Summary of security concerns regarding Internet applications

Author(s):	Trnka Jan	Date:	25.02.2000
Classification:	For internal use only	Reference:	I-MIDI-SEC-0225/E
Keywords:	Security	Distribution:	WMO, SPR
Approval:	<Approval>	Filename:	D:\Company\Project\SWX\I-MIDI-SEC-0225E.doc

1. Introduction

The document assumes that one service provider (SWX) provides software and service to several service consumers (customers). Service consumers (customers) are using the software and thus the services provided by the service provider.

The consumer and the provider are responsible for their own security within there own locations and are free to install appropriate equipment (firewall) to protect them self's against public attack.

There is a trustee relationship between the consumer and the provider established by a signed contract containing the safety regulations and responsibilities.

Before any you start reading this paper, one fundamental statement must be made to the e-commerce situation. Almost all e-commerce solutions visible in the press media are of type business-to-consumer. E.g., a service is provided to the mass of individuals. The security problems in such environment focus on the protection of the service provider from the outside world or consumer's attacks. In opposite to this, the business-to-business type of e-commerce is much less visible in the press and deals with completely different issues. The most significant difference is equal treating of the consumer and provider regarding all security mechanisms. Not only the protection, but a co-operation becomes the focus of attention. As the SWX is dealing mostly with large organisations (banks), the e-commerce is likely to be of the type business-to-business.

2. Security requirements

This chapter provides a description of security requirements from the viewpoint of the service consumer and the service provider.

The basic building blocks of a security concept can be summarised into following points:

- Strong authentication (covered here)
- Access control & Individual accountability (partially covered here)
- Encryption (partially covered here)
- Closed private network & Demilitarised zones (partially covered here)
- Trusted computing environment (partially covered here)
- Physical protection (not covered here)
- Trusted software (not covered here)

This document deals only with issues relevant for INTERNET applications.

2.1 Authentication

Authentication means proof of identity. Both the consumer and the provider must ensure that the identity of the corresponding partner is known. This means that the consumer is sure to be connected to the right provider and the provider is sure to have connection to the known consumer. Successful authentication establishes a trusting relationship between the consumer and the provider in terms of the signed contract. Authentication failures should be written into a log in order to detect break-in trials.

Types of authentication are:

- Authentication of the session
  The authentication must be verified each time a new connection is established and optionally at periodical intervals during the connection.
- Authentication of each message
  The authentication must be verified each time a new message is transmitted.

2.2 Privacy of information

Information exchanged between the particular consumer and the provider must not be made available or disclosed to other customers, third parties or public audience. In particular following is required:

2.2.1 Encryption of data

Data transferred over media, which can be accessed by personnel not authorised by the consumer or by the provider must be encrypted. The encryption must use algorithm accepted by the IT community as a proof and certified by NCSA (National Computing Security Association). The encryption starts on the territory of the consumer at the point closest to the service consumer and ends on the territory of the provider at the point closest to the service execution.

2.2.2 Authorisation

Access to services is restricted to registered users only. The validation of the registration is done by authorisation. Access to any part of the service must be denied prior to successful authorisation. The user must have the possibility to change his registration at any time. The lifetime of the registration should not be endless, periodic change of registration (e.g. password) should be enforced. Authorisation failures should be written into a log in order to detect break-in trials.

2.2.3 User isolation

Two users are treated as independent subjects; even though both have identical authorisation e.g. the user authorises twice. Each user operates independently. There must not be any possibility to act on behalf of another user, except that this is explicitly allowed in the contract and is explicitly declared as the application functionality.

2.2.4 Publication of information

Data exchanged between the provider and the consumer must be passed on the direct way. The consumer is not allowed to forward the data from the provider to others. The provider is not allowed to forward data, which belongs to a consumer to other consumers or to others, except that this is the intended functionality and explicitly allowed in the contract and is explicitly declared as the application functionality.

2.3 Protection of the consumer

The service must be provided in a way, which does not allow the provider, other consumers, third parties or public audience to access the internal infrastructure of the consumer.

2.3.1 Intrusion protection

The consumer must protect his environment with equipment suitable for detection and prevention of intrusion. This equipment protects not only the consumer himself but indirectly also the provider as these partners have a trusting relationship. The communication concept used to exchange data between the consumer and the provider must allow passing this equipment without violation of the consumer security concept.

The consumer must ensure that third parties or public Internet users can not use the infrastructure established for the purpose of the application.

The physical protection of the consumer environment is not covered in this paper.

2.3.2 Initial Connection

The consumer always initiates the exchange of data between the provider and the consumer. The provider is not allowed to send unsolicited data of any type to the consumer, except that this is explicitly allowed by the contract and is explicitly declared as the application functionality.

