Journal of ELECTRICAL ENGINEERING, VOL. 61, NO. 3, 2010, 183–188

TRANSPARENT PROXY FOR SECURE E–MAIL

Juraj Michalák — Ladislav Hudec *

The paper deals with the security of e-mail messages and e-mail server implementation by means of a transparent SMTP proxy. The security features include encryption and signing of transported messages. The goal is to design and implement a software proxy for secure e-mail including its monitoring, administration, encryption and signing keys administration. In particular, we focus on automatic public key on-the-fly encryption and signing of e-mail messages according to S/MIME standard by means of an embedded computer system whose function can be briefly described as a brouter with transparent SMTP proxy.

K e y w o r d s: security of e-mail messages, encryption, signing, software proxy, S/MIME standard, transparent SMTP proxy

1 INTRODUCTION

Nowadays we live in the world where an enormous number of common people with only pseudo-understanding of technical details are fairishly using personal computers and the Internet in a large extent. It was a difficult task for IT personnel to achieve this goal. If we lived in an utopian world, there would be no danger of using all those new technologies. But empathy is an exception rather than a rule. Therefore, security has been developing all the time. Security as the result of humanity or bestiality? Security is a source of the complicacy for common users. There are two basic types of security settings, simplified (mainly for common users) and professional security settings, which have an impact on the quality of security. Finally the acquired security depends also on the end- user interaction. Solutions without end-user interaction like IPsec[1], VPN[2] and TLS/SSL[3] are preferred because of common users who do not like to interact too much and they are the origin of a considerable number of security incidents too.

Solutions mentioned above provide security for the transfer between communication endpoints and are insufficient for such a type of communication like SMTP (Simple Mail Transfer Protocol [6]), where messages travel from the node, where message was created to the destination node potentially via several nodes. S/MIME[4] is an appropriate standard for secure electronic mail messaging [10, 4]. In this paper, S/MIME is employed to provide automatic secure electronic mail messaging between organization boundaries without a common user interaction needed and without necessity of extensive configuration changes to existing infrastructure. The proposed embedded computer system was implemented in GNU/Linux environment with the use of a linux bridge, ebtables, netfilter, iptables, advanced routing and experimental transparent proxy support in linux kernel to achieve brouter with transparent SMTP proxy functionality (see 4.2). In order to get the S/MIME and X.509[5] [2] certification functionality, cryptlib [5] was chosen as the basic cryptographic library.

2 EMBEDDED COMPUTER SYSTEM

Alix system board from PC Engines as embedded computer system was chosen. It contains 5x86 compatible CPU and runs Voyage Linux operating system (Debian derivative) which has been developed directly for this platform. This environment is very similar to standard GNU/Linux PC.

2.1 Technical details

The Alix system board is equipped with [7]:

- 500 MHz AMD Geode LX800 CPU
- 256 MB DDR DRAM
- Storage: CompactFlash socket
- Power: DC jack or passive POE, min. 7 V to max. 20 V, peak power consumption is about 6 W without miniPCI cards and USB devices

[1] IPsec (Internet Protocol Security) is suite of protocols for securing IP (Internet Protocols) communication in network layer of the OSI (Open Systems Interconnection) reference model.

[2] VPN (Virtual Private Network) has many applications. One common is authentication and content encryption.

[3] TLS (Transport Layer Security)/SSL (Secure Socket Layer) are cryptographic protocols for security and integrity of communication over TCP/IP.

[4] S/MIME (Secure/Multipurpose Internet Mail Extensions) is standard for public key encryption and signing of message encapsulated in MIME.

[5] X.509 is an ITU-T standard for PKI(Public Key Infrastructure).

* Slovak University of Technology, Faculty of Informatics and Information Technologies, Ilkovičova 3, 842 16 Bratislava, Slovakia; juraj.michalak@gmail.com, lhudec@fiit.stuba.sk

