Videoconference Security

LUBOMIR CVRK, MARTIN SYKORA, VACLAV ZEMAN
Department of Telecommunications
Faculty of Electrical Engineering and Communication, Brno University of Technology
Purkynova 118, 612 00 Brno
CZECH REPUBLIC

Abstract: - This article deals with the problem of security application on the H.323 videoconferencing. The H.323 communication uses many protocols and messages that are sent by the TCP and the UDP protocols. The data are provided by H.323 terminals and several control applications needed for videoconference administration. Some of these data transmissions need to be protected by encryption. In this article we show how to do this in a very easy way, which is independent of any of the participating H.323 terminals and other control applications. This independence guarantees easy deployment of this approach on any existing videoconference system.

Key-Words: - videoconference, security, H.323, encryption, AES, integer counter mode, STNP, STEP

1 H.323 videoconferencing

Recommendation H.323 [1] provides the ground for audio, video and data communication over networks based on the IP protocol. Recommendation H.323 provides the ability to interconnect audio/video conferencing software and hardware products and tools from various companies. H.323 covers the connection control, the standards for coding, the management of transfer capacity, and it also defines the interface between LAN and other networks. The architecture of the H.323-based network is shown in Fig. 1.

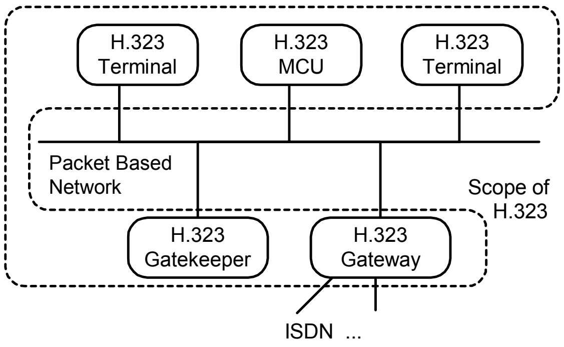

Fig. 1. Architecture of the H.323 based network

2 Design of a videoconferencing system

In the design of the videoconferencing system the following initial requirements were determined:

- user-friendly control,
- capability of recording and storing the sessions,
- security of the data transmissions using encryption techniques.

It follows from the requirements that the emphasis is laid on the simplicity of the control of videoconferencing terminals (VCF) and on the ability to store the communication. Subsequent processing of the session records should also be possible. Fig. 2 specifies the data flows among the components of the conference.

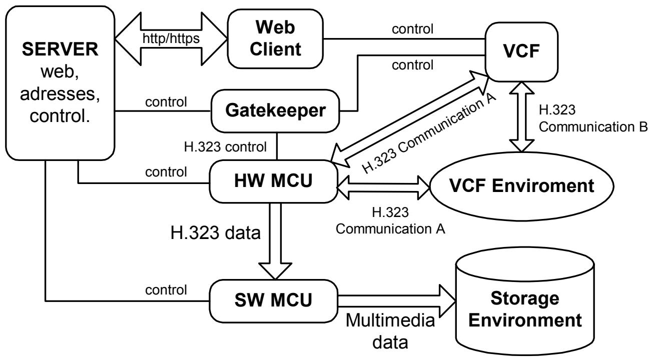

Fig. 2. Data-flows in the videoconferencing system

The main part of the system is the control server. It contains the components for controlling other parts of the system, and also the interface for communicating with the client application (web client) over the HTTP protocol. Part of the server is the database, where the addresses of the videoconferencing terminals and their aliases are stored. Every user can define their own aliases for other participants registered in the system. That is why there is no need to know their IP addresses.

The web client allows a direct control of the videoconferencing terminal through the module which is implemented in the control terminal. This module is different for any type of the VCF terminal.

Another part of the system is the Gatekeeper where all the videoconferencing terminals are registered in order to be able to communicate according to the H.323 standards. The Gatekeeper communicates with the server over the proprietary application protocol based on XML.

The main task of the Gatekeeper is to translate the aliases to the appropriate IP addresses.

