Analyzing the Security Significance of System Requirements

Donald G. Firesmith
Software Engineering Institute
dgf@sei.cmu.edu

Abstract

Safety and security are highly related concepts [1] [2] [3]. Both deal with the protection of valuable assets from harm, and both do this by avoiding, detecting, and responding to incidents that can cause such harm. In both cases, the dangers (hazards and threats respectively) that can cause or enable such incidents to occur are identified and the associated risks are analyzed in order to ensure that these risks are mitigated to acceptable levels.

Safety engineering is typically concerned less with requirements than with its downstream activities (e.g., architecting, design, coding, testing) because much of hazard analysis is based on the existence of an architecture, the components of which can cause accidents if they fail. Yet one central safety engineering technique is to categorize the system requirements based on their safety significance and use this categorization to determine the associated level of development processes necessary to assure a corresponding acceptable level of safety risk.

Based on the similarity between safety and security, this position paper advocates using a similar process to categorize the security significance of non-security requirements and use this information to ensure that an adequate development process is used to assure an acceptable level of security risk.

1. Introduction

Although engineering security requirements is a very significant task [4], this paper deals instead with the security aspects of the other requirements of a system: the non-security requirements. Some functional, data, interface, and quality (e.g., interoperability, performance) requirements clearly have significant security ramifications, whereas others do not. For example, some requirements may address the storing and manipulation of sensitive information, whereas other requirements may address the display of publicly accessible information of relatively little importance. Clearly from a security standpoint, it is more important to properly implement those requirements with major security ramifications than it is to implement those requirements with little or no security significance. The safety community has developed a standard approach to solving this problem of requirements relevance, and the similarity between safety and security implies that it would be well worth considering if something similar can be done for security.

The next section of this position paper briefly points out the many similarities between safety and security. The third section describes how a safety (hazard) analysis is commonly used to categorize requirements so that the limited project resources are concentrated on properly implementing the most "safety-critical" requirements. The fourth section proposes the use of an analogous approach for categorizing requirements in terms of their security ramifications. The fifth section discusses other related approaches. The final section raises questions to be considered by security engineers concerning adapting these safety concepts and techniques to security and suggest certain benefits that may result if they do.

2. Similarities between Safety and Security

Safety and security are closely related quality factors in a system's quality model because they both describe important related attributes or characteristics of the system's overall quality [1] [2] [3]. Safety and security are subtypes of defensibility quality factor because they are both primarily concerned with the protection of *valuable assets* (e.g., people, property, services, and the environment) from *harm* (i.e., significant negative consequences). As illustrated in figure 1, the essential difference between safety and security is that safety deals with *accidental harm*, whereas security deals with *malicious harm*, which is harm resulting from attacks or probes by someone or something (e.g., viruses) playing the role of attacker.

This harm to valuable assets occurs during *incidents*, which are unplanned, unintended, unauthorized, (but not necessarily unexpected) events or series of related events that could cause unintended harm to one or more valuable assets. *Safety incidents* are either *accidents* (harm occurs) or *near misses*, whereas *security incidents* are successful *attacks* (harm occurs), unsuccessful attacks (harm does not occur), and *probes* (i.e., preparations for attacks).

