Internet Firewalls in the DECOS System-on-a-Chip Architecture

Armin Wasicek and Wilfried Elmenreich

***Abstract*— A big part of requests in today's Internet are malicious connection attempts aimed at compromising hosts in order to gain illegal access. Intrusion tools perform automatic scans to seek out promising targets, probe for vulnerabilities, and even mount autonomous attacks. Outgoing from this scenario, this paper discusses approaches to govern access to a network of System-on-a-Chip (SOC) components that provides an Ethernet interface to the Internet for maintenance purposes. Security measures are needed to protect the SOC from unauthorized access to internal information such as diagnostic interfaces or bus communication. Since the SoC should be realized as a compact embedded system, the implementation of security mechanisms has to fit the available processing and memory resources. In order to be able to cope with changing security requirements and different deployment environments a multi-level security architecture is proposed. The architecture partitions the system into intrusion containment regions and provides corresponding access privileges. As part of the architecture, the implementation of an Internet Firewall providing low level authentication to a network of SOC s is shown.**

***Index Terms*— Embedded systems security, firewall, DECOS SOC architecture, Time-Triggered Ethernet.**

I. INTRODUCTION

CONCURRENT distributed systems contain a great amount of information flowing among the participating nodes. Besides the installed hardware and software, this information represents the distributed system's asset. People may deliberately look out for faults and make usage of them to get unauthorized access to this information, thus breaking the system's security policy. A *security policy* partitions the system state into secure (authorized) states and non secure (unauthorized) states. It consists of a set of rules describing *who* may access *what* in which *manner*. In this regard *who* may be substituted by a person or a software agent, *what* through a resource, and the *manner* are the allowed operation modes, thus describing user's privileges in the system. An *attack* is to intentionally seek out operation modes without a permission. A fault being used to cause a security failure is called a *vulnerability*. An attack successfully exploiting a vulnerability is an *intrusion*. An intrusion has a direct impact on a system's security properties, e. g., disclosure of sensitive information, introduction of tampered messages, or denial of service. This terminology is given after [1].

An example scenarios for threats in embedded systems is sketched by Koopman in [2], wherein a thermostat realized as embedded system is connected to the Internet enabling remote control of a building's heating and cooling. He discusses possible threats which result from malicious manipulations like turning off the system in winter to freeze water pipes or exposing pets to the heat of summer. The scenario is extended to an attacker controlling a great deal of thermostats in the same area who may try to trigger a peak electricity demand by simultaneously switching on all air conditioners to cause a denial–of–service in the provision of electricity.

A *System-on-a-Chip (SOC)* integrates the electronics for a complete, working product on a single chip. This technique assembles a product's parts, which were formerly fabricated from various chips and circuit boards, in one silicon die. The benefits from this miniaturization range from higher performance to higher system reliability and lower consumer costs. Consider for example interconnection versus wiring: on-chip interconnections are much cheaper than wires, they have a very high maximum throughput, and it is likely that they will not break until severe damage occurs to the whole chip. An SOC comprises several cores interconnected by a *Network-on-a-Chip (NOC, [3])*. The cores are dedicated to specific tasks, which together perform the system's application.

Connectivity and remote usage of services carried out by embedded systems establish a new level of ubiquitous computing. The drawbacks of these efforts are that failure propagation among similar devices can cause massive security breaches, which can turn into threats to safety. Since embedded system failures can have catastrophic consequences on the users and the environment, a failure in the security domain must not propagate in the safety domain. This requires security to be an early part of a product's design phase.

In this paper we propose a multi-level security architecture for a set of networked SOC components, thus acting as *"system of systems"*. The architecture defines several levels of trust and various intrinsic security requirements. We want to show the application of the concept of an *intrusion containment region* [1] to partition a system and how to figure out different security requirements for each region. The next step taken is the selection of some appropriate security measures for implementation.

