

Designing and Implementing Secure Web Services

by: Steve Purser, 03/02/2005

<http://www.securitydocs.com/library/3075>

Note : This paper has been reformatted and republished from our archives.

1. Introduction

In the past decade, the Internet has become almost synonymous with the world wide web (WWW) and the browser has become the de facto interface for Internet enabled applications. The key advantages of using a browser-based user interface include:

- The familiar 'look and feel' of the browser interface promotes ease of use.
- A wealth of existing standards and development practices already exists in this area.
- Browsers are capable of providing end users with an integrated view of multiple applications.
- Many of the more powerful technologies for building distributed systems can be easily integrated with a browser interface.

However, organizations looking to use standard web protocols to deploy services associated with a high-level of risk to *corporate* customers must recognize and respond to a number of special challenges if they are to meet with success.

This paper presents the key issues associated with designing and deploying highly secure applications for corporate customers over standard web protocols and proposes a number of practical solutions for dealing with these issues.

2. Challenges

Whilst web applications are very appealing from a usability perspective, deploying such systems involves rising to a number of non-trivial challenges. For the purposes of this paper, we can classify such challenges as being associated with the deployment of the client, with security on the network or with the server-side infrastructure.

Ironically, where client-side security mechanisms are concerned, many of the commonly-encountered problems are related to existing security infrastructure. Choosing appropriate mechanisms is complicated by the fact that most commercial organizations adopt a very defensive stance with respect to the Internet and such organizations will usually expect to deploy new applications without modifying their current perimeter defence mechanisms. Because local security necessarily reflects the attitude towards risk of the organization concerned, there can be no standard client-side security configuration. This is of course quite reasonable and one would expect a private bank to adopt an entirely different posture to a software development house or an educational institution. Less obvious, but equally true, is the fact that considerable differences can be found within the same sector of activity, which means that even a targeted deployment can pose problems. Examples of issues associated with the deployment of secure web clients include:

- Different browser configuration options and security settings can have a big impact on the behavior of the client software.
- Proxy servers and firewalls at client site can block standard protocols or require activating unusual protocol options.
- Client side software may not integrate well with the local workstation security configuration.

The problems of protecting data over an insecure network are well-known and standard solutions are well documented, see for example [1,2] or for protocols typically associated with web technology [3]. Nevertheless, standard mechanisms for protecting the confidentiality and/or integrity of data over network connections invariably involve cryptographic protection mechanisms and implementing such solutions can be quite difficult. More importantly, some of the problems associated with dealing with remote clients, such as understanding trust relationships and what they really mean [4] are not technical issues at all and are

rooted in the concepts that underlie the technical deployment. Examples of issues related to protecting data over insecure networks include:

- Establishing trust relationships with unknown third parties.
- Protecting the download of thin clients.
- Protecting the confidentiality and integrity of the transmitted data.
- Ensuring availability of the service.
- Achieving End-to-End (E2E) security for transactional systems.

Problems at the server side are somewhat easier to handle because such problems are in-house and under the control of the institution providing the service. Typical issues in this area are:

- Achieving an acceptable level of integration with the local security architecture.
- Interfacing specialised devices, such as Hardware Security Modules (HSM).
- Defining scalable administration procedures and workflow.

Finally, there are many issues associated with initializing and administering secure systems. These issues include identifying and implementing an appropriate administration model (should administration associated with individual end-user accounts be done centrally or at the customer site?), registering new users and initializing new accounts using out-of-band secrets.

Throughout this paper, we will use examples taken from this section to show how these issues can materialize and how they can be resolved.

3. Risk analysis and requirements

Although business involvement is important in the deployment of most new systems, this is even more true of web-based systems associated with a high level of risk. From an IT Security perspective, the first activity in any potential deployment of a new system is often the initial risk analysis. The goal of this risk analysis is to identify the key risks associated with the target system, to agree with the appropriate business line(s) the mitigating security services and to define and agree with the business the acceptable level of residual risk [5]. The residual risk is the risk, which is not mitigated, and which must therefore be accepted by the business owners. Once completed, this risk analysis is put under change control and updated throughout the lifecycle of the project. In practice, a correctly performed initial risk analysis will identify most key risks and updates will be rare (although as the project progresses, it is usually possible to gain a greater understanding of the risks initially identified, by placing these in the context of the developing system). Typically, the output of the initial risk analysis will be sufficiently accurate to provide a ball park figure for the cost of implementing the chosen security services and can be used to refine any existing cost benefit analysis.

