SECURITY THREAT MODELING AND ANALYSIS: A GOAL-ORIENTED APPROACH

Ebenezer A. Oladimeji
Architecture and eServices, IT
Verizon Communications
Irving, Texas 75038, USA.
email: ebenezer.oladimeji@verizon.com

Sam Supakkul
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas 75083, USA.
email: ssupakkul@ieee.org

Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
Richardson, Texas 75083, USA.
email: chung@utdallas.edu

ABSTRACT

Threat modeling provides a good foundation for the specification of security requirements during application development. When applied during the early phases of software development, threat modeling empowers developers in several ways. These range from verifying application architecture, identifying and evaluating threats, designing countermeasures, to penetration testing based on a threat model. There is however paucity of established techniques and tools for threat modeling and analysis. This paper proposes a goal-oriented approach to security threat modeling and analysis by using visual model elements to explicitly capture threat-related concepts. We introduce the notions of negative softgoals for representing threats and inverse contributions for evaluating design alternatives during analysis, while adapting the formal semantics of the NFR Framework. An analysis procedure is also provided to guide context-sensitive selection of countermeasures. The significance of this approach derives from the strength of the underlining analysis framework. We illustrate this approach by modeling and analyzing the security threats of an online banking system.

KEY WORDS
software security, threat modeling, security requirements engineering, negative softgoal, inverse contributions

1 Introduction

Software security has continued to attract significant attention as society increasingly relies on computer-based systems. The need for designing security into software applications rather than retrofitting it as an afterthought has been well discussed [6, 22, 17]. In a study by the National Institute of Standards and Technology (NIST), it was reported that software systems faulty in security and reliability cost the US economy $59.5 billion annually in breakdowns and repairs [21]. In a related study, the Naval Research Laboratory did an analysis of some 50 security flaws and reported that almost half of them were caused by flaws in the requirements or specifications [13]. These studies underscored the need for incorporating security engineering into mainstream requirements engineering in order to bridge the gap that has hitherto existed between the two fields. Security threat modeling has been identified as a significant part of this endeavor [10, 26].

Security threat modeling (or simply *threat modeling*) is a formal process of identifying, documenting and mitigating security threats to a software system. It enables development teams to understand a system's threat profile by examining the application through the eyes of a potential adversary, and helps to determine the highest-level security risks posed to the system [10]. This process results in a *threat model* that describes the potential attacks on the system; this can be used to understand how attacks can manifest themselves and to evaluate critical decisions that will affect the security posture of the system [26]. It can also form the basis for system penetration testing as the system evolves [27].

By way of definition, a *threat* is simply a potential violation of the security of a system - an event that may have some negative impact [3]. *Vulnerabilities* are actual security weaknesses or flaws that make a system susceptible to an attack. An *attack* is an exploitation of a vulnerability to realize a threat. *Countermeasures* are defensive architectural mechanisms used for mitigating system vulnerabilities. The threat modeling process usually involves identifying information resources to be protected, identifying the entry or access points to these assets, analyzing the threats, evaluating associated risks, and developing mitigating strategies [26]. Ideally, a threat model should be developed during the earliest stages of system development, and then as the application evolves and requirements are better defined, the threat model can be updated as needed.

However, research in security threat modeling has yet to mature as there is paucity of established techniques and tools to aid the threat modeling and formal analysis process. While the importance of starting threat modeling during the requirements analysis phase has been well discussed in the literatures, existing modeling notations such as data-flow diagram [28] and attack trees [23, 19], are largely focused on the design and development phases. Moreover, existing work do not integrate threat modeling notations with a formal threat analysis procedure to aid decision making during security requirements analysis. In this paper, we propose a goal-oriented approach for explicitly modeling and analyzing security threats during requirements analysis. We introduce the notions of negative softgoals for representing threats and inverse contributions for evaluating design alter-

natives during analysis, while adapting the formal semantics of the NFR Framework [5]. An analysis procedure is also provided to guide the context-sensitive selection of countermeasures. We believe that this approach can make the problem of threat modeling more tractable and provide practitioners with a more pragmatic solution. An example of the threat modeling and analysis process for a simplified *Online Banking System* is used throughout the paper to illustrate our major concepts. In this application, retail banking customers are provided with a web based interface for managing their accounts (checking, savings, money-market, etc.), enabling them to perform such account services operations as viewing transactions, bill-pay, check ordering, and funds transfers.

The remainder of the paper is organized as follows: Section 2 provides a an overview of the underlying frameworks. A description of the notions of N-softgoals and inverse contribution is provided in Section 3. Section 4 describes the proposed formal process for threat modeling and analysis. Section 5 considers other related research and discusses the significance and limitation of our proposal, while Section 6 concludes the paper and outlines future directions.

2 Goal Oriented Frameworks

Goal-orientation has been shown to play significant role in requirements engineering with the use of goals for eliciting, specifying, analyzing, and documenting software requirements at large [2, 29]. Two complementary formal frameworks have emerged for goal-oriented requirements engineering namely, the KAOS [7] and the NFR Framework [5]. While KAOS is focused on modeling functional requirements as goals to be satisfied, the NFR Framework is more concerned with modeling quality attributes as softgoals and the qualitative assessments of design alternatives [30].

