A Taxonomy of Security-Related Requirements

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

Abstract

Safety and security are closely related subtypes of defensibility, another quality factor in a quality model. The close similarity between these two quality factors implies that a taxonomy of safety-related requirements is a good place to begin when developing an analogous taxonomy of security-related requirements. The resulting taxonomy consists of pure security requirements specifying minimum acceptable amounts of security, security-significant requirements which are non-security requirements with important security ramifications, security system requirements specifying requirements for security architectural components, and security constraints.

1. The Problem

Security engineering has historically emphasized the use of industry best practices (e.g., firewalls, encryption) as well as performing vulnerability analysis and security (e.g., penetration) testing of existing systems to ensure adequate security. Most books and articles on security do not provide much content with regard to security requirements, and what little that is published tends to emphasize the specification of ambiguous security goals or else focuses on architectural constraints. Rarely is either the required *amount* of a specific type of security specified or the security ramifications of non-security requirements addressed in security processes. Although security requirements may be mentioned, they are rarely defined and a clear taxonomy of the different kinds of security requirements is rarely if ever used.

This paper addresses the problems associated with a lack of a clear security taxonomy by identifying four different types of security-related requirements, providing them with clear definitions, and placing them within an organizing hierarchical taxonomy. This paper does this by recognizing the significant similarity between safety and security as sister subtypes of defensibility within a quality model and reusing the similar identifications, definitions, and taxonomy of safety requirements [1].

2. Similarities to Safety

2.1. Quality Model and Quality Requirements

A quality model is a concept from the quality and measurement communities, in which it is used to make the general term "quality" specific and useful. As illustrated in Figure 1, a quality model consists of a taxonomy of constituent quality factors, their component quality subfactors, and the associated measures that provide a means to quantify and thereby measure them. Thus, quality is primarily defined in terms of quality factors, which are attributes, properties, or characteristics of a work product that characterize important aspects of its overall quality. When a quality model is applied to define the quality of a specific system, system-specific quality criteria are developed which describe the system in terms of the quality factors and subfactors and thereby provide evidence for the existence of those quality factors and subfactors. For example, performance is a quality factor and throughput is one of its quality subfactors. Throughput can be measured in terms of operations completed per unit time. Thus, "the system updates the radar display every 50 milliseconds" is an example quality criterion using the measure "number of updates per millisecond." As illustrated in Figure 2, it is only a small step to add a minimum acceptable threshold to turn the performance criterion into the unambiguous and testable performance requirement: "The system shall update the radar display at least once every 50 milliseconds."

As illustrated in Figure 3, safety, security, and [military] survivability are all subtypes of defensibility, which is a subtype of dependability, which in turn is a type of quality factor [1][2]. As illustrated in Figure 4, safety and security share a great many quality subfactors because they are both subtypes of the same quality factor. These subfactors can be classified using multiple inheritance into the two inheritance trees for defensibility problem types and defensibility solution types.

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 [3][4]. Safety and security are subtypes of defensibility quality factor because they are both primarily concerned with the protection of *valuable assets* from *harm*, which is a significant negative consequence to the asset. These assets are people, property, services, and the environment that are significantly valuable to legitimate stakeholders and for which the system is responsible for protecting from harm.

As illustrated in Figure 5, 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.

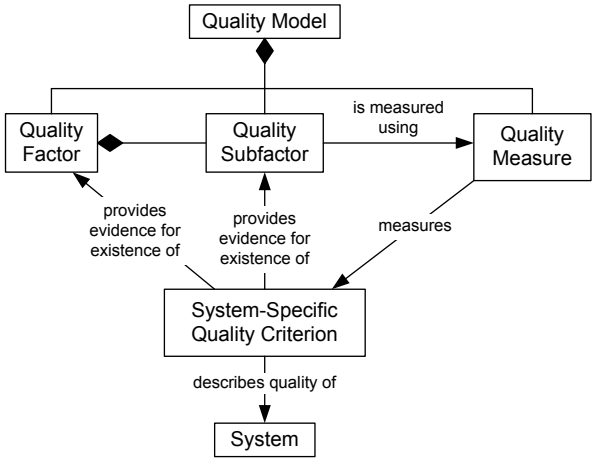

Figure 1: Quality Model

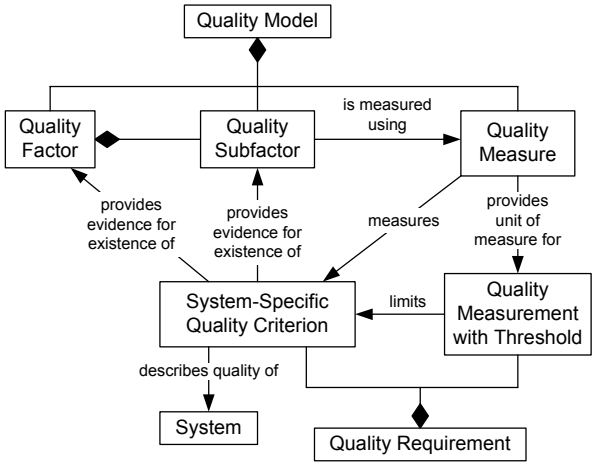

Figure 2: Quality Requirements

As illustrated in Figure 6, unauthorized harm to valuable assets occurs during *incidents*, which are any 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* or *scans* (i.e., preparations for attacks). Note that attacks can be classified multiple ways and that there are many different kinds of attacks. This leads to the existence of multiple security requirements mandating that the system properly handle such attacks.