2.3.3 Network configuration and control

The consumer is responsible for the configuration and controls of his own network. He is free to chose concept suitable for his own needs. The provider should not make any restrictions to the consumer network concepts. The communication concept must not allow the provider to change the consumer network configuration or to cause its performance degradation.

2.4 Protection of the provider

The service must be provided in a way, which does not allow other consumers to work on behalf of other consumers or third parties or public audience to act like regular consumers.

In addition to this, access to the internal infrastructure of the provider by third parties or public audience must be protected.

2.4.1 Intrusion protection

The provider must ensure that third parties or public Internet users can not use the infrastructure established for the purpose of the application and reserved for the regular consumers.

The provider must protect his environment with equipment suitable for detection and prevention of intrusion. This equipment protects not only the provider itself but indirectly also the consumer as these partners have a trusting relationship. The communication concept used to exchange data between the consumer and the provider must allow passing this equipment without violation of the provider security concept.

The physical protection of the provider environment is not covered in this paper.

2.4.2 Application protection

The provided application service must be protected from unacceptable usage. The definition of this term is always specific for the particular application and can not be generically defined.

The following examples show the meaning:

Protection of a Mail-System:
- limitation of the maximal message size
- limitation of the
- forwarding of messages disabled
- establish denied party list

Protection of an interactive application:
- prevent time consuming queries by restriction of selection criteria
- log out inactive users after a timeout

2.4.3 Network configuration and control

The provider is responsible for the configuration and controls of his own network. He is free to chose concept suitable for his own needs. The communication concept must not allow the consumer to change the provider network configuration or to cause its performance degradation (e.g. UDP message flooding).

3. Practical technology usage.

This chapter describes the security relevant technology and its usage.

3.1 Authentication

Several authentication methods exist and should be used according to the security requirements.

3.1.1 By knowledge

The knowledge of come code can be used to authenticate the person who knows the code. Examples are:

- PIN-Code
- Username and password

These authentication methods are suitable for anonymous services with low to moderate security requirements. The person must ensure the confidence of the knowledge and is responsible for the potential misuse. The misuse is simple and can not be detected. The loss of the code is a potentially security leakage.

3.1.2 By ownership

The personal ownership is used to identify the user. Examples are:

- Smartcard / PersCard
- SecurID
- Certificate

These authentication methods are suitable for personal services with moderate to high security requirements. The person must ensure the access to the ownership and is responsible for the potential misuse. The misuse is difficult and can be detected in certain situations. The loss of the ownership does not necessarily mean a potentially security degradation.

Due to the considerable high costs, authentication with SmartCard or SecurID is not suitable for large number of users (>1000) There are several incompatible systems available on the market today.

Certificates are usually stored in electronic format within the consumer infrastructure. It is protected by a simple passwort. This is indirectly a security leak as the certificate can be copied and re-used. Certificates also require a certificate authority (CA) accepted by both all consumers and the provider as a trusted third party.

3.1.3 By personal characteristic

The personal characteristic is used to identify the user. Examples are:

- Fingerprint
- Iris scan

These authentication methods are suitable for highly confidential services with very high security requirements. The confidence is guaranteed in almost all cases and can not be broken. The misuse is practically impossible and can be detected in most cases. The wide spreaded infrastructure (reader) is pre-requisite for this type of authentication.

3.2 Encryption

Almost all encryption methods used are sufficient (RSA, DES, RC2, RC4, RC5, IDEA). The most secure encryption methods are certified by NCSA (National Computing Security Association).

3.3 Authorisation & Access control

The mainly used authorisation method is username and password. In principle, all methods used for authentication can be used also for authorisation. Usually the Authentication is verified before the authentication is done. Consequently the user is already known in such case and the only issue to solve is the application access control.

Username and password with a limited lifetime (with or without history) is the most practicable method.

To control user activities within a particular application, each application applies a different model. Well-known models are:

- Functional model
  functions are assigned to users, who can use them.
- Role based model
  Users have roles and functions are assigned to roles.

3.4 User isolation

True user isolation can be reached only by maintaining a user context. Several methods dependent of the communication type exist:

- Stateless sessions (HTTP)
  - Cookies
    The context is stored locally on the consumer site
  - URL coding
    The HTTP response contains information used by subsequent requests identifying the session
  - HTML coding
    The HTML response contains hidden fields containing the session information. This data is sent by the subsequent request.
- Stateful sessions (TCP/IP)
  - Session process
    The context is maintained within the session server process. When the server crashes because of an error, only the particular user can not work.
  - Session thread
    The context is maintained within a thread of the session server process. When the server crashes because of an error, the entire system is down.