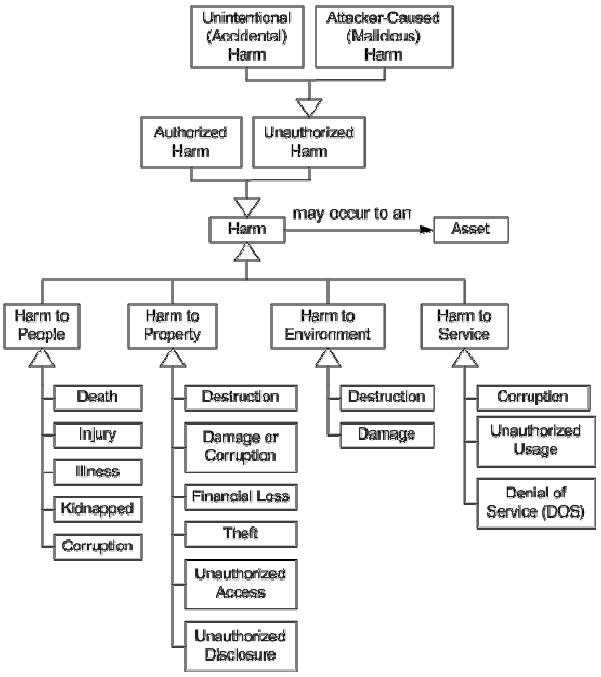

Figure 1: Types of Harm to Valuable Assets

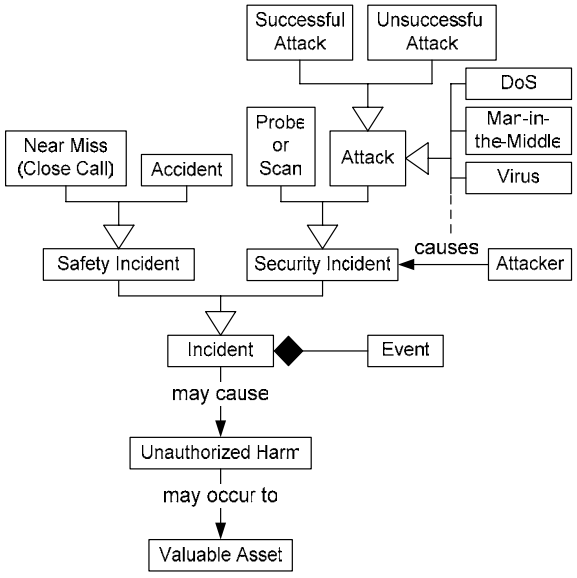

Figure 2: Types of Incidents

In order to prevent these undesired incidents, one typically begins by identifying the dangers that can cause them. Specifically, a *danger* is one or more conditions, situations, or states of a system that in conjunction with conditions in the environment of the system can cause or contribute to the occurrence of one or more related incidents. As illustrated in figure 3, dangers are classified into *hazards* (which can cause safety incidents) and *threats*, which can cause security incidents or survivability incidents (e.g., military attacks).

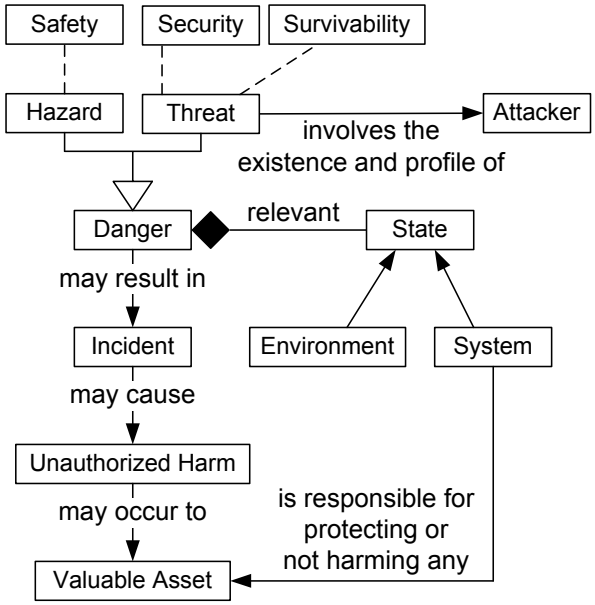

Figure 3: Types of Dangers

One reason that dangers are analyzed (e.g., via hazard analysis or threat analysis) is to determine the associated risks so that risk mitigation can occur. *Risk* is usually defined as the probable magnitude of the potential harm to one or more assets that can occur due to a danger and is conservatively estimated as the maximum credible harm multiplied by the estimated probability that the associated accident / successful attack occurs. And as before, risks can be classified as *safety risks* due to hazards, *security risks* due to security threats, and [military] *survivability risks*.

