both channel types. The key idea for a bitrate limit is to delay transmissions with protocol switches since both channels are linked to protocol switches by design. The approach is validated by experiments and its practical usefulness is evaluated as well.

Figure 4.1 visualizes the link of Chapter 4 to the topics of the previous chapters.

As a new category for such covert channels, the term *protocol switching covert channel* (PSCC) can be used. In the following we use the term PSCC to refer to both PHCC and PC, and if only one of both types is addressed, the terms PHCC or PC are used.

4.1 Concept

PHCC and PC both require that packets arrive in the same order at the receiver as they were sent. By introducing a delay, the packet order can be manipulated by an active warden. Thus, our warden introduces delays only if a protocol switch is taking place. Therefore, the warden monitors the traffic flows on the network in a way that it always remembers the latest protocol used by a sender. If the protocol of a sender switches, the

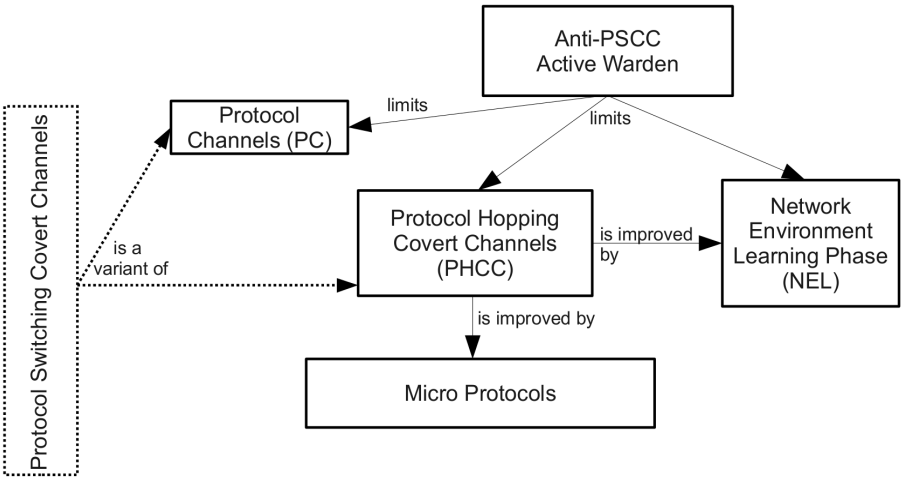

Figure 4.1: The active warden's linkage to topics discussed in previous chapters.

new packet is delayed, otherwise it is directly forwarded.

The active warden must be located on the path between a potential covert channel sender and a potential covert channel receiver (Figure 4.2). For instance, if the goal is to limit covert channels that exfiltrate confidential information from an enterprise network, the active warden could be installed on the network uplink.

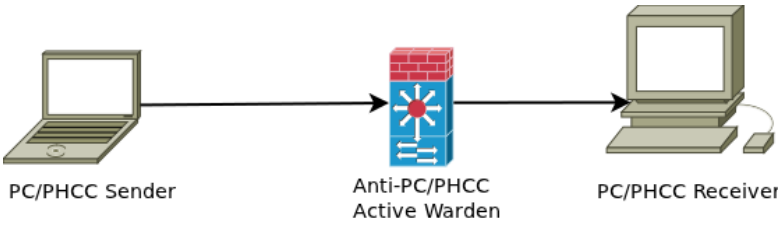

Figure 4.2: Location of the Anti-PC/PHCC active warden

As pointed out by Keller, introducing a delay on protocol switches results in an optimization problem [WK12a]: The higher the delay d is, the smaller is the remaining data transfer rate $R(d)$ for the covert channel, but the higher are the side effects (e.g., modeled in a function $S(d)$) for legitimate traffic. Thus, an administrator has to choose d regarding to his priorities [WK12a].

Example: We imagine a PC based on ICMP and UDP where ICMP represents "0" and UDP represents "1" and the message to be transferred is "0010001". Thus, the sender sends the packet sequence ICMP, ICMP, UDP, ICMP, ICMP, ICMP, UDP. An active warden is located on the path between the sender and the receiver and delays the packets for a time $d = 1$s.

The first two ICMP packets are forwarded directly because no protocol switch is taking place. The following UDP packet results in a protocol switch and thus, will be delayed for 1s. The same delay applies for the next ICMP packet that again results in a protocol switch. The two following ICMP packets are directly forwarded and the last packet (UDP) is delayed again due to the protocol switch. Since d is constant and $d = 1$s is higher than the whole message transfer time of the 7 packets on a modern network, all delayed messages will arrive after the last non-delayed packet arrives.

For larger messages, delayed packets do additionally scramble those packets that are currently on transfer. We evaluated larger messages within our experiments as will be discussed later.

The output of the active warden is therefore: ICMP, ICMP, *ICMP*, ICMP, *UDP*, ICMP, UDP, or "00*0*0*1*01", i.e., the received message comprises two errors.

For PHCCs, the hidden message is not represented by the protocol used but by the packet's content (e.g., in selected bits of the IPv4 TTL or in the TCP ISN). However, since PHCCs also use protocol switches, a delay on protocol switches jumbles the channel's payload as well.

4.2 Bitrate Calculation

A first formula for the bandwidth calculation of local covert storage channels was presented by Tsai and Gligor [TG88]:

$$B = b \cdot (T_R + T_S + 2 \cdot T_{CS})^{-1}, \tag{4.1}$$

where b is the amount of information transferred per message, T_R the time required to receive a covert message, T_S the time required for sending the covert message, and T_{CS} the time required for the context switch between the processes.

While the formula by Tsai and Gligor calculates the bandwidth of a covert channel, we aim on limiting the exact bitrate a PC/PHCC can use depending on the delay d introduced by the active warden. Therefore, we modified the formula in a way that it calculates the maximum error-free bitrate of a covert channel using a uniform distribution of symbols in case the active warden is present. We therefore introduce Formula 4.2.

$$B = b \cdot (p \cdot d + T)^{-1} \tag{4.2}$$

In this formula, d is the applied delay of the active warden and p the probability of a protocol switch for a packet.

For a PC with $n = 2$ underlying protocols, we have two states and thus, $b = \log_2 n$. Since a higher n results in a higher probability for a protocol switch, p raises too, if n raises.

For a PHCC, different underlying protocols provide different cover protocol space $sizeof(CP)$ and thus, b is not linked to the number of underlying protocols n but is the average cover protocol space $b = sizeof(CP_{avg})$ (earlier discussed in Chapter 3.1). Also no link between p and n is given for a PHCC since the channel's coding is not based on the protocol used but on the content and thus, the PHCC can choose the probability switch on demand.

We use T instead of T_R, T_S, and T_{CS} to specify the *transmission* or *gap* time. If T represents the minimal time difference between two packets of the PSCC, we call it the *gap time*. We call T the *transmission time* if the channel sends packets in a sequential manner, i.e., a new packet is not sent to the receiver until the previous one got received (acknowledgements are not taken into account). The transmission time comprises the time required for sending, transferring, and receiving a packet. The gap time is equal to the transmission time in such a sequential communication.

However, in both cases T highly depends on the technical environment and the effect of the technical environment to T is larger than the difference between transmission and gap time in practice. Thus, we do not differ between gap time and transmission time in the remainder and only focus on the transmission time.

4.2.1 Protocol Channels

In theory, the probability of a protocol switch for randomized input using a uniform coding and $n = 2$ protocols is $p = 0.5$: Either, the next packet switches the protocol, or the next packet is of the same protocol as the previous packet. In our experiments, p was slightly lower (0.4738806, cf. Section 4.5.1). However, p also depends on the coding used in the PC (e.g., a Huffman coding causes a different p value, as we will discuss later). In general, we can say that a PC utilizing n protocols is linked to a protocol switching probability of $p = (1 - 1/n)$. Therefore, we can modify Formula 4.2:

$$B = b \cdot ((1 - 1/n) \cdot d + T)^{-1} \tag{4.3}$$

For the same scenario (uniform coding, randomized input), we know that $b = \log_2 n$ and thus,

$$B = \log_2 n \cdot ((1 - 1/n) \cdot d + T)^{-1}. \tag{4.4}$$

We calculated B in case of a present active warden taking different delays into account. The results are shown in Figure 4.3. We therefore used the range of the measured values for T obtained from the PCT program [Wen09a] (introduced in Chapter 2.4.6). As the results show, the active warden can theoretically decrease the bitrate of a PC to less than 1 bit/s using a delay of $d = 2.0s$.

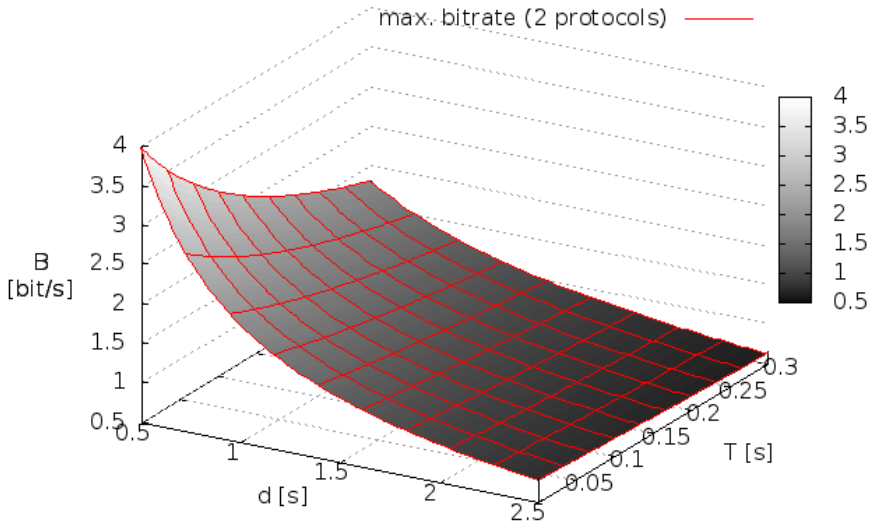

Figure 4.3: A PC's bitrate (B) using a set of $n = 2$ protocols depending on the delay and the transfer time.

4.2.2 Protocol Hopping Covert Channels

Since the amount of transferable information b varies for PHCCs regarding to the cover protocol space in each underlying protocol, b is of higher importance than T. Thus, the following plots focus on different values of b and d instead of T and d. For T we used the average measured value $T = 0.005$s of a virtual LAN between two virtual machines.

Because p is not directly linked to n as it is in the case of PCs, we run two different

calculations: First, we calculated B for a PHCC utilizing $n = 2$ protocols (shown in Figure 4.4) and afterwards, we calculated B for $n = 4$ protocols (shown in Figure 4.5).

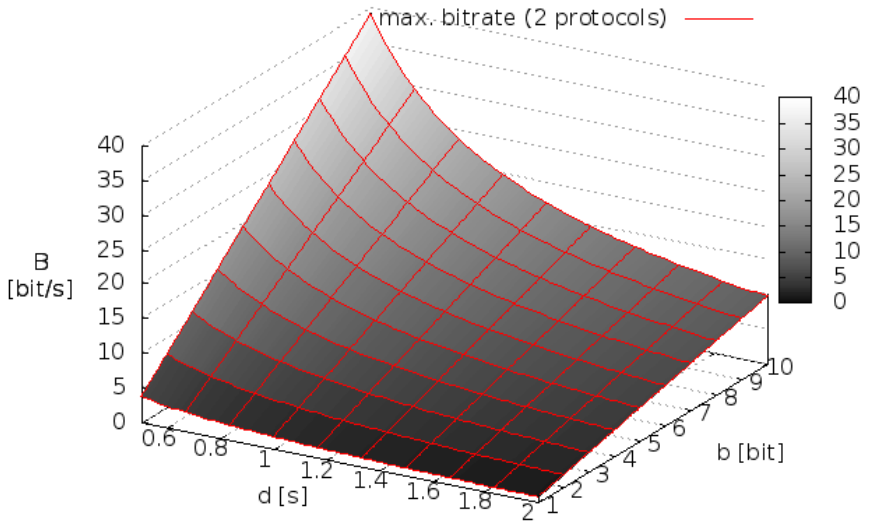

Figure 4.4: A PHCC's bitrate using $n = 2$ protocols, $T = 0.005$s and delays between 0.5s and 2s as well as the capability to transfer between 1 and 10 bits per packet.

Since the input values are the same as for a PC in case of $b = 1$ and $n = 2$, B for a PHCC with the same conditions is equal to B of a PC. However, b and thus B can be higher in case of a PHCC since it can transfer more hidden bits per packet. Besides, p is usually smaller for a PHCC which also contributes to a higher B.

If the number of underlying protocols n increases, the PHCC bitrate B decreases since more potential protocol switches can be made. We can see this effect for $n = 4$ protocols in Figure 4.5.

The following section explains the experimental implementation of the active warden. We afterwards validate the theoretic calculations for B.

4.3 Improved Encoding

As mentioned earlier, we focus on PC and PHCC with a uniform encoding transferring a randomized input. However, improved encodings are linked to different advances which shall be discussed in the context of the introduced active warden concept.

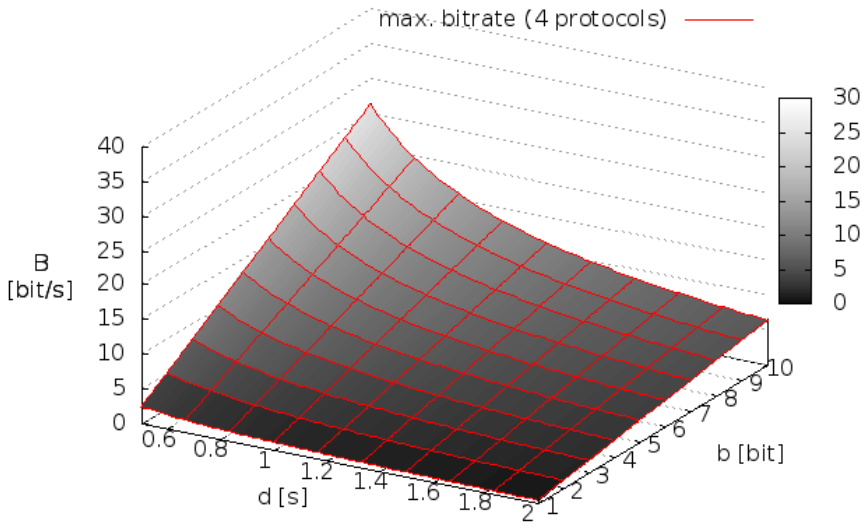

Figure 4.5: A PHCC's bitrate using $n = 4$ protocols, $T = 0.005$s and delays between 0.5s and 2s as well as the capability to transfer between 1 and 10 bits per packet.

4.3.1 Protocol Channels

As the active warden delays packets only if a protocol switch occurs, an optimal encoding would result in as few protocol switches as possible. Alternatively, an encoding could try to only send new packets if a protocol switch occurs and no packet will be sent if no protocol switch occurs. If the delay d is constant, the whole message will be delayed in the same way and would be received without a jumbled packet order. Such a channel would be a combination of a PC and a timing channel since the timing intervals between packets must be measured to calculate the original message. As usual for timing channels, a synchronization between sender and receiver is a mandatory requirement.

For instance, we assume a PC uses a protocol set containing the two elements P_0="0" and P_1="1" and the warden applies the delay d. The message to be transferred to the PC receiver be "00111", i.e., P_0, P_0, P_1, P_1, P_1. If a packet is only transferred on a protocol switch, the sender replaces every repeated occurrence of the same protocol set element with a waiting time t, i.e., the message P_0, P_0, P_1, P_1, P_1 will be transformed to P_0, t, P_1, t, t. The sender sends P_0, waits for time t, sends P_1 and waits for time $2t$. The receiver will receive the whole packet sequence delayed by d.

To signal the end of the message, a new element P_2 in the protocol set can be defined (alternatively, a frame terminating packet sequence can be sent or a constant message length with pause intervals between the messages can be used).

Hybrid PC-timing channels using such an encoding can be countered by introducing a randomized delay d. We therefore developed an improved version of the active warden implementing a randomized d. For each protocol switch, a new delay d is generated (e.g., $d \in [0.1; n]$s) and thus, it is likely that earlier packets overrun later packets if the timing interval t between packets is small. We compare the constant delay and the randomized delay later when we discuss results but will take only PCs, not hybrid PC-timing channels, into account since hybrid PC-timing channels are not in our focus.

As pointed out by Keller, a PC could also use *run length limited* (RLL) encoding that is used for hard disks [MD96] to achieve a higher amount of bits that can be transferred per protocol switch [WK12a]. Such an encoding leads to fewer delays per transferred bit.

Another PC encoding proposed by Keller is to an use unary symbol encoding with $n = 2$ protocols: To send a non-negative integer value $i \in \{0, \ldots, k-1\}$, the sender needs to send $i+1$ packets using P_1 and $k-i$ packets using P_2 [WK12a]. This encoding enables the channel to transfer $b = \log_2(k)/(k+1)$ bits per packet [WK12a]. The PC sender's waiting time between symbols to transfer is required to be high if the delay is randomized and thus not known to the sender since the sender must try to overcome the maximum d value to provide an error-free communication [WK12a]. Thus, the error-free bitrate is at most b/d. If $b < 1$bit/packet, a $d \geq 1$s results in $B < 1$bit/s [WK12a].

As we also pointed out, an optimized Huffman coding can be applied if a suitable symbol distribution (e.g., geometrically) can be used. In such a case, a Huffman coding can minimize the amount of packets required to transfer a given message. However, we assume a cryptographic covert channel input (as usual for covert channel research) and thus, focus on an uniform distribution of the input.

4.3.2 Protocol Hopping Covert Channels With Micro Protocols

Generally, the limitation for PHCCs works in the same way as the limitation of PCs: By delaying protocol switches. However, PHCCs do not need to apply improved codings to overcome the active warden because they can contain micro protocols as they were already discussed in Chapter 2.6.2 and Chapter 3. Such micro protocols can (as in the example of the author's own proof of concept code PHCCT) comprise sequence numbers [Wen08b]; other covert channels such as the one by Ray and Mishra and by Stodle also

comprise micro protocols without being PHCCs, but their sequence numbering technique is the same [RM08a, Stø09]. Sequence numbers allow the receiver-side a re-sorting of covert channel packet content [RM08a, Wen08b]. The jumbled packet order forwarded by the active warden is thus not successful to prevent PHCCs with sequence numbers in micro protocols.

