Independent Policy Oriented Layering of Security Services

Herbert Leitold, Peter Lipp, Andreas Sterbenz

IAIK, Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
Herbert.Leitold@iaik.at, Peter.Lipp@iaik.at, Andreas.Sterbenz@iaik.at

Key words: security, policy, S/Mime, SSL

Abstract: Implementing a security policy has to cope with the diversity of communication requirements and applications. We present a policy oriented approach from the observation of common problems and characteristics given in networked applications. The solution reduces the trust required into the security system to a single entity. This is done in an application independent manner by fooling the applications and feigning a conventional, insecure networking environment that is further transformed to a secure communication infrastructure.

1. INTRODUCTION

Security often is an add on, which is a must for commercial applications and for privacy purposes and has to fit into existing structures. The paper discusses aspects where this fitting into existing structures is scrutinized.

In this context, the overall problem is shown using a very simple example: Electronic signatures [1] will shape communication where it becomes commercially and legally relevant in a specific way. The most common approach followed by the software industry at the moment can be summarized as follows:

- Digital certificates [2] are used to prove identities and attributes
- Secure multipurpose internet mail extension (S/MIME) [3] is the tool of the first choice with deferred communication
- Secure socket layer (SSL) [4] is used mostly with online communication.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35515-3_53

S. Qing et al. (eds.), *Information Security for Global Information Infrastructures*

Besides these approaches which focus on the application layer entities, there exist approaches for other purposes like virtual private networks (VPN) [5] and low level security. Internet protocol security (IPSEC) [6] and features of IP version 6 (IPv6) [7] are just some of the examples in the TCP/IP communication scenarios.

Now lets assume that we have a security enhanced client using either authenticity or confidentiality or both. Newer versions of Web browsers and email clients offer such enhancements. This raises some questions:

- Is it advisable to integrate security with the application? A virus or so may well help avoiding security features without user awareness?
- What if a presentation bug exist in the application? How can this be serviced and maintained without interfering with security policies?
- Will the application use security options if they are optional?

Consider for example active components that modify the real time behavior of browsers, such as ActiveX-components or Javascript [9]. Such active features can launch many malicious attacks. Examples are frame hijacking [10], reading the browser's cache [11] or transmitting malicious code via email to vulnerable clients [12].

In general terms, existing solutions require a single piece of software to be trusted in all dimensions not only "security-wise" but also in the presentation of the content and in the handling. On the other hand there are numerous faults reported on such clients in addition any exploits. Such a fault can influence presentation of the content significantly. Exchangeability of presentation elements as well as exchangeability of security modules is an essential element of trust.

An operational aspect to be considered is, that binding security to a specific application leads to the situation that all relevant applications have to be replaced by security enhanced applications. Given the diversity of applications building the operational environment of an organization, enforcing a security policy becomes a delicate matter: Whereas one application's built in security modules might fit the organization's policy choice, another won't or a specific application important for the organization does not support security mechanisms at all. An approach where a dedicated security layer manages the organizations security requirements is desirable, as this de-couples the security management from application add-ons. Separating the security layer also leads to the situation that existing applications are allowed to survive.

In this paper we follow such an approach of an dedicated security layer and present a set of solutions that meet the requirement to implement security in a policy oriented manner. Therefore, the paper is structured, as follows: In section 2 we survey common aspects in networking enabled operating systems. Basing on the statements made, we describe S/MIME

mapping as an application aware approach in section 3. We broaden the application space observed in section 4 by discussing SOCKS 5 [13] as a tool to enable application independence. By identifying some problems encountered we continue our considerations in section 5 where we describe how to fool the application's access to the communication system. This enables the integration of policy awareness which is discussed in section 6. In section 7 we combine the consideration made by describing videoconferencing as an application example. Finally, conclusions are drawn.

2. TCP/IP INTEGRATION IN COMPUTING

In this section, we address some characteristics when applying a security policy in networked operating systems. The focus is on the transmission control protocol, internet protocol (TCP/IP) stack, i.e., the protocol family employed by the Internet.