The remainder of this paper is structured as follows: The next section gives an overview of firewalls and gateways as a means of restricting access to (sub)networks. Then a short introduction to the DECOS SOC architecture is presented and, in a further section, both concepts are integrated in a multi-level security architecture for a system of systems. Section VI presents a case study of an Internet firewall implementation as perimeter firewall for the SOC architecture. At last, a conclusion is drawn and an outlook on further development in this area is given.

II. FIREWALLS

Kopetz gives in [4] a definition for a gateway: *"The purpose of a gateway is to exchange relative views between two interacting clusters. In most cases, only a small subset of the*

Manuscript submitted January 25, 2007; This work has been supported in part by the European IST project DECOS (IST-511764). It is part of the DECOS SOC implementation realized at the Real-Time Systems Group, Vienna University of Technology.

The authors are with the Institute for Computer Engineering, Vienna University of Technology, Vienna, Austria. (e-mail: armin@vmars.tuwien.ac.at)

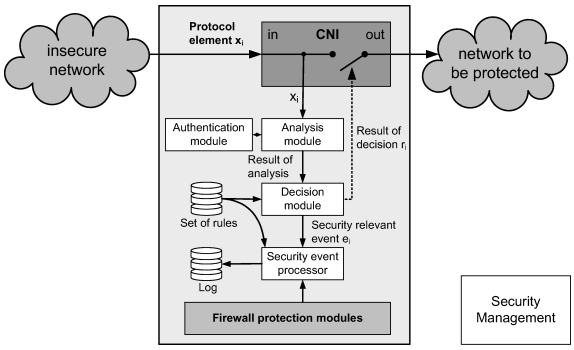

Fig. 1. A possible structure of a firewall element connecting an insecure and a secure network

information in one cluster is relevant to the other cluster." Hence a gateway is concerned with the message exchange between two networks.

A firewall acts as a special gateway extending a normal gateway for communication control features. It can allow or deny inbound or outbound communication requests and thus govern access to a network. A policy defines how access is granted among users, which usually consists of several rules. Message filtering has to be performed according to these rules.

Since an access policy can change as time goes by, it is necessary to adjust the firewall rules. Changes can be triggered for example by adding a new node or a new service to the cluster, or through the occurrence of a certain system event. An interface must be provided to reconfigure the working rule set efficiently, secure, and without message loss.

The basic architecture of a firewall implementation is depicted in Figure 1. A protocol element x_i arriving from an untrusted network is tapped and forwarded to an analysis module, which checks for validity or defrags several coherent elements. Based on a set of rules forming the security policy a decision is taken, either to accept the protocol element in the protected network or to trigger a security relevant event.

Basically, there are two types of firewalls:

- *Perimeter firewalls* [5] connect physically decoupled networks. They are often referred to as *multi-homed gateway* or *screening router*, which is a node that has multiple network interfaces and forwards messages from one network to the others. In most application scenarios, one network is considered as trusted, whereas the others are untrusted. A firewall augments a gateway with a message filter, which is controlled by a set of rules, the security policy. Because of its widespread deployment, people talking about a 'firewall' usually have this type in mind.
- *Distributed firewalls* [5], [6] rest on three notions: (a) a security policy, (b) data replication tools[1] to distribute policies, and (c) IPSec[2] [7] to establish a secured virtual network. Through the last notion each host's identity is cryptographically assured. Thus the addresses used in the protocol, which are transmitted in clear text and can be faked, become irrelevant for filtering decisions. A security policy defining access rules is stipulated in an appropriate policy language, distributed over the secure network, and executed on each host's local firewall.

The second approach is more sophisticated and has some advantages over the first one. Obviously, it eliminates the firewall node as single point of failure, which is a major benefit for both availability and performance. A firewall failure cannot isolate an entire network and throughput is no longer limited by the performance of the firewall. Another feature is that hosts, which are not within a topological boundary, can be protected. These advantages are bought by a more elaborate setup. An overview on firewalls can be found in [8].

