Analyzing Attack Strategies through Anti-Goal Refinement

Tong Li[1], Jennifer Horkoff[2], Elda Paja[1], Kristian Beckers[3], and John Mylopoulos[1]

[1] University of Trento, Trento, Italy
{tong.li,paja,jm}@unitn.it
[2] City University London, London, UK
horkoff@city.ac.uk
[3] Technische Universität München, Garching b. München, Germany
beckersk@in.tum.de

Abstract. Analyzing security from an attacker's perspective has been accepted as an effective approach for dealing with security requirements for complex systems. However, there is no systematic approach for constructing attack scenarios. As a result, the completeness of the derived attack scenarios is subject to the expertise of analysts. In this paper, we propose a systematic process for identifying attack scenarios to support security analysis, founded on anti-goal refinement. In particular, we examine three real attack scenarios in order to understand attack strategies that have been applied in reality. Based on our examination, we propose a comprehensive anti-goal refinement framework, which consists of five anti-goal refinement patterns and an analysis process for using the patterns as part of security design. Finally, we evaluate the proposed anti-goal refinement framework by applying it to a credit card theft scenario.

1 Introduction

Due to ever-increasing complexity, today's systems contain more and more security vulnerabilities, resulting in a broader range of attacks. According to data from the Common Vulnerability Enumeration (CVE)[1], 9625 vulnerabilities were added to the CVE database in 2014, a nearly 30 percent increase from 2013 (7440 vulnerabilities). Given the fast rate of emerging security vulnerabilities, it is challenging to get a comprehensive understanding of the consequences of each vulnerability, not to mention the combined consequences of multiple vulnerabilities, which may lead to multistage attacks. Taking an attacker's perspective to analyze potential security breaches has been advocated as an effective approach, as it sheds light on which vulnerabilities need to be examined, and thus avoids inspecting the full space of vulnerabilities.

Techniques, such as attack trees [1] and misuse cases [2], have been proposed to describe attack scenarios (i.e., steps that attackers use to perform their at-

[1] https://cve.mitre.org

tacks) and to guide security analysis. However, when building attack scenarios from an attacker's viewpoint, there are no specific guidelines to follow [3]. As such, different security analysts can create different, subjective attacker models, and the completeness of the derived attack scenarios is subject to the expertise of the analysts. Threat analysis approaches (e.g., STRIDE [4]) elicit and tackle threats to different parts of systems, but do not capture the attacker's intentions behind each threat and miss the connections between threats. As such these approaches are not well suited to analyze complicated multistage attacks.

Further work has explicitly captured the rationale behind attacker actions using anti-goals [5, 6, 7]. Such approaches capture not only the space of possible attacks, but an attacker's strategy, including alternative plans and combing multiple steps to achieve a malicious goal. For example, to disclose a data asset, one attack strategy can be finding out all software applications that process the data and then hacking the applications to disclose the data, or directly hacking the hardware that stores the data. By systematically developing an attack strategy against a particular scenario, analysts can effectively identify attack scenarios, and then provide corresponding countermeasures.

Our previous work captures an attacker's malicious intentions as anti-goals as part of a holistic security requirements analysis framework [6, 7]. The approach takes into account security issues in various abstraction layers by using a three-layer, goal-oriented requirements model [8]. By iteratively refining root anti-goals into operationalizable anti-goals, we can create an attack strategy that implies a space of attack scenarios, from which related security controls can eventually be derived (Fig. 1). However, we have not studied in-depth how anti-goals can be refined systematically, in order to generate a comprehensive attack strategy. Thus, our primary goal is to *produce a framework, grounded in real evidence, to support systematic exploration of attack strategies, producing strategies which are more complete, leading to a more complete security analysis* (i.e., the highlighted part in Fig. 1). To achieve this goal, we perform the following steps:

1. perform a grounded study on three real attack scenarios [9] in order to investigate how attackers elaborate their malicious intentions in reality, from which we identify five anti-goal refinement patterns.
2. propose an anti-goal refinement framework, which systematically refines anti-goals by leveraging the identified anti-goal refinement patterns, and eventually reveals attack scenarios.
3. evaluate the proposed refinement framework by applying it to a different credit card theft scenario [10], the result of which shows that our framework is able to generate a comprehensive attack strategy, which not only covers the reported attack scenarios, but also reveals new attack scenarios.

The remainder of this paper is structured as follows: we present the anti-goal modeling approach that we use to analyze the attacker's strategy in Section 2. The examination of real attack scenarios is described in Section 3, based on which we propose an anti-goal refinement framework in Section 4. We evaluate the proposed framework in Section 5, and discuss related issues of the framework in Section 6. In Section 7, we compare our proposal with related work. Finally, we conclude the paper and discuss future work in Section 8.