But if a PHCC is *forced* to embed an internal sequence number in the usually small cover protocol space ($sizeof(CP)$), the remaining space available for the actual payload is reduced. Even, if the PHCC already uses a sequence number, the bits utilized for the sequence number might have to be increased to prevent packet order jumbling through the active warden: A higher delay d results in a higher number of delayed packets which can be overrun by other packets. Also, the sequence number can overrun and packets with the same sequence number can thus arrive at the receiver that is unable to distinct those packets. If the size of the sequence number $sizeof(M_{seq})$ in the micro protocol is required to be of a significant size in comparison to the cover protocol space, e.g., $sizeof(M_{seq}) > sizeof(CP)/6$, or if $sizeof(CP) - sizeof(M_{seq})$ is small, the covert channel has to send significantly more packets to transfer the same amount of covert payload.[1] Thus, the active warden can be considered to be of use against PHCCs with micro protocols nevertheless.

4.4 Implementation and Experiment Set-up

To test the proposed concept of a protocol switch delaying active warden, we set up a network between two virtual machines running Ubuntu Linux with Linux 3.0. The first machine acts as covert channel sender while the second machine acts as covert channel receiver. For the virtualization, *VirtualBox* (*www.virtualbox.org*) was used. Both machines were connected via a virtual Ethernet interface and IPv4.

4.4.1 Protocol Channels

The developed proof of concept code monitors the protocol switching behavior of the value specified in the IPv4 "Protocol" field. The Protocol field is easy to evaluate, represents important protocol switches (e.g., between DNS and HTTP), and allowed the us to keep our implementation simple. Besides, additional layers would not have lead to better evaluations since the general concept is layer-independent. The existing

[1] Space-efficiency improvements for micro protocols were already discussed in Chapter 3.5 and will thus not be explained again here.

proof of concept code PCT [Wen09a] (introduced in Chapter 2.4.6) utilized ARP and ICMP. However, to focus on the "Protocol" field, PCT was modified to use UDP and ICMP instead since both protocols are encapsulated in IPv4.

To simulate a cryptographic covert channel payload that generates protocol switches with a probability of approx. $p = (1 - 1/n)$, PCT was also modified to generate its own randomized input using the "rand()" function in Perl.

The active warden was implemented on the virtual machine of the covert channel receiver as well.[2] The active warden requires the *Netfilter/iptables* firewall of Linux. Netfilter/iptables is capable of redirecting kernel-space traffic to the user-space via its so-called "QUEUE" feature.

Berrange developed a program called DELAY-NET [Ber05] capable of utilizing the QUEUE feature via the Perl *IPQueue* module [Mor02]. DELAY-NET was used by Berrange to simulate wide area network behavior with delays. We modified DELAY-NET in a way that it only introduces its delays if a protocol switch occurs.

Additionally, a program was implemented to evaluate the correct order of packets that pass the active warden and that therefore allows to verify the predicted results of Formula 4.2. The program writes the output times of the packets in logfiles. Each PCT bitrate was required to be tested separately with different delays to identify the maximum error-free bitrate of PCT for each delay. A bitrate was considered error-free if 100 packets could be transferred without resulting in an output-error due to the introduced delay.

Figure 4.6 visualized the whole experiment set-up.

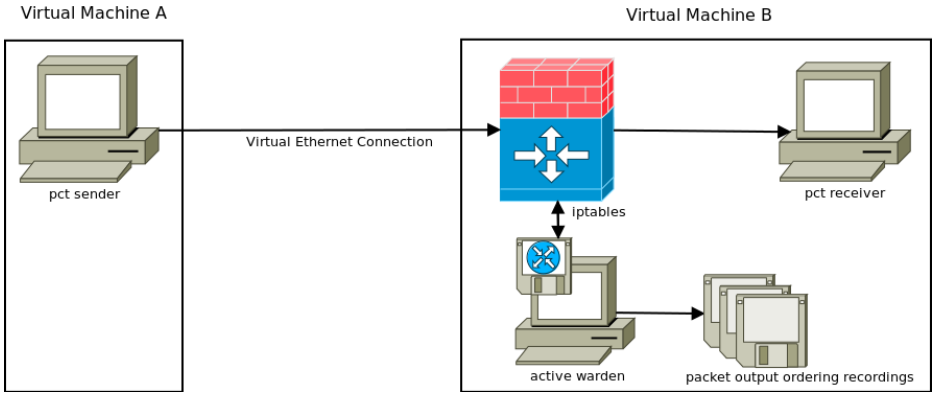

Figure 4.6: Experiment set-up with iptables, PCT and the active warden.

[2]The PCT receiver could also be located on a third virtual machine but that would have introduced additional jitter and would have provided less exact results.

4.4.2 Protocol Hopping Covert Channels

In general, the testing method is valid for both PC and PHCC, since both channels share the protocol switching capability. The difference is that a PHCC does fewer protocol switches per bit (i.e., b/p is higher) and can contain more information per packet (i.e., b is higher) and will, as mentioned earlier, comprise a sequence number that makes the active warden less efficient against PHCCs.

To test PHCCs, we used the proof of concept code PHCCT [Wen07a] (written by the author of this thesis in 2007). The tool was modified at the sender as well as the receiver side to output the elapsed time between the first and the last sent/received packet of a transaction to measure the sender and receiver side time difference between different applied delays in comparison to a direct transmission without a delay. PHCCT utilizes HTTP, plaintext transfer on port 2510, and plaintext transfer on port 20 (actually used for FTP-data transmissions). A *second version* of the active warden was developed to delay application layer protocols used by PHCCT instead of transport layer protocols — therefore the TCP destination port was evaluated.

4.5 Results

To test whether the active warden's application results in the same results as given by Formula 4.2, it was necessary to run the active warden in the previously explained set-up.

Therefore, it was required to determine the local T value. Both previously introduced virtual machines with Linux 3.0 were ran within the same guest operating systems (VirtualBox on a Intel Core 2 Quad CPU Q8400, 2.66Ghz with 4 Gbytes RAM and Ubuntu Linux 12.04, 32 Bit edition, kernel 3.2.0) which results in a very small T value since T becomes slightly higher in real network environments where the distance between sender and receiver is larger and/or slower. By measuring the response-time using *ping* with a high packet preload, an average T value of 0.005s could be determined by dividing the ping response time of 0.01s by 2.

4.5.1 Protocol Channels

The measured protocol switching probability p for a PC using PCT with $n = 2$ protocols in the virtual environment was $p = 0.4738806$. p became slightly higher in a real network environment ($p = 0.53$) because of additional protocol switches (additional ARP packets,

network filesystem packets, and UDP packets (e.g., for DNS) were transmitted and thus, increased p).

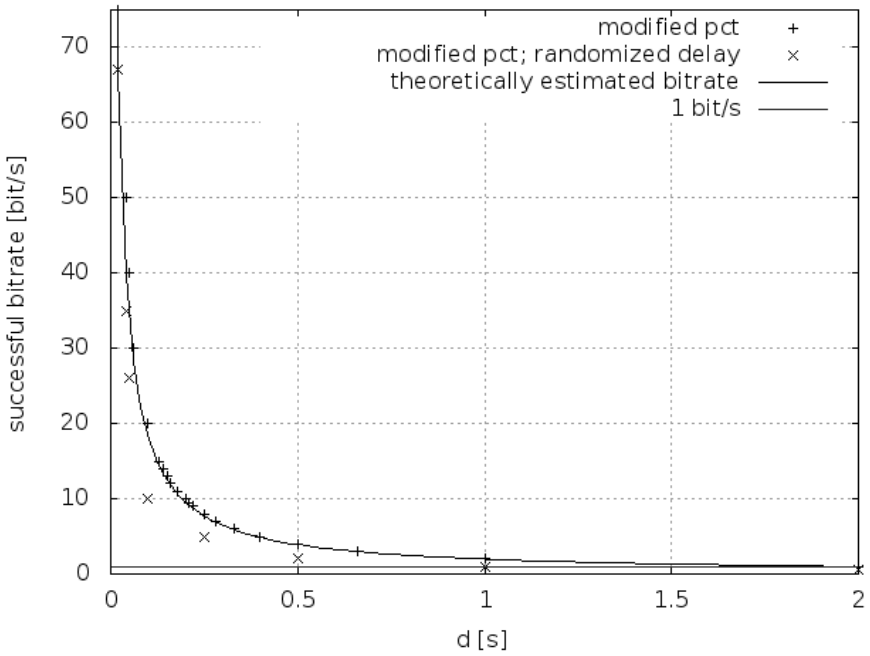

Figure 4.7: Maximum error-free bitrate of PCT dependent on the introduced constant and randomized delay d in comparison to Formula 4.2.

To verify whether an error-free transmission for a given bitrate and a given delay is feasible or not, a PC was established for each measured bitrate b that was tested with different delay values. A PC transfer using a bitrate in conjunction with a delay d was considered error-free if the PC was able to transfer a sequence 100 packets without interruption or jumbling through the active warden. Otherwise, the bitrate was considered too high for the given delay. The results are visualized in Figure 4.7.

The shown results compare the estimated maximum error-free bitrate using Formula 4.2 with the actual measured results. The differences between the calculated bitrate of Formula 4.2 and the measured results with a constant delay are small which motivates the use of the formula.

If we apply a constant delay $d = 2.1$s, we can reduce the maximum error-free bitrate to $1bit/s$. If $d = 1.0$s is applied, the maximum error-free bitrate is reduced to $2.088bit/s$.

Additionally, we evaluated the use of a *randomized delay* d for which the active warden implementation was modified again. The applied randomized delay was uniformly distributed in the range $[0, d[$.

The randomized delay provided better results than the constant delay, i.e., the delay required to achieve the same bitrate limit as with a constant delay was significantly smaller. This improvement is based on the fact that it is more likely that earlier packets overrun later packets if the delay of a packet i is higher than the delay of the packet $i + 1$ plus T. Figure 4.8 visualizes the effect of a randomized delay in comparison of a constant delay. For a delay between $0.02s$ and $2.0s$, the remaining bandwidth B for the randomized delay is between 90% (for $d = 0.02s$) and 50% (most values between $d = 0.1s$ and $d = 1.0s$) of the constant delay.

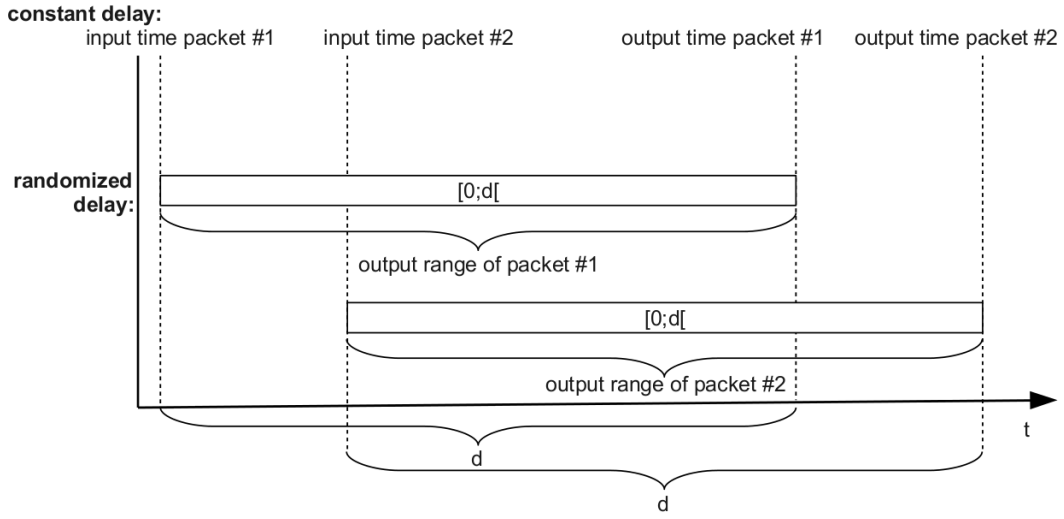

Figure 4.8: Overrun of later delayed packets in case of a randomized delay.

The maximum error-free bitrate of the PC could be reduced to $1bit/s$ if a delay of $d = 1$s was applied. If $d = 2s$ was used, the maximum error-free bitrate was reduced to $0.65bit/s$. Figure 4.7 compares both the constant and the randomized delay.

Another advantage of the randomized delay in the context of a PC is as follows: When a *constant* delay is used, the attacker can aim on re-calculating the original message at the receiver-side. Therefore, error-correcting codes could be used (e.g., by transferring parity bits). However, additional error-correcting information reduces the bitrate of the channel — a result that can also be considered a positive side effect of a present active warden (as well as of network jitter).
An active warden using a randomized delay can reduce this problem since it enforces a smaller error-free bitrate and since the receiver cannot try to determine a a constant d to re-calculate the original message.

4.5.2 Protocol Hopping Covert Channels

The main advantage of a PHCC is its capability to utilize a micro protocol that comprises a sequence number but the active warden can only counter PHCCs without sequence numbers or with very small sequence numbers. As mentioned earlier, a sequence number allows the PHCC to re-sort jumbled packet sequences, i.e., the active warden cannot be considered a valuable means to counter such PHCCs. However, the only available PHCC code PHCCT comprises such sequence numbers and was evaluated nevertheless. Additionally, a modified version of PHCCT was used in which the feature to re-sort jumbled packet sequences was removed.

PHCC without packet re-sorting capability:
PHCCT is capable of transferring $b = 792$ bits per packet – a large value in comparison to PCT. Since PHCCT utilizes 3 underlying protocols, $p = (1 - 1/3) = 0.6666$ in theory — an approximately equal value was measured in practice. The bitrate B of PHCCT was measured as follows: A payload of 100 Kbyte was transferred in an average time of 10.85s, i.e., B is approx. 75.000 bit/s if no active warden is located between sender and receiver. However, since the feature to re-sort jumbled packet sequences was turned off, no error-free data transmission was feasible (even if no active warden was used) if more than 1 Kbyte of payload was transferred. Thus, micro protocols with sequence numbers and a packet sequence re-sorting feature are necessary for a PHCC that uses such a high bitrate.

PHCC with packet re-sorting capability:
On startup, PHCCT establishes peer connections for all three underlying protocols (HTTP, FTP-DATA as well as a plaintext protocol on port 2510). It was observed, that the initial connection setup was delayed by the active warden by d but never resulted in any problems.

We also observed, that the packet output of PHCCT for the payload transfer phase was indeed broken in some tests but that was possible due to a programming problem in the tool itself: If a packet within a TCP connection overruns another packet, the Linux kernel can combine the payload of both packets and hand the combined payload over to the userspace via a system call. PHCCT did not take such situations into account and thus, was not able to locate a second micro protocol header within the same payload received from the kernel. Such problems can either be fixed by searching for multiple occurrences of micro protocol headers within the payload or by using other underlying protocols without a support for combined payloads in the kernelspace buffers (e.g., ICMP). It would alternatively also be thinkable to shrink the TCP buffer watermarks in the kernel.

To verify that the bitrate of PHCCs is indeed not limited by the active warden, the following tests were performed:
A 10 Kbyte plaintext file was transferred using PHCCT. The traffic was not delayed for the first measurement. For the second measurement, a delay $d = 1$s was applied and for the third measurement a delay of $d = 2.5$ s was applied. For each measurement, the 10 Kbyte file was transferred five times. We compared the time at the receiver side's PHCCT that was required to receive all packets. Our results show, that the transfer required indeed slightly more time if a larger d was used, but due to the micro protocol's internal sequence number, all previously jumbled packets were re-sorted again.

To ensure that only the whole message is delayed, a 100 Kbyte plaintext file was transferred as well and the results showed that the overall delay did not increase with the payload size. For each of the previous three situations (no delay, 1s delay, and 2.5s delay), the file was transferred ten times. Since only the time in the PHCCT receiver was measured from the first packet received to the last packet received, a higher delay also results in the situation, that the first one or two packets are received later as usual, e.g., the timer starts with the third packet (the one that was actually received first by PHCCT before the first delayed packets left the active warden). Thus, if the *last* packets of a transaction are *not* delayed but the *first* packets *are* delayed, the overall measured time required for the receiving process can be smaller for a higher d (since the first packets arrive later and the last packets can arrive with a smaller delay).

We can conclude that only whole messages are delayed, i.e., the active warden's effect does not increase with the message size as the comparison of both experimental results revals.

Small Micro Protocol Sequence Numbers:
PHCCs using micro protocols with a small sequence number area (e.g., a 2 bit sequence number as proposed by Ray and Mishra in [RM08a]) can face sequence number overruns on delays if packets are not acknowledged before the next packet is sent (as the active warden monitors unidirectional communications, acknowledgements are not taken into account). Such overruns do only occur if the sending behavior does not foresee a prevention of sequence number overruns and if the introduced delay d is high enough: A constant d must be larger than the sender requires to send enough packets for a complete sequence number range and thus uses a sequence number again. However, since Ray and Mishra use a *stop-and-wait automatic repeat request* (ARQ) protocol that is based on acknowledgements (see Chapter 2.6.2), they prevent sequence number overruns. PHCCs can apply ARQ as well to overcome the problem of sequence number overruns.

Network Environment Learning Phase for PHCCs:
PHCCs utilize a NEL phase to ensure the quality of the utilized cover protocols. Since the active warden is capable of delaying initial protocol switches for connection establishments, it can be considered as useful to delay the NEL phase as well. However, the NEL phase is short and even if it is delayed, it will provide correct results nevertheless as long as delays are smaller than the time that the peers wait for a response message of a cover protocol test.

4.6 Discussion of Practical Aspects

The discussion of the experimental results shall be completed by discussing practical aspects of the active warden. For a practical usefulness, a small delay is of importance to decrease side effects for legitimate users. Even if the introduced delay is small and will only delay a website request, the active warden's acceptance by end users can be low if many website requests are delayed.

In the following, practical aspects will be discussed and several improvements for the active warden will be proposed.

Name service requests: DNS requests occur regularly in TCP/IP networks since they are required to visit websites via domains as well as they are required for other name resolutions. The need for regular DNS requests leads to protocol switches (e.g., DNS→HTTP to visit a website, HTTP→DNS to resolve embedded HTML content from other foreign websites, or DNS→HTTPS to visit secured web content). However, the webserver as well as the DNS server will in almost all cases have different IP addresses. Thus, an easy solution is to only delay network packets for the same destination and not for different destinations — a solution that works as long as no distributed covert channel receiver is present. Therefore, the active warden needs to keep state information about every socket's source (*src*) IP and destination (*dst*) IP and their latest used protocol $p_{src,dst}$. Section 4.8 discusses an approach to whitelist selected communications by using a formal grammar.

Multi-protocol servers: While many enterprises do not run multiple services on the same machine, situations occur, in which multiple services using different network protocols are located on the same server. For instance, a server could run a SMTP server as well as an IMAP server. In such cases, a whitelisting as proposed in Section 4.8 is a solution as well.

