Web Security Infrastructure Issues and Solutions

Proposal

Authors: Rade Adamov, Web Application Development Support (I2EB)
Tobias Christen, Advanced Engineering Center (IWP2)
Thomas Ernst, Advanced Engineering Center (IWP2)
Shane Hurley, Web Server Platform (I0N8)
Thomas Leemann, Web System Platform (I8HU)
Elisabeth Maier, Advanced Engineering Center (IWP2)
Andres Schmid, Web Server Platform (I0N8)

Version: **2.0**

1 INTRODUCTION 6

1.1 SCOPE OF THE DOCUMENT 6
1.2 GOAL OF THE DOCUMENT 6
1.3 INTENDED AUDIENCE 6
1.4 TYPES OF SECURITY RISKS 6

2 ASSUMPTIONS 7

2.1 TECHNOLOGICAL ASSUMPTIONS 7
2.2 RESPONSIBILITIES / CONTACTS 7

3 SECURITY STRATEGIES 9

3.1 POLICY STATEMENTS 9
3.1.1 CENTRAL OBJECT SIGNING ORGANISATION WITHIN THE UBS 9
3.1.2 OBTAINING CERTIFICATES 9
3.1.3 GRANTING PRIVILEGES ON THE INTERNET 9
3.1.4 BROWSERS AND BROWSER MANAGEMENT WITHIN THE UBS 10
3.1.5 INTRANET TRUSTED SYSTEMS GUIDELINES 10

4 TECHNICAL ISSUES 11

4.1 BROWSER RISKS 11
4.1.1 DISCUSSION 11
4.1.2 REQUIREMENTS 11
4.1.3 PROPOSED SOLUTION 11
4.1.4 RESPONSIBILITY / CONTACTS 12
4.1.5 DUE DATES (FOR CRITICAL ISSUES ONLY) 12
4.2 JAVA APPLICATION SECURITY 13
4.2.1 DISCUSSION 13
4.2.2 REQUIREMENTS 13
4.2.3 PROPOSED SOLUTION 14
4.2.4 RESPONSIBILITY / CONTACTS 15
4.2.5 DUE DATES (FOR CRITICAL ISSUES ONLY) 15
4.3 JAVASCRIPT SECURITY 16
4.3.1 DISCUSSION 16
4.3.2 REQUIREMENTS 16
4.3.3 PROPOSED SOLUTION 16
4.3.4 RESPONSIBILITY / CONTACTS 16
4.3.5 DUE DATES (FOR CRITICAL ISSUES ONLY) 16
4.3.6 FURTHER READING / RELEVANT LINKS 16
4.4 CGI / PERL 18

4.4.1 Discussion 18
4.4.2 Requirements 18
4.4.3 Proposed Solution 18
4.4.4 Responsibility / Contacts 18
4.4.5 Due Dates (for critical issues only) 18
4.4.6 Further Reading / Relevant Links 18
4.5 Netscape Plug-Ins, ActiveX and Authenticode, Code Signing, Security of Downloaded Code, Log Files, Cookies 19
4.5.1 Discussion 19
4.5.2 Requirements 19
4.5.3 Proposed Solution 19
4.5.4 Responsibility / Contacts 20
4.5.5 Due Dates (for critical issues only) 20
4.6 Digital Signatures 21
4.6.1 Discussion 21
4.6.2 Requirements 21
4.6.3 Proposed Solution 21
4.6.4 Responsibility / Contacts 22
4.6.5 Due Dates (for critical issues only) 22
4.7 Secure Mail 23
4.7.1 Discussion 23
4.7.2 Requirements 23
4.7.3 Proposed Solution 23
4.7.4 Responsibility / Contacts 23
4.7.5 Due Dates (for critical issues only) 23
4.8 PKI, Certificates (Client- and Server-Side) 24
4.8.1 Discussion 24
4.8.2 Requirements 24
4.8.3 Proposed Solution 24
4.8.4 Responsibility / Contacts 25
4.8.5 Due Dates (for critical issues only) 25
4.9 Code Signing / MS Authenticode 26
4.9.1 Discussion 26
4.9.2 Requirements 26
4.9.3 Proposed Solution 26
4.9.4 Responsibility / Contacts 26
4.9.5 Due Dates (for critical issues only) 26
4.10 APIs 27
4.10.1 Discussion 27
4.10.2 Requirements 28
4.10.3 Proposed Solution 29
4.10.4 Responsibility / Contacts 30
4.10.5 Critical Dates 30
4.11 OS Security 31
4.11.1 Discussion 31

4.11.2 Requirements 31
4.11.3 Proposed Solution 31
4.11.4 Responsibility / Contacts 31
4.11.5 Due Dates (for critical issues only) 31
4.11.6 Further Reading / Relevant Links 31
4.12 Enterprise Server Security 33
4.12.1 Discussion 33
4.12.2 Requirements 33
4.12.3 Proposed Solution 33
4.12.4 Responsibility / Contacts 33
4.12.5 Due Dates (for critical issues only) 33
4.12.6 Further Reading / Relevant Links 33
4.13 File Upload to Web Server 34
4.13.1 Discussion 34
4.13.2 Requirements 34
4.13.3 Proposed Solution 34
4.13.4 Responsibility / Contacts 34
4.13.5 Due Dates (for critical issues only) 34
4.13.6 Further Reading / Relevant Links 34

5 A PROPOSAL FOR AN ARCHITECTURE 36

36

5.1 Credential Mapping 36
5.2 Protection Against Malicious Mobile Code 37
5.3 Synchronizing Users 37
5.4 LDAP Issues 38

6 OPEN ISSUES 40

6.1 General Remarks 40
6.2 PKI Related Issues 40

7 APPENDIX 41

7.1 Public-Key Infrastructure (PKI) 41
7.1.1 PKI Architecture 41
7.1.2 Architectural Issues in Implementing and Using PKIs 42
7.1.3 PKI Standardization Efforts 45
7.1.4 PKI Standards 46
7.2 Guidelines for JavaScript Security 49
7.2.1 Client Side JavaScript 49
7.2.2 Server-Side JavaScript 51

7.3 GUIDELINES FOR THE WEB ACCESS THROUGH CGI/PERL 52
7.3.1 GUIDELINES FOR PROGRAM DEVELOPMENT, TEST & INSTALLATION 52
7.3.2 GUIDELINES FOR THE WEB SERVER ADMIN: 52
7.4 GUIDELINES FOR OS SECURITY 53
7.4.1 GUIDELINES FOR SECURING A SOLARIS BOX IN TRUSTED SYSTEMS ON THE INTRANET OR ON THE INTERNET 53
7.5 SECURITY SETTINGS FOR NETSCAPE ENTERPRISE AND ADMINISTRATION SERVERS IN UBS TRUSTED SYSTEMS OR INTERNET 57
7.5.1 RECOMMENDED SETTING FOR A SECURE INSTALLATION OF ADMINISTRATION SERVER 57
7.5.2 RECOMMENDED SETTING FOR A SECURE INSTALLATION OF ENTERPRISE SERVER 58
7.6 APIS 60
7.6.1 LOW-LEVEL APIS 60
7.6.2 HIGH-LEVEL APIS 62

8 ACRONYMS 63

1 Introduction

1.1 Scope of the Document

With the growing use of both the Inter- and the Intranet within the bank web-related security issues are of outstanding importance. As platforms are set up to allow both easy publishing of documents (Web Publishing Platform) and the rapid development of Web applications (Web Application Platform, WAP) a security infrastructure has to be provided which takes web-specific aspects into account.

1.2 Goal of the Document

The main goal of this paper is to identify security-relevant issues, which emerge with respect to the development and operation of Web applications. Since the topic of IT security is being worked on in various organization units within UBS these efforts need to be identified and coordinated as far as they are relevant for Web applications. With this paper, we want to achieve the following:

- list all the security issues which are relevant with respect to the development and operation of Web applications
- describe the solutions for these issues if available
- describe possible approaches if no solutions are available
- identify persons and/or organization units who are responsible for the individual security issues, who have the ownership of a solution or who will realize the requested solution
- point out security issues which are time-critical, i.e. which must be available at a given date since other projects depend on the availability of a solution.

1.3 Intended Audience

This paper is intended for the following audience:

- For developers of IT solutions (e.g. IT security, IT Sec) in the area of IT security
- For providers of an IT infrastructure (Applications Support and Systems Engineering)
- For developers of Web Applications who need an application-specific security solution.

1.4 Types of Security Risks

IT systems and networks are attacked in order to carry out the following actions:

- Monitor the environment and export collected information
- Change the configuration and/or the behavior of the user's system
- Open a back door to intrude a system

Therefore, IT solutions have to be available to handle the following security risks[1]:

1. Access hard disk and remove delete or change files
2. Access entries in the PC Registry

[1] This list is - of course – incomplete and open to additions.

3. Take control of the screen
4. Take control of the keyboard and / or trace key input
5. Attack web content and replace it with porn or false information
6. Attack processes on the machine in order to crash them
7. Denial of service attack; tie up resources so that service is no longer available.
8. Gain shell access to the box by killing processes
9. Fake certificate authority to take on appearance of UBS certificate authority
10. Trick client into downloading hostile software which may carry out any of these attacks
11. Decrypt encrypted messages with number crunching software
12. Attack servers via client input data (CGI etc.)
13. Directly attack the box hosting software via `ftp`, `telnet`, `rlogin` etc.
14. Use `https` to by-pass firewall restrictions (e.g. Java White List)
15. Attack network hubs and routers to spoof originating IP address
16. Reverse compile programs to gain information on methodology or system

2 Assumptions

2.1 Technological Assumptions

- Unconsolidated browser situation:

  The situation with respect to the recommended browsers within UBS is very heterogeneous. While in the future the Communicator will be the recommended browser, various applications still use the Internet Explorer. Even as far as the Communicator is concerned the situation is still unconsolidated. The following Communicator versions are currently in use:

	GDIP	OLPC	OLNT2.8.x	OLNT3.0x	EUP-D	OLU
Communicator Version	4.04	4.02	4.02	4.04	4.06	4.04
JDK Support	1.1.4	1.1.2	1.1.2	1.1.4	1.1.5	1.1.4
Sun Java Plugin	no	no	no	no	yes	no

- Use of 128 bit encoding. UBS is now allowed to use Netscape Communicator domestic
- Minimization of password usage
- No access on client data from outside Switzerland unless rights are explicitly granted

2.2 Responsibilities / Contacts

General Contacts[2]:

[2] Contacts are itemized. The responsibilities given in this table may be subject to change as the IT organization is restructured. Recently, a new section "Security Engineering" has been set up which will take over some of responsibilities listed in this table.

- Security Architecture & Policy I58I: IT Security
  Contact: Steffen Norbert (NTS), Walter Widmer (WWD)
- Operational Security I70E
  Contact: Hans-Rudolf Kramer (KQH), Paul Reigrotzki (REG)
- IT Security Controlling I541: IT Security Compliance
  Contact: Stefan Vogt (VGF)

Field / Responsibility	**Name / Short Code**	**Unit**
Security Architecture & Policy	Walter Widmer (WWD) *Steffen Norbert* (NTS)	I58I IVYU
Operational Security	Paul Reigrotzi (REG) *Hans-Rudolf Kramer* (KQH)	I70E
IT Security Controlling, IT Security Compliance	Stefan Vogt (VGF)	I541
Object signing for Java (Software Leitstelle)	Lisbeth Suter (S1B)	IM6O
Introduction of new security-relevant technologies (PKI, Java-related developments)	Jürg Ganz (GJU)	IVYU
Certification Authority	NN	IVYU
Distribution of X.509 certificates for Web Servers	Jürg Zwahlen (ZWJ)	IU6W
Privileges for the Internet	Konrad Schmid (SYM)	IBTN
TOC: Control and possibly standardization of plugins, Java applets, and Java applications	Albert Vollmer (V9A) *Thomas Leemann* (LMT)	I7F6 I8HU
Java Application Security	*Tobias Christen* (CKC) *Thomas Ernst* (EYT) *Jürg Ganz (GJU)*	IWP2 IWP2 IVYU SLIB ??
JavaScript Security	*Shane Hurley* (H8H) *Lilian Fischer* (FCL)	I0N9 I0N9
CGI / Perl	*Shane Hurley* (H8H) *Thomas Leemann* (LMT)	I0N9 I8HU
Digital Signatures	*Jürg Ganz* (GJU) *Thomas Ernst* (EYT) (API specification only)	IVYU IWP2
Secure Mail	Peter. Strickler (SPH) Jürg Ganz (GJU) Markus Bitterli (BAJ) Paul Reigrotzki (REG)	IU6W IVYU IVYU I70E
PKI / X509 / V3	Walter Widmer (WWD) Steffen Norbert (NTS) (to be clarified) Jürg Ganz (GJU)	I58I IVYU IVYU
Digital Certificates	Jürg Ganz (GJU)	IVYU

Code Signing	Jürg Ganz (GJU) *Thomas Ernst* (EYT) (Java, API specification only)	IVYU IWP2
APIs	Thomas Ernst (EYT) (SSL) Jürg Wanner (WNU) (GSS) Christoph Frei (FCI) (Java GSS) NN (Directory Server) Daniel Steiner (SI3) (SSL/IIOP/ISI)	IWP2 I3Z4 II1I IJNG
OS Security	Andres Schmid (SJF) Shane Hurley (H8H)	I0N8 I0N8
Enterprise Server Security	Andres Schmid (SJF) Shane Hurley (H8H)	I0N8 I0N8
File Upload to Server	Shane Hurley (H8H)	I0N8

3 Security Strategies

3.1 Policy Statements

Since security strategies are only about to be defined, it is very hard to make concrete statements about security policies. In the following, we try to give some hints in which areas security policies are/will be necessary and who might be the best persons to contact for more information.