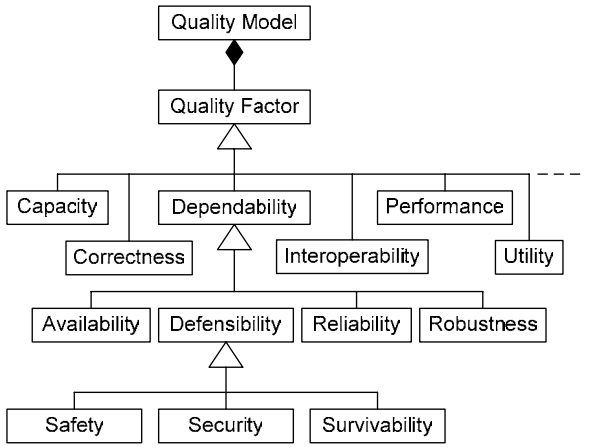

Figure 3: Safety and Security as Kinds of Defensibility

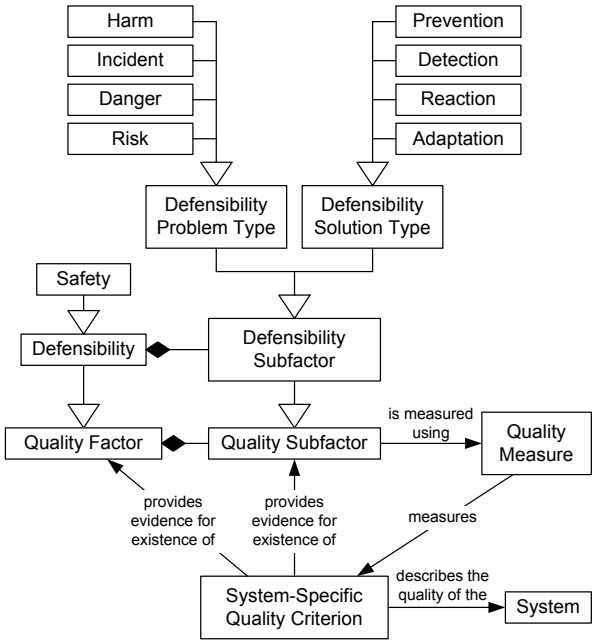

Figure 4: Defensibility Subfactors

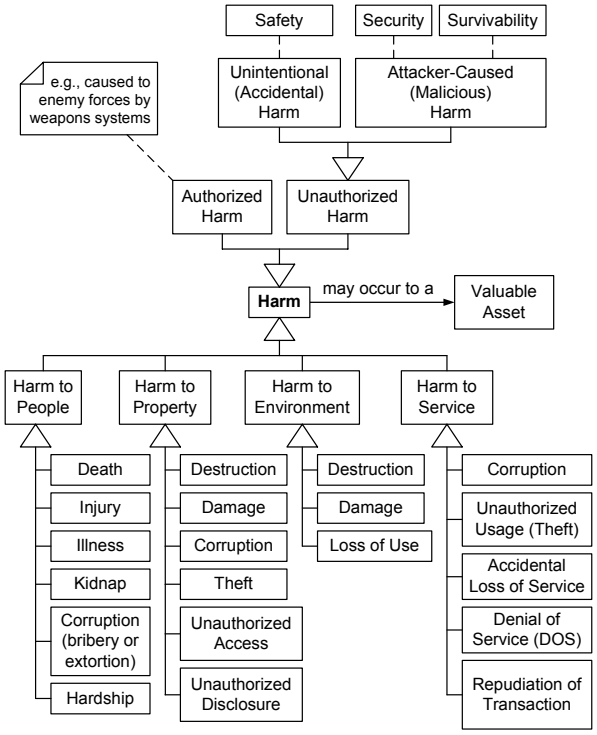

Figure 5: Accidental vs. Malicious Harm

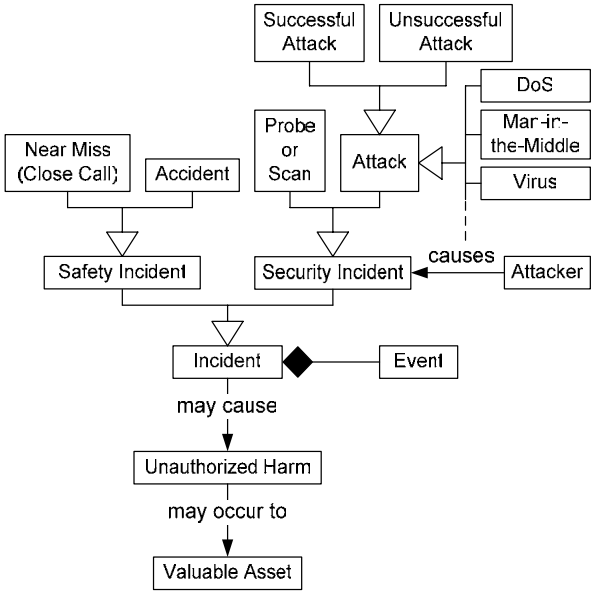

Figure 6: Safety and Security Incidents

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 7, dangers are classified into *hazards* and *threats*, whereby hazards can cause safety incidents and threats can cause security incidents. For example, a safety hazard for an automated people mover would be a moving train (condition 1) with open doors (condition 2).

