Security Modeling with Palladio—Different Approaches

Marcus Hilbrich
marcus.hilbrich@uni-paderborn.de
s-lab – Software Quality Lab
Universität Paderborn,
Paderborn, Germany

Markus Frank
Technische
Universität Chemnitz,
Chemnitz, Germany

Sebastian Lehrig
s-lab – Software Quality Lab
Universität Paderborn,
Paderborn, Germany

Abstract

Security is never perfect, security deals with a lot of uncertainty, and security is complex. Nevertheless, security is one of the non-functional properties, that we, as software architects, have to consider. It is needed to include security in many trade-off decisions (usability, performance, costs, etc. versus security), to compare the security of different architectures, and to check whether legal constraints are meet. Thus it is demanded to include security modeling to approaches like Palladio.

In this paper, we describe two approaches to model and analyze security using Palladio. The first approach is an external one and requires to adapt Palladio. The second approach is proposed by us and does not need to modify Palladio. Furthermore, we explain why we needed to develop a new approach based on a use case and its demanded pragmatism for the model.

1 Introduction

To build secure systems, many different aspects have to be considered. The software has to be programmed without introducing security problems by decreased code quality, external libraries have to be updated to the most secure version, authorization has to be set up correctly, software (e.g. a web server) has to be configured to be invulnerable, encryption mechanisms have to be correctly used, and many more. As software architects, we have to deal with security on an architectural level. Thus, we have to analyze whether a software architecture supports security as needed and which architectural alternatives deliver the best trade-offs considering all (non-)functional properties.

In the next section, we introduce our use case (Sec. 2). Afterwards, we reflect a concept based on analyzing the mean time to the next security incident (Sec. 3). This includes to show pros and cons of the technique and to highlight the challenges for an evaluation of the models. Based on the identified cons and on the use case, we motivate our approach to model security with Palladio (Sec. 4). Our approach is based on so-called security levels and also respects concepts like trust in systems and processes. Thus, it cannot be seen as an replacement, it is an alternative modeling concept with other paradigms and no need to adapt Palladio. We follow with a comparison of the approaches (Sec. 5) and a conclusion (Sec. 6).

2 Our Use Case

In the BMBF founded project "Cloud Computing für sichere Finanztransaktionen in einer digital vernetzten Gesellschaft (SFC)"[1], we are working in the domain of financial industry with customers like banks [2].

The system of our use case has three main tasks: storing, accessing, and analyzing data. For this paper, we consider the analysis task. Supported are different analysis processes that all access encrypted data [2], decrypt the data, process the data and write back encrypted, new data. To manage the analysis, we have introduced a so-called `AnalysisBot` (Fig. 1), which provides the system-interface to perform different kinds of analysis tasks. The processing itself is not executed by the `AnalysisBot` but delegated to additional execution components. Such an execution component models the allocation of a computing resource (e.g., a Docker node (`DockerBot`) or a verified special purpose hardware called Hardware Security Module (HSM) (`HSMBot`)).

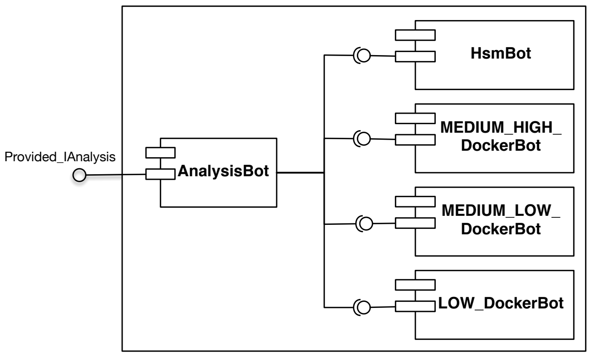

Figure 1: Extraction from the Palladio System diagram for the SFC project.

We have do deal with different kinds of high security data related to financial transactions, end users, business critical strategic information, and public or widely available aggregated status informations, to name just some data types. In addition, we have to protect software, similar to data. The software can hold information that has to be protected, e.g., an analysis process that represents business knowledge.

For our customers, trust is sometimes more important than security. A well known established process is easier to handle than a probably more secure but new one. So some processes are favored to be processed on own hardware, not on a probably more secured cloud/computing center of an external provider.

[1]http://www.vdivde-it.de/KIS/sichere-ikt/sicheres-cloud-computing/sfc

3 Mean Time to Incident Approach by Busch et al.

Busch et al. [1] provide an extension of Palladio to predict the mean time to the next security incident. In brief, you model what to protect (e.g., the data of a database), all ways to access the protected data (e.g., hacking the fronted and then hacking the non-public database), the experience of the attacker, the available knowledge about your system, and the quality of the components you are using in your system.