Multiple senders: As proposed earlier, the active warden should distinguish different source addresses, i.e., it should keep state for each sending system within the internal network separately. Besides the mentioned DNS problem, multiple senders will in almost all cases produce many protocol switches and thus, should be handled separately in practice — such a technique could only be bypassed by a distributed covert channel sender. Otherwise, almost any packet will be delayed.

However, *network address translation* (NAT) is a technique used *within* enterprise networks. If a subnet is connected to the remaining company network using NAT, and if traffic from this subnet tries to access another network that is connected via the active warden, all systems behind NAT appear as only few or, in case of masquerading, as only one system to the active warden. Thus, the active warden cannot distinguish between these systems behind NAT as it cannot obtain information about the actual sender. The active warden would apply many delays in such a set-up, even if it is not necessary. A possible, but not sufficient, solution can be a white-listing of the NAT systems, i.e., to apply no delay to the whole subnet traffic. In such a case, the active warden would be useless for the subnet.

Using *remote physical device fingerprinting* it is feasible to count the number of systems behind NAT [KBc05]. Using the information about the approximate number of systems behind the NAT could be used to adjust the applied delay in order to reduce d if the amount of NAT'ed systems increases (and vice versa).

Alternatively, the active warden could be directly installed on/integrated in the NAT system to obtain all address information.

Redundancy: Our presented active warden concept is designed to operate as a stand-alone system on an organizational network access point or edge router. Like similar systems (traffic normalizers or firewalls), a problem on such a critical network location can cause problems regarding the network's remote access and regarding connections to other networks. Therefore, fault-tolerant operations can be foreseen in future work. One way to achieve a better fault-tolerance would be to develop redundancy protocols as they are already available for firewall systems. One example is the free *common address redundancy protocol* (CARP) used by the OpenBSD packet filter *pf* [Ope13]. Protocols like CARP should therefore be adapted to the presented active warden if redundancy is required and should synchronize state information (e.g., $p_{src,dst}$ tuples) between the active warden's instances.

End User Acceptance: Taking the previously mentioned ideas to limit end user effects into account, the active warden can be considered useful in practice.

An extensive end user study was not part of our research since technical details and the experimental evaluation were the major aspects. However, request-response times for HTTP traffic (10 Mbyte downloads) were made to compare the HTTP performance with and without the active warden.

The download time without an active warden within the local network was in a range of 0.41 to 0.57s. Afterwards, we set up the active warden with a constant delay $d = 2.1s$ to decrease B to $1bit/s$ and ran a parallel PC using PCT with a bitrate $B = 0.25$bit/s to simulate multiple protocol switches (e.g., DNS requests). The download times were afterwards in a range of 0.4 to 3s. For a constant $d = 1.0s$, the 10 Mbyte download took only $0.3to1.6$s. The protocol switch usually took place in the connection establishment phase since the current protocol was changed to TCP (from the active warden's point of view that, as mentioned earlier, monitored the transport layer protocols).[3] A website should load in $4s$ to ensure its end user acceptance [Aka06], i.e., the active warden would fulfill the $4s$ rule in our test set-up. However, slower connections (e.g., using GSM) that require more time to download a website could be affected by more protocol switches (e.g., due to network background processes) and thus, cannot guarantee the fulfillment of the $4s$ rule.

As the mentioned aspects reveal, the practical usefulness of the active warden is given under the assumption that additional functionality (especially whitelisting) is integrated. Section 4.8 therefore presents a grammar-based whitelisting approach to be used in conjunction with the active warden.

However, the use of NAT is problematic and no good solution except the counting of systems behind NAT or the integration of the active warden into the NAT system could be proposed to overcome NAT problems. Remaining practical problems, if they occur, can be decreased by adjusting the active warden's introduced delay d; but as mentioned in Section 4.1, finding an optimal d results in an optimization problem.

4.7 Improved Covert Channel Techniques to Counter the Active Warden

In Section 4.3, we already discussed several encodings which can help to enable PHCCs and PCs to bypass the active warden. In this section, some additional aspects for the improvement of covert channels will be discussed.

[3] The active warden applies the same limitation to the NEL phase and delays it (cf. Section 4.5.2).

Multiple Receivers: As proposed earlier, the different handling of each connection between a source *src* and a destination *dst* using $p_{src,dst}$ is helpful to prevent delays that occur if a sender accesses multiple other systems. However, it is thinkable that a PC/PHCC sender transfers information to a set of i receivers that together form a distributed receiver instead of sending all covert channel traffic to only one receiver. The active warden would not introduce a delay since the receiver differs for each new protocol in $p_{src,dst}$ if each receiver *dst* is linked to only one protocol. Such a covert channel would either be a PHCC with sequence numbers or a hybrid covert channel as it comprises the idea of a PC as well as the idea of a timing channel (as mentioned earlier in Section 4.3.1). The receivers can afterwards re-construct the received message by combining their fragments if their timestamps are synchronized. If a high jitter is present or if the routing paths between the receivers are linked to bigger differences, such a covert channel with multiple receivers must be considered error-prone for a PC (or must use a low bitrate) but is thinkable to be useful for PHCCs with sequence numbers. It is possible, that an active or a passive warden of any thinkable type is monitoring the receiver-side synchronization as well. In such cases, the raised attention of the inter-receiver communication must be taken into account as well and could be reduced by optimizing the micro protocol for PHCCs as discussed earlier (Chapter 3.4 and 3.5).
Thus, the presented approach to apply formal grammar-based whitelisting (cf. Section 4.8) is a better concept in such a case since it does not allow PCs and PHCCs with multiple receivers per sender.

Multiple Senders: Similar to the approach of using multiple receivers outside of the enterprise network, a covert channel sender can be a distributed system as well. If n possible senders are presented, each of the n senders can be associated with a single network protocol. One coordinator system could send commands to the distributed sender that only send their specific protocol through the active warden. The active warden will therefore not notice any protocol switch for a given sender. However, the active warden still forces the sender to be a distributed system in such a case and the distributed operation must be synchronized and can be observed as well, what is analog to the concept of using multiple receivers. Besides it is thinkable to combine both multiple senders and multiple receivers.

PHCCs with Micro Protocols and Dynamic Routing: By transferring a micro protocol message that requires an acknowledgement, a PHCC sender can measure the response time of packets similar to the ICMP echo request and response message. To detect the presence of an active warden, the PHCC sender can compare the delay of

acknowledgements for packets without causing protocol switches with the delay of packets that cause protocol switches. If acknowledgements of packets that caused protocol switches take more time, it is possible, that an active warden is present and if the PHCC sender is capable of using dynamic routing, as presented in [BWK12], the PHCC sender can try to configure another routing path to the receiver.
This scenario is especially interesting to switch from a PHCC to a PC after a routing path was set up that faces no delay and thus, results in a robust PC connection.
Another aspect is that an active warden, that notices many delays, could report a high amount of protocol switches. By detecting an active warden and using an alternative routing path to a PHCC receiver, a PHCC sender can thus contribute to a low attention raising operation.

4.8 Proposal #1: Applying Formal Grammar to Increase Practical Use

As discussed in the previous section, the active warden's practical use can be improved if a number of issues will be addressed. Especially, it would be of value to overcome the following problems:

1. the delay introduced for protocol switches related to servers that run multiple services (e.g., an e-mail server can provide an SMTP as well as a POP3 service at the same time and the client could simultaneously send and receive e-mails and thus, would generate protocol switches),

2. the general delay of selected legitimate protocol switches (e.g., DNS $\leftrightarrow$ HTTP),

3. the delay of protocol switches of a single sender communicating with different destinations (e.g., accessing the e-mail and web-server at the same time).

To overcome the mentioned problems, we propose a formal grammar-based solution. Formal grammar has already been applied in other areas of IT security: Gorodetski et al. made use of formal grammar for attack modeling [GK02b], and Trinius and Freiling applied context-free grammar to create Spam filters [TF12].

Formal grammar was already introduced in Chapter 3.4.2 and is therefore not explained again.

Building a Whitelist:

We propose to apply formal grammar for a *whitelisting* of allowed protocol switching behavior. We present a sample grammar that can be used to overcome the previously enumerated problems. The active warden must be capable of handling such a grammar and after configuring the grammar, it must be tested by the administrator. A test must ensure that valid protocol switching behavior is not delayed (by transferring test traffic over the active warden). Also, it must be tested that policy-breaking PSCC traffic is delayed. However, such tests depend on the network and a general simulation is not feasible. Thus, the contribution of this section is to propose and discuss a formal grammar-based whitelisting, but not to integrate it into the active warden.

In the first step, the allowed protocols in the network must be defined as terminal symbols p_x where x is the protocol, e.g., $\Sigma_1 = \{p_{dns}, p_{http}, p_{https}, p_{smtp}, p_{imap}\}$. Additionally, the network hosts must be defined that are allowed to communicate with, e.g., $\Sigma_2 = \{s_{mail}, s_{name}, s_{web}\}$. Source addresses are not specified and it is also not defined which system is allowed to connect with which other system since traffic filtering can already be done with usual firewall systems.
Both terminal symbol sets are merged to form the grammar's set of terminal symbols $\Sigma = \Sigma_1 \cup \Sigma_2$.

By using Σ, it is not directly feasible to create the set of productions P: Productions must comprise combinations of servers and protocols in the form <*server*> <*protocol*>, e.g., $s_{mail}p_{smtp}$. Such rules must be understood by the active warden as "it is allowed to communicate with the server s_i by using protocol p_j". Based on this convention, a sample grammar can be defined that allows

1. to use both SMTP and HTTP, after sending DNS requests,
2. to switch from HTTP to HTTPS and from HTTPS to HTTP (some HTTP websites comprise HTTPS content and vice versa),
3. to simultaneously use SMTP and IMAP in order to receive and send e-mails at the same time as supported by most e-mail clients,
4. to prevent the delay of services used on the same machine (also demonstrated by SMTP and IMAP used on the server s_{mail}), and
5. to simultaneously use e-mail, DNS, and web connections, since W_2 allows M_2 and vice versa.

The sample grammar only comprises six production rules:

$$
\begin{aligned}
N &= \{S, D, W_1, W_2, M_1, M_2\} && (4.5)\\
P &= \{S \rightarrow D(W_1|M_1)|W_1|M_1 && (4.6)\\
&\quad D \rightarrow s_{name}p_{dns} && (4.7)\\
&\quad W_1 \rightarrow s_{web}(p_{http}|p_{https})W_2 && (4.8)\\
&\quad W_2 \rightarrow DW_1|W_1|M_2|\epsilon && (4.9)\\
&\quad M_1 \rightarrow s_{mail}(p_{smtp}|p_{imap})M_2 && (4.10)\\
&\quad M_2 \rightarrow DM_1|M_1|W_2|\epsilon\} && (4.11)
\end{aligned}
$$

The Active Warden's Operation:

On arrival of a new packet at the active warden, the active warden tries to match the sender's protocol switching behavior to the whitelist. Therefore, the active warden can either try to start a new production from the starting symbol S or can continue with the previously used rule. For each sender, the last production rule as well as the last n received packet information tuples (server, protocol) must be kept in a cache where n is the maximum length (in symbols) of a sentence producible by the grammar.[4] The whole operation process is shown in Figure 4.9.

Example: We assume that a DNS packet was received by the active warden (the protocol can either be identified based on its destination port, or by applying protocol identification tests like discussed in [BTA+06]). The packet will not be delayed since the productions allow $S \rightarrow D(W_1|M_1)$. If a DNS packet will be received afterwards, it is also not delayed because of $S \rightarrow D \rightarrow M_1$.

Additionally, if a number of packets of the same protocol are received, they are also not delayed since no protocol switch is taking place (a larger number of packets of the same protocol usually occurs for downloads via FTP or HTTP as well as for larger e-mails with attachments). Thus, no whitelisting rule is required to handle such downloads.

To generate whitelists of a lower complexity, it is thinkable to model the grammar in a layer-based manner. In such a case, each layer of the TCP/IP (or OSI) model can be modeled in a separate whitelist. One can also take into account, that only layers that are forwarded to other networks must be taken into account since a data leakage to another network must pass the active warden (e.g., data hidden in Ethernet frame

[4]For recursive productions, n should be limited by an administrator.

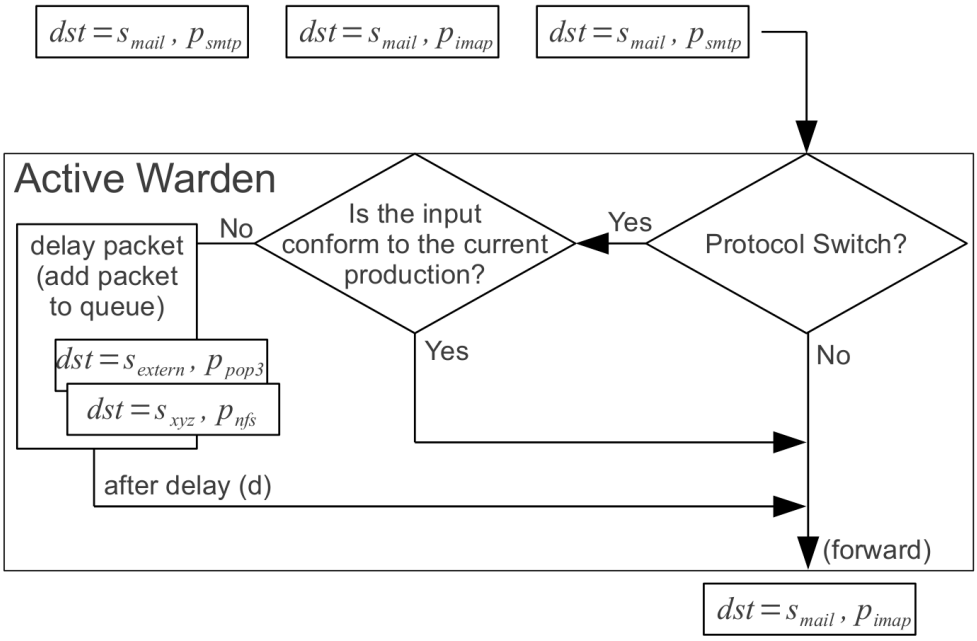

Figure 4.9: Proposal to integrate a formal grammar-based whitelisting into the active warden.

headers would not be forwarded and the same situation applies for ARP messages if the active warden does not explicitly forward them as an ARP proxy).

4.9 Proposal #2: Detection-capable Active Warden to counter PCs

Besides the formal grammar-based whitelisting (or in addition to it), the previously developed detection means for PCs ([WZ12], discussed in Chapter 2.5.4) could be integrated into the active warden. The machine learning-based traffic classification could be used to evaluate traffic from a sender. If the sender's traffic is classified as being covert channel traffic, a delay can be introduced, otherwise, the traffic could be directly forwarded.

Since the training of the C4.5 algorithm took 5,000 packets and since traffic patterns can change over the day, a continuous training and caching of received packets should be done on the fly by the active warden. Also, if not enough packets are recorded, no qualitative traffic filtering can be applied and thus, no decision tree-based delay for the first $n = 5000$ packets should be done.

The problem of first requiring to obtain enough information about already existing connections is called the *cold start problem* for active wardens (traffic normalizers) [HPK01]. Existing traffic normalizers face the cold start problem especially for already established TCP connections for which information about the connection establishment phase and all other earlier packets is not available. Thus, the active warden does not know whether a packet actually belongs to an existing connection or not. Scrubbing packets without knowledge about previous packets can result in negative side effects (e.g., policy-conform packets must be transferred again).
Applying a delay without knowing about earlier packets can thus result in delays of policy-conform protocol switches.

It is thinkable that the active warden keeps a packet queue of the last $n = 5000$ received packets and re-builds the decision tree after a significant amount of new packets got received and also only tries to detect the traffic after 5000 new packets passed the active warden (since traffic classification can become time consuming, it can decrease the active warden's performance and the administrator has to adjust such values).

Because the traffic classification is not perfect (false positives and false negatives will occur [WZ12]), traffic that should be delayed could be directly forwarded in some cases and traffic that should not be delayed, will be delayed in some cases. To limit such negative side effects, the applied delay for a given sender can be increased if the sender's traffic got classified as "PC traffic" multiple times and it can be decreased if the sender's traffic got classified as "*no* PC traffic" multiple times.

4.10 Conclusion

This section presented the first active warden that is capable to decrease the maximum error-free bitrate of protocol channels (PCs) as well as of protocol hopping covert channels (PHCCs). Therefore a new type of active warden for network traffic was developed. The bitrate of these channels was limited by introducing delays on protocol switches issued by a sender. The active warden must be located on an edge of the network to affect a PSCC (e.g., on an edge router of an enterprise network to prevent information leakage).

The active warden was implemented using *netfilter/iptables* in conjunction with a proof of concept code. Evaluations for different bitrates were made for both a constant as well as a randomized delay. The practical usefulness of the approach was discussed in detail and two improvements (the combination with existing detection means for PCs

as well as a formal grammar-based whitelisting) were proposed to prevent the delay of legitimate traffic.

Future work will focus on the problem of senders behind masquerading systems and behind NAT that appear as a single sender to the active warden and in the reduction of delays applied to policy-conform traffic. Also, the development of redundancy protocols for synchronizing states between different instances of the active warden shall be part of further developments.

While PC traffic can be limited in an efficient way, it is challenging to limit PHCCs with internal micro protocols since such micro protocols can contain sequence numbers which allow the receiver-side re-sorting of jumbled packet sequences. However, the active warden forces a PHCC to integrate such a sequence number and thus, less space is available for payload within a PHCC's cover protocol. As mentioned earlier, a sequence number is practically mandatory to ensure a reliable data transfer via PHCCs anyway. Besides, the active warden can delay a PHCC's NEL phase but cannot break the NEL phase's results.

However, it is not expected that the delay-based limitation can be modified to effectively limit PHCCs with sequence numbers within their micro protocols. Therefore, a new method for limiting PHCCs with sequence numbers must be found. A very first approach to counter such PHCCs was discussed by the author in his diploma thesis and aims on detecting increasing sequence numbers of micro protocols [Wen09b], however, that approach needs to be developed further in the future. Besides, various techniques are available to hide data in network protocols — a detection of all possible network covert channels that a PHCC could use, seems impossible. Therefore, another general approach that does not focus on a specific hiding technique must be found to counter PHCCs with micro protocols.

5 Storage Channel Prevention in Building Automation Systems

Chapter 2 already introduced the topic of building automation systems (BAS) and its security aspects. This chapter originates a link between BAS and covert channels as it will discuss covert storage channels in BAS. Additionally, *side* storage channels in BAS will be considered in this chapter to take the aspect of unintentional communication into account as well. The presented means of this chapter should be seen as additions to the existing security means used in the area of BAS since their value is not eliminated by the application of anti-covert channel means. Thus, features such as physical access control (PAC) and BAS traffic encryption are important aspects for BAS security nevertheless.

