Requirements Eng
DOI 10.1007/s00766-015-0229-z

ORIGINAL ARTICLE

Automating trade-off analysis of security requirements

Liliana Pasquale[1] · Paola Spoletini[2] · Mazeiar Salehie[1] · Luca Cavallaro[1] · Bashar Nuseibeh[1,3]

Received: 6 December 2013 / Accepted: 26 April 2015

Abstract A key aspect of engineering secure systems is identifying adequate security requirements to protect critical assets from harm. However, security requirements may compete with other requirements such as cost and usability. For this reason, they may only be satisfied partially and must be traded off against other requirements to achieve "good-enough security". This paper proposes a novel approach to automate security requirements analysis in order to determine maximum achievable satisfaction level for security requirements and identify trade-offs between security and other requirements. We also propose a pruning algorithm to reduce the search space size in the analysis. We represent security concerns and requirements using asset, threat, and goal models, initially proposed in our previous work. To deal with uncertainty and partial requirements, satisfaction security concerns are quantified by leveraging the notion of composite indicators, which are computed through metric functions based on range normalisation. An SMT solver (Z3) interprets the models and automates the execution of our analyses. We illustrate and evaluate our approach by applying it to a substantive example of a service-based application for exchanging emails.

Keywords Security requirements · Trade-off analysis · Goals

✉ Paola Spoletini
pspoleti@kennesaw.edu

[1] Lero - the Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland

[2] Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA, USA

[3] Department of Computing, The Open University, Milton Keynes, UK

1 Introduction

Security may denote the degree to which valuable assets are protected from significant threats posed by malicious attackers [9]. Identifying adequate security requirements to protect critical assets from harm is a fundamental activity in engineering secure systems. Security requirements can be operationalised by applying different sets of security controls (security configurations) that can have a different impact on the satisfaction of security and other system goals. Estimating the consequences of alternative security configurations is essential to provide guarantees that critical assets are protected and that the systems' security goals are satisfied. However, security requirements may compete with other requirements. For example, fixed cost budgets might require relaxing the level of protection of specific assets. For this reason, security requirements may only be satisfied partially and must be traded off against other requirements to achieve "good-enough security" of the assets to be protected.

Security is characterised by incompleteness and uncertainty that make the trade-off among competing requirements imprecise. Incompleteness makes it impossible to guarantee absolute security. That is, the satisfaction of security goals can only be guaranteed for the assets within a specified boundary of protection. Uncertainty also affects the value of security concerns and their mutual relationships. For example, evaluating the presence of vulnerabilities, the criticality level of threats, and the probability of attacks cannot be determined precisely, since numerical data documenting the presence of vulnerabilities, the criticality of threats, and the occurrence of attacks in software systems are not publicly available. Therefore, evaluating the impact of assets and other

Published online: 12 May 2015

contextual factors on system security concerns cannot be performed objectively.

Several goal-based requirements analysis approaches [7, 11, 14, 27, 40] have been proposed to check whether a system meets its security requirements. However, existing work does not relate security requirements to the protection of critical assets and it makes a minimal differentiation among degrees of goal satisfaction and contributions of alternatives. Although existing requirements-based risk assessment techniques [2, 18, 25] provide information about the security risk, they do not address the trade-off among conflicting requirements. Moreover, existing requirement trade-off techniques [4, 19, 23] are usually performed manually and need to evaluate all possible system configurations, making them impractical for a high number of requirements and alternatives.

This paper proposes a novel approach to automate security requirements analysis. First, the approach allows software engineers to investigate how a certain security configuration can protect valuable assets and the maximum achievable satisfaction level that can be guaranteed for security goals. Second, trade-off between security and other requirements are identified and analysed quantitatively. Finally, to decrease the search space size of the analyses, we propose a novel pruning algorithm that removes the security controls that are unnecessary for achieving a certain protection level of assets or a specific satisfaction of security goals.

We represent security concerns (assets, threats, attacks, vulnerabilities, security goals, requirements, and controls) and conventional—functional and non-functional—requirements using asset, threat, and goal models initially proposed in existing previous work [36]. These models represent mutual impact relationships among the security concerns and the requirements of the system. Elements of the model associated with uncertainty are quantified as a float number between 0 and 1. Uncertainty can be determined by vague concepts (e.g. assets required protection level, satisfaction of security goals, presence of vulnerabilities, threats criticality level) or by imprecise measures of probability of events that may happen (e.g. attacks). To quantify the satisfaction of system requirements and security concerns, which often cannot be described by well-defined mathematical functions, we leverage the notion of composite indicators [3]. To deal with uncertainty and partial requirements satisfaction, composite indicators are computed by using metric functions based on range normalisation.

Asset, goal, and threat models—including the formalisation of the relationships among their elements—are used to encode our analysis as a satisfiability problem, which is given as input to an SMT Solver (Z3 [6]). In this way, we can automatically verify whether a configuration of security controls guarantees an adequate protection level of assets and satisfies security goals. We also balance the trade-off among conflicting goals by identifying a target satisfaction value they can achieve for a specific security configuration. We evaluate our approach by applying it to a case study concerned with security of a service-based application for exchanging emails. On the one hand, our case study provides evidence that our analysis results in the appropriate level of security. For example, to guarantee a higher level of protection of certain assets, it is necessary to employ more effective security controls, which better mitigate existing vulnerabilities. On the other hand, we show how our pruning algorithm reduces the number of configurations that must be evaluated during analysis.

The rest of the paper is organised as follows. Section 2 describes our overall approach, and Sect. 3 introduces our case study. Section 4 illustrates asset, goal, and threat models, and Sect. 5 explains their formalisation by representing precisely the metric functions adopted to quantify system requirements and security concerns. Section 6 describes our automated analyses, and Sect. 7 presents our experimental results. Section 8 reviews related work, and Sect. 9 concludes.

2 Overall approach

To support automated security trade-off analysis, our approach comprises three main phases: modelling, formalisation, and analysis, as illustrated in Fig. 1.

During the first phase, a software engineer models the security concerns and relates them to the requirements of a system. In particular, s/he designs an integrated model consisting of the asset, threat, and goal models, as initially proposed in our previous work [36]. The formalisation phase automatically augments the integrated model with a mathematical representation of the metric functions adopted to quantify represented system requirements and security concerns.

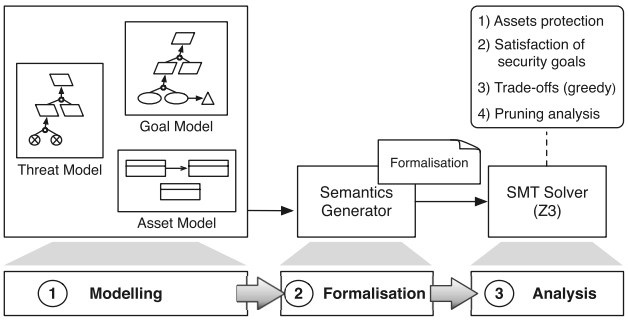

Fig. 1 Our approach for automating security analysis

The analysis phase uses the model formalisation to encode the analysis selected by a software engineer into a satisfiability problem assigned to an SMT Solver (Z3). The satisfiability problem is formulated by adding a set of constraints to the model formalisation, which vary depending on the type of analysis required. A software engineer can check whether a configuration of security controls guarantees a certain level of protection of assets or a certain satisfaction of the system security goals.

A software engineer can also perform trade-off analysis to detect conflicting goals and balance their satisfaction levels according to a specific strategy. A definition of conflicting goals for imprecise requirements was defined formally by Yen and Tiao [43]. This definition assumes a transition-based model of the system on which the requirements are expressed; two goals are considered as conflicting if their degree of conflict exceeds 0.5. The degree of conflict is measured as the ratio between the total change of satisfaction degrees of two requirements for all the system transitions in which one requirement is increasing and the other one is decreasing, and the total change of the satisfaction degrees of the two requirements for any possible system transition. We decided not to adopt this definition of conflict because it only measures the percentage of disagreement in the satisfaction of two goals and it does not consider the severity of a conflict in a single system transition. For this reason, in this paper, we consider two goals as conflicting if they simply cannot achieve their individual (maximum) levels of satisfaction at the same time.

Following this last definition, in this paper, we adopt a greedy strategy to identify requirements trade-off, as we primarily try to achieve the maximum possible satisfaction for the most critical goals, and then, we try to achieve the best possible satisfaction for the least critical goals. Without any loss of generality, our approach can adopt other definitions of conflict and can also be used for the automatic analysis of conventional (functional and non-functional) requirements. This is possible under the assumption that requirements are represented through a goal model associated with a specific semantics, as suggested by Amyot et al. [1].

Finally, we provide a pruning method that identifies the core security controls that are necessary to guarantee a certain protection level of assets or a specific satisfaction level of security goals. This allows us to reduce the search space of the analysis, which only takes into account the security configurations including the necessary security controls.

3 Case study

As a case study, we consider a service-based application—a hypothetical email service—which is used by clients having differing needs and priorities. In this scenario, the security controls applied by the email service need to be engineered appropriately to accommodate this clients' diversity quickly and cost-effectively. To identify the security concerns of the system, we follow the NIST guidelines on electronic mail security [38] describing threats, vulnerabilities, and possible security controls that can be applied on email services. The goals and requirements of our example are acquired from the Email as a Service (EaaS) Blanket Purchase Agreement (BPA) Requirements Document [39].

The main assets to be protected are the email messages and the company's business functions (e.g. sales, research & development, marketing, HR) that rely on the email services. The security goals aim to guarantee the availability of the company's business functions (business continuity) and the confidentiality and integrity of sent/received email messages. Security controls may vary depending on the criticality of the assets to be protected.

Security goals may compete with other goals. For example, if the company has a limited budget, some security controls necessary to guarantee the integrity of messages (e.g. email signatures, blocking attachments download, and mail filters) might not be purchased because they are too expensive. Login should also be "easy to perform" (usable login). This might require choosing credentials that are easy to remember (e.g. short or vocabulary-based passwords). Since such credentials can be guessed easily by an attacker, achieving usable login can conflict with goals aimed to achieve confidentiality and integrity of email messages.

Security goals may also compete among each other. For example, to support integrity of email messages it is possible to block attachments downloading or perform frequent patches updates of the email servers (SMTP, POP3, IMAP) or of their operating system. These security controls may inevitably reduce the availability of the email service, and, consequently, of the company's business functions that rely on it.

Although our case study deals with a service-based application, the proposed approach can still be applied to traditional software systems. Furthermore, the case study does not claim to be exhaustive; however, it is sufficiently complex to illustrate how our approach can support security analysis of realistic software systems.

4 Modelling

As described in the previous section, in this phase a software engineer creates an asset, a goal, and a threat model representing the security concerns and the requirements of the system. To quantify security concerns (i.e. protection level of assets, presence of vulnerabilities, threats criticality level, and probabilities of attack) and the

satisfaction of system goals and requirements, the concept of composite indicator [3] is adopted. This is calculated aggregating values of component indicators (i.e. elements in the model that are related to the target one) through a mathematical function. To deal with uncertainty and partial requirements satisfaction, composite indicators are computed by using metric functions based on range normalisation, which return a "normalised value" comprised between 0 and 1.

4.1 Asset model

The asset model represents assets to be protected, their relationships, and other contextual factors that may affect assets' criticality. An asset is any valuable cyber or physical entity that is critical for the system operation and its stakeholders [42]. An asset can represent physical devices (e.g. servers), digital objects (e.g. email messages, mission critical applications), sensitive information (e.g. user credentials), stakeholders' objectives (e.g. business functions), and intangible properties (e.g. reputation). Assets should be protected from illicit access, use, disclosure, alteration, destruction, and theft, resulting in loss to individuals and organisations [20].

Figure 2 represents the asset meta-model. Each asset is described by a name and a value, which is a float number comprised between 0 and 1. The value of an asset is used during the analysis together with the value of other assets and contextual factors associated with it to compute the asset's *protection level*, which quantifies its importance to be protected. A contextual factor identifies an environmental condition (e.g. a specific time of the day, a location) that may have an impact on the required protection level of an asset. Each contextual factor is characterised by a name (e.g. "time", "location"), a condition (e.g. on time or location), and a value, which quantifies its criticality. Among contextual factors we explicitly identify the role of the users interacting with an asset. In this case, the name identifies the role played by the user in the interaction with an asset, the condition specifies additional constraints on the role, and the value quantifies the user privilege level associated with that role.

Asset-to-asset relationships represent containment or dependency relationships between assets. They express the impact that an asset has on the protection level of its container or dependee asset. More precisely, the protection level of the target asset should be at least equal to the protection level of the related contained or dependent asset. For example, an email attachment affects the protection level of the email message in which it is contained, or the criticality of an organisation business function may affect the protection level of the email messages that are exchanged for its achievement. Context-to-asset relationships associate a contextual factor with an asset and quantify the impact that a contextual factor has on the asset protection level. More precisely, the protection level of an asset should be at least equal to the criticality of its related contextual factors whose condition is satisfied. If the contextual factor is a role, context-to-asset relationships indicate the function played by a user in the interaction with the associated asset. Asset-to-asset relationships as well as context-to-asset relationships are characterised by a weight, which is a float number comprised between 0 and 1, filtering the impact that an asset or a contextual factor has on the protection level of the target asset.