Given that this risk analysis is usually carried out at the beginning of the project lifecycle (when there are typically a lot of unknown factors affecting both the implementation project and the resulting system), Fast Risk Analysis is often a more productive approach than an in-depth study of potential risks. Fast Risk Analysis techniques aim to identify the most important risk scenarios and to characterise each scenario using semi-quantitative measures of probability and impact (usually HIGH, MEDIUM and LOW). In performing such an analysis, it is recommended that any security mechanisms already deployed (such as Firewalls, virus control frameworks, etc.) be ignored, as any existing mechanism, which adds value, should be derived by the process. Taking the opposite approach and starting with an existing set of security services, may involve assumptions which turn out to be false at a later date.

Such an analysis is useful for identifying the key risks that the target system needs to mitigate. It is important however not to forget that certain risks are introduced as a result of implementing a new system. For instance, the risk of an attacker penetrating the network boundary may be offset by identifying Firewalls and Network Intrusion Detection Systems (NIDS) as mitigating services, but using such services might have a negative impact on performance and throughput, which is a risk in itself – particularly for systems that are associated with critical deadlines.

Where web applications are concerned, particular attention should be given to such secondary risks, as they give rise to important requirements. Hence, a decision to deploy smart cards to store private keys at the client-side is associated with a secondary risk of failing to provide adequate logistics and support. Mitigating actions might include a commercial agreement

with a third-party to support smart card readers throughout the world and/or contractual limitations on the degree of liability in the event of a problem with smart cards or readers. Even better might be the provision of a backup storage mechanism based on PKCS#12 files [6] or Personal Security Environment (PSE) standards.

During the security requirements phase, business representatives should work together with the IT Security unit and the implementation team to ensure that the initial risk analysis is correctly translated into a set of clearly defined security requirements. This may involve working with other teams, such as the Legal Department or the Compliance Officer (to check that security requirements are consistent with legal and regulatory restrictions) and customer support teams (to ensure that security procedures can be correctly integrated with customer support functions). The security requirements are extremely important as they usually drive the security acceptance criteria and hence form the basis for accepting the resulting system.

4. Design considerations

The security design is the bridge between the requirements and the detailed implementation model. As such, it should be possible to map design decisions back to the underlying requirements. Such a mapping will not necessarily be simple, as some of the design decisions will be more global in nature and will not necessarily reflect specific requirements. Nevertheless, some degree of formality is desirable and, at a minimum, it should be possible to map identified security services and procedures back to the initial risk analysis and security requirements. This also has the advantage that the resulting deliverable can later be used to support the definition of acceptance tests. It is essential that the design be signed off by all concerned parties and that any change in risk arising out of the design process be accepted by the appropriate business line.

In line with the discussion of section 2, we will look at design issues under several headings:

- Global design considerations.
- Architectural considerations.
- Design issues relating to the user interface.
- Design issues relating to the server-side components.
- Process design.

Global design considerations

Perhaps the most important global design principle is to ensure that technical design and process design are seen as two parts of the same solution and the design is not carried out asynchronously. Unfortunately, it is not uncommon to discover procedural weaknesses due to the fact that critical administration procedures are designed after agreeing on a particular technical design.

The core design should make maximum use of accepted, open standards, as interoperability is critical to success in Internet environments. Unfortunately, there are a large number of such standards and even this constraint is not strong enough (consider for instance the large number of standards in the area of cryptography and cryptographic protocols). Whereas some standards are virtually unavoidable, others can be selected based on the value they add for the problem at hand. Hence, the more important Internet Request For Comments (RFC) documents (see references [7,8] for an introduction) fall into the first category, whereas the applicability of standards such as the Public Key Cryptography Standards (PKCS) [9], particular American National Standards Institute (ANSI) standards [10] or National Institute of Science and Technologies (NIST) guidelines [11] will likely depend on the operating environment. The best approach is to select a set of consistent standards and to stick to it.

Other factors being equal, the final design should favour simplicity. In general, a more complex design results in a more complex implementation, which complicates both testing and problem solving in the live environment. Where complexity is unavoidable, particular attention needs to be paid to documentation throughout the software lifecycle. For instance, dependencies on particular browser security settings and interoperability requirements with perimeter security devices should be clearly flagged to the end user. Ideally, documentation for end users should cover all security aspects with which such users may be confronted and, for more complex implementations, it is worth considering developing a series of targeted documents rather than a single document containing everything. Such an approach helps customers distribute technical

information to the correct recipients within the enterprise.