ISSN 1335-3632

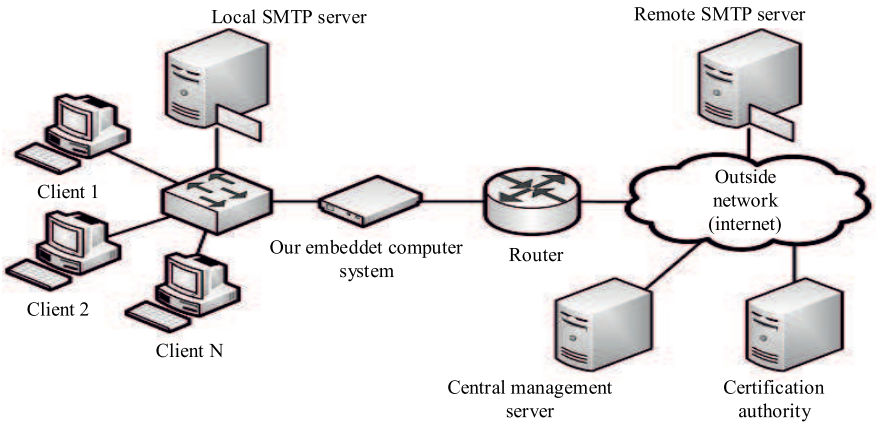

Fig. 1. Network environment overview

- Three front panel LEDs, one pushbutton
- Expansion: 1 miniPCI slot, LPC bus
- Connectivity: 3x 100Mbit Ethernet channels (Via VT6105M 10/100)
- I/O: DB9 serial port, dual USB 2.0 port
- Board size: 6×6 (152.4×152.4 mm)

3 DESIGNED SOFTWARE OVERVIEW

The presented embedded computer system provides automatic security for the messages transported between organization boundaries. More accurately, it provides a transparent SMTP proxy [13] for communication between the local and remote SMTP server and according to the designed logic it is able to encrypt and sign or decrypt and sign verify the authenticity of the transported messages by S/MIME standard. The benefit of this transparent solution is that there is no need to change the existing local network configuration.

The transparent proxy with a secure e-mail application is represented by the local process on the embedded system. SMTP communication passing over the embedded system has to be redirected into TCP/IP stack[6] of Linux kernel in order to be delivered to that local process and the rest of communication is bridged[7]. Therefore, the embedded system is considered a brouter[8]. The local process performs on-the-fly processing of SMTP communication because it has to be transparent, provide proxy functionality and also the storage resources of embedded system are limited. Two email cryptographic standards were analyzed, OpenPGP and S/MIME [10]. Only the latter was adopted because the electronic signature is placed after the signed data in its message format, which is inevitable for on-the-fly signing (signature can be computed after all of message data are known [10]).

As S/MIME was adopted, PKI based on X.509 is used for certificate creation, registration, verification, revocation and other handling. CMP protocol is adopted for certificate registration and embedded system acts as registration authority to the Certification Authority. The central management server is designed for certificate distribution in the community[9] of our embedded systems but also a local administration web interface could be used to manage the local certificate database of one embedded system. At the moment certificates distribution between our embedded systems is done manually and administration via a configuration file (accessible via SSH).

4 DETAILS OF CONCEPT

In the previous section we have described the basic concept of our solution. More details are mentioned in this section.

4.1 Experimental transparent proxy support

The original title of this feature is Tproxy [1] and it allows us to:

- Redirect sessions destined to the outer network to a local process using a packet filter rule.
- Make it possible for a process to listen to connections on a foreign address.
- Make it possible for a process to initiate a connection with a foreign address as a source.

4.2 Brouter with transparent SMTP proxy

Two ethernet interfaces (eth0, eth1) of the embedded system compose a virtual bridge interface (see bridge in Fig. 2) which is implemented by the Linux bridge [12]. All communication passing our system is bridged except SMTP, which is redirected to a kernel routing process.

[6] TCP/IP stack or Internet protocol Suite is set of communication protocols and may be viewed as set of layers.

[7] Network bridge connects multiple network segments at data link layer according to the IEEE 802.1D standard.

[8] Brouter or Bridge Router works as network bridge and as router at the same time.

[9] Community of embedded systems is set of embedded systems which are intended to secure the email messages transferred with one another.