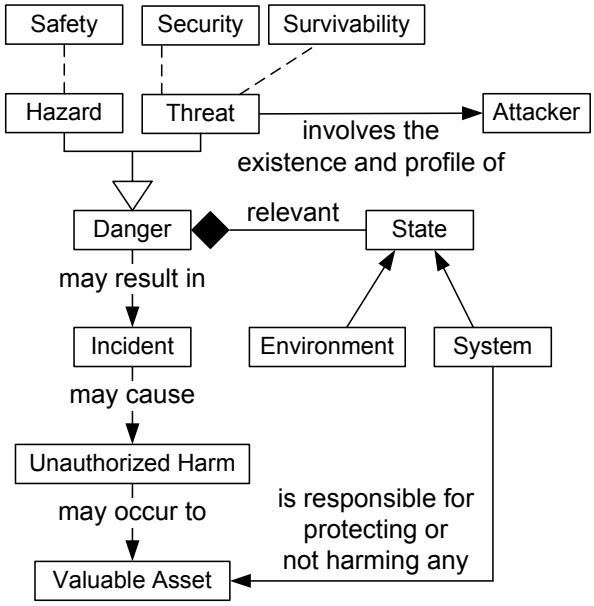

Figure 7: Safety Hazards vs. Security Threats

One reason that hazard or threat analysis is used to analyze dangers 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 or successful attack occurs. And as before, risks can be classified as either *safety risks* due to hazards or *security risks* due to threats.

2.2. Safety-Related Requirements Taxonomy

The great similarity between safety and security implies that it would be beneficial to first look at a previously published taxonomy of the different kinds of safety-related requirements [5] [6] before attempting to produce an analogous taxonomy of security-related requirements.

As illustrated in Figure 8, the taxonomy of safety-related requirements includes:

- **Safety requirements**, which specify a minimum acceptable amount of safety in terms of a pair of orthogonal defensibility subfactors. Such 'pure' safety requirements are specified in terms of a quality criterion involving these safety subfactors

and minimum acceptable thresholds on associated quality measures.

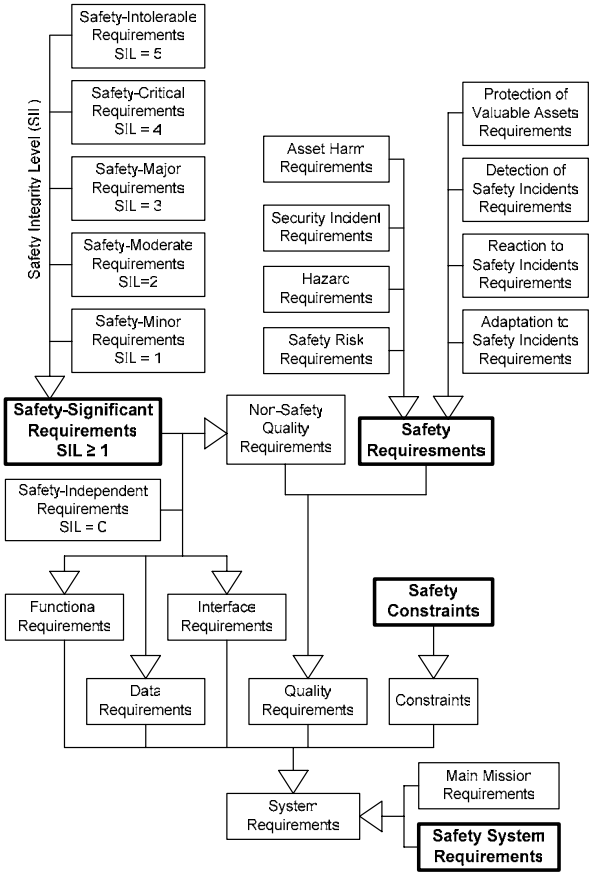

Figure 8: Taxonomy of Safety-Related Requirements

Because as illustrated in Figure 4, there are four defensibility problem types (i.e., harm, incident, danger, and risk) and four defensibility solution types (i.e., prevention, detection, reaction, and adaptation) that can be independently selected, there are 16 different kinds of associated safety requirements. An example of a harm prevention safety requirement is "The automated people mover (APM) system shall not cause a passenger injury requiring hospitalization more than an average of once every 10,000 passenger trips." An example of an incident detection safety requirement is "The APM system shall detect the collision of two vehicles with a relative velocity of more than 2 miles per hour at least 99.9% of the time." An example of a hazard reaction safety requirement is: "If the APM system detects the 'train movement with open doors' hazard, then the APM system shall within 1 second (1) notify the passengers that the train will stop, (2) notify the passengers in the affected cars to stay away from the open doors, (3) begin stopping at the maximum safe speed, and (4) notify the operator."