5.1 Adversary Scenario for Side/Covert Channels

The general adversary scenario for covert channels was already introduced in Chapter 2.2.1 and BAS-specific adversary scenarios were discussed in Chapter 2.7.3. However, covert and side channels in BAS represent a specific use case and thus we will discuss some specific adversary aspects in this section:

5.1.1 Side Channel Adversary Scenario

A side channel can be used by an observer to monitor events taking place within a building. Not only can event monitoring be used to monitor *persons* (subjects) within a building (e.g., employees or inhabitants and *when* a person is *where* for *how long*) but it can also be used to monitor the *building configuration* itself. Therefore, various sample scenarios are thinkable:

1. **External Intruder:** A thief could monitor the lighting configuration of a building remotely to learn when the light in a specific floor or in the basement is automat-

ically turned off. Alternatively, the thief could monitor the presence sensors of a building to increase his chances for a successful break-in.

2. **Malicious Employee:** Similarly to the *external intruder*, a malicious employee could be willing to steal a document from the manager's office: The employee could use BAS side channels to obtain information about the presence of the manager in his office and could steal the document more safely if the manager is currently out of his room.

3. **Selling Health Data:** As explained in Chapter 2.7, buildings with AAL technology comprise sensors for sensitive information such as blood preasure. Such information could be leaked via side channels and, for instance, be sold by a malicious physician or nurse. Even if no AAL technology is installed, health information could be leaked nevertheless (e.g., changes in the heating configuration/room temperature can be a sign of illness).

4. **Business Rival:** Another problem occurs when multiple competing companies, organizations or organizational units are located in the same building that comprises a single BAS. For instance, one company could use side channels to obtain information about working hours in labs of the other company or even on specific research and development activity by, for instance, monitoring an airflow laboratory's power consumption.

After information is leaked through a side channel and if the data comprises private information of a subject, the monitored subject has no control over the data, i.e., cannot control or determine whether an adversary accessed the data, sold the data, or just stored the data for further purposes. A similar problem was mentioned by Enck et al. regarding the privacy of smartphone apps as data provided to apps can afterwards only be controlled in a limited way by the user (e.g., location information provided to an app can be sold to a third party afterwards) [EGC+10].

5.1.2 Covert Channel Adversary Scenario

In comparison to a side channel, a covert channel comprises an intentional sender. Thus, a covert channel can be used to signal confidential data through the BAS as well as to signal confidential data through a BAS into another network/the Internet. Selected sample scenarios will explain the use of covert channels in this context:

1. **Bypassing Enterprise Network Protection:** A member of an organization is willing to leak confidential information to the public but mobile storage devices and the TCP/IP traffic inside the office network are observed and data exfiltration protection is applied. The person could thus use a covert channel through the BAS network to leak the information via the usually non-monitored Internet connection of the BAS. A thinkable scenario would be an employee willing to (permanently) leak business data or a double agent in the building of agency A willing to (permanently) leak secret data to agency B.

2. **Inhouse Data Leakage:** To illustrate the scenario of an inhouse data leakage, we use the example of the **papal conclave**, however, this scenario can be transferred to any organization.
   We assume that the papal conclave (i.e., the meeting that elects the pope in Rome) is taking place in a closed room and no communication is allowed with persons outside of the room until the election's result is about to be released to the public. We additionally assume the existence of a BAS that connects the election room with other rooms of the building. One person within the room (the covert channel sender) is willing to signal the conceivable result of the election to a covert receiver that is informing a journalist before the final decision is made. If there are two light switches in the election room to turn on the light, both switches could be configured to turn on the lighting in other rooms as well if the BAS is configured to do so. Thus, by turning on one of the two light switches, the covert receiver can signal one bit to the covert receiver. Alternatively, both light switches could represent two bits or a timing channel could be established to transfer more bits.

3. **Building Automation Botnet:** Another thinkable example is the development of botnet software for BAS. As various BAS components are connected to the Internet and since these systems are sometimes based on unpatched Linux or Windows systems, it would be possible to install bot software on these devices. Such a bot software would enable the remote control and monitoring of the building. In such a case, the covert channel would be embedded in the communication of interconnected buildings (e.g., one building controls and monitors other buildings) or it would be a stealthy Internet-based command and control channel between the control system (*botmaster*) and the bot.

In general, different devices can be used to signal covert information to other devices. For instance, configuring the heating can be used to signal data to a temperature sensor-

based receiver. Besides boolean values (e.g. turning on/off the heating), floating values are also possible to be used and result in more bits that can be transferred per action. For instance, the covert channel sender in the election room could set the heating level to a value $v_{heating}$. If n election results are possible, n different values (e.g., n heating levels) are enough to signal the covert channel receiver the conceivable result.

5.1.3 Additional Aspects for an Adversary Scenario

A side channel-attack can – as other BAS attacks too [Hol03] – come from an external subject (e.g., thief, spy, or detective) as well as from an internal subject (e.g., other employees, or inhabitants). On the other hand, an *internal* as well as *external* subject cannot only be the receiver but also the subject that unintentionally leaks the information. For instance, external subjects can act as visitors or can trigger movement sensors outside the building.

Due to external BAS access (e.g., using web-interfaces), a covert channel *sender* can (similar to the side channel sender) be an internal or an external subject as well – dependent on the use case. The covert channel *receiver* can also be an internal or an external subject.

Besides the subject-based distinction (external/internal subject), another aspect lies in the technical BAS access that can be remote (using an application or a web-interface) or direct (physical). Under both technical access types, a side and covert channel communication can be realized. A *direct* access in that case means to be able to physically interact with building automation devices (e.g., read a sensor's value on a display or click a button on an actuator device). *Remote* access means *non-direct* (non-physical) access to the building automation components (e.g., web-based monitoring access, network protocol access using a sniffer, or access via a device capable of generating own protocol messages to trigger actuators).
An non-direct access to the BAS would also be possible if two buildings are connected via the Internet or an organizational office network in order to exchange BAS data via a tunnel. The BACnet protocol therefore uses BACnet/IP (BACnet encapsulated in UDP). Such a tunnel connection between two buildings could be exploited for side channels by eavesdroping of BAS packets as well.

5.2 Definition in the BAS Context

As explained in the previous chapters (1.1 and 2.3), we refer to a *side* channel as a *covert* channel without an intentional sender.

As buildings are still not protected with the same means and quality as office or backbone networks in a company, the utilization of a BAS can be considered attractive for a steganographic communication if protection means of an organizational network have to be bypassed.

Regarding to the BLP model introduced in Chapter 2.1, a covert channel exists when a write-down or a read-up is taking place. Thus, a BAS user s must either read information from a higher leveled object o_1 (i.e., subject s has read access to object o_1 although o_1 *dom* s) or must write information to a lower leveled object o_2 (i.e., s has write access to o_2 although s *dom* o_2) to create a covert channel. If the read/write process is not intentional, a side channel is present.

To create covert/side channels in a BAS, we focus on the elements of the field level (sensors, actuators) that are accessible via the automation level. Additionally, the sensors as well as the actuators are put in context of the BLP model's security levels and categories and thus, must be seen as objects.

For instance, the object *light-switch-11* could be located in the manager's office and thus, could be linked to *(level-A, {management})* and the object *temp-sensor-12* could be located in the room of the research and development team and could be linked to *(level-B, {r&d})*.

Using this categorization, an organizational chart can be used to split a BAS' devices into security levels and categories as exemplified in Figure 5.1.

We come back to the previously discussed example of an employee willing to steal a document from the manager's office: If employee Eve can find and use a channel to obtain information about the presence of the *Head of R&D* in his/her office, a covert or side channel is present since *(D, {project-x}) dom (B, {r&d, project-x})* is not true. In other words, if Eve could read the requested BAS sensor data, a write-down would take place (if Eve eavesdrops the information) or a read-up would take place (if Eve requests the information and can receive it afterwards).

5.2.1 High- and Low-Level Channels

In this chapter, we differ between high-level covert and side channels and low-level covert and side channels. While low-level channels, as typical for TCP/IP network

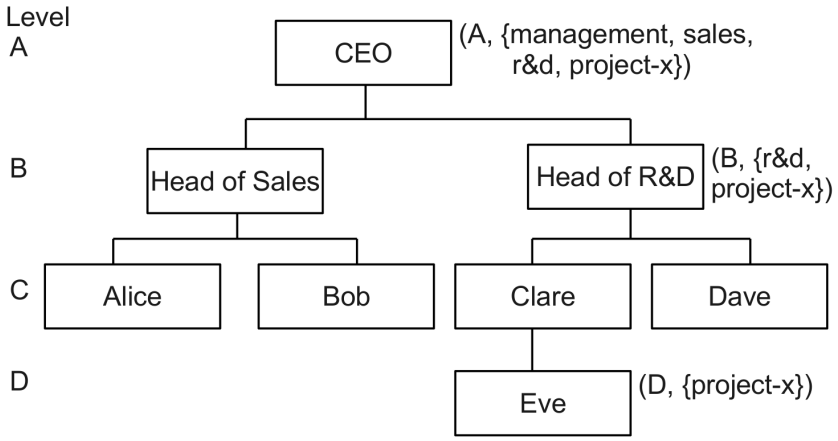

Figure 5.1: A sample organizational chart with security levels and categories.

protocol-based covert channels, utilize reserved or unused bits in BAS network protocol headers, high-level channels do not directly utilize low-level communication but are based on the interaction with the BAS instead (e.g., directly opening a window or reading a temperature sensor value using a web-interface). In other words, high-level covert/side channels abstract from the low-level network protocols and thus do not depend on a specific BAS technology or protocol suite. This applied distinction is similar to the distinction used in [ELP+12] where the authors differ between low-level sensor information in a wireless sensor network and high-level social activity events in the context of pervasive computing.

5.2.2 Requirement of Additional Protection Means

An overview on the related security achievements for BAS was already given in Chapter 2.7.3 but it is important to mention that the protection means in this chapter must be seen as additions. For instance, while write-downs and read-ups shall be prevented, write-ups are still possible in the BLP model and thus, an employee could send a command to a higher leveled manager's office window without violating the security policy. Such access permissions must be (and can already be) handled by the existing protection means.

5.3 A Building-aware Active Warden

Using typical solutions (especially for private homes), BAS users have access to (a part of the) building's devices. For instance, the HomeMatic smart phone interface as well as the HomeMatic web interface can be used to remotely control a home (Figure 5.2). These systems do not provide multi-level secure environments and cannot prevent covert or side channels in the BAS. We decided to introduce an active warden into the BAS to sandbox BAS applications in a way that side channels and covert channels are prevented.

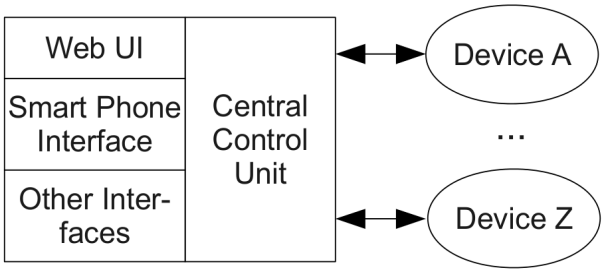

Figure 5.2: Interaction with the HomeMatic BAS. The HomeMatic provides different interfaces, a central control unit (CCU), as well as sensors and actuators (cf. Chapter 2.7.1).

In Chapter 2.5.2 a special variant of an active warden called the *network-aware* active warden was discussed. Such a network-aware active warden has knowledge about the network in which it operates. In this chapter, we present the concept and implementation of a *building-aware active warden*. Instead of being aware of the TCP/IP network, the building-aware active warden is aware of the BAS environment. A building-aware active warden is capable of dropping and modifying API requests regarding to a MLS policy in order to prevent side storage channels and covert storage channels in the BAS. In particular, the building-aware active warden has knowledge about

1. the BAS' users and their security levels/categories,

2. the BAS' devices (sensors and actuators) and their security levels/categories, as well as

3. the mapping of an organizational hierarchy to the BAS (reflected by 1. and 2.).

5.3.1 Implementation

Our implementation of an active warden is designed to counter high-level covert storage channels and high-level side storage channels (we will focus on low-level channels later for the special case of BACnet). The active warden itself is a middleware that is enforcing mandatory access control using the BLP model, and that provides an API to applications. The middleware can solely prevent side and covert channels for applications which use the middleware (and that have no other BAS access besides using the middleware). Software that has protocol level BAS access (i.e., does not use the middleware) cannot be secured and thus, can create and utilize covert and side channels. Direct low-level events are also not protectable using the middleware (e.g., directly sending protocol level covert channels).

However, the middleware additionally enforces that low-level covert and side channel network messages are secured in a way that no data is forwarded to the application layer if denied by the policy (e.g., an application of a LOW user will not be provided with HIGH-level events of the BAS even if the middleware notices and records these events on the BAS network layer). Instead, all events are stored in the local event database and are only provided to applications run by users with the necessary permissions.

Figure 5.3 visualizes the concept of our middleware-based building-aware active warden implementation: Different applications can utilize the middleware's API while the middleware itself enforces a security policy and keeps all present and historic BAS events in a local database as well as the middleware interacts with the actual BAS hardware. Besides of historic BAS events, the database does also contain security information (subjects and objects as well as associated security levels and categories).

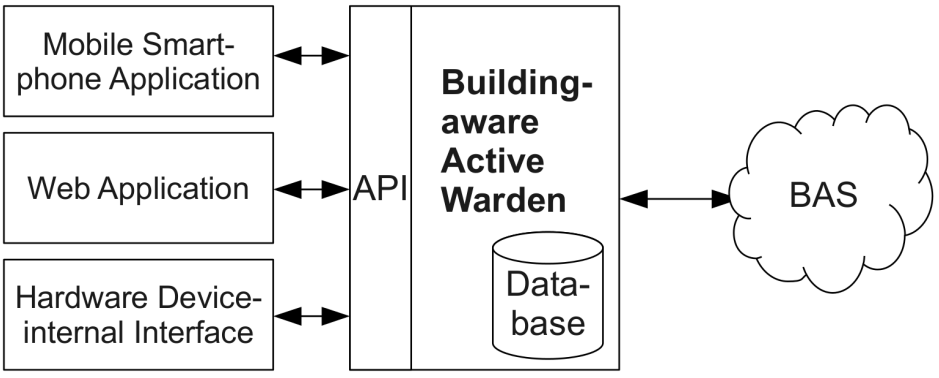

Figure 5.3: The architecture of the building-aware active warden.

Existing middleware solutions for BAS are already capable of enforcing role-based access control (RBAC) [MRH+08], however, since covert channels are defined in the context of multi-level security, we focus on the BLP model.

A first prototype, the *Home Analytical System Interface* (HASI) [RWM+11], of a middleware that uses both the HomeMatic BAS as well as the CurrentCost energy monitoring system, was developed by a project group at the University of Applied Sciences in Augsburg. The development of the approach was co-lead by the author of this thesis but the student project did not comprise MLS features. Security enhancements lead to the integration of basic security features as well as a very simple MLS support by a second project group that was lead by the author at the University of Applied Sciences in Augsburg as well and that comprised support for BACnet environments. However, the author developed a third middleware (without real hardware support, since only stubs were implemented) to integrate the full BLP model including security categories and support for BLP-conform historic event recordings. In the remainder, we speak about this third middleware.

In the prototype, all configuration is entered by hand in the active warden's configuration database that is based on MySQL. However, it is thinkable to link such information to organizational databases in order to automatically adapt the BAS to the organization's settings and organizational changes.
The database includes the users (with security levels), the security categories, the devices (with security levels), the mapping between users and security categories, between devices and security categories, historic event recordings (with security levels), the mapping between historic event recordings and security categories, and additional rules (min/max values and timing considerations, e.g., preventing excessive heating on weekends in an office building). Applications using the middleware must poll device values to receive current sensor values and actuator states. If a device's value is requested and access to the information is granted, the middleware returns the last known value of the sensor or the last known state of the actuator. If the middlware receives a new value from the hardware layer, the value is automatically stored in the list of historic values and assigned to the device's current security level and categories. The importance of this feature will be explained in the context of tranquility (cf. Section 5.3.2).

5.3.2 Tranquility

After MLS is applied to a BAS, situations can occur in which it is necessary to change the security level for a given device (e.g., a room that was previously used by a manager

is now used by trainees). The term *tranquility* means *that subjects and objects may not change their security levels once they have been instantiated* [Bis03]. Besides, the principle of *weak tranquility* exists, which allows the change of security levels as long as it does not violate the security policy [Bis03]. Therefore a trusted party is required that applies security level changes [Bis03].[1]

Bishop provides an example for weak tranquility in which only a trusted administrator is allowed to change security levels of objects [Bis03]. The same idea can be adopted to building automation environments: In our middleware, a trusted building administrator is necessary to change security levels for objects (devices) in the database as well as he is able to add new devices and link them to a security level and to remove devices from the BAS.

Ensuring that no confidential information is leaked cannot be considered trivial in the BAS context: While raising the security level of objects can cause access problems but no security problems, a downgrading is linked to security problems since data that was previously not available for a lower leveled subject is now available to such a subject [Bis03]. For instance, if a room X was used by subject s_1 (HIGH, {management,sales}) and is now used by subject s_2 (LOW, {sales}), it is safe to allow other subjects of (LOW, {sales}) the access to *current* sensor values in the room X. However, a BAS can – as in case of our middleware – also comprise historic information and thus, must ensure that the historic sensor recordings of room X for the time at which the devices within the room were assigned to (HIGH, {management,sales}) are still only accessible by subjects dominating (HIGH, {management,sales}). Otherwise, behavioral information about s_1 would be leaked to s_2.

Therefore, our middleware keeps security level states for historic information. Even if a device is downgraded to a lower security level, the historic information of the device are still linked to their original security levels.

5.3.3 Value Types and Rule Environment

Due to the different device types (e.g., temperature sensors, light switches), different data types must be transferred between the middleware applications and the BAS. We implemented floating point and boolean values. Floating point values are used to communicate with devices that use non-boolean values (e.g., setting heating level 0.4 (40%) or receiving a temperature of 21.2°C). Boolean values are used for on/off switches and

[1] If no changes are possible, no trusted party is required and the principle is called *strong tranquility* [Bis03].

devices with only two states (e.g., a window sensor that can switch between the two window states "open" and "closed").

Besides, values can only be modified (normalized) in a way that is conform to the configured rules of the active warden:

1. **Value-based Modifications:** It is possible to define minimum and maximum values for specific devices (e.g., a level D user can only set the heating level to 80% but not to 100% and thus, the command to set the heating level to 100% is modified to become a command to set the heating level to 80%). On the other hand, temporal values are considered. For instance, the heating cannot be used on weekends within an office building as long as no administrative configuration change is applied in the database.
   However, these value-based modifications implement already known features. The interesting aspect for this thesis is the following (2.).