The recording of the sessions requires the Multi Conference Unit (MCU) hardware and the MCU software to be connected to the system. The MCU software is extended by the capability of recording and storing audio/video data. When recording session is requested, the data pass the MCU hardware (H.323 Commnunication A [1]) and are forwarded to the MCU software, which stores them.

3 Security concept

As shown above, the videoconferencing system produces a lot of data transmissions. All of them need to be secured with respect to the fact that they are heterogeneous. A proof of the heterogeneity is that the data transmissions are realized with confirmative service of the transmission control protocol (TCP) and non-confirmative service of the UDP protocol. The security mechanisms must be applied to all the data transmissions realized by these services.

3.1 Common concepts of communication protection

In general there are several ways how to protect the communication by these protocols. One is the virtual private network (VPN), implemented by IPsec protocols. It cannot be used because the installation and configuration of IPsec for VPN is one of its weaknesses. It requires a well informed administrator to set up correctly all the security rules and conditions of IPsec. Another problem of IPsec is its static configuration.

The next possibility consists in using the SSL/TLS-based VPNs. This also involves a problem, because they are statically configured. The configuration is simpler than that of IPsec, but it is assumed that the other side of H.323 communication is not known at the time of configuring the SSL VPN on the user machine. If somebody just wants to communicate safely with somebody else, the security must work automatically without any need for reconfiguring on the client's side. The current implementation requires establishing a VPN tunnel first.

Securing H.323 communication based on the H.235 ITU-T standard would look like the most suitable solution, but it is the most complicated task. It would require recompiling all participating client software. Updating the clients by their producers could (in particular in the case of black boxes) be expensive and is completely out of anybody‘s control.

Opportunistic encryption as implemented in FreeS/WAN [6] requires DNSSEC [7] to be in full production or it requires access to the reverse-DNS records in order to add a TXT field for publishing the public key. Nobody knows how long DNSSEC will take to be fully accessible in view of the DNSSEC standards being currently redesigned. The FreeS/WAN project has been stopped, and its contributors discontinued the respective research. From the point of view of our goals, opportunistic encryption is the most suitable solution but it is still too complicated for common users whot do not know anything about the DNS system, and, moreover, it works only among the routers, so the "last step" on the LAN remains insecure.

As we can see there is no completely suitable and simple enough solution for this problem right now.

3.2 Independent security concept

The design of an independent security concept is based on the fact that the opposite end of communication may not support this concept of security. That is why at the beginning of the communication the initiator must ask the responder whether it supports it. This will be done by the Simple Tunnelling Negotiation Protocol (see 3.2.1). If it does not support it, the encryption of data for a remote peer must not start and must be transported with no modification (which means no security).

3.2.1 Simple Tunnelling Negotiation Protocol

Simple Tunnelling Negotiation Protocol operates over TCP. Its service monitors on port 4077. The protocol format is shown in Fig. 3.

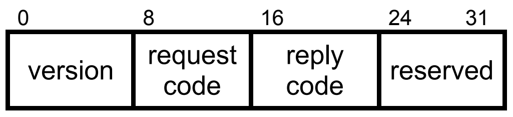

Fig.3 STNP message format

The field version tells the receiver's STNP daemon (negotiation service) the version of STNP protocol of the sender. Right now it is 1.

The field request code is set (>0) when the sender needs to request something from the other side. This field can acquire value 1 – which means "Do you support Client independent security?" Value 2 means "Is the tunnel established OK?"

The field reply code is set (>0) whenever the sender of the message replies to some previously received request message. The reply code can contain 1 – "Yes, I support. Negotiate encryption." Value 2 means "Tunnel established OK!" Value 255 means "Unrecognized request" (see 2.3).

Opening connection: STNP messages are sent over TCP port 4077 to the responder. If the responder implements this security concept (i.e. also STNP) it must reply with the STNP message, where it sets its version of STNP. The request code must remain as set by the initiator (1) and the reply code is set to 1. Otherwise the connection is closed. If the port is not open or the connection is closed, it means that encryption cannot be set.