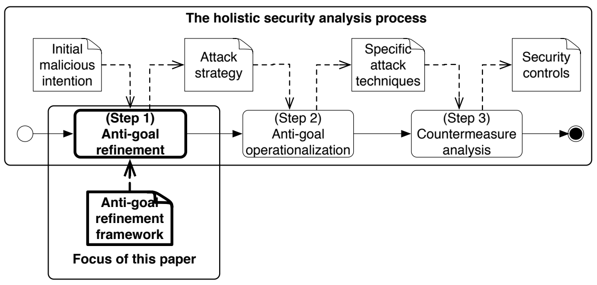

The holistic security analysis process
Initial malicious intention
Attack strategy
Specific attack techniques
Security controls
(Step 1) Anti-goal refinement
(Step 2) Anti-goal operationalization
(Step 3) Countermeasure analysis
Anti-goal refinement framework
Focus of this paper

Fig. 1: Research overview

2 Anti-Goal Modeling

Anti-goals were first used by van Lamsweerde to model an attacker's malicious intentions related to system assets [5]. An anti-goal model presents how the attacker's abstract anti-goals are refined to terminal anti-goals (that are realizable by attackers), which captures the attacker's strategies. By constructing anti-goal models, analysts can effectively identify system threats and use this knowledge to design secure systems.

In this paper, we leverage the anti-goal approach to analyze system threats in order to provide corresponding countermeasures. In particular, we focus on analyzing anti-goal refinements from an attacker's viewpoint by studying real attack scenarios. To this end, we adopt three concepts from a recent goal model approach, *Techne* [11], for building anti-goal models: *Goal*, *Task*, and *Domain Assumption*. A *goal* captures attacker intentions (i.e., anti-goals); a *task* presents detailed attack actions that are performed by attackers; a *domain assumption* describes an indicative property that is relevant to the system. Detailed examples of anti-goal models are presented in Fig. 2. It is worth noting that we do not introduce a new notion *anti-goal* but use the ordinary notion *goal* to model anti-goals, as anti-goals are simply goals from the attacker's viewpoint.

3 Attack Scenario Examination

We examine three real attack scenarios in order to understand attack strategies that have been applied in reality. In particular, we apply the anti-goal modeling to real attack scenarios, and then investigate the rationale behind each anti-goal refinement within the modeled scenarios, and finally extract five anti-goal refinement patterns. In the remainder of this section, we first briefly introduce the real attack scenarios that we examine, and then present our examination on these scenarios in detail.

Sample attack scenarios. To reveal sophisticated attack strategies from the examination, we define three criteria for selecting the attack scenarios to be examined. Firstly, the attacks should cover a wide spectrum of attack techniques, from social engineering to software/hardware hacking. Secondly, we look for multistage attacks that consist of a sequence of steps, rather than an atomic attack that is done by a single exploit. Thirdly, the description of the attacks should present not only attack actions performed by attackers, but also the intentions motivating the actions.

According to the above criteria, we select three attack scenarios that are documented in Mitnick's book [9, Ch. 11]. Each of these attack scenarios involves both social and technical issues, and consists of multiple attack steps. In this case, the author narrates the entire attack process in detail, shedding light on both the why and how for each attack step. The general problems and contexts of these attack scenarios are as follows:

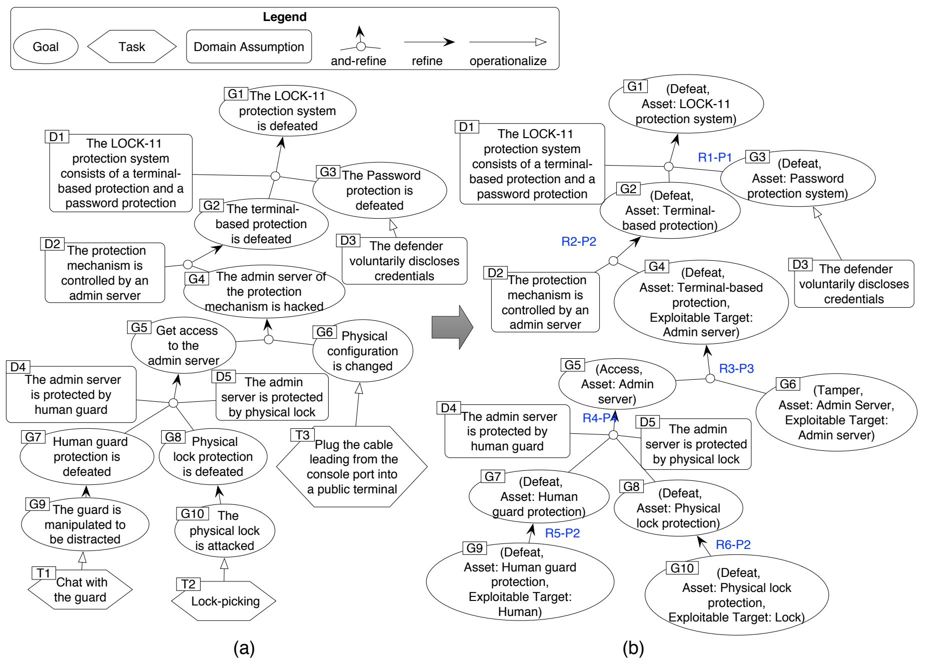
– *Easy Money*: Two attackers aim to defeat a security product that is designed for access control in order to get prize money. The product applies terminal-based security technique, which identifies system users based in part on the particular computer terminal being used.
– *Dictionary as an Attack Tool*: An external attacker intends to steal the source code of a new electronic game, which is developed by a global company. The source code is stored on an unknown server of the company.
– *The Speedy Download*: An external attacker wants to obtain some confidential files of an accounting firm in order to affect the stock price of publicly traded companies. The confidential files are stored on the workstation, which can only be accessed from the company's local network.