We adopt the NFR framework because of its expressiveness in representing and analyzing non-functional requirements (NFRs) such as reliability, performance, security, usability, etc, as well as its strong formal semantics which enriches the threat analysis process. In this framework, NFRs are represented as *softgoals* to be *satisficed*. Softgoals are considered satisficed when there is sufficient positive and little negative evidence for achieving them, or *denied* otherwise. To determine satisficeability, *operationalizing softgoals* representing design decisions for realizing the NFRs are identified and analyzed. *Contributions* of the offspring softgoals to their parents softgoals are evaluated and tradeoffs are made with rationales recorded. The entire process is recorded in a *softgoal interdependency graph (SIG)*. The selected design decisions are then used as the basis for architectural design. Figure 1 shows an SIG for *Security* softgoal for the *Online Banking System*. The light clouds represent NFR softgoals. These and other kinds of softgoals are labeled by nomenclature of the form *Type*[*Topic*] where *Type* is a descriptor (e.g. security, performance) and *Topic* is the context or scope (e.g. Account) of the softgoal.

NFR softgoals may be refined, typically by *type* or *topic*, one at a time. Refinements can also be done using AND-decomposition (denoted by a single arc) or OR-Decomposition (denoted by a double arc). For example, in Figure 1, the high level security softgoal *Security[Online Banking System]* is refined by *topic* into a more specific softgoal *Security[Account]* indicating that customers' accounts are the major information asset to be protected by the system. This is in turn refined by *type* using the AND-decomposition *method* into *Integrity[Account]*, *Confidentiality[Account]* and *Availability[Account]* softgoals. Other refinements can be done using the *method catalogs* described in [4] including operationalization of the more specific softgoals. These *operationalizing softgoals* are shown with dark clouds. For example, the *Confidentiality[Account]* softgoal in Figure 1 is operationalized by the *Role-BasedAccessControl[Account]* softgoal which is further refined into more specific security mechanisms.

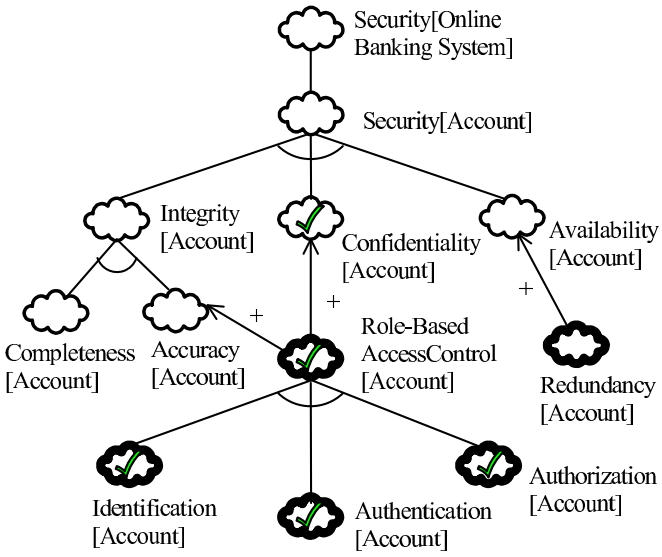

Figure 1. A Softgoal Interdependency Graph (*SIG*) for Secure Online Banking System

The arrows from dark clouds to light clouds indicate *contributions* (i.e. the degree to which the operationalizing softgoal contributes to *satisficing* or *denying* the NFR softgoals). Implicit interactions among softgoals in the forms of conflicts or harmony are modeled as correlations and denoted by dotted-arrows. The notion of "*satisficing*" of security softgoals (denoted by the *green tick* icon) reflect that security risks can only be limited in magnitude, reduced in likelihood, or made detectable, *but not* entirely removed. Denying a softgoal is denoted by *red X* icon. The degree of the contributions can be indicated as highly positive (++ symbol), somewhat positive (+ symbol), highly negative (– symbol), and somewhat negative (- symbol). Once sufficient level of refinements of the operationalizing softgoals is reached, an evaluation procedure (labeling algorithm) described in [5] is used to *satisfice* or *deny* the softgoals from the leaves upward the SIG. Figure 1 shows how satisficing the leaf softgoals *Identification[Account]*, *Authorization[Account]* and *Authentication[Account]* evaluates upward the SIG to satisfice the *Confidentiality[Account]* softgoal.

3 N-Softgoals and Inverse Contributions

Figure 1 shows a simple usage of the NFR Framework to represent and analyze security requirements for our *Online Banking System*. To adapt the framework for threat modeling, we introduce the notions of negative-softgoal (or for brevity, *N-softgoal*) and *inverse contribution* to provide visual support for explicit representation and analysis of security threats during the requirements engineering process.

An *N-softgoal* is a goal that a potential adversary is perceived to have against a system. Security threats and their associated vulnerabilities are modeled as N-softgoals, and denoted graphically with a *grayed-cloud* icon as shown in Figure 2. The *type* of an N-softgoal represents its descriptor (such as Denial-of-Service or Disclosure) while its *topic* represents the class of information assets against which the threat is targeted or the scope of the threat. For example the N-softgoal *UnauthorizedDisclosure[Account]* represents the threat that a potential attacker of the online banking system may gain unauthorized access to account information managed by the system. These N-softgoals can be captured at varying levels of abstraction and can be iteratively refined on *type* or *topic* using AND/OR decomposition and other refinement *methods*, as the requirements engineer may desire.