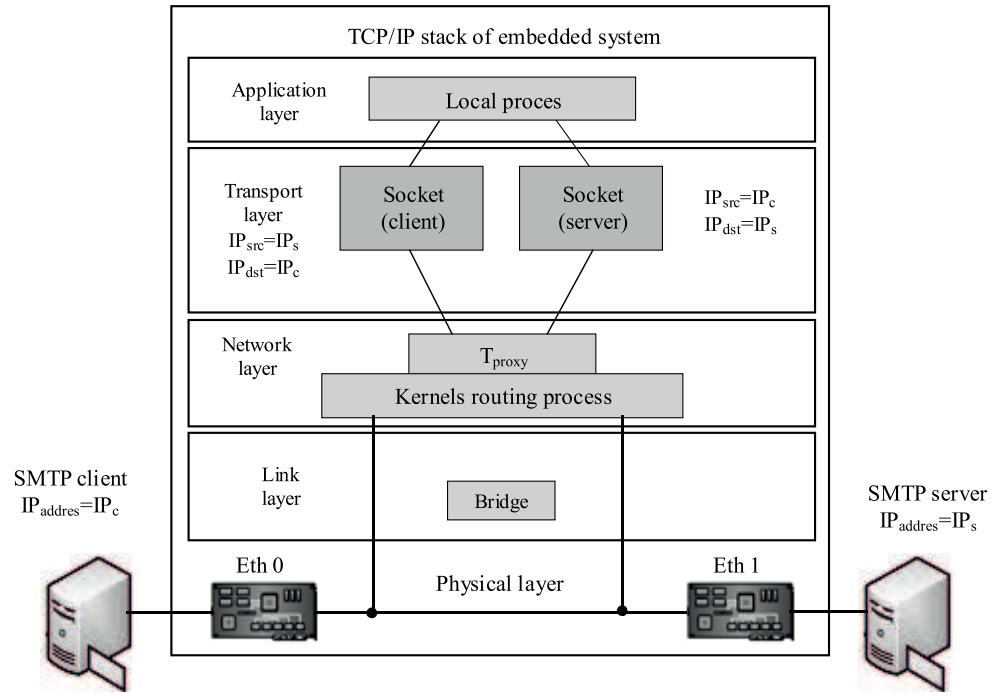

Fig. 2. Detailed view of brouter with transparent proxy

This is implemented by ebtables [3, 9] as follows. The destination MAC address of the incoming SMTP frames (on eth0 and eth1 interfaces) is replaced by a virtual bridge MAC address. So the frames are destined to the virtual bridge interface and the kernel is on assumption that it is acting as a router.

The SMTP communication is then in the kernel routing process redirected to our local process which finally establishes connection with the SMTP client (initiator of connection) and server. Both are assuming that they are communicating directly. This is achieved by an experimental transparent proxy support in the kernel, iptables and policy routing [1]. More precisely the packets related to SMTP communication are marked, on the basis of this marks are routed by a separate routing table and delivered locally. The local process is able to get the IP address of the original destination server via the standard socket API and to create connection to the original destination server with client IP address as source address. This process is illustrated in Fig. 2.

Finally, the local process provides an application proxy with the mentioned automatic on-the-fly encryption and signing as security service for SMTP.

Two modes of bridge interface are designed:

1. *IP mode*. Bridge interface has an assigned local IP address (statically or dynamically by the local DHCP[10] server). So the embedded system is reachable from the local network.
2. *Fully transparent mode*. The bridge interface has a special IP address assigned (only for internal usage, to have functional kernel routing process) and has disabled the ARP[11] protocol. ARP table is managed by a self implemented ARP manager tool which inspects SMTP packets in order to insert address mapping records into ARP table. This mode of operation is unconventional but provides a fully transparent and plug-and-play solution with possible administration through the central management server or locally via a serial console.

4.3 On-the-fly S/MIME

Our local application must understand SMTP [6] and MIME[12] [4] in order to collect enough information to make message encryption or decryption decision. For S/MIME enveloping and signing we have used cryptlib cryptographic library [5]. The output of the used cryptlib routines is in the form of a binary DER[13] encoded CMS[14] object, which has to be encapsulated into MIME entity in order to transfer the secured message via SMTP.

[10] DHCP (Dynamic Host Configuration Protocol) is network application protocol used by devices in order to obtain configuration for its operation in an Internet Protocol network environment.

[11] ARP (Address Resolution Protocol) is protocol for building table which is used for translation of network layer address into data link layer address.

[12] MIME (Multipurpose Internet Mail Extensions) is Internet standard for messages.

[13] DER (Distinguished Encoding Rules) is message transfer syntax standard by the ITU. DER is subset of BER (Basic Encoding Rules) providing exactly one way to encode an ASN.1 value, what is must in cryptography.

[14] CMS (Cryptographic Message Syntax) is the IETF standard for cryptographic protected messages. It is based on PKCS#7.