Figure 3 represents the asset model associated with our example. Assets can be email messages, attachments, and business functions. The protection level of an attachment depends on its value and also on the role of its owner. The value of an attachment may depend on the sensitivity of the document, which is medium (0.6) in this example. Different values of roles are associated with different privilege levels. In our example values 0.3, 0.6, and 1.0 are associated with the privileges of a secretary, a normal staff employee, and a manager, respectively. For other roles we assume the privilege level is equal to 0. In this example, the attachment's owner is a staff member, as her privilege level is 0.6. The protection level of a business function is only determined by its value, which depends on its criticality. In Fig. 3 this asset has medium criticality, i.e. its value is equal to 0.5. The protection level of an email message depends on its value, the protection level of its attachment—if present—and of the business functions that rely on it, as well as on the criticality of the role of its sender

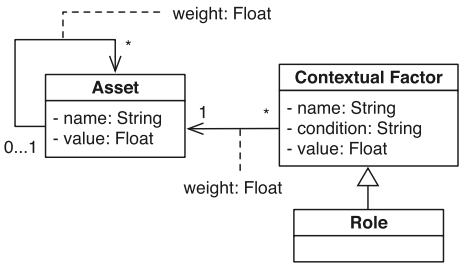

Fig. 2 Asset meta-model

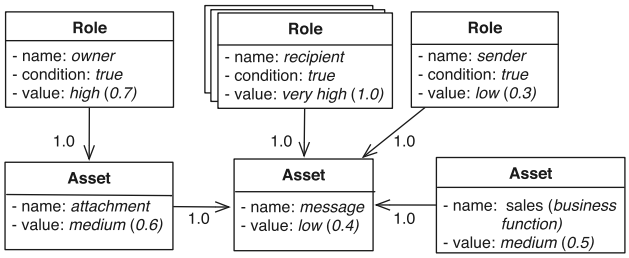

Fig. 3 Asset model of the email service example

and recipients. The value of an email message may be determined by the sensitivity of the email or by the persons that are mentioned in the email. In the example of Fig. 3, the value of the email message is low (0.4), the sender is a secretary having low privileges (0.3), while one of the recipients is a manager having high privileges (1.0). All weights of asset-to-asset and context-to-asset relationships are equal to 1.0 because we are not interested in giving more importance to the contribution of a specific asset or role to compute the value of the target asset. In particular, the protection level of an attachment and of a business function, and the privilege level of its sender and recipients have all the same importance in determining the protection level of an email message.

Note that taking into account the value of an email message for computing its protection level allows us to consider those situations in which a user having low privileges (e.g. secretary) sends an email message on behalf of another one having high privileges (e.g. manager). More precisely, in these cases the value of the message will be very high, as it will take into account the persons involved in the email discussion. The metric function adopted to compute the protection level of an asset (see Sect. 5.1) guarantees that the asset required protection level will still be high, since it will return the maximum among all the factors (components) considered to compute the protection level of an asset.

4.2 Goal model

The goal model represents the functional and non-functional requirements of the system; it extends the KAOS [41] goal model with a representation of vulnerabilities, security goals, and security controls. We explicitly distinguish goals for which there is no clear-cut definition or criteria as to whether they are satisfied or not (soft goals) [24]. These goals can be partially satisfied, i.e. their satisfaction is between 0 and 1, and, as claimed by Glinz [13], can represent both functional and non-functional requirements. We also use conditional and negative contribution links; the latter have been proposed and used in other goal-based requirements modelling approaches having a specific focus on security, such as Secure Tropos [33]. Conditional contribution links represent the fact that a target entity should be included in the model only if the source entity is enabled, i.e. the value of the source element is >0. Negative contribution links represent a negative impact of an element (a) onto the decomposed one (b), i.e. the complement w.r.t. 1 of value associated with a is considered to compute the value of b.

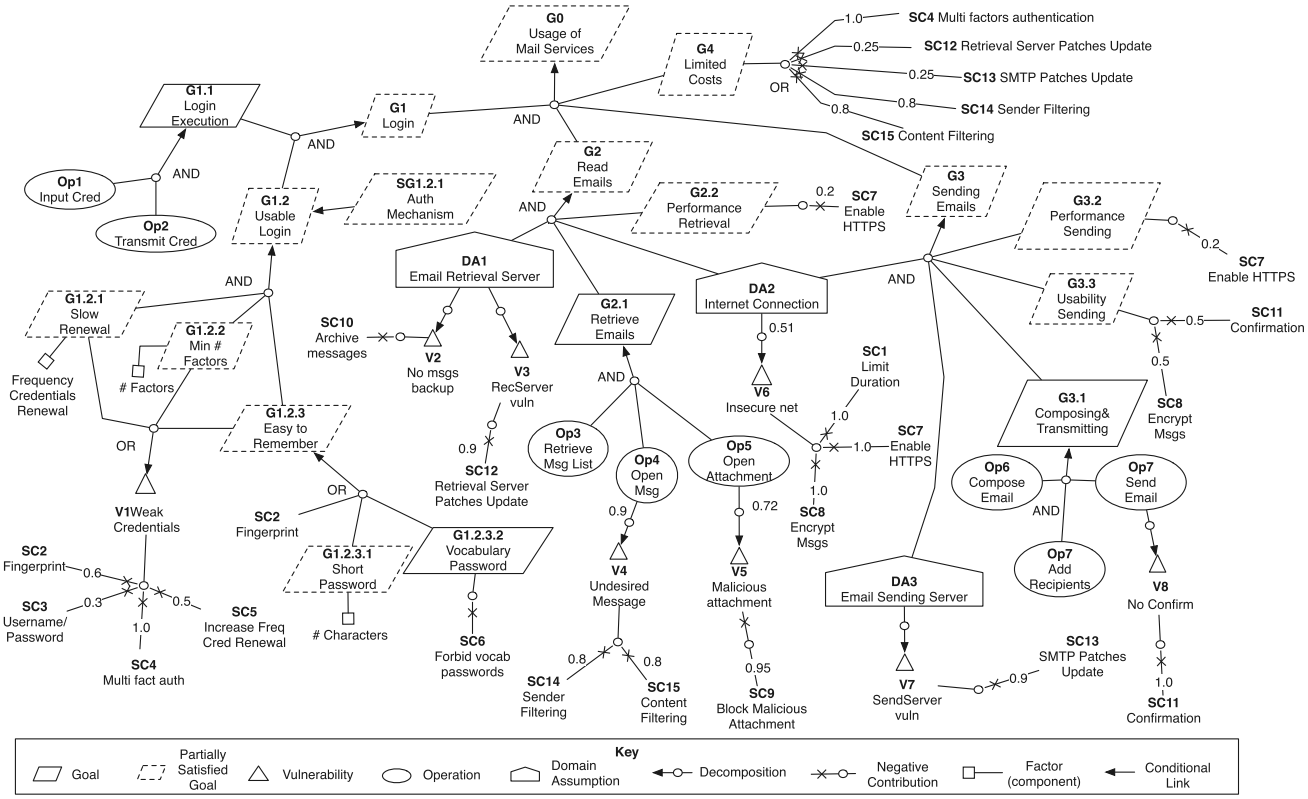

Fig. 4 The goal model for the email service example (except security goals)

Figure 4 represents the goal model of our example. We only describe some parts of this example to clarify the meaning of the elements and relationships represented in the goal model. The email service should allow users to perform login (G_1), read, and send emails (G_2 and G_3, respectively) with limited costs (G_4). To login, a user has to insert his/her credentials (Op_1) and transmit them to the identification provider (Op_2). Login should also be usable ($G_{1.2}$), i.e. easy to perform. This means that the time to renew the credentials should be as high as possible ($G_{1.2.1}$), the number of authentication factors should be as small as possible ($G_{1.2.2}$) and the credentials should be easy to remember ($G_{1.2.3}$). Satisfaction of this last goal depends on whether fingerprint authentication is enabled (security control SC_2), or a short or vocabulary password is adopted (goals $G_{1.2.3.1}$ and $G_{1.2.3.2}$, respectively). The conditional link between $SG_{1.2.1}$ and $G_{1.2}$ suggests that goal Usable Login ($G_{1.2}$) should be considered in the model only if an authentication mechanism is adopted ($SG_{1.2.1}$).

Estimating goals and obstacles satisfaction by using both qualitative [17] and quantitative analysis techniques [1, 5] has been widely explored in the literature. In this paper, the satisfaction level of a leaf goal that is partially satisfied is quantified depending on additional factors (components) [3]. For example, the satisfaction of goals Slow Renewal ($G_{1.2.1}$), Min # of Factors ($G_{1.2.2}$) and Short Password ($G_{1.2.3.1}$) depend, respectively, on the number of factors adopted for authentication, the frequency of credentials renewal, and the password length, as described by the functions shown in Fig. 5.

Goals, domain assumptions, or operations may also bring vulnerabilities, representing system weaknesses that can be exploited intentionally by a malicious agent [37]. This relation is represented in our model through decomposition links, representing the fact that the presence of vulnerability is lower bounded by the aggregation of the satisfaction of the elements that bring it. Moreover, some elements of the model can bring vulnerabilities with a certain probability, which is explicitly quantified. For example, goals $G_{1.2.1}$, $G_{1.2.2}$, $G_{1.2.3}$ may lead to the selection of weak credentials (vulnerability V_1). The open message operation (Op_4) can bring vulnerability, because undesired messages could be opened (V_4). The open attachment operation (Op_5) can also bring vulnerability, since the attachment can be malicious (V_5), i.e. it can contain a virus or a Trojan horse. For reading emails, we assume a user leverages an email retrieval server (DA_1)—POP3 or IMAP—and is connected to the Internet (DA_2). The email retrieval server (DA_1) might have some internal vulnerabilities (V_3), or it might not support messages backup (V_2). Furthermore, adopted internet connection can be insecure (V_6). The probabilities adopted to quantify the likelihood of presence of V_4–V_6 are taken from the literature. Indeed, the probability of an email message to be abusive or to contain a malicious attachment is 0.9 [31], and 0.72 [16], respectively, while the probability of using an insecure wireless internet connection is around 0.51 [32]. Note that these probability values can also be determined by a software engineer who possesses relevant domain expertise.

The goal model also includes security goals. The security community generally identifies the following key security goals: confidentiality, integrity, availability, and accountability, labelling them CIAA [35]. In this paper, security goals are treated as soft goals as they specify the degree to which valuable assets are protected from significant threats posed by malicious attackers [9]. Leaf security goals are operationalised by security controls that mitigate modelled vulnerabilities. A security control is a risk-reducing measure that can include hardware and software functionalities [37]. Each root security goal is associated with the asset it protects. Computing the satisfaction level of root security goals is fundamental to assess whether the protection level required for that asset is met. More precisely, the aggregated satisfaction level of the security goals associated with an asset must always be greater than or equal to the protection level required for that asset. Security controls can just be activated or deactivated, or they can be activated with a different degree of strength. The impact of a security control in mitigating

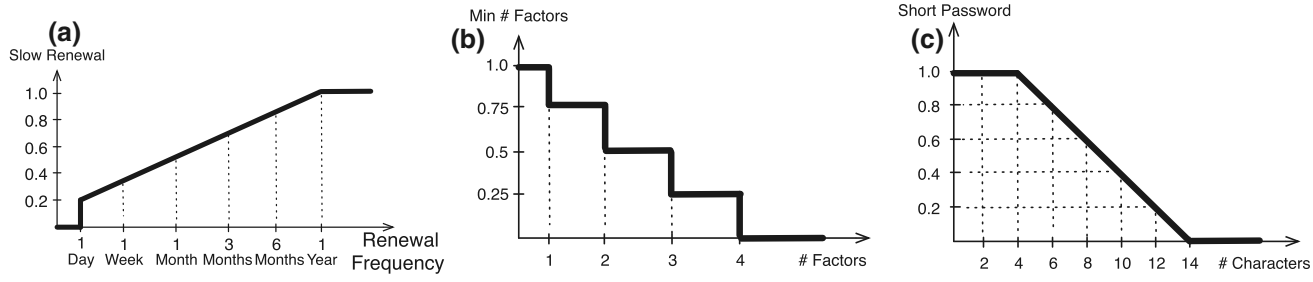

Fig. 5 Functions adopted to compute the satisfaction level of goals Slow Renewal ($G_{1.2.1}$), Min # of Factors ($G_{1.2.2}$) and Short Password ($G_{1.2.3.1}$). **a** Satisfaction of goal Slow Renewal. **b** Satisfaction of goal Min # Factors. **c** Satisfaction of goal Short Password

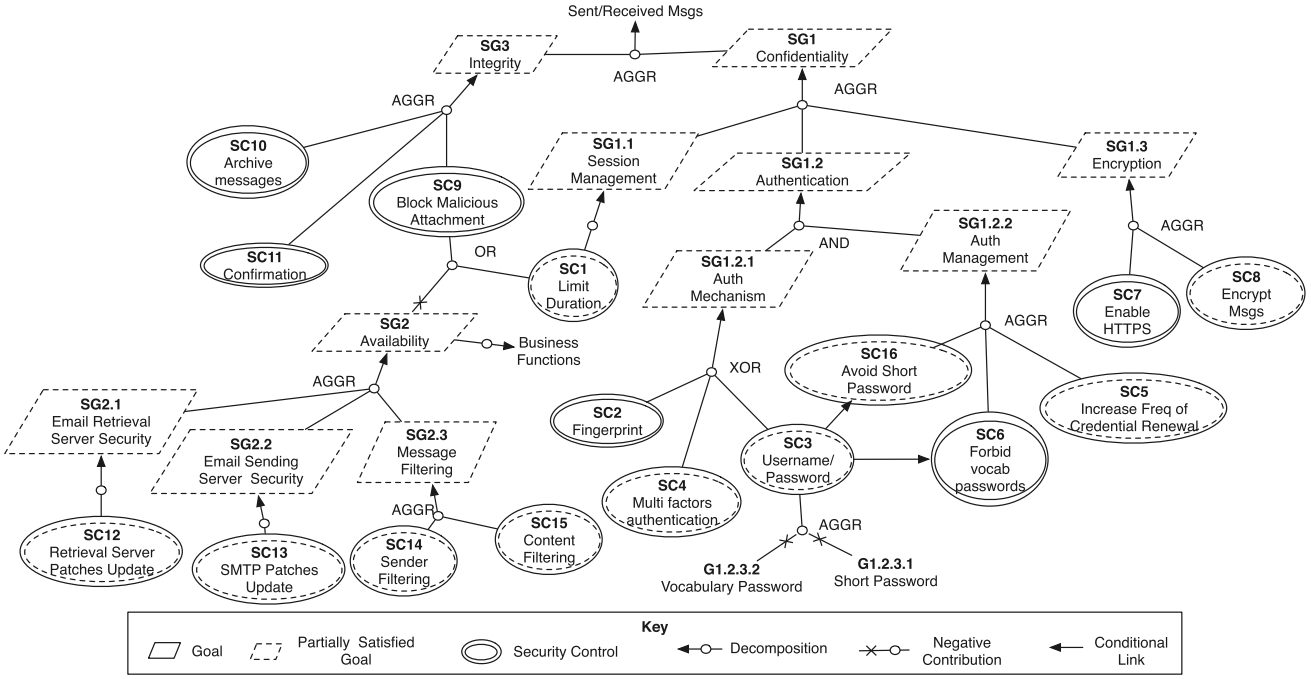

Fig. 6 Security goals and controls for the email service example

vulnerability is represented explicitly in the model—if necessary—and it is decided by the software engineer based on her own common sense.