- **Safety-significant requirements**, which are non-safety requirements that have significant safety ramifications (e.g., safety-critical requirements). These are the requirements along the upper left side of Figure 8 that have a safety importance level (SIL) from 1 (Minor) through 5 (Intolerable). For example, the functional requirements to open doors, close doors, starting trains, and stopping trains all have significant safety ramifications and a SIL of 3 or 4 , whereas functional requirements to announce upcoming stations do not and have a SIL of 0.
- **Safety system requirements**, which are requirements for safety systems or subsystems such as the emergency core-coolant systems of nuclear power plants or the fire detection and suppression systems of airplanes.
- **Safety constraints**, which are safety-related architecture, design, coding, or testing constraints that are imposed as if they were normal requirements. An example safety architecture constraint is "The APM system architecture shall not include a component the failure of which can cause a SIL 4 requirement to be violated." An example safety implementation constraint is "The APM system software shall be programmed using a safe subset of C++."

3. Corresponding Security Taxonomy

This section of the paper presents a taxonomy of security requirements that is analogous to a previously published taxonomy of safety requirements. It is based largely on the similarity of security and safety as sister subfactors to the quality factor defensibility.

3.1. Security Requirements

If you consider security to be a quality attribute of the system, the pure security requirements are no different than any other quality requirement. All quality requirements have the same structure; they specify a minimum amount of the quality factor in terms of a quality criterion and one or more required thresholds on associated measures [6]. A quality criterion is a system-specific statement about the existence of a subfactor of the quality factor. Therefore, a security requirement should consist of a security criterion together with associated minimum required threshold(s) on appropriate measure(s). As a kind of dependency, security like safety has two sets of security subfactors:

- **Defensibility Problem Type**:
  - Malicious **Harm** to Valuable Asset
  - **Security Incidents** (e.g., attacks and probes)
  - **Security Threats** (security subtype of danger)
  - **Security Risks**, which are a combination of the frequency of attack or threat occurrence and the resulting harm
- **Defensibility Solution Type**:
  - **Prevention** of malicious harm, security incidents, security threats, and security risks
  - **Detection** of malicious harm, security incidents, security threats, and security risks (e.g., detection of an attack or the existence of a security hazard such as the existence of new viruses)
  - **Reaction** to detected malicious harm, security incidents, security threats, and existence of security risks (e.g., recording information about the incident, notifying security personnel, limitation and repair of damage, and returning the system to a secure state)
  - **Adaptation** of system to avoid or minimize the negative consequences of malicious harm, security incidents, security threats, and security risks

As illustrated in figure 9, the classification of malicious harm to valuable assets into denial of service (DoS), unauthorized disclosure, corruption, repudiation leads to the corresponding traditional primary types of security policy violations: availability, confidentiality (including privacy and anonymity), integrity, and nonrepudiation. Thus, security requirements related to preventing, detecting, reacting, and adapting to malicious harm can be further classified as availability, confidentiality, integrity, and nonrepudiation requirements. Security requirements associated with identification, authentication, and authorization can now be viewed as derived requirements because only identified and authenticated people, applications, and systems are authorized to:

- Use services (availability).
- See private or classified data or be anonymous (confidentiality).
- Create, modify, and delete assets (integrity).

The following are examples of pure security requirements:

- **Prevention of Harm**: "The system shall prevent the corruption of confidential data by at least 99% of the attacks that last no longer than 1 hour and are made by attackers with profile X."
- **Detection of Attack**: "The system shall detect the presence of at least 99.9% of denial of service attacks."
- **Reaction to Harm**: "Upon detection of corrupted data, the system shall notify the security engineer within 5 seconds at least 99.99% of the time."
- **Adaptation to Attacks**: "The system shall use virus definitions and a virus scan engine that have been updated within the previous 24 hours, if such an update exists."

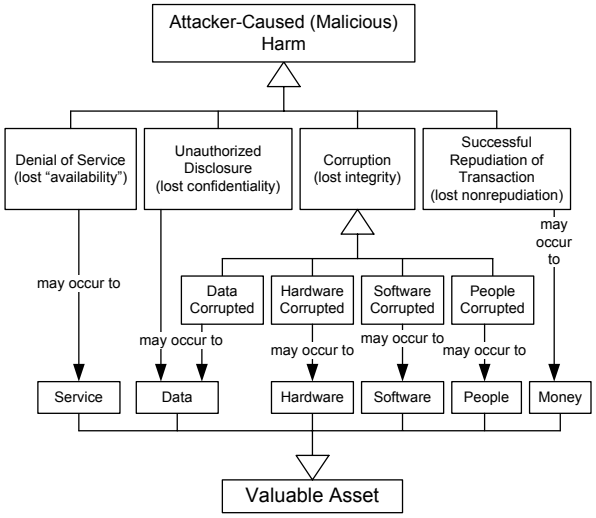

Figure 9: Harm to Valuable Assets (Security)

Pure security requirements are not traditionally specified for several reasons. First of all, requirements and security engineers have typically not been trained in how to identify, analyze, and specify such requirements. Good security requirements are difficult to specify because it is difficult to determine quantitative thresholds that are acceptable to both customer and developer representatives. Such unambiguous requirements must be verifiable in practice as well as theory. Thus, appropriate types and levels of verification techniques need to be determined to be sufficient to meet the thresholds of the corresponding security requirements. These verification techniques can include inspection, demonstration, and security testing such as penetration testing.