4.3.1 S/MIME enveloping

S/MIME e-mail message securing is based on encapsulation. The process of signing and encrypting the message is composed of subsequent steps. Firstly the message is signed and the result is CMS object which is encapsulated into MIME entity. That MIME entity is then encrypted and again encapsulated into MIME entity.

For example we look at S/MIME encryption process [10]:

1. Generate pseudo-random session key for symmetric encryption algorithm.
2. For each recipient, encrypt the session key with his public key.
3. For each recipient, prepare a block RecipientInfo that contains an identifier of the recipients public key certificate, an identifier of the session key encryption algorithm, and the encrypted session key.
4. Encrypt the MIME entity content with the session key — the CMS object was created which consists of RecipientInfo block and encrypted MIME entity.
5. Create new MIME entity from that CMS object which has to be encoded into textual form (*eg* by Base64 encoding).

4.3.2 S/MIME de-enveloping

De-enveloping is a counter operation to enveloping. Theoretically the process of encapsulation in S/MIME enveloping has no constrained depth. In our application we have implemented on-the-fly de-enveloping with a kind of automatic abstract data structure (Autodeenvelope), which can handle theoretically an unlimited S/MIME encapsulation depth. Autodeenvelope works in the following manner. It analyzes the MIME header of message to identify S/MIME encrypted or signed message. Then if needed or possible the message is decrypted or the signature is checked. This process is repeatedly applied to the message inside Autodeenvelope in order to get the original/plain form of the message. It means that the input is the S/MIME secured message and the output is a plain message.

4.4 Message securing logic

First we divide the community of users into two sets called local and remote users. Local users own e-mail accounts on the local SMTP server and by contrast remote users own e-mail accounts on remote SMTP servers. Every embedded system has its own private key and certificate (key pair) for common security service. Its certificate (public key) has to be distributed to other systems and then the incoming messages, encrypted with its public key, are decrypted with its private key. Additionally for the advanced security, users also can have their own private key and certificate on the embedded system.

The basic premise is that our application on the embedded system has available a local private key database of local users and certificate database of remote embedded systems, local users and remote users. The latter one contains certificates of all remote embedded systems and users for which the messages will be encrypted (also as premise). These databases are managed locally by SSH access, by web interface or in an automatic fashion by the central management server. Messages can be signed only with the use of the embedded systems private key because message authenticity is always considered with regard to the embedded system. The embedded system signs messages not users workstation. This is called domain signing seeing that our embedded system is assigned to domain.

Basic rules for message securing:

- Encrypt outbound message if the recipients certificate (or only remote embedded systems certificate) is available. Then sign the message with private key of sender or embedded system.
- If recipients certificate is not available, outbound message is not encrypted. Optionally sign the message with use of senders private key, but create detached S/MIME signature because it is probable that recipient has no S/MIME capable tool. This can lead to expansion of S/MIME because recipient can be interested by presence of that digital signature in received message.
- Inbound encrypted message is decrypted if the corresponding private key is available.
- If inbound message is signed, the signature is checked with use of attached certificate or certificate chain. Then that certificate (or certificate chain) has to be checked if it can be trusted. It is trusted if it is already in local database or its trust can be checked via its CA certificate if it is in local database. Else the attached certificate is stored into database of untrusted certificates for later trust check.

The question is what to do with messages for multiple mixed recipients who are recipients with and without certificate in local database. There are two policies for this situation, the first one is restrictive where message is encrypted and sent only to recipients with available certificate. Second is liberal, message is handled as with the first policy and in addition it is sent unaltered to recipients without certificate in local database.

Problem is that message sending has to be divided in encrypted message sending and unaltered message sending. It has two solutions:

- Second concurrent connection to original destination SMTP server is created for unaltered message delivery. Some SMTP servers prohibit multiple concurrent connections from one source, what means that this solution is not usable every time.
- Second concurrent connection to dedicated SMTP Relay server is created for unaltered message delivery. That server then delivers unaltered message to original destination SMTP server. Our embedded system attempts to apply the first solution and the result of this effort is stored into database for given destination