An *inverse contribution* of an offspring softgoal to its parent softgoal represents a relation in which the softgoal effectively negates its parent softgoal. This is denoted graphically with an "*inv*" icon as shown in Figure 2. It follows that if the offspring softgoal is *satisficed*, the parent softgoal is *deniable* (or *denied* when there are no contributions to the contrary from other offspring softgoals). Similarly if the offspring softgoal is *denied*, the parent softgoal is *satisficeable* (or *satisficed* when there are no contributions to the contrary from other offspring softgoals). This notion is similar to the *Break* contribution type described in [5] but is more appropriate for modeling security threats, given the negative reasoning logic involved. Intuitively, we use inverse contributions to relate N-softgoals (of potential adversary) to the high level security softgoals (of system stakeholders), and also to relate operationalizing softgoals (representing countermeasures) to the N-softgoals. For example, in Figure 2(b), the threats denoted by the N-softgoals *UnauthorizedDisclosure[Account]* and *Spoofing[Account]* have inverse contributions to the security softgoal *Confidentiality[Account]*, while the operationalizing softgoals *RoleBasedAccessControl[Account]* and *PacketFiltering[Account]* respectively have negative contributions to these N-softgoals.

More formally, let $Softgoals$ and $Contributions$ respectively denote the set of all softgoals and contributions. Also let $Satisficed$ be a predicate which is true of *satisficed* goals or contributions and false of others. Similarly, let $Denied$ be a predicate which is true of goals and contributions that have been shown unsatisficeable. If

$$Propositions = Softgoals \cup Contributions$$

denotes the set of all propositions over the problem space,

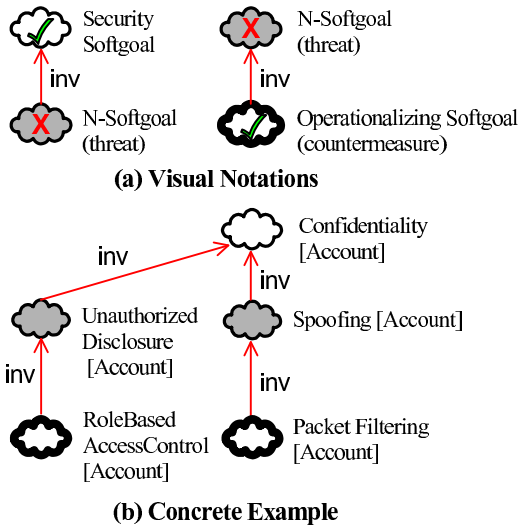

Figure 2. N-Softgoal and Inverse Contributions

then $Satisficed$ and $Denied$ are predicates taking a proposition as argument. Sometimes a proposition will be found to be *satisficeable* due to one refinement, and *deniable* due to another. For instance, the accuracy softgoal for a set of data might be satisficeable thanks to a validation procedure adopted for all such data, but at the same time deniable because of a user interface that permits non-discretionary access to this information. To deal with such conflicting cases, we distinguish between a proposition being satisficed or denied, on the one hand, and a proposition being potentially satisficeable or deniable on the other. Accordingly, we define two more predicates, $Satisficeable$ and $Deniable$ to deal with such cases. Then, for any parent softgoal S_p and its offspring softgoal S_o, the inverse contribution inv is defined as:

$inv : Propositions \times Propositions$ such that
$Satisficed(S_o) \wedge Satisficed(inv(S_p, S_o))$
$\Rightarrow Deniable(S_p)$
and
$Denied(S_o) \wedge Satisficed(inv(S_p, S_o))$
$\Rightarrow Satisficeable(S_p)$

4 The Threat Modeling and Analysis Process

Our proposed modeling and analysis process for security threats is depicted in the activity diagram of Figure 3. The process includes four high-level steps which encompass defining what security means for the system, eliciting threats, analyzing threats and their associated risks, and evaluating how countermeasures lead to the achievements of security objectives. The entire process is documented in a *threat-SIG* that forms the threat model to be used throughout the development life cycle. The metamodel defining the relationships between the conceptual model elements is shown with the UML class diagram of Figure 4. Next we elaborate on each step, using the *Online Banking System* for illustrations.