3.2. Security-Significant 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 is definitely worth considering if something similar can be done for security [7].

As illustrated in Figure 10, the security team can generate a security importance level (SIL) matrix based on a security analysis of credible harm to valuable assets, potential security incidents, associated threats, and security risks. They can then use the SILs to categorize non-security requirements in a manner similarly to the using of safety integrity levels to categorize non-safety requirements. The resulting categories are useful for assigning security evidence assurance levels (SEALs) for certification and accreditation purposes. Note that ALARP in the following matrix means "As Low as is Reasonably Practical" and is used to signify that the corresponding SEAL for the implementing components should mandate the use of sufficient tools and techniques to make their contribution to security risk as low as is reasonably practical.

Security Risks / Security Importance Levels (SILs)					
	Frequency of Attack / Threat 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 10: Security Importance Levels (SILs)

The equivalence of safety SEALs and security SEALS is not without precedence in industry. It has for example been informally addressed on one large aerospace program that I have been associated with in a security policy that requires that all software with major security ramifications (i.e., the equivalent of "security-critical" software) be given a high safety SEAL level and developed to the same strict standards. The identification of security-significant requirements with analogous SIL and SEAL levels merely formalizes this relationship and enables security to have its own appropriate levels and more importantly its own appropriate mandated techniques (e.g., required amounts of threat analysis and penetration testing) for security certification.

3.3. Security-System Requirements

Many safety-critical systems have associated subsystems that exist purely for safety's-sake. The emergency core-coolant system of a nuclear power-plan is the canonical example, but an airplane's fire detection and suppression system is another common example.

Similarly, systems with significant security needs also often have subsystems or other major architectural components that exist only to ensure adequate security. Examples include:

- **Access control subsystems**, which are responsible for providing identification, authentication, and authorization.
- **Encryption/decryption subsystems**, which are responsible for ensuring required levels of confidentiality, non-repudiation, and the integrity of messages.
- **Antivirus packages**, which are responsible for ensuring required levels of the integrity of software.

Assuming that they are derived subsystem-level requirements, any requirements associated with such security subsystems are clearly security-related and are **security-system requirements**.

3.4. Security Constraints

A **constraint** is typically an engineering decision that is treated during requirements engineering as if it were a requirement even though it would ordinarily be made during architecture development, design, implementation, integration, or testing. Thus, a **security constraint** is typically a mandated security policy or countermeasure such as mandating specific approaches for ensuring a security requirement or requiring specific configuration levels when using a countermeasure. Whereas security constraints are in many ways no different than other types of constraints, because they are specified for security reasons, they are subject to security certification and accreditation like other security-related requirements. Some security constraints are required by a relevant security regulation, standard, or law. In fact, some security constraints merely mandate compliance with such a regulation, standard, or law, and therefore act as a way to group the numerous constraints included in the regulation, standard, or law.

The following are typical examples of security constraints:

- **Identification and Authentication.** "The application shall require users to use a password that is at least 6 characters long and modify their

passwords at least once a month." It is well known that user identifiers (IDs) and passwords are typically the least secure methods for identification and authentication. Requirements should not unnecessarily constrain the architects from using more secure countermeasures such as biometrics and smart cards. Yet when teaching use case modeling, any mention of identification and authentication almost always includes requiring the user to enter his or her password, thereby preventing the architect from considering the incorporation of a potentially more effective thumbprint reader [8][9][10].

- **Encryption.** "The application shall use a COTS public-key encryption and decryption package to ensure that confidential data remains secure." Although using such commercially-available encryption product is typically the best choice for ensuring the confidentiality of confidential data, other security policies and countermeasures exist and should be considered. For example:
  - Too often, applications unnecessarily collect and store confidential data.
  - Too often, applications do not delete confidential data when it is no longer needed.
  - Steganography can be used to hide messages within pictures.
- **Integrity and Nonrepudiation**. "The application shall use the MD5 128-bit hash code to ensure the detection of corrupted messages." Although hash codes are commonly used to create digital signatures that enable the message sender to be identified as well as any corruption of the message, there are multiple types of hash codes, 128-bits may or may not provide an appropriate amount of security given the value of the data to be protected as well as any performance requirements, and the constraint seems to imply that the developers must develop their own security software rather than using a commercially-available package.