Firewalls present an efficient tool to control access to a network, but they also have some restrictions, one has to to be aware of following threats when deploying a firewall [9], [10]:

- *Insiders* are a major threat. A firewall can prevent users from outside the network accessing it and users from the inside to leak information, but it can do nothing about attacks carried out from inside the network. This is only partly true for a distributed solution.
- *Hidden channels* are communication channels from and to the network which are not monitored by the firewall. Therefore, they are not part of the security concept.
- *Restricted design:* A firewall can only avert threats for which it is designed. It can be vulnerable to new modes of attack.
- *Malicious logic* (e. g., viruses, worms) delivered within the message body cannot be recognized by simple firewalls, because most of the filtering is based on communication protocol elements.
- *Tunneling* is encapsulating one communication protocol inside another one. This technique is used to transport a network protocol through a network which would otherwise not support it, or to provide various types of a virtual private network (VPN) functionality. It can also be used to bypass a firewall system. In this case, firewall-blocked data is encapsulated inside a commonly allowed protocol. Again, this is only partly true for a distributed solution.

III. Authentication

An *authentication procedure* establishes a relationship with a certain degree of trust between two principals. Usually authentication is used to confirm a principal's identity and, in a next step, to transfer some privileges to the authenticated principal. Credentials are an important piece of information to prove a principal's authenticity.

Widely used authentication methods include Kerberos [11], SSL/TLS [12], SOCKS [13], and SSTCP [14] as known as *port knocking* [15].

We focus on these client–server based network authentication protocols, which work in a challenge–response fashion. To request authentication for some service, a challenger submits his credentials, which are (eventually cross) checked, and, if valid, the authentication procedure has succeeded.

Table I summarizes some important features of the above mentioned methods. Column *authentication* describes who is authenticated, *cryptosystem* lists the used cryptographic class of algorithm. A *trusted third party* includes some mutually trusted principal like a certification authority (CA), which

[1] E. g., UNIX's `rsync` command, or Microsoft's *Server Management System*

[2] A suite of protocols for authenticating or encrypting IP packets

TABLE I
AUTHENTICATION METHOD OVERVIEW

	authentication	**cryptosystem**	**trusted 3^{rd} party**	**OSI layer**	**computation**	**minimal messages**	**memory footprint**
Kerberos (GSSAPI)	mutual	symmetric	yes	application	high	10	292kb
SSL/TLS handshake	optionally mutual	asymmetric	eventually CA	transport	high (server–side)	7+	50-60kb
SOCKSv5	client to server	plaintext	no	transport	low	6	68kb
SSTCP	client to server	plaintext, one–time pad	no	network	low	1	256kb

is used to establish a trusted relationship. The OSI *layer* indicates at which level in the TCP/IP protocol suite the method is implemented. *Computation* make an estimation about the required computational ressources based on the cryptographic algorithms used. The *minimal messages* column lists the minimal number of messages exchanged for a successful connection request. At last the *memory* column approximates memory usage footprints of sample implementation, just to give a rough impression. The measurements were conducted partly by ourselves, partly they origin from the literature [16], [17].

The table shows that authentication procedures using strong cryptography are computationally very intensive and more complex than simplistic ones like SOCKS and SSTCP. An interesting option for lightweight solutions is to use one–time pads which on the one hand offer strong encryption and protection against replay attacks, but on the other hand need ressources for storing and maintaining keys. Nevertheless, there are efforts to implement a SSL/TLS enabled embedded web server [16] with minimal memory usage and computational requirements. With the advent of elliptic curve cryptography (ECC), which is much more ressource efficient than traditional RSA ciphers, strong assymmetric cryptography will find its way in embedded systems.

IV. DECOS SOC COMPONENT MODEL

Embedded systems pose restrictions on dependability, cost, real–time performance, power consumption, and physical security [18], [19]. In the Dependable Embedded Components and Systems (DECOS, [20]) Project, a concept for composable systems and components was developed and applied to the SOC paradigm in [21]. This lead to a design methodology for embedded SOC components.