We also introduced XOR and AGGR decompositions in the goal model. The former represents those cases in which only one goal/operation/security control can be selected among a set of alternatives. An AGGR decomposition is used when it is necessary to take into account the overall contribution of all the children in a decomposition. More precisely, the metric function adopted to compute the satisfaction level of a goal refined by an AGGR decomposition is the average among the satisfaction level/strength of the security goals/controls belonging to the decomposition.

Figure 6 represents the security goals and controls of our email service example. Note that we represented security goals in a different figure only for reasons of space.[1] Security controls that can be enabled with different degrees of strength are represented through dashed lines. A detailed description of the meaning of strength levels associated with the security controls adopted in our example is provided in "Appendix". Security controls are associated with the vulnerabilities they mitigate through negative contribution links.

To guarantee confidentiality of sent and received messages, it can be possible to perform session management ($SG_{1.1}$), authentication ($SG_{1.2}$), and encryption ($SG_{1.3}$). In this case, we use an AGGR decomposition because the satisfaction of the confidentiality will be affected by the contributions of all its sub-goals. Session management deals with limiting the session duration (SC_1), whose strength depends on the session length. Authentication is supported by applying different mechanisms ($SG_{1.2.1}$), such as fingerprint (SC_2), username/password (SC_3), and multi-factors authentication (SC_4). Note that we use a XOR decomposition because only one authentication mechanism can be applied. SC_2 can simply be enabled and disabled, the strength level of SC_3 is negatively affected by the satisfaction of goals Vocabulary Password and Short Password, and the strength level of SC_4 depends on the number of factors used for authentication. Authentication management ($SG_{1.2.2}$) should also be supported to regulate the frequency of credential renewal (SC_5), forbid vocabulary passwords (SC_6), and avoid short password (SC_{16}).

Security controls mitigate vulnerabilities with a different degree, which is explicitly quantified. This relation represents the fact that the probability of presence of vulnerability is upper bounded by the aggregation of the strength levels of the security controls that mitigate it, each of them weighted by their mitigation degree. The negative impact of security controls is represented through negative contribution links. Security controls can also have a negative impact on other goals/operations with a different degree, which is explicitly quantified.

As shown in Fig. 4, all security controls adopted to support authentication (SC_2–SC_6) mitigate with different degrees vulnerability V_1, which is brought by the usage of

[1] The root goals of Figs. 4 and 6 must be considered as sharing the same "father" goal, which has been omitted in the paper.

weak credentials. To guarantee availability of the email service, the provider should perform prompt patch updates of email retrieval and SMTP servers (SC_{12} and SC_{13}, respectively). Although these security controls mitigate internal vulnerabilities of the email retrieval and SMTP servers (V_3 and V_7, respectively), they can increase the email service costs—negative impact on goal G_4. In other words, the satisfaction of G_4 depends on the complement of the aggregation of the strength level of SC_4, and SC_{12}–SC_{15}, each of them weighted by an impact factor selected by the software engineer. Moreover, the security controls that limit the session duration (SC_1) and block the downloading of potentially malicious attachments (SC_9) can have a negative impact on the availability of the company's business functions (SG_2) that leverage the purchased email service.

4.3 Threat model

A threat model (or anti-model) [37] typically includes threat agents, i.e. threat sources or counter-stakeholder, threat goals, and attacks, i.e. threat actions. Threat agents can be natural (e.g. flood), human (e.g. hacker), or environmental (e.g. power failure). In this paper we do not consider threat agents as a part of our anti-models, and we simply represent their potential goals. Threat goals represent motivations of threat agents to attack a system. Some of them can be decomposed into anti-goals (i.e. negation of security goals [40]). This paper represents anti-goals as KAOS obstacles [40] and, for this reason, the satisfaction of a threat goal has an immediate negative impact on the satisfaction of a security goal represented through a negative decomposition link. Attacks are actions through which threat goals can be achieved [37] and security goals can ultimately be violated. Therefore, attacks can be modelled as operationalisations of threat goals. Each vulnerability is also linked to the attacks it facilitates through positive decomposition.

Figure 7 represents the threat model for the email service example. An attacker might impersonate the victim (T_1)—and compromise the integrity of exchanged email messages—by performing unauthorised login ($T_{1.1}$) and sending compromising emails (A_1). This is facilitated by the lack of a confirmation from the authorised user for sending emails (V_8). Unauthorised login can be performed when credentials are stolen ($T_{1.1.1}$). This threat can be achieved through a malware that controls the users' session (A_3), which can be installed when a malicious attachment is opened (V_5). Other possible attacks through which credentials can be stolen are brute force (A_4) and SMTP or IMAP/POP3 malware (A_5 and A_6, respectively). These attacks are facilitated, respectively, by weak credentials (V_1) or internal vulnerabilities of the servers used to send or receive emails (V_7 and V_3). Integrity of email messages can also be harmed by removing emails (T_2). This threat can be

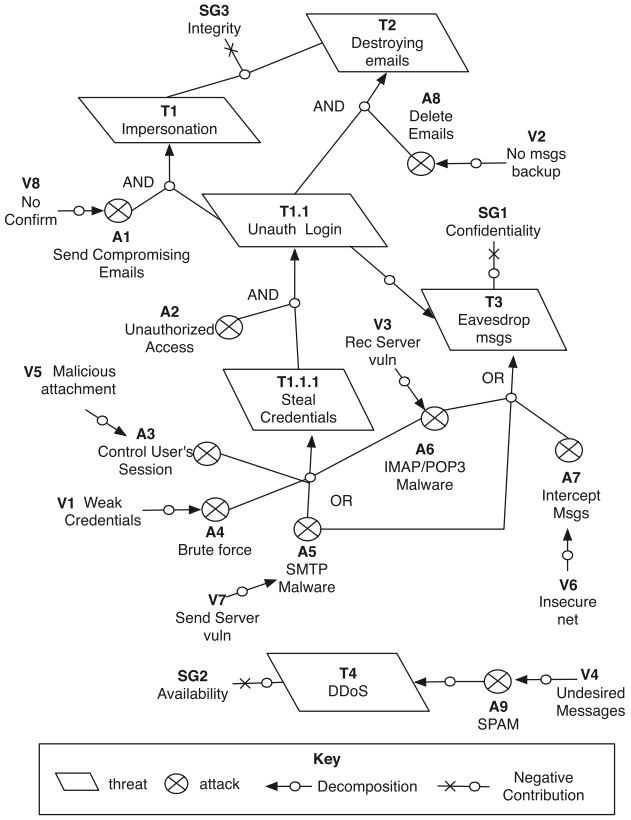

Fig. 7 The threat model for the email service example

achieved by performing unauthorised login ($T_{1.1}$) and deleting emails (A_8). This attack is facilitated by the absence of email messages backup (V_2).

An attacker might also eavesdrop exchanged email messages (T_3) and harm their confidentiality. This goal can be achieved by performing unauthorised login ($T_{1.1}$) or by perpetrating alternative attacks, such as SMTP and IMAP/POP3 malware (A_5 and A_6, respectively), or by intercepting messages (A_7). This last attack is facilitated by the presence of an insecure network connection (V_6). Finally, an attacker can compromise the availability of business functions (T_4), by, for example, sending SPAM (A_9). This attack can be facilitated by the vulnerability that a user can open undesired messages (V_4).

5 Model formalisation

This section describes how the elements and connections in the asset, goal, and threat models can be formalised[2] using an extended version of propositional logic, defined by the following grammar:

[2] A complete formalisation of the case study can be found at https://sites.google.com/site/resexperiments/.

$$
\begin{aligned}
\texttt{freq} ::= & \texttt{creq} \mid \texttt{fexpr}\ \texttt{fcomp}\ \texttt{fexpr} \mid \\
& \texttt{freq} \curlywedge \texttt{freq} \mid \texttt{freq} \curlyvee \texttt{freq} \mid \sim \texttt{freq} \\
\texttt{fexpr} ::= & \texttt{fconst} \mid \texttt{fvar} \mid \texttt{f}\,(\texttt{fexpr}^{+}) \mid \texttt{bexpr} \\
\texttt{fcomp} ::= & \succ \mid \prec \mid \succeq \mid \preceq \mid \approx \mid \not\approx \\
\texttt{creq} ::= & \texttt{bexpr}\ \texttt{bcomp}\ \texttt{bexpr} \mid \\
& \texttt{creq} \wedge \texttt{creq} \mid \texttt{creq} \vee \texttt{creq} \mid \neg \texttt{creq} \\
\texttt{bexpr} ::= & \texttt{bconst} \mid \texttt{bvar} \mid \texttt{f}\,(\texttt{bexpr}^{+}) \\
\texttt{bcomp} ::= & > \mid < \mid \geq \mid \leq \mid = \mid \neq
\end{aligned}
$$

A crisp formula (creq) can only be true or false, i.e. it can only assume values 0 or 1. As in classical propositional logic, a crisp formula can use boolean connectives ($\wedge$, $\vee$, $\neg$) and relational operators. A fuzzy formula (freq) has a truth value comprised between 0 and 1. This formula has the same structure of its crisp counterpart, with the only difference that it uses fuzzy connectives and fuzzy relational operators (fcomp) on float variables (fvar) and float constants (fconst). Float variables and constants can assume values in the interval $[0, 1]$.

The fuzzy connectives $\sim$, $\curlywedge$, and $\curlyvee$ can be interpreted by using the functions described in Table 1. In this paper, according to Zadeh's semantics [44], we interpret fuzzy connectives $\sim$, $\curlywedge$ and $\curlyvee$ as the complement w.r.t. 1, the minimum, and the maximum, respectively. This interpretation is a compromise between a more conservative interpretation, i.e. product and probabilistic sum, for $\curlywedge$ and $\curlyvee$, respectively, and a less conservative interpretation, i.e. Lukasiewicz and bounded sum, for $\curlywedge$ and $\curlyvee$, respectively.

Fuzzy relational operators (fcomp) are interpreted according to a trapezoidal membership function. In particular, operator $\approx$ is interpreted using an isosceles trapezoid with height equal to 1 for the values in the domain that are close to the term of comparison. Operator $\not\approx$ is interpreted as the complement w.r.t. 1 of the result obtained by applying operator $\approx$. Inequalities are interpreted as rectangular trapezoids. For example, for relational operators $\succ$ and $\succeq$, the height of the trapezoid is equal to 1 for the values in the domain that are greater or greater equal than the term of comparison. Function f is used to formalise the AGGR decomposition that, in this paper, is interpreted as the average function (f_{AVG}). A weighted average is adopted if weights are explicitly added to the refinement links associated with the children elements of the decomposition.

Table 1 Functions for interpreting fuzzy connectives

Type	Function	
$\sim a$	Complement	$1 - a$
$a \curlywedge b$	Minimum	$min(a, b)$
	Product	$\top(a, b) = a * b$
	Lukasiewicz	$\top(a, b) = max(0, a + b - 1)$
$a \curlyvee b$	Maximum	$\bot(a, b) = max(a, b)$
	ProbabilisticSum	$\bot(a, b) = a + b - a.b$
	BoundedSum	$\bot(a, b) = min(a + b, 1)$

Notice that changing the interpretation given to fuzzy connectives (e.g. by using Lukasiewicz's [29] interpretation), to fuzzy relational operators (e.g. by using smoother functions), and to function f (e.g. by using the geometric mean) does not require changing the modelling language, which is independent from its interpretation. However, the evaluation of functions interpreted on real numbers will be limited to a fixed number of digits of decimal precision.