In this paper, we illustrate the examination process using the "Easy Money" scenario. The complete set of examination results can be found online[2].

Construct initial anti-goal models. We first build initial anti-goal models according to the textual description of the attack scenarios. The construction of anti-goal models is carried out by combining top-down and bottom-up analysis. The content of each node is described in natural language, using a particular part of the scenario description. Fig. 2(a) presents the entire anti-goal model that is built from the "Easy Money" attack scenario. Note that we capture the attack actions as *tasks* so as to provide a full view of the scenario, but our analysis focuses on the anti-goal refinement rather than the anti-goal operationalization. To easily reference to the elements of the anti-goal model, we annotate each element with regard to the type of the element. In particular, G stands for *Goal*, T stands for *Task*, and D stands for *Domain Assumption*.

Characterize anti-goals. It is our goal to capture anti-goals and their refinements, such as in Fig. 2(a), in a more structured and abstract way. Thus we characterize each anti-goal with a structured description language, which is specified in Table 1 by using EBNF syntax.

[2] `http://disi.unitn.it/~li/poem15/result.pdf`



Legend
Goal
Task
Domain Assumption
and-refine
refine
operationalize
G1 The LOCK-11 protection system is defeated
D1 The LOCK-11 protection system consists of a terminal-based protection and a password protection
G3 The Password protection is defeated
G2 The terminal-based protection is defeated
D3 The defender voluntarily discloses credentials
D2 The protection mechanism is controlled by an admin server
G4 The admin server of the protection mechanism is hacked
G5 Get access to the admin server
G6 Physical configuration is changed
D4 The admin server is protected by human guard
D5 The admin server is protected by physical lock
T3 Plug the cable leading from the console port into a public terminal
G7 Human guard protection is defeated
G8 Physical lock protection is defeated
G9 The guard is manipulated to be distracted
G10 The physical lock is attacked
T1 Chat with the guard
T2 Lock-picking
(a)
G1 (Defeat, Asset: LOCK-11 protection system)
D1 The LOCK-11 protection system consists of a terminal-based protection and a password protection
R1-P1
G3 (Defeat, Asset: Password protection system)
G2 (Defeat, Asset: Terminal-based protection)
R2-P2
D3 The defender voluntarily discloses credentials
D2 The protection mechanism is controlled by an admin server
G4 (Defeat, Asset: Terminal-based protection, Exploitable Target: Admin server)
G5 (Access, Asset: Admin server)
R3-P3
G6 (Tamper, Asset: Admin Server, Exploitable Target: Admin server)
D4 The admin server is protected by human guard
R4-P4
D5 The admin server is protected by physical lock
G7 (Defeat, Asset: Human guard protection)
G8 (Defeat, Asset: Physical lock protection)
R5-P2
R6-P2
G9 (Defeat, Asset: Human guard protection, Exploitable Target: Human)
G10 (Defeat, Asset: Physical lock protection, Exploitable Target: Lock)
(b)

Remark: the label *Rx-Py* means the refinement *Rx* falls into the pattern *Py* (Table 2)
Fig. 2: Anti-goal models that are built from the “Easy Money” attack scenario

Table 1: The EBNF syntax of the structured description language

Rule_1:	<anti-goal> ::= <threat>, <attribute-description>+
Rule_2:	<threat> ::= ‘tamper’ | ‘disclose’ | ‘spoof’ | ‘repudiate’ | ‘deny’ | ‘reach’ |‘access’ |‘control’ | ‘defeat’
Rule_3:	<attribute-description> ::= <attribute>, <descriptor>
Rule_4:	<attribute> ::= ‘asset’ | ‘exploitable target’ | ‘interval’

Each anti-goal is characterized by one threat and one or several attributes (*Rule_1*). A *threat* presents an undesired state that an attacker wants to impose on the targeting system. We classify threats using an existing, established threat categorization, STRIDE, provided by Microsoft [4]. STRIDE is an acronym that stands for six threat categories: *Spoofing*, *Tampering*, *Repudiation*, *Information disclosure*, *Denial of service*, and *Elevation of privilege*. These threat categories provide comprehensive coverage of security threats and have been adopted and investigated in both academia and industry [4, 12]. Note that we describe the threat categories in terms of their essential actions rather than the full description (*Rule_2*), as the threat actions are more succinct and intuitive when combined with other attributes. For the threat *Elevation of privilege*, we specifically consider three threat actions *reach*, *access*, and *control*, each of which implies a particular level of privilege. When comparing the available categories in STRIDE