The DECOS System-on-a-Chip (SOC) component model defines a *SOC cluster* as a collection of micro components interconnected over a dedicated Network-on-a-Chip (NOC). The SOC component consists of arbitrary micro components, each representing an intellectual property (IP) block connected through a standardized linking interface to the NOC , and dedicated micro components managing the NOC , namely the Trusted Network Authority (TNA) and the Resource Management Authority (RMA). One node in the network holds a connection to the internal network – the NOC – as well as to an outer network, e. g., a Time–Triggered Ethernet or an Event–Triggered network like the Internet. This special node in the network works as gateway and is capable to enable or disable information flow from and to the SOC .

This design of a generic SOC component model with a set of *highly autonomous* and *possibly heterogeneous* micro components that are *interlinked by a time–triggered conflict–free on–chip network* [21] is depicted in Figure 2.

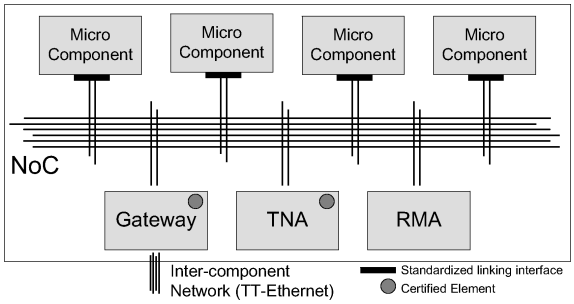

Fig. 2. DECOS SOC component model

V. SECURITY REQUIREMENTS FOR A SOC CLUSTER

Embedded systems face several design challenges and security requirements as surveyed in [22] and [23]. When designing a secure embedded system, restrictions on cost, real–time performance, power consumption, and physical security have to be met. Furthermore, embedded systems are often deployed in safety–critical areas, where the consequences of a malfunction from a compromised embedded system can result in severe physical harm or even death of its users and third parties. In the past security for embedded systems has been mostly neglected, besides some niche applications like electronic immobilizers (e. g., ignition locks). Reasons are that there was no or little incentive for malicious manipulations and that computer security has not been a first order citizen during the design phase.

For *maintenance* purposes, a mechanism to perform secure updates on software executed in a micro component is mandatory. This is due to overcome "chip tuning" attacks which install unsecure software that might operate the controlled system out of its specification [24]. The *protection of sensitive data* must be assured, because often sensible user data is stored in an embedded device, which may not be disclosed e. g., after loss or theft of the device. The *protection of intellectual property* requires that implementation details must not be accessible to others than producer and maintainer. Embedded systems are often used to *support business models* like prepayment meters, where a secret code has to be entered to unlock some service, e. g., gas, electricity, or telephone credits [25]. Many embedded devices implement functions to monitor and control *legal obligations*, e. g., the digital tachograph [26].

Against this background of requirements and constraints we suggest a security concept implementing a multi-level security

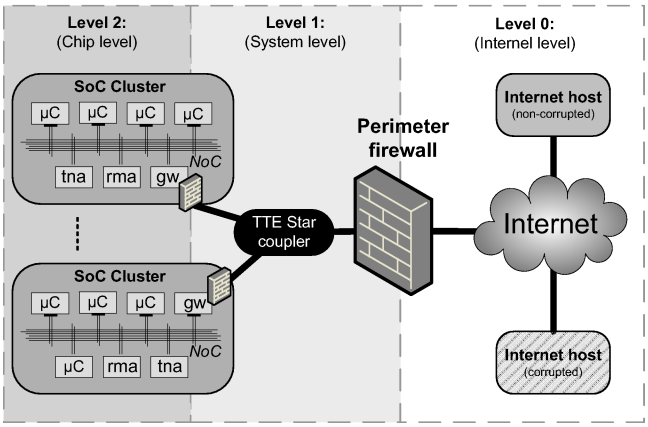

Fig. 3. Multi-level Security architecture

architecture for a system integrating several SoC clusters with Internet connectivity. Our example architecture in Figure 3 depicts a couple of SoCs connected over a Time-Triggered Ethernet (TTE, [27]). Internally, the micro components in a DECOS SoC are interconnected over a dedicated NoC, which operates time-triggered as well. If the SoC supports external clock synchronization over the gateway, a *global time base for the whole "system of systems"* can be established and the single SoCs are synchronized over TTE.