For most web applications it is a good idea to build scalability and flexibility in from the design phase. Scalability and flexibility are measures of how easy a system can adapt to changes in the external environment (the word system here denotes both technical and procedural components). Scalability refers to the ability of a system to cope with increased volumes and flexibility refers to the ease with which functional changes can be made [12]. At the architectural level, security services should be cooperative – that is, they should be designed to reinforce each other, rather than getting in each other's way. Similarly, the E2E security model should aim to provide defence in depth by putting several obstacles in the way of an attacker.

Finally, the overall design should represent a balance between the practical advantages of certain security services and the corresponding cost of implementation. True non-repudiation services for example are implemented using complex security protocols. Part of the security lies in the cryptography and part lies in the protocol. To be reliable, these mechanisms must be supported by logs, which may need to be cryptographically protected themselves (think of deleting signed messages from logs for example). Note that even when such a system has been successfully constructed, it may have no legal value – but this is another discussion.

Architectural design considerations

When designing security mechanisms for distributed applications it is necessary to take an architectural view of the system and to resist the temptation to consider the system as a disconnected set of components. In other words, it is important to ensure that the security model is consistent when analyzed from an End-to-End (E2E) perspective. Note that even defining the end point can be tricky if the web application is to offer services to a wide range of different institutions – if the client workstation is the end point at the customer site for instance, secure protocols will need to function correctly through proxy servers, firewalls and other intermediate devices.

Failure to analyse the security of the system from an E2E perspective may result in weaknesses at the interfaces between its different components. Indeed, we most expect to find weaknesses at the interfaces between different technologies, as it is here where different security subsystems need to co-operate with each other and current levels of standardisation in this area are low. Those organizations that have designed and implemented a security architecture will be in a strong position to deliver standard security services using the established infrastructure. Such an approach helps to decrease time to market without compromising on security functionality.

In order to derive an architectural view of the system security requirements it helps to map the initial risk analysis and security requirements onto the proposed architectural solution. This will result in a set of risks and security requirements for each identified component (client workstation, network, gateway, server etc) and will identify which services need to be distributed over several components and which services are localised. Because securing high-risk web applications to run over the Internet usually involves deploying cryptographic network security mechanisms, it is useful to look at two examples in this area of how an architectural design results in a more secure system.

For example, such systems normally have demanding requirements concerning data and session integrity. For the type of system we are considering, the threat to data integrity might be HIGH on the client workstation and the Wide Area Network (WAN), MEDIUM on the organizations gateway components, and LOW on the application server and internal networks. This suggests an integrity protection mechanism such as a Message Authentication Code (MAC) or digital signature spanning all components between the client and application server. Conversely, the threat of system penetration may be assumed to be significantly higher on the client workstation than on other components due to the fact that this component of the system is out of the control of the organization - this would then be modeled as a localized risk.

As a second example, we will look at the problems associated with implementing the standard Secure Sockets Layer (SSL) protocol directly to the desktop in corporate environments. Here, it is important to understand that most large enterprises implement proxy servers for Internet access, which means that individual users will access the web application through such a server and not directly. In many cases, there will be no requirement to implement SSL to the desktop, in which case a secure SSL session can be established between the web server and the appropriate proxy server. However, where it is important to authenticate and track actions of individuals within the corporate environment, such a requirement can occur. To handle these different requirements, proxy servers can handle SSL connections in two ways [13,14]:

- They can *bridge* the SSL session on behalf of the user, which results in a secure connection between the proxy server and the web server.
- They can *tunnel* the SSL session, which results in a secure session between the client workstation and the web server.

These two modes of operation are illustrated in figure 1 and figure 2 below.