3.5. Security-Requirements Taxonomy

Figure 11 illustrates the taxonomy of security-related requirements resulting from the combination of the different kinds of security-related requirements in the previous 4 sections. The two sets of 4 subtypes of security requirements in the upper right part of the figure combine to produce the 16 different kinds of pure security requirements. The 5 different kinds of non-security requirements along the upper left side of the figure with a security importance level (SIL) from 1 through 5 form the security-significant requirements. The security system requirements in the lower right part of the figure are those system requirements allocated to security subsystems. Finally, the security constraints are shown on the right side of the figure as a subtype of ordinary constraints.

Note that this taxonomy is quite different than the Volere Requirements Taxonomy [11], which follows a more traditional and simpler view of security requirements.

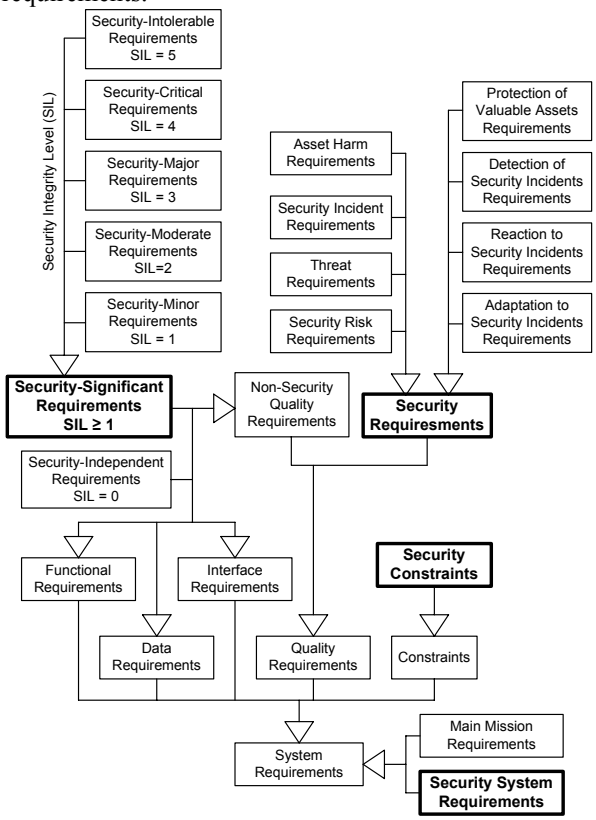

Figure 11: Taxonomy of Security-Related Requirements

4. Uses of the Taxonomy

Using a taxonomy of security-related requirements to elicit, organize, and manage requirements provides the following benefits to requirements engineers and security engineers:

- **Training.** Engineers can use the taxonomy to learn about the different kinds of security-related requirements including their names, definitions, similarities, and differences. Taxonomies may also have associated ancillary information such as good and even bad examples with associated rationales for the problems with the bad examples.
- **Improved Communications.** Requirements and security engineers can use the common terms and concepts of the taxonomy to improve their

communication, and thereby improving their collaboration.

- **Requirements Completeness.** Engineers can check the requirements specifications against the taxonomy to ensure that each kind of security-related requirement is present. Engineers can also check the security policies to see if they contain security requirements and constraints in addition to policies.
- **Adequate Emphasis.** In current practice, security constraints and security system requirements have been emphasized over security requirements and the analysis of security-significant requirements (i.e., those with a security importance level greater than 0). The use of a taxonomy identifying these four types of security-related requirements helps to ensure that each type is adequately emphasized.
- **Table of Contents.** The taxonomy can be used to structure the part of the requirements specification table of contents covering the security-related requirements.
- **Checklist.** The taxonomy can be used as a starting point to create a checklist for evaluating the security-related requirements in the requirements repository, requirements specification, or security policies.
- **SILs and Cost.** The use of security SILs will help the security and safety engineers to estimate the costs of specifying the associated requirements because a requirement with a high security SIL requires a corresponding high security SEAL for its implementing components, which very greatly increases their development, testing, and certification costs.
- **Architecture and SEALs.** Knowledge of security SEALs will help the architect develop the security architecture, especially with regard to software decomposition and allocation to hardware. The high costs of implementing architectural components with high SEAL levels will lead to architectures exhibiting high modularity with regard to SEALs. However, architects should be aware that the high costs of high SEAL components may cause strong management pressures to lower SEAL levels below that which the associated SILs imply.
- **Requirements Tracing.** The use of security SILs and SEALs provides a way of tracing security-related requirements via SILs through the architecture via SEALs to the development process and security testing, which supports the analysis needed during change management.