A global time base allows the temporal coordination of the distributed system nodes, which has several implications like timestamps being also meaningful outside the micro component where an event has occurred, thus exchanging real-time observations between several SoCs can be facilitated. From a security point of view, this makes the global time base one of the most important assets to protect.

Clock synchronisation algorithms used in distributed computer systems are able to detect and handle many failure modes like e. g., message syntax failures, fail–silent failures, byzantine failures, babbling idiot failures, masquerading failures, and SoS (Slightly–Off–Specification) failures [28]. However, with external clock synchronisation the synchronisation messages in a time–triggered system can be used to maliciously cause different drift rates in different clusters, which might lead to their desynchronisation and force them to give up to deliver a common service. Thus, a corruption of the global time base threatens a system's availability,

In embedded systems design, the concept of a *fault containment region* (FCR, [29]) is to *"prevent the effects of faults in one region from interfering with another region"*. The concept of an *intrusion containment region* (ICR, [1]) is defined in analogy to a FCR and is aimed at *"guaranteeing certain security properties, despite the fact that some components of the systems might be compromised"*. In our architecture we facilitate a system partitioning in ICRs. An *access point* in this terminology is a potential communication partner, a host, or a node in the network. A cryptographic boundary [30] represents a similar concept, but stresses the presence of some cryptographic means within a single boundary.

In our multi-level security architecture in Figure 3, the ICRs are structured into three levels with increasing levels of trust. This structure corresponds to the well-known military *defense–in–depth* strategy, which states that even if the attacker has passed one defense, there should be another one ready to stop him.

- In *level 0 (Internet level)*, the "system of systems" contains one Internet host and thus one ICR. The corruption of an access point is not important to the rest of the system. At this level, the participating nodes work independently, the whole network is regarded as untrusted. Corrupt access points have to be distinguished from non-corrupt access points by authentification.
- In *level 1 (System level)* single SoCs are connected over a TTE with an Internet firewall/gateway. This network is one ICR. The nodes exchange synchronisation messages for the global time base as well as diagnostic information and application dependent information. A corrupt node could lead to disclosure of sensible information or to corruption of the global time base. Besides from a denial-of-service caused from the latter, every connected SoC node continues to work properly, if one node has been compromised. If one node at this level is corrupt, the whole ICR is corrupt.
- *Level 2 (Chip level)* comprises the individual DECOS SoCs. One SoC is a ICR. Possible attack targets are access to diagnostic interfaces, execution of instrumentation code, direct manipulation of real–time entities, and disclosure of IP related information. A successful intrusion at this level is a serious security violation, which may result e. g., in catastrophic consequences to the environment (depending on the entities under control) or finacial loss for the manufacturer (IP–theft).

To satisfy the requirements of the different ICRs, appropriate security measures have to be found. Implementing security is risk management. The costs of the measures must be seen in contrast to the assets.

The lowest expected cost for anyone to discover and exploit a vulnerability in a system is called the *cost to break* (CTB). The concept of this method has its origin in cryptography and reflects the cost/benefit proportion of an attacker seeking financial gain. It would be uneconomic to spend more money than the system's asset expected value to break into system [31]. The CTB represents a practical method to judge a measure's worth and is also mentioned in [4]. By cost we mean not only financial charges to develop and deploy the measures, but also system resources needed to provide the security services. This is particularly important in the context of embedded systems, where only limited resources are available.

VI. PERIMETER FIREWALL TO THE INTERNET

The concept of a firewall has been found most suitable to facilitate the implementation of an ICRs as discussed in the previous section. From a system designer's point of view, a firewall is *not a single device or a group of devices, but the implementation of a security policy* [32]. We use this notion to define a transition between two ICRs as *authenticated firewall traversal*. Hence, a firewall grants or denies access to a connecting node based on his credentials. We built a prototype

firewall for a DECOS SOC cluster using SSTCP (port knocking) as authentication procedure.