Networking represents an integral part of today's operating systems. In particular integration of the Internet by employing the TCP/IP protocol stack in both the operating systems and the applications has numerous benefits for the user and represents a trend towards promoting network computing. Consequently, the application space and the networking space is no longer to be considered as separated dimensions, it is rather a simultaneous usage of both.

Awareness about security concerns is given by the user, as well as by the software industry. Although efforts to integrate security in the operating system kernel space are made, such as implementing IPSEC, security is in the general case built into the application, not into communication. Well-known examples are S/MIME enabled mail clients, or browsers that support SSL. They allow to set security options, or simply event driven pop ups that warn the user in the case of potentially dangerous actions.

In order to describe the operational environment we limit our scope for concerns of simplicity, although not a necessity, to the Microsoft Windows™ operating system family. Applications interface to the TCP/IP transport layer employing the socket interface offered by the operating system the WinSock dynamic link library (DLL) in our sample case.

Let us consider an organization that wants to enforce that signed and encrypted mail is used within the organization and with affiliate companies, that mail to customers is signed, and unsigned mail is used otherwise. Today's email clients usually cede the decision of signing or encrypting emails to the user. Thus, enforcing above sketched security policy for email communication is do be done by organizational means. Even if some email

clients allow for enforcing our sample scenario by technical means, this would require exclusive use of that certain client. A technical solution that is independent from the email client and, thus, increases the trust in the security system by exchangeability of the application, is presented in the following section 3.

3. S/MIME MAPPER, AN APPLICATION AWARE POLICY

In this section we discuss the implementation of an application aware security tool that assists in enforcing a certain security policy for email communication in an application independent approach by introducing a dedicated security layer. This layer enforces the policy regardless the user's decision whether to click or not to click the "encrypt" or the "sign button" when sending a mail. In addition, a major objective at that time was to provide strong cryptography, e.g., Triple data encryption standard (TripleDES) with 168 bit effective key length or 1024 bit signatures, even if "exportable" email clients with restricted cryptographic strength were used.

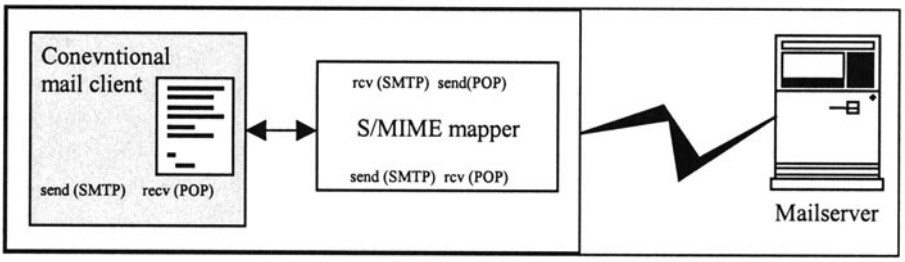

Figure 1. The S/Mime-Mapper

The approach we followed was to implement a security layer that is running on the user's machine and feigns a mail server, i.e. "talks" and "understands" SMTP and POP-3. When sending a mail (the SMTP path) the mail body the data section of the email the user sends is extracted from the mail, converted to an S/MIME attachment and sent to the mail server Vice versa, when receiving email (the POP-3 path), the original mail body is extracted from the email and presented to the mail client. We term that approach "S/MIME mapping" and the tool resulting from that ideas is called "S/MIME mapper".

Figure 1 illustrates above described scenario of an dedicated security layer. The S/MIME mapper is implemented in JAVA, therefore platform independence is provided. The solution has been tested on several platforms,

such as Windows NT™, Windows98™, LINUX, or Solaris™. From the user's point of view the only difference to "conventional" email communication is that *localhost* is entered as the SMTP server and POP server.

Where does our sample policy we have sketched in section 2 fit in? Regarding key management, S/MIME mapper includes several tools to perform key generation and basic certification authority (CA) functions. User certificates can be stored in a LDAP database and automatically be retrieved by S/MIME mapper.

