Threat Model of a Scenario Based on Trusted Platform Module 2.0 Specification

Jiun Yi Yap and Allan Tomlinson

Information Security Group
Royal Holloway, University of London
Egham, Surrey
TW20 0EX, United Kingdom
Jiun.Yap.2012@live.rhul.ac.uk, Allan.Tomlinson@rhul.ac.uk

Abstract. The Trusted Platform Module (TPM) is a device that can be used to enhance the security of web applications. However, the TPM has to be used in a proper manner in order to benefit from its security properties. A threat model will contribute towards developing a better understanding of how to use the TPM and serve as a reference for future work. In this paper, a web application scenario based on the TPM 2.0 specification is developed and the threat model is constructed using Microsoft's security development lifecycle threat modelling tool. The threats to each element in the model are analysed and the appropriate mitigations are worked out.

Keywords. Trusted Platform Module 2.0, Threat Modeling, Web Application, Secure Hardware.

1 Introduction

Protection offered by hardware security mechanisms, such as the TPM, can significantly strengthen the security of a web application. This is because the TPM provides assurance of the trustworthiness of the computing platform and offers security functions that build upon the established trust.

Several papers have been presented in the past discussing attacks on TPM 1.2 specification [1-4]. These works focused on examining TPM protocols, identifying weakness, and suggesting solutions to the problems. However, there is a need to provide easier to understand information to people who wish to use TPM technology. Threat modelling can be conducted on use scenarios based on TPM as the process helps to develop a better understanding of this technology. In addition, the results from threat

analysis and mitigations highlight potential security issues to be considered when conducting further research into the applications of TPM.

In this paper, a scenario based on the TPM 2.0 specification is crafted. Microsoft's security development life cycle threat modelling tool is then used to develop the threat model for this scenario [5]. The threats identified are analysed and the appropriate mitigations are worked out.

Paper Overview Section 2 gives a brief overview of the TPM and Section 3 explains the threat modelling methodology. In Section 4, we describe the scenario for the threat model and it is followed by threat identification and mitigations in Section 5. Section 6 concludes the paper.

2 Brief Overview of TPM

The TPM specification is developed by the Trusted Computing Group (TCG). Some software such as Microsoft's BitLocker uses the TPM to enhance its protection against cyber threats. On the other hand, there are Intel and AMD CPU architecture enhancements that leverage on the TPM to provide security functions for trusted computing. TPM 2.0 is the latest specification from TCG and it replaces the previous TPM 1.2 specification. The most recent revision to TPM 2.0 was published in March 2013 [6].

The three roots of trust, roots of trust for measurement, storage and reporting, provide the minimum functionality required to describe the attributes that contribute towards a platform's trustworthiness. The TPM aims to provide these three roots of trust. In most TPM implementation for the personal computer, the device is attached to the computer motherboard and exchanges data with the rest of the computer components through the Low Pin Count (LPC) data bus.

The key components of TPM 2.0 are shielded storage location, protected program instructions, cryptographic engines and random number generator. A Trusted Computing Base (TCB) can be a BIOS or OS that has proved to be secure and hence can be trusted. When a TCB works together with a TPM 2.0 device, they can offer the capabilities of integrity measurement and reporting, protected data storage location, certification and attestation and authentication.

The changes and enhancements to TPM 2.0 compared to the existing TPM 1.2 include: support for additional cryptographic algorithms, enhancements to the availability of the TPM to applications, enhanced authorisation mechanisms, simplified TPM management and additional capabilities to enhance the security of platform services.

3 Threat Modelling

Besides Microsoft's secure development life cycle threat modelling tool, there are an array of threat modelling frameworks and tools, such as OCTAVE from Carnegie Mellon University's Software Engineering Institute [7] and the open source TRIKE [8]. The Open Web Application Security Project (OSWAP) recommends Microsoft threat modelling process [9] and hence the Microsoft tool is chosen to be used in this work. At the beginning of the threat modelling process, the tool resolves the target scenario using a Data Flow Diagram (DFD). A DFD will show all the elements involved in that scenario. An element can be an external entity, a process, a data store or a data flow. A boundary that represents the separation between system components or privilege level will then be defined. This is followed by applying the STRIDE model to identify threat categories for every element in the DFD. STRIDE stands for ***S***poofing, ***T***ampering, ***R***epudiation, ***I***nformation disclosure, ***D***enial of service and ***E***levation of privilege. Only certain threat categories can apply to certain elements [10], see table 1.