In general, Jürg Ganz from IT Security Infrastructure Support (IVYU) is the person to contact when it comes to the introduction of new technologies. This especially applies to the Public-Key Infrastructure being built and to Java Security.

3.1.1 Central Object Signing Organisation within the UBS

At present, Lisbeth Suter from SW-Leitstelle (IM6O) controls the process of object signing for Java applets. The corresponding Certification Authority has been set up by IT Security Infrastructure Support (IVYU). In the future, this organizational unit may also be responsible for the signing of Java applications.

3.1.2 Obtaining Certificates

User X.509 certificates are distributed via the same means and according to the same guidelines as PERSAUTH smartcards.

Server X.509 certificates are currently only distributed for Web servers. The decision is upon the Network Control Center (Jürg Zwahlen?).

3.1.3 Granting Privileges on the Internet

Privilieges for the Internet are granted by the UBS unit Interworld (IBTN) according to the terms, conditions and responsibilities described in the "UBS directive PF/6/008 Internet Access for UBS Staff".

3.1.4 Browsers and Browser Management within the UBS

Thomas Leemann

3.1.5 Intranet Trusted Systems Guidelines

For the development of a secure Web Infrastructure a number of plans and policies have to be known:

- Firewall policy
- LDAP solution (not yet defined)
- PERSAUTH and its integration into the Web (IT Security, I58I)

For each of these points the responsible persons and/or organizations need to be known. Decisions made concerning the above-mentioned issues need to be communicated as soon as possible.

4 Technical Issues

4.1 Browser Risks

4.1.1 Discussion

Today the World Wide Web and in our case the internal Bank Web is considered to be a platform for building distributed applications. This evolution of a new business model is made possible by general purpose browsers with processing capabilities and by programming environments that allow web application designers to embed real programs into HTML documents. Downloading such documents and executing included code from anywhere on the Internet or Intranet can cause severe security problems. A systematic and in-depth analysis of possible security breaches in the browser and related technology is necessary to reach a sufficient level of confidence for the enterprise as a whole and the single user.

Browsers can be open to a variety of security attacks which fall into the following three basic classes:

- monitoring the environment and exporting collected information
- changing the configuration and/or the behavior of the user's system
- opening a back door to intrude a system

Many attempts to attack a user's system succeed because the implementation of the browser software is weak (weak specification, specification flaws or poor implementation) or the environment in which the browser is executed has flaws. The open nature of today's software contributes substantially to the weakness of the environment. The old paradigm that a program is once installed and configured forever is no longer valid for open and mobile environments using Java, JavaScript and the Web (Internet, Extranet and Intranet).

4.1.2 Requirements

The following requirements have to be met to ensure secure communication:

- A browser technology, which uses the best security model available on the market.
- Strong encryption for browsers and selected applications.
- Blocking of unwanted technologies at firewalls (Intranet-Internet and Intranet-Intranet).
- Engineering of browser supported security features (policies for: cookies, the insecure sending of forms, ...)
- Support of certificates and digital signatures
- Detailed guidelines which describe what browser-related technologies (Java, JavaScript, ActiveX, runtime specific DLL's) are allowed for Web-based applications and in what geographic context (region Switzerland, international, country-specific).

4.1.3 Proposed Solution

- The browser policy for the bank defines that Netscape is currently the standard browser. Due to the fact that more and more external vendor applications as well as internal applications make use of parts of the Microsoft Internet Explorer 4.0x (MSIE), MSIE will be installed on the user's system for EUP-D. The engineering of MSIE will be in such a way that MSIE is invisible to the user and cannot be used for browsing. Furthermore the proxy/firewall Intranet-to-Internet blocks all MSIE requests to the Internet as requested by IT Security Switzerland.

- Control of the open and mobile environment by means of pre-configuring and locking of certain settings (i.e. proxy settings) in the browsers package.
- Central control of the browser configuration via mechanisms like Netscape Mission Control.
- Implementation and use of certificates and the necessary signing processes for the various Java Virtual Machines (JVM) sitting on the user's desktop. Currently on all platforms within UBS the Netscape JVM is signed against an UBS certificate.
- Use of US-domestic browser technology.
- In the Workplace TOC (Head A. Vollmer) a process is currently under definition which is intended to control all plugins, Java applets and Java applications. This will ensure that standards can be better enforced. (Contact: T. Leemann)

4.1.4 Responsibility / Contacts

T. Leemann

4.1.5 Due Dates (for critical issues only)

4.2 Java Application Security

4.2.1 Discussion

Java is rapidly emerging as an industry standard, for thin-client architectures as well as for sever implementations. Also, in UBS AG more and more mission-critical applications are written in Java. The existence of an adequate security infrastructure for these architectures and applications therefore becomes a crucial issue.

Java applications should at least profit from the same infrastructure (e.g., the same APIs) as applications written and deployed in other languages (e.g., C++, Smalltalk). Currently, such an infrastructure is partially built up, but still does not provide all the features required, as described in the following section.

4.2.2 Requirements

The following requirements are based on the use of the Java Development Kit (JDK) 1.1.x. They will have to be adapted and extended with the use of JDK 1.2.

4.2.2.1 Authentication, Privacy, and Data Integrity

- SSL

  Secure Sockets Layer (SSL) is a world-wide used protocol standard addressing the need for authentication, privacy, and data integrity. To use this protocol, both server and client must be SSL-enabled which in essence means that they must be able open SSL sockets instead of plain sockets. Therefore, application developers must be provided with a Java SSL package which allows them to make the appropriate replacement. Ideally, this is the only thing an application developer should be concerned with. All other SSL aspects like defining the cipher suites to be used, verification of the X.509 certificates being exchanged, lookup of certificate revocation lists, retrieval of private keys (from smart cards via PKCS#11) etc. have to occur under the hood according to a specific security policy.

- GSS-API

  For the Generic Security Service (GSS) API, requirements similar to the SSL requirements can be formulated. Application developers must be provided with a Java GSS package which encapsulates the calls to the already existing GSS infrastructure. They should be able to just replace plain sockets by some kind of GSS sockets. Decisions like which mechtype to use must be hidden.

- Generic security library

  Application developers must be given the possibility to use generic security functions like IDEA encryption or RSA signing inside their applications. Such a security package should conform to the Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) defined by Sun. In this way, it will be interoperable with other libraries conforming to the same standard and can later be exchanged with minimal effort.

4.2.2.2 Authorization

- OLAU

  Authorization via OLAU (OpenLan Authorization) is and will be the strategic means to authorize users in the environment of decentralized applications. Therefore, there must be an OLAU API to use from inside Java applications. To exploit the multithreading power of Java, this library should be a pure Java implementation and not just a wrapper around existing C libraries. Currently, OLAU authorization is more flexible with respect to business data-driven authorization restrictions than Java's

access control list API (java.security.acl and related). However, this may change in the future and must be taken into consideration.

4.2.2.3 Audit Trails (only server-side)

Preferably, it should be possible to generate audit trails using the corresponding CORBA services (via IIOP/SSL!). At least, applications must be presented with an API to make their own audit entries.

In the future, it might also be desirable to provide means and APIs for generating digital signatures to enable non-repudiation. (Currently, non-repudiation is only planned in the context of signed documents.)

4.2.2.4 Security Policy

In JDK 1.2, the security policy to be used for a particular application can be specified in a special policy file. This policy file can be placed on the local disk or be taken from a directory server (i.e., LDAP). The policy file defines permissions for classes based on their digital signature and their origin. In this way, a very fine-grained level of access control can be established, such as allowing socket connections to a particular host, reading and writing on a particular directory etc.

On the one hand, this approach makes it very easy to customize a security environment, on the other hand, the following two problems must be addressed (beside others):

- Who defines the security policies?
- How are security policies distributed or where does the local environment take it from (local hard disk, LDAP etc.)?

Also see section 4.9.

4.2.3 Proposed Solution

- SSL

  There are several commercial Java libraries available, all of them conforming to the JCE:

  - IAIK-JCE from the Technical University of Graz (http://jcewww.iaik.tu-graz.ac.at/). Also licensed by Entrust Technologies.
  - Java SSL from JCP Security Services (http://www.jcp.co.uk). Also licensed by IONA Technologies.
  - J/Crypto from Baltimore Technologies (http://www.baltimore.ie/jcrypto.htm). Also licensed by RSA Data Security.

- GSS-API

  There is already a UBS GSS implementation for Java. It might be desirable to adapt this library to the Java GSS binding currently being defined by Sun Microsystems (draft-ietf-cat-gssv2-javabind-00.txt, downloadable from http://www.ietf.org/internet-drafts/).

- Generic security library

  The companies providing Java SSL libraries (see above) also provide Java cryptography libraries.

- In the Workplace TOC (Head A. Vollmer) a process is currently under definition which is intended to control all plugins, Java applets and Java applications. This will ensure that standards can be better enforced. (Contact: T. Leemann)

4.2.4 Responsibility / Contacts

T. Christen, T. Ernst
IT Security, IT Operations
Team File Transfer Services

4.2.5 Due Dates (for critical issues only)

4.3 JavaScript Security

4.3.1 Discussion

The problem of JavaScript security can be approached from two different angles:

- Client-side JavaScript
- Server-side JavaScript

4.3.1.1 Client-Side JavaScript

The following issues are critical with respect to the security of client-side JavaScripts:

- Sensitive information must not be revealed
- For Signing of JavaScript Scripts (Object Signing)
  - the weakest script's authority (for SSL Servers and Unsigned Scripts) must be assumed
  - only use Communicator 4.x should be used
  - Principals need to be checked for Windows and Layers
  - Privileges must not be granted to external ecripts
  - Disable Code Base Principals
  - Sensitive functions must not be exported
  - Porting has to be prevented
  - The Trusted Code Base needs to be minimized

4.3.1.2 Server side JavaScript

See section on CGI; most of what is said there applies to Server Side JavaScript. The Netscape JavaScript Application Manager improves security via access control and passwords.

4.3.2 Requirements

4.3.3 Proposed Solution

Detailed guidelines, which describe how to resolve the above-mentioned client-side and server-side security issues are included in section 7.2 of the Appendix.

4.3.4 Responsibility / Contacts

Shane Hurley, Lilian Fischer

4.3.5 Due Dates (for critical issues only)

4.3.6 Further Reading / Relevant Links

- Netscape JavaScript security model
  http://developer.netscape.com/docs/manuals/communicator/jssec/index.htm
- JavaScript Apostle:
  http://developer.netscape.com/viewsource/goodman_sscripts.html
- Meta-FAQ:
  http://ugweb.cs.ualberta.ca/~thompson/programming/javascript/meta-FAQ.html

- FAQ http://www.irt.org/script/faq.htm
- object signing: http://developer.netscape.com/docs/manuals/signedobj/trust/index.htm
- Java Capabilities API http://developer.netscape.com/docs/manuals/signedobj/capabilities/index.html

4.4 CGI / Perl

4.4.1 Discussion

4.4.1.1 Security Aspects of Deploying Programs accessed via CGI on a Web Server

CGI is a method of calling a program from a web page. The remote user is therefore running a program on an internal server and may cause damage by forcing this program to crash. Allowing use of CGI to a web server is very dangerous; weak programs may even give the remote user a shell as root on the server. See http://www.w3.org/Security/Faq/wwwsf4.html for an overview of the problem area. These attacks are usually achieved by forcing the program to crash or by creating a buffer overflow giving bad input to the program.

4.4.2 Requirements

Apart from the solution described in the CGI-Guidelines (see section 7.3) a process needs to be setup where scripts are

- tested for the a number of weaknesses described in the CGI-Guidelines
- tested within the production environment (Internet DMZ, Intranet)
- passed by "Operational Security" (Hans-Rudolf Kramer, KQH)

4.4.3 Proposed Solution

The solution for the deployment of programs accessed via CGI on a Web Server are described in two guidelines which are given in the Appendix of this document:

- Guidelines for the Program Development, Test & Installation (see section 7.3.1)
- Guidelines for the Web Server Admin (see section 7.3.2)

In the case where a CGI is required a minimum set of guidelines needs to be followed together with guidelines from other sources. The list of weaknesses is always growing as hackers find more and more points of attack.

4.4.4 Responsibility / Contacts

S. Hurley, T. Leemann

4.4.5 Due Dates (for critical issues only)

4.4.6 Further Reading / Relevant Links

Netscape Secure Application Server: http://www.w3.org/Security/Faq/wwwsf4.html

4.5 Netscape Plug-Ins, ActiveX and Authenticode, Code Signing, Security of Downloaded Code, Log Files, Cookies

4.5.1 Discussion

Netscape Plug-Ins:

In general plug-ins inherently pose a potential risk since they can access the system without restrictions.Currently the packaged UBS Communicator includes the standard plug-ins as they are distributed by Netscape together with a very limited number of 'very important' plug-ins like the Macromedia Authorware plug-in under GDIP and EUP-D.

Code Signing, Security of Downloaded Code:

The UBS version of Netscape Communicator incorporates a UBS signed JVM. All Java applets, which are in need of leaving the Java sandbox (to access system resources) need to be signed with the UBS certificate. Unfortunately this signing process cannot be used for other JVMs but the Netscape JVM.

ActiveX and Authenticode: will be specified later

Cookies: will be specified later

4.5.2 Requirements

Netscape Plug-Ins:

- Forthcoming packages of the browser will include just the standard Plug-Ins as delivered by Netscape; this will be valid as long as the functionality makes sense for the banking environment.
- To reduce package internal dependencies of the Communicator package other plug-ins need to be packaged in separate packages and need to pass the standard UBS system test.

