Global Co-Operation in the New Millennium
The 9th European Conference on Information Systems
Bled, Slovenia, June 27-29, 2001

WEB SECURITY: THE EMPEROR'S NEW ARMOUR

Audun Jøsang*

Distributed Systems Technology Centre*, Queensland University of Technology, GPO Box 2434, Brisbane Qld 4001, Australia>
Tel.: +61 7 3864 1052, Fax:+61 7 3864 1282
ajosang@dstc.edu.au

Peter M. Møllerud

Telenor, Postboks 6701 St. Olavs Plass, 0130 Oslo, Norway
Tel.:+47 23250500, Fax:+47 23250505
peter.mollerud@telenor.com

Eddy Cheung*

Distributed Systems Technology Centre, Level 7, GP South, University of Queensland, Qld 4072, Australia
Tel: +61 7 3365 4310, Fax:+61 7 3365 4311
cheung@dstc.edu.au

ABSTRACT

The World Wide Web originally provided no security services because it was not designed to support sensitive applications. As the Web evolved to become a platform for all types of Internet applications security mechanisms were added. Many Internet players, especially in the e-commerce sector, claim that the Web now can provide adequate security protection. In this paper we analyses some aspects of Web security, and our conclusion is that despite strong cryptographic mechanisms standard Web security solutions can only provide casual protection. We also conclude that major design changes needs to be introduced in order to strengthen Web security.

1. INTRODUCTION

Web security is a complex topic including computer system security, network security, firewall configuration, authentication services, privacy, cryptography, public-key infrastructure (PKI) and trust management. Web communication security is based on the SSL [9] security protocol combined with a

* The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Australian Federal Government's CRC Programme (Department of Industry, Science & Resources)

public-key infrastructure (PKI) for the generation and distribution of private/public key pairs and public-key certificates. The Web architecture follows the traditional Internet paradigm where a client machine accesses a server machine usually protected by a firewall as illustrated in Fig.1.

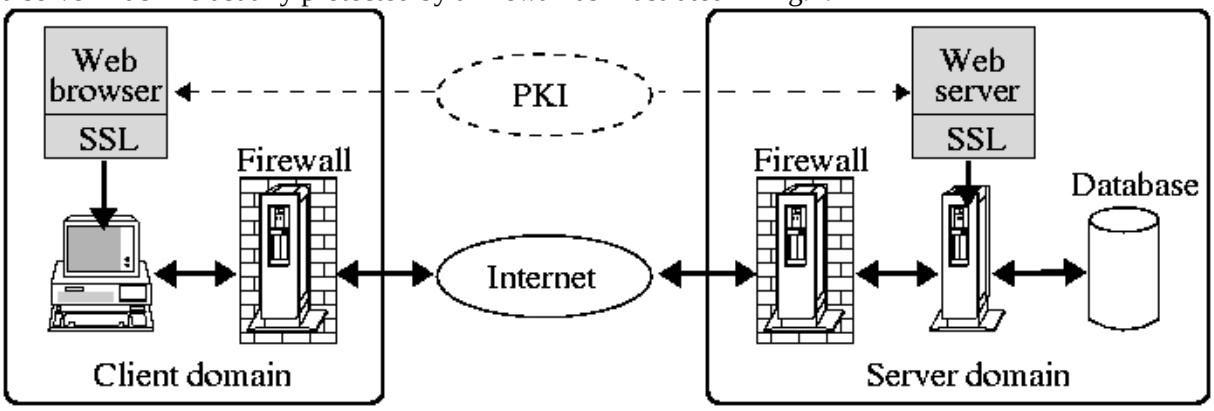

***Figure 1**: Web security Architecture*

The *Web browser* is the software that runs on the client machine and the *Web server* is the software that runs on the server machine. The theoretical foundation for the cryptographic mechanisms that form the basis of Web communication security has been developed by researchers over the past 25 years and it is only in the past 5 years that this has been implemented into commercial products. Strong cryptographic mechanism is only one of the necessary factors for achieving overall security, other factors are for example system security, key management, practice and procedures, user awareness and a good user interface. Unfortunately there seems seems to have been too much focus on cryptography and too little on the other factors. Security is only as strong as the weakest link in the chain, and one would get a false impression of security by only looking at cryptographic strength.