On the border between level 0 and level 1 resides a perimeter firewall, which was implemented for evaluation purposes. This node holds the sole connection to the Internet. Every request from and to the system passes this node, which makes it particularly suitable for a perimeter defense. It's aim is to thwart simple attacks like port and vulnerability scans, which are used to collect information on potential target hosts on the Internet. To continue with the example from the introduction, a possible scan could be: "Find as many thermostats running a software version with a known vulnerability." Such connection attempts without proper authentication are blocked by the perimeter firewall leaving the inner network free of unwanted connection attempts. In case the number of connection attempts increases and results in a denial-of-service attack, only the perimeter host will be affected and the SOC nodes providing the system's core services continue to work properly.

The implementation was done on a Soekris Engineering net4801 board running a Linux kernel 2.6.9, which is part of the DECOS SOC development system. The port knocking server was implemented by a set of netfilter/iptables rules using the `ipt_recent` and `ipt_state` modules. Entering a valid code through a couple of pre–defined port knocks unlocks access over SSL/TLS [12] to a web interface listening available services associated with the entered code. This enables a simple user privileges system. To counter replay attacks, the codes can be encrypted using one–time pads, which make the code only usably once.

In online banking systems, these two mechanisms are known as the PIN code used for authentication and the TAN code for transaction processing. The SSL/TLS handshake requires the connecting host to have a valid X.509 certificate [33] and thus provides a second layer security mechanism. After selecting a service the firewall accepts and forwards a connection from the client to the service providing SOC component over the inner TTE. The whole process is depicted as state chart diagram in Figure 4.

The rather high memory usage results from using the high–level language of netfilter/iptables rules. An implementation done in a low–level language will yield a significantly smaller memory footprint.

Maintenance procedures on the firewall are performed by modifying the current ruleset. Additionally, new services can be registered with the secure web server.

The SOC gateways between level 1 and level 2 can be realized as single perimeter firewalls or united to a distributed firewall. The latter is a more complicated, but seems to be a more promising approach. This is a topic of current research.

Our main objective when implementing the prototype was to evaluate the usefulness of lightweight authentication techniques allowing thin clients. We found the concept of port knocking (SSTCP), which alienates a protocol element and uses it during the authentication process to transmit some credentials, very interesting. This approach features an easy implementation, reuse of existing networking code, and interoperability with every device implementing the communication protocol, which saves in return ressources. By the use of appropriate cryptographic tools, the transmission can be secured.

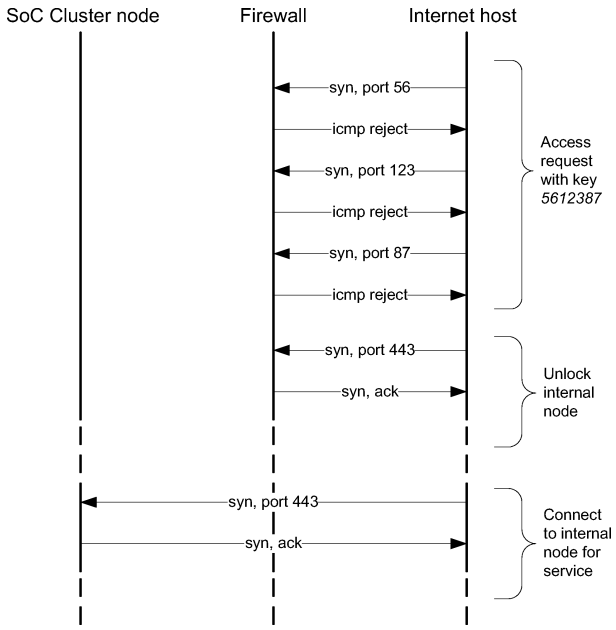

Fig. 4. State chart for Internet firewall

VII. CONCLUSION

The proposed multi-level security architecture partitions the system into intrusion detection regions. Due to different security requirements for each region, manageable access controls must be implemented. Two types of firewalls are at hand to realize this system partitioning at the implementation level. Perimeter firewalls govern access from external networks, distributed firewalls enforce a security policy among inner network nodes.