5. Opposition to Security Taxonomy

Not everyone will agree with the security taxonomy presented in this paper. Some members of the security community feel that their discipline is so different from safety that relatively little can be learned from safety engineering. They are very uncomfortable with security requirements such those described and listed in section 3.1 because they believe that either development organizations will not sign up to implement such requirements or else such requirements are inherently not verifiable. They argue that it is impossible to prove that a system is absolutely secure so mandating best industry practices is the best that can be done, whereby best practices include use of policies, firewalls, coding standards, and penetration testing. Finally, they are concerned because security involves a constant arms race with attackers as technology and techniques improve. Thus, even if a system meets its security requirements during acceptance testing, it may not meet those same requirements upon delivery if an effective new exploit is published the following day. These concerns can and should be addressed:

- **Similarities.** Although legitimate differences exist between safety and security engineering, the many well documented similarities [1][2] justify at least considering whether the lessons learned from the safety discipline are worth exploring.
  Unfortunately, security engineering is such a large and complex discipline that requires so much concentration and study to master that little time is typically left for learning other disciplines. Their inherent similarity, ignorance of related disciplines, and occasional instances of the not-invented-here (NIH) syndrome may partially explain why safety and security have each tended to grow into the other. Thus, security books sometimes include accidental harm as well as malicious harm within the security domain. These factors may have also partially prevented or delayed reuse, resulting in unnecessary differences between the original approach in one domain and its associated reinvented wheels in the other.
- **Acceptability.** It is certainly easier for development organizations to use industry best practices instead of (as opposed to in addition to) security-related requirements. But the acquisition organizations really do have goals (i.e., minimum levels of security) and best industry practices are only an indirect way of trying to achieve them. Converting ambiguous goals into unambiguous requirements has been shown to be an effective approach in all other disciplines, so why not

security engineering? Development organizations will learn to accept security requirements when they understand them and customers demand such requirements from the suppliers of their systems.

- **Verifiability.** Well written security requirements are verifiable because they are both unambiguous and feasible. Such requirements do not specify that the system be absolutely secure, but instead specify that it be secure in a specific way to a certain degree. Detractors should note that the testing processes for all kinds of requirements of non-trivial applications are always incomplete because it is impractical or impossible to perform exhaustive testing. This is why testers develop an incomplete set of test cases that nevertheless meets test coverage and completeness criteria. The system is said to fulfill its requirements if it passes its acceptance tests. Consider the requirement: "The system shall prevent the corruption of confidential data by at least 99% of the attacks that last no longer than 1 hour and are made by attackers with profile X." This requirement has an associate operational testing approach. If for example testing is to provide 95% confidence that requirements have been fulfilled, then one or more security testers should perform sufficient penetration tests trying to corrupt representative confidential data using the techniques and tools of attackers with the specified profile to provide the required confidence that 99% of such attacks will fail. After this testing, the system will have been determined to have either passed or failed to fulfill this requirement. In this sense, a security requirement is no different than any other type of requirement. No requirement is ever proved to be 100% fulfilled; just fulfilled to the extent shown by appropriate testing.
- **Security Arms Race.** This is the most interesting argument against specifying security requirements because it addresses a fundamentally real problem. After acceptance testing, external conditions may (and probably will) change, causing the system to no longer meet its specified requirements. The proper response to this problem is to then update the system so that it once again meets its requirements. This is no different from a weapons system that is required to carry out its mission in spite of the existence of specific types of military (as opposed to security) threats. A military arms race may require the weapons system to be upgraded because it can no longer fulfill its required mission because it no longer meets its military survivability requirements. Once again, if the system passes mutually agreed upon acceptance testing, the system can be delivered and must be accepted. If new or changed security threats or military threats subsequently change the environment, the system needs to be upgraded so that it once again fulfills its unchanged requirements even if a new development contract or contract modification or a maintenance contract is needed for the upgrade.

The bottom line is that the stakeholders have security goals and requirements that they need the system to meet. Using security best industry practices is appropriate, but insufficient to ensure that these needs are met. One task of security requirements is to provide that assurance.

6. Conclusions

Based on the similarity between safety and security as subfactors of the quality factor defensibility, this paper presents a taxonomy of security requirements that corresponds to an analogous taxonomy published for safety requirements. Thus, security requirements can be decomposed into pure security requirements, security-significant requirements, security system requirements, and security constraints. Pure security requirements specify minimum thresholds of various subfactors of security; they can be decomposed into sixteen subtypes (4 defensibility problem types multiplied by 4 defensibility solution types) of pure security requirements that specify minimum acceptable levels of the prevention of, detection of, reaction to, and adaptation to malicious harm, security incidents, threats, and security risks. Security-significant requirements are derived by using harm, incident, threat, and risk analysis to categorize non-security requirements in terms of their relative security risks. Security system requirements are those requirements associated with security subsystems. And security constraints are engineering decisions that are treated during requirements engineering as if they were security requirements even though they would ordinarily be made during architecture development, design, implementation, integration, or testing.

Thus, there exist multiple types of security-related requirements, and the requirements team should consider all of them. It is inappropriate for the requirements to consist only of security constraints mandating traditional and common security countermeasures because:

- Different systems must protect different valuable assets within different environments against potentially different types of attackers with different profiles.

- Security technologies and the corresponding security products that implement them are rapidly evolving.
- Requirements engineers should not unnecessarily constrain the choices of security engineers who have the experience and expertise to make better architectural choices when selecting appropriate countermeasures.