3.5 Publication of information

The prerequisite to proper function of information publishing is a proof IP addressing and routing concept.

The best protection against re-publishing of the information on the client side provides the restriction of the available information for the consumer. The re-use of information is a legal issue and must be covered by an appropriate contract.

3.6 Intrusion detection

Each security concept defines security levels for nodes in the company network. Access from one area to the other is controlled. Protocol must be written when user pass security level boundaries. The analysis of these protocols proves the activities of the user.

Usually the security levels are:

- Public internet (outer firewall)
- DMZ LAN (demilitarised zone)
- Internal LAN (inner firewall)
- Restricted LAN (production LAN)
- Application

This schema can be enhanced with other components required for highly secure applications.

Difficult situation arises on the front of the outer firewall, as this is the access point for anonymous users and thus the potential point of an attack. There is no common solution and each of the available products implements a different security features.

3.7 IP routing

The proper set-up of IP-Routing is the basis for the delivery of messages within the network. Please the concepts of IP routing in the

Special issues are:

- NAT (Network Address Translation)
  NAT can cause problems when the translation happens on the consumer side. It represents a potential security leak as other parties may act on behalf of other parties. You can not use CORBA or DCOM with NAT. This is because both DCOM and CORBA store raw IP addresses in the interface marshalling packets and if the client cannot connect to the address specified in the packet, it won't work. For this reasons You cannot use DCOM or CORBA through firewalls that do address translation
- IP Forwarding
  The forwarding of IP traffic may cause distribution of messages over VPNs, which do not belong to the desired network and so publishing of information to other parties.
- IP Routing for DHCP clients
  When DHCP is used statically defined routing should restrict the possibility of distribution of the messages to the wrong places.
- Dynamic versus static IP routing
  Both methods have advantages and disadvantages. The statically defined routing is generally more secure but is not suitable for large networks with thousands of individual IP addresses or when load balancing is used. When dynamic routing is used, log should be written when a new route is established.

3.8 UDP protocol

Broadcasting of messages via UDP protocol is an efficient way to distribute large number of the same information to multiple consumers at the same time. However, several troubles exist with this method:

- The UDP protocol can not pass routers and thus is not suitable for the Internet. Even an in-house LAN topology with several VPNs connected via Routers can break the UDP flow of messages.
- The UDP Protocol is generally not reliable, the sender never knows if the message has been delivered or not.
- Customers do not like UDP, because it may cause massive network performance degradation when the network is flooded by UDP messages.

3.9 IP Tunnelling

IP tunnelling is a good method to exchange sensitive data between two sites. The prerequisites are at least two firewalls through which the tunnel must pass. The trouble with tunnels is their peer-to-peer nature requiring maintenance of each connection separately. Tunnels usually do NAT on the client side and so allows the client to use IP addresses from a public range. This is important because otherwise all nodes in the network would have to agree on a single IP address schema. However in such case you can not use CORBA or DCOM. See Chapter IP routing.

3.10 Real time streaming

Real time streaming has not been investigated.

3.11 JAVA applets

The usage of Java Applets is <u>not</u> treated as secure unless the applet is certified.

3.12 Microsoft ActiveX Components

ActiveX is a technology developed by the Microsoft Corporation for distributing software over the Internet. Like Java Applets, an ActiveX "control" can be embedded in a Web page, where it typically appears as a smart interactive graphic. A number of ActiveX controls are available for the Microsoft Internet Explorer (the only browser to support them so far), including a scrolling marquee, a background sound generator, and an ActiveX control that executes Java applets. Unlike Java, which is a platform-independent programming language, ActiveX controls are distributed as executable binaries, and must be separately compiled for each target machine and operating system. The ActiveX security model is considerably different from Java applets. Java achieves security by restricting the behaviour of applets to a set of safe actions. ActiveX, on the other hand, places no restrictions on what a control can do.

The main problem with the ActiveX security model is that it is difficult to track down a control that has taken some subtle action, such as silently transmitting confidential configuration information from the user's computer to a server on the Internet, seeding the LAN with a virus, or even patching Internet Explorer so that the code authentication engine no longer functions correctly. This type of action may escape detection completely or at least for a long period of time. Even if the damage is detected immediately, Internet Explorer offers no secure audit trail that records which ActiveX controls were downloaded. This makes identifying the control responsible for damaging your system a non-trivial task.

An independent information on ActiveX can be found in: http://www.w3.org/Security/wwwsf7.html#Q65

The usage of ActiveX components is not treated as secure.

3.13 Cookies

A "cookie" is a mechanism developed by the Netscape Corporation to make up for the stateless nature of the HTTP protocol. Normally, each time a browser requests the URL of a page from a Web server the request is treated as a completely new interaction. The fact that the request may be just the most recent in a series of requests as the user browses methodically through the site is lost. Although this makes the Web more efficient, this stateless behaviour makes it difficult to create things like shopping carts that must remember the user's actions over an extended period of time.