Element Type	Threat Types					
	S	***T***	***R***	***I***	***D***	***E***
External Entity	X		X			
Process	X	X	X	X	X	X
Data Storage		X	X	X	X	
Data Flow		X	X		X	

Table. 1. STRIDE-per-element matrix from [10]

The tool will automatically generate the threat categories for each element based on table 1 but each threat category has to be analysed manually. The tool guides the identification of specific threats by providing a set of questions. For every identified threat, an appropriate mitigation should be worked out. Before the threat model report can be generated, additional information on assumptions, external dependencies and security notes can be entered into the tool. It is important to note that the threat model report is a live document and it should be constantly updated whenever a new threat is detected or there is a configuration change to the target scenario.

4 Description of Scenario

In this simplified scenario, TPM 2.0 is used to encrypt the cryptographic key used for encrypting data for sharing with a group. This allows the key to be securely exchanged. This scenario is selected because it uses TPM's shielded storage feature and is applicable to a web application situation where certain sensitive web data has to be securely shared with other user over a computer network. The scenario illustrated in figure 1 describes how a symmetric key used for encrypting data is shared using TPM's key duplication function. References to TPM commands from chapter 3 of

TPM 2.0 specification are made at key points of this process. It is noted that TPM 2.0 commands are different from TPM 1.2.

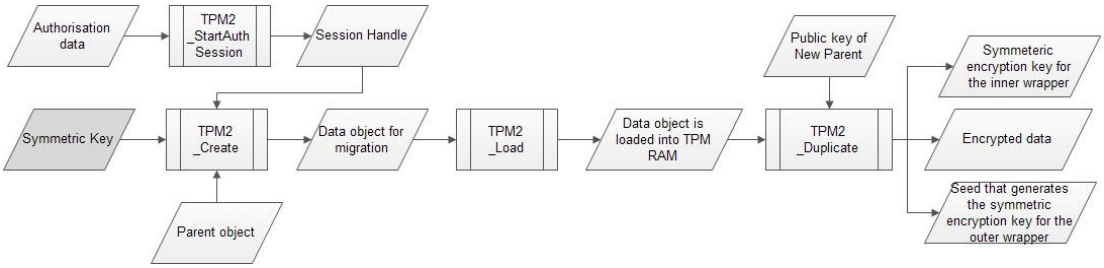

Fig. 1. To encrypt symmetric key for group share

In figure 1, TPM2_Create is used to package the key into a TPM object. But before the command can be executed, an authorisation session for the use of a parent object to create the child TPM object has to be started. Upon successful authorisation, TPM2_Create command will execute and produce a data object that contains the key. This data object will have a flag setting indicating that it can be duplicated. In addition, a user can specify an authorisation policy to control access to this data object. The next step is to load this data object into the TPM RAM using the command TPM2_Load. This command will return a handle to the key object. The final command to run is TPM2_Duplicate whereby this data object is repackaged and encrypted. The output from TPM2_Duplicate is the encrypted duplicated object, the symmetric encryption key used to encrypt the inner wrapper and a seed that generates the symmetric encryption key for the outer wrapper. The confidentiality of the seed value is protected by a public key provided by the destination TPM. These outputs can be transferred to the destination TPM using mechanism that protects the confidentiality and integrity of the duplicated object and check the authenticity and authorisation of the destination TPM.

