

Trade-off Analysis of Misuse Case-based Secure Software Architectures: A Case Study

Joshua J. Pauli¹, Dianxiang Xu²

¹ Dakota State University, Madison, SD USA 57042

² North Dakota State University, Fargo, ND USA 58105

Abstract. Based on the threat-driven architectural design of secure information systems, this paper introduces an approach for the tradeoff analysis of secure software architectures in order to determine the effects of security requirements on the system. We use a case study on a payroll information system (PIS) to show the approach from misuse case identification through the architecture tradeoff analysis. In the case study, we discuss how to make tradeoff between security and availability with respect to the number of servers present.

1 Introduction

In today's software world, there is a great need for making software resistant to potential attacks [5,6]. Based on the threat-driven architectural design of secure software [12], this paper introduces an approach for the tradeoff analysis of secure software architectures. Tradeoff analysis is needed to determine the effects of non-functional requirements on the system. For most software there is no clear mapping from requirements specifications to architecture design. With use cases, misuse cases, and mitigation use cases as the source, requirements include both the functional uses and security related issues of the system. Our previous architectural analysis approach takes the completed use case model into account when proposing candidate architectures [12]. This ensures the architectures map directly back to the requirements specifications. This paper will take that research one step further by analyzing the proposed architectures. The paper is organized as follows: section 2 summarizes misuse cases, architecture identification, and the ATAM. Section 3 covers the identification of architectures. Section 4 covers the Architecture Tradeoff Analysis Method as it relates to the PIS case study. We conclude in section 5.

2 Background

The three most important topics discussed in this paper are misuse cases, architecture analysis, and the tradeoff analysis of the architectures. Misuse cases, i.e. use cases with hostile intent, appear to be a new avenue to elicit security requirements

[1,2,4,11,13,14]. Use case modeling is a proven method for the elicitation of, communication about, and documentation of functional requirements [7]. The integral development of use cases and misuse cases provides a systematic way for the elicitation of various system requirements, both functional and non-functional [2]. A critical issue is how misuse case based security requirements specification can further facilitate the design and implementation of software systems where security is a major concern. Prior papers presented an approach to bridge the gap between misuse case based security requirements and high-level architecture design [12]. We treat identification of threats as part of requirements elicitation and model them with misuse cases [7,11]. We then drive architecture design by dealing with identified security threats in the process of application decomposition; this is opposite to the threat modeling approach that determines and mitigates threats later in development [15]. The treatment of security threats in the early phases of development can reduce overall development cost due to the absence of a variety of vulnerabilities. We also map between the use/misuse cases and the architectural components, since the software security requirements are taken into account in the architecture design, the architecture specification is an invaluable resource for detailed design, implementation, and validation. The Architecture Tradeoff Analysis Method (ATAM) is used as a way to ensure that the architecture is behaving in the intended way [3]. The ATAM is a structured technique for understanding the tradeoffs inherent in the architectures of software systems, thus providing a way to evaluate an architecture's fitness with respect to security [10].

3 Architecture Identification

The idea of identifying architectures from use case models was first covered in our prior research in an attempt to bridge the gap between requirements specification and architecture design [12]. Having misuse case models created early in the development cycle allows for both the misuses and mitigations to be used throughout the development cycle. To begin, the actors and their use cases are identified to gain an overall context of the system. Because of space constraints, this case study will only look at a part of the payroll information system (PIS). We are also interested in which use cases have a relationship with security because it is these use cases that open the door for possible attacks. Listed are the users of the PIS, the use cases for each, and identification, (*), of security-related use cases. The Payroll Staff user has the use cases Enter User Information (*), Edit User Information, View Payroll Reports, Complete Administrative Tasks, and Log on to the System (*). The General Employee has the use cases Log on to the System (*), Request Payroll Information (*), and View Payroll Information. The Web Developer user has the use cases Create Code (*), Create Web Pages (*), and Create / Manage User Accounts. The Payroll Auditor user has the use cases Access Audit Entries (*), Review Audit Entries, Report on Audit Entries, and Log on to the System (*). Known misuses cases are also identified after use cases have been identified. The Malicious User misuser has the possible misuse case of View Confidential Payroll Information. The Spoof Computer misuser has the possible misuse case of Spoof Computer Identification. The Attacker misuser has the possible

misuse cases Launch DoS Attack, Manipulate Payroll Information, Upload Rogue Code, Upload Rogue Web Pages, and Elevate Privileges. Once the use cases and the possible misuse cases have been identified, a use case model can be constructed as figure 1 shows. To gain a further understanding of the use case model, textual templates are used to convey more details about each use case and misuse case [4]. Because of space constraints, no textual case is shown. This same approach was applied to the misuse cases and the mitigation use cases.