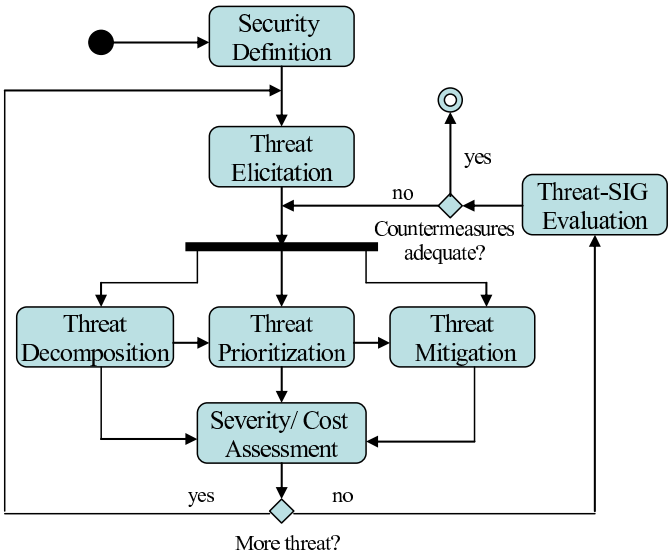

Figure 3. Threat Modeling and Analysis Process

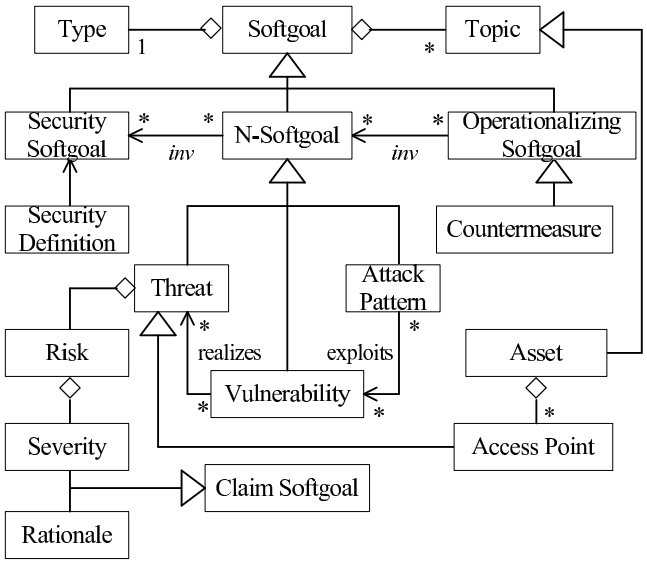

Figure 4. Threat Modeling and Analysis Metamodel

4.1 Security Definition

Security means different things to different applications depending on the nature, environment, and scope of the systems. For example, security of a *Student Information System* may be defined in terms of authorized access to student's information such as social security number. For our *Online Banking System* however, security may include availability of the system for 24-hour online customer service, as well as auditability of online transactions to avoid repudiation. Therefore, the first step in the security threat modeling process is to explicitly define what security (or *secure*-ness) means in the context of the system being developed.

We model the definition of security for a system by specifying a high level *security softgoal* and refining it by *type*, *topic*, or AND/OR decomposition to more specific softgoals. For example, in the *threat-SIG* shown in Figure 5, the top-level softgoal *Security[Online Banking System]* is refined by *topic* to *Security[Account]* indicating that customer accounts are the major information assets to be protected in this application. This is further AND-decomposed into *Confidentiality[Account]*, *Integrity[Account]*, and *Availability[Account]* softgoals. The refinement of the definition can be done iteratively until sufficient level of details are captured to describe security objectives of the system and their scopes.

4.2 Threats Elicitation

After defining what security means to the system under development, the next step is to elicit security threats. Traditional requirements elicitation (finding out what users really need) employs such techniques as introspection, interviews, questionnaires, and protocol, conversation, interaction, and discourse analysis [9]. These techniques can also be adapted to elicit security threats by focusing attention on the perspectives of potential adversaries. In addition, Myagmar et-al describe in [20] a method of stepping through each of system's assets, reviewing a list of attack goals for each asset. These techniques can be complemented by examining the following sources of threat information:

- ***System goals***: Security threats can be derived from business goals since these are the same goals that potential adversaries may aim at preventing from being achieved. For example, in our online banking system, examining the business goal of providing uninterrupted 24-hour customer service via the web can yield the denial-of-service threat to the application, as shown in Figure 5.

- ***Threat catalogs***: Common threats and attack patterns, as well as experiences with security violations (a changing phenomenon) can all be documented in a *threat catalog* for later reuse. Such a catalog shows classes of threats, vulnerabilities and their associated attack patterns. This catalog becomes useful source document during threat elicitation.

- ***Classification schemes***: Threat classification schemes such as *STRIDE* (Spoofing, Tampering, Repudiation, Denial of service, and Elevation of privilege) [26] can be used to elicit security threats for a particular system. The system context and environment can be cross-examined against these classes of threats to determine if they are relevant to the system being developed.

- ***System Boundary Assessment***: Every legitimate system entry or exit point is a threat, as well as other possibly illegitimate access points to a system. A critical assessment of these boundaries will yield useful information about associated threats and vulnerabilities.

- ***Scenario Analysis***: Analyzing possible future events by considering alternative possible outcomes (scenarios) can yield significant information about security threats in a system. Existing scenario analysis methods such as SCRAM [25] can be adapted here by focusing on

the potential adversaries' perspectives. Correlations and probabilities can also be assigned to such scenarios to strengthen decision making process during the threat and risk analysis stage.

- ***Insiders Threat***: In recent years, there have been increasing numbers of incidents of sabotage committed by *insiders* - individuals who were, or previously had been, authorized to use the information systems they eventually employ to perpetrate harm. In a joint study on e-crime or intrusion by the United States Secret Service and the SEI-CERT, it was reported that 29 percent of incidences were perpetrated by insiders [12]. This shows that insiders pose substantial security threat to any application by virtue of their knowledge of, and access to, employer systems and/or databases. Thus, some what-if analysis on abuse of privileges by insiders can yield a lot of threat and vulnerability information.

Since security threats can only be realized through attacks when system vulnerabilities are not effectively mitigated, the identified threats can be further analyzed to identify associated vulnerabilities and possible attack patterns. These are all modeled as N-softgoals and documented in the threat-SIG. For example, in Figure 5, the vulnerability denoted by the N-Softgoal *PoorSessionManagement[Account]* is identified as a weakness that can be explored to realize the threat represented by the *UnauthorizedDisclosure[Account]* softgoal. Similarly the N-softgoal *Flooding[Account]* is an attack pattern for realizing the threat represented by the *DenialOfService[Account]* N-softgoal.

