THREAT MODELLING FOR SECURITY TOKENS IN WEB APPLICATIONS

Danny De Cock, Karel Wouters, Dries Schellekens, Dave Singelee and Bart Preneel
COSIC Research Group, Dept. Electrical Engineering-ESAT, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

Abstract: In the last couple of years, several European countries have started projects which intend to provide their citizens with electronic identity cards, driven by the European Directive on Electronic Signatures. One can expect that within a few years, these smart cards will be used in a wide variety of applications. In this paper, we describe the common threats that can be identified when using security tokens such as smart cards in web applications. We illustrate each of these threats with a few attack scenarios. This paper is part of a series of papers, written by several academic teams. Each paper focuses on one particular technological building block for web applications.

Key words: Smart card applications, security, threat modeling

1. INTRODUCTION

This paper analyzes threats that occur when smart cards are used in web applications. This analysis is part of the *Designing Secure Applications* (DeSecA) project [5], funded by Microsoft. The aim of this project is to provide an application developer with a tool that allows him to prevent the exploitation of a broad range of threats. Several academic teams investigated common threats in five areas, each focusing on one particular technological building block for web applications [1,2,3,4]. One of these is the smart card, and in particular the electronic identity card (eID card). Many European countries have started to develop such smart card projects, driven by the European Directive on Electronic Signatures [11]. We expect that these projects will be very successful in several European countries in the short term and in most European countries within a couple of years from now. This means that it is important to identify common threats before large scale software applications are developed and used. These threats involve the use of smart cards where the application developer's decisions can make the difference: well known attacks on smart cards (e.g. timing attacks, power analysis and fault attacks [8,9,10]), are not included in the scope of this study. In the following sections, we give an overview of the targeted security tokens, a list of attack entry points and the most relevant mitigation techniques. We conclude with an enumeration of threats with potential countermeasures.

2. SECURITY TOKENS

Our threat modeling for security tokens mainly focuses on electronic identity (eID) cards and smart cards used for digital signatures.

An eID card is mainly used to

- Obtain strong authentication of the cardholder, e.g. through client authentication when using a website secured with SSL. This client authentication is accomplished by having the client sign data, which is specified by the challenge-response protocol of SSL. The digital signature is computed by the cardholder's security token, e.g. a smart card, securID token or a software key store. The token only produces the digital signature after some form of cardholder verification, e.g. using a personal identification number (PIN) or a password. Once the client has successfully been authenticated, specific services may be available to the client (cf. authorization).
- Generate advanced electronic signatures. These signatures, in combination with a qualified certificate, are equivalent to hand-written signatures (cf. [11]). The production of an advanced electronic signature also relies on the cardholder verification through PIN or password validation. Note that the generation and verification of advanced electronic signatures is a very complicated matter (more legally than technically).
- Obtain information on the cardholder (e.g. address, social security number, date of birth, gender). It is common to acquire this type of information without any cardholder verification.
- Decrypt confidential data which is intended for the cardholder only.

Figure 1 gives an overview of the different entities which are active in a web application scenario that uses strong client authentication. The smart card may contain user data (e.g., e-business card, home address, birth information, picture), secret information (private keys to sign or decrypt information) and reference data (e.g. a genuine copy of the root certificate of the cardholder's certification authority). The smart card is inserted into a smart card reader which is connected to the user's PC. The smart card communicates with an application on the user's PC through the smart card reader, and the user authenticates himself to his smart card using a PIN or password, depending on the web server's requirements. This PIN or password can be entered on the smart card reader, on the keyboard of the user's PC, or it may have been cached on the user's PC for convenience reasons.