Code Signing, Security of Downloaded Code:

With the forthcoming EUP-D platform the Sun Java Plugin is distributed as a second JVM as it further enhances the Java functionality (e.g. Swing) on the user system. To allow applets to use the features of the Sun Java Plugin, IT Security needs to define a signing process similar to the existing one for Netscape.

ActiveX and Authenticode: will be specified later

Cookies: will be specified later

4.5.3 Proposed Solution

Netscape Plug-Ins:

- The system platform must reject the installation of plug-ins by the user; the user would need local admin rights.
- Plug-Ins must be owned by a data owner who either owns only a plug-in or who also owns the application which requests the plug-in.
- In the Workplace TOC (Head A. Vollmer) a process is currently under definition which is intended to control all plug-ins, Java applets and Java applications. This will ensure that standards can be better enforced. (Contact: T. Leemann)

Code Signing, Security of Downloaded Code:
IT Security has to define a process to sign Java applets designed for the Sun Java Plugin.

ActiveX and Authenticode: *to be specified*
Log Files: *to be specified*
Cookies: *to be specified*

4.5.4 Responsibility / Contacts

IT Security
T. Leemann

4.5.5 Due Dates (for critical issues only)

4.6 Digital Signatures

4.6.1 Discussion

Digital signatures bind a document to the possessor of a particular key and are the digital equivalent of paper signatures. *Signature verification* is the inverse of a digital signature; it verifies that a particular signature is valid.

Currently, no infrastructure for signing documents over the Web is in place at UBS. However, with the increasing use of the Intranet for exchanging information a strong need will emerge. In particular, document signing enables secure workflow processes.

Digital signatures are also be used for the delegation of access rights as well as for determining the authenticity of e.g. transaction requests.

4.6.2 Requirements

4.6.2.1 Public-Key Infrastructure

The existence of a productive Public Key Infrastructure (PKI) is a strong prerequisite for exploiting the potential of digital signatures. The PKI also addresses related problems like where the private key for signing should be stored (physical device like hard disk, removable media, smart cards and others), where certificates for signature verification should be retrieved from, which certification authorities are accepted, etc.

4.6.2.2 Legislation

Other questions (which however cannot be answered by IT alone) are questions like

- Is any legislation necessary for internal use of digital signatures or just for external use?
- How is risk allocated?

Answers to these questions have to be found before any infrastructure for digital signatures can be deployed.

4.6.2.3 Tools

Besides using the browser to sign and verify HTML documents, tools for off-line signature generation and verification must be provided. It is not yet clear, however, whether DSS or PKCS7 will be adopted as a standard for the format of digital signatures.

4.6.3 Proposed Solution

(technical issues only)

SSL is not an appropriate means for this purpose since it is just a secure transport channel; it does not provide any authenticity to Web documents. Thus only the delivering web server can be authenticated but not the author of a document.

Secure HTTP (S-HTTP, http://www.terisa.com/shttp/intro.html) in contrast addresses the need for signed documents, i.e., it enables document-oriented security: With S-HTTP it is possible to wrap HTML documents into a secure envelope containing digital signature and key information.

The Digital Signature Initiative from the WWW Consortium (W3C, http://www.w3.org/DSig/) is another approach to provide attachable digital signatures to web documents. More generally, the goal of the project DSig is to provide a mechanism to make the statement "signer believes statement about information resource".

4.6.4 Responsibility / Contacts

J. Ganz; T. Ernst serves as contact person as far as the specification of APIs is concerned

4.6.5 Due Dates (for critical issues only)

4.7 Secure Mail

4.7.1 Discussion

Sending mail securely requires the following features in the mail system:

1. Authentication of the sender
2. Guarantee for the integrity of the message
3. Confidentiality of messages

Points 1 and 2 are satisfied using signing techniques, point 3 is satisfied using encryption techniques. MIME encoding extensions to SMTP allow 8 bit binary to be transferred in addition to 7 bit ASCII.

Further expansion of MIME to S/MIME (RFC 2311 to RFC 2315) allows a standard way of passing secure data and the information required by the system to recognize secure mail.

There are other methods like PEM and PGP which are not considered as UBS will be deploying a certificate infrastructure so it would be pointless to duplicate this cost/effort solely for email.

4.7.2 Requirements

The mail backbone (currently HP) must

- be S/MIME compliant, i.e. must comply with the envelope definition
- not change the header information specifically the "from" information since in that case signing is invalidated

Furthermore,

- the UBS CA must be in place where clients can request certificates including mail information.
- the deployed mail reader must support S/MIME (e.g. Outlook and Netscape Messenger)

4.7.3 Proposed Solution

Deploy S/MIME solution as specified above.

4.7.4 Responsibility / Contacts

J.P. Strickler, J. Ganz, Bitterli, Reigrotzki

4.7.5 Due Dates (for critical issues only)

4.8 PKI, Certificates (Client- and Server-Side)

4.8.1 Discussion

PKI represents a set of security services that supports the use of the public key cryptography and X.509 certificates in a distributed computing environment. These services must be available not only within a single network, but across networks and the Internet in order to enable electronic commerce and secure communications.

PKI products/services help in establishing security domains in which keys and certificates are issued. PKI should enable the use and management of both keys and certificates, such as key management (key generation, key update, recovery, escrow), certificate management (generation, revocation), and policy management.

Certificates are one of the core elements of a Public Key infrastructure. They enable global proof-of-identity which, for instance, allows for secure messaging and code signing, and for the delegation of identities (see the corresponding sections). Certificates can be equally assigned to users and machines/servers.

Consult the Appendix for a discussion of PKI architectures, components, standards, and related issues.

4.8.2 Requirements

Many of the currently available certificate server products do not fully address the infrastructure-related needs of large enterprises. Most of these are primarily concerned with creation, publication and management of certificates. They support storage and management of certificates in appropriate repositories (directories). Revocation of certificates is supported through certificate revocation lists (CRLs).

While these functions are important, PKI products must also offer key management services as well. By most experts key management is regarded the most significant challenge facing enterprise customers. Using certificates implies using keys, i.e. having multitude entities with one or more keys for each that have to be managed. Services like key backup, recovery, and update are hardly possible without a robust infrastructure.

Enterprise-level PKI must support automation of scary tasks like managing multiple sets of keys and certificates for each entity. PKI must also include policy-based management tools; these are to enable policy matching (e.g.: key life cycles, key usage) with the directory-enabled infrastructure for managing those policies.

There is no de-jure standard for certificates but the X.509v3 standard is the one widely adopted by the Internet community (the X.509 certificate syntax is specified in ITU-T Recommendation X.509). A commitment for this standard is highly required since it enables worldwide communication and compatibility between the UBS Intranet and the Internet, not only for user/server authentication but also for secure email, for instance. Moreover, all code-signing technologies utilize this standard and many commercial application servers (e.g. Netscape's) require the availability of X.509v3 certificates in their authentication scheme.

Currently, X. 509v3 certificates are only rarely used in UBS. The main reason for this is the lack of an appropriate infrastructure and of the corresponding policies. The present smartcard solution PERSAUTH still uses a proprietary certificate format. This must change in the future; a pilot is planned to integrate X.509v3 certificates with PERSAUTH cards.

4.8.3 Proposed Solution

It is currently unclear which of the vendors offer PKIs, which satisfy the bank's needs. The following steps may be conducted in order to come up with a solution:

- Define the group of vendors to be contacted regarding PKIs. This should be done together with other UBS-units dealing with Web/IT security.
- Conduct an extensive comparative analysis of PKIs from the selected vendors.
- Examine how well a given PKI integrates with the existing/planned IT infrastructure
- Conduct extensive tests to check the scalability of each candidate PKI's.
- Evaluate the interoperability of a given PKI
- Find out about the level of compliance/support for PKIX proposals

X.509v3 certificates as specified above should be used and integrated with PERSAUTH.

4.8.4 Responsibility / Contacts

IT Security, W. Widmer, J. Ganz, S. Norbert (responsibility to be clarified)

4.8.5 Due Dates (for critical issues only)

4.9 Code Signing / MS Authenticode

4.9.1 Discussion

The new version 1.2 of the Java Development Kit (JDK) provides an extended security model which is heavily based on code signing and therefore can only be exploited if an infrastructure for code signing is in place.

4.9.2 Requirements

4.9.2.1 Public Key Infrastructure

Productive Code signing is not possible without the existence of a Public Key Infrastructure (PKI) which also defines where certificates to verify signatures should be retrieved from or which certification authorities are accepted.

4.9.2.2 Tools

Tools to enable code signing must be provided to application developers. Because the technologies are not interoperable, a decision in favor of one technology has to be made (see below).

As there are at least two parties involved in code signing - the author of the code and the certifying authority - a workflow scheme, which ideally is platform and programming-language independent, is required.

4.9.3 Proposed Solution

4.9.3.1 Java Applications

Currently, three technologies for signing Java applications exist. Unfortunately, these technologies are not interoperable. However, all of them are based on X.509 certificates.

- Netscape Object Signing (http://developer1.netscape.com:80/docs/manuals/signedobj/trust/owp.htm). This technology is already used in UBS AG to sign Java applets.
- Sun (http://java.sun.com/products/jdk/1.2/docs/guide/security/index.html) provides the jarsigner tool to sign JAR (Java Archive Format) files and to verify the signature(s) of signed JAR files. Due to the US export restrictions, currently only signatures according to the Digital Signature Standard (DSS) are supported.
- MS Authenticode (http://www.microsoft.com/security/tech/misf8.htm)

4.9.4 Responsibility / Contacts

J. Ganz; T. Ernst serves as contact person as far as code signing of Java applications is concerned

4.9.5 Due Dates (for critical issues only)

4.10 APIs

4.10.1 Discussion

IETF has been reluctant in setting API standards with the exception of GSS, despite the fact that standard APIs enable application portability. APIs are often extensions of implementation work and must live in the context of the operating system and/or environment in which they exist.

Furtheron, API standards are de facto in nature, and are determined more by the market share than by fiat. Developers use APIs that give them access to the largest markets for their applications. APIs are defined by successful infrastructure products. As a consequence, we expect that the likelihood of a single standard API for the WSI is low.

4.10.1.1 Low-Level APIs

As a matter of fact, developers needing access to WSI can choose from a variety of APIs. Leading interfaces seem to be emerging from the pack. Microsoft's CryptoAPI and Intel's Common Data Security Architecture (CDSA) are the leading contenders in the WSI API category. Both vendors enjoy significant presence on the market. It is to be expected that most of the PKI vendors will eventually support both APIs.

Entrust, IBM, Netscape, and TIS made a proposal to the Open Group for cryptography and certificate management interfaces. All three vendors based their interface proposals on CDSA, along with extensions based on the Entrust's CMS API and the key recovery API defined by IBM.

It seems to be a common consensus that customers should consider CryptoAPI and CDSA as the important (low-level) interfaces to which they should be paying attention. Both APIs are described in the Appendix

4.10.1.2 High-Level APIs

Both of the APIs described above define low-level service provider interfaces. These are mostly for system vendors who create specific WSI products and write modules to these interfaces which make these services accessible to applications using the security services manager API. Modules may be compared with drivers that provide connectivity to specific implementations of WSI services.

The Security services manager API (e.g.: CSSM in CDSA) is a low-level API, too difficult for mainstream (application) developers to use. Most security problems do not come from breaking cryptographic algorithms, but from programmers making mistakes when implementing these algorithms in software. High-level APIs help prevent these problems by preventing developers from having to use complex security subsystems.

Here are a couple of examples how a high-level abstraction might help an application programmer when interfacing the WSI services:

- "Certificate" Java Beans might abstract the use of certificates.
- An entire security service (e.g.: Secure Electronic Transactions - SET) could be abstracted by implementing it on top of CDSA or CryptoAPI. Instead of calling the CSSM API directly, an application programmer calls the service directly. The service implementation uses the underlying CDSA architecture to access specific crypto services, needed to complete steps required by the application. Punctual implementations of entire security services already exist.

Full-blown high-level interfaces to WSI services should be provided by security development toolkits. Some (very few) high-level toolkit implementations exist. Toolkits based on some accepted standards are especially hard to find.

Open Group has officially recognized several candidate-high-level APIs as front runners, including the GSS-API and its equivalent for store-and-forward: Independent Data Unit Protection (IDUP) GSS-API.

Software Development Kit (SDK) is available from Netscape for application programmers to interface with the various Netscape servers. They are mostly available in Java and C, some are also available in JavaScript and Perl 5 (see http://developer.netscape.com/software/index.html and the Appendix)

4.10.2 Requirements

4.10.2.1 Requirements for Low-Level API(s)

- Low-level APIs should define/enable an overall architecture for implementing security services, including the relevant WSI functions.
- Digital certificates should be used as the primary method of identifying parties in a secure communication.
- Digital certificates should be used as the primary method of establishing policies that determine the actions that certificate holders can take.
- Low-level APIs should be built on a layered service-provider architecture. It should define a consistent application interface and a service layer that maps that interface to potentially different implementations of specific services. Developers thus gain independence from given implementations of the services while obtaining access to those services in a consistent fashion.
- Low-level APIs should nicely integrate into an existing Web infrastructure. This implies both the current infrastructure as well as the extensions/enhancements regarding (Web) security which are to be introduced soon.