The title of the paper makes allusion to H.C.Andersen's fairy tale "The Emperor's New Clothes" [1] in which two swindlers tell the emperor that they can make a dress that is so beautiful that it will be invisible to those who are either foolish or incompetent in their job. In his vanity the emperor provides the fraudulent dress makers with all the finest silk, gold threads and money they ask for. The swindlers hide the treasures in a bag and start making the fabric with empty weave shuttles. When the prime minister enters to inspect their progress he discovers to his terror that the fabric is invisible. In order to hide his incompetence he praises the fabric to the skies. The same happens to everyone else including the emperor. During the procession where the emperor shows his new clothes everyone on the street loudly praises the beautiful dress until the truth finally is revealed by an innocent child who cries out that the emperor has nothing on.

2. DEADLY CONTENTS

The two main Web browser manufacturers are Microsoft and Netscape with their respective *Internet Explorer* and *Netscape Navigator*, and the competition between these two companies constitutes one of the driving forces behind Web technology innovation.

Before active content was available Web pages were mainly static displays of information coded in the Hyper Text Markup Language (HTML). Active content allows sound and image animation and provides the user with the ability to interact with the server side during a Web session. Active content exists in many forms. Java applets and ActiveX controls are some of the best known but there are also JavaScripts, VBScripts, MSWord Macros and even images. All these basically consist of mobile code that is sent from the Web server and loaded into the client machine for execution there.

All this is very appealing from a functionality and flexibility point of view but it poses a formidable threat to the integrity of the client machine. Firewalls offer little protection because they are usually configured to let http traffic and active content through. Unless the active content is constrained in what it can do all files and network connections can be accessed and (mis)used, making it impossible to operate any secure applications

on the client machine. The browser vendors have used two different strategies to protect the client machine from malicious active content.

Java applets are executed by the Java Virtual Machine (JVM) developed by Sun Microsystems and the initial strategy used to constrain Java applets was based on the JVM *Java sandbox*. It basically consists of the *Bytecode Verifier* which statically verifies that the applet only contains safe instruction (e.g. no data type violations), the *Applet Class Loader* which at load time verifies that the Java applet does not contain any illegal Java class names (e.g. equal to existing Java classes in the JVM) and finally the security manager which dynamically verifies that the applet does not access any forbidden resources such as files and network connections. The philosophy introduced by Netscape and Sun Microsystems was to protect the client machine and ultimately the user by mechanisms. The advantage of this approach is that the user does not have to worry about malicious applets as this is taken care of by the system. The disadvantage is that the applets become quite restricted in what they can do, thereby limiting the range of applications for which applets can be used. After all, allowing applets to access system files and network connections provides great potential for advanced services.

Microsoft introduced a totally different philosophy. Their initial approach was to give ActiveX controls full access to all system resources and instead let the user decide whether a particular ActiveX control should be allowed to execute or not. This is done by digitally signing ActiveX controls and thereby allowing the identification and authentication of the companies that have produced them. In addition the user can "tune" the browser to different security policies. For example the browser can be tuned so that all ActiveX controls are accepted by default or alternatively so that only controls coming from certified companies are accepted by default and all the rest trigger a dialog box, or for paranoid users so that all controls trigger a dialog box. The dialog box basically asks the user whether he or she wants the control to be executed and whether controls coming from the same company shall be executed in the future. The advantage of this philosophy is that it allows great flexibility in that controls can use all system resources. The disadvantage is that the users must make security decisions on the fly while using Web services. The problem is that the users must base their decision solely on the software manufacturer's identity of which they in most cases never have heard of. The result is that the users are left in the dark when deciding whether or not to accept a piece of active content. Experience shows that users almost always accept active content when asked by a dialog box simply because they want the functionality and because most active content is benign anyway. This means that when the user is presented with a piece of malicious active content he or she will almost certainly make the wrong decision and accept it. The user simply does not receive sufficient evidence to make an informed decision.

As would be expected Netscape has tried to come up with similar solutions to Microsoft's by digitally signing applets and granting them access to more system resources. In a similar way, Microsoft is introducing constraining of their proprietary active content. More details about the security tradeoffs of these two philosophies can be found in Princeton (1997) [11]. A discussion of threats and risks posed by active contents can be found in Jansen (2001) [5].