As result you get the mean time to the next security incident. An incident occurs whenever the data that has to be protected is accessed by an attacker. So a compromised web server is not a completed incident as long the database is not hacked, according to Busch et al.

The modeling approach from Busch et al. allows to compare the security of different architectures. Therefore, it can be used to compare architectures and determine, e.g., the most secure one. This comparison is also the pragmatism for models based on this approach. Some of the model parameters are hard to estimate, e.g., how experienced an attacker is or how good the quality of components is. Based on the fact that you want to compare architectures that use more ore less the same estimates, we still expect reasonable results.

More complicate to model are insider attacks, shortcuts, more general security problems, and components omitted from the model: Insiders have more knowledge and rights, and do not need to attack the system from outside of the system's border. Therefore, they behave differently than the modeled outside attacker. Shortcuts can occur when an attacker uses rights of a component to access data without hacking additional components. A typical example is an SQL injection on a web server. The web server's privileges are used to get the data from a database without hacking the database. In principle, such paths can be modeled but it is hard to consider all relevant paths (e.g., think of bios firmware of PCs and network equipment). The assumption that the security of components is independent is sometimes hard to establish. If components use the same libraries or middlewares, they will fail all at the same time when the library or middleware is broken (an example is the Heartbleed bug CVE-2014-0160). Thus they have to be modeled separately (e.g., as infrastructure component), but to identify all used libraries, probably reengineering is needed. Often, a system is modeled incomplete. A client system or an additional system with different purpose in the same company is, e.g., not part of the model. Such external systems use the same authentication or hold credentials. Thus, they can be used by an attacker to hack the modeled system.

A discussion in our group was the topic of security by obscurity. Busch et al. have argued that it takes time to get the needed information of a system while e.g. Pfitzmann [3] argues that security by obscurity is no valid security concept at all.

It would be interesting to evaluate the concept from Busch et al. by analyzing real world security incident of relevant companies. Such an evaluation would show whether the pragmatism of the model based on the approach of Busch et al. is adequate or not. It is also clear that it is hard to get enough valid data for such an evaluation. In addition, it could be also used to get to reasonable values for the used parameters to model similar systems.

Even if the approach from Busch et al. has some open issues, using it can help to get to an informed decision regarding the choice of an architecture based on security. In addition the issues in above discussion, we discovered the following challenges that should optimally be tackled for a practical relevance:

- For many customers, understanding the meaning of the mean time to incident is difficult.
- Whenever the system changes (e.g. security updates) or new attack vectors emerge (knowledge of the attacker changes) an update of the model is required.
- The approach is centered on data—business processes and knowledge represented by software is not directly covered.
- It is not checked whether an architecture fulfills a minimum security specification (e.g., those by legal regulations).
- Often it is needed to model the trust in a component not only the security.
- Different protection goals for different kinds of data are not addressed.
- It is unclear what happens after an incident occurs.

4 Security Level Based Approach

In the following, we describe our approach that does not require changes to Palladio. The approach follows our use case and addresses the drawbacks we identified for the approach by Busch et al. Moreover, our approach is based on security levels. The required security levels of all information and knowledge (processed and stored data and running of analysis processes) are annotated to the data. The security levels of components can be reached by, e.g., cryptographic methods (like signing and encryption) or organizational methods (e.g., by fences, physical locks, or four-eyes principle). Whether a data set can be processed on a component is determined based on a mapping. This mapping is also modeled with Palladio. Thus, we build an architecture that respects the required security level of all information and knowledge by following a known concept to analyze security.

Security levels are annotated to system calls (via an additional parameter) within Palladio's usage scenarios. The hardware is also categorized along the security levels. Because the mapping is manually modeled and not generated, the hardware abstraction is not annotated yet. In our use case, the mapping is modeled by the Service Effect Specification (SEFF) of the `AnalysisBot` (see Fig. 2). The SEFF uses a `GuardedBranch` to select an `ExternalCallAction` of a component that allocates computing and storage demands on a hardware with the demanded security level. Thus, the system's analysis service (left side of Fig. 1) can process data of different security levels. An appropriate interface is provided by the `AnalysisBot` that realizes the selection of the execution hardware and accesses this hardware by additional components.