4.3 Threat and Risk Analysis

By doing threat elicitation in a context-sensitive manner, identified threats will be related to the system being developed. However, they must be analyzed to determine the degree to which the system is susceptible to them. Risks due to these threats must also be analyzed, prioritized and well managed in order to arrive at effective countermeasures.

A good risk management procedure for threat modeling has been proposed in [20]. This procedure aims at balancing what is possible (in terms of countermeasures) against what is acceptable (in terms of risk and cost). The DREAD methodology (Damage potential, Reproducibility, Exploitability, Affected Users, and Discoverability) described in [10] can be used for defining and prioritizing risks. We complement these approaches by providing visual model elements for capturing the risk analysis and decision making process. Furthermore, the following analysis steps can be used in an iterative manner and entire process documented in the *threat-SIG*.

Threat Decomposition: An identified threat modeled as an N-softgoal can be refined using AND/OR decomposition methods, or made more specific on *type* or *topic*. Then, associated vulnerabilities and attack patterns are identified and are also modeled as N-softgoals. After sufficient level of context-sensitive refinements have been attained, countermeasures then be identified to negate (*inv*) each of the lowest level N-softgoals. Each threat, vulnerability, or attack pattern must be addressed by at least a countermeasure. These countermeasures are denoted in the threat-SIG as operationalizing softgoals. For example in Figure 5, *RoleBasedAccessControl[Account]* is identified as operationalizing the *UnauthorizedDisclosure[Account]* N-softgoal, using inverse contribution semantics.

Threat Prioritization: Since it may not be feasible to eliminate all actual threats, it is important to acknowledge their presence and prioritize their associated risks in order to identify those that are *most crucial for the system* being developed. Two aspects of identified threats may be captured in the threat model, namely *impact* and *probability*. The *impact* of a threat is a measure of the consequences of its realization in the event of a compromise or an attack. The *probability* represents the likelihood that some vulnerability will be exploited to realize the threat. Quantitative metrics such as *severity* (product of *impact* and *probability*), as well as qualitative indices such as claims based on expert knowledge and experience, can be used to prioritize the identified threats. The relative priorities are then shown in the threat-SIG with exclamation marks next to the N-softgoals that represents the threat or vulnerability. The rationale for the prioritization can be explicitly captured with claim-softgoals. For example, in Figure 6, the expert judgement denoted by the claim softgoal *Claim[Customers confidentiality is most crucial]* captures the rationale for prioritizing both *Confidentiality[Account]* softgoal and the threat represented by the *UnauthorizedDisclosure[Account]* N-softgoal.

Severity/Cost Assessment: It is very easy to go overboard when designing countermeasures against identified threats. Effort may be put into addressing a threat that may as well have low severity. Even if the severity of a threat is considered high, the costs of the countermeasure must be factored into the solution design decision, else the solution may not be cost-effective. These assessments are captured as claim softgoals in the documenting threat-SIG. For example, in Figure 6, the claim softgoal *Claim[Cost of IP Filtering outweighs the spoofing risk]* represents such an assessment.

Threat Mitigation: After due analysis, a management strategy must be developed. Threats and vulnerabilities can be *accepted* (due to low severity) *transferred* (e.g. via insurance coverage), or *mitigated* as described in [20]. For the threats that must be mitigated, alternative countermeasures are to be considered and their effectiveness evaluated using refinements, correlations, and the semantics of inverse contributions. For example, Figure 6 shows that the threat denoted by the *Spoofing[Account]* N-softgoal is *accepted* based on the above cost assessment. Other identified threats in the figure are mitigated with countermeasures as shown.