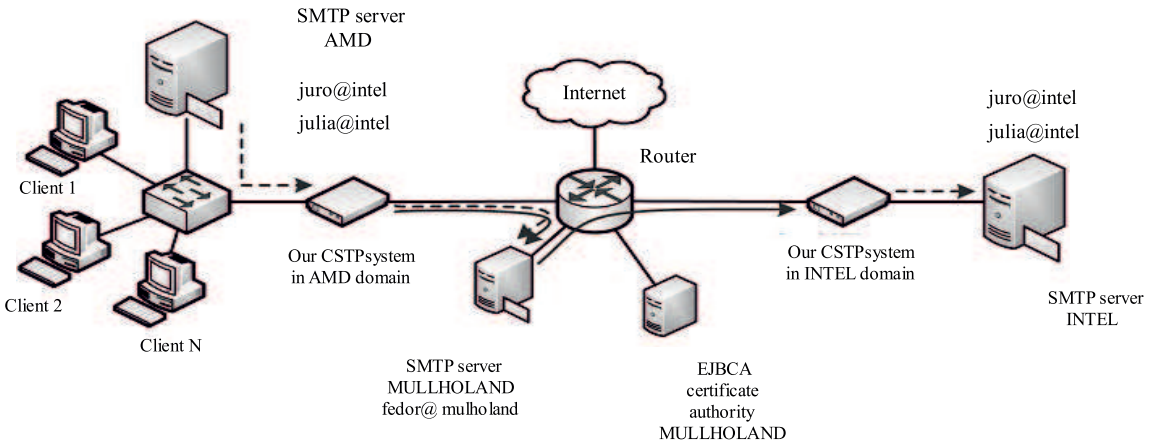

Fig. 3. Development and test environment – network topology

SMTP server. If result is negative second solution is used.

4.5 Example

Our embedded system is called as CSTP (Cryptographic SMTP Transparent Proxy) system. Test environment (Fig. 3) is virtual (VMWare Workstation) except one real CSTP system in Amd domain. Amd SMTP server is configured to forward all outgoing messages via Mulholland SMTP server.

First example of usage is the most complicated one. It is processing of message for multiple mixed recipients. In Amd domain we send message from juro@amd to juro@intel, julia@intel and fedor@mulholland. The SMTP server Amd starts the transfer of that message to the Mulholland server. The CSTP system in Amd domain has certificate of the Intel CSTP system and "julia@intel in its local database, so the message for Intel domain users is going to be encrypted. The Amd CSTP system creates second connection to the Mulholland SMTP server for unaltered message tranfer to fedor@mulholland. Then the message is signed and encrypted. Fedor receives original message, which could be optionally signed by the Amd CSTP system. The Mulholland SMTP server forwards the signed and encrypted version of message to the Intel SMTP server. The Intel CSTP system uses its private key for decryption of forwarded message and checks its signature. Juro and Julia receive original form of message.

Second and last example is similar but message is sent only to julia@intel who is specific user because her private key is not in the Intel CSTP system database (it is stored on her own workstation). Message is signed and encrypted by the Amd CSTP system. The Intel CSTP system is not possible to decrypt message — lack of Julia's private key. So Julia receives secured S/MIME message and it is decrypted by her e- mail client application. Also the signature is checked, but with warning that message was signed by cstpbox@amd and not by its sender juro@amd.

4.6 Concurrent server design

Our embedded system has to handle many hundreds or thousands of concurrent connections between SMTP clients and server. Nowadays the huge volume of spam is also considerable. So there is a need to have good concurrent server design.

Thread pool design pattern was used in form of prethreaded server with main thread connection accept [11].

5 CONCLUSION

Our embedded computer system provides confidentiality, integrity and authenticity for electronic messages transferred between organization boundaries according to S/MIME standard. Network transparency and minimal requirements for changes in the existing infrastructure and configuration are its worthwhile features. For its administration, a serial console and local SSH access can be used (in the future also web interface and central management server). The central management server is also responsible for automatic distribution of certificates in the community of embedded systems. At this time it is done manually by SSH. At least one certification authority is required because of PKI and optionally one SMTP relay server is required because of the mentioned solution for delivery of messages with mixed recipients.

The design of some application parts is modular which affords the opportunity to easily implement for example a new cryptographic module with the use of another cryptographic library (*eg* openssl).

We see a large field for expansion of our solution. Itsgeat potential is in the early SPAM detection combined with tar-pitting (slow down of suspected connections) and in protection against DoS and DDoS via hybrid network architecture of our application – event-based approach combined with thread-based and with the use of connection multiplexing.