The policy itself is defined with the configuration of S/MIME mapper upon the sender's email address (POP-3 path) or receiver's email addresses (SMTP path). Wildcards allow to group user's. Figure 2 shows the essential parts of a sample email policy, defining TripleDES and signed mail within the organization *our.org*, signed plaintext mail to our customers *cutomers.com* and unsigned mail otherwise.

```
our.org.smime.proxy.cipher = *@our.org:TripleDES:signed
our.org.smime.proxy.cipher = *@costumers.com:plain:signed
our.org.smime.proxy.cipher = *:plain:plain

# policy preferences for incoming messages,
our.org.smime.proxy.receive-preferences = *@our.org:TripleDES:signed
our.org.smime.proxy.receive-preferences = *@customers.com:plain:signed
```

Figure 2. A sample email policy

Two policy sets are defined, one for sending and one for receiving mails. Not only is the security policy enforced when sending mails, the user is also informed of violations of the incoming policy by a mail body explaining the reason, the mail causing body the violation is MIME attached to that violation message.

With reference to the questions raised in the introduction, the dedicated, application-aware solution to enforcing a security policy for email communication allows to give the following answers:

- Bugs or viruses in the application do not affect the security layer.
- Presentation errors in the application do not interfere with the security policy.
- Neither the application, nor the user needs to volunteer for the security policy, as this is detached from the application and there is no need for the user to decide whether to encrypt or to sign a message.

What is obviously missing in the approach is that the mail body is encrypted, but the remaining communication passes the S/MIME mapper more or less unmodified to the mail server, such as usernames and passwords. As the cryptographic tool employed [16] includes an SSL implementation, advancing to connection level security by establishing SSL-

tunnels between the S/MIME mapper and the mail server would be quite straightforward. However, we now broaden our mind to the wide range of applications in use by organizations, in particular access to an Intranet. from the Internet. This is done in the following section 4.

4. SSL AS SECURITY TOOL

In this section we attempt to present a security solution basing on communication aware policy enforcement. Therefore, we present a VPN-like approach that employs SSL to control access to the Intranet from the Internet or to link an organization's network via the Internet.

As a simple example, we first consider a policy, where the organization's policy defines that a telnet session or a hyper text transfer protocol (HTTP) session to specific servers within an Intranet is allowed from the Internet, if and only if the session is encrypted and authenticated by the user's certificate. Application-awareness as discussed in the previous section is usually not required in that case. We therefore followed an approach we call "SSL tunneling".

Again, the security layer is dedicated to be independent from the application, respectively the solution is implemented in JAVA to reach platform independence. What is different in the tunneling approach is, that the security layer at the client site has a counterpart at the server site. The application accesses the security layer installed at the local machine at the specific transport layer port assigned to that service service. The security layer maps the transport layer session to a SSL secured session to the server within the Intranet itself running the SSL counterpart and re-mapping the service to the plaintext port. To enforce the security policy, a firewall blocks the original services, but allows the mapped services to pass through to the servers involved. The difference to "conventional" telnet or HTTP communication is that the client accesses *localhost* instead of the intended server. The SSL-mapper is in charge of securing the communication at the transport layer.

5. APPROACHING APPLICATION INDEPENDENCE

The problem addressed in the previous section origins in attempting to feign a server at the same machine where the client is running, but a significant class of applications acts in a client's and a server's role simultaneously, i.e., listening on a certain port in the server's role and using

the same port when connecting to a server in the client's role. An example of such applications are video-telephony applications, acting as a server when being called, vice versa as a client when placing a call.

As we cannot modify the applications to fit our requirements to use alternate ports as a client, we would have to modify our approach. On the other hand, having an operational solution, why not approaching application independence by fooling the whole range of applications? We are currently developing such an approach by re-mapping the application's access to the operating system's TCP/IP protocol stack. Again limiting our scope to the Windows™ operating systems, we replace the WinSock DLL (*wsock32.dll*) by a dummy function that offers the WinSock function calls, as well as its messages to the application and maps it to the original library (let us call this library *orig.dll*). Thus we gain full control of the access to the TCP/IP protocol stack. This enables to pass function calls from the spoofed *winsock32.dll* to *orig.dll*, or to change the parameters such as port numbers.