Once the system is [partially] implemented, it may have *vulnerabilities* (e.g., a lack of input validation) that can cause both accidents as well as enable successful attacks. Occasionally, the same *controls* used to overcome these vulnerabilities can work as *safeguards* (safety), *countermeasures* (security), and *defenses* (survivability). Finally, the safety engineers on an endeavor construct a *safety case* [5], which documents their:

- *Claims* that the system is adequately safe (i.e., meets its safety requirements)
- *Arguments* for their claims of adequate safety.
- Compelling *evidence* backing up their arguments.
- Any *assumptions* on which their arguments may be based.

3. Safety Analysis and Categorization

Safety engineers typically perform the following types of safety analyses:

- *Asset analysis* to determine which assets are valuable to legitimate stakeholders and how valuable they are.
- *Harm analysis* to determine the credible types and severities of the accidental harm that can occur to these assets, whereby harm severities are typically categories of the magnitude of harm.
- *Incident analysis* to determine the kinds of accidents that can cause harm to the assets as well as the kinds of near misses that need to be addressed.
- *Hazard analysis* to determine the hazards (i.e., hazardous conditions) that can lead to safety incidents as well as their causes and consequences. Note that the phrase "hazard analysis" is typically used to refer to the union of all of these different types of safety analysis.
- *Risk analysis* to categorize the safety incidents and hazards by levels of safety risk, such as intolerable, undesirable, as low as reasonably practical (ALARP), and acceptable.

Once safety risks have been identified and categorized, safety engineers use the results of these analyses to assign corresponding:

- *Safety Integrity Levels (SILs)* to individual requirements or collections of related requirements.
- *Safety Evidence Assurance Levels (SEALs)* to the associated architectural, design, and code components that implement these requirements.

Figure 4 shows an example categorization of harm severity, incident/hazard occurrence frequency, and resulting safety risks / safety integrity levels. For example, an accident with catastrophic consequences that is estimated to occur frequently would have an intolerable safety risk. Any requirement (or set of related requirements that would cause such a safety risk would have an intolerable risk, an associated SIL value of 4, and have to be either changed to lower the risk or else dropped as a requirement. On the other had, a requirement that could result in a remote chance of causing a critical accident would be assigned a SIL value of 2, which means that steps would need to be taken (e.g., architectural decisions, coding standards) to reduce the risk to *as low as reasonably practical* (ALARP). Finally, any requirement that only has a remote chance of causing an accident, the severity of which is negligible would have a SIL value of 1, meaning that the safety risk is acceptable and that no extra steps would need to be taken to reduce the risk.

Safety Risks / Safety Integrity Levels (SILs)					
	Frequency of Accident / Hazard Occurrence				
Harm Severity	Frequent	Probable	Occasional	Remote	Implausible
Catastrophic	Intolerable	Intolerable	Intolerable	Undesirable	ALARP
Critical	Intolerable	Intolerable	Undesirable	ALARP	ALARP
Marginal	Undesirable	Undesirable	ALARP	ALARP	Acceptable
Negligible	ALARP	ALARP	ALARP	Acceptable	Acceptable

Figure 4: Safety Risk Categorization Matrix

As mentioned previously, corresponding to each safety risk and requirements safety integrity level would be a corresponding safety evidence assurance level (SEAL) that would determine both the extra measures that would need to be taken to assure acceptable safety as well as the associated evidence that would need to be collected to support the safety certification (e.g., flight certification) of the system. For example, these measures could include anything from the formal specification of the requirements and formal proofs of correctness of the resulting design and implementation to the use of Fagan inspections, a safe subset of the programming language, and specific types of testing and test completeness criteria.

