Secure Architecture Design Automation

Katja Tuma[1]

University of Gothenburg, Sweden

Abstract. Architectural threat analysis is a pillar of security by design and is routinely performed in companies. To this day, companies perform threat analysis using techniques like STRIDE, where security experts manually identify and assess security threats. Techniques such as STRIDE aim towards maximizing the completeness of discovered threats. This leads to identifying more threats than can be addressed (due to budget constraints). However, at that point analysts have spent precious time on low-prioritized threats already, which is inefficient. Therefore, there is a need for a more efficient use of the allocated resources. Because secure architectures consider security as a cross-cutting concern, testing, maintenance and compliance checking provide continuous feedback on system state. Automation of architectural analysis can not only reduce the manual labor, but also cater to late-stage security related activities. This short paper discusses two recent attempts to automate architectural threat analysis and discusses their limitations.

Keywords: Architectural threat analysis, Automation, Security threats

1 Introduction

Security-by-design is a practice within organizations which enables planning for security in early phases of the product life-cycle. Our experience in the industry reveals that experts have little time and resources available to get the job done. Designing for security begins with a disciplinary regard to best practices, security standards [13] [15] [14] [19] and derived security policies. As a result several secure design notations have emerged in the past [9]. Some of these notations have emerged in the model-driven development domain. A large body of knowledge has shown that security-design models can be used as a basis for generating systems and their architectures. If models are extended with security semantics, formal signatures can be used to reason around security-related properties of systems. Security policy enforcement in the model-driven domain has been researched by Basin et al. [5].

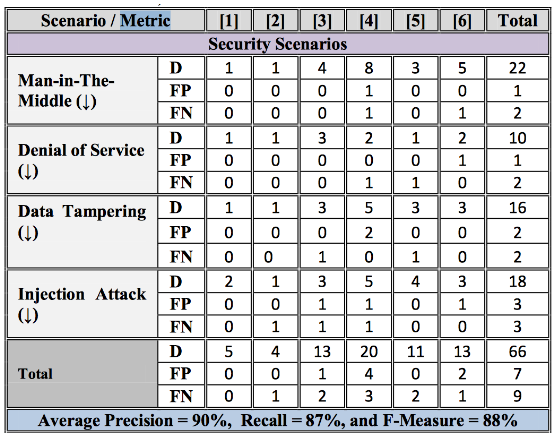
However, as security issues progressed apace with technological and organizational innovations, complementary approaches emerged to strengthening secure design practices. Threat analysis is a method for identifying, analyzing and prioritizing threats of early software architectural models. STRIDE is most commonly used to this aim. This method's systematicity and repetitiveness results in a so called threat explosion, where too many unimportant threats are identified and analyzed [20]. This results in a waste of time for the analyst, hence there is a need for more focused approaches to emerge. Even though, this technique is not formalized, a variety of tool support exists (most commonly used is Microsoft Threat Modeling Tool [1]). Schaad et al. [21] implemented a tool for light weight semi-automated security analysis of architectural diagrams, adopting the STRIDE technique. Apart from STRIDE there are other methods used to perform threat analysis. In particular, we observed a significant amount of semi-automated approaches for attack-centric methods. Closely related to goal-oriented requirements engineering methods, architectural analysis has been studied in the context of automating attack tree [18] [10], graph [22] and paths [11] generation. Aforementioned approaches aim toward a static architectural analysis, where the implemented solution is not the main focus. For a more complete list of attack and defense modeling methods, we refer the reader to the work of Kordy et al. [17]. Complementary to architectural security patterns, so called *problem frames* [16] [6] [12] and *treat patterns* have emerged. Abe et al. [2] proposed to model negative scenarios (as defined by the CC standard [13]) in a semi-automated way with threat patterns and use them during business process modeling.

We continue to describe two recent initiatives to automate architectural analysis and discuss them briefly.