4.10.2.2 Requirements for High-Level API(s)

High-level APIs should provide interfaces to all cryptographic and key management functions required by applications. They are not supposed to allow developers a direct access to symmetric and public-key algorithms. By being nunable to manipulate the cryptographic software directly, developers are protected from making security errors. An attempt is made here to come up with a list of APIs for some of the most common types of applications that we have to deal with:

- APIs tailored to the requirements of store-and-forward applications (e-mail, electronic forms, electronic commerce). The sending party does not need to establish a communication channel with the receiving party and the receipt of data is not required to occur in real-time. The following data formats should be supported:
  - S/MIME (Secure MIME)
  - PEM (Privacy Enhanced Mail)

  Since S/MIME uses the PKCS #7 format to deliver secure MIME-encoded data objects, and PEM is compatible with PKCS #7, we will probably end up having to support most of the PKCS #7. PKCS #7 supports encryption and signing of data blocks as well as transfer of certificates, CRLs, and certificate chains.

  This API should be available for C/C++ and Java, perhaps also for Smalltalk.
- APIs tailored to the requirements of on-line, real-time applications (remote access applications, database access applications, Web browser plug-ins). These applications secure real-time data exchanges between two parties engaged in on-line transactions. Applications of this type require confidentiality, authentication, and integrity for real-time data exchanges. Two standards that have been proposed/adopted by the IETF seem to represent appropriate choices:
  - GSS-API (already in use within the UBS)

- Key management protocol SPKM (Simple Public Key Mechanism)

In addition to this, one or more key establishment mechanisms should be supported, e.g. DH. The two key-related issues SPKM and KEA should be discussed with IT Security.

The GSS-API has only been defined for C. It would probably make sense to provide a Java wrapper around GSS-API.

This API should be available for C/C++ and Java. It is an open issue whether a solution is needed for Smalltalk as well.

- APIs for performing cryptographic operations:
  - API for PKCS #7
  - SSL
  - JCE
- APIs for supporting key lifecycle management and certificate management (e.g.: key updates), trust model management (e.g. cross-certification), and certificate management (e.g.: certificate revocation). This assumes that applications already contain cryptographic algorithms and do not require the APIs mentioned above.
- APIs that help reuse the public-key infrastructure for IP security. Relevant standards that were defined by the IETF are: AH (Authentication Header), ESP (Encapsulated Security Protocol), and ISAKMP (Internet Security Association Key Management Protocol). APIs of this sort help create and manage security credentials for each entity on the network. The necessity to provide this kind of API needs to be checked with IT Security.
- Java WSI-APIs. Since Java is supposed to be the mainstream programming language for the UBS, this part discusses in more detail, which WSI-related Java APIs should be provided. One of the open issues is whether and how much attention should be devoted to JDK 1.1. Some of the items in the following list have already been mentioned above:
  - SSL Java API providing the most widely used method of secure TCP/IP communications on the Internet.
  - Java Cryptographic Extension providing the necessary algorithms, extending the (insufficient) default offering by Sun Microsystems.
  - Generic Java API for ASN.1 providing encoding for numerous X.509 structures (certificates, standard extensions, CRLs).
  - X.509 API providing cross-certification, certificate revocation, checking and supporting X.509 v3 standard extensions.
  - PKCS #7 API providing encoding and decoding of this standard enveloping format for encrypted and/or digitally signed data - already mentioned above.

4.10.3 Proposed Solution

It is currently unclear which of the vendors offer products/APIs that correspond to our needs. The following steps need to be conducted in order to come up with a solution:

- Define the group of vendors to be contacted regarding WSI APIs. This should be done together with other UBS-units dealing with Web security.
- Contact each of the selected vendors to obtain evaluation copies of their WSI-API products.
- Conduct an extensive comparative analysis of WSI-API suites from the selected vendors.
- Use the results to decide on the vendor whose WSI-APIs should be used as. It is probably easier to chose one single WSI-API vendor than several, e.g. one for each API. The selected vendor's offering must nicely integrate with the UBS PKI solution.

- Determine which additional UBS-specific abstraction layers should be built.
- Build these layers; integrate them with the selected vendor's API offering.

4.10.4 Responsibility / Contacts

concerning SSL: Thomas Ernst ?
concerning GSS: J. Wanner
concerning Directory Server: NN
concerning SSL/IIOP/ISI: Dani Steiner ?

4.10.5 Critical Dates

4.11 OS Security

4.11.1 Discussion

It is assumed that the Intranet and Internet Web Servers are protected by UBS Firewalls and that defined monitoring and logging procedures are in place[3]. This section deals **only** with securing the OS of the physical box the web server will be hosted on.

Currently only UNIX (Solaris) is used for hosting Web Servers on the Internet or as trusted system on the Intranet. Not enough work has been done to ensure secure deployment on NT.

To provide a secure Web server everything from top to bottom i.e. from client application to low level OS kernel must be secured (see the Guidelines for Securing a Solaris box in Trusted Systems on the Intranet or on the Internet in Section 7.4.1 of the Appendix).

4.11.2 Requirements

Provide a Secure box for hosting web content and applications.

Other packages required:

- SSH Secure Shell "SSL" Package
- IPF IP Filtering Package

4.11.3 Proposed Solution

See the following Guidelines which are given in the sections 7.4 of the Appendix:

- Guidelines for securing a Solaris box in trusted systems on the Intranet or on the Internet

4.11.4 Responsibility / Contacts

Andres Schmid (SJF) and Shane Hurley (H8H), IWC1/IM8V/IT42

4.11.5 Due Dates (for critical issues only)

4.11.6 Further Reading / Relevant Links

- G.Block Security Report: http://bww.eitp.ubs.com/web/dk/private/secreview/SecurityReview.htm
- Securing ESs (see section 4.12)
- IP Filtering: http://bww.webcc.ubs.ch/people/shane-hurley/projects/secureweb/filter.dmz.htm
- Secure Shell and SSL : http://bww.webcc.ubs.ch/people/shane-hurley/projects/secureweb/ssh.html
- Corresponding Checklist is available at http://bww.webcc.ubs.ch/people/shane-hurley/projects/secureweb/notes.security.html
- Sun World Security FAQ: http://www.sunworld.com/common/security-faq.html

[3] For more information on the procedures used within UBS and on the configuration of firewalls contact GTEC.

- http://bww.webcc.ubs.ch/people/shane-hurley/projects/secureweb/secure.solaris.htm
  Secure your Solaris (Web) server Peter Galvin & Hal Pomeranz (Sun)

4.12 Enterprise Server Security

4.12.1 Discussion

The web server and its related administration server must be configured as described in section 7.5 of the Appendix when deployed on the Internet or as a trusted system on the Intranet. If CGI is to be enabled then see section 4.4.

Configuration of the Netscape Server:

The servers should run under a special user id not root and in a group reserved for this user.(e.g. user www, group www). The file permissions should be restricted to this user. The physical box shall also be secured.

4.12.2 Requirements

ES3.5.1 is available as a Solaris Package (contact D.Luca)

4.12.3 Proposed Solution

See the following Guidelines, which are given in the sections 7.5.1 and 7.5.2 of the Appendix:

- Recommended setting for a secure installation of administration server
- Recommended setting for a secure installation of enterprise server

4.12.4 Responsibility / Contacts

Andres Schmid, Shane Hurley

4.12.5 Due Dates (for critical issues only)

4.12.6 Further Reading / Relevant Links

- http://bww.webcc.ubs.ch/people/kevin-gerard/projects/haws/AS_Configuration.htm (K.Gerard)
- http://bww.webcc.ubs.ch/people/kevin-gerard/projects/haws/ES_Configuration.htm (K.Gerard)

4.13 File Upload to Web Server

4.13.1 Discussion

Web content should be published - regardless of the publishing tool - to a central staging area.

The content should be tested and approved at this staging area before being replicated to a hosting area. Clients should then access the data at this hosting area. This hosting area will be secured against attack (see above sections on OS and ES security). In the case of the Internet this will be in the DMZ where the content will be replicated to many servers to provide load sharing and failover.

The web content at the central staging area should also be protected. Restrict access to this box to a limited number of publishers.

4.13.2 Requirements

4.13.3 Proposed Solution

4.13.3.1 Intranet

Single Files

`ftp`

`put` over http

`put` over https

`suftp`

Multiple Files

`rdist`

`sdist (rdist` over ssl)

4.13.3.2 Internet or Trusted Systems Intranet

Single Files

`put` over https

`suftp`

Multiple Files

`sdist` (`rdist` over ssl)

4.13.4 Responsibility / Contacts

S.Hurley

4.13.5 Due Dates (for critical issues only)

4.13.6 Further Reading / Relevant Links

- File upload using NSAPI
  http://developer.netscape.com/docs/examples/nsapi/fupload.html
- File upload element on an HTML form

http://developer.netscape.com/docs/manuals/communicator/jsref/txt3.htm

- File upload using server-side Java
  http://developer.netscape.com/docs/examples/java/file_uploading.htm

5 A Proposal for an Architecture

Taking all the security-relevant components into account we developed the following security architecture for Web-based systems (see figure 1):

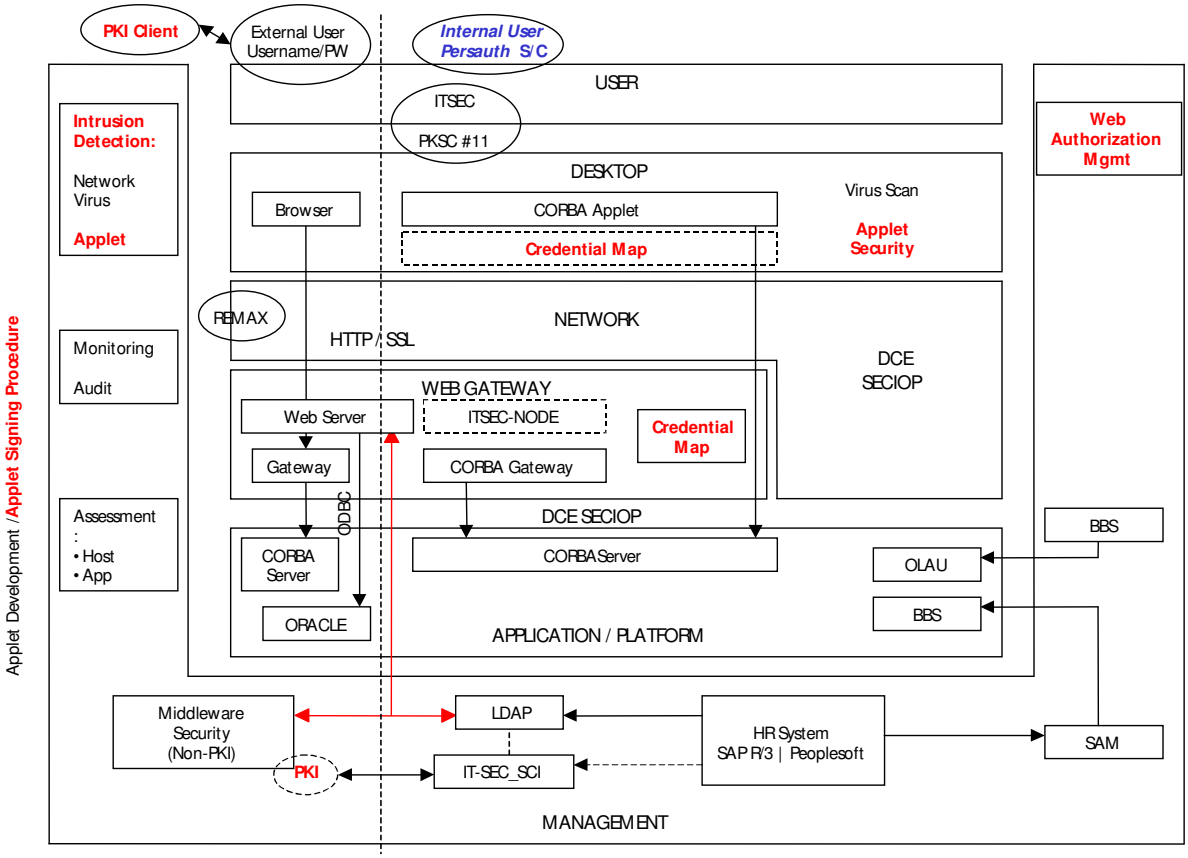


Figure 1: A proposal for a Web Security Infrastructure

This figure integrates components that are part of the current UBS system architecture. Components which are relevant to ensure secure components are highlighted where they are either not fully consolidated (*italicized*) or entirely missing (**bold**). In the following we will focus on these missing components; for the other parts of our architecture we refer to documents like e.g. the UBS reference architecture (see http://bww.ubs.ch/logi/cios/orki/orse/dsa/app_dev/index.html).

5.1 Credential Mapping

- This is a dictionary-like facility that supports mapping of IDs between the PKI environment and the middleware, e.g. mapping between DNs and CORBA Principals.
- Credential mapping might be regarded as part of the middleware, e.g. as service provided by the middleware (see the Public Key Infrastructure RFP - ec/98-11-03, by the OMG).
- There exists a *DCE 1.2.3 Public Key Certificate Login - Functional Specification* by the OpenGroup from July 1998 that concentrates on many issues that are of relevance in case of adopting a DCE-based middleware along the lines of CB Series.
- The functional specification mentioned above supports a security model that suggests using the PKI (X.509v3 public key certificates) for authentication, and DCE for

authorization. It further suggests to move the long-term user data e.g. from the DCE registry into an LDAP directory, thus consolidating the logical storage and access of PKI and DCE information. Here are the benefits:

- Basic authentication flows are made more secure by deploying public key cryptographic methods, coupled with large signature and asymmetric key-pairs.
- Massively improved scalability over PKI only deployments. Assuming C clients and S servers where every client connects to every server using SSL with client-side certificates and session establishment exchange, we would need C x S computationally expensive public key cryptographic operations. In the same scenario in which PKI is used for authentication and a middleware authentication service for obtaining shared secret keys, one ends up with only C + S public key cryptographic operations required; this leads to a massive improvement in cases where S and/or C are high.