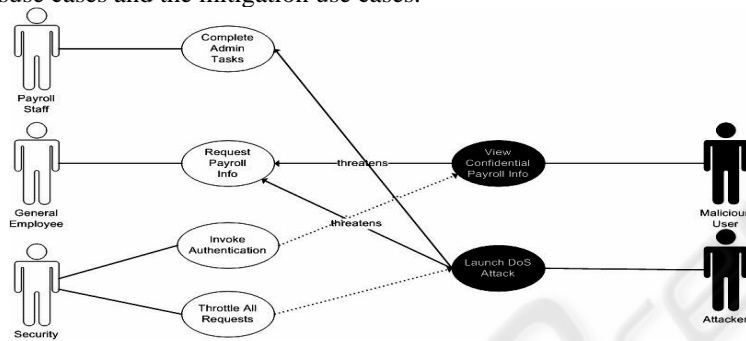


Fig. 1. High level use case model with users, misusers, and mitigating actors

From detailed textual use cases and high level use case models, a more detailed use case model is created. Because of space constraints only one detailed use case is presented as figure 2 introduces. Sequence diagrams were also constructed to show the order of steps that each use case, misuse case, and mitigation use case take. These diagrams also help in the identification of candidate architectural components [13,15]. No sequence diagram is shown, but every use case, misuse case, and mitigation use case was modeled in this manner.

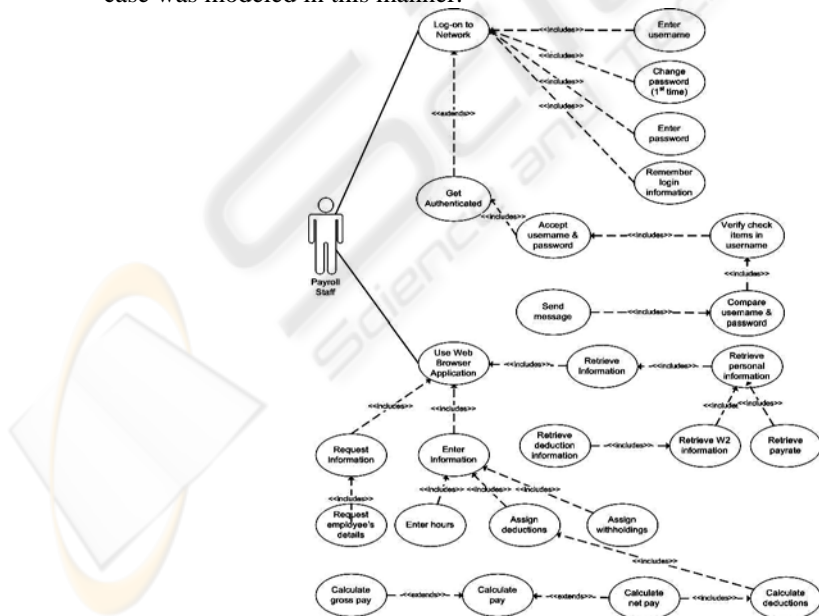


Fig. 2. Detailed use case for “Complete Administrative Tasks” for the Payroll Staff actor in PIS

Architectures can now be proposed that are constructed from these UML notations [9]. This is done by inspecting the detailed complete use case model, the detailed sequence diagrams, and the textual use cases. The issues in the architecture (fig. 3) that will be examined are what effect the number of servers will have on the system in terms of availability.