5.1 Assets

Each asset A is associated with a float variable, PL_A, that indicates the minimum protection level that must be guaranteed for A. The required protection level of an asset depends on its value—if specified—and on the contextual factors associated with it, including the role of the users that interact with it. More precisely, the protection level PL_A of an asset A, with value v_A, associated with n contextual factors $CF_1, \ldots, CF_n$, and m roles $R_1, \ldots, R_m$, is bounded according to the following constraints:

$$
\begin{aligned}
& c_{CF_1} \Rightarrow PL_A \succeq w_{CF_1-A} * v_{CF_1}, \\
& \ldots, \\
& c_{CF_n} \Rightarrow PL_A \succeq w_{CF_n-A} * v_{CF_n}, \\
& c_{R_1} \Rightarrow PL_A \succeq w_{R_1-A} * v_{R_1}, \\
& \ldots, \\
& c_{R_m} \Rightarrow PL_A \succeq w_{R_m-A} * v_{R_m}, \\
& PL_A \succeq v_A.
\end{aligned}
$$

The first n constraints lower bound the protection level of A using the value of the associated contextual factors if their corresponding condition ($c_{CF_1}, \ldots, c_{CF_n}$) is satisfied. Each value is multiplied by the weight ($w_{CF_1-A}, \ldots, w_{CF_n-A}$) assigned to the link connecting the corresponding contextual factor to A. The following m constraints lower bound the protection level of A using the value of the associated roles if their corresponding condition ($c_{R_1}, \ldots, c_{R_n}$) is satisfied. Each value is multiplied by the weight ($w_{R_1-A}, \ldots, w_{R_n-A}$) assigned to the link connecting the corresponding role to A. The last constraint lower bounds the protection level of A using its value.

The protection level of an asset also depends on the protection level of other assets that are related to it. More precisely, if assets $A_1, \ldots, A_k$ are associated with an asset A, the protection level of A has to be at least equal to the protection level of the dependent assets:

$$PL_A \succeq \text{Max}\{PL_{A_1}, \ldots, PL_{A_k}\}.$$

For our case study, three PL variables are defined to represent the protection level of assets business function

(PL_{bf}), email message (PL_{msg}), and attachment (PL_{attach}). Following the asset model shown in Fig. 3 and the formalisation presented above, the protection level of an attachment is constrained as follows:

$$\text{true} \Rightarrow PL_{attach} \succeq v_{owner},$$
$$\text{true} \Rightarrow PL_{attach} \succeq v_{attach},$$

where v_{owner} and v_{attach} are the protection levels of role owner and asset attachment, respectively. The protection level of an email message depends on the protection level of its attachments—if present—and of the business functions that rely on it, and on the privilege level of its sender and recipients. In the example in Fig. 3, the message protection level must satisfy the following constraint:

$$PL_{message} \succeq Max\{PL_{attach}, PL_{bf}, v_{send}, v_{rec}\}.$$

5.2 Goals

Each goal in the model is associated with a variable expressing its satisfaction. More precisely, goals that can be partially satisfied are represented as float variables, while goals that can only be satisfied/unsatisfied are represented as boolean variables. For example, float variable *limitedCost* is used to represent the satisfaction level of goal Limited Cost (G_4); the closer its value to 1, the higher G_4's satisfaction. Any operation is conceived as a proposition, which is equal to 1, in case it is performed successfully, and it is equal to 0, otherwise. Instead, security controls are represented as crisp or float variables according to their representation in the model. Vulnerabilities are represented as their probability of presence. For example, V_1 is represented by proposition *insecureNet* that expresses the probability of presence of an insecure network. Domain assumptions are represented as crisp variables. All the goals that are decomposed by a domain assumption can have satisfaction level greater than 0, only if the associated domain assumptions (or the boolean combination of the associated domain assumptions) is true. In particular, if a goal G is R-refined (with $R \in \{\text{AND, OR, XOR}\}$[3]) by n domain assumptions $DA_1, \ldots, DA_n$, its satisfaction level is affected by their satisfaction as follows:

$$G > 0 \Rightarrow R(DA_1, \ldots, DA_n).$$

XOR is not represented through a native operator, and it is formalised using conjunction, disjunction, and negation, in order to guarantee the exclusivity of the OR.

[3] R cannot include AGGR refinements since, from a modelling point of view, aggregation of domain assumptions, which are boolean variables, is meaningless.

As an example of domain assumptions decomposition, it is possible to consider goal Reading Emails (G_2) that can only have a satisfaction level greater than 0 if an email retrieval server is used (DA_1), such as IMAP or POP3, or if an internet connection is employed (DA_2). This relationship is formalised as follows:

$$readEmails \Rightarrow internetConnection \wedge retrServer.$$

As described in Sect. 4.2, our model includes different kinds of goal decompositions. The satisfaction level of the parent goal of an AND decomposition is computed as the conjunction of the formulae representing the children elements (goals, operations, and/or security controls), while the satisfaction level of the parent goal of an OR decomposition is computed as the disjunction of the formulae representing the children elements. For example, the satisfaction level of goal Usable Login ($G_{1.2}$) is formalised as:

$$usableLogin \approx slowRenew \curlywedge minNumFact \curlywedge easyRem$$

where goals $G_{1.2.1}$, $G_{1.2.2}$, and $G_{1.2.3}$ are represented by float propositions *slowRenew*, *minNumFact* and *easyRem*, respectively. Fuzzy connective $\curlywedge$ is adopted in this case, because goal Usable Login can be partially satisfied; crisp connective $\wedge$ would have been used otherwise.

For a XOR decomposition, the value of the variable associated with the children element (goal/operation/security control) having value greater than 0 will determine the satisfaction of the parent element in the decomposition. More precisely, if a goal/operation/security control C is XOR decomposed by n goal/operation/security controls ($C_1, \ldots, C_n$), the value of C is constrained as follows:

$$(C_1 > 0) \Rightarrow (C_1 \approx C) \wedge \cdots \wedge$$
$$(C_n > 0) \Rightarrow (C_n \approx C).$$

Moreover, a XOR decomposition requires that exactly one among the goals/operations/security controls in the decomposition is enabled. This can be formalised as follows:

$$(C_1 > 0) \Rightarrow (C_2 = 0) \wedge \cdots \wedge (C_n = 0) \wedge$$
$$\cdots$$
$$\wedge (C_n > 0) \Rightarrow (C_1 = 0) \wedge \cdots \wedge (C_{n-1} = 0) \wedge$$
$$(C_1 > 0) \vee \cdots \vee (C_n > 0).$$

For example, the satisfaction of security goal Authorisation Mechanism ($SG_{1.2.1}$) can be expressed as follows:

$$(fing > 0) \Rightarrow (auth \approx fing) \wedge$$
$$(usrPwd > 0) \Rightarrow (auth \approx usrPwd) \wedge$$
$$(multiFact > 0) \Rightarrow (auth \approx multiFact).$$

These constraints are paired with the constraints that enforce the satisfaction of the XOR decomposition:

$$(fing > 0) \Rightarrow (usrPwd = 0) \wedge (multiFact = 0) \wedge$$
$$(usrPwd > 0) \Rightarrow (fing = 0) \wedge (multiFact = 0) \wedge$$
$$(multiFact > 0) \Rightarrow (fing = 0) \wedge (usrPwd = 0) \wedge$$
$$(fing > 0) \vee (usrPwd > 0) \vee (multiFact > 0).$$

For an AGGR decomposition, the satisfaction of the parent goal is computed depending on the aggregate value of its children. Hence, if a parental goal G is decomposed by n goals/operations/security controls ($C_1, \ldots, C_n$), each of them associated with a weight ($w_1, \ldots, w_n$), its satisfaction is computed as follows:

$$G \approx f_{AGGR}(w_1 * C_1, \ldots, w_n * C_n).$$

For example, the value of security goal Authorisation Management ($SG_{1.2.2}$) is computed as follows:

$$authMng \approx f_{AVG}(incrFreq, forbidShort),$$

since the average is the aggregation function adopted in this paper. Since weights are not explicitly associated with the children security controls of the AGGR decomposition of goal $SG_{1.2.2}$, f_{AVG} corresponds to the plain average.

If a security control has an impact on a (security) goal (G), this is represented as a positive or negative contribution to the satisfaction of G. In this case, the strength of a security control must always be multiplied by the weight assigned to the contribution link. Hence, if a goal G is R-refined by the n security controls $SC_1, \ldots, SC_n$, with weights $w_{SC_1-G}, \ldots, w_{SC_n-G}$, respectively, and whose the first $m < n$ have a positive contribution and the last $n - m$ have a negative contribution, the satisfaction level of G is formalised as follows:

$$G \approx D(w_{SC_1-G} * SC_1, \ldots, w_{SC_{SC_m}-G} * SC_m,$$
$$w_{SC_{m+1}-G} * \sim SC_{m+1}, \ldots, w_{SC_n-G} * \sim SC_n).$$

For example, security goal Availability (SG_2) of email services is not only positively affected by the goals that support secure email retrieval and sending ($SG_{2.1}$ and $SG_{2.2}$, respectively) and message filtering ($SG_{2.3}$), but it is also affected negatively by other security controls that limit the session duration (SC_1) or block potentially malicious attachments (SC_9):

$$availability \approx f_{AVG}(retServerSecurity,$$
$$SMTPServerSecurity, msgFiltering,$$
$$1 - limitDur, 1 - blockAttachment).$$

Note that, in the example above, we substituted the negation with its Zadeh's interpretation.

Our model also comprises conditional links that represent the fact that a target element E_T should be included in the model only if the source element E_S is enabled. This is formalised as follows:

$$E_T > 0 \Rightarrow E_S > 0.$$

For example, the relationship between the security goal that represents the authorisation mechanism ($SG_{1.2.1}$), and the security goal usable login ($SG_{1.2}$) is a conditional link, where $SG_{1.2.1}$ is the source and $SG_{1.2}$ is the target, and can be expressed as follows:

$$usableLogin \Rightarrow auth.$$

The probability of presence of vulnerability V is lower bounded by the R-refinement (with R $\in$ {AND, OR, XOR, AGGR}) of the elements that bring it, such as operations, goals, domain assumptions, and security controls ($E_1, \ldots, E_n$), each of them multiplied by its assigned weight ($w_{E_1-V}, \ldots, w_{E_n-V}$). The higher the value of these elements, the higher the probability of presence of V. Such a constraint is formalised as follows:

$$V \succeq R(w_{E_1-V} * E_1, \ldots, w_{E_n-V} * E_n).$$

For example, the assumption that a user is connected to the internet (DA_2) influences the probability of using an insecure network (V_6):

$$insecureNet \succeq 0.51 * internetConnection.$$

Security controls are linked to the vulnerabilities they mitigate through negative contribution links. More precisely, the negation of the aggregated value of the weighted security controls imposes an upper bound to the value of V. This is represented by the following constraint:

$$V \preceq \sim R\{w_{SC_1-V} * SC_1, \ldots, w_{SC_n-V} * SC_n\},$$

where V is R-refined (with R $\in$ {AND, OR, XOR, AGGR}) by a set of security controls ($SC_1, \ldots, SC_n$), each of them connected to V through negative contribution links with weights $w_{SC_1-V}, \ldots, w_{SC_n-V}$, respectively. For example, the relationship between the crisp security controls that limit the duration of a session (SC_1), enable HTTPS (SC_7) and encrypt attachments (SC_8) and the vulnerability of using an insecure network (V_6) can be expressed as follows:

$$insecureNet \succeq 1 - Max\{limitDur, encrypt, enHTTPS\}.$$

The satisfaction of the security goals that aim to protect an asset must always be at least equal to the protection level required for that asset. This guarantees that only the configuration of security controls that provides an adequate level of protection of the assets are plausible configurations. Hence, if a set of security goals $SG_1, \ldots, SG_n$ aims to protect an asset A, and $w_{SG_1}, \ldots, w_{SG_n}$ are the weights quantifying the impact of $SG_1, \ldots, SG_n$ on the protection level of A, the satisfaction level of the security goals and the protection level of the asset are related as follows:

$$PL_A \preceq f_{AGGR}(w_{SG_1} * SG_1, \ldots, w_{SG_n} * SG_n).$$

For example, considering that SG_1 and SG_3 are aimed to guarantee the confidentiality and integrity of the email messages, their weights are equal to 1.0 and are also formalised as fuzzy propositions *confidentialityM* and *integrityM*, respectively, their aggregated value must satisfy the following constraint:

$$PL_{msg} \preceq f_{AVG}(confidentialityM, integrityM).$$

5.3 Threats

The propositions representing threats and attacks express their probability of success. The probability of success of an attack *att* is lower bounded by the vulnerabilities $V_1, \ldots, V_n$ that facilitate it:

$$P(att) \succeq f_{AGGR}(V_1, \ldots, V_n)$$

For example, the probability of success of an IMAP/POP3 Malware attack (A_6) is lower bounded by the retrieval server vulnerability (V_3).

$$P(attIMAP\_POP\_Malware) \succeq recServerVuln.$$

The probability of success of a threat T is approximately equal to the fuzzy disjunction of the probability of success of the sub-threats $T_1, \ldots, T_n$ and attacks $att_1, \ldots, att_m$ in its decomposition. The disjunction can be interpreted by using the functions described in Table 1, i.e. maximum, bounded sum, and probabilistic sum. In this paper the probability of success of a threat is evaluated as the most successful of its sub-threats and attacks, as we have chosen the maximum to compute the fuzzy disjunction. This is a compromise choice between the most conservative interpretation ensured by the probabilistic sum and the least conservative interpretation determined by the bounded sum. The probability of success of a threat is formalised as follows:

$$P(T) \approx Max\{P(T_1)\ldots, P(T_n), P(A_1), \ldots, P(A_m)\}$$

For example, the probability of success of the threat that aims to eavesdrop messages (T_3) is related to a function that aggregates the contributions of its sub-threats, such as unauthorised login (T_1), and attacks, such as SMTP malware (A_5), IMAP/POP malware (A_6), and intercepting messages over an unprotected network (A_7).

$$\begin{aligned} P(eaveMsg) \approx Max\{&unauthLogin,\\ &SMTP\_Malware,\\ &attIMAP\_POP\_Malware,\\ &attInterceptMsg\}. \end{aligned}$$

Each threat contributes negatively to the satisfaction of security goals. For example, the satisfaction of security goal Confidentiality (SG_1) can be expressed as the aggregation of the negative contributions coming from threat eavesdrop messages and the positive contributions coming from sub-goals Session Management ($SG_{1.1}$), Authentication ($SG_{1.2}$), and Encryption ($SG_{1.3}$):

$$\begin{aligned} confidentialityM \approx f_{AVG}(&1 - P(eavMsg), authentication,\\ &sessionManagement, encryption). \end{aligned}$$

6 Analysis

To support security analyses, we encode the formalisation of assets, goals, and threats into a model expressed in the language of an SMT Solver (Z3) (Sect. 6.1). Our analyses[4] are performed by solving a set of satisfiability problems obtained by constraining the model in different ways. Supported analyses allow verifying whether a configuration of security controls protects critical assets of the system adequately (Sect. 6.2) and guarantees a certain satisfaction of security goals (Sect. 6.3). Trade-off analyses are also supported to balance the satisfaction among competing goals (Sect. 6.4). Finally, a pruning technique is proposed to reduce the search space size of the analyses (Sect. 6.5).

