Security: A Critical Quality Attribute in Self-Adaptive Systems

Anthony Peruma
Rochester Institute of Technology
Rochester, NY
axp6201@rit.edu

Daniel E. Krutz
Rochester Institute of Technology
Rochester, NY
dxkvse@rit.edu

ABSTRACT

Self-Adaptive Systems (SAS) are revolutionizing many aspects of our society. From server clusters to autonomous vehicles, SAS are becoming more ubiquitous and essential to our world. Security is frequently a priority for these systems as many SAS conduct mission-critical operations, or work with sensitive information. Fortunately, security is being more recognized as an indispensable aspect of virtually all aspects of computing systems, in all phases of software development. Despite the growing prominence in security, from computing education to vulnerability detection systems, it is just another concern of creating good software. Despite how critical security is, it is a quality attribute like other aspects such as reliability, stability, or adaptability in a SAS.

CCS CONCEPTS

• **Security and privacy** → *Software security engineering*; • **Computer systems organization** → *Self-organizing autonomic computing*; • **Software and its engineering** → *Software design engineering*;

KEYWORDS

Self-Adaptive Systems, Security, Autonomous Systems

ACM Reference Format:
Anthony Peruma and Daniel E. Krutz. 2018. Security: A Critical Quality Attribute in Self-Adaptive Systems. In *SEAMS '18: SEAMS '18: 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems , May 28–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3194133.3194134

1 INTRODUCTION

Quality attributes have been defined as "Factors that affect run-time behavior, system design, and user experience" [1]. Some examples of quality attributes include *Maintainability*, *Performance* and *Scalability* [4]. The idea of quality attributes is relatively generic, one that relates to virtually all software systems ranging from small scripts to large enterprise applications. Security is considered to be one of the quality attributes in a system [1].

The security of a SAS is often a priority due to its autonomous, and typically business and mission-critical nature. A SAS will often need to react to a security incident in the span of only a few milliseconds, and may need to balance various concurrent security concerns with different mission-critical operations. In a SAS, security affects all facets of initially defined quality attributes. For example, the run-time behavior of a system may be significantly impacted by security-related tactics, or even tactics that take security implications into their decision-making process.

Some people will contend that security is a *special* concern of the system, one that deserves a far different type of attention and consideration than 'normal' system requirements. While many systems will have a monumental set of security concerns, it is merely another type of quality attribute that system engineers must account for. Additionally, although SAS pose unique opportunities and challenges for engineers due to a myriad of reasons, it is still a type of software application like all the others. *While security in a SAS is a critical concern, it can be considered in the same regard as any other quality aspect of the system.* In the following paper, we will present a systematic argument why security is just another quality attribute in SAS.

2 WHY SECURITY IS A QUALITY ATTRIBUTE

There are several primary reasons why we firmly believe that security is a quality attribute in SAS.

(1) **Vulnerabilities are caused by engineering failures** Security failures do not typically occur due to completely new and profound types of vulnerabilities or attacks methods. Security problems overwhelming occur due to a failure to properly address specific types of known attacks. For example, as devastating as HeartBleed[1] was, it was merely another type of cryptographic weakness. Most vulnerabilities invariably occur due to engineering failures which may exist at virtually any phase of the development process ranging from the requirements gathering to maintenance phases. Requirements failures could lead to vulnerabilities due to a requirement not being correctly defined or adhered to [7, 8]. The primary difference between a SAS and a conventional system is that the requirement failure can stem from many other possible causes including tactics, or other system decisions. However, an engineering failure is still the primary cause of the problem. Software engineering practices should adequately account for all necessary quality attributes of any system, SAS are no exception to this rule. Security is just another quality attribute of a SAS that should be accounted for during the software engineering process, from the requirements to the maintenance phase. Although the idea of mitigating all threats merely using proper engineering efforts may be a simplistic view, we believe that proper engineering can eliminate the vast majority of vulnerabilities. The designer of the SAS also needs to pay special attention

[1] http://heartbleed.com/

SEAMS '18, May 28–29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5715-9/18/05.
https://doi.org/10.1145/3194133.3194134

to ensure that the adaptation decision does not directly lead to a vulnerability; which is often a challenging task. While no system can ever be assumed to be 'completely secure', adhering to proper engineering processes and guidelines can help alleviate as many security vulnerabilities and concerns as possible. Ultimately, a properly engineered system must be able to balance the cost of implementing a security feature, against the benefits produced from this feature. Regardless of the type of risk (safety or security), any system should strive to minimize these risks as much as possible. Whether it is a risk for a web system in accepting specific types of user input or a drone identifying a target, minimizing this risk is imperative for any type of system.