5.2 Protection Against Malicious Mobile Code

Products by a company called Finjan Software (www.finjan.com) seem to offer the necessary components in this area. The following features are supported by Finjan Software's product line:

- Code Inspection. Enables granular management of JavaScript at the Internet gateway level, in addition to Java, ActiveX, and VisualBasic content inspection. Finjan Software claims to have a patent-pending code inspection technology that can detect malicious mobile code.
- Policy management: Unresolved control - applet profiles are entered into the (centralized) database for automatic blocking. This data base may be updated/extended by administrators.
- Protocol monitoring. FTP channels are monitored for mobile code, keeping track of code that could otherwise sneak in unnoticed from the Internet. HTTP traffic is also monitored.
- Logging. Detailed logging and mobile code reports are provided in order to support a "minimum level of care" security - often legally required in public companies.
- Multi-Browser support that allows the user to run different browsers at the same still being fully protected.
- Security configuration. Facilities in this area include:
  - File access permissions enabling Java applets and/or ActiveX controls without compromising internal security.
  - Network connection restrictions for Java applets and ActiveX controls.
  - Protection against corrupting the Windows Registry.
  - Prohibition of specific Windows System Calls, e.g. exit Windows and re-boot.
- Finjan Software has initiated a Java Security Alliance, which seems to include some of the major players in this area: RSA Data Security, VeriSign, Novell, Cisco Systems, Trusted Information Systems, etc.

5.3 Synchronizing Users

It will be necessary to synchronize user data originating from several areas:

- Middleware. This area includes:
  - Users identified by their principals - Novel and/or Unix users carrying ISI/CORBA principals. These may reside in an DCE Registry if DCE-based middleware becomes strategic (e.g. CB Series).

  - User data managed by the current authorization mechanisms (ex-SBC: BBS, ex-UBS: OLAU).
  - User data managed by HR (SAP R/3, PeopleSoft)
- Web Servers. This area includes:
  - External users coming through Internet Web servers, whose data are kept in an external (LDAP) directory.
  - Internal users coming through
  - Intranet Web servers, whose data are kept in an internal (LDAP) directory.
- PKI
  - External users, certified by e.g. VeriSign, Entrust
  - Internal users, certified by an UBS CA
- Others, e.g. (Oracle) DB users.

5.4 LDAP Issues

The rich functionality of the X.500 information model is widely recognized. The integration of a PKI directory with other applications seems to offer some significant advantages, e.g. for the integration of a white pages telephone directory with the PKI or for the directory synchronization with the email infrastructure.

Listed below are some important issues that should be looked at when planing a distributed, scaleable directory infrastructure that supports smooth integration with existing applications:

- **Interoperability**

  There are several reasons why directory servers - possibly from different vendors - may have to communicate with each other:
  - It is easier for administrators to manage one single point of access to the directory than to have multiple access points to multiple servers. A single point of access has several advantages:
    - Distribution of data is hidden from users and easier/flexible to manage.
    - Standard access with alternative protocols may be controlled from a central gateway.
    - Security controls (firewalls, proxy gateways) may impede multiple access points.
  - Data replication may boost the performance by allowing queries to be resolved locally, rather than performing an expensive/lengthy connection to the remote server. Data replication may also be useful in providing controlled subsets of information to secondary servers thus protecting sensitive corporate data.

    Replicated information should be able to provide the same security checks as those configured by the initial supplier.

    Directory replication may be done by a batch and/or off-line mechanism without involving inter-server communication. Automated replication scheduled as frequently as needed or immediately upon update provides more accurately synchronized data. This is very important when certificate information data is replicated.
  - In cases where the distribution of data among multiple servers is warranted, inter-server communication allows sharing of information that controls the distributed behavior of the directory. This might allow servers to update each other's knowledge references, replication agreements, access control changes, or even changes in the DIT structure.
- **Scalability**

A scalable directory infrastructure must accommodate both the size of the user community, i.e. the number of clients accessing the directory - and the size of the data and the number of entries in the directory. This involves

- A Directory Information Tree (DIT) being distributed between multiple servers; this implies inter-server communication for the purpose of data sharing, data replication, and resolution of user requests.
- A standardized data administration model to be shared between servers.
- A standardized access control mechanism. LDAP does not currently provide a standard access control model.

If the distribution involves WANs/Internet, additional security issues should be regarded (see below)

- **Information model - Schema**
  - DIT – a tree-like hierarchical structure. A DIT structure contributes to each entry's unique Distingushed Name. A DIT may be optimized for the management of information distribution between servers.
  - OO Entry Definition - object type and content of each entry are determined by object class hierarchies to which it belongs. Auxiliary object classes allow the dynamic addition/removal of attributes.
  - Flexibility - content rules that may be used to define combinations of object classes and optional attributes in entries.
  - Attribute hierarchies - attributes sharing common semantics may be grouped together. Queries on supertypes may deliver information involving subtypes.
  - Collective attributes - may be used to assign common attribute values to entire subtrees of entries. This eases administering information that spans subtrees.
  - Administrative areas - allows the DIT to be partitioned into different administrative areas covered by distinctive administrative authorities.
  - Schema distribution - allows the server to publish information about its schema to other servers and users. This implies standard operational attributes to describe the schema and a distributed model for the administration of the schema.
  - Contexts - tag attribute values with contexts so that the selection of appropriate values may be determined by e.g. language or geographic context. This corresponds to LDAP's Attribute Descriptions.
- **Security**
  - Access controls - specifying who can access what information for which purpose.
  - Authentication - distributed authentication should allow all servers to cooperate to validate the identity of the original user.
  - Secure storage - anything from secure physical storage to storing data in encrypted form.
  - Protocol security - involves digital signing for authenticity and data integrity as well as encryption for confidentiality.
- **Performance and Availability**
  - It is important to be able to estimate the maximum load of the server based on some metric data provided by the vendor in order to determine the size and the configuration of the system.
  - Usage profiles/scenarios should be available, so that the quality of service parameters may be negotiated with vendors/service providers.
  - Availability involves:
    - HW and SW Mean Time Between Failure

    - Service provider committing to well-defined disaster recovery plans.
- **Search Functionality**
    - Support for equality matching rules: "begins with" (mandatory), "ends with", "contains." The last two may be difficult to integrate with existing indexing schemes.
    - Approximate matching rules (e.g. phonetic matching.) These are not yet standardized.
    - Support for extended character sets.

6 Open Issues

6.1 General Remarks

- There is no fully Smart Card-Based solution with delegation.
- There are currently no commercially available PKIs supporting delegation. There have been some prototype implementations, e.g. by DEC.
- The Common Data Security Architecture proposal by Intel will probably not be fully supported by major players in the near future, in spite of commitments/promises.
- PKIX compliance might be expected sometime in 1999.

6.2 PKI Related Issues

- There must be an architectural bridge between the PKI and the bank's CORBA/IIOP-based middleware. This involves:
    - Mapping between the external Certificate-based IDs (e.g.: Distinguished Names) and the middleware principals
    - Synchronization mechanisms with the directory services (e.g.: LDAP).
    - Open Group has proposed Credential Mapping API for middleware security (http://www.opengroup.org/tech/dce/tech/pki/RFC68_4_d06.pdf). See the discussion below.
- Non-repudiation may be achieved internally without using the PKI

7 Appendix

7.1 Public-Key Infrastructure (PKI)

7.1.1 PKI Architecture

7.1.1.1 Certification Authorities (CAs)

Certifcation authorities

- Must be trusted
- Create and issue certificates for a finite community of users (security domain)
- Maintain and publish Certificate Revocation Lists (CRLs)
- Take over two distinct roles:
  - creation, signing and management of certificates, e.g. an off-the-shelf certification server.
  - issue certificatesas formal auhority. In this context, a CA will need to verify the identity of the person requesting the certificate and set the policy for doing so.

7.1.1.2 Registration Authorities (RAs)

- Optional component of PKI. Subordinate of CA to which it delegates management functions, e.g. personal authentication tasks, reporting of revoked certificates, etc.
- RAs are useful in distributed environments for creating implementations that are highly scalable - they allow large organizations to distribute functionality over the network.
- RAs operate on LANs under the auspices of a single CA:
  - RAs have a distributed functionality
  - RAs are end-entities ,which the CA and its clients must trust.
- CA must certify RAs, which means that the security loop is lengthened due to increased complexity

7.1.1.3 PKI Repositories

The PKI architecture relies on two repositories:

- A private repository for backing up current keys and for archiving out-of-date keys. This repository operates under the same security provisions as the CA itself. This may be any kind of repository
- A public repository for storage and distribution of both certificates and CRLs. The most logical choice for this repository is a directory service based on LDAP.

7.1.1.3.1 Directory Integration

Requirements the PKI has with respect to the directory service:

- Creation, management and reading of directory entries relevant to PKI components, not just certificates
- Directory replication:
  - Automatic propagation of data to replicas of the directory database anywhere in an organization.
  - Support of multimaster replication by directories that support PKIs.

- Extensible directory schema. Much remains to be done in standardizing schema and other issues associated with certificate storage and retrieval.

7.1.2 Architectural Issues in Implementing and Using PKIs

7.1.2.1 Certification Hierarchies

- The PEM specification (RFC 1422) describes a hierarchical model in which CAs as formal agencies create paths/chains of trust.
- Users relying on different CA can create trust relations in several ways:
  - CAs can certify each other
  - One CA signs the certificate containing the public key used by the other CA.
- This is only practical in relatively small numbers.
- Starting point for most organizations as they begin early PKI deployments.
- The implementation of transitive trust in certification hierarchies (certification path processing) implies:
  - Only certification hierarchies consisting of multiple CAs can address scalability issues.
  - If users find the CA that they both/all trust, they can use CA certificates to confirm validity of each other's public keys.

7.1.2.1.1 PEM and Strict Hierarchies

- RFC 1422 proposes rigid hierarchies with a central certifying authority that every certificate-using system must ultimately trust.
- RFC 1422 defines a rigid hierarchical structure consisting of 3 types of CAs:
  - Internet Policy Registration Authority (IRPA) is the root of the hierarchy. All certification paths start with the IRPA which is operated by the Internet Society.
  - IRPA issues certificates only for Policy Certification Authorities. PCAs comprise the 2-nd level of the hierarchy.
  - CAs are at the subsequently lower levels of the hierarchy, e.g. level 3 CAs are certified by the PCAs, level 4 CAs are certified by level 3 CAs, etc.
- A Name Subordination Rule says that a CA can issue certificates only for subordinate entries in the X.500 naming tree.

7.1.2.1.2 Problems with Hierarchies

- Main objection: hierarchical CAs are too complex
- High unlikeliness that everyone needing a certificate will trust one imposing "big brother" authority.
- Direct trust relationship (cross-certification) avoids the overhead of traversing the hierarchy to accomplish simple transactions. Departments and divisions within corporations will cross-certify each other for intra-company communication (bottom-up approach).
- Management of trust should preferably not be taken out of a company or organization.
- Local control makes key distribution and update operations easier because customers can control them locally. Certification paths for users within a company should start with a CA in the user's own domain, not at the top of a hierarchy controlled by a third party.

- "Trust by consensus" (two parties trusting each other because they both trust a third party) is impractical for some applications. Stricter, more direct, confirmations of trust may be necessary for high-dollar transfers.

7.1.2.1.3 More Flexible Hierarchies

- The PKIX working group has proposed a more flexible CA structure; e.g.: X.509v3 certificates support extensions that can communicate policy-related data directly to users of certificates.
- More flexible hierarchies raise new problems: crossing the boundaries of a can cause problems related to the matching of different certificate usage policies, certificate life-cycle policies, CRL policies, etc.
- The Simple PKI working group (SPKI) has proposed to eliminate hierarchies.

7.1.2.1.4 Reality

- The Business community will not submit to a strict hierarchy of any sort.
- Companies and system users will participate in different security domains, often with different keys and certificates, depending on their role.

7.1.2.2 Certificate and Key Management

- Public key cryptography does not solve any of the key management problems.
- Certificate and key management are the most important issues using PKI systems.
- Users will have multiple keys; at least one pair of keys for digital signature and the other for data encryption.
- Users may also have different keys for different applications, based on levels of trust and authority.
- Management issues:
  - Employees leaving the company (key revocation)
  - Recovering data encrypted with lost keys
  - Keys used to sign sensitive documents must be available for indeterminate lengths of time

7.1.2.3 Certificate Management (CM)

- CM is the administration of certificates that validate a public key and the CRLs used to ensure the validity of a certificate.
- Functions of CM:
  - Certificate generation
  - Certificate revocation / CRL creation and maintenance (biggest problem)
  - Creation and management of trust relations between CAs

7.1.2.3.1 Certificate Revocation

- Enterprise customers must consider the use of CRLs an important part of their PKI. Ongoing use of CRLs is one of the hardest management problems associated with PKIs.
- Caching of CRLs increases performance and supports off-line use; CRL distribution points must be checked.
- CRLs pose a problem of scalability over time, as they grow in size. Version 2 of the CRL format allows the CAs to partition the certificate lists into pieces that are more manageable. CRLs may be partitioned for different kinds of revocations.

- CRL size is directly dependent on the length of time a CA maintains a revocation record - in the financial sector this may be quite long.