Table 2: Summarized refinement patterns

No.	Pattern Name	Pattern Influences	Occurrence
P1	Asset-based refinement	Modify asset	2
P2	Target-based refinement	Add exploitable target	7
P3	Threat-based refinement	Modify asset; modify threat; remove exploitable target	10
P4	Protection-based refinement	Modify threat; modify asset; remove exploitable target	4
P5	Interval-based refinement	Modify interval	2

to the anti-goals collected from the real cases, we find the need to add an additional threat category, specifically, "defeated security mechanism" which captures the attacker intention to break system protections.

Moreover, we characterize anti-goals with three other attributes (*Rule_4*): an *asset* is anything of value to stakeholders, it is normally the object of a threat; an *exploitable target* is a component of a system, which involves assets and has vulnerabilities that are exploitable by attackers; an *interval* represents the time period, during which attackers carry out attacks. Note that values of these attributes are described in text (*Rule_3*). By using the structured description language, we characterize the anti-goals in the initial anti-goal model, resulting in a characterized model as shown in Fig. 2(b).

Identify refinement patterns. Once the characterized anti-goal model is obtained, we investigate each refinement relation in detail, on the basis of which we can identify refinement patterns.

We first investigate the influences of refinement relations on the refined anti-goals, i.e., what have been changed from the refined anti-goals to their sub-goals. For example, as shown in Fig. 2(b), the influence of refinement *R1* is that the asset of the anti-goals *G2* and *G3* have been modified from their parent goal *G1*. After performing such analysis on all 25 refinement relations in the three attack scenarios, we cluster refinement relations with similar influences, based on which we summarize five refinement patterns. Table 2 presents the identified refinement patterns, as well as their influences and number of occurrence in the three attack scenarios. Examples of the application of the refinement patterns can be found in Fig. 2(b), where each refinement relation is annotated with its corresponding refinement pattern.

4 An Anti-Goal Refinement Framework

The extracted five anti-goal refinement patterns shed light on various ways to refine an anti-goal, based on which we propose an anti-goal refinement framework. The framework efficiently leverages the proposed refinement patterns to refine an attacker's high-level anti-goals and to generate a comprehensive attack strategy, the analysis process of which is shown in Fig. 3.

Each of these steps makes use of one particular refinement pattern, and the detailed guidelines for performing these steps are presented below (the steps are illustrated as part of the evaluation in Section 5) It is worth noting that we describe the anti-goal refinement framework from an attacker's perspective to clearly show the rationale of the strategy, but the corresponding analysis is actually performed by security analysts with a complete set of system information in order to discover all potential attack scenarios. In particular, the description of each analysis step focuses on addressing the following issues:

– *Rationale.* We first describe the rationale of each analysis step, which explains the design of the analysis process (Fig. 3). Note that the proposed anti-goal refinement framework is a specific way to analyze attack strategy, and does not exclude other possible ways (more discussions in Section 6).
– *Input.* We then specify the inputs that are required for performing the analysis step. It is worth noting that our proposal is a general framework, which is not associated with specific models. Thus, for inputs, we only describe the types of information that are required, and all models that capture the corresponding information can be used. In Section 6, we will discuss the potential of using a three-layer requirements goal model from our previous work [8] to support the analysis.
– *Sanity check.* Our framework is intended to cover various attacks and thus provides a comprehensive security analysis. As a result, a single anti-goal can lead to a very large model. To deal with this complexity, we propose to prune the model as part of its construction, i.e., performing sanity checks after each analysis step in order to reduce the refinement space.
– *Stop criteria.* Finally, we describe the stop criteria of each analysis step.

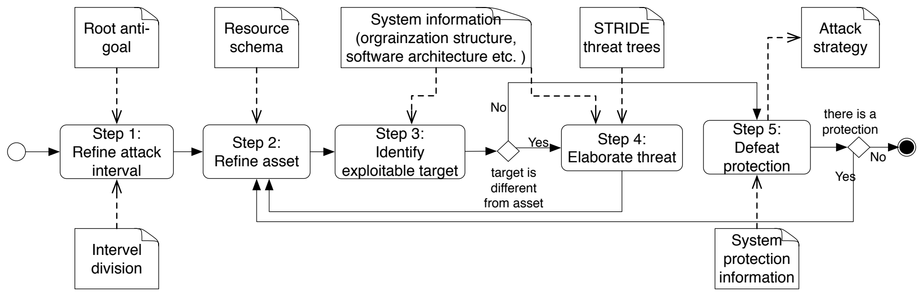

Root anti-goal
Resource schema
System information (orgrainzation structure, software architecture etc.)
STRIDE threat trees
Attack strategy
Step 1: Refine attack interval
Step 2: Refine asset
Step 3: Identify exploitable target
No
Yes
target is different from asset
Step 4: Elaborate threat
Step 5: Defeat protection
there is a protection
No
Yes
Intervel division
System protection information