Closing connection: When the sender wants to stop encrypting, they set the appropriate flag in the Simple Tunnelling Encryption Protocol, see 3.2.5.
Packets with other settings must be discarded.
The IP address of the opposite end of communication is given to the negotiation service by the API kernel-to-user space, which communicates with the kernel level packet capturer.

Subsequent connection establishment: Let us have two nodes – A and B. A supports this concept of security and B does not. When A detects packets for B and initiates the STNP session, it is unsuccessful. Communication between them will then be insecure. If B later runs CIS, then (in the event the packet is directed to A) it asks A to open an encrypted communication channel. The secured communication will immediately negotiate and start.

3.2.2 Authentication and key-negotiation

Authentication and key-negotiation are the initial part of every secured communication and will be performed by the negotiation service based on the ISAKMP protocol [3]. This approach supports two ways of authentication and key-negotiation: (1) using public-key infrastructure (recommended), (2) using PGP certificate database (optional). There is no need to discuss these methods here, they have been fairly described and are well known.

After the key has been negotiated, the STNP must verify that the tunnel is established OK.

3.2.3 STNP service (daemon)

The STNP daemon must have a user interface available that is to inform a user about the IP addresses that communicate safely and about the addresses where the key-negotiation is running. From this interface the daemon must be able to shut down the CIS. This interface also interacts with the user in the matter of certificate acceptance.

3.2.4 Encryption process

The encryption process will be driven by the AES algorithm with a 128 bits long key, working in the integer counter mode, because it will process a stream of data of unpredictable length.

In general, the encryption process counts packet losses or reorder events so that none of them depends on any other packet. Each packet must be encrypted / decrypted separately.

For a correct control of this process the same scheme will be used as designed in the SRTP protocol [4].

3.2.5 Simple Tunnelling Encryption Protocol

This protocol is used for the modification of captured IP packets that have to be encrypted with respect to the authentication of each packet and the possibility of packets being lost. STEP modifies the IP packet according to the scheme in Fig.4.

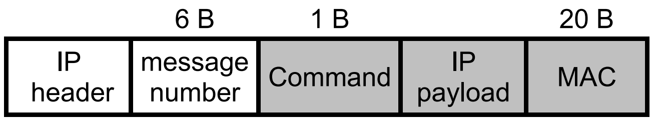

Fig.4 STEP in the context of IP protocol, grey means encrypted fields

When the IP packet has been processed, STEP encapsulates the IP payload between its overhead fields. The process computes the MAC and adds it after the IP pay-load. Then it runs the encryption function and encrypts the string made of these three fields (command, IP payload, MAC).

The message number contains a 48bit value representing the number of the message being sent to one destination (determined by the destination IP address). The command field controls the communication process. It is used to stop the encryption. Whenever the source node needs to stop the encryption, it sets the command value to 1. After the destination has decrypted and authenticated the IP packet and accepted the command to be 1, it stops packet processing from that source IP address. All subsequent packets are expected to be common IP packets so they are passed to the upper layers unmodified.

All IP packets encapsulated by STEP have in the protocol field of IP header the value 99 (IANA's "any private encryption scheme" [5]).

3.3 Communication basis and packet processing

The core of this approach is the IP packet capture at the operating system kernel level. The driver itself is dependent on the operating system so the implementation requires an internal API, which eliminates the platform-dependent amount of source code needed for packet processing implementation.
The whole system works in the following steps:

1. The STNP daemon and kernel driver are started and the driver captures the incoming and outgoing IP packets.
2. In the event of the first outgoing packet to destination IP address A, the driver tells the STNP daemon the destination IP address.
3. The STNP daemon checks whether the destination supports this concept of security. If it does, then it negotiates the keys. While checking and negotiating, the packets with destination address A pass the kernel driver unencrypted. After the STNP session has finished the

daemon tells the driver the negotiated key associated with the IP address.

4. Once the driver has a key associated with the IP address, it starts the encryption – the tunnel appears to be established.