7.1.2.3.2 Online Verification

- Latency problem: the infrequency of the CRL publication may be a problem in E-Commerce and in legal transactions.
- The PKIX working group proposed to use online revocation methods as an alternative to X.509 CRL. Applications contact the CA directly to verify the validity of a certificate, thus eliminating the latency problem.
- Online revocation raises the following security and performance issues:
  - The Application must trust the online validation service
  - Scalability of the server providing the online service becomes an issue.

7.1.2.4 Key Management

Key management is defined as the secure administration of keys.

7.1.2.4.1 Key Generation

- Some PKI products include key generation and management services along with CA services.
- Client initialization, i.e. secure transmission of keys, is an important issue, requiring secure channels or a special distribution mechanism.

7.1.2.4.2 Key Backup and Recovery

- Key must be backed up (key lost, no sole possessor of critical master keys, certificates must be available over the life of a contract)
- Minimal enterprise-level PKIs should support secure key storage, backup and recovery.
- Enterprise-ready PKIs should also allow users to recover their keys and set new passwords without the administrator having to come into contact with the key pair (e.g. Entrust/Manager).
- Each user will have at least two pairs of keys, one for digital signature, and one for data encryption. The signing key pair usually does not have to be backed up.
- Key life-cycle issues must be resolved:
  - How long are the keys and certificates valid?
  - How long are copies of revoked keys and certificates maintained?

7.1.2.4.3 Key Escrow

- The idea of key escrow for purposes of law enforcement (US federal government) raises many legitimate privacy issues.
- Enterprise customers should be concerned with the key escrow issue as part of key backup and recovery plans. The options for possible agency types are:
  - Third party escrow agency that an independent authority has certified.
  - Internal independently operating agency within the organization.
- Any escrow effort can compromise the security of a system by introducing another process or person into the security chain. Key escrow might be more a business than a technology issue.

7.1.2.4.4 Key Update

- Company policies shape security in general and in particular, the use of PKIs. They determine how often users should get new key pairs in order to ensure the ongoing security of data.
- Updating keys for large number of users can be a huge task. This should be automated without directly involving the administrator.
- Old key pairs/Key histories must be kept/maintained, e.g. for decrypting data encrypted with old keys.

7.1.2.5 Scalability

Scalability should be evaluated on multiple criteria:

- Computational Performance
  - Public key cryptography is more scalable than its secret key rival, since managers do not have to generate keys as often.
  - Key-pairs take longer to generate
  - CRL mechanism creates its own storage and performance demands.
- Manageability, cost of ownership and ease of use
  - PKI creates work
  - PKI's ability to automate mundane tasks affects scalability significantly.
- Mathematical model to estimate the size of user community a PKI can support:
  - Model for PKI's utilization of network resources
  - Model of end-user's actions
- Assumptions:
  - WAN connecting LANs; end-users on specific LAN segments
  - WAN and LANs are data communication resources
  - WAN is dedicated to nothing but PKI, which is a dangerous assumption.
- RAs are principally human resources
- Deployment of several techniques designed to increase scalability, e.g. replication of the master repository in each of the LAN segments, local caching of the CRLs.

Results:

- CA is the limiting factor for scalability; it may be improved by adding CAs; CAs may be then either replicated or used to create hierarchical trust relationships.
- The management burden that hierarchical certification can create is often overlooked. A paper by Entrust claims that in a network with only one CA the need to renew certificates will limit the scale of a single CA to ~ 1000000 users.

7.1.3 PKI Standardization Efforts

- Interoperability requires standards
- PKI Task Group (Open Group: X/Open + OSF) has defined a set of requirements for interoperable PKIs.
- RSA Data Security: PKCSs
- Internet standards community (produced only drafts until now)
- PKIX (IETF Public Key Infrastructure working group):
  - Aimed at developing Internet standards for interoperable PKIs based on X.509.
  - Proposals have considerably modified the PEM architecture to increase its flexibility
  - A set of specifications is available (PKIX specifications).

- SPKI (Simple PKI) working group
  - Proposed a much simpler model.
  - Defined a bare-bones certificate model that reduces the X.509's complexity - similar to the way credit cards are used today.
  - The model is based on capability and not identity (the user either has the card or not)
- Best-case scenario: The industry is at least 18 months to 2 years away from significant implementation of PKIX proposals (July 1997)

7.1.3.1 Profiles

- Standards do not meet the security needs of all applications and systems.
- Security profiles are necessary since different combinations of technologies and features can serve different purposes.
- Security profiles define standard combinations of technologies and features for a particular entity:
  - Application specific security profile
  - Policy based security profile
  - Multiple choices will be possible within a profile; the negotiation of technologies between systems, based on standard profiles will be necessary.

7.1.4 PKI Standards

7.1.4.1 Digital Signatures

- Foundation element of the PKI.
- RSA's PKCS #7 is the foundation for digital signatures. Most vendors use it.
- Federal government's DSA/DSS - competes with PKCS #7
- For now, vendors should probably support both

7.1.4.2 Certificate and CRL Formats

- Certificates must be usable by applications and systems from multiple vendors.
- Interoperability depends on the use of a specific subset of what X.509 makes possible.
- IETF must set a standard for use of the X.509 certificates and CRLs; the PKIX working group has proposed such a standard: "X.509 Certificate and CRL Profile".

7.1.4.2.1 Certificate Profile

- Defines a profile for public key certificates in order to enable interoperability between systems that use certificates.
- Profile includes basic certificate fields from X.509: subject to whom the certificate was issued, subject's public key, name of the CA, digital signature of the CA, the algorithm used for signing, certificate version number, dates of validity, and the certificate serial number.

7.1.4.2.2 Certificate Extensions Profile

Critical extensions (must be supported):

- Basic Constraints – identify whether the subject of the certificate is a CA and the depth of the certification path.

- Key Usage - defines the purposes for which people and applications can use the key contained in the certificate.
- Certificate Policies - define the policy under which the CA issued the certificate and the purposes for which applications and system users can use the certificate.
- CRL and CRL Extension Profile

Possibly critical extensions:

- Alternative subject name - allows the CA to bind additional entities to the subject of the certificate. If the subject filed is empty (e.g. in a role-based certificate), systems might use the alternative name extension to mark it as critical
- Issuer alternative name - allows the CA to associate other Internet identities with the certificate issuer.
- Name constraints - define directory subtrees that place restrictions on names that may be included within the certificate issued by a given CA.
- Policy constraints - CA can use the extension to prevent applications from circumventing policy by mapping the certificate to other policies

Non-critical extensions:

- CRL distribution point extensions - define how applications and other CAs obtain the CRL applicable to the domain of users in which the CA issued the certificate.
- Subject Key Identifier - allows a certificate to identify a specific public key to be used with an application
- Private Key Usage Period - allows a certificate to specify the valid time period for the keys used in digital signatures
- Policy Mappings - allow certificate users to associate CA policies with their own internal policies

7.1.4.2.3 CRL and CRL Extension Profile

- Required and optional fields for CRLs are defined.
- Support for the CRL Number Extension is also required in order to allow users and applications to determine when the particular CRL supersedes another CRL (CRL versioning).
- Issuing Distribution Point Extension is a critical extension that identifies the place where a CA publishes a CRL.
- Delta CRL Indicator Extension - identifies a delta-CRL, which provides a list of only those changes that have occurred since the CA published the last CRL.

7.1.4.2.4 Certificate Path Validation

- Certificate path processing verifies the binding between the subject and the subject's public key.
- Standard specification does not explicitly define a process such as path validation.
- Applications/Certificate-using systems may implement this in a variety of ways - this may be essential to performance and manageability.

7.1.4.2.5 Algorithm Support

The PKIX proposal defines:

- One-way hash functions (message digests): SHA-1 (preferred), MD2
- Digital signature algorithms: RSA, DSA
- Uniform identifiers for each of the algorithms
- The proposal does not require the use of specific algorithms

7.1.4.3 Certificate Retrieval Protocols

- PKIX has proposed basic protocols for certificate retrieval (Operational Protocols Draft).
- LDAP seems to be an ideal mechanism for retrieving certificates and CRLs.
- Two scenarios:
  - Repository Read: requesting client knows the name of the certificate or CRL
  - Repository Search: requesting client does a search for a certificate or CRL based on its attributes
- Both scenarios are well within the bounds of most LDAP implementations.
- CAs communicating with each other may need access to information other than publicly available certificates and CRLs ; CA-to-CA communication requires strong authentication.

7.1.4.3.1 Online Certificate Status Protocol

- Defined by PKIX for obtaining certificate revocation information on-line
- OCSP enables applications to determine the validity and revocation state of a certificate.
- Application issues a status request to the CA and suspends further certificate acceptance processing until the CA responds with status indication.
- OCSP over HTTP implementation is defined by the PKIX

7.1.4.4 Certificate Management Protocols

- PKCS #10
- Describes syntax for certification requests
- PKIX Proposal, Part III defines message formats and protocol implementations for:
  - Registration
  - Certification
  - Key pair recovery
  - Key pair update
  - Revocation requests
  - Cross-certification
- Most products have not yet reached the level of sophistication that the PKIX proposal requires.

7.2 Guidelines for JavaScript Security

In general

- When your script is finished check that it behaves as expected with all browsers deployed in your customer base.

7.2.1 Client Side JavaScript

- **Do Not Reveal Sensitive Information**

  Do not reveal network info, UBS info, or methodology in scripts. (see section 1)

- **For Signing JavaScript scripts (Object Signing):**
  - ***Only use Communicator 4.x*** which properly performs origin check on named forms and also closes the src_from_non_file security hole. Navigator 3.0 provided data tainting to provide a means of secure access to specific components on a page. Because signed scripts provide greater security than tainting, tainting has been disabled in Communicator 4.x.

    Users shouldn't set this preference to true:

    `user_pref("javascript.allow.file_src_from_non_file", false);`

    It opens a security hole, which is open in Navigator 3.0: when you use

    `<SCRIPT SRC="...">` to load a JavaScript file, the URL specified in the SRC attribute could be any URL type (e.g. file, http), regardless of the URL type of the file that contained the SCRIPT tag.

  - ***Assume the weakest script's authority (for SSL Servers and Unsigned Scripts)***

    Unsigned scripts will act as though they were signed. Communicator treats all pages served from a SSL server as if they were signed with the public key of that server. You do not have to sign the individual scripts for this to happen.

    When JavaScript scripts produced by different principals interact, it is much harder to protect the scripts. Because all of the JavaScript code on a single HTML page runs in the same process, different scripts on the same page can change each other's behavior. For example, a script might re-define a function defined by an earlier script on the same page.

    To ensure security, the basic assumption of the JavaScript signed script security model is that mixed scripts on an HTML page operate as if they were all signed by the intersection of the principals that signed each script.(i.e. they all assume the weakest scripts authority)

  - ***Check Principals for Windows and Layers***

    In order to protect signed scripts from tampering, Communicator 4.0 adds a new set of checks at the container level, where a container is either a window or a layer. To access the properties of a signed container, the script seeking access must be signed by a superset of the principals that signed the container.

    These cross-container checks apply to most properties, whether predefined (by Communicator) or user-defined (whether by HTML content, or by script functions and variables). The cross-container checks do not apply to size and layout properties of window.

  - ***Do not Grant Privileges to External Scripts***

    UBS Users should not grant (or should be blocked from granting) these privileges to external scripts. May be a white list like for external Java Applets should be introduced.

    A script can request access to the file system the registry the screen and trace key input from the user. These all pose a very high risk to the users system. The

normal non-IT user should be blocked from granting access to external scripts via mission control. The firewalls and proxies cannot protect against this risk if the script is hosted on a https server. The client is left with the decision on whether to grant access or not.

There are over 100 different privileges a script can request, see

http://developer.netscape.com/docs/manuals/signedobj/targets/index.htm

for a complete list of targets/privileges and their risk level.

Privileges are granted only in the scope of the requesting function. This scope includes any functions called by the requesting function. When the script leaves the requesting function, privileges no longer apply.

Use the minimal capability required for the task in order to reduce your exposure to exploits or mistakes.

- **Disable Code Base Principals**

  As does Java, JavaScript supports code base principals. A code base principal is a principal derived from the origin of the script rather than from verifying a digital signature of a certificate. Since code base principals offer weaker security, they should be disabled and are disabled by default in Communicator. Scripts should not rely on code base principals being enabled. With code base principals enabled, when the user accesses the script, a dialog displays similar to the one displayed with signed scripts. The difference is that this dialog asks the user to grant privileges based on the URL and doesn't provide author verification. It advises the user that the script has not been digitally signed and may have been tampered with.

  ```
  user_pref("signed.applets.codebase_principal_support", false);
  ```

- **Use a Secure Server (JavaScript and mission control)**

  It is unclear whether UBS systems should

  - accept signed scripts from outside
  - host signed scripts

  An alternative to using the Page Signer tool to sign your scripts is to serve them from a secure server. Communicator treats all pages served from an SSL server as if they were signed with the public key of that server. You do not have to sign the individual scripts for this to happen.

- **Do not export sensitive functions**

  Exporting a function name makes it available to be imported by scripts outside the container without being subject to a container test. You are in effect transferring any trust the user has placed in you to any script that calls your functions. UBS signed scripts should not export sensitive functions as hostile scripts assume all rights assigned to that function.

- **Prevent porting**

  By default another user may host your signed script and use it under your signature. In order to prevent this, you can force your scripts to work only from your site.

```
<SCRIPT ARCHIVE="siteSpecific.jar" ID="a" LANGUAGE="JavaScript1.2">
if (document.URL.match(/^http:\/\/www.ubs.com\//)) {
   netscape.security.PrivilegeManager.enablePrivilege(...);
   // Do your stuff
}
```

```
</SCRIPT>
```

Then if the JAR file and script are copied to another site, they no longer work. If the person who copies the script alters it to bypass the check on the source of the script, the signature is invalidated.