3. THE WEB PKI

Public-key cryptography is the basis of several important security services such as non-repudiation and authentication and is an essential building block in SSL (Secure Sockets Layer) [9] that is used for securing Web communication. A public/private key pair is used for encryption and digital signature and it is predicted that every player on the Internet will have its own public/private key pair which will form the basis for the user's or organisation's digital identity in the electronic environment. This requires the secure generation and distribution of potentially hundreds of millions of public/private key pairs, which poses a formidable key management challenge. A PKI refers to an infrastructure for distributing public keys where the authenticity of public keys is certified by Certification Authorities (CA). A certificate basically consists of the CA's digital signature on the public key together with the owner identity, thereby linking the two together in an unambiguous way. The structure of digital certificates is standardised by the ITU X.509 standard [3]. In order to verify a certificate the CA's public key is needed, thereby creating an identical authentication problem. The CA's public key can be certified by another CA etc., but in the end you need to receive the

public key of some CA, usually called the root CA, out-of-band in a secure way, and various solutions can be imagined for that purpose. In order for two users to verify the authenticity of each others public keys it is sufficient that there exists a certification path between them. We will use an example from the X.509 standard to illustrate this. Fig.2 ([3] Figure 4) illustrates a fragment of a PKI where the CAs form a hierarchy. In the X.509 notation the public key certificate of a user with name ***Z*** produced by the certification authority with name ***Y*** is written as ***Y«Z»*** The signature in the certificate can be checked for validity by any user with knowledge of ***Y***'s public key Y_p, and thereby obtain an authenticated copy of ***Z***'s public key Z_p. This process is denoted by:

$$Z_p = Y_p \cdot Y\langle\langle Z\rangle\rangle$$

Besides the information shown in the boxes we assume that each user knows the public key of its nearest certification authority, as well as its own public and private keys. In particular it can be seen that each CA has verified and certified the node above and below.

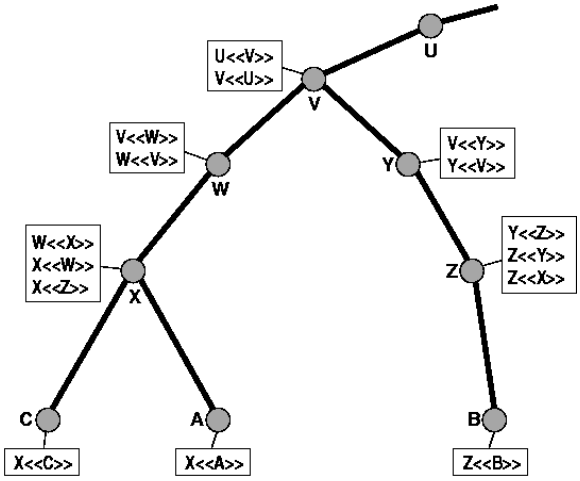

Figure 2: *X.509 CA hierarchy [3]*

If for example ***A*** wants to verify the authenticity of ***B***'s public key he must first acquire the certificates from the CAs in the hierarchy along the path between them. The following certificates must be acquired: ***X«W»***, ***W«V»***, ***V«Y»***, ***Y«Z»***, and ***Z«B»***. The chain of certificates can be resolved by resolving each certificate successively to obtain an authenticated copy of ***B***'s public key:

$$B_p = X_p \cdot X\langle\langle W\rangle\rangle\ W\langle\langle V\rangle\rangle\ V\langle\langle Y\rangle\rangle\ Y\langle\langle Z\rangle\rangle\ Z\langle\langle B\rangle\rangle$$

Certification between nodes is directed from the certifier to the owner of the certified key. Certification is unidirectional when an agent ***X*** certifies the public key of another agent ***Y***, and bidirectional when the agents ***X*** and ***Y*** certify each others public keys. In the PKI jargon *"certifying a user"* means *"certifying the user's public key"*. Chained certification can form different topologies.

Most commercial PKIs are strict hierarchies, as illustrated in Fig.3, and most only consist of one or two levels. The certification paths go strictly from the top root CA, eventually via intermediate CAs, and down to users, where the users are assumed to be certified by the leaf nodes.

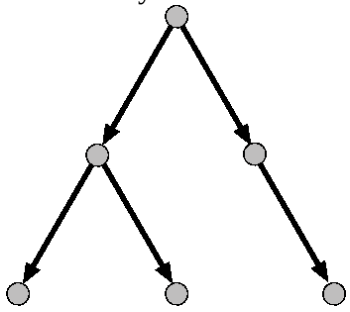

Figure 3: *A strict certification hierarchy*

In a strict hierarchy all users can be easily identified and found because of the hierarchic structure. A user must know the public key of the top root in order to resolve certificate chains and establish a certification chain to any other user in the hierarchy.