Note that, as an alternative, the same analyses could have been expressed as a Constraint Solving Problem (CSP), where the relations among the elements of the asset, goal, and threat models are represented as constraints and the objective of the problem is to maximise or minimise the function that relates assets and goals of interest. However, Z3 is a more mature tool compared with available software solutions developed for CSP problems. In addition, SMT approaches can efficiently solve problems that, at a first sight, do not have a typical SMT flavour. For example, problems where models are sought such that a given cost function is minimised [34].

6.1 Preliminaries

Translating the formalisation of assets, goals, and threats into the language of Z3 is straightforward. Crisp propositions and fuzzy propositions are declared as boolean and real variables in $[0, 1]$, respectively. Note also that all formulae must be expressed in the prefix form adopted by Z3. For example, the following constraint expresses the relation between the vulnerability related to the use of an insecure network (V_6) and the domain assumption that brings it. More precisely, if an internet connection is adopted, the

[4] The complete set of experiments can be found at https://sites.google.com/site/resexperiments/.

probability of using an insecure network is less than equal 0.51:

$(\texttt{assert}\,(=\,>\,\texttt{internetConn}\,(<\,=\,\texttt{PinsecureNet}\,0.51))).$

Fuzzy formulae must also be converted into crisp ones in order to be processed by Z3. This is possible because fuzzy formulae always employ fuzzy relational operators in their specification. In particular, the fuzzy relational operator employed in a fuzzy formula is translated into the corresponding crisp one and the term of comparison is relaxed by adding and removing a constant k. This is equivalent to using a rectangular membership function for interpreting relational operators. For example, the following constraint expresses the AGGR decomposition of fuzzy goal *msgFiltering* into security controls *sendFiltering* and *contentFiltering*:

```
(assert (and
   (<= (- (/ (+ sendFiltering contentFiltering)
      2) k) msgFiltering)
   (<= msgFiltering (+ (/ (+ sendFiltering
      contentFiltering) 2) k)))).
```

In other words, $f_{AVG}(sendFiltering, contentFiltering) - k <\,=\, msgFiltering <\,=\, f_{AVG}(sendFiltering, contentFiltering) + k$.

6.2 Assets protection

The analysis presented in this section checks whether a security configuration guarantees satisfactory protection level of an asset. For example, a software engineer may want to verify whether disabling the filtering of email messages depending on the sender (*sendFiltering*) and on the content of the message (*contentFiltering*) would still allow protecting adequately one of the business functions relying on the email service. In this case, the required protection level of the business function (*PLbus*) should be greater than equal to 0.8. This analysis can be performed by adding the following set of constraints to the Z3 model:

$(\texttt{assert}\,(\,>\,=\texttt{PLbus}\,0.8))$
$(\texttt{assert}\,(=\texttt{sendFiltering}\,0))$
$(\texttt{assert}\,(=\texttt{contentFiltering}\,0)).$

If the model is satisfiable, the constraints specified on the desired configuration of security controls guarantee the desired protection level of the assets. The problem mentioned above is not satisfiable, since it is not possible to achieve the desired protection level of the business function by disabling both sender and content filtering. Alternatively, it is possible to verify that a configuration of security controls that enables patches updates for the email retrieval/sending servers and sender filtering can guarantee an adequate protection level of the business functions. This analysis can be performed by adding the following set of constraints to the Z3 model:

$(\texttt{assert}\,(\,>\,=\texttt{PLbus}\,0.8))$
$(\texttt{assert}\,(=\texttt{sendFilte intentionally ring}\,1))$
$(\texttt{assert}\,(=\texttt{retPatchesUpdate}\,1))$
$(\texttt{assert}\,(=\texttt{SMTPPatchesUpdate}\,1)).$

6.3 Satisfaction of security goals

Another possible analysis consists of identifying the maximum satisfaction that can be guaranteed for a security goal. To achieve this aim, a binary search is performed in the interval $[0, 1]$, as described in Algorithm 1. This algorithm receives as input a reference to the Z3 model (M), the goal to be maximised (fG), and the lower (*low*) and upper (*high*) bounds of the search interval. At the beginning, *low* and *high* are set to 0 and 1, respectively. The algorithm identifies a satisfaction *value* in the middle of the search interval (line 2) and runs Z3 to verify whether the input goal can reach it (lines 3–4). In case a solution cannot be found, i.e. the model is unsatisfiable, the algorithm is re-invoked by performing the search in the interval $[low, value)$ (lines 5–6). Otherwise, it is necessary to verify whether a solution can still be found in case the *value* is incremented by 0.001 (line 8–10). If the problem is unsatisfiable, it means that the previous *value* is the maximum satisfaction that the input goal can achieve (lines 11–12). Otherwise, the search continues in the interval $(value, high]$. In the worst case, the complexity of this algorithm is $\Theta(log(10^3) * C_{Z3}(M))$, since the search is performed up to three digits of decimal precision ($|_{td}$ in line 2). $C_{Z3}(M)$ is the complexity of the satisfiability problem that is polynomial, as quantifier-free satisfiability problems for linear arithmetic are solvable by Z3 in polynomial time.[5]

The algorithm guarantees to find the absolute maximum and not a local one, because, in our settings, variables are monotonic. In problems dealing with oscillating variables, this would not be guaranteed and Algorithm 1 should be replaced by a linear search for the maximum. In this case, the complexity would be equal to $\Theta(10^3 * C_{Z3}(M))$ if the search is performed up to 3 digits of decimal precision.

[5] http://research.microsoft.com/en-us/um/redmond/projects/z3/mcmaster07.pdf.

Algorithm 1 Identifying max satisfaction of fuzzy goals.

```
1: function COMPUTEMAX(M, fG, low, high)
2:     value= (high+low)/2 |td
3:     N ← (M,"(assert (>="+fG+" "+value+"))");
4:     out←runZ3(N);
5:     if out=="UNSAT" then
6:         return COMPUTEMAX(M, fG, low, value);
7:     else
8:         x←value+0.001;
9:         N ← (M,"(assert (>="+fG+" "+x+"))");
10:        out←runZ3(N);
11:        if out=="UNSAT" then
12:            return value;
13:        else
14:            return COMPUTEMAX(M, fG, x, high);
15:        end if
16:    end if
17: end function
```

For example, Algorithm 1 can identify the maximum satisfaction of availability of a business function when an internet connection is enabled, the email sending and retrieval servers are adopted, and operations open messages/attachments, and transmit email are performed. To achieve this aim, the model given as input to the algorithm must be augmented with the following constraints:

(assert (not (not internetConnection)))
(assert (not (not retrServer)))
(assert (not (not sendServer)))
(assert (not (not openMsg)))
(assert (not (not openAtt)))
(assert (not (not transMail))).

In this case, the maximum satisfaction of availability is 0.823, and this can be achieved when email retrieval and sending patches updates, and sender/content filtering are enabled with the maximum strength. This experiment required to invoke Z3 for 18 times, and the time necessary to perform each run was negligible (<1 ms).

6.4 Trade-off

As goals can compete among each other, it is useful for software engineers to identify conflicting goals, to understand their mutual relationships, and to perform trade-off analyses.

To identify two conflicting goals, it is initially necessary to compute their maximum satisfaction value by using Algorithm 1. Then, the Z3 model must be further constrained to assert that both goals must achieve their maximum individual satisfaction level at the same time. If the problem is unsatisfiable, these goals are in conflict. For example, to verify whether goals Limited Costs and Availability are in conflict, their maximum satisfaction level (0.92 and 0.823, respectively) is identified and the Z3 model is further constrained by forcing these goals to achieve their maximum satisfaction at the same time. From identified conflicts between goals, it is also possible to identify their mutual relationships.

For example, Fig. 8 shows the maximum satisfaction of goals, which is obtained by using Algorithm 1, depending on the value of their conflicting goals. In particular, it shows the relationship between goals Availability and Limited Costs, between Confidentiality and Usable Login, between Confidentiality and Availability, and between Confidentiality, Integrity, and Usability Sending. Note that, in the second diagram in Fig. 8, Confidentiality reaches its maximum satisfaction (1.0) when a multi-factors authentication is employed. Its satisfaction is equal to 0.91 in case a fingerprint authentication is adopted, or it decreases depending on the password length in case a username and password authentication is used and fingerprint authentication is disabled.

Algorithm 2 Identifying max satisfaction of two conflicting goals.

```
1: function CONCURMAX(M, G, L, pnz)
2:     for i = 0 → G.Length − 1 do
3:         N ←add(M,"(assert (>="+G[i]+" "+L[i]+"))")
4:     end for
5:     out←runZ3(N);
6:     if out=="UNSAT" then
7:         for i = 0 → G.Length − 1 do
8:             L[i] =L[i] - pnz[i];
9:         end for
10:        return CONCURMAX(M, G, L, pnz/2 |td);
11:    else
12:        for i = 0 → G.Length − 1 do
13:            X[i]← L[i]+0.001;
14:            N ←add(M,
15:                "(assert (>="+G[i]+" "+X[i]+"))");
16:        end for
17:        out←runZ3(N);
18:        if out=="UNSAT" then
19:            return L;
20:        else
21:            for i = 0 → G.Length − 1 do
22:                L[i] =L[i]+pnz[i];
23:            end for
24:            return CONCURMAX(M, G, L, pnz/2 |td);
25:        end if
26:    end if
27: end function
```

To balance the trade-off among two or more conflicting goals, Algorithm 2 is adopted, which performs a binary search to identify the maximum individual satisfaction that two or more goals can achieve at the same time. For each goal, the search is performed in the interval $[0, M_G]$, where M_G is the maximum satisfaction value that can be obtained by each goal in isolation. The adopted algorithm receives as input a reference to the Z3 model (M), the goals to be maximised (array G), the maximum satisfaction that each

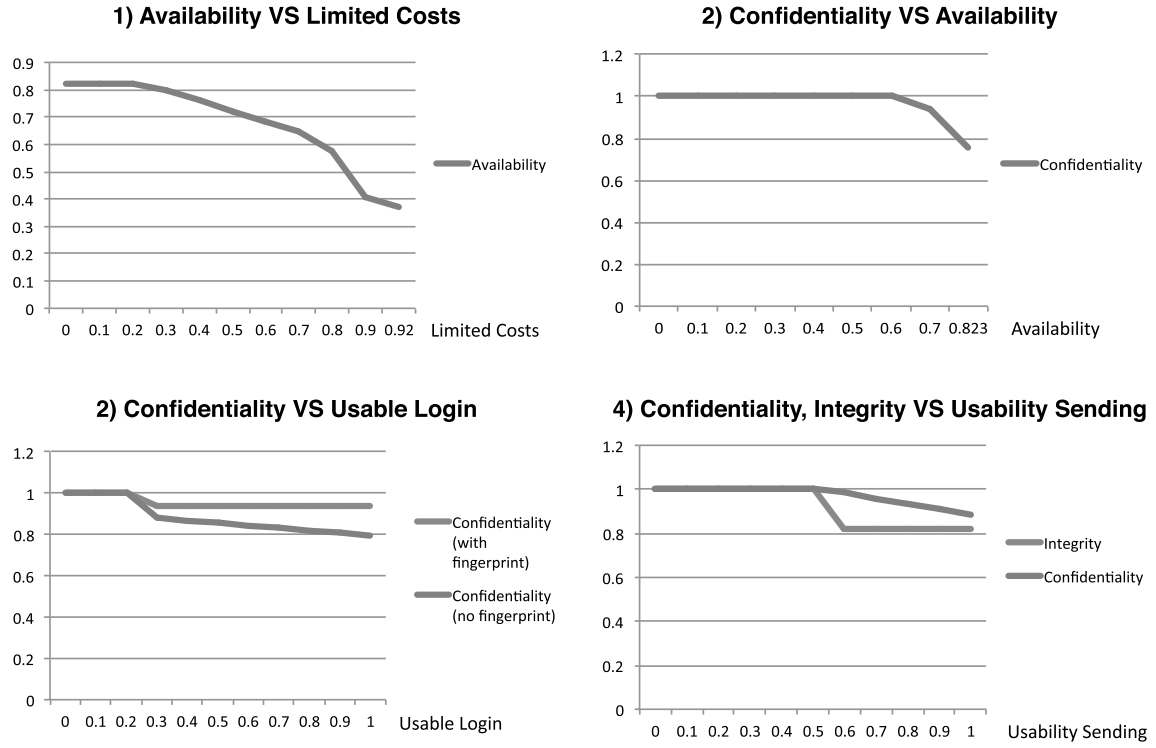

Fig. 8 Maximum satisfaction of (**1**) availability depending on limited costs, (**2**) confidentiality depending on usable login, (**3**) confidentiality depending on availability, (**4**) integrity and confidentiality depending on usability sending

goal can reach (array L), and the penalisation applied to the maximum satisfaction of each goal (array *pnz*). At the beginning, L contains the maximum level of satisfaction that the goals can reach in isolation and the penalisation is set to half of the maximum level of satisfaction that the goals can reach in isolation.

