Agree to Disagree

Security Requirements Are Different, But Mechanisms For Security Adaptation Are Not

Thein Tun
The Open University
Milton Keynes, United Kingdom
thein.tun@open.ac.uk

Amel Bennaceur
The Open University
Milton Keynes, United Kingdom
amel.bennaceur@open.ac.uk

ABSTRACT

We describe a dialogue between a proponent and an opponent of the proposition “security is not just another quality attribute in self-adaptive systems”. The dialogue is structured in two steps. First, we examine whether security requirements are different from other system-level requirements. Our consensus is that security requirements require specific methods for elicitation, reasoning, and analysis. However, other requirements (such as safety, usability and performance) also require specific techniques. Then, we examine the adaptation mechanisms for security and compare them with other properties. Our consensus is that most adaptation techniques can be applied to maintain security and other requirements alike.

CCS CONCEPTS

• **Security and privacy**; • **Software and its engineering**;

KEYWORDS

Security requirements, Self-adaptation

1 INTRODUCTION

For each issue considered, the proponent and the opponent both make their arguments, which need not be in conflict.

2 SECURITY REQUIREMENTS

Adaptive systems seek to maintain requirements satisfied despite change and uncertainty by using mechanisms to change the structure and behaviour of the system. Therefore, good understanding of the differences between security and other requirements to manage adaptation is important to clarify whether self-adaptation for security requires different techniques and mechanisms. We now consider possible ways in which security requirements might be different from other requirements.

2.1 Elicitation

P1. System-level security requirements are typically very difficult to define. John Rushby [6] argues that they are like counterfactuals — we do not really know what our security requirements are until some attack has occurred. This suggests that we need to know everything an attacker can do — which seems impossible at the system level. On the other hand, we can define requirements for performance and safety properties fairly accurately.
O1. Self-adaptive systems often operate in uncertain environments characterised by incomplete knowledge which challenges all types of requirements at different levels of granularity. Consider for example an autonomous drone; ensuring its stability would involve assuming an acceptable range of wind speed. It is rarely the case that one has the resources to consider all possible events in the environment *a priori*. This justifies the need for self-adaptive systems.

2.2 Attacker's Perspective

P2. Capabilities of an attacker are fundamental to the design of security properties. The notion of semantic security relies on the difficulty of attackers solving certain classes of computation problems efficiently. Although it is easy to think of an attacker as part of the environment, the fact that an attacker has ill intention, and might observe the system in ways not intended by the system designer (such as observing the power consumption of a machine or the response time to an online request) is unique to security. Very few safety and performance threats emerge as frequently as those for security. The difference is often described as programming Satan's computer versus Murphy's computer [1]. Other requirements do not assume such a strong adversary in the environment.
O2. The environment can often act as an adversary or an attacker. This is exacerbated by human agents in the environment who can interact with the system in unforeseen way. Consider for example usability, it is hard to predict how users will use and misuse the system, not necessarily in a malicious way. Self-adaptive system aim to change its behaviour at runtime as new knowledge about the environment and users is acquired.

2.3 Refinement

P3. Secure properties do not preserve well when they are refined: in fact, additional security issues tend to emerge when lower-level specifications are made. For example, protocol specifications cannot really address something like timing attacks without knowing the properties of the programming language used to implement it. Although some of the security-related decisions can be made before the system is designed and implemented (such as the choice of programming language), many of the security requirements can be identified only after the system design is clear.
O3. The feasibility and implementation of other requirements such as performance is often considered through refinements. For example, deciding to use multi-threading to improve the performance of an application can only be done at the implementation stage and will depend on the programming language.

SEAMS '18, May 28–29, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5715-9/18/05.
https://doi.org/10.1145/3194133.3194151

2.4 Hierarchical properties

P4. Like safety and performance, we tend to think of security properties at the system level. However, security properties have a strong hierarchical nature. Higher level security properties are built from lower level ones, with typically stronger formal guarantees. For example, some authentication protocols are built from primitives such as hash functions and block ciphers. Vulnerabilities in the lower-level propagate upwards. It does not seem that other system-level properties having such strongly hierarchical nature.

O4. All requirements are refined into smaller requirements and operations. Performance might be achieved through enabling multiple tasks to execute concurrently. This in turn is achieved by implementing concurrency protocols, which them use concurrency primitives. In constrained environments, higher mathematical guarantees can be provided. I would argue that functional requirements can be refined into finer-grained primitives, which considering the constrained settings in which they operate can often provide stronger guarantees compared with complex system-level properties.