2. **Request/Command Preventions:** Due to the BLP model enforcement, the active warden prevents requests and commands of subjects for which the required object access is not given (i.e., if a NRU or NWD rule violation would take place).

5.3.4 Shared Rooms and Devices

If a device or a whole room with its devices is shared by subjects of different security levels, a problem arises that is visualized in Figure 5.4: If the devices in a shared room (e.g., a meeting room, a conference room, or an elevator) are linked to a high security level, the configuration results in access problems for lower leveled subjects since the active warden prevents read-up requests, e.g., no lower leveled subject can request the room temperature.
If the devices in the room are linked to a low security level, commands sent to actuators would result in a write-down that is prevented by the active warden, i.e., high-leveled users would not be capable of controlling the room.

We propose to solve the conflict in one of the following ways:

1. **Exclusive Booking:** The *objects* (devices) in the shared room could be temporarily downgraded if it is necessary for low-level subjects to access the room. For instance, a low-leveled user could book a conference room. The trusted administrator is capable of achieving this task in the context of weak tranquility.

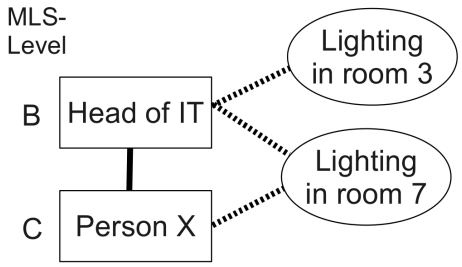

Figure 5.4: Devices shared by subjects of different levels (e.g., the lighting in a meeting room).

Thus, lower leveled subjects will only have access permission to the room, if required, otherwise, high-leveled subjects will have access to the room, but never both. Since the middleware keeps security state information of all historic data of objects, a temporary downgrade would not result in data leaks.

2. **High-level Downgrade:** The high-level *subjects* are temporarily assigned to a lower security level (the level of the objects in the room) than their actual clearance in order to provide them access to the devices in the shared room. However, the result would be a write-down of information if a low-leveled subject notices the presence of high-level subjects in the room and thus could conclude that these subjects are currently not in their high-level'ed rooms – this situation leads back to the scenario of an employee willing to steal a document from the manager's office. Thus, a timing side channel would be present.

3. **Invisibility:** For the time slice in which the room is used by high-level subjects, the room's devices could be made *invisible* to all lower leveled subjects. However, this solution would result in the same side effect as proposal 2 since invisibility implies the presence of subjects with a high clearance in the room.

Depending on a given room, one has to select the most suitable solution. For instance, the exclusive booking solution is useful for conference rooms, while a kitchen room in an office building will be better operated with the high-level downgrade solution.

To realize such a downgrading or such invisibility settings, the administrator would need to change the configuration of security levels by hand in the database. However, it is thinkable to integrate comfortable user interfaces for such purposes or to provide scripts to speed up such processes.

Besides the mentioned timing side channel, proposals 2 and 3 would also allow to create a covert timing channel since a low-level subject is capable of polling the invisibility state/occupation times of the room: A high-level sender could signal hidden information by altering the visibility of the room's devices as well as by altering the room's state between occupied/not occupied.

5.3.5 Emergency Situations

If an emergency situation occurs, it is not always protective to enforce the BLP model.[2] For instance, the only way to escape a fire might be a way through the manager's office and thus, should be available for everybody's safety in an emergency situation. Current BAS already support emergency features. For instance, fire alarm systems are usually even implemented in a dedicated network with only a single connection to the BAS [SR09] and besides, face regular testings. Another example for ensuring safety is the commando prioritization in BACnet with the two highest priorities *manual-life safety* and *automatic-life safety*.

Due to the high priority of emergency systems, their aspects should be integrated in the BAS as usual (i.e., independent from the middleware). The middleware should thus only display emergency information but physical accessible devices (e.g., emergency buttons) which do not depend on possibly error-prone middleware-based applications can be considered a better choice to ensure safety.

5.3.6 Results

In the following, the results of the presented active warden will be discussed.

Requirement of high-level access: To realize protection, the building-aware active warden requires a high-level access using the provided API. If an application can access the BAS at the protocol level or if a user can directly interact with the BAS, the middleware can be bypassed as the application is not sandboxed by the middleware. Thus, older software that was not programmed to utilize the middleware creates a **legacy software problem**. As long as middleware-bypassing software is present, covert and side channels will be possible in the BAS. Therefore, read/write requests can be sent to the BAS, and **low-level covert channels** and **low-level side channels** can be established/utilized – an aspect that will be discussed in the following section. Additionally, if a physical interaction with the building is taking place, a covert channel/side channel

[2]The author would like to thank Jörg Keller for pointing out this safety aspect.

can be established nevertheless. Therefore, a high-leveled user must alter a state of a low device by hand (e.g., a manager turns on the heating in an employee's room by hand). However, such interactions would be conspicuous and unnecessary since a manager could – in such a case – also directly talk to an employee.

Prevention of Read-Ups and Write-Downs: Due to the enforcement of the BLP model, the middleware provides its applications a sandbox in which read-ups and write-downs are prevented. In practice, configuration errors (e.g., a user of a lower level is associated with a higher level due to a side effect of an error-prone SQL command) can lead to policy violations nevertheless.

Coming back to the papal conclave example, the election room's objects and the electors could be associated with the highest security level that is only valid and assigned to the attendees of the election for the time of the election, e.g. *(top-secret,* {*election*}*)*. If an elector would try to turn on the lighting in another room with a lower security level, the write-down would be prevented if commanded by a sandboxed application.

However, consider a practical situation with a CEO willing to control all devices in his own company where the control of lower-leveled devices is only feasible through a write-down. We can conclude that each of these write-downs would break the BLP model. The weak tranquility helps to allow such downgrades if a trusted building administrator allows selected write-downs. Nevertheless, using a trusted subject is a slow process and the practical acceptance of this approach could be low.

Additional Protection Means: The middleware solution can prevent covert and side storage channels but since various other security aspects must be considered for BAS, additional means such as physical access control (PAC), network/application layer data unit encryption, and secure storage of eHealth information for BAS in AAL environments must be introduced (these means were discussed in Chapter 2.7.3).

Covert Timing Channels: The middleware counters covert/side storage channels but not timing channels. Timing channels in a building with rooms shared by multiple levels are feasible if a room is made "invisible" (as discussed earlier) to lower level subjects if utilized by a higher leveled subject: A covert channel sender can assign a hidden information to the invisibility as well as to the visibility to signal a hidden message as a combination of (in)visibilities. Therefore, the covert channel receiver needs to poll the room's status after each timing interval t. As usual for timing channels, sender and receiver must synchronize a priori and must agree on a time interval t used between the bits of the hidden message, i.e., between the alternations of the visibility.

Means to counter covert timing channels can probably be applied in building automation as well (cf. Chapter 2.5 for details). If fuzzy timing as proposed by Hu ([Hu91], discussed in Chapter 2.5.4) is applied in request/command messages via the active warden, a timing channel could be made less efficient but cannot be prevented. However, since timing events in buildings (e.g., opening or closing a window) are already slow, a timing channel will not provide a high bandwidth. A detection of timing channels based on statistical monitoring of API call behavior or by using a decision tree is thinkable as well. Therefore, the discussed approaches of Zander et al. [ZAB07a], Cabuk et al. [CBS09] as well as Berk et al. [BGC05] could probably be adopted to BAS environments. We discuss additional means to counter covert channels in BAS in Section 5.5.

Prevention of Energy Consumption-based Side and Covert Channels: Covert channels can be established using another approach based on the energy consumption of devices. If a hidden message is to be transferred, a covert channel sender can activate different power consuming electrical devices in the building to signal a receiver a hidden message based on the sender's energy consumption profile. Therefore, already discovered side channels based on smart meters [GGJL12] can be exploited or can become a covert channel if used for a message transfer with an intentional sender. However, such channels are not directly linked to the BAS and if the BAS comprises energy consumption information nevertheless, it can apply protection means for this data in order to enforce mandatory access control. For instance, the CurrentCost energy monitoring system can monitor the energy consumption of selected devices. As done in [RWM+11], these device-specific consumption values can be stored in a building automation middleware database and thus, can also be linked to security levels as well to prevent covert and side channels based on the energy consumption.

We can conclude that although our approach faces limitations, most application-based covert and side channels can be prevented if the building-aware active warden is applied.

5.4 Low-level Covert Channel Prevention in BACnet

In the previous section, we discussed a means to counter high-level covert and side storage channels in building automation environments. This section takes the technology-specific low-level covert channels into account. Therefore, we first introduce sample covert channels in BACnet and afterwards present a prevention technique.

5.4.1 Covert Channel Set-Up in BACnet

This section does not cover *all* potential covert channels in BACnet. As a four layered protocol stack that can comprise various network protocols, BACnet naturally provides room for many covert channels and finding covert channels in such a protocol stack can be considered trivial. Therefore, we only introduce sample covert channels in BACnet and use these channels for the discussion of the prevention technique presented afterwards.

Low-level covert channels in BAS are similar to network covert channels in TCP/IP since they place hidden data in unused fields of network packets, in network traffic behavior, or in the timings of network packets. The presented covert channels require two BACnet devices of which one is acting as a covert channel sender while the other device is acting as a covert channel receiver. We assume that covert channel sender and covert channel receiver did – as usual for covert channel communication – agree on a common coding in advance. As also usual for a covert channel communication, high data transmission rates and all other caused anomalies can raise attention and can thus lead to a detection of the covert channel.

In the remainder, we assume that a covert channel sender can create BACnet frames by using manipulated BACnet devices or by using devices attached to the BACnet environment by the adversary himself. Similarly, the receiver of the covert or side channel needs a device to read frames within the BACnet network. Alternatively, sender or receiver (or both) can access BACnet tunnels over TCP/IP, i.e., they need no hardware BACnet device in that case.

Protocol Channels in BACnet

BACnet messages comprise a "message type" in messages of both the network as well as the application layer. One might alter the value in the message type on both layers to signal hidden messages and therefore create two different protocol channels. The first protocol channel utilizes the network layer message type field and alters its value between the two message types "Who-Is-Router-To-Network" and "I-Am-Router-To-Network". "Who-Is-Router-To-Network" is sent to discover the correct router that must be used to send traffic to a given network. The information that a device is a router to a given network is announced via the "I-Am-Router-To-Network" message.

For the application layer, we selected two message types as well: "Who-Has" and "Who-Is". The "Who-Has" message type requests the object name, device ID and object ID of BACnet objects while "Who-Is" requests information about the address

and device identifiers of devices in the network via broadcast.

Our covert channel assigns each message type with a value ("Who-Has"=0 and "Who-Is"=1 for the application layer as well as "Who-Is-Router-To-Network"=0 and "I-Am-Router-To-Network"=1 for the network layer) to signal a covert message using the same way as with a *protocol channel*. In both protocol channel cases, 1 bit can be transferred per packet but additional message types can be taken into account as well to increase the number of transferable bits per packet.

For the implementation of the covert channel, the *BACnet Protocol Stack*[3] – an open source user-space implementation of BACnet/IP – was used. To send the hidden messages, a Perl script was programmed that calls the protocol stack's demo tools. Each tool is responsible for a different BACnet message type: *bacwi* sends Who-Is, *bacwh* sends Who-Has, *baciamr* sends I-Am-Router-To-Network, and *bacwir* sends Who-Is-Router-To-Network. The tools were modified in a way that they do not wait for a response message (e.g., for a response to the Who-Is-Router-To-Network message).
Wireshark[4] was used to monitor the transferred hidden messages by hand and thus, no receiver was needed since our focus was not on the perfect message transfer (and therefore, does not include error correcting/detecting codes) but on the demonstration and prevention of these covert channels.[5] The whole message transfer was simulated using the localhost (*lo*) and the Ethernet (*eth*) network interfaces and BACnet/IP.

Covert Storage Channels in BACnet

The previously discussed protocol channels utilize a storage area and thus can be considered as storage channels as well. However, additional covert storage channels in BACnet are feasible. Therefore, unused header areas can be utilized as cover protocols. Determining utilizable areas in network protocol headers is accomplished in the same way for BACnet as it was done for traditional TCP/IP networks and is assumed to be a straightforward task. For instance, the network number specified in the "Who-Is-Router-To-Network" message type can be used to signal hidden information by requesting the information about the router for the given network ID. For a storage channel, we do not depend on the number n of utilized protocols of a protocol channel to transfer $\log_2 n$ bits per packet, but instead on the amount of cover protocol space $sizeof(CP)$ and thus can transfer $sizeof(CP)$ bits per packet (cf. Chapter 3.1).

[3] *http://bacnet.sourceforge.net/*
[4] *http://www.wireshark.org/*
[5] As mentioned earlier, the placement of covert channels in network protocols is a straightforward task.

Covert Timing Channels in BACnet

The previously discussed protocol channel can be used to create a simple covert timing channel. Therefore, only one message type is required to be transferred. Instead of representing the hidden message through message type values, a timing channel introduces inter protocol gaps (see Chapter 2.4.5) to encode a hidden message in timing intervals. The covert channel receiver needs to observe the inter protocol gaps of the sender to retrieve the encoded message. Since Wireshark displays packet arrival timings, it can act as a very simple covert timing channel receiver as well.

Alternatively, BACnet messages with message IDs or sequence numbering could be used to create a covert timing channel based on packet ordering as proposed by Ahsan and Kundur [AK02] (cf. Chapter 2.4.5).

Broadcast Communication for BACnet Covert/Side Channels

As some BACnet message types (e.g., "Who-Is") are broadcasted to all devices in the network, the receiver can stay unknown since the sender does not have to specify the receiver's address. Even if the covert channel sender will get detected, the receiver can still stay anonymous in the set of potential receivers if the number of the potential receivers is high (i.e., the *anonymity set* comprises enough elements [PK01]). Since all devices can send broadcast messages, the receiver must ensure to only interpret messages sent from the covert channel sender and not those sent from other devices (therefore the MAC address of received broadcast messages can be evaluated).

Using BACnet Broadcast Management Devices (BBMDs, cf. Chapter 2.7.4), covert channels between inter-connected buildings can be realized over the Internet. Alternatively, packets could be spoofed from Internet hosts or could be received by Internet hosts located on the path between two buildings. Due to this broadcast-interconnection, BBMDs can be considered an optimal choice for data exfiltration from enterprise networks to Internet-based receivers outside of a building.

5.4.2 An Active Warden for BACnet

While Section 5.3 introduced the first approach to counter *high-level* covert storage channels and side storage channels in BAS, this section will introduce the first approach to counter *low-level* covert and side channels in a BAS protocol suite, namely BACnet. The design and implementation of the low-level BACnet covert channel protection is

based on joint work with Benjamin Kahler, Thomas Rist and Masood Masoodian that was published in [WKR12] as well as on unpublished work [WRKM13].

Covert channels in BACnet can be differentiated into those based on messages that request information (*requests*) from other devices and those based on messages that provide information to other devices or that acknowledge information (responses and announcements, in the remainder *response*). The following Table 5.1 summarizes the possible communications between low-level and high-level devices. As shown, the only policy-breaking messages are requests from low to high-levels and responses from high to low-levels. We therefore propose to block these messages.

Indeed, low-level covert channels can be created if read-down and write-up messages are utilized to carry hidden information, but by disallowing these messages as well, the only remaining communication that could take place is the communication within the same security level and thus, the practical usefulness of the approach would be very low. However, our approach can be configured to either prevent only the obvious read-ups and write-downs or alternatively, to block all communication between different security levels. Alternatively, a mix of both configurations is feasible, e.g., a write-down from subnet A to subnet B is allowed, but not to subnet C – an administrator would just need to change filter rules.

	Low to High	**High to Low**	**Low to Low**	**High to High**
Request	Read-Up	Read-Down(^{a})	policy-conform	policy-conform
Response	Write-Up(^{a})	Write-Down	policy-conform	policy-conform

Table 5.1: Overview on the policy-conformity of different message types. (^{a} These channels *can* represent low-level covert/side channels and can, as previously mentioned, be blocked dependent on the active warden's configuration.)

Figure 5.5a visualizes the aforementioned covert channels based on requests and responses through read-ups and write-downs. Figure 5.5b shows a potential covert channel that can be exploited if write-up and read-down messages are utilized to carry hidden data since the high-level abstraction of read/write operations is not conform to the low-level representation of network messages that can embed hidden data in all packets.

To protect the BACnet environment against low-level covert storage channels, we introduce a multi-level security architecture that enforces the BLP model within BACnet. Therefore, we propose to change the network topology of BACnet environments and integrate the *BACnet Firewall Router*[6] (BFR) to ensure the enforcement of the BLP

[6] *http://sourceforge.net/projects/bfr/*

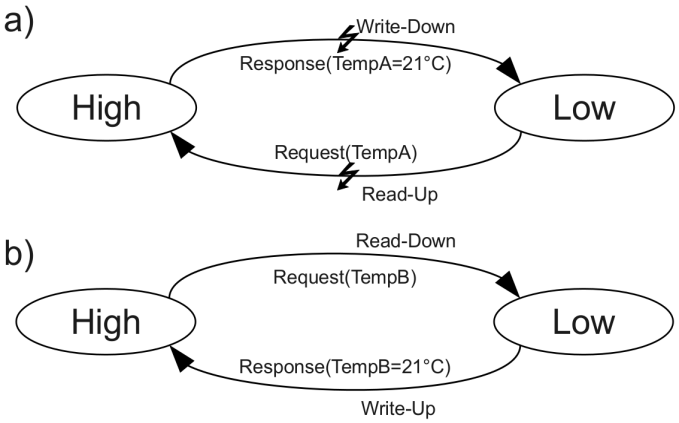

Figure 5.5: Covert channels in low-level BAS messages: a) prevention of read-ups and write-downs, b) utilization of read-downs and write-ups to create covert channels nevertheless.

model. The BFR handles BACnet/IP traffic and is not limited to the capabilities of a plain firewall but is also a BBMD (cf. Chapter 2.7.4) and provides NAT functionality.

Our approach is based on the assumption that devices of different security levels and categories are physically separated from other devices (i.e., a device of a level i is located in another room or floor than a device of level $i+1$). In a default BACnet environment, devices of BACnet (sub)networks can communicate with each other and our approach aims on blocking traffic between these networks if it is not conform to the BLP model.

Therefore, we introduce one BFR for each physically separated environment of a given security level. For instance, a floor can comprise n rooms with devices assigned to security level x while devices in the other rooms of the floor are linked to security level y – in this case, it must be ensured, that the the communication between these rooms is only possible through a BFR. These BFRs are only connected to BACnet devices of the same security level and category and are called *secondary BFRs*.
Additionally, we introduce a global routing BFR called the *primary BFR* that connects all secondary BFRs. Figure 5.6 visualizes our concept.