Fig. 3: An analysis process of anti-goal refinement

Step 1: Refine attack interval. The system security settings can change over time, affecting an attacker's anti-goals. As the first step, an attacker applies the interval-based refinement pattern in order to concentrate on specific time intervals. Thus, this analysis step requires specific domain knowledge regarding the division of time intervals. In particular, for each interval-based refinement, the analyst should check whether the system security settings have been changed from the original interval to its sub-intervals. If the security settings remain the same, this refinement will not contribute to disclosing new attack scenarios and should be pruned. The interval refinement analysis is completed once the finest-grained intervals have been reached via refinements.

Step 2: Refine asset. Given a composite asset, it is easier for an attacker to attack a fine-grained part of the asset rather than attacking the composite asset as a whole. An attacker can leverage the asset-based refinement pattern to generate sub-goals that focus on more specific sub-assets. The asset-based analysis takes the system resource schema as input, which documents “part-of” relations between system resources. To identify system assets among system resources, we refer to the asset identification process that is specified in ISO27005:2011 [13, Annex B], which deals with both the primary assets and the supporting assets. In particular, the primary assets include *business processes and activities* and *information*; the supporting assets include *hardware*, *software*, *network*, *personnel*, *site*, and *organizations structure*. This analysis step is done when all identified assets in the resource schema are analyzed.

Step 3: Identify exploitable target. Once the an attacker has determined the assets he intends to impair, he needs to find out corresponding vulnerable system components (a.k.a. exploitable targets), by exploiting which the assets will be damaged. In particular, an asset can be involved in system components in different ways according to the type of the components, e.g., an information asset can be accessed by people, processed by software, or stored in hardware. Note that the asset and the exploitable target of an anti-goal can be the same, if the asset itself is a vulnerable system component.

We here consider the types of vulnerable system components in line with the list of supporting assets presented in ISO27005:2011 [13, Annex B.1.2]. As such, corresponding system information is required, e.g., information of system infrastructure, software architecture, and organization structures. When identifying the exploitable target, analysts should check the risk of exploiting the target, e.g., using the CORAS approach [14]. If the risk is under certain threshold, determined by the analysts, the target is assumed to be secure and is excluded from this refinement step. After using the target-based refinement pattern to identify all potential exploitable targets, this analysis step is complete.

Step 4: Elaborate threat. If an attacker aims to impose a threat to an asset by exploiting a target, which is different from the asset, then the attacker should identify new threats that he wants to impose on the exploitable target in order to successfully impose the original threat to the asset. For example, if an anti-goal is intended to disclose (*threat*) confidential files (*asset*) that are stored in a database (*exploitable target*), then it can be refined to getting access to (*new threat*) the same database (*new asset*) by using the threat-based refinement pattern.

When applying the threat-based refinement pattern, the system information and related security knowledge are required to support the threat elaboration. Specifically, we refer to 19 STRIDE threat trees as the security knowledge sources, which describe alternative ways about how a threat category can be refined to other categories. As we specify the threats of anti-goals using the STRIDE threat categories, the application of the STRIDE threat trees can be seamlessly integrated into this analysis step. In order to discover all potential attack scenarios, once we identify the new threats to the exploitable target that can lead to the original threat to the asset, we iteratively analyze the new threats to the exploitable target through the analysis step 2 and 3, i.e., we treat the exploitable target as a new asset. Such as in the aforementioned example, the newly introduced sub-goal "getting access to (*new threat*) the database (*new asset*)" concerns the database as a new asset, which was the exploitable target in the parent anti-goal.

Step 5: Defeat protection. From an attacker's perspective, security protections are obstacles to his attacks. If the attacker targets a system component which is protected by some security mechanisms, such as encryption and firewalls, then he needs to first defeat the mechanisms in order to achieve their anti-goals. According to the knowledge about system security design, the attacker can use the protection-based refinement pattern to generate anti-goals against related security protection mechanisms.

Each of the newly generated anti-goals concerns a specific protection mechanism as its asset and is intended to defeat it. Similar to the last analysis step, as long as new assets have been identified in the new anti-goals, subsequent analysis will iteratively refine assets and identify targets for the new anti-goals, i.e., going back to the analysis step 2. It is worth noting that during the anti-goal refinement, we focus on identifying which protection mechanisms need to be defeated by exploiting which targets, not answering which specific attack techniques to be used to defeat the mechanisms. Once there are no further security protections to be defeated, i.e., there are no new assets have been found, the analysis reaches an end as all potential attack scenarios have been obtained.