2 Automated Software Architecture Security Risk Analysis using Formalized Signatures

Almorsy et al. [4] introduce a risk-centric architectural analysis approach. Their approach, accompanied by a tool, supports a security risk analysis by means of formalized signatures of security scenarios and metrics. Authors develop a prototype software tool using the Eclipse Modeling Framework (EMF), which takes as input a (sub)set of design, architecture and code level artifacts. Using formalized signatures defined with the Object Constraint Language (OCL), the tool is able to identify signature matches in the architectural model. In addition to formalizing attack scenarios, the prototype tool also formalizes signatures of a set of security metrics (attack surface, fail securely, compartmentalization, least privilege, defense in depth, isolation metric). The authors have experimentally evaluated their approach on several open-source applications which resulted in a number of discovered flaws and metrics (not shown here), as shown in Figure 1a. Based on the evaluated signatures for attack scenarios and security metrics, the authors implement a final trade-off analysis. This makes it possible to compare the risk levels for several architectural models at a time.

The main disadvantage of the developed approach is that the correctness of the analysis depends heavily on the formalized signatures. In consequence, such rules need to be tailored for each system by security experts. What is more, despite the mention of the possibility to limit the type and quantity of input artifacts, it seems that in this case, the signatures would not have enough information to be evaluated. Nevertheless, authors' evaluation shows promising



Scenario / Metric		[1]	[2]	[3]	[4]	[5]	[6]	Total
Security Scenarios								
Man-in-The-Middle (↓)	**D**	1	1	4	8	3	5	22
	FP	0	0	0	1	0	0	1
	FN	0	0	0	1	0	1	2
Denial of Service (↓)	**D**	1	1	3	2	1	2	10
	FP	0	0	0	0	0	1	1
	FN	0	0	0	1	1	0	2
Data Tampering (↓)	**D**	1	1	3	5	3	3	16
	FP	0	0	0	2	0	0	2
	FN	0	0	1	0	1	0	2
Injection Attack (↓)	**D**	2	1	3	5	4	3	18
	FP	0	0	1	1	0	1	3
	FN	0	1	1	1	0	0	3
Total	**D**	5	4	13	20	11	13	66
	FP	0	0	1	4	0	2	7
	FN	0	1	2	3	2	1	9
Average Precision = 90%, Recall = 87%, and F-Measure = 88%								

(a) Average precision, recall and F-measure for discovering flaws with OCL signatures.

CWE Top 25 (D)etected/(V)ulnerable			
		App A	App B
89	1	D	V
78	2	D	V
120	3		
79	4	V	V
306	5	V	
311	8		V
352	12	V	V
22	13	D	

CWE Top 25 (D)etected/(V)ulnerable			
		App A	App B
327	19	V	
134	23		
190	24		
759	25	V	V
288		V	
319			V
602		V	V

(b) Detected threats and vulnerabilities related to CWE entries and supporting rules.

Fig. 1. Presented results from selected papers.

results in reverse engineering and analyzing the architecture of several open source applications.

3 Automatically Extracting Threats from Extended Data Flow Diagrams

J. Berger et al. [8] have recently proposed a new approach for an automated architectural risk analysis. Their work is supported by a tool which automatically extracts architectural vulnerabilities based on an extended DFD (EDFD) notation and proposes known mitigations. The EDFD contains necessary information about the communication channels and asset source/target. The obtained threat model is created with the use of formalized rules, which are build based on information obtained from lowering EDFDs to simple graphs. Initially, a security expert creates a catalog of security-related design patterns. Additionally, the expert defines the knowledge base (rules for detecting flaws) and creates an EDFD schema. Next, the domain expert uses the pattern catalog and creates an instance of the EDFD schema. Finally, the developed rule checker uses the predefined rules to find vulnerabilities in the instantiated EDFD and matches them with known mitigations from vulnerability repositories (CAPEC, CWE). Evaluation results show that they were indeed able to find certain threats 1b, however a more sound evaluation approach would be more insightful.

The authors recognize the need to limit the time spent by security experts and therefore separate the responsibilities to include the expert only when needed. However, the knowledge base rules are used to discover only cataloged vulnerabilities, as opposed to finding all possible threats. Moreover, their approach does not handle threat explosion problem. Because of these reasons, additional effort by security experts is needed to verify the analysis after the threat model is built.