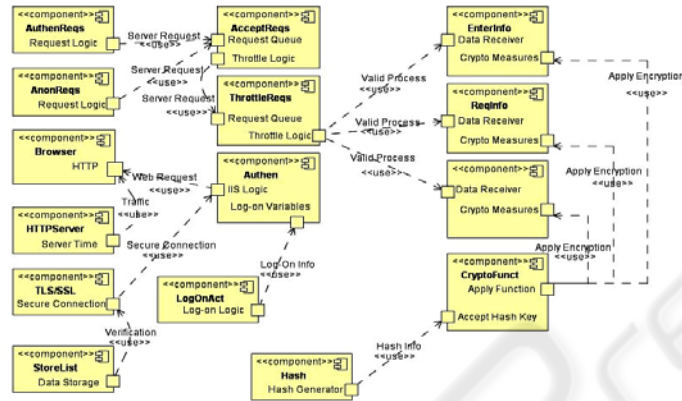


Fig. 3. Candidate Architecture for the PIS

4 Trade-Off Analysis of Architecture

The main idea of completing a tradeoff analysis is to compare the effects of architectural decisions on the entire system [10]. There is a natural tradeoff between security and availability in all software. This analysis follows the research done in [3] and can be used as a guide for equations and values used concerning the ATAM. The first major step of ATAM is to identify sensitivities; any component that could incur changes based on the findings of the ATAM. These include “EnterInfo” (changes to this component), “Browser” (changes), “Crypto” (inclusion of this component), “Attach Hash” (inclusion), and “Digital Signature” (inclusion). Next, tradeoffs, defined as those architectural decisions that impact the sensitive components, are identified. Our only tradeoff is the number of servers in the PIS because as the number of servers increases it will adversely affect the security of the system by creating possible holes to exploit. At the same time, as the number of servers increases the availability of the PIS increases because numerous servers can act as back-ups for other servers. To start the tradeoff analysis, reasonable initial values are needed for the availability and the security of the PIS. The values for availability are Number of Servers (S) set to 2, Failures per Year (F) set to 2, and Repairs per Day (R) set to 2.

According to [3], the fraction of downtime in availability analysis can be figured as:

$$D = (2 * F^2) / ((R^2 + 2) * F * (R + 2) * F^2) \tag{1}$$

For a hardware failure (power supply), assume that 1 to 2 failures occur per year and require a 4 hour visit by a technician to repair. The worst case would be F=2 failures/year and P=2 repairs/day (730 repairs/year). For a software failure (O/S

crash) it ranges from 8 to 24 failures/year and requires restarting the server (10 minutes). The worst case would be F=24 failures/year and P=144 repairs/day (52,560 repairs/year). Solving each equation for the highest possible values for Hardware Fault (HF), Software Fault (SF), and the combined downtime (D) yields the following values

$$HF = (2 * 2^2) / (730^2 + 2 * 2 * 730 + 2 * 2^2) = 1.4930164157E-5 \quad (2)$$

$$SF = (2 * 24^2) / (52,560^2 + 2 * 24 * 52560 + 2 * 24^2) = 4.1662483059E-7 \quad (3)$$

$$D = HF + (1 - HF) * SF = 1.5346782767E-5 \quad (4)$$

After converting these fractions of a year into hours, the results are shown in table 3.

Table 3. PIS availability findings with 2 servers present

Hardware failures (4 hours repair)			Software failures (10 minute repair)			Combined Failures	
Failures per year	Availability (1-HF)	Hrs. down/year	Failures per year	Availability (1-SF)	Hours down/year	Availability	Hours down/year
2	.99998507	.1307	24	.99999958	.0036	.99998465	.1344

PIS security analysis is accomplished by calculating the probability of a successful attack. Once these values are known, it is up to the stakeholders to decide what is the optimal number of servers for the PIS to be as secure and as available as possible. The initial values that are reasonable for this case study are Exposure Window (EW) set at 90 minutes, Attack Rate (AR) set at .08 systems per minute, TCP Intercept (TCPI) set at .5 probability, Kill the Server Connection (KSC) set at .75 probability, Kill the Server (KS) set at .25 probability, Decrypt the Data (DD) set at .0006 probability, Replay (RP) set at .047 probability, and Key Distribution (KD) set at .082 probability. These values would differ for each system, but are good estimations for the PIS [3]. Figuring the likelihood of an attack succeeding is just a matter of calculating the probability of each attack. “TCP Intercept” Attack with encryption is figured in equation 5.