5 Evaluation

In order to evaluate the proposed anti-goal refinement framework, we apply it to a credit card theft scenario. In this section, we first introduce the evaluation scenario, and then illustrate the application of this framework to the scenario, finally, we evaluate the resulting anti-goal model. Due to space limitations, we only present part of the resulting model (Fig. 4) for illustrating the application of the framework, and the full version can be found online[3].

Credit card theft scenario. This scenario presents a complicated multistage attack in reality, which is documented in Skoudis's hacking book [10, Ch.12] and is different from the source of the previous three real attack scenarios. Specifically, in this scenario, there is a widgets corporation which operates more than

[3] Available at `http://disi.unitn.it/~li/poem15/evaluation_model.pdf`

200 retail stores. Each retail store communicates with the central corporate network by using a VPN, and all credit card transactions are seamlessly moved from individual stores back to the central database. Each store has several Point-of-Sale (POS) terminals, which access the local store network using wireless access points. Each store also has a store server, which processes credit card transactions and forwards the transactions back to the company server.

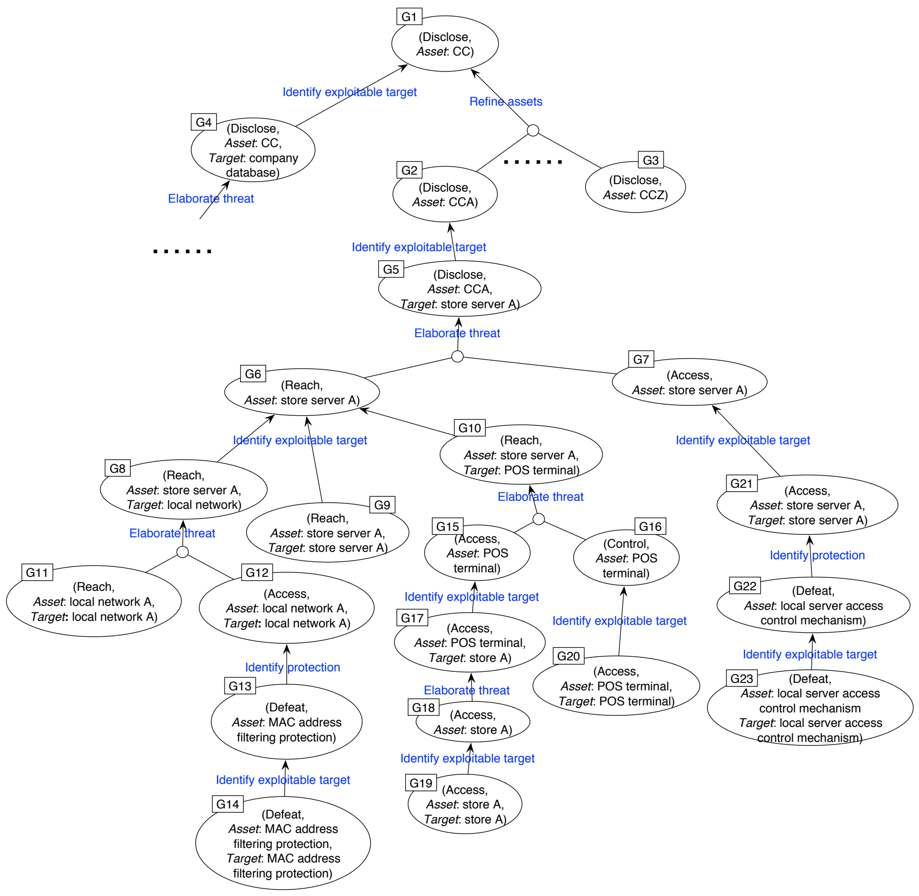

G1 (Disclose, Asset: CC)
Identify exploitable target
Refine assets
G4 (Disclose, Asset: CC, Target: company database)
G2 (Disclose, Asset: CCA)
G3 (Disclose, Asset: CCZ)
Elaborate threat
Identify exploitable target
G5 (Disclose, Asset: CCA, Target: store server A)
Elaborate threat
G6 (Reach, Asset: store server A)
G7 (Access, Asset: store server A)
Identify exploitable target
G10 (Reach, Asset: store server A, Target: POS terminal)
Identify exploitable target
G8 (Reach, Asset: store server A, Target: local network)
G9 (Reach, Asset: store server A, Target: store server A)
G21 (Access, Asset: store server A, Target: store server A)
Elaborate threat
Elaborate threat
G15 (Access, Asset: POS terminal)
G16 (Control, Asset: POS terminal)
G11 (Reach, Asset: local network A, Target: local network A)
G12 (Access, Asset: local network A, Target: local network A)
Identify protection
Identify exploitable target
Identify exploitable target
G17 (Access, Asset: POS terminal, Target: store A)
G22 (Defeat, Asset: local server access control mechanism)
Identify exploitable target
Identify protection
G13 (Defeat, Asset: MAC address filtering protection)
Elaborate threat
G20 (Access, Asset: POS terminal, Target: POS terminal)
G23 (Defeat, Asset: local server access control mechanism, Target: local server access control mechanism)
G18 (Access, Asset: store A)
Identify exploitable target
Identify exploitable target
G19 (Access, Asset: store A, Target: store A)
G14 (Defeat, Asset: MAC address filtering protection, Target: MAC address filtering protection)