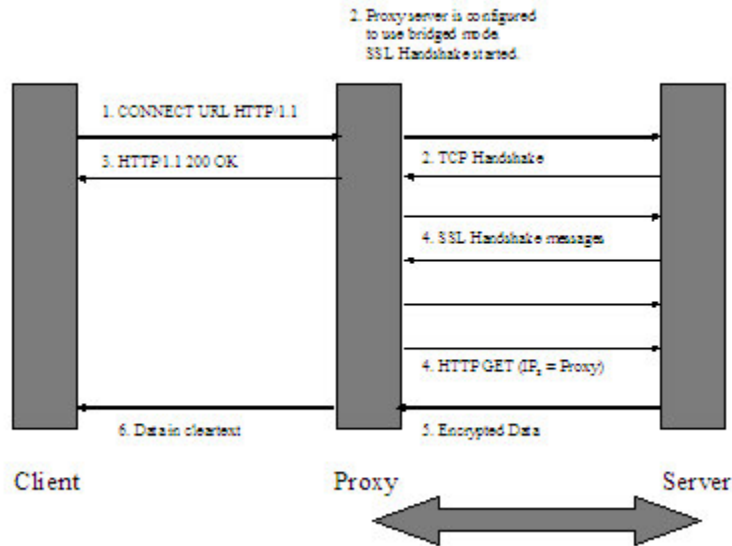


Figure 1: Bridged SSL

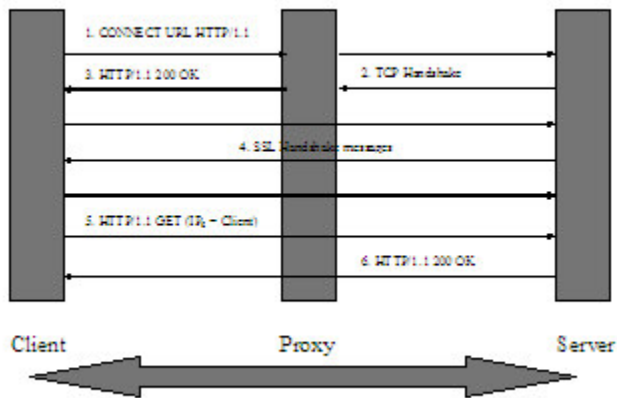


Figure 2: Tunnelled SSL

It is important to realize that the design chosen has an impact on the client-side code. If the tunnelled mode of operation is required for instance, the client will need to test for the existence of a proxy server by retrieving information stored within the browser settings and initiate the appropriate version of the protocol handshake (which involves sending a special packet, known as a 'CONNECT SSL' packet in this case) [13]. Furthermore, if the proxy server requires users to authenticate before permitting outbound connections, the client side code will have to be capable of supporting this authentication step.

These short examples should illustrate the importance of an appropriate architectural design and give a flavor of the problems that can arise when such a design is not appropriate.

User interface security design considerations

One of the most difficult aspects of developing secure customer facing systems is dealing with the variety of approaches to security at the customer site. One way of minimizing the impact here is to ensure that the security requirements and design of the target system do not make any assumptions about customer sites and therefore offer flexibility in configuring the security component. Unfortunately, this results in extra work as it becomes necessary to cater for the most demanding, whilst offering the possibility to downgrade security to suit the requirements of customers prepared to take more risk.

One of the main design decisions in this area is whether to use a thick client or a thin client. Advantages of thick clients include the fact that customers are more likely to feel comfortable installing software from a CD, that such software can generally execute unimpeded once installed and there is no need to keep the code compact (as it is not being transferred over a network connection). Disadvantages include the requirement for an efficient software distribution mechanism to support the rollout process, the fact that it is difficult to introduce changes quickly to client side code and a relatively high cost.

The opposite is true of thin clients. Customers may be wary accepting mobile code on the desktop even when it is digitally signed and, when installed, the software will be subject to restrictions imposed by the appropriate mobile code policy file. This latter restriction can become extremely important if mobile code is being used to access external devices (e.g. a java applet using a PKCS#11 interface to access a smart card – here it may be necessary to provide client-side policy statements to ensure that the applet can access the device). Mobile code must be kept reasonably compact so as to achieve reasonable download times (the fact that no assumptions are being made about the client-side environment means that we have to cater for slow connections). On the plus side however, mobile code can be quickly changed at the central site and be operational as of the next login. A big cost advantage is that it does not require any special logistics for distribution - although associated software, such as virtual machines, plug-ins or specific drivers might do.

One useful trick when designing client-side interfaces is to include verification software into the user interface. The goal of such software is to verify the configuration of the browser (and potentially the underlying operating system) and to report any problems directly to the user. Where thin clients are being used and software is downloaded from the web server, such software should be cryptographically signed so that the customer can verify the origin before allowing it to execute. Such an approach can considerably speed up deployment and reduce the pressure on customer help desks from customers that are having problems in initializing the system.

Cryptographic key storage presents particular problems at the client-side, as we can make no assumptions about client-side security. According to Kerckhoffs' principle the effective level of security offered by any cryptographic system systems is determined by the extent to which the underlying cryptographic secret ('private' or 'secret' key) is protected [15]. Client-side approaches to protecting cryptographic keys include software vaults, smart cards and dongles. There is usually a requirement for the user to introduce a secret (typically a PIN code or a password) to unlock the client-side key. External smart card readers equipped with PIN-pads provide one example of how this secret can be supplied to the device in a secure fashion (see figure 1).