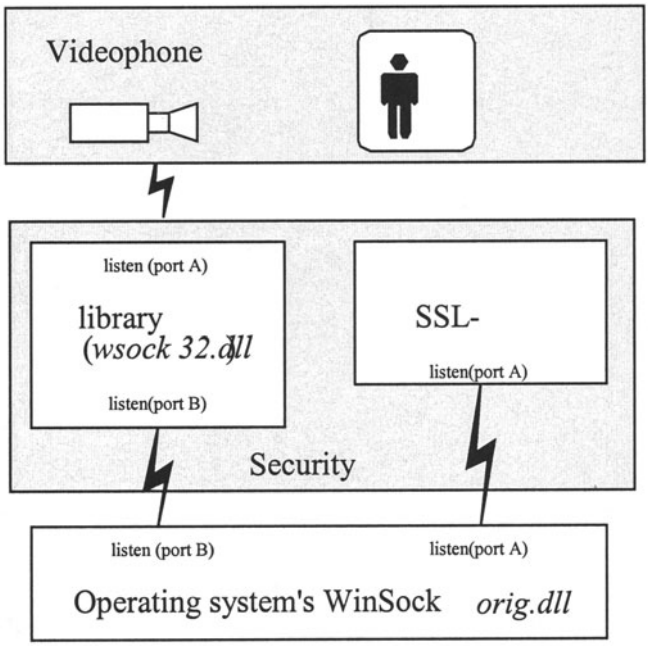

Figure 3. Protocol stack re-mapping

Figure 7 shows the new scenario. The security layer now consists of the two components "library mapper" and "SSL mapper". What has been lost is the platform independence. As the required functionality in the library mapper is mainly to monitor and redirect function calls and to modify

parameters, this is assumed feasible in several operating systems. What has been gained is a powerful tool in terms of enforcing security policies, which is described in the following section 6.

6. POLICY AWARE SOCKETS

The paper has so far described four tools, three of it platform independent as completely implemented in JAVA. These tools are the "S/MIME mapper" capable of securing email communication because of application-awareness, the "SSL mapper" and the "SOCKS-5 enhanced SSL mapper" providing the functions to establish a transport layer oriented, SSL-secured VPN and, finally, the "library mapper" that gives full control over the application's access to the TCP/IP protocol stack at the transport layer.

Applying the methods described for the S/MIME mapper in section 3 to the socket oriented tools, a comprehensive tool is given that is independent from the application. As the application is not aware of any security mechanism employed by the TCP/IP protocol stack a wide range of applications is supported. We have gained a concept we term "policy aware sockets". In order to explain the policy definition figure 8 illustrates such a policy that defines a SSL-secured telnet session to a server.

```
CipherSuite = *strong

acceptHost = localhost

# Client authentication
ClientAuthFile = certs\clientKey.der
TrustedRoots = certs\caCert.der
### pop-up the PKCS12Password = none


TelnetTunnel.InPort = 23
TelnetTunnel.OutPort = 20023
TelnetTunnel.connectHost=server.our.org
TelnetTunnel.OutputUsesSSL=true
```

Figure 4. A sample policy-aware socket

Applying the policy-aware socket approach to an organization results in integrating the security services to a dedicated security layer. The security policy can be assigned to the users in an obligatory approach. This increases the quality of the security services. By restricting the access to the layer mapping tools, i.e. shielding the operation system's protocol space from the application or user's space, the users and applications are bound to the defined security policy. This avoids intentional or unconscious bypassing of the defined security policy. We assume that approach even attractive for the

user's, as the need of scrutinizing actions to be taken against the policy is reduced.

7. SECURE VIDEO AN APPLICATION EXAMPLE

We finally give an application example that together with the S/MIME mapper discussed in section 3 is planned to be presented in action at the Case Studies Track (see statement at the first page), i.e., interspersing the presentation of the paper with the application example. In that demonstration we aim to present a video telephony application, where two scenarios are shown: Firstly, transferring the information without security measures and, secondly, to apply the policy aware socket approach. The comparison of the two scenarios shall proof, that the Quality of Service (QoS) is not affected by shielding the application's access to the protocol stack and applying strong cryptography. Figure 9 illustrates the aimed demonstration.