- ***Minimise the Trusted Code Base***

  In security parlance, the trusted code base (TCB) is the set of code that has privileges to perform restricted actions. One way to improve security is to reduce the size of the TCB, which then gives fewer points for attack or opportunities for mistakes.

7.2.2 Server-Side JavaScript

See section on CGI most of which applies to Server Side JavaScript. The Netscape JavaScript Application Manager improves security via access control and passwords. All steps should be taken to prevent the SSJS-program from crashing the web server, i.e. run the SSJS out of a process or on a separate box. Test the script against attacks listed in the CGI section.

7.3 Guidelines for the Web Access through CGI/Perl

7.3.1 Guidelines for Program Development, Test & Installation

- Read the FAQ http://www.w3.org/Security/Faq/wwwsf4.html concerning CGI security problems
- All input to the program (compiled or not) whether from client, environment or scripts **must** be checked for **:**
  - maximum length: (prevent buffer overflow) even restricting to 600 chars will help.
  - content: only allow characters which you expect e.g. `$OK_CHARS='-a-zA-Z0-9_.@';` This avoids hackers inserting hidden commands into input.
- **Never** pass this data to a shell command without the above checks.
- Beware the `eval`-statement: Languages like PERL and the Bourne shell provide an `eval` command which allows you to construct a string and have the interpreter execute that string. This can be very dangerous.
- Be careful with `popen` and `system`. If you use any data from the client to construct a command line for a call to `popen()` or `system()`, be sure to place backslashes ahead of any characters that have a special meaning to the Bourne shell before calling the function.
- If the input fails any of the above checks **don't** publish the CGI program on your web server.
- Use a CGI Wrapper (e.g. sbox) which prevents crashing CGI programs from giving a shell back to the remote user.
- Statically compile all programs accessed via CGI; do not publish the code on the web as this makes it much easier to hack the code.
- Never use a sample counter clock etc. downloaded from the web; they are all vulnerable as hackers have had a lot of time to find holes in the code.
- Test all CGI programs against the known attacks (see section 1.4).
- Restrict all CGI programs to a CGI-bin directory and allow only an administrator to install programs there. Check the file permissions.
- Do not display or give system information to the remote user.(e.g. IP, paths, username)
- Host CGI programs on a Unix server preferably not the same server as the static content. NT was cracked by adding a space to the end of the ".cgi " and many other ways.
- Turn off server-side includes for your script directories. The server-side includes can be abused by clients which prey on scripts which directly output things they have been sent.

7.3.2 Guidelines for the Web Server Admin:

- secure the box against other attacks to the file system or to block ftp telnet etc.

7.4 Guidelines for OS Security

7.4.1 Guidelines for Securing a Solaris Box in Trusted Systems on the Intranet or on the Internet

This information is kept updated at
http://bww.webcc.ubs.ch/people/shane-hurley/projects/secureweb/notes.security.html

The box must not be connected to the network until it has been secured.

All ports should be blocked using IP Filtering except the subset of the following that will be used

80	HTTP
8080	HTTP Proxy(or whatever is defined)
443	HTTPS
444	Administration Server
636	SLDAP Secure Directory
389	LDAP Directory
22	SSH, SDIST & SCP over SSL
37 out	NTP (network time protocol)
25 out	SMTP Mail
?	configured Search Engine Port or other required services

CERT

The system administrator should be on the mailing list (cert-advisory-request@cert.org) for security bug reports from the CERT Co-ordination Center, which periodically issues the CERT Summary. This document draws attention to the types of attacks currently being reported to their incident response team, as well as to other noteworthy incident and vulnerability information.

Operating System Configuration

This deals with core OS issues and the configuration of the native operating system installation. All systems should be built from a clean system installation and formatted drives; no previous software installations of any kind should exist, other than the minimal native OS installation, on any system destined for the DMZ. Steps taken to build DMZ systems should be clearly documented, repeatable, and verifiable through automation to ensure that configurations have not changed and applications have not been added.

See also http://bww.eitp.ubs.com/web/dk/private/secreview/SecurityReview.htm

Configuration Management

The following operations should be performed on any system within the DMZ, and the permissions of these files checked for improperly set world/group/owner read/write/execute capabilities:

- `/etc/services` should be rebuilt to contain only those services which are a part of the applications required by the system for its day-to-day and management usage. It should be set to 644 and owned by `root/wheel`.

  ```
  chmod 644 /etc/inet/services
  ```

- /etc/inetd.conf should be removed, if empty; otherwise, it should be pruned down to the smallest possible set of daemons required to run the service. It should be set to 600 and owned by root/wheel.
  chmod 600 /etc/inet/inetd.conf
- /etc/passwd should be rebuilt from scratch to contain a minimum set of users to run the service and manage the system. Create an administrative user with the ability to su to root. All users should have a valid home directory other than the root partition, including root.
- /etc/group should be rebuilt from scratch to contain a minimum set of groups to run the service and manage the system. If it doesn't exist already, create the wheel group for root and other high-level administrative accounts to belong to.
- .rhosts files anywhere on the system should be deleted. In all home directories, an rhosts file, owned by root, set to 400, and length of 0 should be written to prevent the file from being created or modified.
  find / –name .rhosts –print
  cd <home dir>
  type > .rhosts (make an empty hosts file)
  chmod 400 .rhosts
- /etc/hosts.equiv should be a zero-length file, owned by root, set to 400. This file should never be used.
  type > /etc/hosts.equiv (make an empty hosts file)
  chmod 400 /etc/hosts.equiv
- /etc/default/login, which is ideally unnecessary, should still be checked to see that it is root owned, set to 644, and checked to verify that login will prevent root login on anything except the console.
  chmod 644 /etc/default/login
- Sendmail and all related components should be obliterated. If mail-send functionality is required, simple commands which send mail via SNMP, but which have no local deliver functionality, are available.
- etc/aliases/ and all sendmail-related files should be deleted.
  rm /etc/aliases (for mail from forms some sendmail functionality is needed)
  mail requirements need to be discussed. May be necessary for outgoing reports.
- UUCP's L.cmds, and indeed any configuration files or commands related to UUCP, should be removed.
- netrc, .exrc, and .forward files in any home directories (including the root partition) should be zero-length files, set to 400, owned by root/wheel.
  find / –name .netrc –print
  type > .netrc
- (make an empty hosts file)
  chmod 400 .netrc (make root read only)
  find / –name .exrc –print
  type > .exrc (make an empty hosts file)
  chmod 400 .exrc (make root read only)
  find / –name .forward –print
  type > .forward (make an empty hosts file)
  chmod 400 .forward (make root read only)
- /etc/syslog.conf should log all information to a file that is managed, parsed, and rotated properly. Default configurations do not do this. Permissions should be set as root-owned, 644.
  chmod 644 /etc/syslogd.conf
- SSHD configuration files should ensure, along with ownership by root/wheel and permissions set to 644 that:
  FascistLogging=YES
  IgnoreRhosts=YES
  KeepAlive=YES

```
KeyRegenerationInterval=2000
PermitEmptyPasswords=NO
PermitRootLogin=NO
RhostsAuthentication=NO
RhostsRSAAuthentication=YES     (allows .shosts to define keys)
ServerKeyBits=1024              (or greater)
X11Forwarding=NO
```

- /etc/logindevperm should be set to 644 and owned by root/wheel; it should contain entries for /dev/console which set the permissions to 0600 for, at least, the following devices:

```
/dev/console0600/dev/kbd:/dev/mouse
/dev/console0600/dev/sound/*
/dev/console0600/dev/fbs/*

more /etc/logindevperm
chmod 644 /etc/logindevperm
```

Any and all commands which are not required for system operation or maintenance should be disabled (by turning off the execute, sticky, and setuid/setgid bits) or (optimally) removed entirely from the system. These include:

- R-commands such as rcp, rsh, rlogin. These can be replaced with ssh.

```
rm /usr/bin/rcp
rm /usr/bin/rsh
rm /usr/bin/rlogin
```

- Setuid and setgid applications should have their sticky bits removed, when possible, restricting them to run as root; if these commands are not needed, they should be removed.
- Portmapper and any RPC-related systems should be disabled or removed entirely, if possible. If not, secure replacements, such as Wietse Venma's portmapper daemon should be used in its place and securely configured to restrict access to the portmapper daemon.
- YP and NIS/NIS+ server and client functionality should be disabled or removed entirely. NIS/NIS+ represent an unnecessary security risk in a DMZ.
- Any and all inetd daemons and inetd itself (if not in use) should be disabled or removed entirely. At minimum, this **must** include the finger daemon.

```
finger chargen echo inetd...etc
```

- RPC's nonexistence should be verified using /usr/bin/rpcinfo –p.
- Expreserve functionality should be removed. Users who crash during an ed or vi session will not be able to recover changes before saving to disk.
- X-windows and all related components should be removed
- No development environments should be permitted on service machines. CC, GCC, any and all system header files and includes should be wiped clean.

```
which cc, which CC, which gcc
```

- FTP, telnet, and TFTP and all related components should be wiped out.

```
rm /usr/bin/ftp
rm /usr/bin/telnet          (make sure SSH is installed and working first)
rm /usr/bin/tftp
```

In addition, the following permissions should be verified. Except where otherwise listed, all files should be owned by root.

- /etc/utmp: 644
- /etc/sm, /etc/sm.bak: 2755
- /etc/state: 644
- /etc/motd: 644
- /etc/mtab: 644
- /etc/syslog.pid: 644

- The kernel should be `root/wheel, 644`.
- `/etc, /usr/etc, /bin, /usr/bin, /sbin, /usr/sbin, /tmp, /usr/tmp, /usr/local/bin,` and `/usr/local/etc` should all be owned by `root`.
- UMASKS of users should be set to 027 or 077.
- Verify that all files `in /dev` are special.
- Validate `/usr/kvm/sys/*`, if it exists, is not `group-writable`, and remove the `SETGID` bit `on /usr/kvm/crash`.

In addition, the following steps are examples of the kind of actions, which should be taken:

- Find all device files and store a list to compare with as a part of a regular security check:
  ```
  find / \ (-type b -o -type c \) -print
  ```
- Find all `setuid` and `setgid` files. Periodically verify that the list of files have not changed and that all of the files are of the same size:
  ```
  find / -perm -g+s -print
  find / -perm -u+s -print
  find / -perm -t+u -print
  ```
- Find all files without owners. This should never happen, and it should be corrected immediately.
  ```
  find / -nouser -print
  ```
- Find all files that are world-writable. This should be carefully scrutinized and watched warily.
  ```
  find / -perm -2 -print
  ```
  (also returns softlinks)

In addition, make sure that all logfiles are being monitored and rotated. Logs should be placed in their own partition, so that logging attacks cannot fill root or swap partitions and cause application failure. The following lists some of the system log-files, which may exist on your system and need monitoring

- `/var/adm/sulog` - log of the SU command
- `/var/adm/wtmp` - login accounting, date changes
- `/var/adm/pacct` - process accounting info
- `/var/adm/messages` - system messages, `sulog, cron log`
- `/var/adm/loginlog` - list of failed login attempts
- `/var/cron/log` - Log of all `cron` output

Additional Security Packages and Suggestions

Several other things can be done to improve the security of these systems.

- BIN ownership: **Note the following carefully -** change all files which **are** non-setuid, **are** non-setgid, **are** world readable, and **are not** world-writable, **and are owned by user bin** to owner root, group wheel.
- In `/etc/rc2.d/s69.inet`, add the following lines to turn off IP forwarding and forwarding of source-routed packets, respectively:
  ```
  ndd -set/dev/ipip_forwarding0
  ndd -set/dev/ipip_ip_forward_src_routed0
  ```
- Remove all graphics cards in service machines. Edit the `/etc/fbtab` to match.

This provides a list of applications to install to aid in securing, monitoring, and managing the security of systems within the DMZ.

- install SSH 1.2.22 or greater on systems that require login capability over network connections
- IPFilters should be installed (http://coombs.anu.edu.au/~avalon) and used to filter access to specific ethernet ports and connection types. Filter out, at minimum, tight

source routing, loose source routing, short IP fragments, filter outbound ethernet ports to only allow outbound traffic from the assigned IP address, lock down sshd to accept only specific addresses, lock down the Netscape Admin Server and any insecure LDAP access ports, etc.)

- limits the TCP/IP packet forwarding to a minimal set on needed communications (bit like a sunscreen). (PORTS: 80, 443 http https, 444 admin server, 636 389 ldap sldap, ssl with ssh and rdist/sdist over it, time SNTP) IPFilters should also block fragmented headers. A help file is in man(5)"ifp headers tables macros".
- ipf switched on
  `chmod 700 /sbin/ipf` (only root can start and stop filtering)

Extra Application-Level software

- Install `'top'` (who is using the cpu) and `'isf'` (list of open files)
- Run Satan like tool against machine to scan for weaknesses available at www.iss.net
- Tripwire product sees strange activity and alerts the admin (e.g. three errors in a row). It also creates log information of system activity. It can be configured to see who is doing what and establish why. It can also close down the box.
- Crawl the web content and verify it does not differ from the master (expected) copy.

7.5 Security Settings for Netscape Enterprise and Administration Servers in UBS Trusted Systems or Internet

7.5.1 Recommended setting for a secure installation of Administration Server

This should be also be used as a checklist after the installation has been completed. Each section (table) corresponds to a menu button in the topmost frame of the AS administration interface.