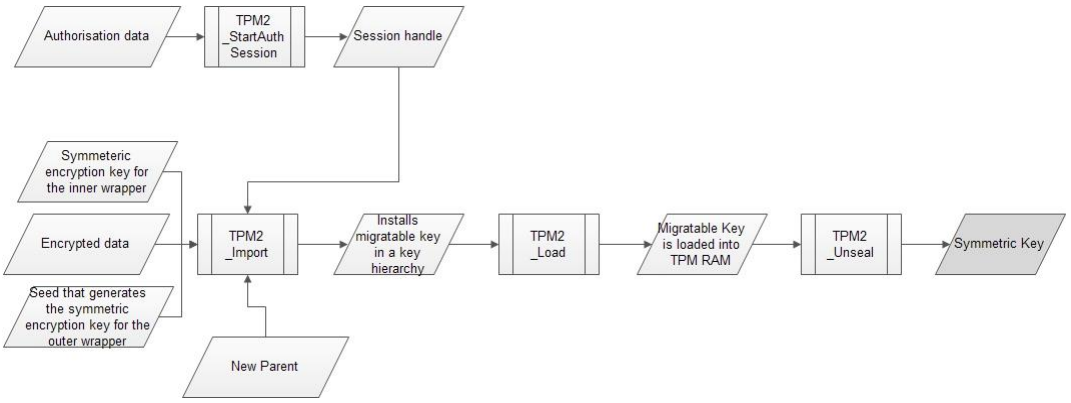

Fig. 2. To recover symmetric key for group share

At the destination TPM, the reverse is carried out. Referring to figure 2, TPM2_Import is used to transfer the duplicated object into the destination TPM. An authorisation session for the use of the new parent object is started. Upon successful authorisation, the command will execute and the duplicated object is decrypted. To protect the confidentiality of the key object, it is encrypted with an encryption key derived from the new parent. This key object is then loaded into the TPM RAM using the command TPM2_Load. A handle to the loaded key object is returned to the user. To obtain the symmetric key, the authorisation data and key object handle are provided to the command TPM2_Unseal. When this command executes successfully, the symmetric key is presented to the user.

5 Threats Identification and Mitigation

Using Microsoft‘s secure development lifecycle threat modelling tool, two DFDs were drawn to represent the scenario of encrypting and decrypting the symmetric key for group share. The DFDs are shown in figure 3 :

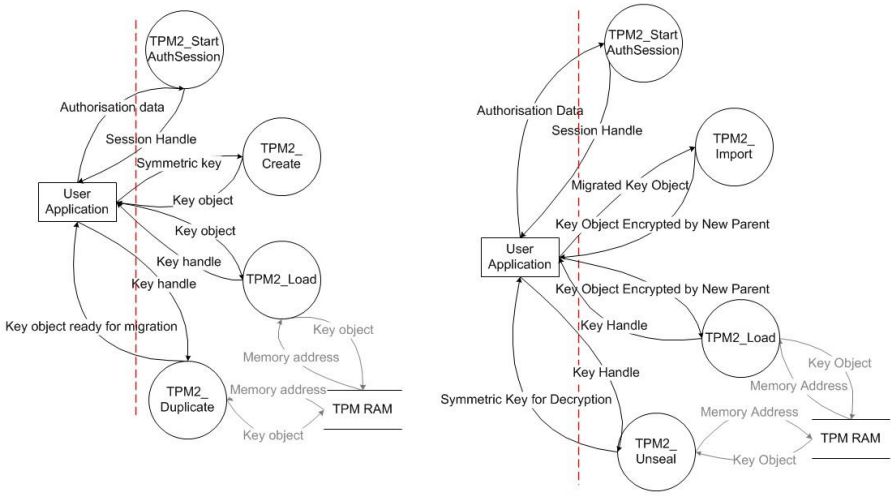

Fig. 3. DFD for encrypting symmetric key (left) and for recovering symmetric key (right).

The tool analyzed the two DFDs individually and threat categories for every element were generated. A total of 101 potential threats were identified for the process of encrypting the symmetric key for group share while a total of 96 potential threats were identified for the decrypting process. The data flow between the processes and TPM RAM are not accessible externally and hence they were not analyzed (grey coloured lines). Appropriate mitigations for all the identified threats were worked out. TPM 1.2 attacks [1-4] identified in earlier studies could not be applied directly to this threat model as the protocols and commands for TPM 2.0 have been changed.

In this paper, it is impossible to present all the threats and mitigations for this scenario but some of the more critical ones will be discussed in table 2.