4 Discussion and concluding remarks

In this short paper we briefly described and discussed two novel approaches for automated architectural analysis in the design phase of the SDL. With the changing trends and architectural styles (such as microservice architectures), there is a need for an efficient way to model and analyze security in the design phase. In order to do so, more information needs to be obtained from the architectural model (as was shown by J. Berger et al.). However, a large number of secure design automation approaches are only semi-automated, leaving the difficult task of finding relevant threats to the analysts. Moreover, the industry is often skeptical with adopting such approaches, as automation can also be counter-productive, if it steers the analyst in a wrong direction. In addition, there is a lack of understanding the level of threat granularity (i.e. when does the analysis stop?). We plan to further investigate the matter, especially in connection with defining rules for compliance checking [3], [7] (feedback loop to the implemented architecture).

References

1. Sustainable Application Security microsofts new threat modeling tool. https://blog.secodis.com/2016/07/06/microsofts-new-threat-modeling-tool/, accessed: 2017-05-15
2. Abe, T., Hayashi, S., Saeki, M.: Modeling security threat patterns to derive negative scenarios. In: Software Engineering Conference (APSEC), 2013 20th Asia-Pacific. vol. 1, pp. 58–66. IEEE (2013)
3. Abi-Antoun, M., Wang, D., Torr, P.: Checking threat modeling data flow diagrams for implementation conformance and security. In: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering. pp. 393–396. ACM (2007)

4. Almorsy, M., Grundy, J., Ibrahim, A.S.: Automated software architecture security risk analysis using formalized signatures. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 662–671. IEEE Press (2013)
5. Basin, D., Clavel, M., Doser, J., Egea, M.: Automated analysis of security-design models. Information and Software Technology 51(5), 815–831 (2009)
6. Beckers, K., Hatebur, D., Heisel, M.: A problem-based threat analysis in compliance with common criteria. In: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on. pp. 111–120. IEEE (2013)
7. Berger, B.J., Sohr, K., Koschke, R.: Extracting and analyzing the implemented security architecture of business applications. In: Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on. pp. 285–294. IEEE (2013)
8. Berger, B.J., Sohr, K., Koschke, R.: Automatically extracting threats from extended data flow diagrams. In: International Symposium on Engineering Secure Software and Systems. pp. 56–71. Springer (2016)
9. van den Berghe, A., Scandariato, R., Yskout, K., Joosen, W.: Design notations for secure software: a systematic literature review. Software & Systems Modeling pp. 1–23 (2015)
10. Birkholz, H., Edelkamp, S., Junge, F., Sohr, K.: Efficient automated generation of attack trees from vulnerability databases. In: Working Notes for the 2010 AAAI Workshop on Intelligent Security (SecArt). pp. 47–55 (2010)
11. Chen, Y., Boehm, B., Sheppard, L.: Value driven security threat modeling based on attack path analysis. In: System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on. pp. 280a–280a. IEEE (2007)
12. Hatebur, D., Heisel, M.: Problem frames and architectures for security problems. In: International Conference on Computer Safety, Reliability, and Security. pp. 390–404. Springer (2005)
13. ISO/IEC common criteria for information security evaluation. Standard, International Organization for Standardization (ISO) and Enternational Elecrotechnical Commission (IEC) (2009)
14. ISO/IEC Information technology-security techniques-Information security management systems-Overview and Vocabulary. Standard, International Organization for Standardization (ISO) and Enternational Elecrotechnical Commission (IEC) (2009)
15. ISO/IEC Information technology-security techniques-Information security management systems-Requirements. Standard, International Organization for Standardization (ISO) and Enternational Elecrotechnical Commission (IEC) (2005)
16. Jackson, M.: Problem frames: analysing and structuring software development problems. Addison-Wesley (2001)
17. Kordy, B., Piètre-Cambacédès, L., Schweitzer, P.: Dag-based attack and defense modeling: Dont miss the forest for the attack trees. Computer science review 13, 1–38 (2014)
18. Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K.: Security attack analysis using attack patterns. In: Research Challenges in Information Science (RCIS), 2016 IEEE Tenth International Conference on. pp. 1–13. IEEE (2016)
19. Security and Privacy Controls for Federal Information Systems and Organizations. Standard, National Institute of Standards and Technology (NIST)) (2013)
20. Scandariato, R., Wuyts, K., Joosen, W.: A descriptive study of microsofts threat modeling technique. Requirements Engineering 20(2), 163–180 (2015)
21. Schaad, A., Borozdin, M.: Tam 2: automated threat analysis. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 1103–1108. ACM (2012)

22. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on. pp. 273–284. IEEE (2002)