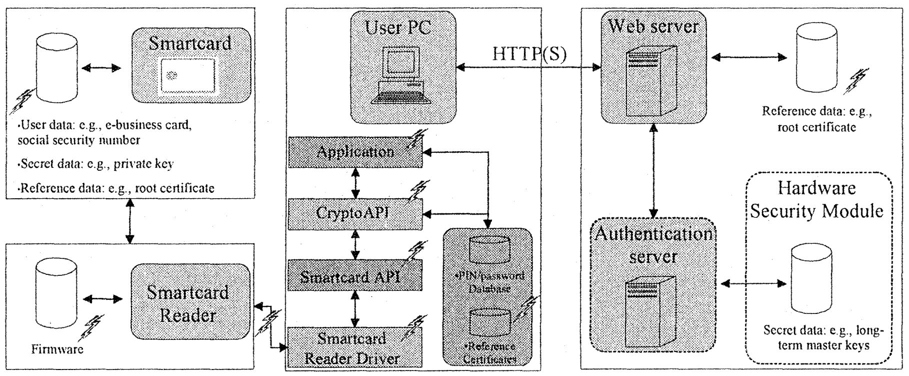
Smartcard
·User data: e.g., e-business card, social security number
·Secret data: e.g., private key
·Reference data: e.g., root certificate
Smartcard Reader
Firmware
User PC
HTTP(S)
Application
CryptoAPI
Smartcard API
Smartcard Reader Driver
•PIN/password Database
•Reference Certificates
Web server
Reference data: e.g., root certificate
Hardware Security Module
Authentication server
Secret data: e.g., long-term master keys

Figure 1. Overview of the entities in a web-based application using smart cards.

3. DESCRIPTION OF THE ATTACK ENTRY POINTS

The overview of the entities involved when using smart cards in a web application, shown in Fig. 1, points out the different attack entry points: every entity and every connection between the entities can be hijacked. It is therefore very important to use and carefully configure every entity to prevent these attacks.

In the following, we list the different attack entry points. Note that we do not mention the Crypto API, as such an API is typically available on any user machine.

3.1 The smart card itself

An obvious point of attack is the smart card itself, as it contains private or secret cryptographic keys. Different keys may be stored in the smart card, depending on the required security level of the application:

1. A smart card used for data authentication usually contains secret long-term master keys (which are commonly used to compute session keys) or private signing keys. The signing key used to produce advanced electronic signatures should ideally be generated in the smart card; it should not be possible for the signing key to leave the smart card. Before performing a private key operation, the presence of the cardholder must be verified by means of a PIN or password.
2. A smart card used for user authentication usually contains one or more signing keys or one or more secret long-term master keys.
3. A smart card used to protect the confidentiality of data contains decryption keys. As with signing keys, all usage of these keys must be protected by a cardholder verification using a PIN presentation. It is clear that a backup copy of these keys should be available to restore encrypted data if the smart card should be lost or damaged.

3.2 The smart card reader

This device should behave as expected and should forward the information exchanged between the PC and the smart card. If it is possible to upgrade the firmware of the smart card reader, this upgrade should not compromise the correct behavior of the reader.

3.3 The smart card driver software

The smart card API, which is used to address the smart card, is essential for the system to operate as specified. The introduction of a Trojan horse or a modified version of this driver software may enable an attacker to gain unauthorized access to the computer system and the smart card.

3.4 The application

The application using the Crypto API or the smart card API performs PIN management. This may be a vulnerable point. The PIN and/or passwords database should never hold PINs or passwords that may compromise the trustworthiness of advanced electronic signatures.

3.5 The user

It is clear that the cardholder also may be an attack entry point: the security of a system in which smart cards are used depends on (a) the possession of the smart card, (b) the knowledge of the appropriate PIN or password to activate some of the card's functions, and (c) the cardholder's approval of the secret or private key operation, which gives us three attack entry points. Also, the visualization of the signature/decryption request can be a point of attack.

3.6 Web servers

A web server usually delegates the validation of the authentication information of a client to an authentication server. This authentication server typically relies on a hardware security module to validate this information. Strong client authentication can only be achieved by consulting up-to-date certificate status information. A possible attack would be to prevent and/or to spoof the gathering of these data.

4. MITIGATION TECHNIQUES AND RECOMMENDATIONS

The mitigation techniques to alleviate security threats can be divided into two sub categories: technical and procedural solutions. Technical solutions can be implemented on all the different components of the system: the smart card, the smart card reader, software components on the user's PC and the web server.

4.1 Technical solutions for the security token

- An adequate access control mechanism should protect the token (smart card, hardware security module, key store...). The software designer should, if possible,
  - Limit the number of password/PIN trials.
  - Specify a fixed number of operations which can be performed without requiring a new or additional user authentication.
- Security tokens which allow the recovery of (secret) data should be avoided: it should not be possible to recover data which the user has invalidated.
- The user of a security token should have the possibility to specify for each security-critical service offered by his security token an individual

PIN or password: if a single PIN or password is used to activate the generation of an advanced electronic or authentication signature, the decryption of encrypted data, the retrieval of privacy-sensitive information, updating privacy-sensitive information, etc, then the user has no control on the actual operations that are launched to the security token.