EXTERNAL READER WITH PIN PAD

EXTERNAL READER WITHOUT PIN PAD

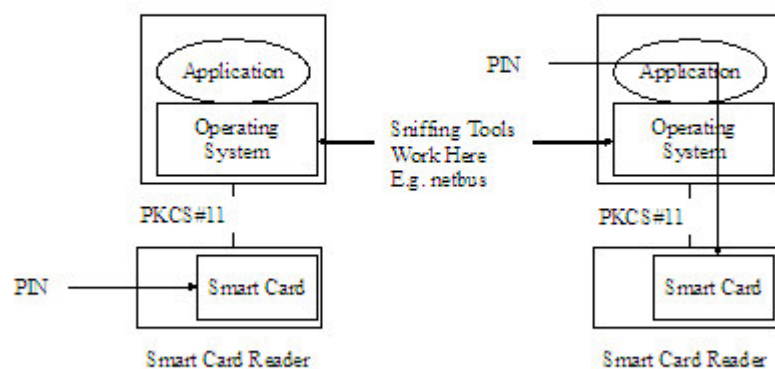


Figure 3: Smart cards for client-side key storage

Here, the essential point is that the PIN is never visible to the operating system of the client device and cannot therefore be sniffed - there is a trusted path between the PIN-pad and the smart card. However, in line with the principle that we cannot make assumptions about client-side security, it is important to allow for a reserve key storage mechanism even if most customers state a preference for a highly-secure storage device. This may be required for several reasons:

- The customer security policy may forbid externally connected devices.
- There may be compatibility issues with the hardware and software used by the end-user.
- External devices limit mobility as there is usually a requirement to install driver software on the client system.
- The cost may be prohibitive.

In these cases, use of standards-based software key storage, such as PKCS#12 files, may be an acceptable alternative to using tamper resistant equipment. [Server-side security design considerations](#)
The security design of the server-side components is typically dominated by issues related to integration and interoperability.

Where integration is concerned, server-side components should be integrated into the existing security architecture. If there is no architecture as such, these components should integrate cleanly with the security infrastructure that does exist. As an example, integrating a system developed using COM+ technology and running on windows platforms into an environment that is exclusively UNIX based is likely to present several issues. In this particular case, it may be necessary to update vulnerability scanners to cover a new operating system, new network protocols may be required (such as NBT protocols), new software might be required to implement secure protocols, such as SSH, new security baselines will have to be developed and procedures will have to be adapted. On top of all this, staff will require training to administer the new technology.

Interoperability problems occur for a variety of reasons and are particularly common when dealing with server-side cryptographic equipment. Hardware Security Modules (HSM) for instance may offer limited possibilities for integration with other software components. Web server software might support a PKCS#11 interface for certain versions of HSM software, but not for others. When support is provided, it may only be partial – continuing with the example of the PKCS#11 interface, the basic cryptographic operations may be supported, but there might be limited support for some of the ancillary functions. Similarly, there may be limitations at the network level, such as support for the SCSI interface but not for an Ethernet interface – this is an important consideration if there is a requirement for an HSM to support several systems.

If in doubt, this is one area where it is worth looking for existing implementations and trying to arrange a site visit with the aim of identifying problem areas and solutions. Site visits are useful to distinguish between what is possible in theory and what is practically achievable.

Process design considerations

Process design is important because once the system has been implemented, it needs to be administered and maintained and badly designed procedures can have a big impact on the real Return On Investment (ROI) of the initiative. Inefficient procedures result in unnecessary costs and may also result in delays when serving customers. Ineffective procedures on the other hand can result in unforeseen security exposures. It is therefore worthwhile taking the time to carefully design the process side of the service.

As a first example of where a good process design can make a big difference to the operating model, consider the issue of user administration. In traditional systems, the user account details are maintained at the server side in a central database of some kind. Where X.509 certificates are used as the basis of the authentication, there is a possibility to offload much of this administration to a security officer at the customer site. This gives more control to customers and decreases the amount of administration that needs to happen at the central site. One way of achieving this is to ask customers to delegate local security administrators and to provide them with tools for generating and managing public key pairs and associated user profiles for their organization. The site security administrators are then registered with the central system via a registration process. Once these administrators have been successfully registered however, and their certificates are in the LDAP directory of the web service provider, they can generate and assign key pairs to internal users and register the corresponding public certificates with the central site by signing a certificate request message (such as a PKCS#10 message).