The algorithm runs Z3 to verify whether the input goals can reach a satisfaction level that is equal to their maximum minus the penalisation (lines 2–5). In case the model is unsatisfiable, the maximum satisfaction of each goal in L is decremented by the corresponding *pnz* and the algorithm is re-invoked, by receiving as input $pnz/2$ (lines 6–10). Otherwise, it is necessary to verify whether a solution can still be found in case all the maximum satisfactions L are incremented by 0.001 (line 12–17). If the problem is unsatisfiable for the new values in L, it means that the previous values are the maximum satisfaction that the input goals can achieve together (lines 18–19). Otherwise, the search continues by incrementing all the values in L by the corresponding *pnz*. Again, in the worst case, the complexity of this algorithm is $\Theta(log(10^3) * C_{Z3}(M))$, because the search is performed up to three digits of decimal precision.

Notice that Algorithm 2 does not search for the best average satisfaction between conflicting goals, but it finds the maximum satisfaction of each goal together with the other conflicting goals when they have the same priority. This does not always correspond to the maximum of their average. For example, if we consider goals A and B that can autonomously reach maximum value 0.8 and 1, respectively, our algorithm works as follows. First, it checks whether the model can be satisfied when A is equal to 0.8 and B to 1. If this is not the case, then it checks the satisfiability for values of A (0.4) and B (0.5) decremented by half of their value. Suppose that the model is satisfiable in this case, but not when A is equal to 0.401 and B is equal to 0.501. This means that 0.4 and 0.5 are the maximum values that A and B can reach together by changing them by an amount proportional to their maximum satisfaction level in isolation. In this case the average is 0.45. However, since B can individually reach satisfaction value 1, an algorithm that maximises the average would reach at least the value 0.5, corresponding to B equal to 1 and A equal to 0.

However, to compute the best average between conflicting goals, Algorithm 1 can be reused by applying it to an artificial goal that aggregates the goals of interest.

Table 2 represents the results obtained by running Algorithm 2 to identify the maximum satisfaction values that can be reached by goals Limited Costs and Availability together.

To identify the trade-off among all conflicting goals it is necessary to group them depending on their priorities.

Table 2 The trace of Algorithm 2 for goals availability and limited costs

Run	Limited costs	Availability	Result	Value
1	0.92	0.823	UNSAT	(−0.5)
2	0.42	0.323	SAT	(+0.25)
3	0.67	0.573	SAT	(+0.125)
4	0.795	0.698	UNSAT	(−0.063)
5	0.732	0.635	UNSAT	(−0.032)
6	0.7	0.603	SAT	(+0.016)
7	0.716	0.619	SAT	(+0.008)
8	0.724	0.627	SAT	(+0.004)
9	0.728	0.631	SAT	(+0.002)
10	0.73	0.633	SAT	(+0.001)
11	0.731	0.634	SAT	–

After a trade-off value is identified for the goals with the highest priority, this value is used to constrain the satisfaction of goals having priority immediately lower, and so on. For our example, goals Limited Costs and Availability have highest priority, while Confidentiality and Integrity have lower priority. The maximum satisfaction that Limited Costs and Availability can reach together is, respectively, 0.731 and 0.634. Then, these values are used to further constrain the Z3 model to compute the maximum satisfaction of Confidentiality and Integrity, which cannot be both greater than 0.697. A configuration of security controls in the solution (model) identified by Z3 is shown in Table 3. To reduce the impact on the costs, multi-factors authentication is disabled, the message content filtering and the message sender filtering also have low strength. To reduce the impact on the availability, the session duration is kept as long as possible.

6.5 Pruning

After analysing a trade-off among conflicting goals, it is possible to identify the configurations of security controls that satisfy them. On the one hand, evaluating all configurations of security controls is intractable. In our example, we include 6 security controls that can only be enabled/disabled and 10 security controls that can have different degrees of strength [six of them can have 6 degrees of strength, i.e. disabled (0.0), very low ($\leq$0.2), low ($\leq$0.4), medium ($\leq$0.6), high ($\leq$0.8), very high ($\leq$1.0), while four of them can have four degrees of strength, i.e. disabled (0.0), low ($\leq$0.4), medium ($\leq$0.6), high ($\leq$1.0)]. For this reason, the model given as input to Z3 includes $2^6 \times 6^6 \times 4^4 = 764,411,904$ configurations. To avoid evaluating all configurations it is necessary to identify security controls that must necessarily be enabled/disabled or that must be activated with a certain minimum/maximum strength to allow a predetermined group of goals to achieve a target satisfaction. All the configurations that do not include the security controls that must be enabled, or that include the security controls that must be disabled, or that do not enable some security controls with the necessary strength are pruned and can be neglected in the model to be evaluated by Z3.

Table 3 Security controls for availability = 0.634, limited costs = 0.731, confidentiality = 0.691 and integrity = 0.691

ID	Name	Value
SC1	Limit duration	Disabled (0)
SC2	Fingerprint	Enabled (*true*)
SC3	Username/password	Disabled (0)
SC4	Multi-factors authentication	Disabled (0)
SC5	Inc. Freq. Cred. renewal	Very High (0.96)
SC6	Forbid vocabulary password	Disabled (*false*)
SC7	Enable HTTPS	Enabled (*true*)
SC8	Encrypt message	Very High (1.0)
SC9	Block malicious attachment	Enabled (*true*)
SC10	Archive message	Disabled (*false*)
SC11	Confirmation	Enabled (*true*)
SC12	Ret. server patches update	Very High (1.0)
SC13	SMTP patches update	Very High (1.0)
SC14	Message sender filtering	Low (0.36)
SC15	Message content filtering	Low (0.36)
SC16	Avoid short password	N/A

Algorithm 3 Identifying the necessary security controls.

```
1: function COMPUTESC(M, cSC, fSC)
2:     reqSC = ∅
3:     for i = 0 → cSC.Length − 1 do
4:         N ←add(M, "(assert (not"+cSC[i]+"))");
5:         out←runZ3(N);
6:         if out=="UNSAT" then
7:             reqSC ← reqSC ∪ {(cSC[i])};
8:         else
9:             N ←add(M, "(assert ("+cSC[i]+"))");
10:            out←runZ3(N);
11:            if out=="UNSAT" then
12:                reqSC ← reqSC ∪ {(notcSC[i])};
13:            end if
14:        end if
15:    end for
16:    for i = 0 → fSC.Length − 1 do
17:        min←FINDMIN((M, fSC[i], 0,1));
18:        max←FINDMAX((M, fSC[i], 0,1));
19:        reqSC ← reqSC ∪ {(min ≤ fSC[i] ≤ max)};
20:    end for
21: end function
```

Algorithm 3 identifies the necessary security controls that must be enabled/disabled or that must be activated with a minimum and maximum strength. This algorithm receives as input the model M for Z3 that constrains a group of goals

to achieve a desired satisfaction. The algorithm also receives as input the security controls that can only be enabled/disabled (*cSC*) and the security controls that can be activated with different degrees of strength (*fSC*). For each security control in *cSC*, the algorithm checks whether Z3 finds a solution when the security control is disabled. If no solution is found, the considered security control is necessary and is added to the list *reqSC* of required security controls (lines 6–8). Otherwise, the algorithm checks whether Z3 finds a solution when the security control is enabled. If no solution is found, the considered security control must be disabled (lines 11–13). Analogously, for each security control in *fSC*, the algorithm identifies its minimum and maximum strength necessary for Z3 to find a solution for the model received as input (lines 16–18). The security control is finally added to the list *reqSC* of required security controls and is associated with a minimum and maximum required strength (line 19).

For example, if the satisfaction of goals Confidentiality and Usable Login are, respectively, 0.8 and 0.62, a security configuration must guarantee that security control Block Attachment is enabled, Limit Session Duration is always greater than 15 min (its strength is greater than 0.42), a Username/Password authentication is enabled with a password shorter than 10 characters (i.e. strength of Avoid Short Password is less than 0.6), credential renewal has at least a weekly frequency (i.e. strength of Increase Frequency Renewal is greater than 0.4), and Multi-factor authentication is disabled. In this case, the model will only include a subset $((1 * 2^4) * (3 * 4 * 3 * 1 * 6^3 * 4^3) = 31,850,496)$ of the original security configurations, which represents only the 4.17 % of the total number of configurations. Note that the first term in parenthesis is determined by the security controls that can only be enabled/disabled, while the second term in parenthesis depends on the security controls that can be enabled with a different strength. For example, since Increase Frequency Renewal must have strength greater than 0.4, it can only be considered as having three possible strength levels (medium, high, and very high).

7 Experimental results

This section describes the experimental results obtained by applying the analysis proposed in the previous section on the email service case study. First, we show how the security configuration varies depending on the desired protection level of assets (Sect. 7.1) and on the desired satisfaction level of security goals (Sect. 7.2). We also compare the advantages of applying our pruning technique in terms of the number of configurations that can be neglected during the asset protection and goal satisfaction analysis. Section 7.3 describes how the results of the trade-off analysis vary depending on the priorities associated with conflicting goals.

7.1 Assets protection

Table 4 shows different security configurations for different desired protection levels of the business functions (*PLBus*) and of the email messages (*PLMsg*). It shows *minimal* security configurations, i.e. for which the security controls have the minimum strength. Some security controls are always disabled because they have a negative impact on the protection level of a specific asset or they are irrelevant for the asset protection. For example, security control Limit Session Duration (SC_1) does not contribute to the satisfaction of security goal Availability and on the protection level of the related business functions, and, for this reason, it is always disabled. Only one among security controls Fingerprint (SC_2), Username/Password (SC_3) and Multi-factors authentication (SC_4) is chosen, in order to satisfy the XOR decomposition.

As the required protection level of an asset increases, some necessary security controls have higher strength. For example, the security controls that support security goal Availability, such as Retrieval Server Patches Updates (SC_{12}), SMTP Server Patches Updates (SC_{13}), Sender/Content Filtering (SC_{14}/SC_{15}), have an increasing strength as the required protection level of the business functions increases. Similarly, the security controls that directly support Confidentiality and Integrity, such as Limit Session Duration (SC_1), Increase Frequency of Credential Renewal (SC_5), Encryption ($SC8$), and Archive Message ($SC10$), must have an increasing strength as the required protection level of the email messages increases.

Note also that to identify a suitable security configuration necessary to guarantee a higher protection level of an asset the model will include a smaller number of configurations. In fact, after applying our pruning technique, a higher number of security controls is identified as necessary and, indeed, a smaller number of security configurations will be included in the model (only those containing the necessary security controls). For example, for $PLBus = 0.5$ only security control Block Malicious Attachment (SC_9) is identified as necessary. Since this security control can only be enabled or disabled, the number of configurations evaluated is reduced by 50 %. While for $PLBus \geq 0.84$ the necessary security controls that can only be enabled/disabled are Enable HTTPS (SC_7), Block Malicious Attachment (SC_9), Archive Messages (SC_{10}), and Confirmation (SC_{11}). The necessary security controls that can be enabled with different levels of strength are Increase Frequency of Credential Renewal ($SC_5 \geq 0.522$), Sender

Table 4 Security configurations for different desired protection levels of assets business function and email message (*PLBus* and *PLMsg*, respectively)

	PLBus			PLMsg		
	≤0	=0.5	≥0.84	=0.3	=0.6	≥1.0
Security controls						
SC1	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	VH (∼1.0)
SC2	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)
SC3	VL (0.1)	VL (0.1)	VH (0.9)	VL (0.1)	VH (1.0)	Dis (0.0)
SC4	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.82)	VH (1.0)
SC5	Dis (0.0)	Dis (0.0)	M (0.522)	Dis (0.0)	VH (1.0)	VH (1.0)
SC6	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)
SC7	Dis (false)	Dis (false)	En (true)	Dis (false)	Dis (false)	En (true)
SC8	Dis (0.0)	VH (1.0)	VH (1.0)	Dis (0.0)	VH (1.0)	VH (1.0)
SC9	En (true)	En (true)	En (true)	En (true)	En (true)	En (true)
SC10	Dis (false)	Dis (false)	En (true)	Dis (false)	Dis (false)	En (true)
SC11	Dis (false)	Dis (false)	En (true)	Dis (false)	Dis (false)	En (true)
SC12	Dis (0.0)	M(0.5)	VH (1.0)	Dis (0.0)	M (0.6)	VH (1.0)
SC13	Dis (0.0)	M (0.5)	VH (1.0)	Dis (0.0)	M (0.6)	M (0.6)
SC14	Dis (0.0)	H (0.7)	VH (1.0)	VL (0.2)	VH (1.0)	VH (1.0)
SC15	Dis (0.0)	H (0.7)	VH (1.0)	VL (0.2)	VL (0.2)	VL (0.2)
SC16	Dis (0.0)	Dis (0.0)	H (0.7)	Dis (0.0)	H (0.7)	Dis (0.0)
# Conf with pruning	382, 205, 952	382, 205, 952	46, 656	382, 205, 952	382, 205, 952	110, 592
(% Conf removed)	(−50 %)	(−50 %)	(−99.99 %)	(−50 %)	(−50 %)	(−99.99 %)

Note that *Dis* disabled, *VL* very low, *L* low, *M* medium, *H* high, *VH* very high, *En* enabled

Filtering ($SC_{14} \geq 0.956$), Encrypt Messages ($SC8 \geq 0.084$), Retrieval Server Patches Updates ($SC_{12} \geq 0.978$), Content Filtering ($SC_{15} \geq 0.956$), and SMTP Server Patches Updates ($SC_{13} \geq 0.978$). In this case, the size of the model will only include $(2^2) * (3 * 1 * 3 * 1 * 1 * 1 * 6^4) = 46656$, which is only the 0.0001 % of the original number of configurations.