Admin Preferences:			
Item	**setting**	checked	**Description**
Network Settings	**Set Port to 444**		**The Admin server must be run in secure mode (SSL). See 'Encryption On/Off'.**
Administrative user Access Control	**Try to be as restrictive as possible here.**		**Example:** **Hostnames to allow: *.ubs.com** **IP addresses to allow: *.*.yyy.zzz (where 'yyy.zzz' is the IP equiv of '*.ubs.com')**
Distributed Admin	**Select 'No' for 'end user access' and distributed admin**		**Use some pre-defined group (very restrictive) if distributed admin is enabled.**
Encryption ON/OFF	**Select 'yes' for secure Administration (SSL).**		**Select a port number 636 for secure connection. The server needs a certificate issued by UBS CA.**

Encryption Preferences	**All on except "No encryption ...MD5..."**		

Global Settings:

Item	setting	checked	Description
Configure Directory Service	**Select to use or not a Directory Server**		**Recommended: use of SSL.** **Provide the optional DN and password of a user with allowed access to the Directory Server. Else it will do searches with anonymous access and this can be disabled in the directory server.**
Cron Control	**switch off**		
SNMP Master Agent Control	**switch off**		

Keys & Certificates:

Item	setting	checked	Description
List Aliases	**one row/alias combination should be defined.** **This is used to secure the various servers on the machine.**		**A row consists of:** **1) alias identifier** **2) key filename** **3) certificate filename**

Cluster Management:

Item	setting	checked	Description
Cluster Control	**This should yield a page with header 'Cluster Database is Empty'**		

7.5.2 Recommended setting for a secure installation of Enterprise Server

Each section (table) corresponds to a menu button in the topmost frame of the ES administration interface.

Server Preferences:

Item	setting	checked	Description
Mime Types	**Set 'magnus-internal/cgi' only for 'cgi' (thus remove**		**This is done to avoid having files with these extensions to be executed accidently.**

	'exe' and 'bat'		
Network Settings	**set Server Port 443**		**https (http over ssl) Select a port number (443 for secure connections).**
Error Responses	**provided by the client**		
Dynamic configuration Files	**all blank**		**This should remain disabled (default setting)**
UNIX Chroot	**No directory entry**		
Symbolic Links	**soft file links Never** **hard file links No**		**Limit Symbolic Links**
Restrict Access	**Access Control List Management**		**In section A: only 'The entire Server' should appear. by clicking 'Edit Access Control' . The 'Allowed Users' is a sample of a group which has been allowed access. Modify this to your needs.** **In section B: only 'agents' and 'default' should appear. Editing the ACL for default should yield the same as in section A above. Agents will be disabled so no worries.** **Section C: empty**
Encryption ON/OFF	**'ON'**		**The server needs to own a certificate from UBS Certificate Authority.**
Encryption Preferences	**Allow SSL Ver 2 & 3** **Require Client Cert = No** **Check RC4 and RC2 for SSL 2 and 3** **Do Not Check the No encryption MD5 option**		**No MD5 !!!** **If you wish to require client certificates, select 'yes' for that option. This implies that all the users which are granted access to your site hold a personal certificate issued by UBS CA.**

Programs:

Item	setting	checked	Description
CGI Directory	**no CGI directory**		**Leave blank**
CGI File Type	**CGI Off**		**Only one entry ('the entire server') should appear in the topmost listbox. If a CGI directory is added this will have to be updated too.**
Query Handler	**leave blank**		**Only one entry ('the entire server') should appear in the topmost listbox.**
WAI Managment	**Select 'No'**		
Java	**Select 'No'**		
Server Side Javascript	**set activate to No**		**If you do activate the SSJS environment, make sure you do either of the following:** **1) require the administration server password**

			for SSJS Aplication Manager by selecting 'yes' in that section. **2) set up a style with restrict access enabled for a selected group/user to whom you allow access. This still should then be applied to the URL Prefix 'http://xxx.yyy.zzz/appmgr'. View ES 3.5.1 help files for these operations.**

Server Status:

Item	setting	checked	Description
SNMP Configuration	**Select 'Off' (**Error! Bookmark not defined.**)**		

Content Management:

Item	setting	checked	Description
Remote File Manipulation	**Select 'No'**		
Document Preferences	**No indexing**		**Want to avoid letting users find data without direct links**

Web Publishing:

Item	setting	checked	Description
Web Publishing State	**Select 'Off'**		

7.6 APIs

7.6.1 Low-Level APIs

In the following, the CryptoAPI and CDSA are described in more detail.

7.6.1.1 CryptoAPI

CryptoAPI represents a service abstraction layer, similar to MAPI or ODBC, providing an uniform set of API calls. A Cryptographic Services Provider (CSP) handles those calls in an implementation-specific form. CSPs are analogous to drivers supporting vendor-specific implementations of services to which the CryptoAPI provides access.

CryptoAPI 1.0 supports context management functions, key storage and exchange, hashing, digital signatures, signature verification, and data encryption. CryptoAPI 2.0 adds ability to generate requests for certificates, storage, retrieval, parsing, and verification of certificates. A lot of work has been done to ensure support for X.509 and PKCS within the CryptoAPI, enabling it to work with security and authentication frameworks.

The service-provider architecture provided by the CryptoAPI makes applications independent of security service implementations.

Drawbacks of CryptoAPI are:

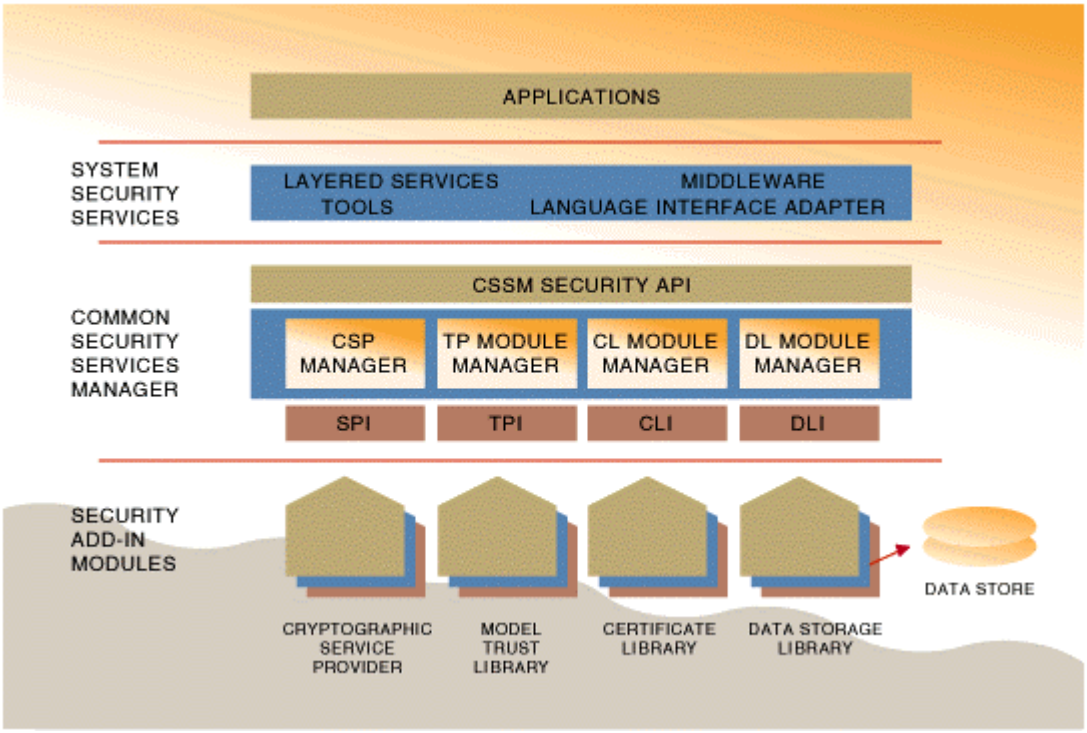
- CryptoAPI lacks full directory support (as does Microsoft)
- CryptoAPI is very Windows-centric. Microsoft representatives say that they will implement the API for other platforms. Considering similar unfulfilled promises regarding other APIs, there is little hope that this drawback will be rectified anytime soon.

7.6.1.2 CDSA

CDSA defines an overall architecture for building and deploying of security services. As CryptoAPI, CDSA is also built on a layered service-provider architecture. It defines a consistent application interface and a service layer which maps that interface to so-called security modules. Security modules represent implementations of specific services. Applications written to CDSA can work with any system for which developers provide security modules.

The Common Security Services Manager (CSSM) represents the core component of the CDSA. It defines the top-level API that applications and services use to access WSI services. Through CSSM API applications can perform diverse cryptographic operations, determine whether a certificate holder is to be trusted, manipulate certificates, or access a repository where important security data may be stored.

CSSM manages these functions by delegating tasks to a number of managers (CSP manager, TP Manager, CL Manager,...). CSSM gives applications and/or services to the specific APIs defined by each of these managers. Each of the managers defines respective APIs for service providers. System vendors write modules for their WSI products to make the services of their products available to applications through the CSSM API.



Here is a short summary of each of the security managers:

- Cryptographic Service Manager manages the Cryptographic Service Providers who provide modules performing cryptographic operations, e.g. bulk encryption, digital signatures, message digests. CSPs also provide secure key storage for private keys.
- Trust Policy Services Manager manages trust policy modules. These modules implement policies. The Trust Policy Services Manager does not deal with policies; it just provides an infrastructure for installing and managing of policy-specific modules. Applications and/or security services are thus able to perform trust operations via the CMS API, e.g. verify trust in a certificate for signing and/or revoking another certificate.
- Certificate Services Manager manages the Certificate Library. Applications are thus capable of manipulating certificates and certificate revocation lists. Modules must be available for specific certificate formats.
- Data Store Services Manager defines an API for accessing the secure repository of diverse security objects, e.g.: certificates or CRLs. Storage and retrieval of security objects may thus be directly performed by applications, through this API. Data storage modules may be gateways to traditional DBMSs, customized services layered over file systems or other forms of stable storage (e.g. LDAP).

As represented in the above figure, most applications will not use the CSSM directly, but will use lower-level toolkits and services written to it. In a majority of the cases, an abstraction layer will be necessary to make WSI services accessible to mainstream (application) developers.

7.6.2 High-Level APIs

7.6.2.1 Netscape Software Development Kits (SDKs)

The following SDKs can be downloaded from http://developer.netscape.com/software/index.html:

- Component Developers Kit (Java, JavaScript C/C++) for building crossware components which are linked together via Java JavaScript or CORBA/IIOP
- Mission Control Desktop SDK
- Directory SDK (Java, JavaScript C/C++, Perl) otherwise known as LDAP SDK
- Compass Server SDK for Compass server database
- Mail and News Messaging Access SDK
- Live Connect/Plug in SDK
- Composer Plug in SDK
- Security SDK
- Certificate Mapping SDK
- NetHelp2 SDK

8 Acronyms

ACL	Access Control List
AH	Authentication Header
API	Application Programming Interface
AS	Administration Server
ASN	Abstract Syntax Notation (???)
CA	Certification Authority
CDSA	Common Data Security Architecture (Intel)
CGI	Common Gateway Interface
CERT	Computer Emergency Response Team (www.cert.org)
CL Manager	Control List Manager
CM	Certificate Management
CMS	Certificate Management Services
CORBA	Common Object Request Broker Architecture
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
CSSM	Common Security Services Manager
DBMS	Database Management System
DIT	Directory Information Tree
DMZ	Demilitarized Zone
DN	Distinguished Name
DSS	Digital Signature Standard
ES	Enterprise Server
ESP	Encapsulated Security Protocol
EUP-D	End User Platform Domestic
FTP	File Transfer Protocol
GSS	Generic Security Service
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDEA	International Data Encryption Algorithm
IDUP	Independent Data Unit Protection
IE	Internet Explorer
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
IPF	Internet Protocol Filtering
ISAKMP	Internet Security Association Key Management Protocol
ITU-T	International Telecommunication Union - Telecommunications
JAR	Java Archive Format
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDK	Java Development Kit

JVM	Java Virtual Machine
KEA	Key Exchange Algorithm
LDAP	Light Weight Directory Access Protocol
MAPI	Messaging Application Program Interface
MD5	Message Digest Number 5
MIME	Multimedia Internet Mail Extension
MSIE	Microsoft Internet Explorer
NIS	Network Information Services
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
ODBC	Open Database Connectivity
OLAU	OpenLan Authorisation
OS	Operating System
PEM	Privacy Enhanced Mail
PERSAUTH	Persönliche Authentisierung
PGP	Pretty Good Privacy
PKCS	Public-Key Cryptography Standard
PKI	Public-Key Infrastructure
PKIX	IETF Public Key Infrastructure Working Group
RFC	Request for Comment
RA	Registration Authority
RSA	Rivest-Shamir-Adleman
SDK	Software Development Kit
SET	Secure Electronic Transactions
S-HTTP	Secure HTTP
SLDAP	Secure LDAP
S/MIME	Secure MIME
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNTP	Simple Network Transfer Protocol
SPKM	Simple Public Key Mechanism
SSH	Secure Shell
SSHD	Secure Shell Daemon
SSJS	Server-Side JavaScript
SSL	Secure Sockets Layer
TCB	Trusted Code Base
TCP/IP	[Transmission | Transfer] Control Protocol/Internet Protocol
TFTP	Trivial FTP
TP Manager	Trust Policy Manager
URL	Uniform Resource Locator
UUCP	UNIX-to-UNIX Copy Protocol
WAI	Web Application Interface
WSI	Web Security Infrastructure
WWW	World-Wide Web
YP	Yellow Pages