The system initialization process provides a second example in this area. Here, the problem to be solved is “how do we authenticate for the first time a new user, with whom we currently have no secure established secure channel?” This inevitably involves using one or more out-of-band channels to send an initial authentication secret to the end user (note that if a distributed administration model has been selected, as in the above example, this end user will be an administrator). Good registration procedures foresee the necessary background checks and use established communications channels effectively. As registration procedures can be quite onerous, a bad design can result in considerable operational inefficiency.

Last but not least, the impact of new server components on the administration overhead should be taken into consideration as from the design phase. Components that can be smoothly integrated into an existing architecture are less likely to require big changes to existing procedures than components based on new technologies or concepts. This in turn promotes standardisation and can result in considerable economies of scale if planned that way from the start.

Quite often, the high-level design contains enough information to identify the skill sets and estimate the administration workload that will be required to support the final system. This information, which should be checked as the implementation proceeds, can then be used to prepare for the rollout process and to identify future budgetary requirements. **5.**

Implementation and testing

Typically, the IT security unit will be the major driving force behind the initial risk assessment, the requirements and the design phase. This is not to say that the security officer takes the decisions in these areas, but he/she is usually the person working behind the scenes to ensure that the approach to security is proactive and that any decisions in these areas are taken by the right people, suitably informed.

Once it comes to implementation however, control passes almost entirely to the project team and the main activity of the IT security unit is to support this team by providing expertise and guidance as required. The degree to which the security design is successfully implemented will depend largely upon how well IT security is integrated into the development or acquisition process and it is extremely important here to offer strong support for the methodology adopted by the project team. Imposing methods that are not compatible with the usual methodology for acquiring or developing new products often leads to confrontation and uncomfortable compromise.

Where packages are being acquired, the IT security unit should assist the project team in refining high-level requirements and design elements to the level of detail required by the Request For Proposal (RFP) or comparative study. Where development is to be carried out in-house, the unit should be able to help the project team identify any security trade-offs required as a result of development issues. In both cases, implementation teams will almost certainly require proactive support wherever specialized material, such as smart cards or HSMs is being used. For those organizations that choose to build the security functionality into the application itself, rather than simply have the application use a secure connectivity solution, the IT security unit may have to support the use of cryptographic toolkits. Although such an approach is capable of achieving a higher level of security, this is likely to be quite time consuming. Indeed, it is to be expected that the project team will need extensive guidance, not only on the underlying principles, but also on how to handle particular implementation issues.

The translation of security requirements into acceptance criteria and acceptance test scenarios should be business driven, but requires close collaboration with the IT Security unit and the implementation team. It is a good idea to develop a test strategy before starting to define individual test scenarios and this strategy should reflect the issues outlined in previous sections. Where web applications are concerned, penetration testing and tests involving a representative set of customers are worth a special mention. Penetration tests essentially seek to ensure that the central site cannot be compromised as a result of a vulnerability in the system or in the way that the system integrates into the IT infrastructure of the provider. A documented, successful penetration test can be useful in convincing potential customers that the system has been correctly secured. Tests involving representative customers on the other hand, aim to identify potential issues at the customer site and to ensure that these issues are resolved. This is where unwarranted assumptions about the customer site are most likely to come to light.

Throughout the development and acquisition process, the business should be made aware of issues having a functional impact or requiring a modification of the risk analysis and the way of dealing with these issues should be mutually agreed. Obviously, this dialogue needs to avoid technical complexity and concentrate on functional issues.

6. Deployment

As with any new application, it is critical to manage the deployment phase correctly, as this is the first point of contact for the majority of customers. A deployment that does not proceed smoothly may negatively influence the opinion of the customer for some time. Key activities during the rollout process include:

- Defining the deployment strategy.
- Defining how software (and potentially hardware) will be distributed.
- Preparing customer organizations to deal with configuration issues.
- Training end-users to use the security functionality.
- Training in-house staff to administer the system.
- Verifying and distributing documentation.
- Providing the necessary support functions.
- Planning and execution.

In order to be correctly prepared, work on the deployment strategy should begin as soon as possible and the overall approach should be modified as the implementation proceeds. This strategy should take into account all the different aspects of the rollout process including the logistics of distributing software and hardware, possible impact on the technical infrastructure and procedures of the end user, training requirements and how the deployment will be supported.