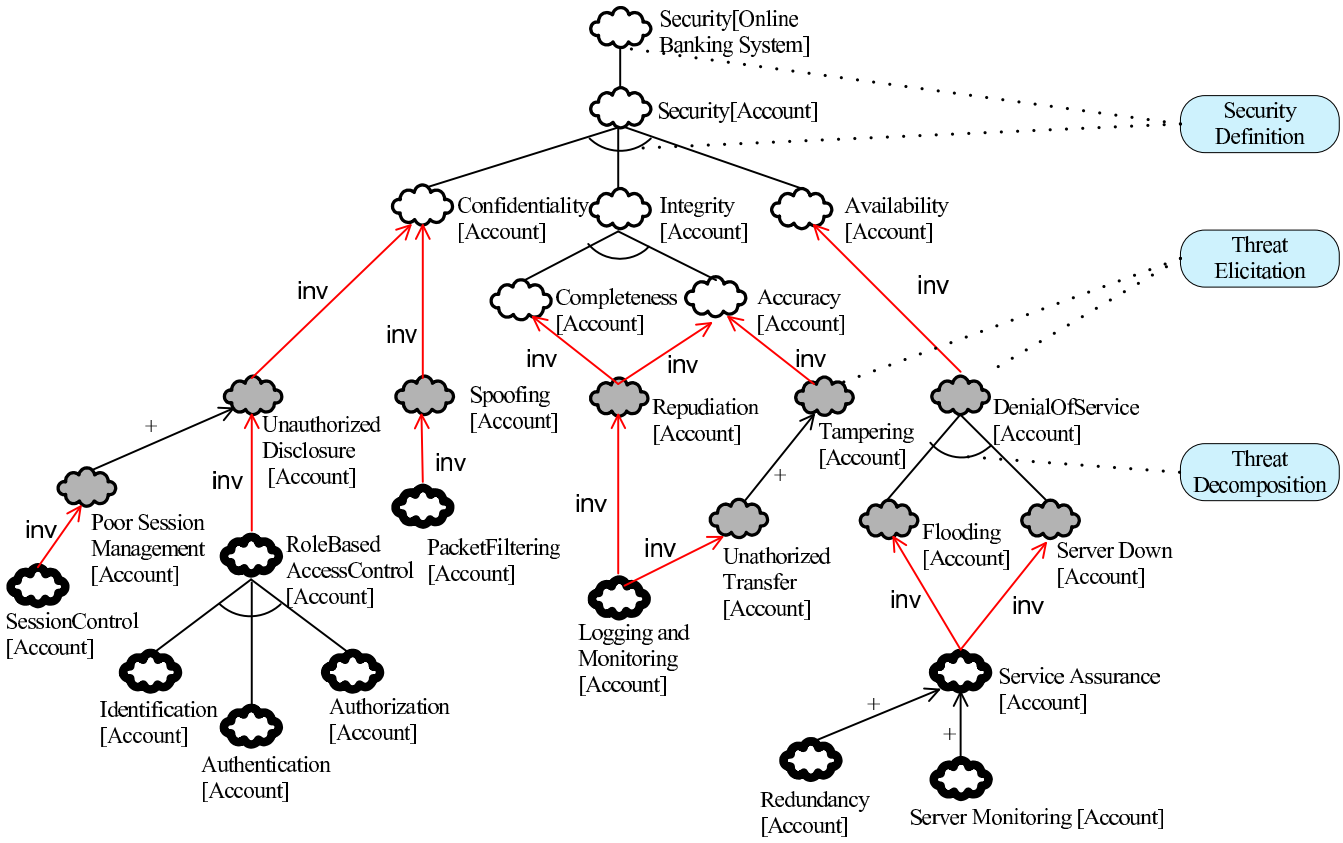

Figure 5. Initial *Threat-SIG* for Online Banking System

The above process can be done iteratively, and the threat-SIG that documents the entire process can be version-controlled at different levels of abstractions.

4.4 Threat-SIG Evaluation

At this stage, the threat-SIG documenting the process would have contained sufficient information about the threats and their mitigation strategies. The evaluation procedure (labeling algorithm) described in details in [5] is then used to evaluate the threat-SIG. Explicit contributions and implicit correlations of the countermeasures to the identified threats and vulnerabilities are specified. These evaluations are propagated upward the threat-SIG to the satisficing or otherwise of the top-level security softgoal. Figure 6 shows the resultant threat-SIG for our *Online Banking System* after evaluation. It shows that the all the expert judgements denoted by the claims softgoals are satisficed (upheld to be valid) and their effects reflected. The threat denoted by the N-softgoal *Spoofing[Account]* is accepted due to the highly negative contributions from the claim softgoal *Claim[Cost of IP Filtering outweighs the spoofing risk]*. This would have resulted in the denial of the *Confidentiality[Account]* softgoal if not for the inverse contribution from the denied *UnauthorizedDisclosure[Account]* N-softgoal. Other operationalizing softgoals are satisficed and their effects propagated upward the threat-SIG as shown. These propagations lead to the satisficing of the top-level security softgoal.

5 Related Work and Discussions

Security requirements engineering has received significant attention in recent times. Methodologies for capturing stakeholders' security concerns early enough are believed to be capable of reducing costs and improving overall security of a system. Some attempts including [15, 11, 8] have been made at incorporating security requirements into the UML. Other approaches [16, 14, 17] attempt to enhance existing requirements engineering frameworks with security-specific concepts such as agent orientation, organizational and social contexts (ownership, consent, trust, and delegation).

Threat modeling is considered as a significant step in the security requirement engineering paradigm. Its promises include revealing the highest security risks to a software product, discovering how attacks can manifest, helping to find bugs, and guiding penetration testing based on a threat model [10]. In order for any threat modeling framework to deliver these promises, certain features are desirable. These include the ability to explicitly capture all threat-related concepts, an expressive notation susceptible to rigorous analysis, a usable threat model, as well as capturing of decision rationales. It turns out that the NFR Framework [5] provides these features among others, hence its adoption in this work.

Generic threat modeling frameworks such as OCTAVE [1] provide methods for identifying, evaluating and managing information security risks at large. McDermott and Fox adapt the UML use case concept to capture and analyze se-