$$TSA = (((AR * TCPI) * EW) * DD) + (((AR * TCPI) * EW) * RP) + (((AR * TCPI) * EW) * KD); \quad (5)$$

$$TSA = (((.08 * .5) * 90) * .006) + (((.08 * .5) * 90) * .0470) + (((.08 * .5) * 90) * .0820) = .4860$$

“Kill Server Connection” and “Kill Server” are shown in equations 6 and 7.

$$TSA = (((.08 * .75) * 90) * .006) + (((.08 * .75) * 90) * .0470) + (((.08 * .75) * 90) * .0820) = .7290 \quad (6)$$

$$TSA = (((.08 * .25) * 90) * .006) + (((.08 * .25) * 90) * .0470) + (((.08 * .25) * 90) * .0820) = .2430 \quad (7)$$

The amount of likely successful attacks on the system within the 90 minute window of opportunity are TCPI = .486, KSC = .729, KS = .243; totaling 1.458. In 60 minutes the total becomes .972 attacks. The tradeoff between security and availability is summarized in the table 4 with the sensitivity to the number of servers in the PIS being negatively correlated with respect to the number of servers in the PIS. Based on

these findings, it is our opinion that having 2 servers in the PIS would be the optimal solution based on this tradeoff analysis. This would virtually eliminate any down time, while still keeping successful attacks under 1 per hour.

Table 4. Sensitivity to the number of servers in the PIS

	1 Server	2 Servers	3 Servers
Combined downtime	27.922 hours/year	.1344 hours/year	.0010 hours/year
Hardware: (F=2/yr, R=730/yr)			
Software:(F=24/yr, R=52560/yr)			
Successful Attacks F = 10/year	.486 attacks/hour	.972 attacks/hour	1.944 attacks/hour

5 Conclusion

This case study shows an added step to our previous research [12] by completing a tradeoff analysis on the architecture derived from a use case model. This tradeoff analysis, which followed the ATAM, helped solidify our overall approach [3]. It is important to realize that the architecture was constructed from the system information presented in the use case models and sequence diagrams. This bridges a gap between requirements specification and architecture design.

References

1. Alexander, I. Initial industrial experience of misuse cases. In Proc. of IEEE Joint International Requirements Engineering Conference, (2002) pp. 61-68
2. Alexander, I. Misuse cases: Use cases with hostile intent. IEEE Software, (2003) pp. 58-66
3. Barbacci, M., Carriere, J., Kazman, R., Klein, M., Lipson, H., Longstaff, T., and Weinstock, C. Steps Toward an architecture trade-off analysis method: Quality attribute models and analysis. CMU/SEI-97-TR-29, (1997)
4. Firesmith, D. Security use cases. Journal of Object Technology, (2003)Vol. 2, No. 3, 53-64.
5. Høglund, G. and McGraw, G. Exploiting software: How to break code. Addison-Wesley. (2004)
6. Howard, M. and LeBlanc, D. Writing secure code. Microsoft Press. 2nd edition, (2003)
7. Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, (1994)
8. Kantorowitz, E., Lyakas, A., and Myasqobsky, A. Use case-oriented software architecture. CMC03 (2003)
9. Kazman, R., Abowd, G., Bass, L., and Clements, P. Scenario-based analysis of software architecture. IEEE Software. pp.47-55, (1996)
10. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., and Carriere, J. The architecture tradeoff analysis method. In Proc. of the 4th International Conference on Engineering of Complex Computer Systems (ICECCS98), (1998)
11. McDermott, J. and Fox, C. Using abuse case models for security requirements analysis. In Proc. of the 15th Annual Computer Security Application Conference, pp. 55-66, (1999)
12. Pauli, J., and Xu, D., Threat-driven architectural design of secure information systems. In Proc. of ICEIS'05. Miami, May 2005. To appear.

13. Ruhe, G. and Eberlein, A. Trade-off analysis for requirements selection. *International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, No. 4 (2003) 345-366.
14. Sindre, G. and Opdahl, A.L. Eliciting security requirements by misuse cases. In *Proc. of TOOLS Pacific 2000*, pp. 120-131, (2001)
15. Swiderski, F. and Snyder, W. *Threat Modeling*. Microsoft Press. (2004)



SciTeP Press
Science and Technology Publications