Configuration of a Secondary BFR

The secondary BFRs are configured to only route policy-conform traffic from and to the BACnet devices they are connected to. If the network traffic from a protected BACnet device passes the secondary BFR, the secondary BFR will forward the traffic to the primary BFR. The primary BFR is a plain router that afterwards sends traffic to the destination network without taking security aspects into account. To reach its

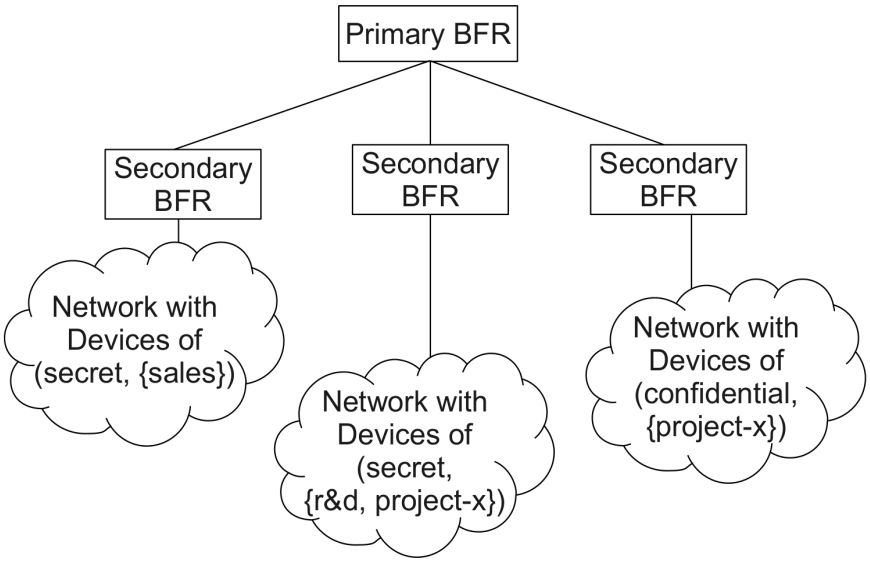

Figure 5.6: BACnet MLS architecture based on BFR.

destination in another network, the primary BFR forwards the traffic to the destination network's secondary BFR. The destination network's secondary BFR again only forwards traffic to the destination device instead of applying any security inspections. Thus, traffic is always inspected by the secondary BFR that is directly connected to the sending BACnet device to ensure that policy-breaking traffic will not reach the primary BFR or the secondary BFR of the destination network.

As pointed out by Kahler, the current BFR implementation cannot counter all types of covert channels since its filtering functionality is limited to a subset of the possible protocol attributes (e.g., it cannot filter application layer traffic) and thus, can also only prevent some of the possible covert channels [WKR12]. Kahler set up virtual LANs to simulate one device per VLAN and configured the XML-based filter configuration to block selected messages between higher and lower leveled VLANs to implemented MLS. The "I-Am-Router-To-Network" message was blocked if it represented a write-down of routing information while the "Who-Is-Router-To-Network" message was blocked if it represented a read-up due to requesting routing information of higher levels.

The BFR distinguishes between *upstream* and *downstream* filters. These two different filters are used to determine the direction a traffic flow takes (from interface 1 to interface 2, or, vice versa). The direction represented by both filter types depends on the configuration. The following sample code was written by Kahler [WRKM13] and represents a filter configuration on a secret level router for traffic coming from and going to a top secret level network. In this case, *downstream* traffic represents traffic from the

top secret level network to the local secret level network while *upstream* traffic represents traffic from the local secret level network to a top secret level network.

```
<Filter [...] >
   <Downstream>
     <!-- Accept read-down messages -->
        <Accept function="WHO-HAS" />
        <Accept function="WHO-IS-ROUTER-TO-NETWORK" />
        <Accept function="WHO-HAS" />
        <!-- Reject a sample write-down message
        (IM-RTN stands for I-am-Router-to-Network) ->
        <Reject function="IM-RTN" />
   </Downstream>
   <Upstream>
     <!-- Accept a write-up message -->
        <Accept function="IM-RTN" />
        <!-- Reject read-up messages -->
        <Reject function="WHO-HAS" />
        <Reject function="WHO-IS-ROUTER-TO-NETWORK" />
        <Reject function="WHO-HAS" />
   </Upstream>
</Filter>
```

Both BACnet network interfaces of the BFR are afterwards interlinked with the *Router* tag:

```
<Router>
   <Adapter client="ip0x" net="1" />
   <Adapter client="ip1x" net="2" />
</Router>
```

Configuration of a Primary BFR

Kahler used the *Switch* tag of the BFR to configure a primary BFR [WRKM13]. The switch tag connects interfaces like a network switch by forwarding data from one interface to the other and vice versa [Ben11]. In comparison to the *Router* tag, the *Switch* tag does not change source addresses at the network layer since the network between the

primary BFR and the secondary BFRs build a single network. BACnet headers are usually required to change their source or destination address inclusion at the network layer header for a real routing. In such a case, the *Router* tag would be required.

The following XML configuration was written by Kahler as well [WRKM13] and connects two Ethernet devices on the primary BFR. Since all security features are implemented by the secondary BFRs, the primary BFR configuration does not need to include such functionality and is straight forward.

```
<BFR>
    <Ethernet device="eth0" server="secret"/>
    <Ethernet device="eth1" server="topsec"/>
    <Switch>
        <Port client="secret" />
        <Port client="topsec" />
    </Switch>
</BFR>
```

5.4.3 Results

The mentioned protocol channel, the mentioned covert storage channel, as well as the proposed covert timing channel can easily be prevented if they are based on read-ups and write-downs (these are covert channels represented by Figure 5.5a).

The BFR-based active warden does not distinguish between intentional and unintentional information leakage and thus can counter both side and covert channels. Additionally, protocol channels, storage channels, and – as a positive side effect – timing channels can be prevented as long as they represent write-downs or read-ups. It must be seen as an additional advantage that our solution does not depend on a specific header area in which covert channel data is embedded, i.e., the approach is independent from the *cover protocol*.

However, our approach cannot prevent covert channels using read-downs and write-ups (Figure 5.5b) without side effects since if read-ups, read-downs, write-ups and write-downs are prevented, the BFR will only allow a communication within the same security level.
Such a read-down/write-up-based covert channel can even be set up in a bi-directional way if two devices (one device of the sender's level and one device of the receiver's level) are available to sender and receiver while providing different access operations (read or

write). To explain this covert channel, an example shall be used: A manager wants to write-down confidential information to an employee. Therefore, the manager sends a request to read a low-level temperature sensor and embeds hidden information in the message. The employee extracts the hidden information and responds by sending a command message to a high-level actuator that also comprises a hidden information. The same channel can be realized as a timing channel if the timing behavior instead of values are observed. Thus, neither storage nor timing channels in write-ups, nor in read-downs can be prevented if not explicitly configured in the BFR. As mentioned earlier, such a set-up will cause the side effect of limiting most of the possible inter-network communication in BACnet.

As already discussed in the context of the building-aware active warden (Section 5.3.4), shared rooms with shared devices must be considered problematic for BACnet environments as well since they can enable data leakage (e.g., a BACnet device could poll the invisibility state of a device; the device will respond if the room is not invisible and will not respond, if the room is invisible, what results in a timing side/covert channel).[7]

The proposed model of introducing filter systems into BAS networks that block non-MLS-conform traffic can be adopted to other BAS systems besides BACnet. However, it requires topological changes in the BAS network and is thus not easy to apply to already existing installations. Additionally, the approach was not developed for wireless BAS in which – at first sight – no easy traffic filtering is feasible. However, if a device-specific encryption would be provided, i.e., the communication between each device pair would be encrypted with different keys in order to allow the BFRs the encryption and decryption of traffic, the approach would be adoptable to wireless BAS as well. BACnet, on the other hand, demands broadcast messages which would not fit into such an encryption concept. As mentioned earlier, broadcasts are an attractive carrier for covert information in BACnet since broadcasts support the anonymity of the channel's receiver.

Since building automation environments do not regularly change (e.g., rooms are not assigned to other security levels on a regular basis, except for shared rooms), the MLS-conform protection can be set up while the BAS is configuried for the organization that uses the building. Later changes are feasible but could require the integration of additional secondary BFRs.

Because our hierarchical model demands a primary BFR, this BFR can be considered as a single point of failure and thus should be implemented in a redundant way. BFR does not support redundancy features but future developments could integrate such features

[7]For realizing invisibility, the BACnet *data hiding* feature (cf. Chapter 2.7.4) could be used.

to provide a high availability based on a redundancy protocol. However, since the last release of the BFR was presented in 2004, we cannot expect a recent improvement in this direction and a project *fork*[8] would be reasonable. Also, the configuration of the secondary BFRs can be considered complex and thus error-prone what could lead to security problems. If an attacker could actively modify the configuration of the BFR, he could break the rules of the BLP model.

Since we introduce the BFR software into the BACnet environment, the BFR must also be seen as a security risk itself. If an attacker can modify a BFR system, he can also run additional malware on the system – a Linux-based BFR host will provide much more possibilities for attacks than an embedded BAS device will provide as the BFR host will comprise a better CPU, more memory, additional libraries and probably the development environment used to compile BFR (a C++ compiler with development libraries).

These mentioned drawbacks of the BFR software underline the need for a better BACnet-capable firewall software or the development of a new alternative that runs on an embedded BACnet device and would thus provide fewer options to execute malware. Additionally, the support for encryption would be valuable for wireless environments. Another important feature that should be integrated in such a software is the support for a redundancy protocol to overcome the mentioned problem that the BFR is a single point of failure.

Another problem is that our current BFR-based approach does not consider the management level of the building automation hierarchy. To provide control and monitoring functionality to a management level, all traffic between the building automation devices and the management level must be directly passed through. However, such a solution would violate the security policy by design but is necessary for the practical use of building automation environments.

The blocking of write-downs and read-ups between different security levels results in less functionality: Higher leveled devices cannot control lower leveled devices (e.g., the manager cannot open a window in an employee's room) and lower leveled devices cannot read information of a higher level (e.g., a control program with a low security level is not able to adjust the heating at a higher level since the write-up would be blind because BLP prevents the read-up of temperature sensor information from the manager's office).

Besides the management layer, another problem lies in the fact that administrative

[8]Project forks are a common method in the open source community in which a copy of a project's source code is taken by a number of developers. Afterwards, the development of the code copy is driven independently from the original project's source code.

persons (e.g., janitors) need access to the building's rooms and devices as well and thus must be treated separately. We propose that physical access control systems (PACs) are provided a direct communication with the central authentication system, i.e., that the filter devices forwards such information unaltered.

A problem can arise if a BACnet transfer is taking place between two buildings connected via the Internet (or an organizational network): An observer in the connecting network can read packets if he is acting as an eavesdropper. In such a case, it would not matter whether the packets were sent with intention from a covert channel sender or whether the packets are leaked trough a side channel. Therefore, the attacker has to read the BACnet frames encapsulated in UDP. To overcome this problem, we propose to encrypt BACnet traffic between buildings using the provided encryption features of the standard (cf. Section 2.7.4).

However, using our approach the most covert and side channels can be prevented as long as they are either represented by write-downs and read-ups or if the BFR is configured in a strict way to also disallow write-ups and read-downs. Our solution, although focusing on low-level channels, can thus prevent all high-level covert and side channels, except for shared devices, if read-ups and write-downs are blocked. Therefore, the approach is also a valuable extension to the building-aware active warden. Additionally, our solution is uncomplicated and, as mentioned before, several of the discussed problems could be solved if a better firewall software than BFR would be available or developed.

5.5 Additional Approaches to Counter Covert and Side Channels in BAS

The previous sections introduced the building-aware active warden and the BFR-based prevention of low-level covert channels in BACnet. The use of the pump and fuzzy time to enhance the use of the building-aware active warden was already discussed in Section 5.3.6 and assumed to be valuable.
In this section, additional means to counter covert and side channels in building automation environments will be discussed.

5.5.1 Device Isolation

After a side or covert channel in an BAS environment got prevented (e.g., a write-down was prevented by the BFR) or a covert channel got detected (e.g., by traffic observation or other means), it is thinkable to temporary isolate a covert channel sender device to prevent additional covert channel data transfer. A BFR could, for instance, prevent the forwarding of packets arriving from the covert channel sender device – even if the forwarding would not break the security policy.

Therefore, it would be important to introduce a rating of devices into the isolation concept: The higher the importance of a device for the building or for the safety of inhabitants, the higher its rating. A smoke detector or an elevator control would, for instance, be linked to a higher rating while the heating actuator in a room would be linked to a lower rating. The lower the rating, the less important side effects will result from an isolation of a covert channel sender device.

A problematic aspect of isolations is the potential of denial of service attacks. For instance, in BACnet, an attacker could spoof BACnet messages. If an attacker sends messages that break the NRU or NWD rules with a spoofed address of any available device, all inter-network communication would be blocked by the BFR.

5.5.2 Traffic Observation

Another thinkable approach to counter side and covert channels in BAS is to introduce traffic observation. Therefore, statistical behavior of network data flow can be taken into account. For this purpose, the approaches discussed in Chapter 2.5.4 could be adopted. For instance, the observation of inter packet gaps as proposed by Berk et al. [BGC05] could be modified in a way that the inter packet gaps of BACnet frames could be observed to detect abnormal behavior. Similarly, the inter packet gaps could be evaluated using machine learning as done by Zander [Zan10].

Another important aspect of traffic observation is the detection of side channels in encrypted BAS traffic. If sensor updates are broadcasted, the occurrence of a message from a sensor device's source address leaks the information of a changed state even if the message's value is encrypted but the NPDU header remains unencrypted: The occurrence rate of messages could therefore be observed to detect abnormal state switching behavior. It must be taken into account that the carried information of such a state change depends on the device type: A message from a presence sensor either shows that the presence of a person was detected or that a person left a given location. If many

messages are sent from a presence sensor, one or many persons are at a given location.[9] On the other hand, only few information is leaked by a temperature sensor in a room. One cannot easily conclude whether the temperature increases, decreases or simply alternates between two precise values (e.g., between 20.1°C and 20.2°C). A similar approach for ZigBee in unencrypted environments was done by Kahler in [Kah12]. Kahler observed that the presence sensor values in office rooms follow a typical distribution of state changes (events) per time of the day. While Kahler's approach aims on detecting physical intrusions, it is assumed to be useful to detect covert and side channels. We will evaluate the usefulness of Kahler's approach to detect both channel types in joint future work.

5.5.3 Adoption of the Anti-PSCC Active Warden

In Chapter 4, the anti-PSCC active warden was introduced. Since the active warden was designed to work with all protocol channels it can distinguish, we modified it to counter the previously discussed BACnet-based protocol channel as well. Therefore, support for UDP packets with BACnet encapsulation was integrated to distinguish alternating *message type* values on the BACnet network layer. This modification allows both the verification that the active warden can operate in BACnet environments and the verification that the discussed BACnet protocol channel can be limited using delays.

To provide comparable results with the protocol channel limitation discussed in Chapter 4, the same virtual machines as used for PCT were used to transfer BACnet/IP traffic as well. The sending script of the BACnet protocol channel was modified to send random input as used for the PCT evaluation.

Figure 5.7 compares the maximum error-free bitrates of PCT and the BACnet-based protocol channel: Since T has the same value for both cases (the computing power, the available memory, and the network connection are the same; the packet sizes are nearly equal), both channels result in nearly exactly the same maximum error-free bitrates depending on the introduced delay d.

As a side effect, BACnet clients can repeat unacknowledged requests after a waiting time Δt. If the applied delay $d > \Delta t$, BACnet clients could send delayed packets again to BACnet servers. This side effect can be considered unproblematic since Δt is usually $\geq 2 - 3s$ what is enough to limit B to an acceptable value.

[9] Indeed, a presence sensor could also be triggered by other events, such as plant motions caused by a storm.

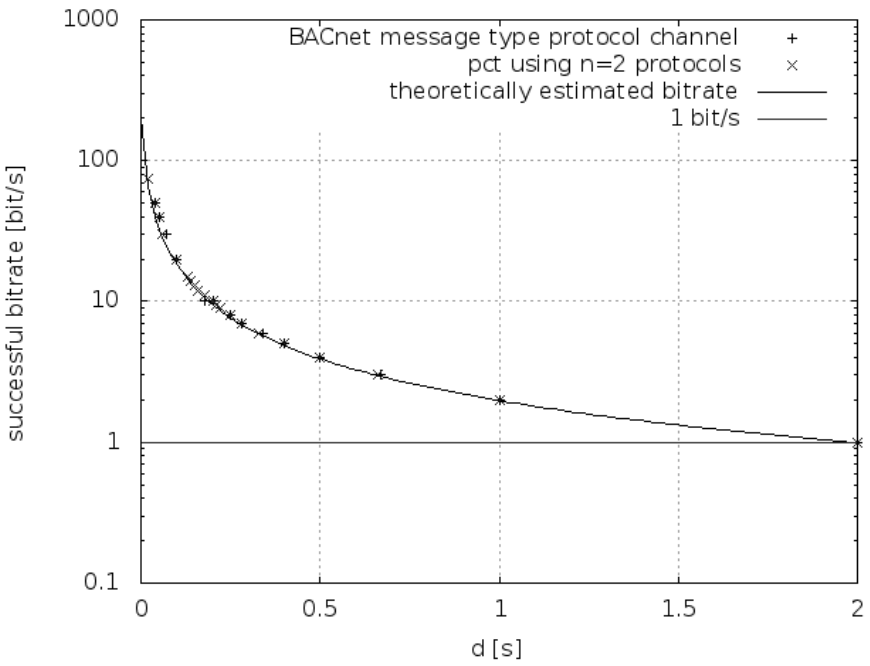

Figure 5.7: Maximum successful bitrates of PCT and the BACnet-based protocol channel dependent on the active warden's constant delay.

For slower BACnet connections, e.g., BACnet over MS/TP (between 9600 bps and 76800 bps) or BACnet over KNX (usually 9600 bps), T would raise and the maximum error-free bitrate of the protocol channel would be lower while the active warden's efficiency would be higher.

5.6 Conclusion and Future Work

This chapter discussed the presence of covert (and side) storage channels in BAS as well as their usefulness for adversaries. It was shown that two different types of covert (and side) channels exist for BAS: High-level and low-level covert (and side) channels. High-level channels are abstract channels based on interactions of subjects with the BAS and low-level channels are typical network covert (or side) channels that embed confidential information within network packets.
While high-level covert channels do directly represent read-ups and write-downs, low-level covert channels, as comprising requests and responses, can break the NRU and NWD rules even if no higher leveled sensor information is read or lower leveled actuator settings are changed because network messages can comprise hidden information

independent from their abstract (i.e., high-level) operation type (read/write).