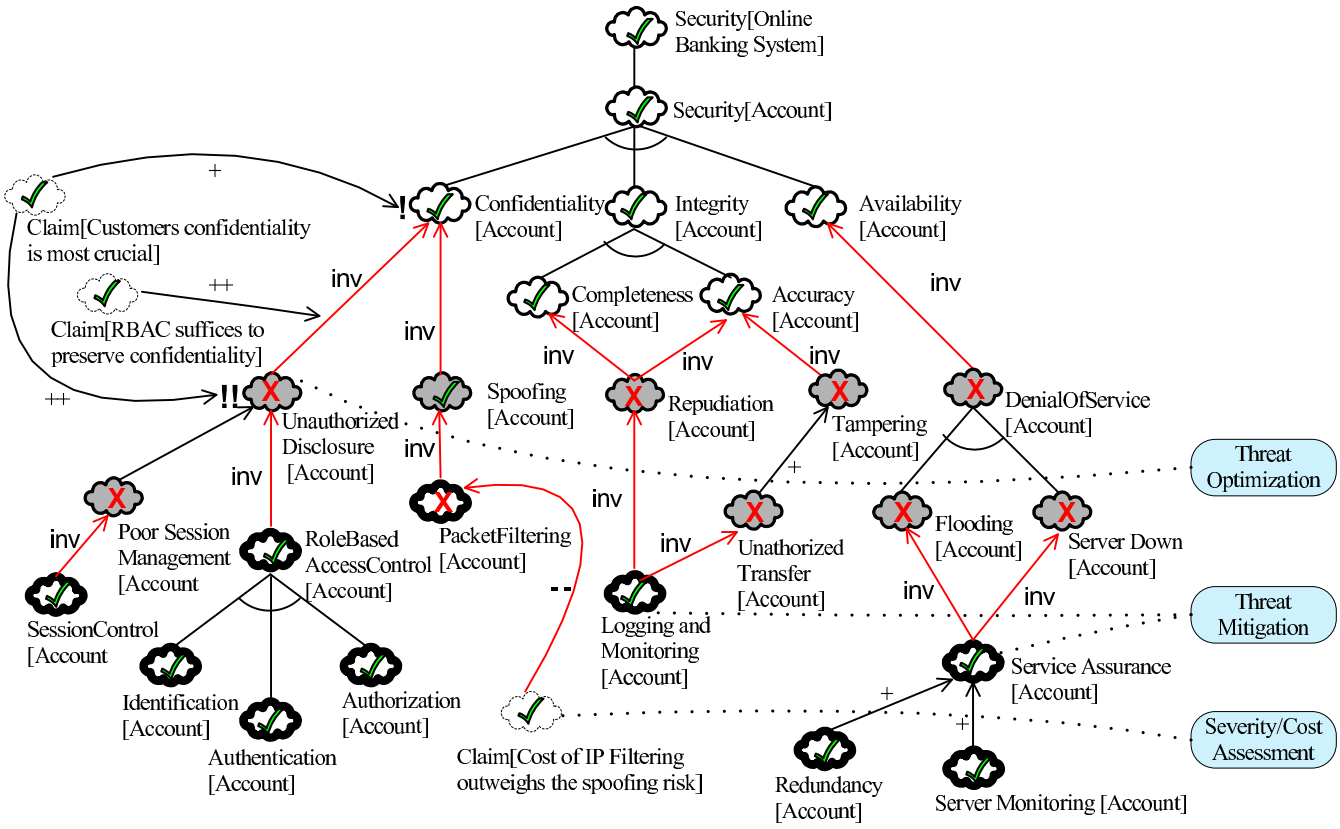

Figure 6. Evaluated *Threat-SIG* for Online Banking System

curity requirements with what they call *abuse case* models [18]. They define an abuse case as an interaction between a system and its actors where the results of the interaction are harmful to the system. A similar concept of *misuse cases* - the inverse of UML use cases - describes functions that a software system should not allow [24]. These approaches are complemented by the concept of *anti-goals* proposed in [31]. However, most of the earlier works on threat modeling use *data-flow diagrams* [28] or *attack trees* [23] as the modeling notation. While they provide useful models for identifying security threats, data-flow diagrams and attack trees have weak supports for capturing high-level security goals, countermeasures against identified threats, and the formal threat analysis process as described in this paper.

The goal-oriented approach presented in this paper complements earlier approaches by drawing on the established formal analysis and evaluation procedures of the NFR Framework. The approach supports the threat-driven elicitation of security requirements, formal threat analysis, and countermeasure. It also provides visual model elements for capturing relevant concrete and conceptual entities and the relationships between them. The resultant *threat-SIG* becomes a valuable artifact for guiding the development teams throughout the development life cycle.

We believe that the proposed approach can offer some benefits. First, once major threat-related concepts are visually represented in a threat-SIG, it becomes easier to discover any omission, conflict or harmony. Second, the threat-SIG can be used as a basis for brainstorming among system stakeholders enabling them to agree or disagree with the concepts captured. In addition, the decision rationale captured can be very useful in documenting why things are done the way they are. For example, the rationale captured by the claim softgoal *Claim[Cost of IP Filtering outweighs the spoofing risk]* in Figure 6 can help a new team member who later joins the team to understand the reason why spoofing is not addressed with a countermeasure. Moreover, with the threat-SIG, we feel that team members are better equipped with a comprehensive threat-model for guiding the security features of the system throughout the development life cycle.

However, while we opine that producing threat-SIGs at different levels of abstractions can help, we make no claims about the ease of producing very large threat-SIGs. Thus, a robust tool support that can produce feature-rich threat-SIGs from basic user inputs is desirable. Such a tool should also be able to generate useful security reports such as access control matrices. The details of such tool remains to be worked out.

6 Conclusions

Threat modeling plays a significant role in the design of the overall security model for a system because it can help to ensure that security is built into applications, rather than addressed as an afterthought. This paper presents a goal-oriented approach for modeling and analyzing security threats during requirements engineering. More specifically,

the proposed framework provides support for (1) guiding the threat modeling and analysis process using the notions of negative softgoals for capturing security threats and inverse contributions for mitigation analysis, (2) the use of visual model elements for documenting the entire threat modeling process, and (3) the capturing of rationales for choosing among alternative countermeasures. Future work will be directed at extending the existing tool for the NFR Framework (The NFR-Assistant) to provide supports for the automation of the proposed process. More case studies are also needed to determine the strength and weakness of the proposed framework.