When dealing with commercial customers, it is almost inevitable that there will be a need for software or hardware distribution of some kind. Using a thin client, where the software is downloaded at connection set-up can minimize the effort involved, but even thin clients require supporting software. Whenever additional software such as plug-ins and drivers are required, the customer should have an easy method of obtaining and installing the correct version. In many cases, it will be possible to direct customers towards a web site where the software can be downloaded (but note that the download mechanism may not be secure in this case – a fact that could compromise an otherwise highly-secure system). In others, service providers may opt to reach an agreement with the companies providing the support software and bundle this into the standard distribution. Unfortunately, the distribution logistics become much more complicated if secure storage devices are being used at the client side. This too has an impact at the customer side, as many departments may be involved in this exercise. For instance, acquiring devices may involve purchasing and logistics units, installation may involve the IT department and end-users of the system need to be trained to use them correctly.

The latter point is important because modern security mechanisms do not necessarily function in the same manner as their classical counterparts. An example is provided by a system that uses an authentication mechanism based on cryptographic challenge-response. Typically, the end user will be issued with a password or Personal Identification Number (PIN) to unlock the private key, which is used in the authentication process. If a distributed administration model is used, where organizations are responsible for administering their own users, passwords will be issued at the customer site and not centrally. Users should therefore be aware of the fact that this particular password is unlocking a local file (or providing access to a local device) and is not the basis of the authentication with the target system. More generally, users will be required to understand the basics of how the security system works in order to avoid inappropriate behavior. For example, an end-user who generates and uses an electronic signature outside the scope of the application for which the corresponding certificate was issued (if this is possible), may well render the organization liable for any consequences.

To respond to these and similar issues it is useful to establish contact with someone within the customer organization (typically the security officer) who understands the risks associated with conveying highly sensitive information over potentially hostile networks and how these risks are mitigated by the application. This in turn allows the organization to define how it will configure the security subsystem to reflect its own requirements. By ensuring that the design of the application does not make assumptions about the approach to security at the local site, it should be possible to configure the resulting system so as to be compatible with the requirements of the latter.

This whole process is greatly facilitated by good, targeted documentation. If the documentation is well-presented, not too

lengthy and informative, there is a good chance that it will be read, which should decrease the amount of direct support that needs to be given in order to get customers operational. Where certificates are being used, it is a good idea to check that the technical documentation is consistent with the Certificate Practice Statement (CPS) and Certificate Policy (CP) and that all these documents are in line with the contractual documentation.

As for traditional applications, customer support requirements will depend on a variety of factors including the type of customer, typical working hours, working language and geographical distribution. It is common to distinguish between first-line support staff, who are used to dealing with customers and have the necessary appreciation of the business and language skills, and second-line support staff, who provide in-depth technical support.

Finally, once a position has been adopted with respect to each of the above points, the rollout process has to be planned. Perhaps the best approach here is to be conservative – a customer will usually welcome an early deployment, whereas few customers will appreciate a late one. **7. Administration and monitoring**

Although administering and monitoring are activities that occur at the end of the system lifecycle, they need to be planned for at an early stage. Indeed, as such activities are ongoing, requirements for extra resources are likely to have a big impact on the business case. Worse still, if requirements are not flagged up front, it may not be possible to secure these resources later on, which could compromise the whole setup.

Ideally, administration procedures will be closely aligned with those in existence for other applications. Where the security model is based on the use of certificates for example, it should not be difficult to re-use existing procedures for certificate management (although some modifications may be necessary to reflect differences in risk when compared with other applications). Using common procedures minimises the need for special training and permits economies of scale.

Obviously, administration procedures should be coherent when viewed as a whole. In particular, the goals of the procedures associated with the web service in question should be compatible with those used to administer the underlying architecture. Hence, it may not make a lot of sense to impose severe technical constraints and procedures on the application if the administration of the underlying operating system is lax.

Last but not least, administration procedures should be operationally realistic. It is good to have procedures for regularly analyzing the content of security logs, but this is only useful if the person analysing the logs has the ability to recognize the signs of a problem. Unfortunately, it can be quite challenging to understand certain technical logs, which implies that a skilled resource will be carrying out the analysis. Skilled resources however are not usually over keen to perform this kind of work and may easily become bored. This in turn can lead to oversights and mistakes and renders the whole process inefficient. A creative approach can go some way to resolving these issues. This might involve technical measures such as using intelligent filtering rules and automatic comparison with previous logs or procedural changes, such as rotating the log analysis task within the team to avoid boredom [12].