In contrast to a strict hierarchy a general hierarchy includes two-way certification between CAs, as illustrated in Fig.4.

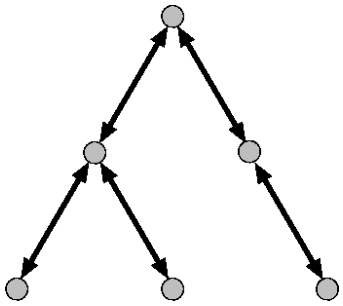

Figure 4: *A general certification hierarchy*

When certification takes place in both upwards and downwards direction, each user will only need to obtain an authentic copy of the nearest CA's public key, while still being able to establish a certification path to every other user in the network. The X.509 standard [3] suggests a general hierarchy of this type, but the Web PKI consists of a handful of isolated strict hierarchies as illustrated in Fig.5 below.

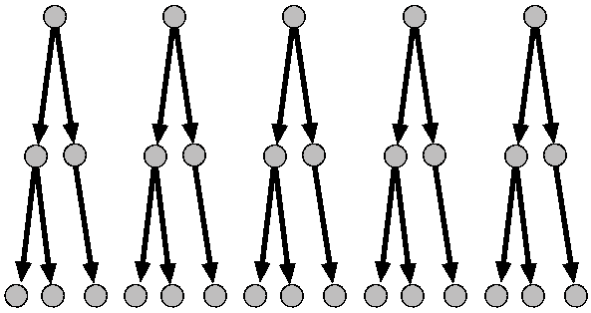

Figure 5: *Isolated strict certification hierarchies of the Web*

The root of each isolated hierarchy is a so-called self signed CA certificate. At present the major Web browsers come with about 50 pre-stored root and intermediate CA certificates which are loaded onto the user's computer when the browser software is installed. This actually represents an out-of-band channel for distributing root CA public keys to users.

It seems that the choice of using only strict hierarchies in the Web PKI has been dictated by the need for a business model and not by the requirements of trust and security. The strict hierarchies are based on distributing the same root certificates to every client machine, and the idea is that root CAs must pay the browser manufacturers (Microsoft and Netscape) for hard coding the root certificates into their browser software. Having paid the price for an entry in the list of root certificates the root CAs now have full control of the server certificate market because every certification path has to start from one of the root CAs. If a general hierarchy was used then subordinate CAs not figuring on the list of CAs with prestored certificates would be able to issue server certificates and bypass the root CAs' control. These certificates would still be verifiable by any browser by simply issuing a certificate directed from the subordinate CA to one of the root CAs.

One of the reasons why the Web PKI consists of isolated hierarchies is that cross certification would allow users and Web sites to buy certificates from cross-certified CAs and thereby dilute the root CAs' control in a similar way. Cross certification would however be very convenient because it would be possible to buy certificates of CAs other than those that come hard coded in the browser.

A PKI also requires that the relying party is able to obtain an authentic copy of a CA's public key out-of-band. We claim that this is difficult to achieve with a handful of global CAs serving the whole Internet community as the case is for the Web PKI. If CAs are either local and/or serve a limited number of relying parties trust relationships can be much stronger, and out-of-band distribution of CA public keys and user private keys can be much more secure. Other problems related to PKI are described in Ellison and Schneier (2000) [2].

4. PKI MANAGEMENT

Public-key infrastructures (PKI) simplify key management and distribution but create trust management problems. A PKI is a complex infrastructure consisting of a range of interdependent procedures and elements which need to be properly managed. The security breach of a single element can cause major damage and make large parts of networks insecure, so all these procedures and elements must be trusted.