7.2 Goals satisfaction

Table 5 represents different security configurations for different desired satisfaction levels of Integrity and Confidentiality. The table does not provide the security configurations associated with different satisfaction levels of Availability because these resemble those obtained by considering an increasing protection level of the business functions. As the required satisfaction level increases, some necessary security controls have higher strength. For example, for security goal Confidentiality, security controls Enable HTTPS (SC_7) and Encrypt Message (SC_8) have an increasing strength and Multi-factor authentication (SC_4) substitutes Username/Password authentication (SC_3), which is weaker. When a higher satisfaction of security goal Integrity is required, the security controls that have a high impact on its satisfaction are enabled, such as Block Malicious Attachment (SC_9), Archive Messages (SC_{10}), and Confirmation (SC_{11}). The security controls that are not directly related to the satisfaction of a security goal are always disabled. For example, for security goal Confidentiality, security controls Archive Messages (SC_{10}), Confirmation (SC_{11}), SMTP Server Patches Updates (SC_{13}), and Sender Filtering (SC_{14}) are always disabled.

Although some security controls do not have a direct impact on the satisfaction of a security goal, they can mitigate some vulnerabilities that can facilitate the attacks that can harm that security goal. For example, for security goal Confidentiality, security control Retrieval Server Patches Updates (SC_{13}) is applied to mitigate vulnerability V_3 that can facilitate attack IMAP/POP3 malware (A_6), which in turn can harm the messages confidentiality. For this reason, this security control has a higher strength when it is necessary to support a higher satisfaction of Confidentiality.

7.3 Trade-off

As shown in Table 6, we consider two trade-off analyses, in addition to those presented in Sect. 6.4.

For the first trade-off analysis (Case 1), we assign to goals Confidentiality and Availability the highest priority (5). We give medium priority (3) to goals Limited Costs and Integrity, and minimum priority (1) to goals Usability

Table 5 Security configurations for different desired protection levels of security goals confidentiality and integrity

	Confidentiality			Integrity		
	=0.3	=0.6	≥1.0	=0.3	=0.75	≥1.0
Security controls						
SC1	Dis (0.0)	Dis (0.0)	VH (0.98)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC2	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)
SC3	Dis (1.0)	Dis (0.0)	Dis (0.0)	VH (0.9)	VH (0.9)	VH (0.9)
SC4	Dis (0.0)	VH (1.0)	VH (1.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC5	VH (0.0)	VH (1.0)	VH (1.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC6	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)
SC7	Dis (false)	Dis (false)	En (true)	Dis (false)	Dis (false)	Dis (false)
SC8	VL (0.2)	VH (1.0)	VH (1.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC9	En (true)	En (true)	En (true)	En (true)	En (true)	En (true)
SC10	Dis (false)	Dis (false)	Dis (false)	Dis (false)	Dis (false)	En (true)
SC11	Dis (false)	Dis (false)	Dis (false)	Dis (false)	En (true)	En (true)
SC12	Dis (0.0)	VH (0.9)	VH (1.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC13	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC14	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)	Dis (0.0)
SC15	VL (0.2)	VL (0.2)	VL (0.2)	VL (0.2)	VL (0.2)	VL (0.2)
SC16	H (0.7)	Dis (0.0)	Dis (0.0)	H (0.7)	H (0.7)	H (0.7)
# Conf with pruning	382, 205, 952	382, 205, 952	165, 888	382, 205, 952	382, 205, 952	95, 551, 488
(% Conf removed)	(−50 %)	(−50 %)	(−99.98 %)	(−50 %)	(−50 %)	(−87.5 %)

Note that *Dis* Disabled, *VL* very low, *L* Low, *M* Medium, *H* High, *VH* very high, *En* enabled

Table 6 Trade-off analyses for different goals prioritisation

Goals	Case 1		Case 2	
	Priority	Max Sat	Priority	Max Sat
Availability	5	0.615	3	0.191
Limited costs	3	0.684	3	0.21
Confidentiality	5	0.751	–	0.95
Integrity	3	1.0	1	0.68
Usability sending	1	0.52	1	0.54
Usable login	1	1.0	1	1.0

Sending and Usable Login. To perform this analysis, we execute Algorithm 2 for goals Availability and Confidentiality and, after 9 runs, we can identify their trade-off satisfaction (0.615 and 0.751, respectively). Since in the example used in this paper we assume that goals Limited Costs and Integrity do not have conflicts, we identified their maximum satisfaction separately, after constraining the value of Availability and Confidentiality to their maximum trade-off value. Indeed, the maximum satisfaction for goal Limited Costs is 0.684 (after 13 runs of Z3), while the maximum satisfaction of Integrity is 1.0 (after 1 run). Finally, since there is no conflict between goals Usability Sending and Usable Login, their maximum satisfaction are identified separately and are, respectively, 0.52 and 1.0.

In the second case we set the satisfaction of goal Confidentiality to 0.95. Then, we consider Availability and Confidentiality goals as having medium priority and Integrity, Usable Login and Usability Sending goals as less important. After 13 runs of Z3 the maximum trade-off values of Availability and Limited Costs are 0.191, and 0.21, respectively. The maximum satisfaction obtained for Integrity, Usability Sending and Usable Login goals are 1.0, 0.54, and 1.0, respectively. Note that since the satisfaction of Confidentiality is higher, the satisfaction of potentially conflicting goals (e.g. Limited Costs and Availability) is reduced.

7.4 Threats to validity

As already recognised by Letier and van Lamsweerde [26], the meaning of numbers capturing partial satisfaction and partial contribution in a goal model is subjective. We try to ground the specification of the goal model on measurable phenomena, and, where not possible, we trust the judgement of the software engineer who has relevant security expertise. As shown in our example, the contribution links between operations/domain assumptions and

vulnerabilities are identified from the data published in scientific articles documenting the occurrence of vulnerabilities. At the same time, we leveraged the judgement of the software engineer to assess the impact of security controls on vulnerabilities and on the system requirements.

Furthermore, as also highlighted by Horkoff and Yu [17], the adoption of goal models to support decision-making has the potential to produce differing alternative selections, when different assumptions concerning goal concepts and value propagation are adopted. However, the results of our security trade-off analyses are not intended to provide ultimate decisions, but are considered as a heuristic for the software engineer in the selection of the security controls. The process of modelling and evaluating may be as useful as the analysis results, as the process may force the software engineer to examine the models and their domain knowledge and assumptions.

Finally, the ordering of application of some security measures is not captured by the modelling language, and this might be a problem for security requirements that need to be balanced against other non-functional objectives such as cost and performance. For example, content filtering can be achieved via several sub-filters, with different levels of efficiency and speed. Assume that Filter A has a 90 % efficiency and takes 0.05 s per message, whereas Filter B has a 40 % efficiency but takes 0.5 s per message. Together, A and B offer a 95 % efficiency while taking 0.55 s. In terms of performance (and indirectly usability), doing A before B is better than doing B before A because if the message has a problem, chances are that A will find it first, without having to resort to B.

8 Related work

UML-based approaches [21, 28, 30] have been proposed to intuitively model interactions between a system and one or more actors that could be harmful [30], verifying the impact of design choices on the satisfaction of the system security requirements [21] and security policies [28], by annotating and extending UML diagrams. Goal-based requirements analyses have been adopted extensively to demonstrate that a system meets certain security requirements. Van Lamsweerde [40] proposes anti-models to represent the objectives of potential attackers to support the elicitation of adequate security requirements able to mitigate them. Anti-models build on a goal-oriented framework for generating and resolving obstacles to requirements achievement [41]. Security goals and threats are also formalised by using temporal operators. Even though our approach uses anti-models, we do not formalise goals and threats by using a temporal logic, as we are not concerned with describing the dynamic behaviour of the system under analysis. Instead, we are interested in formalising the relationships among system goals and security concerns in order to identify the effectiveness of different configurations of security controls.

Liu et al. [27] identify potential threats from organisational relationships among the stakeholders and use the i* goal model to identify potential attackers and the roles that must be played by modelled actors for security issues. Giorgini et al. [11] enrich the Tropos goal model with the concepts of ownership, trust, and delegation and show how trust requirements can be derived and analysed. Haley et al. [14] develop arguments to demonstrate that security requirements are satisfied. This framework was extended with a risk assessment technique [10] for identifying rebuttals and mitigations of security requirements satisfaction and to support risk prioritisation. However, none of this work computes the impact of design choices and security requirements on the protection of the critical assets of the system.

Recently, Calliau and van Lamsweerde [5] proposed a goal-based probabilistic framework to perform risk analysis and assess the consequences of a risk in terms of degree of loss of satisfaction of a specific objective. The specification language [41] for goals and obstacles is extended with a probabilistic layer allowing behavioural goals to be characterised in terms of their estimated and required degree of satisfaction. The specification of goals and their obstacles has a formal semantics in terms of system behaviours, allowing probabilities to be grounded on measurable, application-specific phenomena. The severity of obstacle consequences is estimated by probability propagations through the obstacle and goal models.

Unlike our work, the focus of the approach proposed by Calliau and van Lamsweerde [5] is not on estimating the cost-effectiveness of security controls and trade-off among competing goals, but on identifying and mitigating the most critical obstacles. Moreover, our work focuses on threats, which are intentional (malicious) obstacles and are profoundly different from hazards that are unintentional obstacles. The threats refinement process must be guided by the attacker's own goals and by the target of deriving observable vulnerabilities and implementable threats. Hazards, instead, are inferred by negating a leaf goal and relating it to accidental events that can be measured probabilistically, as they are not linked to the intentions of a malicious agent.

Identifying adequate security requirements and their associated security controls is also not enough to engineer secure systems, as security engineers achieve desired levels of security while trading off competing requirements. A number of requirements-based security trade-off analyses and risk assessment approaches have been proposed [2, 18, 25]. Security Verification and security solution Trade-off

analysis—SVDT [18]—supports design trade-off and risk assessment by using Bayesian Belief Nets to structure the trade-off problem. The main limitation of SVDT is its use of a fixed set of parameters (instead of user-defined parameters) for the trade-off analysis, which cannot be context-dependent. Security controls are not configured and can only be enabled/disabled and asset criticality is not considered in prioritising security risks. Lee [25] uses Multi-Entities Bayesian Networks (MEBN) to express and analyse the causal relationships among different risk components (assets, threats, security controls). However, this approach is not automatic and does not help solving trade-off among conflicting requirements. Asnar et al. [2] provide a goal-based framework for risk assessment. However, unlike our work, this approach does not deal with uncertainty and requires to check all possible system alternatives to estimate the security risks.

Different requirements-based decision techniques [8, 15, 43] have been proposed in the literature. Feather et al. [8] propose a model to make strategic decisions in the early project phases, based on the coarse quantification of risk and its interactions with requirements and mitigations. Yen and Tiao [43] propose a formal framework for the trade-off analysis of conflicting requirements expressed in fuzzy logic. The formality of this paper in the conflict definition is embraced in our contribution for formulating different techniques for conflict resolution. AGORA [22] proposes to annotate goal models with contribution values and stakeholders preferences to help an analyst choose among the alternatives and recognise the conflicts among goals. The only approach that supports automatic decisions is the one proposed by Heaven and Letier [15]. This work leverages genetic algorithms for evaluating the impact of alternative system designs on high-level goals and for finding optimal design options among the alternatives. This builds on a previous framework [26] for specifying goals with measurable objective functions and modelling the impact that alternative system designs have on these goals. However, none of the aforementioned requirements-based decision techniques is tailored to security and does provide support to express the impact of security requirements on the assets' protection.

Given the difficulty of obtaining quantitative information, qualitative evaluation is used in a number of goal-oriented early RE approaches [12, 24]. A thorough comparison and evaluation of qualitative goal-oriented satisfaction analysis techniques has also been provided by Horkoff and Yu [17]. These techniques typically rely on the structural refinement of AND–OR goal graphs to make fine-grained distinctions among alternatives, while making a minimal differentiation between degrees of goal satisfaction and contributions of alternatives. A more precise analysis approach that combines qualitative and quantitative satisfaction levels of the actors and intentional elements (e.g. goals and tasks) was proposed by Amyot et al. [1]. Our approach, instead, assesses the impact among security concerns and the other requirements of the system through fuzzy inference rules and encodes them into a specification given in input to an SMT solver, which makes possible to automatically support our trade-off analyses.