S/N	Element	Type	Description	Mitigation
1	TPM2_Import	***S***	Attacker attempts to load a duplicated key object that is not generated by a TPM.	The source TPM can insert an unique identifying value into the key object when when using TPM2_Create. The destination TPM will verify the authencity of the key object by inspecting this identifying value.
2	TPM2_StartAuthSession	***I***	The cryptographic protection for the authorized sessions can be weakened if the nonce and salt value used in the generation of the session key have low entropy.	The method used by the software application to generate the nonce and salt value has to meet security requirements, for example NIST SP 800-90A. An alternate method is to use TPM's random number generator (RNG) to provide these values. However, TPM's RNG has to meet security requirements as well.
3	Key object (TPM2_Create to User Application)	***I***	The sensitive part of the key object is symmetrically encrypted using a key derived from the parent object. A random value is included in the process as an initialization vector (IV). When an object is created for duplication, the IV is set to zero. The key objects can be susceptible to cryptographic analysis if the parent object is reused multiple times.	The user application has to avoid reusing the parent object multiple times when creating an object for duplication.
4	TPM2_Create	***R***	User denies executing this command.	TPM will have to rely on the TCB to keep a log of the commands performed on TPM. The availability of a log is crucial to forensic investigation in the event of a security incident. An example of a guideline for the security management of the log will be NIST SP 800-92.

Table 2. Threat descriptions and mitigations

Since TPM's design objectives do not include protection from physical attacks, this paper will not dwell on this threat but a user should be aware of the types of physical attack [11,12] and take appropriate mitigations.

6 Conclusion

In this paper, the threat modelling process is used to develop a better grasp of TPM technology and its application. A scenario on using TPM to share a symmetric cryptographic key is crafted and the threat model is produced. Although the scenario is simple, the amount of threats and the required mitigations are substantial. Hence, it is beneficial that TPM users conduct threat modelling on their use scenarios. Meanwhile, this work highlights some potential pitfalls that should be considered when conducting further research into the applications of TPM.

Acknowledgements. We would like to thank Graeme Proudler and Liqun Chen from HP Labs, UK for their advice on TPM 2.0 specification.

References.

1. Liqun, C and Mark, R.: Offline Dictionary Attack on TCG TPM Weak Authorisation Data, and Solution. In: David, G., Helmut, R., Ahmad-Reza, S., and Claire, V. (eds.) Future of Trust in Computing. Vieweg & Teubner (2008)
2. Liqun, C. and Mark, R.: Attack, Solution and Verification for Shared Authorisation Data in TCG TPM. In: Pierpaolo, D. and Joshua D, G. (eds.) FAST 2009. LNCS, vol. 5983, pp. 201-216. Springer, Heidelberg (2010)
3. Danilo, B., Lorenzo, C., Andrea, L. and Mattia, M.: Replay Attack in TCG Specification and Solution. In: ACSAC 2005, pp. 127-137. IEEE Computer Society (2005)
4. Sigrid, G., Carsten, R., Dirk, S., Marion, A. and Rainer, P.: Security Evaluation of Scenarios Based on the TCG's TPM Specification. In: Joachim, B. and Javier, L. (eds) ESORICS 2007. LNCS, vol. 4734, pp. 438-453. Springer, Heidelberg (2007)
5. Microsoft Secure Development Life Threat Modelling Tool, http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx
6. Trusted Computing Group.: Trusted Platform Module Library Family "2.0" Level 00 Revision 00.96. 15 March 2013
7. OCTAVE Threat Modelling Tool, http://www.cert.org/octave/
8. TRIKE Threat Modelling Tool, http://www.octotrike.org/
9. The Open Web Application Security Project Threat Risk Modelling, https://www.owasp.org/index.php/Threat_Risk_Modeling
10. Shawn, H., Scott, L., Tomasz, O. and Adam, S.: Uncover Security Design Flaws Using the STRIDE Approach. MSDN Magazine, November 2006
11. Christopher, T.: Semiconductor Security Awareness, Today & Yesterday. Black Hat DC 2010
12. Bryan, P.: Bootstrapping Trust in a "Trusted" Platform. In: HOTSEC 2009, Art. 9. USENIX Association (2008)