- The data stored in a security token should be time stamped to avoid replay or substitution.
- The integrity of the data stored in a security token should be protected so that unauthorized modifications can easily be detected.
- Confidential data should not be stored in the clear.
- One should only use security tokens which support and implement state of the art cryptographic algorithms.

4.2 Technical solutions for the smart card reader

- A smart card reader with a secure PIN pad and a secure display should be used to guarantee that the information showed to the user is genuine. The secure PIN pad reader should ideally only display information which comes from the smart card, i.e., which is sent through a secure channel between the smart card and the smart card reader.
- The firmware of the smart card reader should require some kind of user authentication so that a malicious firmware update cannot take place without being noticed.

4.3 Technical solutions for the user's PC

- PINs and passwords which give access to the security token's functionality should never be stored persistently (e.g., cache, swap file, post-it), and should always be obfuscated when used in volatile memory.
- The software designer should inform the user of the reason why a PIN is requested (e.g., user authentication), which type of operation requires this PIN or password (sign, decrypt, read or write sensitive data…), and in which type of session this occurs (e.g., decryption session, authentication session …).
- The most recent accesses and usages of the token should be logged, and the integrity of this log should be protected.
- The PC application or web server should set up a secure communications channel with the security token to avoid attacks which involve a malicious smart card reader.

- The access conditions to services of the security token should be specified as restrictive as reasonable, e.g., so that only the functionality which is strictly necessary can be used.
- The integrity of all the relevant software (e.g., client applications, driver software...) should be protected, e.g., using code signing techniques which can easily be validated.
- One should not rely on garbage collector techniques to erase or clear sensitive information (cryptographic keys, user data, PINs, passwords...). This data should explicitly be reset by the software designer and developer.

4.4 Technical solutions for the web server

- All security-critical information of a web server should be stored in trustworthy hardware such as a hardware security module.

4.5 Procedural techniques

- The targeted audience should be trained to become (more) security-aware. Users who deal with security tokens must be informed such that they know how to correctly use their security tokens and how to spot potential attacks or preparations for such attacks. The users should be informed: of how to protect their security tokens both logically and physically to avoid them from being stolen, lost or damaged; and that their PINs and passwords must be kept secret and only used in the correct context.
- All procedural mitigation techniques should be clearly described and documented in easy to understand terms.
- The procedures which specify how the user should effectively protect his machine (including operating system and applications) should also specify guidelines to avoid malicious software, e.g., using a virus scanner and a personal firewall.

Note that legitimate users can also be potential attackers. It is therefore very important not to forget the impact of the threats triggered by insiders as they can easily study the (in)security of a given system, even without raising any suspicion, use the system legitimately and subsequently mount an attack on its vulnerabilities.

5. LIST OF THREATS

In this section, we give an overview of threats that are relevant to the use of smart cards in (web) applications. For each threat, we describe some scenarios that lead to the successful exploitation of the threat. We also propose some countermeasures, specific to the use of smart cards. General countermeasures, such as the installation of virus scanners, are mentioned in the previous section and are not repeated here. A much more detailed description of the threats can be found in [5].

5.1 List of threats

5.1.1 Token is no longer in the possession of the genuine holder.

If a token is no longer in the possession of the genuine holder, it is not easy to determine whether an attacker has obtained the token, or whether the token will remain inactive forever. This type of threat can be seen as a denial of service attack as the legitimate user can no longer use the token's services. An attacker may also prevent the user from using the services of the token, in order to make the token available to him/herself.

In order to avoid potential abuse of a token which is lost or damaged, the security services of that token should be revoked as soon as the legitimate user discovers the unavailability of the security token.

5.1.2 Token is damaged or unusable

If a security token is damaged, it may no longer operate correctly. This may render the key material and other sensitive data in the token unusable, both for the genuine token holder and a potential attacker, e.g., due to a hardware fault or an attacker's intervention. This type of threat is a denial of service attack as the legitimate user can no longer rely on the token's services.

Any physical damage of a security token can be detected easily. If this event occurs, the token must be revoked to avoid potential abuse of this hardware failure.

5.1.3 Secret data extraction from the token

The attacker obtains the secret data stored in a token. This leads to the existence of cloned tokens. The attacker can use this new token to impersonate the genuine user of that token. This attack is very hard to counter if the token user has not been properly trained to discover abnormal behaviour while (s)he is using his/her token.