9 Conclusions

This paper proposed a requirements-driven approach to automate security trade-off analysis. First, the approach enables security engineers to investigate which security configuration can provide adequate protection of assets and satisfaction of security goals. Second, software engineers can identify and analyse trade-offs among conflicting goals. To deal with a large analysis space, we proposed a pruning algorithm, which helps identify the security controls that are necessary to guarantee a certain protection/satisfaction level of assets/security goals. Our approach is based on the assumption that numerical evaluations of security concerns can be provided, and experimental results suggest that our analysis leads to the appropriate level of security. For example, to guarantee a higher level of protection of certain assets, it is necessary to employ more effective security controls, which better address existing vulnerabilities. On the other hand, we also show how our pruning algorithm provides a significant reduction in the number of configurations that must be evaluated during the analysis. This approach is not only useful to engineer secure systems, but may also help identify situations where none of the possible configurations of security controls guarantees adequate protection of assets.

We plan to use this approach to analyse other non-security requirements models characterised by vagueness and uncertainty. Moreover, the usability of the approach will be further validated with real stakeholders (security engineers). We will also extend our modelling language by using temporal operators to formalise goals and security concerns. This would allow us considering different orderings of application of security controls, which can have a different impact on non-functional objectives, such as cost and performance. Finally, we recognise that tuning the requirements model is a critical activity that relies on human judgement and can affect the analysis results. Therefore, we are considering the use of machine learning techniques to better understand and determine impact relationships among requirements and security concerns, in order to improve the reliability of the analysis.

Appendix: Strength levels of security controls

Figure 9 and Table 7 describe the meaning of different strength levels that can be assigned to those security controls that cannot just be enabled/disabled. We assume that a security control with strength level equal to 0 is disabled. SC_1 aims to limit the duration of a session related to the usage of the email service; increasing strength levels are associated with progressively decreasing session lengths, which are expressed in minutes (min). SC_4 aims to apply multi-factors authentication, and its strength level depends on the number of authentication factors adopted. Increasing strength levels are associated with an increasing number of factors. SC_5 aims to increase the frequency of credential renewal; increasing strength levels are associated with progressively increasing frequencies, which are expressed in days. SC_8 aims to encrypt transmitted email messages; increasing strength levels are associated with increasing encryption key lengths. SC_{12} and SC_{13} aim to apply patch updates to the retrieval and SMTP email server, respectively. Increasing strength levels are associated with progressively increasing frequencies of patch updates, which are expressed in days. SC_{14} and SC_{15} aim to filter incoming email messages depending on its sender and content, respectively. When a low strength level is applied (0.3), suspicious messages, i.e. those matching the filtering conditions, are only required to be reviewed by its recipients, without being blocked. When a medium strength level is applied (0.6), suspicious messages are put in quarantine. While when a high strength level is assigned (1.0), suspicious messages are blocked without requiring the recipient authorisation. Finally, SC_{16} aims to avoid short password; increasing strength levels are associated with progressively increasing password lengths.

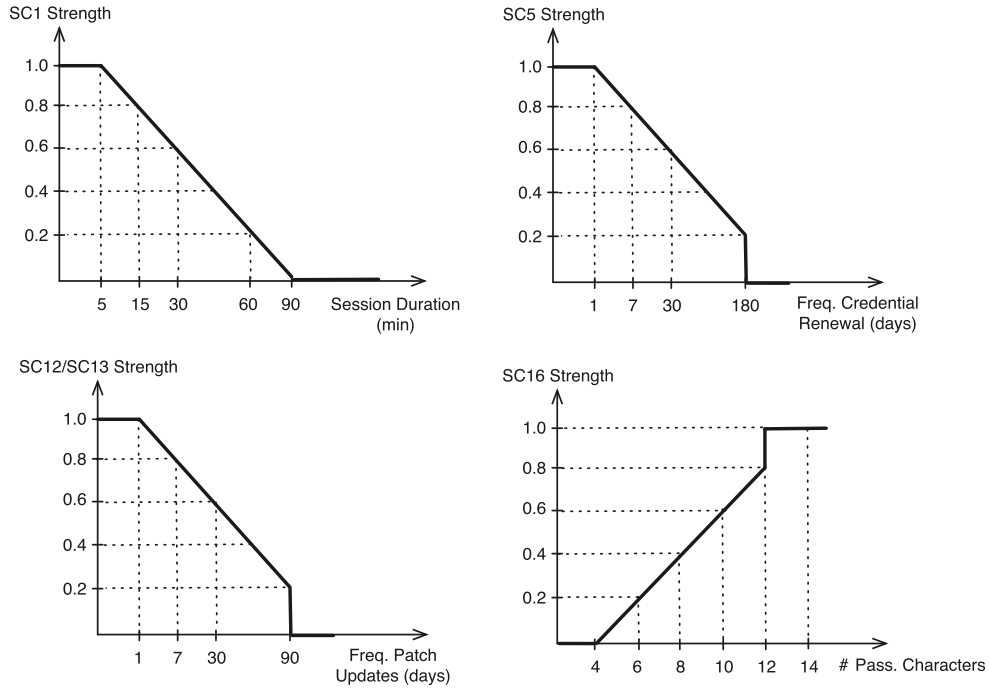

Fig. 9 Meaning of strength level of security controls SC1, SC5, SC12, SC13, and SC16

Table 7 Meaning of strength levels (sl) of security controls SC4, SC8, SC14, and SC15

Security controls	$sl = 0.3$	$sl = 0.6$	$sl = 1.0$
SC4	# Auth factors = 1	# Auth factors = 2	# Auth factors ≥ 3
SC8	Key length = 128 bits	Key length = 192 bits	Key length = 256 bits
SC14	Review suspicious msg	Quarantine suspicious msg	Block suspicious msg
SC15	Review suspicious msg	Quarantine suspicious msg	Block suspicious msg

References

1. Amyot D, Ghanavati S, Horkoff J, Mussbacher G, Peyton L, Yu ESK (2010) Evaluating goal models within the goal-oriented requirement language. Int J Intell Syst 25(8):841–877
2. Asnar Y, Giorgini P, Mylopoulos J (2011) Goal-driven risk assessment in requirements engineering. Requir Eng 16(2): 101–116
3. Barone D, Jiang L, Amyot D, Mylopoulos J (2011) Reasoning with key performance indicators. In: Proceedings of the 4th IFIP WG 8.1 working conference on the practice of enterprise modeling, Springer, Berlin, pp 82–96
4. Boehm B, Bose P, Horowitz E, Lee MJ (1994) Software requirements as negotiated win conditions. In: Proceeding of the 1st international requirements engineering conference, pp 74–83
5. Cailliau A, van Lamsweerde A (2013) Assessing requirements-related risks through probabilistic goals and obstacles. Requir Eng 18(2):129–146
6. De Moura L, Bjørner N (2008) Z3: an efficient SMT solver. In: Proceedings of the 14th international conference on tools and algorithms for the construction and analysis of systems, pp 337–340
7. Elahi G, Yu ESK (2007) A goal oriented approach for modeling and analyzing security trade-offs. In: Proceedings of the 26th international conference on conceptual modeling. Springer, Berlin, pp 375–390
8. Feather MS, Cornford SL, Hicks KA, Kiper JD, Menzies T (2008) A broad, quantitative model for making early requirements decisions. IEEE Softw 25(2):49–56
9. Firesmith D (2004) Specifying reusable security requirements. J Object Technol 3(1):61–75
10. Franqueira VNL, Tun TT, Yu Y, Wieringa R, Nuseibeh B (2011) Risk and argument: a risk-based argumentation method for practical security. In: Proceedings of the 19th international requirements engineering conference, pp 239–248
11. Giorgini P, Massacci F, Mylopoulos J, Zannone N (2005) Modeling security requirements through ownership, permission and delegation. In: Proceedings of the 13th international requirements engineering conference. IEEE Computer Society, pp 167–176
12. Giorgini P, Mylopoulos J, Nicchiarelli E, Sebastiani R (2003) Formal reasoning techniques for goal models. In: Spaccapietra S, March S, Aberer K (eds) Journal on data semantics I. Lecture notes in computer science. Springer, Heidelberg, pp 1–20
13. Glinz M (2007) On non-functional requirements. In: Proceedings of the 15th international requirements engineering conference. IEEE Computer Society, pp 21–26
14. Haley CB, Laney RC, Moffett JD, Nuseibeh B (2008) Security requirements engineering: a framework for representation and analysis. IEEE Trans Softw Eng 34(1):133–153
15. Heaven W, Letier E (2011) Simulating and optimising design decisions in quantitative goal models. In: Proceedings of the 19th international requirements engineering conference, pp 79–88
16. Hoffman S (2012) Kaspersky: malware attachments up 50 percent. http://channelnomics.com/2012/08/24/kaspersky-malicious-attachments-50-percent/
17. Horkoff J, Yu ESK (2013) Comparison and evaluation of goal-oriented satisfaction analysis techniques. Requir Eng 18(3):199–222
18. Houmb S, Georg G, Jürjens J, France R (2007) An integrated security verification and security solution design trade-off analysis approach. In: Integrating security and software engineering: advances and future visions, pp 190–219
19. In HP, Olson D (2004) Requirements negotiation using multi-criteria preference analysis. J Univ Comput Sci 10(4):306–325
20. ISO/IEC 13335–1:2004: Information Technology (2008) Security techniques—management of information and communications technology security—part 1: concepts and models for information and communications technology security management. http://www.iso.org/iso/catalogue_detail.htm?csnumber=39066
21. Jürjens J (2002) UMLsec: extending UML for secure systems development. In: Proceedings of the 5th international conference on the unified modeling language, pp 412–425
22. Kaiya H, Horai H, Saeki M (2002) AGORA: attributed goal-oriented requirements analysis method. In: Proceedings of the 20th international requirements engineering conference
23. Karlsson J, Ryan K (1997) A cost-value approach for prioritizing requirements. IEEE Softw 14(5):67–74
24. Lawrence C, Nixon BA, Mylopoulos J (1999) Non-functional requirements in software engineering. Kluwer, Dordrecht
25. Lee S (2011) Probabilistic risk assessment for security requirements: a preliminary study. In: Proceedings of the 5th international conference on secure software integration and reliability improvement. IEEE Computer Society, pp 11–20
26. Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. In: Proceedings of the international symposium on foundation of software engineering, pp 53–62
27. Liu L, Yu ESK, Mylopoulos J (2003) Security and privacy requirements analysis within a social setting. In: Proceedings of the 11th international requirements engineering conference. IEEE Computer Society, pp 151–161
28. Lodderstedt T, Basin DA, Doser J (2002) SecureUML: a UML-based modeling language for model-driven security. In: Proceedings of the 5th international conference on the unified modeling language, pp 426–441
29. Łukasiewicz J (1970) Selected works by Jan Łukasiewicz, chap. on three-valued logic. North-Holland, Amsterdam
30. McDermott JP, Fox C (1999) Using abuse case models for security requirements analysis. In: Proceedings of the 15th annual computer security applications conference. IEEE Computer Society, pp 55–64
31. Messaging Anti-Abuse Working Group (MAAWG): Email Metrics Program: The Network Operators' Perspective, Report 15—first, second and third quarter 2011 (2012). http://www.maawg.org/sites/maawg/files/news/MAAWG_2011_Q1Q2Q3_Metrics_Report_15.pdf
32. Mills E (2010) The unvarnished truth about unsecured Wi-Fi. http://news.cnet.com/8301-27080_3-20021188-245.html
33. Mouratidis H, Giorgini P (2007) Secure tropos: a security-oriented extension of the tropos methodology. Int J Softw Eng Knowl Eng 17(2):285–309
34. Nieuwenhuis R, Oliveras A (2006) On SAT modulo theories and optimization problems. In: Proceedings of the 9th international conference on theory and applications of satisfiability testing, pp 156–169
35. Pfleeger CP, Pfleeger SL (2003) Security in computing. Prentice Hall Professional, Englewood Cliffs
36. Salehie M, Pasquale L, Inah O, Ali R, Nuseibeh B (2012) Requirements-driven adaptive security: protecting variable assets at runtime. In: Proceedings of the 20th international requirements engineering conference. IEEE Computer Society, pp 111–120
37. Stoneburner G, Goguen A, Feringa A (2002) Risk management guide for information technology systems. Nist Spec Publ 800(30):800–830
38. Tracy M, Jansen W, Bisker S (2002) Guidelines on electronic mail security. NIST Special Publication, Gaithersburg, pp 45–800
39. US General Services Administration: Email as a Service (EaaS) Blanket Purchase Agreement (BPA) Requirements Document. (2013). www.gsa.gov/portal/content/112223

40. van Lamsweerde A (2004) Elaborating security requirements by construction of intentional anti-models. In: Proceedings of the 26th international conference on software engineering. IEEE Computer Society, pp 148–157
41. van Lamsweerde A (2009) Requirements engineering—from system goals to UML models to software specifications. Wiley, London
42. Wunder J, Halbardier A, Waltermire D (2011) Specification for asset identification 1.1. Tech. Rep. 7693, NIST
43. Yen J, Tiao W (1997) A systematic tradeoff analysis for conflicting imprecise requirements. In: Proceedings of the 3rd international symposium on requirements engineering, pp 87–96
44. Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353