For example, how do you know that CAs correctly identify and authenticate certificate owners before issuing public-key certificates? How do you know that certificate owners protect their private keys? How do you know that a certificate has not been revoked? How do you revoke a certificate if your system is attacked and put out of function? What is the owner liability for damage caused by misuse of private keys? What is the CA liability for damage caused by misuse of false certificates? At present these questions can not be answered in a satisfactory way. Users have no way of knowing whether certificates are revoked or not because Web browsers are not enabled to access certificate revocation lists (CRLs). Private keys are often stored on Web servers, and because Web servers are often successfully attacked these keys are vulnerable to theft and misuse. VeriSign which is the largest Web CA has admitted that it has issued false certificates because it failed to correctly identify certificate owners, as reported e.g. by the Microsoft Security Bulletin MS01-017 [8]. These false certificates can be misused and although theoretically revoked they are still valid for all practical purposes because Web browsers are not CRL enabled. Such false certificates will be accepted as genuine by most Web browsers. The only way to recover from the situation is to have CRL enabled browsers.

The purpose of using a PKI is that key distribution can be done online. For example, the distribution of user and server certificates takes place online and usually in an automatic fashion. It must however not be forgotten that PKIs also require various types of communication to take place out-of-band in order to create an uninterrupted chain of trust. The out of band channels can for example be physical delivery or encounter, physical mail, telephone or email. A common characteristic of out-of-band channels is that they are usually expensive to operate in comparison with the normal online channel. Out of band channels are often used for bootstrapping security systems, for providing additional evidence, for error recovery and for emergency situations. In a PKI setting out-of-band channels must be used for distribution of root CA public keys, distribution of private keys and certificate revocation, and and it could be used for software distribution and transaction confirmation for increased assurance.

Client and server certificate distribution which takes place online represents high volume traffic whereas distribution of private keys and root CA public keys as well as certificate revocation hope-fully will represent low volume traffic. E-commerce solutions should be both efficient and secure, and the need to use out-of-band channels could be seen as a bottleneck on efficiency. Architecture and application designers should not be tempted to trade off security to increase efficiency by trying to take out-of-band communication lightly.

5. BROWSER (UN)INTEGRITY

Root certificate integrity is a crucial factor for Web security, because if they can not be trusted then none of the security services based on digital certificates can be trusted. Certificate integrity necessarily depends on the security of the system where the certificates are stored but we will not discuss this here. See e.g. Loscocco *et al.* (1998) [7] for a discussion of this topic.

All the standard root certificates like those of VeriSign and Thawte are hard coded in the Netscape and Microsoft browsers. The Netscape browser copies the root certificates to a special file called cert7.db. Whenever a user accepts a non-standard CA certificate, this is also stored in the cert7.db file. If for one reason or another this file is removed, it will be automatically regenerated by Netscape, thus loosing all the user accepted certificates.

This certificate database is stored in the user specific directory, so each user can maintain his or her own database. This goes for the Windows versions, as well as for the Unix versions of Netscape. The files are interchangeable between Unix and Windows.

The Netscape browser does not check the integrity of an existing cert7.db file. Although the file has a special format we were able to use a normal text editor to change all occurrences of VeriSign's certificates (this is just an example, the problem itself has nothing to do with VeriSign certificates!) into XeriSign. After editing the file, we restarted Netscape and it reported no errors on tampering with the cert7.db file.

However, visiting a site which has been signed by a VeriSign root CA the browser reported this root CA as unknown, and the only way to establish a SSL connection was to manually accept the server certificate. Although this clearly represented a breach of integrity we did not get any hint about errors by opening the Security Menu option for manual inspection of the modified certificates, and the user would need to discover the integrity breach by other means. We were able to restore the integrity by simply deleting the current cert7.db file (thereby loosing the user accepted certificate as well), and have the Netscape browser generate a new one the next time it was started.

By using publicly available SSL software it is possible to create root CA certificates using any CA name. It is therefore possible to use the same issuer information as that of one of the standard root CAs, and a user will not see any difference between a valid CA certificate and the false CA certificate. Only by comparing the signature of the false certificate with the genuine bit by bit would it be possible to detect the difference. In order to tell which is which it would require the exactly knowledge of how each certificate was obtained, because nothing in the certificates themselves would be of any help. The false certificate can be used to sign false server certificates and for example masquerade as genuine Web sites on the Internet. In effect it is possible to create signed server certificates that appear to be a valid and trustworthy, and that only differ from authentic certificates by their cryptographic signature.

When visiting a Web server with a false server certificate, the Netscape browser would still present the user with the dialog to accept this certificate. So in this case the user is still able to detect the fact that the certificate is false. However, it is possible to use the Netscape browser to visit a false server set up for this purpose and exercise the dialog to permanently accept the false server certificate. This causes a false server certificate to be stored in the cert7.db file. Alternatively by studying the structure of the cert7.db file it is possible to insert the false CA certificate as a standard root certificate into the file.