On the other hand, requirements engineers should collaborate with security engineers to ensure that the resulting requirements adequately cover all types within the taxonomy of security-related requirements and have the proper characteristics of good requirements. By using such a taxonomy to produce a complete set of security-related requirements, it is hoped that the architecture team will incorporate the correct amount of the right types of security countermeasures, without having their decisions limited by unnecessary constraints.

While this paper has provided strong reasons to consider using the taxonomy it provides, several major questions remain:

- Because the key difference between safety and security is whether harm is accidental or malicious, is the active nature of attacks sufficiently different from the passive nature of accidents to override the similarities upon which the taxonomy is based? While the author does not think so and has addressed how this affects the fact that systems that pass security requirements one day may not pass the same requirements the next day as attacks improve. However, others disagree and the issue requires significantly more research and experience before being decided.
- To what extent will using this taxonomy help requirements and security engineers communicate and produce unambiguous verifiable security-related requirements?
- Is the taxonomy really complete? To what extent will using this taxonomy help ensure that important types of security-related requirements are no longer overlooked during requirements engineering?
- How will the practical limit on the possible amount of security testing limit the corresponding probabilities of attacks defeated that can be specified in security requirements? A statistical analysis should be performed to determine the minimum number of security test cases required to achieve a specific level of confidence that a specific percentage of attacks will be thwarted assuming that the tests are representative of the actual attacks of a given attacker profile.
- Unlike with hardware, it is very difficult to estimate the probability of a safety hazard or associated accident when software is involved. It will be at least as difficult to estimate the probability of a security risk or associated successful attack. Safety hazards and accidents should be rare, but security threats and attacks (if hopefully not successful) will probably be all too common. How will this affect the development of a security risk/SIL matrix? Will it be adequate to merely have fewer, broader columns and fewer associated security SILs?
- What are the optimum number of security SIL and SEAL levels? How should they be defined? What are the appropriate techniques to use at each SEAL level to ensure adequate security is achieved? To what extent will this evidence be appropriate, acceptable, and useful during certification?
- Will the use of the same acronyms (SIL and SEAL) for both safety and security improve the acceptability of security SILs and SEALs, or will it instead merely lead to confusion, especially if a single requirement has different safety and security SILs (almost certainly) and a single component has different safety and security SEALs (quite possible)? Is it adequate to merely prefix the acronym with the appropriate term if the scope is not obvious from context?
- Safety SILs and SEALs have historically been assigned only to safety-significant requirements and their associated implementing components, and not to safety requirements, safety system requirements, and safety constraints. For the sake of consistency and ensuring adequate use of scarce development resources, would it not be better to assign security SILs and SEALs to all security-related requirements and their components?
- Will this taxonomy help requirements and security engineers recognize security constraints as such and enable them to minimize the number of unnecessary constraints?
- Finally, what about survivability, which protects valuable military assets from harm resulting from attack by enemy military forces? As illustrated in Figure 3, safety, security, and survivability are all subtypes of the quality factor defensibility and therefore share similar properties. Should an analogous taxonomy for survivability-related requirements exist?

5. 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] A. Avižienis, J.C. Laprie, and B. Randell, "Dependability and its Threats: A Taxonomy," Proceedings of the Building of Information Society: Proceedings of IFIP 18th World Computer Congress: Topical Sessions. 22-27 August 2004, Toulouse, France, R. Jacquart (ed), pp. 91-120, Boston, London: Kluwer, 2004.

[3] D. Cooper and D. Jackson, *SafSec: Integration of Safety and Security Certification, SafSec Methodology: Standard*, Issue 2.6, S.P1199.50.2, Praxis Critical Systems 13th May 2004 <http://www.safsec.com/safsec_files/resources/50_2_SafSec_Method_Standard_2.6.pdf.>

[4] D. Cooper and D. Jackson, *SafSec: Integration of Safety and Security Certification, SafSec Methodology: Guidance Material*, Issue 2.6, S.P1199.50.3, Praxis Critical Systems 13th May 2004 <http://www.safsec.com/safsec_ files/resources/50_2_SafSec_Method_Standard_2.6.pdf.>

[5] D.G. Firesmith, "A Taxonomy of Safety-Related Requirements," *Requirements Engineering'2004 (RE'04) Requirements for High Assurance Systems (RHAS) Workshop*, IEEE Computer Society, Washington, D.C., 6 September 2004.

[6] D.G. Firesmith, "Engineering Safety Requirements, Safety Constraints, and Safety-Critical Requirements," *Journal of Object Technology (JOT)*, 3(3), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, pp. 27-42, March/April 2004.

[7] D.G. Firesmith, "Analyzing the Security Significance of System Requirements," *Requirements Engineering'2005 (RE'05) Symposium on Requirements Engineering for Information Security (SREIS)*, IEEE Computer Society, Washington, D.C., September 2005.

[8] A. Cockburn, *Mastering Writing Effective Use Cases*, Addison-Wesley, 2001.

[9] D. Kulak and E. Guiney, Use Cases: Requirements in Context, Addison-Wesley, 2000.

[10] G. Schneider and J. Winters, *Applying Use Cases: A Practical Guide*, Addison-Wesley, 1998.

[11] Suzanne and James Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999.