Acknowledgement

This paper was worked out in the frame of a Diploma thesis Embedded systems for encryption of application protocols and in the frame of the VEGA project No. 1/0649/09 entitled Security and reliability in distributed computer systems and mobile computer networks.

References

[1] BALAZS, S.: Introduction to TProxy and its Features, [Online; accessed December 2nd, 2008]. Available at: http://www.balabit.com/downloads/files/tproxy/README.txt.

[2] COOPER, D.—SANTESSON, S.—FARRELL, S. *et al*: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, [Online]. May 2008. RFC (Request for Comments) series 5280. Available at: http://tools.ietf.org/html/rfc5280.

[3] Ebtables: Ethernet Bridge Frame Table Administration. [Online; accessed December 2nd, 2008]. May 2007. Available at: http://ebtables.sourceforge.net/ebtables-man.html.

[4] FREED, N.—BORENSTEIN, N.: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. [Online], November 1996. RFC (Request for Comments) series 2045. Available at: http://tools.ietf.org/html/rfc2045.

[5] GUTMANN, P.: Cryptlib Security Toolkit [Online]. Version 3.3.2, July 2008. Available at: ftp://ftp.franken.de/pub/crypt/cryptlib/manual.pdf.

[6] KLENSIN, J.: Simple Mail Transfer Protocol. [Online], October 2008. RFC (Request for Comments) series 5321. Available at: http://tools.ietf.org/html/rfc5321.

[7] PC Engines GmbH: Alix.2 / Alix.3 / Alix.6 series system boards. [Online; accessed February 16th, 2009]. Available at: http://www.pcengines.ch/pdf/alix2.pdf.

[8] RAMSDELL, S.: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. [Online], July 2004. RFC (Request for Comments) series 3851. Available at: http://tools.ietf.org/html/rfc3851.

[9] SCHUYMER, B.—FEDCHICK, N.: Ebtables/Iptables interaction on a Linux-based Bridge, [Online; accessed December 2nd, 2008]. Available at: http://ebtables.sourceforge.net/br_fw_ia/br_fw_ia.pdf.

[10] STALLINGS, W.: Cryptography and Network Security Principles and Practices, 4th ed. Prentice Hall, 2005..

[11] STEVENS, W. R.—FENNER, B.—RUDOFF, A. M.: UNIX Network Programming: The Sockets Networking API, 3rd ed. Vol. 1, Chapter 30. Client/Server Design Alternatives, Addison Wesley, 2003 -13-141155-1..

[12] The Linux Foundation: Net: Bridge. [Online; accessed December 2nd, 2008]. Available at: http://www.linuxfoundation.org/en/Net:Bridge.

[13] Transparent SMTP proxy. In Wikipedia: the free encyclopedia [Online]. St. Petersburg (Florida): Wikimedia Foundation, 2001-, last modified on 24 January 2009. Available at: http://en.wikipedia.org/wiki/Transparent_SMTP_proxy.

Received 24 September 2009

Juraj Michalák (Ing) was born on June 12, 1985 in Trnava. Currently he is software developer at the Innovatrics s.r.o., Bratislava. In 2007 he received Bc diploma with magna cum laude in computer engineering from Faculty of Informatics and Information Technologies, Slovak Technical University and was awarded by the Dean's price for his Bachelor project. In 2009 he received Ing. diploma at the same faculty and his Master's thesis was awarded by the Dean's price.

Ladislav Hudec (doc, Ing, CSc), currently Associate Professor of Computer Science and Engineering and Director in charge of the Institute of Applied Informatics, Faculty of Informatics and Information Technology, Slovak Technical University. In 1974 he received Ing diploma with summa cum laude in electronics from Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University, Prague, in 1985 he received CSc degree (PhD) in Computer Machinery from the Faculty of Electrical Engineering, Slovak Technical University, Bratislava, in 1989 he was appointed Associate Professor. He is author or co-author over 40 scientific papers published in journals and proceedings of the conferences and over 50 technical papers in the field of fault tolerant computing, embedded systems, parallel computing and computer security. He led over 20 research grants and industrial projects. He reads lectures on Computer Architecture, Computer Security and Security in Internet. Dr Hudec is member of the IEEE, he is member and Vice-President of the Slovak Association for Information Security (SASIB), member of the Information System Audit and Control Association (ISACA) and the Heraldic club of Slovakia. During 1993-2010 he served as Slovak national coordinator at the European Cooperation in Science and Technology (COST).