The application designer should pay special attention to the handling of sensitive (user) data, and should never manipulate them in unobfuscated form to avoid the extraction of secret information on this data, even if the memory is dumped to disk.

5.1.4 Bypassing the access control mechanisms of the security token

The attacker may obtain a copy of the (software) security token, or obtain the PIN or password which gives access to the services of the security token, e.g., by observation of the PIN while the genuine user uses the token, or by social engineering. Once the attacker has access to the functionality of the token, (s)he can impersonate its legitimate user whenever the user uses the token in an environment controlled by the attacker.

This attack is very hard to counter as the attacker may be very well prepared, and the user may discover the abuse of his token much later than the moment of the attack.

5.1.5 Remote private/secret key operations

An attacker may set up a scenario in which he can perform operations which use the cryptographic keys of the legitimate user's security token, e.g., to generate authentication or advanced electronic signatures, or to decrypt encrypted data, without the consent of the legitimate token owner. The attacker may gain this access through social engineering or after a successful Trojan Horse attack which can either reveal the legitimate user's PIN or password, or which immediately executes additional operations which were not authorized by the legitimate token user.

This type of attack is very hard to counter and requires careful training of the token users. An additional countermeasure consists of the limitation of the number of operations that a security token can execute before a new user authentication is necessary.

5.1.6 Tampering with data in the token

An attacker may be able to alter the data stored in a security token, e.g., through the update, insertion, invalidation or removal of data without the user's explicit approval. E.g., if the token contains a root certificate of a certification authority that the token holder blindly trusts, the attacker can add an additional root certificate of a malicious party during the same session which was authorized by the user. By doing this, the malicious party abuses the authorization by the legitimate token user.

6. CONCLUSION

This article focuses on the countermeasures and recommendations, which a software developer of web applications in which security tokens are used for client and/or server authentication should keep in mind in order to counter common threats. We have described the most important threats, proposed reasonably possible attack scenarios to exploit them, and recommended countermeasures. We conclude that the awareness of the user is the most vulnerable link in the security chain which builds secure (web) applications which rely on user tokens. It is clear that the need for proper user education and training cannot be over-emphasized enough. This should not come as a surprise, as the entire purpose of using security tokens such as smart cards is actually moving the trust (and hence the target of an attack) to the user. The application developer can play a crucial role in the user awareness process. Another important element is the user's computer system: if this gets compromised in one way or another, the entire system's security guarantees are at risk.

7. ACKNOWLEDGEMENTS

This work reported in this paper was developed as part of the *Designing Secure Applications* (DeSecA) project, funded by Microsoft. Partners within this project were the Universita' degli Studi di Milano, the Technical University of Ilmenau, the University of Salford, and the COSIC and DistriNet research groups of the Katholieke Universiteit Leuven.

8. REFERENCES

[1] D. W. Chadwick. Threat Modelling for Active Directory. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), September 2004, UK, pp 203-212

[2] R. Grimm and H. Eichstädt. Threat Modelling for ASP.NET – Designing Secure Applications. In *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004),* September 2004, UK, pp 175-187.

[3] E. Bertino, D. Bruschi, S. Franzoni, I. Nai-Fovino, and S. Valtolina. Threat modelling for SQL Server. In *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004),* September 2004, UK, pp 189-201

[4] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen. Threat Modelling for Web Services Based Web Applications. In *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004),* September 2004, UK, pp 161-174.

[5] *Designing Secure Application project (DeSecA).* final report, May 2004.

[6] M. Howard and D. LeBlanc. *Writing Secure Code 2nd edition.* Microsoft Press, 2003

[7] Microsoft Patterns and Practices, *Improving Web application security: Threats and Countermeasures.* Microsoft Press, June 2003.

[8] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO '96,* pages 104-113, Lecture Notes in Computer Science 1109, N. Koblitz, Ed., Springer-Verlag, 1996.

[9] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology Conference – CRYPTO '99*, pages 388-397, Lecture Notes in Computer Science 1666, M. Wiener, Ed., Springer-Verlag, 1999.

[10]D. Boneh, R.A. DeMillo and R.J. Lipton. On the importance of eliminating errors in cryptographic computations, *J. Cryptology*, 14(2):101-119, 2001.

[11]European Parliament. *Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community Framework for Electronic Signatures.* Official Journal L 013, p.0012-0020, January 2000.