Once this is completed the cert7.db file can be copied on the (end user) system to be attacked. Whenever that user on that system visits your false Web site, its certificate is tacitly accepted, and when the user wants to check out the validity of the certificate, both the issuer and subject appear to be as expected. By combining the overwriting of the cert7.db file with adding an entry for false Web site in the host's bookmark file on that very same machine, it is possible to effectively redirect all traffic intended for a particular Web site to the false server, and decode all http traffic.

Overwriting these files on other users' machines can be done with high probability of success by different types of malware such as for example by sending a virus by email or an active component from a Web server. It might not be possible to attack a specific user, but it would be easy to successfully attack a considerable proportion of a group of users. If this attack is target to masquerade a well known e-commerce server on the Internet, it would be possible to change orders, reroute them to another delivery address, and so forth. This is explained in more detail in the next section.

Microsoft's browser Internet Explorer utilises the operating system registry to store digital certificates under the registry keys: *HKEY_LOCAL_MACHINE\Software\Microsoft\SystemCertificates* and

\HKEY_CURRENT_USER\Software\Microsoft\SystemCertificates. Each of these has a *ROOT* store containing certificates of the root CAs, and a *CA* store containing certificates of Intermediate CAs.

Certificates can be viewed and displayed using the Microsoft Certificate Manager. When a certificate has been corrupted the Certificate Manager simply ignores it without any warning or alert. Similarly to Netscape Navigator it is possible to let Microsoft Internet Explorer accept a false certificate, and in that case the Certificate Manager displays it in the normal way. A registry store with false certificates can then be used to overwrite registry stores of systems to be attacked.

6. MAN-IN-THE-MIDDLE ATTACK WITH FALSE CERTIFICATES

Assume a user ***A*** who wants to access Web services from secure server ***B*** which for example can be a bank providing financial transaction services for its clients. In the normal scenario, user ***A*** points his Web browser to bank ***B***'s Web site. The Web server returns ***B***'s server certificate $\mathbf{Cert}_{\boldsymbol{B}}$ to ***A***'s browser which verifies the certificate using the pre-stored public key of the root CA that generated $\mathbf{Cert}_{\boldsymbol{B}}$. After successful certificate validation ***A***'s browser continues the communication with server ***B*** in secure SSL mode.

In the following we will assume that an attacker who wants to masquerade as ***B*** is using a false certificate in the name of ***B***. This could for example happen:

- by obtaining a false server certificate in the name of ***B*** by pretending to represent ***B*** when buying the certificate, as in the case of [8]; the false server certificate will be accepted by the genuine CA root,
- by installing a false root certificate in ***A***'s browser by using malicious active contents; this false root CA will accept any false server certificate produced by that false root CA.

Fig.6 below shows user ***A***'s client machine on the left and bank ***B***'s server on the right side. We will show that the intruder ***B′*** in the middle is able to make both ***A*** and ***B*** think they are communicating with each other although they in fact communicate with ***B′***.