High-level authentication services that offer asymmetric key exchange and use an established web of trust from public key infrastructures (PKI) will prevail in the embedded systems domain. But whenever public key infrastructures are not available or not reasonable to be deployed, lightweight techniques to provide a good service at low cost are required. Moreover, embedded systems have limited resources, high level techniques implementing access controls using e. g., strong cryptography are often not available. Low-level techniques have to be used with two goals in mind (a) to save resources and (b) to support compatibility within a wide range of interacting applications.

We have presented a case study implementation of a perimeter firewall that implements the security boundary between internet level and system level using port knocking as a lightweight security technique. Future work will include security concepts for NOC gateways between system level and chip level.

ACKNOWLEDGMENTS

The authors thank Hermann Kopetz for the opportunity to report on this topic, Christian El Salloum, Bernhard Huber, and Roman Obermaisser for the discussions on the practical part and Klaus Steinhammer for his advice. We also thank Kathrin Steiner for her support.

REFERENCES

[1] D. Powell and R. Stroud, "Conceptual model and architecture of MAFTIA," MAFTIA deliverable D21, January 2003, http://www.maftia.org.

[2] P. Koopman, "Embedded systems security," *IEEE Computer*, vol. 37, no. 7, pp. 95–97, July 2004.

[3] M. Schöberl, "A time-triggered network-on-chip," Technische Universität Wien, Institut für Technische Informatik, Research Report 114/2006, 2006.

[4] H. Kopetz, *Design Principles for Distributed Embedded Systems*, 4th ed. Kluwer Academic Publishers, 1997.

[5] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin, *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison Wesley, 2004.

[6] S. M. Bellovin, "Distributed firewalls," *;login: The usenix magazine*, pp. 37–39, November 1999, special Issue on Security.

[7] S. Kent and R. Atkinson, "Security architecture for the internet protocol," RFC 2401, Internet Engineering Task Force, November 1998, http://www.rfc-editor.org/rfc/rfc2401.txt.

[8] A. Wasicek, "Security considerations in embedded systems," Institute of Computer Engineering, Vienna University of Technology, Tech. Rep. 108/2006, 2006.

[9] L. K. Balakrishnan, S. Krishnamurthy, V. K. Kumarasamy, and L. C. Pothula, "Firewalls," November 2001, http://www.utdallas.edu/ sxk010620/Firewalls/firewalls.pdf.

[10] S. M. Bellovin and W. R. Cheswick, "Network firewalls," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 50–57, 1994.

[11] J. Kohl and B. C. Neuman, "The kerberos network authentication service (V5)," RFC 1510, Internet Engineering Task Force, September 1993, http://www.rfc-editor.org/rfc/rfc1510.txt.

[12] T. Dierks and C. Allen, "The TLS protocol," RFC 2246, Internet Engineering Task Force, January 1999, http://www.rfc-editor.org/rfc/rfc2246.txt.

[13] M. Leech, M. Ganis, Y.-D. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS protocol version 5," RFC 1928, Internet Engineering Task Force, March 1996, http://www.rfc-editor.org/rfc/rfc1928.txt.

[14] P. Barham, S. Hand, R. Isaacs, P. Jardetzky, R. Mortier, and T. Roscoe, "Techniques for lightweight concealment and authentication in IP networks," Intel Research Berkeley, Tech. Rep. IRB-TR-02-009, July 2002.

[15] M. Krzywinski, "Howto: Port knocking," *Linux Journal online*, June 2003, http://www.linuxjournal.com/article/6811.

[16] A. Steffen, "Secure communications in distributed embedded systems," in *Proceedings of the 2nd Mechatronic Systems International Conference*, October 2002, p. 111 ff.

[17] A. Fox and S. D. Gribble, "Security on the move: Indirect authentication using kerberos," in *Proceedings of the ACM Mobile Computing and Networking Conference*, 1996, pp. 155–164.

[18] D. Dzung, M. Naedele, T. P. von Hoff, and M. Crevatin, "Security for industrial communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, June 2005.