References

[1] C. J. Alberts, S. G. Behrens, R. D. Pethia, and W. R. Wilson. Operationally critical threat, asset, and vulnerability evaluation (octave) framework. Technical Report CMU/SEI-99-TR-017, SEI, Carnegie Mellon University, May 2005.

[2] A. I. Antonn and C. Potts. The use of goals to surface requirements for evolving systems. In *Proc. of the Inter'l Conference on Software Engineering (ICSE '98)*, Kyoto, Japan, 1998.

[3] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2002.

[4] L. Chung. Dealing with security requirements during the development of information systems. In *Proceedings of 5th Int. Conference on Advanced Information Systems Engineering (CAiSE)*, June 1993.

[5] L. Chung, B. A. Nixon, E. Yu, and J. Mylopolos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[6] R. Crook, D. Ince, L. Lin, and B. Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proceedings of IEEE Int'l Requirements Engineering Conference (RE'02)*, 2002.

[7] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[8] F. den Braber, M. S. Lund, K. Stølen, and F. Vraalsen. Integrating security in the development process with UML. In *Encyclopedia of Information Science and Technology (III)*, pages 1560–1566. 2005.

[9] J. A. Goguen and C. Linde. Techniques for requirements elicitation. In *Proc. of the IEEE Int. Symposium on Requirements Engineering*, pages 152–164, Los Alamitos, CA, Jan. 1993.

[10] M. Howard and D. LeBlanc. *Writing Secure Code*. Microsoft Press, 2nd edition, 2002.

[11] J. Jrjens. UMLsec: Extending UML for secure systems development. *UML 2002, LNCS 2460, Springer-Verlag*, 2002.

[12] M. Keeney, E. Kowalski, D. Cappelli, A. Moore, T. Shimeall, and S. Rogers. Insider threat study: Computer system sabotage in critical infrastructure sectors. Technical report, Software Eng. Institute, Carnegie Mellon University, May 2005.

[13] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi. A taxonomy of computer program security flaws, with examples. Technical Report NRL/FR/5542-93-9591, Naval Research Laboratory, Washington, DC, Nov. 1993.

[14] L. Liu, E. Yu, and J. Mylopoulos. Security and privacy requirements analysis within a social setting. In *Proceedings 11th IEEE International Requirements Engineering Conference (RE'03)*, pages 151–161. IEEE Press, Sept. 2003.

[15] T. Lodderstedt, D. Basin, and J. Doser. SecureUML: A UML-based modeling language for model-driven security. *UML 2002, LNCS 2460, Springer-Verlag*, pages 426–441, 2002.

[16] P. G. F. Massacci, J. Mylopoulos, and N. Zannone. Requirements engineering meets trust management: Model, methodology, and reasoning. In *Proceedings of iTrust'04, LNCS 2995*, pages 176–190, 2004.

[17] P. G. F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In *Proc. of 13th IEEE International Conference on Requirements Engineering (RE'05)*, Aug./Sept. 2005.

[18] J. P. McDermott and C. Fox. Using abuse case models for security requirements analysis. In *Proc. of the 15th IEEE Annual Computer Security Applications Conference (ACSAC'99)*, pages 55–67, Phoenix, 1999. IEEE CS Press.

[19] A. P. Moore, R. J. Ellison, and R. C. Linger. Attack modeling for information security and survivability. Technical Report CMU/SEI-2001-TN-001, Software Engineering Institute, Carnegie Mellon University, March 2001.

[20] S. Myagmar, A. Lee, and W. Yurcik. Threat modeling as a basis for security requirements. In *Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS'05)*, Paris, France, Aug. 2005.

[21] NIST. Software errors costs U.S. economy $59.5 billion annually. Technical Report NIST 2002-10, National Institute of Standards and Technology, June 2002.

[22] J. Rushby. Security requirements specifications: How and what. In *Proceedings of the IEEE Symposium on Requirements Engineering for Information Security (SREIS'01)*, Indianapolis, March 2001.

[23] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobbs Journal*, pages 21–29, December 1999.

[24] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, January 2005.

[25] A. Sutcliffe. Scenario-based requirements analysis. *Requirements Engineering*, 3(1):48–65, 1995.

[26] F. Swiderski and W. Snyder. *Threat Modeling*. Microsoft Press, 2004.

[27] H. H. Thompson. Application penetration testing. *IEEE Security and Privacy*, 03(1):66–69, 2005.

[28] P. Torr. Demystifying the threat-modeling process. *IEEE Security and Privacy*, 03(5):66–70, Sept/Oct 2005.

[29] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proceedings of the 5th IEEE Inter'l Symposium on Requirements Engineering (RE'01)*, pages 249–263, Toronto, Canada, 2001.

[30] A. van Lamsweerde. Goal-oriented requirements engineering: A roundtrip from research to practice, invited keynote paper. In *Proc. of the 12th IEEE Joint Inter'l Requirements Engineering Conference (RE'04)*, pages 4–8, Kyoto, Sept. 2004.

[31] A. van Lamsweerde, S. Brohez, R. D. Landtsheer, and D. Janssens. From system goals to intruder Anti-Goals: Attack generation and resolution for security requirements engineering. In *Proc. of the Workshop on Requirements for High Assurance Systems (RHAS'03)*, Monterey (CA), Sept. 2003.