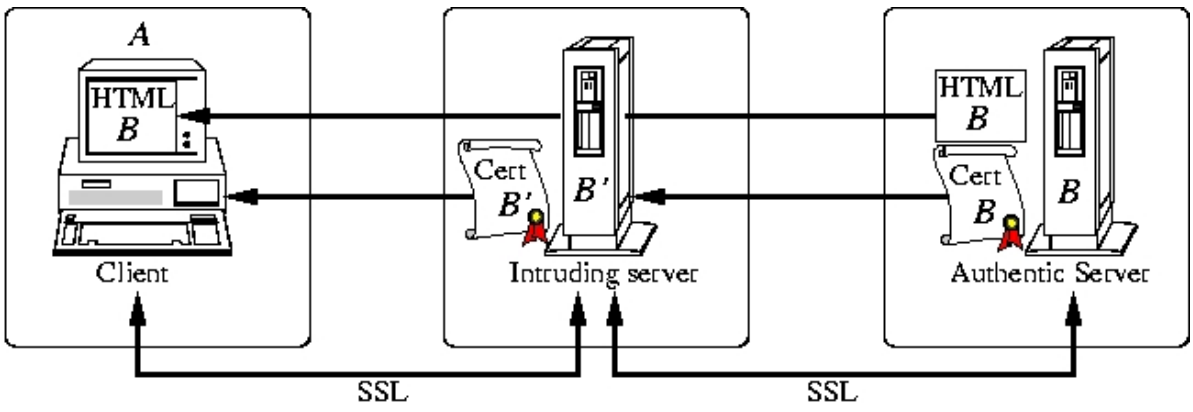

Figure 6: *Man-in-the-middle attack with false certificate*

In the attack, the intruding Web server ***B′*** acts as a relay between ***A*** and ***B*** passing the HTML pages from ***B*** to ***A*** and the requests from ***A*** to ***B***. For the attack to work user ***A*** must initially point her browser to ***B′*** instead of to ***B*** and this can happen in various ways. A false URL can for example be placed on a portal until somebody accesses ***B′*** from the portal in the belief that he or she accesses ***B***. After successful attack, the false URL can be removed in order not to leave any evidence of where the attack came from. Another possibility is to simply create a masquerade Web site and post the false URL to the major search engines so that people searching for the authentic web site find the false URL. Yet another possibility is to send malicious active contents attached to email messages in order to modify the recipient's personal Web browser and place false URLs in the file containing 'book marks' or 'favourite' Web links.

When the client ***A*** wants to establish a SSL connection to the bank the false server certificate $\mathbf{Cert}_{\boldsymbol{B}}$ gets validated by the public key contained in the CA certificate stored in the client browser. The intruding server then immediately establishes a SSL connection to the genuine bank . The intruding server ***B'*** can now simply relay all the http traffic between ***A*** and ***B*** via two different SSL connections, including possible user passwords, so that ***A*** and ***B*** think that they are communicating with each other. When user ***A*** sends a request to transfer money from her own account for paying a bill, ***B'*** is able to modify the destination account number and the amount.

The padlock icon on the browser window will be closed indicating that the Web session is using SSL. If the user in addition wants to verify that the certificates are correct by clicking on the padlock icon the browser will not give any error messages and the information regarding certificate issuer and holder will correspond to what the user expects although in reality the certificate is false. This is possible because either 1) the CA issued a false certificate, or 2) a false root CA certificate has been stored in the client machine.

A similar man-in-the-middle attack which is not based on false certificates is described in Jøsang, Patton and Ho (2001) [6]. Instead of using false certificates that attack is based on the lack of user awareness by not manually inspecting the certificates as well as the difficulty of judging whether certificates are correct even when inspected. The exploitable vulnerability in that attack is the poor browser interface and lack of good evidence.

7. IMPROVING WEB SECURITY

Active contents combined with poor Web browser integrity represents a major Web security vulnerability. Abandoning active content would drastically improve security because systems could no longer be compromised by for example down-loading and running applets. Unfortunately the result of applying this simple solution would be to set Internet Web functionality back several years and destroy the business model of most e-commerce companies. A solution that is less in conflict with Web functionality would be to constrain active content to operate in compartments of the platform where they can do no harm. We do not see digitally signing active content as a solution for the general Web user because he will never be able to make correct security decisions on the fly.

A Web browser can be “tuned” to different security levels by the security settings. This translates into either executing active contents from sources of a particular category by default, or to let the user decide by presenting him or her with a dialog box. This might give users the impression that security can tuned up and down depending on the sensitivity of the transaction. It is evident however that if a browser has ever been operated at low security it can not be assumed that it will become secure simply by going to high security settings because it is perfectly possible that it has already been compromised by malicious active content. Security settings should always be set to highest by default and appropriate warnings should be given when they are reduced.

There is a fundamental difference between knowing somebody and knowing somebody’s identity. Present PKIs only provide evidence about the latter which is not sufficient for example when deciding whether to enter into a transaction or not. Users therefore need other types of evidence about potential transaction partners on the Internet, for example by reading about them in the printed press, hearing about them on radio and TV, by physically visiting them, by talking to their representatives and by previous interactions through the real world or through the Internet. Without this additional information a PKI would be meaningless, and interacting with a securely authenticated Web site would for all practical purposes be equivalent to interacting with a complete stranger. It has however been proposed to use PKIs as a vehicle for credit rating and trust assessments about players on the Internet, see e.g. Jacobsson and Yung (1998) [4]. When visiting previously unknown Web sites the user could for example receive a credit rating of the subject Web site in addition to its authenticated identity through a digital certificate.