The components for differently secured hardware are allocated on different servers. This allows to realize, e.g., different computing speeds, prices, and network bandwidths for each hardware.

This very easy modeling approach allows to respect different demands for security like trust and legal constraints. It can, e.g., be modeled that some tasks

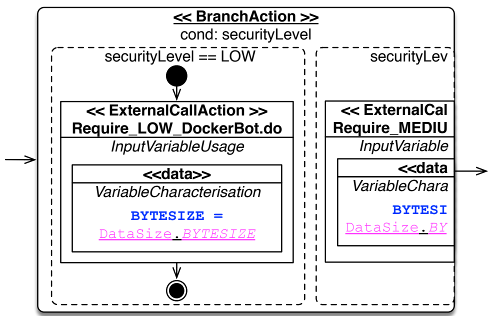

Figure 2: Extraction of the Branch from the SEFF to select the hardware based on security level.

are only executed on hardware located in computing centers to establish trust for high secure data and to respect the demand for hardware located in, e.g, Germany, to process individual related data. This allows to respect the demand for trust of our customers. Those decide which hardware installations are considered trustworthy for which purpose.

An other important aspect of our project is to have a software architecture that supports a security certification process. Therefore, we restrict which data is processed on which hardware/location. Different locations have to be certified individually for the required security level.

Similar to the security level of data, we handle the security level of software. We set the demanded security level not only based on the data, but also based on the processing software. Further, the output of a process can be annotated with another security level than the input data. This is, e.g., needed to model that aggregated data (like a monthly report) has lower security demands while the result of an analysis probably needs a higher security level than the original data.

In our models, a security level is not static. To hold a security level, a set of requirements have to be fulfilled by organizational and physical security measures and have to be continuously evaluated. Thus, security is considered as a continuous process.

5 Comparison of the Approaches

Purpose: Busch et al. focus on comparing different architectures based on their security, while we focus on other non-functional attributes (e.g. costs) of architectures that have a reasonable security and trust.

Managing changes: A change of the software (e.g. security updates) or additional known attack vectors require to change models based on Busch et al. In our approach, changes or reactions to new attacks are part of the continuous security validation and do not change the model.

Incident management: Busch et al. consider a security incident when the protected data is compromised. We consider every partly successful attack as incident that probably needs a revalidation of the security levels.

End of lifetime: When the data to protect is accessed by an attacker, the knowledge of the attacker is high enough to break the system whenever he wants, according to Busch et al. Afterwards the architecture needs to be redesigned. In our model we have to change the security of some computing centers but can continue with our model.

Security of processes: Based on our approach, we can model where a program can be executed. Thus, the program code, that holds secure knowledge, can be protected based on the modeled architecture.

Key management: Based on our approach, we modeled key management processes. Key generation can be only performed on special purpose hardware, other hardware is only allowed to hold short time temporary keys.

Used metrics: Busch et al. use a set of hard to estimate metrics like the knowledge of an attacker or the security based quality of components, while we use security levels that are e.g. based on certification demands or already established trust.

Side channel attacks: This kind of an attack is based on a combination of soft- and hardware. So for Busch et al. this is part of the quality of a component. Our approach uses certification processes to consider side channel attacks for soft- and hardware of selected security levels.

Different protection demands: Our approach addresses different strong protection goals for different categories of data within one architecture. Busch et al. do not address this.

6 Conclusions

We have identified numerous differences between the approach of Busch et al. and our one. This is caused by different requirements for both approaches. For our use case, we need to consider different security levels for data, software, and computing centers. We want to model architectures that respect the required security aspects and levels. To check whether an architecture is more secure than another is not so important in our case. We are more interested in an architecture that allows to satisfy demands to performance, scalability, and elasticity with minimal costs and which is secure enough. Therefore, we provide a modeling approach that does not need changes of Palladio and helps to advance our SFC project.

References

[1] A. Busch, M. Strittmatter, and A. Koziolek. Assessing security to compare architecture alternatives of component-based systems. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 99–108, Aug 2015.

[2] M. Hilbrich, R. Petrlic, and S. Becker. Towards a Secure Cloud Usage for Financial IT. In D. Hühnlein, H. Roßnagel, R. Kuhlisch, and J. Ziesing, editors, *Open Identity Summit 2015, 10-11 , 2014 November 2015, Berlin, Germany*, pages 153–158. GI-Edition, Lecture Notes in Informatic, 2015.

[3] A. Pfitzmann, editor. *Steganography Secure against Cover-Stego-Attacks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.