More information on cookies can be found in http://www.w3.org/Security/wwwsf7.html#Q66

The usage of cookies is not treated as secure.

3.14 Legal and other Issues

Links:

http://www.w3.org/Security/

http://www.ebk.admin.ch/

http://www.boersenaufsicht.de/

http://www.iosco.org/

http://www.icsa.net/

4. Appendix A ERM security concept

It has been decided to use Java standalone application instead of Java Applets. This decision was driven by the resistance of several banks against applets. The usage of certified applet was dismissed due to troubles with applet signing.

The usage of cookies or Active-X components was newer considered.

4.1 Authentication

Username, password and Key-file for the AltaVista Tunnel are used for authentication. Within an open AltaVista tunnel session, keys are changed (and synchronised between the client and the server) periodically in an interval of 30 minutes.

4.2 Encryption

The AltaVista Tunnel with DES 128-bit key encryption is used.

4.3 Authorisation

The AltaVista Tunnel IP-address together with the application specific username and password are used to authorise the user.

4.4 User Isolation

Within a running USP/RPC session, all requests are passed to a single server process. This process (TXS server) holds the session context and so provides absolute user isolation.

4.5 Publication of information

Properly defined IP routing prevents network traffic to be passed between consumers.

4.6 Intrusion detection

The USP/Concentrator placed with a DMZ between two firewalls plays role of an application gateway and ensures a bullet-proof security for the consumer.

On the SWX side the AltaVista Firewall and the AltaVista Tunnel Server provide the required security.

4.7 Initial request

With the USP/RPC and USP/BCT concept, all connections are initiated by the consumer.

4.8 Network configuration and control

The USP/RPC and USP/BCT do not care about the consumer network configuration. The USP/RPC and USP/BCT are statically pre-configured IP services. There is no possibility to change remotely the participant configuration or the SWX configuration.

4.9 Internet technology specific issues.

- Java standalone application was chosen to solve difficulties with Java-Applets and their certification.
- ERM is using pure Java clients with no Active-X components.
- ERM does not use cookies. The session context is built-in functionality of USP/RPC.

5. Appendix B Consumer Security

The following chapter is a abbreviation of an Web client security concept, issues and solutions of a unnamed Bank.

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

5.2 Browser Risks

5.2.1 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).

5.2.2 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 (Switzerland, international, country-specific).

5.2.3 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). 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.
- 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 the bank internal certificate.
- Use of US-domestic browser technology.
- The usage of plugins, Java applets and Java applications is currently under definition.

5.3 Java Application Security

5.3.1 Discussion

Java is rapidly emerging as an industry standard, for thin-client architectures as well as for sever implementations. Also, in our bank 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.

5.3.2 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.

5.3.2.1 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 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.

5.3.2.2 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.)

5.3.2.3 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.)?

5.3.3 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 bank internal 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.
- The usage of Java applets and Java applications is currently under definition.

5.4 JavaScript Security

5.4.1 Discussion

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

- Client-side JavaScript
- Server-side JavaScript

5.4.1.1 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

5.4.1.2 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.

5.4.2 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

5.5 Netscape Plug-Ins

In general plug-ins inherently pose a potential risk since they can access the system without restrictions.Currently the packaged 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 required by certain applications.

5.5.1 Requirements

- 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 system test.

5.5.2 Proposed Solution

- 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.

5.6 Code Signing, Security of Downloaded Code

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

5.6.1 Requirements

With the forthcoming new 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.

5.6.2 Proposed Solution

IT Security has to define a process to sign Java applets designed for the Sun Java Plugin.

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

5.7.1 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.

5.7.2 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 world-wide 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 utilise 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 the bank. 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.

5.7.3 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 organisational 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.

5.8 Code Signing / MS Authenticode

5.8.1 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.

5.8.2 Requirements

5.8.2.1 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.

5.8.2.2 Tools

Tools to enable code signing must be provided to application developers. Because the technologies are not interoperable, a decision in favour 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.

5.8.3 Proposed Solution

5.8.3.1 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)

5.9 CGI / Perl

5.9.1 Discussion

5.9.1.1 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.

5.9.2 Requirements

Apart from the solution described in the CGI-Guidelines a process needs to be set-up where scripts are

- tested for the a number of weaknesses described in the CGI-Guidelines
- tested within the production environment (Internet DMZ, Intranet)

5.9.3 Proposed Solution

- The solution for the deployment of programs accessed via CGI on a Web Server are described in two guidelines.

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.