The SSL protocol is designed for mutual authentication in case both sides have a public key certificate. At present only Web servers have certificates so that SSL in practice does not provide user authentication. If it can be assumed that users can store private keys securely then Web security could be improved by providing

users with private/public key pairs and public-key certificates and thereby allowing user authentication by the server. Smart cards can provide a relatively secure storage device as well as a practical solution for key distribution.

Distributing public keys of root CAs as self signed certificates does not serve any purpose other than possibly allowing uniform key storage by the browser as all public keys are stored as certificates. Self signed certificates do not have the semantics of public-key certificates because there is no way of verifying their validity, and they should therefore not be called certificates at all. Sec.5 described how this makes it possible to create false root certificates. A better idea would be to actually store public keys of root CAs as real certificates signed by the browser manufacturer, and let the public key of this super root CA be hard-coded in the browser and never be written to a separate system file. In this way it would be possible to verify the validity of root certificates and the attack described in Sec.6 would no longer be possible. As an alternative the users could sign the root CA public keys using their own private keys and thus become their own personal super root CA. As long as the users' private keys are well protected the certificates stored in the browser can no longer be tampered with.

The Web PKI is no stronger than its weakest links, and it is particularly difficult to assess the practices followed by the CAs when issuing certificates. It can not be expected that commercial market forces alone will lead to high standards. Accreditation schemes for CAs under the control of governments is a natural option to investigate. Many countries are following this strategy and for example the Australian government already has established CA accreditation through it's Gatekeeper initiative [10].

PKIs are based on real trust between relying parties and CAs. Physical, cultural and political distance is a barrier to trust, and it is therefore difficult to imagine that a handful of global CAs can serve the whole world. Local CAs are in a much better position to be trusted by a community of users. Also it is much easier to operate secure out-of-band channels within a local community than across the globe.

8. CONCLUSION

In this paper we have pointed out some security weaknesses in the current Web architecture, and we have suggested some ways in which the security can be improved. Despite strong cryptographic mechanisms Web security does in our opinion not give strong protection. We find it surprising that Web security suffers from so many serious design weaknesses considering that the development of this technology has been given top priority by some of the world's most prestigious software companies. The only explanation we can think of is that Web innovation is driven by market forces, and that strong security would make the Web more difficult to use and thereby pose a threat to many existing business models. The fact that Web based e-commerce is thriving shows that strong security is not essential for its success, contrary to what many analysts are claiming.

REFERENCES

[1] H.C. Andersen. *Andersen's Fairy Tales*. Grosset & Dunlap, 1945. Translated by Mrs. Lucas and Mrs. Paull.

[2] C. Ellison and B. Schneier. Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal*, 16(1), 2000.

[3] ITU. *Recommendation X.509, The Directory: Authentication Framework* (also ISO/IEC 9594-8, 1995). International Telecommunications Union, Telecommunication Standardization Sector(ITU-T), June 1997.

[4] M. Jakobsson and M. Yung. On assurance structures for WWW commerce. In *Proceedings of Financial Cryptography 98*, 1998.

[5] Wayne A. Jansen. Guidelines on Active Content and Mobile Code – NIST Special Publication 800-28. Technical report, National Institute of Standards and Technology, 2001.

[6] A. Jøsang, M.A. Patton, and A. Ho. Authentication for Humans. In B. Gavish, editor, *Proceedings of the 9th International Conference on Telecommunication Systems (ICTS2001)*. Cox School of Business, Southern Methodist University, Dallas, March 2001.

[7] Peter A. Loscocco et al. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. In *Proceedings of the 21st National Information System Security Conference*, pages 303–314. NSA, October 1998.

[8] Microsoft. Microsoft Security Bulletin MS01-017: Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard. URL: http://www.microsoft.com/technet/security/bulletin/MS01-017.asp.

[9] Netscape. *The SSL (Secure Sockets Layer) 3.0 Protocol*. Netscape Communications Corp, 1995.

[10] Office of Government Information Technology. Australian Federal Government's Gatekeeper Initiative. URL: http://www.govonline.gov.au/projects/publickey/Gatekeeper.htm.

[11] Princeton Secure Internet Programming Team. Security Tradeoffs: Java vs. ActiveX. Technical report, Princeton University, April 1997. URL: http://www.cs.princeton.edu/sip/java-vs-activex.html.