Fig. 4: Application of the anti-goal refinement framework to the scenario of credit card theft (excerpt)

Applying the anti-goal framework. As a pre-step, we first process the scenario description to extract information that is required by the analysis. Specifically, we capture the attacker's high-level malicious intention, i.e., steal customer's credit card information, and model it as his root anti-goal (*G1* in Fig. 4). In addition, we capture related domain information that is required for the analysis, such as asset relations, system infrastructure etc. Having the root anti-goal and related domain information as input, we apply the proposed framework (Fig. 3) to refine the root anti-goal into operational anti-goals and thus generate a comprehensive attack strategy. We summarize this process as follows.

1. As the scenario only deals with the credit card system in a general time span and does not describe any particular time interval, we opt not to apply the first step. In other time-sensitive cases, this step would be applied.
2. We refine the asset of the root anti-goal *G1* according to the composition relations among assets. As the entire set of credit card information is composed of information of credit cards that are processed in different retailer stores, the root-goal *G1* should be and-refined to more than 200 sub-goals, each of which is intended to disclose credit cards of one particular store. Since all the retailer stores have homogeneous design and configuration, the attack scenarios about these retailer stores are the same, i.e., the more than 200 anti-goals will be refined in the same way. Thus, as shown in Fig. 4, we only focus on the first sub-goal *G2* in later analysis.
3. We identify exploitable threats to the assets. Due to the domain knowledge that the information of credit cards that are used in store A (i.e., *CCA*) is kept in the *store server A*, *G2* is refined to *G5*, targeting the *store server A*. In addition, we refine *G1* to *G4* because the entire set of credit card information as a whole is stored in the company database. Due to space limitations, we will skip the illustration of this branch in the later analysis steps, which can be found in the online full model.
4. As the asset and the exploitable target of *G5* are not the same object, we need to elaborate *G5* to identify which threats should be imposed to the *store server A* in order to *disclose* the *CCA*. According to the threat knowledge, *G5* is and-refined to *G6* and *G7*, which are intended to reach the server and to access into the server, respectively.
5. As *G6* and *G7* introduce a new asset *store server A*, iterative analysis should be performed to these two goals from the secondly step until the analysis no longer introduces new assets. As shown in Fig. 4, the longest refinement paths {*G6, G10, G15, G17, G18, G19*} iterates three times.
6. After identifying an exploitable target, we check whether there are security mechanisms that have been applied to protect it. Take *G21* for example, it is refined to *G22* as there is an *access control mechanism* that has been applied to protect the *store server A*. Because *G22* promotes the *access control mechanism* as a new asset, another round of analysis starts from the second step.
7. Performing the iterative analysis on *G13* and *G22* results in the anti-goals *G14* and *G23*, respectively. As the iterative analysis has not introduced new assets, the anti-goal refinement reaches an end.

Results and analysis. The evaluation finally results in an anti-goal model with 46 anti-goals and 48 refinements, which takes one author 5 hours to build. By analyzing the and/or refinement operators, we identify that the final model implies a total of 11 alternative attack scenarios.

To assess the effectiveness of the proposed framework, we carry out a bottom-up analysis to check whether the documented credit card theft scenarios can be covered by the attack scenarios that are identified by our approach. Specifically, we identify all specific attack actions that have been performed by the attacker in the scenario description, including both successful and failed attack actions. Then, we check whether the intention of these actions can be linked to the leaf goals of the anti-goal model. Our examination turns out that all the attack actions documented in the scenario can be linked to the leaf goals, i.e., the identified attack scenarios completely cover the real attack scenarios. Specifically, 6 out of the 11 potential attack scenarios are reported in the scenario description (2 successful, 4 failed), while the other 5 potential attack scenarios are not mentioned in the scenario description, revealing previously unconsidered attacks.

6 Discussion

Diversity of anti-goal refinement. Our proposal is based on the examination of three real attack scenarios that come from the same book [9], and examining other scenarios from different sources may have different results. In addition, the examination process reflects our specific interpretation of attack strategies, and the outcome of the examination can vary from person to person. As such, the proposed anti-goal refinement framework is a particular way of refining anti-goals, which has been evaluated as effective to analyze attacker's malicious intentions in the credit card theft scenario. We believe this method can also be effective when applied to further cases, and we will continue to evaluate this method as part of our larger security analysis framework.