[19] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, no. 3, pp. 461–491, 2004.

[20] R. Obermaisser, P. Peti, B. Huber, and C. E. Salloum, "DECOS: An integrated time-triggered architecture," *e&i journal (journal of the Austrian professional institution for electrical and information engineering)*, vol. 3, pp. 83–95, March 2006.

[21] R. Obermaisser, "DECOS system-on-a-chip component specification," DECOS Deliverable 2.2.4, 2006, http://www.decos.at.

[22] P. C. Kocher, R. B. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proceedings of the 41th Design Automation Conference, DAC*. ACM, June 2004, pp. 753–760.

[23] C. Paar, "Embedded IT security in automotive application – an emerging area," in *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, K. Lemke, C. Paar, and M. Wolf, Eds. Springer-Verlag New York, Inc., 2006, pp. 1–13.

[24] W. Stephan, S. Richter, and M. Müller, "Aspects of secure vehicle software flashing," in *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, K. Lemke, C. Paar, and M. Wolf, Eds. Springer-Verlag New York, Inc., 2006, pp. 17–26.

[25] R. Anderson, *Security Engineering*, 1st ed. John Wiley and sons, Inc., 2001.

[26] I. Furgel and K. Lamke, "A review of the digital tachograph system," in *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*, K. Lemke, C. Paar, and M. Wolf, Eds. Springer-Verlag New York, Inc., 2006, pp. 69–94.

[27] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (TTE) design," in *Proceedings of the 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC)*, May 2005.

[28] A. Ademaj, A. Hanzlik, and H. Kopetz, "Tolerating arbitrary failures in a master–slave clock–rate correction mechanism for time–triggered fault–tolerant distributed systems with atomic broadcast," *15th International Conference on Real–Time and Network Systems*, March 2007.

[29] H. Kopetz, "On the fault hypothesis for a safety–critical real–time system," in *Proceedings of the Workshop on Future Generation Software Architectures in the Automotive Domain (ASWSD)*, ser. Lecture Notes in Computer Science, vol. 4147. Springer, January 2004.

[30] F. P. 140-2, "Security requirements for cryptographic modules," National Institute of Standards and Technology, 2002.

[31] S. Schechter, "Quantitatively differentiating system security," *The First Workshop on Economics and Information Security*, 2002.

[32] P. Srisuresh and K. B. Egevang, "Traditional ip network address translator (traditional NAT)," RFC 3022, Internet Engineering Task Force, January 2001, http://www.rfc-editor.org/rfc/rfc3022.txt.

[33] R. Housley, W. Ford, T. Polk, and D. Solo, "Internet X.509 Public key infrastructure – Certificate and CRL profile," RFC 2459, Internet Engineering Task Force, January 1999, http://www.rfc-editor.org/rfc/rfc2459.txt.

Armin Wasicek is a research assistant at the Institute of Computer Engineering, Real-Time Systems Group at the Vienna University of Technology. He has studied Computer Science at TU Vienna and received the Dipl.-Ing. (Master's) degree in 2006. During his course of study Armin Wasicek visited the SUPAERO in Toulouse, France and the University of Otago, Dunedin, New Zealand, each for one semester as exchange and graduate student in Computer Science. His research interests include Real-Time Systems, the Time-Triggered Architecture, and Security. Currently, Armin Wasicek works on a PhD about security in embedded systems with Professor Hermann Kopetz as his research advisor.

Wilfried Elmenreich is an assistant professor at the Institute of Computer Engineering at Vienna University of Technology. He studied at the Engineering School for Electrotechnics and Control in Weiz, Styria and graduated at the Vienna University of Technology in Austria. He received a Master's degree in computer science in 1998 and a doctoral degree in technical sciences in 2002. His doctoral thesis addressed the sensor fusion problem in time-triggered systems. Wilfried Elmenreich has contributed significantly to the development of the TTP/A fieldbus protocol and the standardization of the OMG Smart Transducer Interface Standard. In the last five years, Wilfried Elmenreich has published over 40 papers in the field of embedded real-time systems.