(2) **Vulnerabilities are just another type of software quality challenge** Flaws in any software system can lead to an infinite variety of problems. These could range from small visual mistakes, to miscalculations leading to serious system failures. Security vulnerabilities are just a type of quality issue that could affect a system, albeit with different consequences [2]. For example, in a cloud-oriented SAS, system flaws may be caused by the system not correctly accounting for the *tactic latency* of bringing a new virtual machine (VM) online to handle the increased workload. In this case, provisioning a new VM to handle an increased workload could take two minutes. If the SAS does not account for this latency, and the system fails then this could be considered a poor software quality attribute. This flaw could mean that the system's users encountered a slow or even unavailable system. However, this flaw could have security implications as well. For example, if an attacker wanted to create a denial-of-service attack (DoS attack), they could exploit this flaw and make the system unavailable. Although the flaw remains the same, it has both functional and security implications.

(3) **Self Adaptive Systems offer unique challenges and opportunities, but are still software** SAS offer tremendous benefits over 'conventional' software. Systems are capable of self-management and making autonomous decisions that were impossible not long ago. Despite these benefits, a SAS is still a software system. The software is still developed using the same basic development practices as all other software. Requirements are still gathered at the inception of the system, and designed, developed and tested just like any other software application. Numerous existing works consider security to be a quality attribute in 'traditional' software systems [1], so why should security in a SAS be considered to be anything different?

One concern for the SAS is that the adaptation strategy could potentially be used against the system. For example, an attacker could collect a sufficient amount of information to reverse engineer the adaptation strategy in an attempt to deceive the system. This specific situation is primarily unique to SAS in the sense that the vulnerability is tactic associated. Hence the vulnerability associated with the execution of one tactic need not necessarily be the same as the execution of another tactic. Therefore, at the time of tactic definition, software architects need to understand the drawbacks and potential vulnerabilities associated with implementing this tactic and base their decisions on the balance of trade-offs and benefits of the tactic.

(4) **Security concerns are just another type of uncertainty** Many SAS contain uncertainty that may be caused by environmental or system variabilities, or even attackers [6]. While uncertainty comes in an infinite amount of variations, it is ultimately up to the system to ensure that any uncertainty is held in a manageable, and safe state. An example uncertainty is the response time of a remote server. There are two types of uncertainty that must be addressed by a SAS. *External* uncertainty is caused by environment or domain factors. An example of this is weather conditions for a drone. *Internal* uncertainty is caused by the challenges of determining the impact of adaptation requirements on the quality objectives of the system. One example is understanding the implications of making one tactic-based decision over another [3].

Uncertainty in SAS is being researched in a variety of different manners [5]. Many aspects of security can broadly be considered to be another element of uncertainty, as both internal and external uncertainty can have security-related implications. An example of external uncertainty is that while specific attack patterns can be anticipated, it is impossible to predict how all adversaries will attempt to comprise a system, or even what aspects of the system they will attack. An example of internal uncertainty are system decisions that place the system in a vulnerability state. For example, connections being made with devices that shouldn't be connected to, or sensitive data being unnecessarily exposed. A SAS must be ready to appropriately react to both security and non-security related uncertainties. A challenging aspect of uncertainty is when an action is benign (trusted), or malicious. This is a concern that any software system has, and is not unique to SAS. All software systems need to know if an actor/action is potentially malicious and account for this uncertainty.

REFERENCES

[1] Chapter 16: Quality attributes. https://msdn.microsoft.com/en-us/library/ee658094.aspx.

[2] O. H. Alhazmi, Y. K. Malaiya, and I. Ray. Measuring, analyzing and predicting security vulnerabilities in software systems. *Computers & Security*.

[3] N. Esfahani and S. Malek. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 214–238. Springer, 2013.

[4] I. Gorton. Software quality attributes. *Essential Software Architecture*.

[5] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl. Efficient decision-making under uncertainty for proactive self-adaptation. In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pages 147–156. IEEE, 2016.

[6] A. J. Ramirez, A. C. Jensen, and B. H. C. Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '12, pages 99–108, Piscataway, NJ, USA, 2012. IEEE Press.

[7] M. Salehie, L. Pasquale, I. Omoronyia, R. Ali, and B. Nuseibeh. Requirements-driven adaptive security: Protecting variable assets at runtime. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 111–120. IEEE, 2012.

[8] E. Yuan, N. Esfahani, and S. Malek. A systematic survey of self-protecting software systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 8(4):17, 2014.