2.5 Multi-Layering

P5. In practice, security is achieved by multiple mechanisms in different parts of the system. For example, access to a file may be controlled by restriction to join the network, restriction to connect to a node in the network, restricted privileges once connected to a node, and so on. This is the idea behind "defense in depth". Other system properties do not need to be satisfied in the same way.

O5. The adaptation of quality requirements is often contextual. For example, ensuring some performance requirements would require load balancing and the deployment of additional servers. Even functional requirements can implement graceful degradation depending on the resources and context.

2.6 Assurance

P6. Consider an operating system you normally uses. What behaviour of that system shows that it is secure? You would experience security breaches, for example, when you have ransomware on your machine. Facts such as no one has reported security vulnerabilities for that operating system in the last few months tell little about how secure your software will be in future. It seems impossible to validate the requirements for system security. Contrast that with the performance or usability of the operating system. Here you will have a lot to say about your experience of usability and performance. A user never experiences security but other properties.

O6. Assurance can only be considered when the requirements are clearly defined. Any requirement must be testable [4]. Therefore, security requirements need to be formulated in a way that enables us to validate and verify them.

2.7 Ethics and Values

P7. Security properties have very clear political and moral implications [5]. One can use security tools against state surveillance and censorship, promote freedom of speech online, and achieve a degree of anonymity. Although requirements such as those for safety often have a legal and moral basis, these concerns are restricted to those engaged in the development of the system and rarely concern its users directly. In security, the personal values and ethical behaviour of programmers matter a lot to the users of their software.

O7. Even functional requirements may need to consider ethical issues such as those leading to addiction or how recommender systems can lead to biases. Hence the debate around values and ethics for artificial intelligence systems and software engineering in general [3]. Overall, as software systems become an integral part of society, further considerations of their impact on individuals and society in general must be considered.

3 MECHANISMS FOR SECURITY ADAPTATION

Let us assume that security requirements are indeed different from other requirements. How about the mechanisms for adapting those requirements?

P8. Once implemented in code, security properties can be adapted using the same tools and techniques as other properties [2]. Whether we use machine learning algorithms or rule-based reasoning techniques or controller synthesis approaches, they can be used to adapt any system behaviour.

O8. Although many adaptation techniques can be reused [7], security involves a great degree of change, which is not only technical (e.g., the discovery of new vulnerabilities), but also organisational or business related (e.g., new security policies). Reacting to these changes rapidly is paramount and is key to minimising the damage of discovered attacks. Assurance is also more challenging when it comes to security. Not only do we need to ensure that security and quality requirements are met but also that no vulnerabilities have been introduced during adaptation.

4 CONCLUSION

Security requirements might be different in nature from other requirements as they need different methods and tools to reason about the attacker, delimit the scope, and test an implementation against those requirements. However, the adaptation of those requirements may rely on the same techniques and tools as for other properties. Therefore, a well-engineered self-adaptive system maintains the satisfaction of all requirements, be they security or not. Adaptation makes security more complex without fundamentally changing it.

ACKNOWLEDGMENTS

We thank Bashar Nuseibeh for comments on an earlier draft and the support of ERC and EPSRC (Johnny, Jenny and SAUSE).

REFERENCES

[1] Ross Anderson and Roger Needham. 1995. Programming Satan's computer. *Computer Science Today* (1995), 426–440.

[2] Amel Bennaceur, Thein Than Tun, Arosha K. Bandara, Yijun Yu, and Bashar Nuseibeh. 2017. Feature-driven Mediator Synthesis: Supporting Collaborative Security in the Internet of Things. *ACM Trans. on Cyber-Physical Systems* (2017).

[3] Maria Angela Ferrario, William Simm, Stephen Forshaw, Adrian Gradinar, Marcia Tavares Smith, and Ian C. Smith. 2016. Values-first SE: research principles in practice. In *Proc. of International Conference on Software Engineering*. 553–562.

[4] Suzanne Robertson and James Robertson. 2012. *Mastering the requirements process: Getting requirements right.* Addison-wesley.

[5] Phillip Rogaway. 2015. The Moral Character of Cryptographic Work. *IACR Cryptology ePrint Archive* 2015 (2015), 1162.

[6] John Rushby. 2001. Security requirements specifications: How and what. In *Symposium on Requirements Engineering for Information Security (SREIS)*, Vol. 441.

[7] Eric Yuan, Naeem Esfahani, and Sam Malek. 2014. A Systematic Survey of Self-Protecting Software Systems. *ACM Transactions on Autonomous and Adaptive Systems, TAAS* 8, 4 (2014), 17.