4. Security Analysis and Categorization

A complete security analysis is a valuable part of managing security risks [6] [7]. As with safety, security analysis should include:

- *Asset analysis* to determine the assets that are sufficiently valuable to legitimate stakeholders to be worth protecting from attackers.
- *Harm analysis* to determine the types and severities of the malicious harm that can occur to these assets so that the level of investment in security countermeasures will be commensurate with the value of the assets being protected.
- *Incident analysis* to determine the kinds of security attacks that could cause malicious harm as well as the kinds of probes to be addressed.
- *Threat analysis* to determine the threats (i.e., threatening conditions) that can lead to security incidents. Because security involves attacks, these threats can be considered to be the existence of

attackers with specific profiles (e.g., means, motive, and opportunity) or the proactive tools of their trade (e.g., viruses and worms).

- *Risk analysis* to categorize the security incidents and threats by levels of security risk, such as intolerable, undesirable, as low as reasonably practical (ALARP), and acceptable.

Security engineers usually specify general kinds of security goals regarding confidentiality, integrity, non-reputability of transactions, and availability in the face of denial of service (DoS) attacks. They also rely on implementing industry best practices such as the use of standard countermeasures (e.g., firewalls, encryption) and performing security testing (e.g., penetration tests).

But given the similarity between safety and security, security engineers should consider taking the following steps to better address security engineering during requirements engineering:

1. *Security Analysis*. Perform security analysis *early* in the development cycle when it can still influence the requirements. Include asset analysis, harm analysis, threat analysis, and risk analysis.
2. *Security Risk Categorization Matrix*. Group potential malicious harm severities and potential frequencies of successful attacks and threat occurrences into meaningful categories. Use these to develop a security risk categorization matrix similar to a safety risk categorization matrix in figure 4.
3. *Security Importance Levels (SILs)*. Group security risk categorization matrix cells having similar levels of security risk into security risk categories. Use this categorization to formally define the endeavor's security importance levels (SILs). Note that the use of the analogous phrase "security *integrity* levels" is not recommended because integrity has a specialized meaning within the security discipline.
4. *Security Evidence Assurance Levels (SEALs)*. For each security SIL, define a corresponding security SEAL in terms of the best industry practices that should be followed to protect valuable assets from attack. These mandated practices will include both countermeasures (e.g., the use of firewalls and more secure coding standards) that are very general in that they provide widespread protection, as well as other more selective countermeasures (e.g., encryption) that will directly apply to smaller numbers of requirements. Although most of the measures mandated by these security SEALs will tend to be architecture-level countermeasures, the highest SEALs may also include the mandate to expend more resources during requirements engineering. For example, the highest SEALs may mandate the generation of pure security requirements associated with the non-security requirements as well as mandate the formal specification of such high SIL non-security requirements to ensure that they are complete and unambiguous.
5. *Requirements Risks*. Use the results of the preceding security analysis and the security risk categorization matrix to estimate the potential security risks associated with the non-security requirements, considering both individual requirements and where practical collections of related requirements. For each [group of] non-security requirements, assign an appropriate security SIL by considering all types of:
   - *Valuable assets* needing to be protected that are mentioned within the requirement.
   - *Harm* that can occur to these assets due to attacks, especially if the requirement is not implemented correctly, thereby creating an associated vulnerability. Consider the loss of confidentiality including both privacy and anonymity. Consider the loss of integrity of data, communication, software (e.g., via viruses and worms), hardware (e.g., via tampering), and people (e.g., via corruption and bribery). Consider the loss of availability of data access and services. Finally, consider any need to avoid repudiation of transactions.
   - *Security incidents* (attacks and probes) that could be used to attack the assets and cause malicious harm.

   When categorizing the non-security requirements by SIL, use any mention of users (e.g., as actors in use case models) and interface requirements to help identify requirements that will require derived identification, authentication, and authorization requirements.
6. *Requirements Implementation*. Use the SILs when tracing the requirements to architectural, design, and implementation components so that the appropriate level of process and best industry practices can be used to both properly implement the requirements and generate useful evidence for security certification purposes.
7. *Security Case*. Use the evidence resulting from applying the appropriate SEALs to provide useful arguments and evidence to build a *security case* that is similar to a safety case.

5. Related Work

Some of the similarities between safety and security have prompted others to recommend the use of safety techniques when performing security engineering.