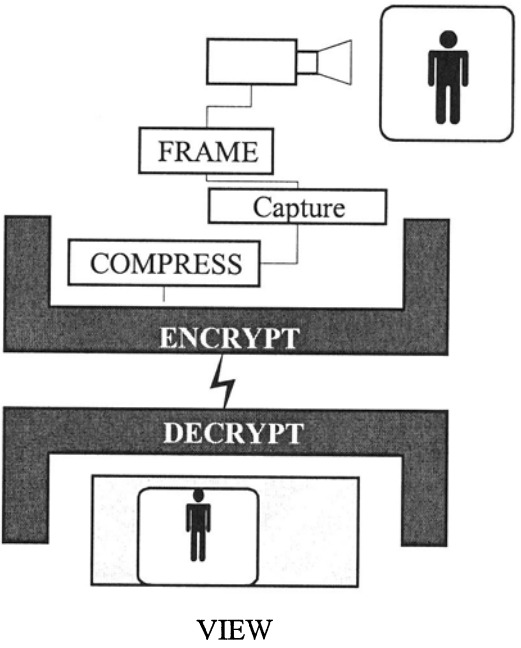

Figure 5. Shielding the application's protocol stack access

8. CONCLUSIONS

In this paper we have presented an approach to securing an organization's operational environment in terms of network protocols used and applications required. The approach represents a user and policy oriented VPN that applies cryptography at the transport layer on a subset of the communication. This is done by fooling the application and feigning a conventional access to the protocol stack. For the special case of email communication, an application-aware approach has been discussed which is termed S/MIME mapping.

By employing a dedicated, application independent security layer, enforcement of the crypto policy is supported. The approach even allows to define the security policy regardless a situation where the supplier of a certain application is limited in the cryptographic strength of application build-in security because of export restrictions.

9. REFERENCES

[1] European Union: Draft Directive of the European Parliament and of the Council on a common framework for electronic signatures, The European Council, March 1999.

[2] ITU T, The directory authentication framework, International Telecommunication Union, Recommendation X.509, 1989.

[3] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, S/MIME Version 2 Message Specification, RFC 2311, March 1998.

[4] A. O. Freier, P. Karlton, P. C. Kocher, SSL Protocol, Version 3.0

[5] S. Salamone, VPN's Defining Moment: What exactly is it?, CPN Media, Internet Week, Issue 749, January 1999.

[6] R. Thayer, Bulletproof IP: With authentication and encryption IPSec adds a layer of armour to IP, CMP Net, November 1997, online available at http://www.data.com/tutorials/bullet.html

[7] S. Deering, R. Hinden, Internet Protocol, Version 6 (Ipv6) Specification, RFC 2460, draft standard, December 1998.

[8] MSDN, Introducing Active Server Pages, Microsoft Developer Network, 1999.

[9] Netscape Corporation, Client-Side JavaScript Specification, Version 1.3, 1998.

[10] W3C, Can one Website hijack anothers content?, the WWW Security FAQ, Client site security, Q.68, World Wide Web Consortium (W3C), 1999.

[11] C. Oakes, Cache and Carry, Wired News, online available at http://www.wired.com/news/news/technology article 15285, September 1998.

[12] Wired, Email Links Carry Bad Code, Wired News, available at http://www.wired.com/news/news/technology, article 14288, August 1998.

[13] M. Leech, M. Ganis, Y. Lee, R. Kuris, SOCKS Protocol Version 5, RFC 1928, proposed standard, April 1996.

[14] J. Postel, Simple Mail Transfer Protocol, RFC 821, standard, August 1982.

[15] J. Myers, M. Rose, Post Office Protocol, Version 3, RFC 1933, standard, May 1996.

[16] IAIK, IAIK JCE 2.5, Java Cryptography Extensions, Reference Manual, 1999.