For the prevention of high-level covert (and side) storage channels, the concept of a building-aware active warden was introduced. The active warden is a middleware solution that sandboxes all BAS applications by enforcing the BLP model.

Afterwards, the presence of covert and side channels in BACnet was shown and the prevention of BACnet-based covert and side channels using topological changes in the BACnet environment in combination with the integration of the *BACnet Firewall Router* (BFR) was discussed.

We explained that both the high-level as well as the low-level protection approach can counter some of the discussed covert channels. However, shared resources (e.g., meeting rooms with shared devices) require additional means to prevent especially timing channels.

Future work will include research on the presence of covert channels in other BAS protocol suites (e.g., EIB/KNX).

We do not expect micro protocols to be useful in the context of BAS – especially due to the usually limited size of building automation networks and their limited routing capabilities. However, if future BAS should become more based on TCP/IP (this is currently rarely the case) and thus, would be linked to a direct Internet connectivity as well as to advanced routing capabilities, dynamic covert channel overlays based on micro protocols could become useful for building automation environments as well.

6 Discussion and Future Work

The previous chapters concluded and discussed future work in the context of their particular topic. This chapter will provide a more general conclusion and outlook.

While covert channels were primarily investigated by a small number of researchers from a technical point of view within the last decades, we can assume that covert channels *with micro protocols* will become a more valuable means to counter Internet censorship as an increasing number of the world's population will be provided with Internet access. Therefore, we expect a shift from only specialist users towards an increasing number of end users like journalists. On the other hand, such novel approaches for network covert channels will improve malware communications and must thus be seen as a risk – another reason why research for such techniques and for means to counter malware communication is a necessity.

The *first part* of this thesis contributed to the development of covert channels in a way that supports covert channel design in general instead of improving only selected techniques (e.g., improving the hiding of data in a selected area of a protocol's header): Smaller and low-attention raising micro protocols and a better adaptiveness within the *network environment learning* phase enrich protocol hopping covert channels.

On the other hand, a new means to counter protocol switching covert channels was presented and especially evaluated to be efficient against protocol channels and protocol hopping covert channels *without* micro protocols. The developed active warden allows configurable limits for such covert channels in a way that the usability for legitimate network users is only affected in a limited way.

However, this thesis could not present an efficient means to counter protocol hopping covert channels *with* micro protocols that provide a reliable data transfer. Such micro protocol embedding covert channels are considered hard to limit and block. Future work must find new means capable of achieving either a limitation or a prevention of these channels. The focus of this thesis are storage channels and the use of micro protocols in timing channels was not discussed. Future work may therefore study micro protocols in timing channels.

The *second part* of this thesis presented the existence and a definition of covert and side channels in building automation systems (BAS). These channels can be used to observe events in BAS and to exfiltrate confidential information. It was discovered that two different kinds of covert channels (and side channels) are present in such environments: High-level covert and side channels based on the interaction with the building and low-level covert and side channels based on network packets. Like in Chapter 3 for the case of micro protocols, Chapter 5 did also aim on providing a general focus instead of discussing a few selected covert channel embedding techniques: Two active wardens were presented to counter both high and low-level covert and side channels in building automation environments. While the low-level approach was BACnet-specific, it can be applied to other systems besides BACnet as well and is independent from the cover protocol. Besides, the active warden used to limit protocol switching covert channels in Chapter 4 was shown to be able to counter protocol channels in BACnet as well.

With the increasing popularity and shrinking equipment prices, building automation will become more important. Preventing the observability of subjects in buildings will thus become more important as well. Therefore, future work should evaluate means like the *network pump* or *fuzzy time* in the context of building automation environments. However, building automation security research will not be limited to covert and side channel prevention, detection, and limitation as hardening of BAS components, the re-engineering of existing insecure equipment and means to counter other security problems as discussed in Chapter 2.7.3 must be addressed as well. Besides, the acceptance for BAS will only be high on a long-term basis if end users will trust their vendor's protection means. Cunningham et al. see trust on a scale instead of viewing it as a black/white issue [CMA10], i.e., we can assume that BAS users (e.g., inhabitants) do not have total trust or zero trust in the BAS but it must be ensured that the trust in a system is as high as possible to provide a good *benefit* (e.g., comfort) besides the *costs* (e.g., expensive hardware or known security problems). If private homes are understood as a save haven for inhabitants, the threat of side channel-based observation is of a high importance to ensure the trust of inhabitants in the BAS. Besides side channels, covert channels must be primarily seen as a threat for the trust in building automation environments by organizations.

Bibliography

[Aga00] J. Agat. Transforming out timing leaks. In *Proc. 27th ACM Symposium on Principles of Programming Languages (POPL)*, pages 40–53. ACM Press, 2000.

[Ahs02] K. Ahsan. Covert channel analysis and data hiding in TCP/IP. Master's thesis, University of Toronto, 2002.

[AK02] K. Ahsan and D. Kundur. Practical data hiding in TCP/IP. In *Proc. Workshop on Multimedia Security at ACM Multimedia '02*, December 2002.

[Aka06] Akamai. Retail web site performance, 2006. http://www.-akamai.com/dl/reports/Site_Abandonment_Final_Report.pdf, retrieved: January 2013.

[ALJY12] D. Anthony, P. Lutz, D. Johnson, and B. Yuan. A behavior based covert channel within anti-virus updates. In *Proc. 2012 International Conference on Security and Management (SAM'12)*, pages 3–7, 2012.

[AM11] J.C. Acosta and J.D. Medrano. Using a novel blending method over multiple network connections for secure communication. In *Proc. Military Communications Conference 2011 – MILCOM 2011*, pages 1460–1465, 2011.

[AM12] J.C. Acosta and J.D. Medrano. NBCS: Secure communication via distributed covert channels in active network traffic. *Security and Communication Networks Journal*, 2012. (submitted).

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

[ANS10] ANSI/ASHRAE. Addendum "g" to ANSI/ASHRAE standard 135-2008 (BACnet — a data communication protocol for building automation and control networks), 2010.

[AQDS10] H. Al-Qaheri, S. Dey, and S. Sanyal. Hiding inside HTML and other source codes. *CoRR*, abs/1003.3457, 2010. http://arxiv.org/abs/1003.3457, retrieved: January 2013.

[AW09] R. Accorsi and C. Wonnemann. Detective information flow analysis for business processes. In *Proc. Business Process, Services Computing and Intelligent Service Management (BPSC)*, volume 147 of *LNI*, pages 223–224. GI, 2009.

[AW11a] R. Accorsi and C. Wonnemann. InDico: Information flow analysis of business processes for confidentiality requirements. In *Proc. 6th International Workshop on Security and Trust Management (STM 2010)*, pages 194–209. Springer, 2011.

[AW11b] R. Accorsi and C. Wonnemann. Informationsfluss-Mechanismen zur Zertifizierung von Cloud-basierten Geschäftsprozessen. In *Deutscher IT-Sicherheitskongress des BSI*, pages 149–160. SecuMedia-Verlag Bonn, May 2011. (in German).

[Bac12] P. Backs. Automatisches Routing in einem Mikroprotokoll für Covert-Channel Netze. Master's thesis, University of Hagen, 2012. (in German).

[Bau03] M. Bauer. New covert channels in HTTP: adding unwitting web browsers to anonymity sets. In *Proc. 2003 ACM Workshop on Privacy in the Electronic Society*, pages 72–78. ACM, 2003.

[BBM+05] M. Baldoni, C. Baroglio, A. Martelli, et al. Verifying protocol conformance for logic-based communicating agents. In *Proc. 5th International Workshop on Computational Logic in Multi-Agent Systems*, pages 192–212. Springer, 2005.

[Bec11] K.-B. Becker. *Internetzensur in China*. VS Verlag für Sozialwissenschaften, 2011. (in German).

[BEF+00] A. Bouajjani, J. Esparza, A. Finkel, et al. An efficient automata approach to some problems on context-free grammars. *Information Processing Letters*, 74(5-6):221–227, June 2000.

[Ben11] J. Bender. BACnet firewall router documentation, version 0.2, 2011. http://bfr.svn.sourceforge.net/viewvc/bfr/tags/0.2/Documentation/, retrieved: January 2012.

[Ber05] D. Berrange. Simulating WAN network delay, 2005. http://people.redhat.com/berrange/notes/network-delay.html, retrieved: January 2013.

[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.

[BGNS06] E. Brickell, G. Graunke, M. Neve, and J.-P. Seifert. Software mitigations to hedge AES against cache-based software side channel vulnerabilities. Cryptology ePrint Archive, Report 2006/052, 2006.

[Bis03] M. Bishop. *Computer Security. Art and Science*. Pearson Education, 2003.

[BK07] A. Baliga and J. Kilian. On covert collaboration. In *Proc. 9th Workshop on Multimedia & Security*, pages 25–34. ACM, 2007.

[Bor08] T. Borland. Guide to encrypted dynamic covert channels, December 2008. http://turboborland.blogspot.com/2008/12/guide-to-encrypted-dynamic-covert.html, retrieved: January 2013.

[BP09] K. Borders and A. Prakash. Quantifying information leaks in outbound web traffic. In *Proc. 30th IEEE Symposium on Security and Privacy*, pages 129–140, 2009.

[BR05] R. Bidou and F. Raynal. Covert channels. Technical report, IV2-Technologies, 2005. http://www.iv2-technologies.com/CovertChannels.pdf, retrieved: January 2013.

[BS80] G. V. Bochmann and C. A. Sunshine. Formal methods in communication protocol design. *IEEE Transactions on Communications*, COM-28(4):624–631, April 1980.

[BTA+06] L. Bernaille, R. Teixeira, I. Akodkenou, et al. Traffic classification on the fly. *SIGCOMM Computer Communication Review*, 36(2):23–26, April 2006.

[BWK12] P. Backs, S. Wendzel, and J. Keller. Dynamic routing in covert channel overlays based on control protocols. In *Proc. International Workshop on Information Security, Theory and Practice (ISTP-2012)*, pages 32–39. IEEE, 2012.

[CBS04] S. Cabuk, C. E. Brodley, and C. Shields. IP covert timing channels: design and detection. In *Proc. 11th ACM Conference on Computer and Communications Security*, pages 178–187. ACM, 2004.

[CBS09] S. Cabuk, C. E. Brodley, and C. Shields. IP covert channel detection. *ACM Transactions on Information and System Security (TISSEC)*, 12(4):22:1–22:29, April 2009.

[CMA10] S. J. Cunningham, M. Masoodian, and A. Adams. Privacy issues for online personal photograph collections. *Journal of Theoretical and Applied Electronic Commerce Research*, 5(2):26–40, 2010.

[Cra98] S. Craver. On public-key steganography in the presence of an active warden. In *Proc. Information Hiding*, volume 1525 of *LNCS*, pages 355–368. Springer, 1998.

[dae97] daemon9. LOKI2 (the implementation). *Phrack Magazine*, 7(51), 1997. http://www.phrack.org/issues.html?issue=51&id=6, retrieved: January 2013.

[dAJ05] R. deGraaf, J. Aycock, and M. Jr. Jacobson. Improved port knocking with strong authentication. In *Proc. 21st Annual Computer Security Applications Conference*, ACSAC '05, pages 451–462. IEEE Computer Society, 2005.

[Das12] A. Das. Steganography: Secret data hiding in multimedia. In *Signal Conditioning*, volume 180 of *Signals and Communication Technology*, pages 275–295. Springer, 2012.

[Dep85] Department of Defense. Trusted computer system evaluation criteria (TCSEC, DoD 5200.28-STD, orange book), 1985.

[Eck12] C. Eckert. *IT-Sicherheit. Konzepte, Verfahren, Protokolle.* Oldenbourg Wissenschaftsverlag GmbH, 7 edition, 2012. (in German).

[Ega05] D. Egan. The emergence of ZigBee in building automation and industrial control. *Computing Control Engineering Journal*, 16(2):14–19, 2005.

[EGC+10] W. Enck, P. Gilbert, B.-G. Chun, , et al. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6. USENIX Association, 2010.

[ELP+12] C. Efstratiou, I. Leontiadis, M. Picone, et al. Sense and sensibility in a pervasive world. In *Proc. 10th International Conference on Pervasive Computing*, Pervasive'12, pages 406–424. Springer, 2012.

[Eße05] H.-G. Eßer. Ausnutzung verdeckter Kanäle am Beispiel eines Web-Servers. Master's thesis, RWTH Aachen, 2005. (in German).

[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*, pages 18–35. Springer, 2003.

[Fis12] D. Fisk. Cyber security, building automation, and the intelligent building. *Intelligent Buildings International*, 4(3):169–181, 2012.

[FMS11] W. Frączek, W. Mazurczyk, and K. Szczypiorski. How hidden can be even more hidden? In *Proc. 3rd International Conference on Multimedia Information Networking and Security (MINES)*, pages 581–585. IEEE, 2011.

[FS11] K. R. Fall and W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley Professional Computing Series. Addison-Wesley, 2nd revised edition, 2011.

[GBC06] A. Giani, V. H. Berk, and G. V. Cybenko. Data exfiltration and covert channels. In *Proc. SPIE 6201, Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, pages 620103–620103–11. SPIE, 2006.

[GGJL12] U. Greveler, P. Glösekötter, B. Justus, and D. Loehr. Multimedia content identification through smart meter power usage profiles. In *Computers, Privacy and Data Protection*. 2012. https://www.nds.rub.de/media/nds/veroeffentlichungen/2012/07/24/ike-2012.pdf, retrieved: January 2013.

[GGLT03] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts. Covert messaging through TCP timestamps. In *Proc. 2nd International Conference on Privacy Enhancing Technologies*, pages 194–208. Springer, 2003.

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

[GK02a] V. Gorodetski and I. Kotenko. Attacks against computer network: Formal grammar-based framework and simulation tool. In *Recent Advances in Intrusion Detection*, volume 2516 of *LNCS*, pages 219–238. Springer, 2002.

[GK02b] V. Gorodetski and I. Kotenko. Attacks against computer network: Formal grammar-based framework and simulation tool. In *Recent Advances in Intrusion Detection*, volume 2516 of *LNCS*, pages 219–238. Springer, 2002.

[GKNP06] W. Granzer, W. Kastner, G. Neugschwandtner, and F. Praus. Security in networked building automation systems. In *Proc. 2006 IEEE International Workshop on Factory Communication Systems*, pages 283–292, 2006.

[GPK10] W. Granzer, F. Praus, and W. Kastner. Security in building automation systems. *IEEE Transactions on Industrial Electronics*, 57(11):3622–3630, November 2010.

[GRK08] W. Granzer, C. Reinisch, and W. Kastner. Denial-of-service in automation systems. In *Proc. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008)*, pages 468–471, 2008.

[Gro09] J.F. Grosse, editor. *Building Automation. System Integration with Open Protocols*. American Technical Publishers, 2009.

[Gös10] R. Gössner. Vom Ende der Vertraulichkeit – Überwachungskosmos der modernen Tele-Kommunikation. In *Die Funktion "verdeckter Kommunikation". Impulse für eine Technikfolgenabschätzung zur Steganographie*, volume 9 of *Social Issues*, pages 121–166. LIT Verlag, 2010. (in German).

[Har77] J. Harangozó. An approach to describing a data link level protocol with a formal language. In *Proc. 5th Data Communication Symposium*, pages 4–37–4–49, September 1977.

[Har78] J. Harangozó. Protocol definition with formal grammars. In *Proc. Computer Network Protocols Symposium*, pages F6–1–F6–10, February 1978.

[HBG06] D. G. Holmberg, J. Bender, and M. Galler. Using the BACnet firewall router. *BACnet Today (A Supplement to ASHRAE Journal)*, pages B10–B14, November 2006.

[HKM11] M. Hall, R. Koornstra, and M. Mowbray. Efficient prevention of credit card leakage from enterprise networks. In *Proc. Communications and Multimedia Security*, volume 7025 of *LNCS*, pages 238–240. Springer, 2011.

[Hol03] D. G. Holmberg. Enemies at the gates. *BACnet Today (A Supplement to ASHRAE Journal)*, pages B24–B28, 2003.

[Hol05] D. G. Holmberg. Secure messaging in BACnet. *BACnet Today (A Supplement to ASHRAE Journal)*, pages B23–B26, 2005.

[HPK01] M. Handley, V. Paxson, and C. Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. 10th USENIX Security Symposium*, volume 10, pages 115–131, 2001.

[Hu91] W.-M. Hu. Reducing timing channels with fuzzy time. In *Proc. 1991 Symposium on Security and Privacy*, pages 8–20. IEEE, 1991.

[HU94] J. E. Hopcroft and J. D. Ullman. *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Addison-Wesley, 1994. (in German).

[Jac90] V. Jacobson. Compressing TCP/IP headers for low-speed serial links (RFC 1144), February 1990.

[JLSN09] L. Ji, H. Liang, Y. Song, and X. Niu. A normal-traffic network covert channel. In *Proc. International Conference on Computational Intelligence and Security*, pages 499–503, 2009.

[JLY09] D. Johnson, P. Lutz, and B. Yuan. Behavior-based covert channels in cyberspace. In *Proc. 4th International ISKE Conference on Intelligent Systems and Knowledge Engineering Intelligent Decision Making Systems*, pages 311–318, November 2009.

[JMS10] B. Jankowski, W. Mazurczyk, and K. Szczypiorski. Information hiding using improper frame padding. *eprint arXiv:1005.1925*, 2010.

[Kah12] B. Kahler. Physical Intrusion Detection für die Gebäude-Automation. Ein lastbasierter Ansatz. Technical report, Computer Science Department, Augsburg University of Applied Sciences, 2012. (unpublished; in German).

[KBc05] T. Kohno, A. Broido, and kc claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[KBR+07] T. Kleinberger, M. Becker, E. Ras, et al. Ambient intelligence in assisted living: Enable elderly people to handle future interfaces. In *Proc. Universal Access in Human-Computer Interaction. Ambient Interaction*, volume 4555 of *LNCS*, pages 103–112. Springer, 2007.

[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.

[KHE11] G. Kempter, M. Herburger, and N. Ess. Neue Einsatzmöglichkeiten von Gebäudetechnologie in betreubaren Seniorenwohnungen. In *Intelligent Wohnen. Zusammenfassung der Beiträge zum Usability Day IX*, pages 46–52. Pabst Science Publishers, 2011. (in German).

[Kle78] L. Kleinrock. Principles and lessons in packet communications. *Proc. IEEE*, 66(11):1320–1329, November 1978.