Sacha Brostoff and M. Angela note that safety and security are perceived by many people as expensive activities that are only critical when they fail [2]. When accidents and successful attacks finally occur, the last individual in the chain of events leading up to the incident is often made a scapegoat when blame can be spread across multiple people and processes. They therefore recommend using the Generic Error Modeling System (GEMS) for analyzing these individual and organization sources of failures [8]. This work, however, does not address similarities between safety and security requirements, but rather the causal chains of events that lead to accidents and successful attacks.

Nancy Leveson and Mats Heimdahl also recognize many similarities between safety and security [3]. Based on these similarities, they recommend using the safety engineering techniques Intent Specifications [9] and Software Deviation Analysis (SDA) when performing security engineering. Intent specifications are a way of organizing project documentation, and level 1 (of 7) includes formally-specified requirements. SDA takes these formally specified requirements as an input and therefore is not relevant to categorizing security requirements. They do not address the security relevance of non-security requirements.

6. Conclusion

The contents of this position paper are heavily based on the strong similarity between the concepts and associated analyses underlying safety and security engineering. Safety engineering categorizes non-safety requirements using safety integrity levels (SILs), uses safety evidence assurance levels (SEALs) to enforce the additional measures needed to develop the more safety-critical parts of systems and to ensure the existence of the documentation needed to build safety cases and obtain safety certification. This paper suggests that security engineers should consider doing the same by:

- Using security importance levels (SILs) for categorizing non-security requirements in terms of their security relevance.
- Using security evidence assurance levels (SEALs) to enforce the additional measures needed to develop the more security-critical parts of systems and to ensure the existence of the documentation needed for security certification.
- Using security SILs and SEALs to build security cases.

Safety and security engineering are typically considered to be separate specialty engineering disciplines, and these disciplines are therefore usually not well integrated with mainstream engineering disciplines such as requirements engineering, architecting, design, implementation, and testing. Thus, safety and security policies are sometimes under emphasized during architecting because they do not adequately affect the requirements specifications. But safety engineers have increased safety engineering's influence early in the development cycle by using safety SILs and SEALs. It is possible that the use of security SILs and SEALs could similarly be a way to:

- Enable security engineers to prioritize the requirements so that they can concentrate their limited resources on the most important requirements and their implementations
- Better integrate security engineering into requirements engineering and architecting.
- Ensure that security is better addressed early in the development cycle before the architecture is largely completed and frozen.

7. References

[1] D.G. Firesmith, *Common Concepts Underlying Safety, Security, and Survivability*, Technical Note CMU/SEI-2003-TN-033, Software Engineering Institute, Pittsburgh, Pennsylvania, December 2003.

[2] S. Brostoff and M. Sasse, "Safe and Sound: a Safety-Critical Approach to Security," *New Security Paradigms Workshop'01*, 11-13 September 2001.

[3] N. Leveson and M. Heimdahl, "New Approaches to Critical-Systems Survivability: Position Paper," 1998 Information Survivability Workshop (ISW'98), IEEE, 28-30 October 1998.

[4] D.G. Firesmith: "Engineering Security Requirements", in *Journal of Object Technology*, vol. 2, no. 1, January-February 2003, pp. 53-68. http://www.jot.fm/issues/issue_2003_01/column6

[5] P. Bishop and R. Bloomfield, "A Methodology for Safety Case Development," Adelard, London, United Kingdom, 1999 www.adelard.co.uk/resources/papers/pdf/sss98web.pdf.

[6] C. Alberts and A. Dorofee, *Managing Information Security Risks*, Addison-Wesley, Boston, Massachusetts, 2003.

[7] C. Alberts and A. Dorofee, *Managing Information Risks: The Octave[sm] Approach*, Addison Wesley, 2003.

[8] J. Reason, *Human Error*, Cambridge University Press, Cambridge, UK, 1990.

[9] N. Leveson, "Intent Specifications: An Approach to Building Human-Centered Specifications," *IEEE Transactions on Software Engineering*, SE-26(1), January 2000.