5. In the event of the first packet with protocol number 99 from the opposite side of the previous key-negotiating communication the driver switches to the decryption / encryption state and encrypts and decrypts all traffic between the two nodes (authentication within).

6. For tunnel verification, the STNP daemon on the initiator side sends the STNP request "Is tunnel established OK?" and waits for the reply, which must be "Tunnel established OK!" since both messages will be encrypted and decrypted only if all is OK. If the daemon does not receive the STNP message "Tunnel established OK!" within 10 seconds, it tells the driver to switch off the encryption into IP address A and resets the protocol state. For clarity, Fig. 5 shows the entire system simplified.

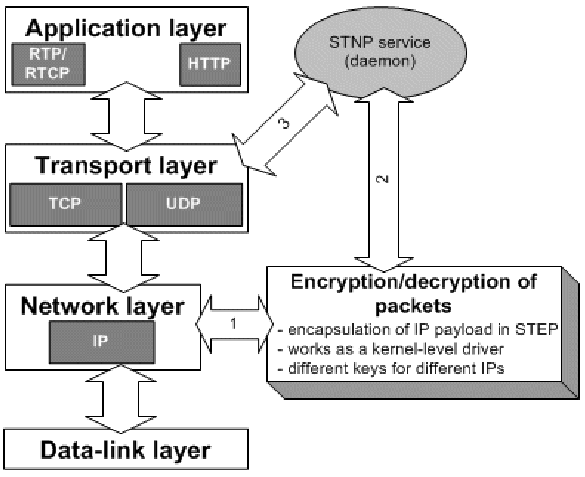
Application layer
RTP/ RTCP
HTTP
STNP service (daemon)
3
Transport layer
TCP
UDP
2
Network layer
IP
1
Encryption/decryption of packets
- encapsulation of IP payload in STEP
- works as a kernel-level driver
- different keys for different IPs
Data-link layer

Fig. 5. Schema of the security approach

The whole traffic between the nodes of H.323 communication is secured. The numbers inside the arrows relate to the numbers in the paragraph above.

4 Conclusion

This work concerns an application of security to videoconference data transmissions. It defines the conditions for a simple application of security and confronts them with available security solutions such as IPsec, SSL/TLS, H.235 based security, etc. None of the current security approaches fits the conditions completely and therefore a new approach is introduced.

The approach is based on packet capture and its encryption with authentication. For a correct functionality the approach defines two communication protocols, which implement the mechanisms of secured tunnel establishment, data encryption and encryption process control. The security system is application independent, so no H.323 client software recompilation is needed for security deployment. The deployment is very easy and does not need any configuration. It just needs installing on a machine and a public key cryptography certificate. The tunnel establishment and encryption process are controlled automatically.

It works in networks with MCU, in peer-to-peer mode and it works with network address translation (NAT) if the NAT box runs the CIS system. It does not work with multi-cast yet but this is a topic for further study.

References:

[1] ITU-T, Packetbased multimedia communications systems, *Recommendation H.323,* ITU-T, 1999.

[2] OPEN H323 PROJECT, http://www.openh323.org.

[3] Maughan, D., Shertler, M., Schneider, M., Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", *RFC 2408*, 1998.

[4] Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K., "The Secure Real-time Transport Protocol (SRTP)", *RFC 3711*, 2004.

[5] Internet Assigned Numbers Authority, http://www.iana.org/assignments/protocol-numbers.

[6] FreeS/WAN project, http://www.freeswan.org.

[7] Dierks, T., Allen, C., "The TLS Protocol Version 1.0", *RFC 2246*, 1999.

[8] OpenVPN project, http://openvpn.sourceforge.net

[9] DNSSEC project web page, http://www.dnssec.net

[10] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed- Hashing for Message Authentication", *RFC 2104*, February 1997.

[11] Fergusson, N., Schneier, B., *Practical Cryptography*, Wiley Publishing, Inc., Indianopolis USA, 2003.

[12] Chraibi, C., Web Services Deployment and Security, In *WSEAS Transactions on Communications*, Issue 4, Volume 2, October 2003 ISSN1109-2742, pp.482-488.