[KM93] M. H. Kang and I.S. Moskowitz. A pump for rapid, reliable, secure communication. In *Proc. 1st ACM Conference on Computer and Communication Security*, pages 119–129, November 1993.

[KNSN05] W. Kastner, G. Neugschwandtner, S. Soucek, and H.M. Newman. Communication systems for building automation and control. *Proc. IEEE*, 93(6):1178–1203, June 2005.

[KRS+11] M. Kugler, F. Reinhart, K. Schlieper, et al. Architecture of a ubiquitous smart energy management system for residential homes. In *Proc. 12th Annual Conference New Zealand Chapter of the ACM Special Interest Group on Computer-Human Interaction*, CHINZ'11, pages 101–104. ACM, 2011.

[KW13] B. Kahler and S. Wendzel. How to own a Building? Wardriving gegen die Gebäudeautomation. In *Beiträge zum 20. DFN Workshop zur Sicherheit in vernetzten Systemen*, 2013. (in German, to appear).

[Kön03] H. König. *Protocol Engineering. Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen*. Teubner, 2003. (in German).

[Lam73] B. W. Lampson. A note on the confinement problem. *Comm. ACM*, 16(10):613–615, 1973.

[LB73] L. J. LaPadula and D. E. Bell. Secure computer systems: Mathematical foundations. MITRE Technical Report 2547, Volume II, MITRE, May 1973.

[LCC07] X. Luo, E.W.W. Chan, and R.K.C. Chang. Cloak: A ten-fold way for reliable covert communications. In *Proc. 12th European Symposium On Research In Computer Security (ESORICS 2007)*, volume 4734 of *LNCS*. Springer, 2007.

[LdILB+09] D. López-de Ipiña, X. Laiseca, A. Barbier, et al. Infrastructural support for ambient assisted living. In *Proc. 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*, volume 51 of *Advances in Soft Computing*, pages 66–75. Springer, 2009.

[LGC08] Z. Li, A. Goyal, and Y. Chen. Honeynet-based botnet scan traffic analysis. In *Botnet Detection: Countering the Largest Security Threat*, 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 International Conference on Autonomic and Trusted Computing*, ATC'11, pages 106–117. Springer, 2011.

[LLC06] N. Lucena, G. Lewandowski, and S. Chapin. Covert channels in IPv6. In *Proc. 5th International Workshop on Privacy Enhancing Technologies (PET 2005)*, volume 3856 of *LNCS*, pages 147–166. Springer, 2006.

[LLC07] G. Lewandowski, N. Lucena, and Steve C. Analyzing network-aware active wardens in IPv6. In *Proc. Information Hiding*, volume 4437 of *LNCS*, pages 58–77. Springer, 2007.

[LM83] R. J. Linn and W. H. McCoy. Producing tests for implementations of OSI protocols. In *Proc. Protocol Specification, Testing, and Verification*, pages 505–520, 1983.

[LZCZ11] X. Li, Y. Zhang, F.T. Chong, and B.Y. Zhao. A covert channel analysis of a real switch. Technical report, Dep. of Computer Science, University of California, Santa Barbara, 2011.

[McC88] D. McCullough. Noninterference and the composability of security properties. In *Proc. 1988 IEEE Symposium on Security and Privacy*, pages 177 –186, 1988.

[McH95] J. McHugh. Covert channel analysis. technical memo 5540:080a, Naval Research Laboratory, 1995.

[McH01] J. McHugh. An information flow tool for Gypsy – an extended abstract revisited. In *Proc. 17th Annual Computer Security Applications Conference*. The Applied Computer Security Associates (ACSA), 2001.

[MD96] C. Denis Mee and Eric D. Daniel. *Magnetic Storage Handbook*. McGraw Hill, 2 edition, 1996.

[Mem07] M. Memelli. g00gle crewbots, 2007. http://gray-world.net/projects/papers/gbots-1.0.txt, retrieved: January 2013.

[MHH09] H. Merz, T. Hansemann, and C. Hübner. *Building Automation. Communication systems with EIB/KNX, LON and BACnet*. Signals and Communication Technology. Springer, 2009.

[Mil99] J. Millen. 20 years of covert channel modeling and analysis. In *Proc. 1999 IEEE Symposium on Security and Privacy*, pages 113–114, 1999.

[ML05] S. J. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In *Proc. Information Hiding Conference 2005*, volume 3727 of *LNCS*, pages 247–261. Springer, 2005.

[MND12] M. Milutinovic, V. Naessens, and B. De Decker. Privacy-preserving scheduling mechanisms for eHealth systems. In *Proc. 13th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS 2012)*, volume 7394 of *LNCS*, pages 198–200. Springer, 2012.

[Mor02] J. Morris. IPTables::IPv4::IPQueue module for Perl, 2002. http://search.cpan.org/~jmorris/perlipq-1.25/IPQueue.pm, retrieved: January 2013.

[MRH+08] A. Maña, C. Rudolph, M. Hoffmann, et al. Towards semantic resolution of security in ambient environments. In *Proc. Developing Ambient Intelligence*, pages 13–22. Springer Paris, 2008.

[Mur07] S. J. Murdoch. *Covert channel vulnerabilities in anonymity systems*. PhD thesis, University of Cambridge (Computer Laboratory), 2007.

[MWJH00] G.R. Malan, D. Watson, F. Jahanian, and P. Howell. Transport and application protocol scrubbing. In *Proc. 19th Annual Joint Conference IEEE Computer and Communications Societies (INFOCOM 2000)*, pages 1381–1390, 2000.

[New10] H. M. Newman. Broadcasting BACnet. *BACnet Today (A Supplement to ASHRAE Journal)*, pages B8–B12, November 2010.

[NTP07] T. Novak, A. Treytl, and P. Palensky. Common approach to functional safety and system security in building automation and control systems. In *Proc. 12th IEEE Conference on Emerging Technologies and Factory Automation*, pages 1141–1148, 2007.

[NWET05] P.A. Nixon, W. Wagealla, C. English, and S. Terzis. *Security, Privacy and Trust Issues in Smart Environments*, pages 249–270. Wiley, 2005.

[NY85] N. Nounou and Y. Yemini. Development tools for communication protocols. Technical report, Computer Science Department, Columbia University, New York, February 1985.

[OOS+96] N. Ogurtsov, H. Orman, R. Schroeppel, et al. Covert channel elimination protocols. Technical report, Department of Computer Science, University of Arizona, 1996.

[Ope13] OpenBSD. pf.conf - packet filter configuration file (manual page), January 2013.

[PAK99] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Information hiding – a survey. *Proc. IEEE*, 87(7):1062–1078, 1999.

[Pfi96] B. Pfitzmann. Information hiding terminology - results of an informal plenary meeting and additional proposals. In *Proc. First International Workshop on Information Hiding*, volume 1174 of *LNCS*, pages 347–350. Springer, 1996.

[PK91] P. A. Porras and R. A. Kemmerer. Covert flow trees: A technique for identifying and analyzing covert storage channels. In *Proc. IEEE Symposium on Security and Privacy*, pages 36–51, 1991.

[PK01] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity – a proposal for terminology. In *Proc. Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 1–9. Springer, 2001.

[Pos81] J. Postel. RFC 793: Transmission control protocol. DARPA Internet programm protocol specification, September 1981.

[Rie10] T. Riegler. *Komfort und Energieeinsparung mit HomeMatic*. Franzis, 2010.

[RIMT06] D. Ritter, B. Isler, H.-J. Mundt, and S. Treado. Access control in BACnet. *BACnet Today*, November 2006.

[RM08a] B. Ray and S. Mishra. A protocol for building secure and reliable covert channel. In *Proc. 6th Annual Conference on Privacy, Security and Trust (PST 2008)*, pages 246–253. IEEE, 2008.

[RM08b] B. Ray and S. Mishra. Secure and reliable covert channel. In *Proc. 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to meet the Cyber Security and Information Intelligence Challenges ahead (CSIIRW'08)*, pages 8:1–8:3. ACM, 2008.

[ROL12] R. Rios, J.A. Onieva, and J. Lopez. HIDE_DHCP: Covert communications through network configuration messages. In *Proc. IFIP TC 11 27th International Information Security Conference*. Springer, 2012.

[Row97] C. H. Rowland. Covert channels in the TCP/IP protocol suite. *First Monday*, 2(5), May 1997. http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/-view/528/449, retrieved: January 2013.

[RS80] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, 1980.

[Rut04] J. Rutkowska. The implementation of passive covert channels in the Linux kernel, 2004. Speech held at the 21st Chaos Communication Congress, Berlin, Germany, http://events.ccc.de/congress/2004/fahrplan/files/319-passive-covert-channels-slides.pdf, retrieved: January 2013.

[RWM+11] T. Rist, S. Wendzel, M. Masoodian, et al. Creating awareness for efficient energy use in smart homes. In *Intelligent Wohnen. Zusammenfassung der Beiträge zum Usability Day IX*, pages 162–168. Pabst Science Publishers, 2011.

[RWMA12] T. Rist, S. Wendzel, M. Masoodian, and E. André. Next-generation home automation systems. In *Technik für Menschen im nächsten Jahrzehnt. Beiträge zum Usability Day X*, pages 80–87. Pabst Science Publishers, June 2012.

[SC99] S. Sharples and V. Callaghan. A multi-agent architecture for intelligent building sensing and control. *Sensor Review Journal*, 19(2):135–140, 1999.

[SdSNL06] A. Singh, A.L. Mora dos Santos, Ö. Nordström, and C. Lu. Stateless model for the prevention of malicious communication channels. *International Journal of Computers and Applications*, 28:285–297, 2006.

[Sha02] U. Shankar. Active mapping: Resisting NIDS evasion without altering traffic. Technical Report UCB//CSD-2-03-1246, Computer Science Division, University of California Berkeley, December 2002.

[Sim83] G. J. Simmons. The prisoners' problem and the subliminal channel. In *Proc. CRYPTO*, pages 51–67, 1983.

[SK06] M. Smeets and M. Koot. Research report: Covert channels. Technical report, University of Amsterdam, February 2006.

[SKZV06] N. Schear, C. Kintana, Q. Zhang, and A. Vahdat. Glavlit: Preventing Exfiltration at Wire Speed. In *Proc. 5th ACM Workshop on Hot Topics in Networks (HotNets-V)*, pages 133–138. ACM SIGCOMM, November 2006.

[Sł04] J. Sławiński. Reverse tunneling techniques: theoretical requirements for the GW implementation, 2004. http://gray-world.net/projects/papers/rtt.txt, retrieved: January 2013.

[SM03] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

[Smi07] G. Smith. Principles of secure information flow analysis. In *Malware Detection*, volume 27 of *Advances in Information Security*, pages 291–307. Springer, 2007.

[SMM+08] K. Szczypiorski, I. Margasinski, W. Mazurczyk, et al. TrustMAS: Trusted communication platform for multi-agent systems. In *Proc. On the Move to Meaningful Internet Systems: OTM 2008*, volume 5332 of *LNCS*, pages 1019–1035. Springer, 2008.

[Sno03] D. Snoonian. Smart buildings. *IEEE Spectrum*, 40(8):18–23, August 2003.

[Sno12] Snort Project. Snort users manual 2.9.3, May 2012.

[SR09] J. Simpson and M. Riesberg, editors. *Building Automation. System Integration with Open Protocols*. American Technical Publishers, Inc., 2009.

[SS00] A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proc. 13th IEEE Workshop on Computer Security Foundations*, CSFW'00, pages 200–214. IEEE Computer Society, 2000.

[SSP+12] A. Swinnen, R. Strackx, P. Philippaerts, et al. Protoleaks: A reliable and protocol-independent network covert channel. In *Proc. 8th International Conference on Information Systems Security (ICISS 2012)*, volume 7671 of *LNCS*, pages 119–133. Springer, 2012.

[ST03] C. Schwaiger and A. Treytl. Smart card based security for fieldbus systems. In *Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2003)*, volume 1, pages 398–406, 2003.

[Stø09] D. Stødle. Ping tunnel – for those times when everything else is blocked, 2009. http://www.cs.uit.no/~daniels/PingTunnel/, retrieved: January 2013.

[Sun78] C. A. Sunshine. Survey of protocol definition and verification techniques. In *Proc. Computer Network Protocols Symposium*, pages F1–1–F1–4, February 1978.

[SZ12] S. Soucek and G. Zucker. Current developments and challenges in building automation. *e & i (Elektrotechnik und Informationstechnik)*, 129(4):278–285, 2012.

[TA05] E. Tumoian and M. Anikeev. Network based detection of passive covert channels in TCP/IP. In *Proc. IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, pages 802–809, 2005.

[Tea05] Gray-World Team. Covert channels through the looking glass, 2005. http://gray-world.net/projects/papers/cc.txt, retrieved: January 2013.

[TF12] P. Trinius and F.C. Freiling. Filtern von Spam-Nachrichten mit kontextfreien Grammatiken. In *Proc. Sicherheit'12*, volume P-195 of *LNI*, pages 163–174, 2012. (in German).

[TG88] C.-R. Tsai and Virgil D. Gligor. A bandwidth computation model for covert storage channels and its applications. In *Proc. IEEE Conference on Security and Privacy*, pages 108–121, 1988.

[vAH04] L. v. Ahn and N. Hopper. Public-key steganography. In *Proc. Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 323–341. Springer, 2004.

[WAM09] C. Wonnemann, R. Accorsi, and G. Müller. On information flow forensics in business application scenarios. In *Proc. IEEE COMPSAC Workshop on Security, Trust, and Privacy for Software Applications*, pages 324–328, 2009.

[WBZ10] O. P. Waldhorst, R. Bless, and M. Zitterbart. Overlay-Netze als Innovationsmotor im Internet. Spontane virtuelle Netze: auf dem Weg zum Internet der Zukunft. *Informatik Spektrum*, 33(2):171–185, 2010. (In German).

[Wen07a] S. Wendzel. PHCCT, 2007. http://www.wendzel.de/dr.org/files/Projects/phcct/, retrieved: January 2013.

[Wen07b] S. Wendzel. Protocol hopping covert channels, 2007. http://www.wendzel.de/dr.org/files/Papers/protocolhopping.txt, retrieved: January 2013.

[Wen08a] S. Wendzel. Protocol channels as a new design alternative of covert channels. *CoRR*, abs/0809.1949:1–2, 2008.

[Wen08b] S. Wendzel. Protocol hopping covert channels. *Hakin9*, 08(03):20–21, 2008. (in German).

[Wen09a] S. Wendzel. PCT, 2009. http://www.wendzel.de/dr.org/files/Projects/pct/, retrieved: January 2013.

[Wen09b] S. Wendzel. Protokollwechsel zur Realisierung von Covert Channels und Header-Strukturveränderungen zur Vermeidung von Covert Channels. diploma thesis, Kempten University of Applied Sciences, 2009. (in German).

[Wen12] S. Wendzel. The problem of traffic normalization within a covert channel's network environment learning phase. In *Proc. Sicherheit'12*, volume P-195 of *LNI*, pages 149–161, 2012.

[WK11] S. Wendzel and J. Keller. Low-attention forwarding for mobile network covert channels. In *Proc. Communications and Multimedia Security*, volume 7025 of *LNCS*, pages 122–133. Springer, 2011.

[WK12a] S. Wendzel and J. Keller. Design and implementation of an active warden addressing protocol switching covert channels. In *Proc. 7th International Conference on Internet Monitoring and Protection (ICIMP 2012)*, pages 1–6. IARIA, 2012.

[WK12b] S. Wendzel and J. Keller. Systematic engineering of control protocols for covert channels. In *Proc. 13th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS 2012)*, volume 7394 of *LNCS*, pages 131–144. Springer, 2012.

[WKR12] S. Wendzel, B. Kahler, and T. Rist. Covert channels and their prevention in building automation protocols – a prototype exemplified using BACnet. In

Proc. 2nd Workshop on Security of Systems and Software Resiliency, pages 731–736. IEEE, 2012.

[Wol89] M. Wolf. Covert channels in LAN protocols. In *Proc. Local Area Network Security*, volume 396 of *LNCS*, pages 89–101. Springer, 1989.

[Wra91] J. C. Wray. An analysis of covert timing channels. In *Proc. 1991 Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society, 1991.

[WRKM13] S. Wendzel, T. Rist, B. Kahler, and M. Masoodian. Countering covert and side channels to enhance privacy in building automation systems, 2013. (unpublished).

[WS97] A.C.W. Wong and A.T.P. So. Building automation in the 21st century. In *Proc. 4th International Conference on Advances in Power System Control, Operation, Management (APSCOM-97)*, pages 819–824, November 1997.

[WVD+06] A. Wood, G. Virone, T. Doan, et al. ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring. Technical Report CS-2006-13, Department of Computer Science/University of Virginia, 2006.

[WZ12] S. Wendzel and S. Zander. Detecting protocol switching covert channels. In *Proc. 37th IEEE Conference on Local Computer Networks (LCN)*, pages 280–283, 2012.

[XM04] N. Xie and N. R. Mead. SQUARE project: Cost/benefit analysis framework for information security improvement projects in small companies. Technical Report CMU/SEI-2004-TN-045, Software Engineering Institute, Carnegie Mellon University, November 2004.

[YDL+08] F. V. Yarochkin, S.-Y. Dai, C.-H. Lin, et al. Towards adaptive covert communication system. In *Proc. 14th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2008)*, pages 153–159. IEEE Computer Society, 2008.

[YDL+09] F.V. Yarochkin, S.-Y. Dai, C.-H. Lin, et al. Introducing P2P architecture in adaptive covert communication system. In *Proc. 1st Asian Himalayas International Conference on Internet (AH-ICI 2009)*, pages 1–7, 2009.

[ZAB07a] S. Zander, G. Armitage, and P. Branch. Covert channels and countermeasures in computer network protocols. *IEEE Comm. Magazine*, 45(12):136–142, December 2007.

[ZAB07b] S. Zander, G. J. Armitage, and P. Branch. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys and Tutorials*, 9:44–57, 2007.

[Zan10] S. Zander. *Performance of Selected Noisy Covert Channels and Their Countermeasures in IP Networks*. PhD thesis, Centre for Advanced Internet Architectures, Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, May 2010.

[Zda04] S. Zdancewic. Challenges for Information-flow Security. In *Proc. 1st International Workshop on the Programming Language Interference and Dependence (PLID'04)*, 2004. http://www.cis.upenn.edu/~stevez/papers/Zda04.pdf, retrieved: January 2013.

[ZMU+12] M. N. Zawawi, R. Mahmod, N. I. Udzir, et al. Active warden as the main hindrance for steganography information retrieval. In *Proc. 2012 International Conference on Information Retrieval and Knowledge Management*, pages 277–280, 2012.

Index