8. Conclusions

There are many advantages in using web protocols and a browser-based interface to implement highly secure services over the Internet. However, offering such services to corporate customers presents a series of challenges to the service provider. At the customer site, secure web services must be capable of operating smoothly in the presence of established security infrastructure, such as proxy servers and firewalls. Additionally, the installation and day-to-day use of specialised equipment, such as smart card readers, should not place unreasonable demands on the end user. At the provider site, it may not be trivial to smoothly integrate security software and hardware into the existing security infrastructure and the processes required to support the new service can easily become costly and arduous.

The initial risk analysis exercise should clearly identify the key risks to be mitigated, together with the residual risk that remains once all technical counter-measures and procedures have been implemented. This risk analysis should also take account of secondary risks – those risks that arise as a result of the risk mitigation actions. The residual risk should be signed-off by the appropriate business line, who will then work with the IT security unit to transform this analysis into a set of security

requirements.

It is useful to think of design criteria in several areas. Global design criteria relate to the system as a whole and relate to fundamental issues, such as design complexity and the degree to which the final system needs to be flexible and scalable. Architectural design criteria aim to ensure that security services are deployed in an effective and efficient way when viewed from an E2E perspective. Examples were discussed that illustrated why it is important to have a sound architectural view of the proposed security solution and where problems can occur if this is not the case. User interface security design considerations include whether to use a thin-client or a fat-client, the use of verification software and how cryptographic key storage will be achieved. Server side considerations on the other hand are largely concerned with integration and interoperability. Finally, it is critically important to consider process design and technical design as two complementary halves of the same solution.

During the implementation phase, the IT security unit assists the implementation team by providing expertise and guidance. Where packages are being acquired, this might involve completing the security section of the RFP. Alternatively, for a development initiative it might involve helping developers come up to speed with security toolkits and helping the team quickly resolve any problems. The IT security unit will also play a big role in designing the testing strategy for security-related functionality and may also be involved in test design.

The extent to which the deployment exercise is successful will depend on a number of factors, including the degree to which the strategy responds to customer demand, whether or not distribution logistics are under control, how training will be handled, the underlying customer support process and the amount of planning that has preceded the exercise.

About the Author

Steve Purser is Director ICSD Cross-Border Security Design and Administration at Clearstream Services, Luxembourg. Steve is also a founder Member of the “Club de Sécurité des Systèmes Informatiques au Luxembourg (CLUSSIL)” and author of “A Practical Guide to Managing Information Security” (Artech House, 2004).

References

- [1] William Stallings, “Network Security Essentials (International Edition)”, Pearson US Imports & PHIPs (2002).
- [2] Charlie Kaufman, Radia Perlman, Mike Speciner, “Network Security: Private Communication in a Public World”, Prentice Hall PTR (2002).
- [3] Stephen Thomas, “SSL and TLS Essentials: Securing The Web”, John Wiley & sons Inc (2000).
- [4] Steve Purser, “A simple Graphical Tool for Modelling Trust”, Computers & Security, Vol. 20., No. 6 (2001).
- [5] Steve Purser, “A Pragmatic Approach to Managing Information Security”, Artech House (2004), pp. 27-30, 241-248.
- [6] RSA Laboratories, “PKCS#12: Personal Information Exchange Syntax Standard”, <http://www.rsasecurity.com/rsalabs/node.asp?id=2138>
- [7] J. Postel, “RFC 1539 – The Tao of IETF – A Guide For New Attendees of the Internet Engineering Task Force” (1989)
- [8] G. Malkin, “RFC 1111 – Request For Comments on Request For Comments: Instructions to RFC Authors” (1993).
- [9] RSA Laboratories, “Public Key Cryptography Standards (PKCS)”, <http://www.rsasecurity.com/rsalabs/node.asp?id=2124>
- [10] The American National Standards Institute (ANSI), <http://www.ansi.org/>

[11] The National Institute of Science and Technology (NIST), Computer Security Resource Center (CSRC), <http://csrc.nist.gov/>

[12] Steve Purser, "A Pragmatic Approach to Managing Information Security", Artech House (2004), pp. 13-14, 155-179.

[13] A. Luotonen, "Tunnelling TCP Based Protocols Through Web Proxy Servers", draft-luotonen-web-proxy-tunnelling-01, <http://www21.ocn.ne.jp/~k-west/SSLandTLS/draft-luotonen-web-proxy-tunneling-01.txt>

[14] R. Khare, S. Lawrence, "RFC 2817 – Upgrading To TLS Within HTTP/1.1"

[15] B. Schneier, Crypto-Gram Newsletter, May 15 2002.