The role of anti-goal refinement in security analysis. Our proposal, in this paper, serves as an important step in our holistic security analysis framework [6, 7]. The resulting anti-goal model represents a comprehensive attack strategy, which discloses various potential attack scenarios. In particular, according to a comprehensive attack pattern repository (CAPEC)[4], each leaf anti-goal in the resulting model will be operationalized into concrete attack actions that use specific attack techniques and tools. Failing to operationalize a leaf anti-goal implies the corresponding attack scenario is unrealizable. Then, for each realizable attack scenario, we will assess its risk in terms of likelihood and severity (such information is available in the CAPEC attack patterns). Finally, regarding the risk of the realizable attack scenarios, we can design corresponding security controls to prevent or mitigate concrete attack actions of the attack scenarios.

In addition, to deeply integrate the anti-goal refinement framework into the holistic security requirements analysis framework, we plan to leverage the three-layer goal model [8] that is used in the holistic security framework to support the anti-goal refinement analysis. In particular, the three-layer goal model captures various system components in different abstraction layers, as well as the connections between the components. Moreover, the security protection information is

[4] `https://capec.mitre.org/`

also captured by the security goals and security tasks within the three-layer goal model. Consequently, with only minor extensions, the three-layer goal model is able to provide the related information that is required by the anti-goal refinement framework.

Threats to validity of the evaluation. A major threat to the conclusion validity is that the evaluation is only performed to one single scenario. Although the scenario is relatively complicated and involves various security issues, in the future, we need to evaluate our approach with more real attack scenarios. To this end, an efficient prototype tool is required to support the analysis process. In addition, the entire evaluation is performed by only one author, imposing a threat to the external validity. Subsequent work will use multiple and varied evaluators to apply the method.

Scalability. Our framework is designed to provide a comprehensive anti-goal refinement analysis, i.e., covering all potential attack scenarios. As such, the scalability issues are raised due to the large refinement space. To deal with this problem, we have proposed sanity checks for each analysis step in Section 4, in order to prune the model as part of its construction.

In addition to the checks, we also observe a further phenomenon which helps to mitigate scalability. During the anti-goal refinement, it is possible to obtain repeated anti-goals, i.e., different anti-goals can be refined into the same anti-goals. This is because one anti-goal can have various influences, e.g., accessing to the server of a retailer store not only discloses credit card information stored in that server but also enables the attacker to penetrate the company internal network. As such, it is important to detect and merge the repeated anti-goals during the anti-goal analysis as new anti-goals are generated. Otherwise, the repeated anti-goals will be further refined separately and the size of the model can grow exponentially. Note that merging repeated anti-goals is performed by adding all refinement links of these anti-goals to one anti-goal and removing other anti-goals. As such, the derived model is not a tree but a directed acyclic graph (DAG).

Finally, we plan to develop a modeling and analysis tool, extending our existing tool MUSER [15] in order to (semi-)automate anti-goal refinement. In particular, we are defining formal inference rules for the five anti-goal refinement patterns that are proposed in this paper. On top of these inference rules, the tool will further implement the analysis process of anti-goal refinement (shown in Fig. 3) in order to support the automation of anti-goal refinement. To guarantee the correctness of the analysis and to reduce model complexity, the tool will interact with analysts in order to support manual revision after each analysis step, allowing the analyst to, for example, perform sanity checks (see Section 4) over the refinements.

7 Related Work

In this paper, we analyze attack strategies by examining three real attack scenarios that are documented in a security textbook [9]. Apart from this book, we have found other potential security knowledge sources. Attack patterns were first proposed by Moore et al. [16] to summarize reusable attack knowledge from repeated attacks in support of system security analysis. In particular, CAPEC (Common Attack Pattern Enumeration and Classification) is a comprehensive attack pattern repository, which was first released in 2008 and has accumulated 463 attack patterns [17]. However, these attack patterns indeed describe low-level attack knowledge about how to use specific attack techniques and tools to perform a particular attack, such as "exploit user-controllable input to perform a format string injection". Thus, CAPEC attack patterns do not fit our need of analyzing high-level attack strategies in this paper, but they can be used to operationalize anti-goals and support security analysis as discussed in Section 6. Another security threat knowledge source is the STRIDE threat trees [4], which focuses on how one threat can be refined into other threats. However, these threat trees only capture a single step of threat elaboration and cannot account for multistage attacks. As a result, we do not examine these threat trees for analyzing attack strategies, but use them as the security knowledge source to support the threat elaboration analysis in our anti-goal refinement framework.

Anti-goals were first proposed by Lamsweerde to capture attacker intentions and to construct anti models in order to provide security requirements for potential threats [5]. To refine anti-goals, apart from the ad-hoc way (asking why and how questions), the author proposed to use formal goal refinement patterns, which were designed for refining requirement goals [18]. However, the nature of attack analysis requires that the anti-goal refinement should cover all potential attack scenarios in order to provide comprehensive and reliable security design, which cannot be supported by the typical goal refinement patterns. In contrast, our anti-goal refinement framework reflects attack strategies investigated from real attacks and is designed to reveal all potential attack scenarios.

Attack trees are a typical way of representing attack scenarios. Although there is no unique way of creating attack trees, different researchers have proposed their own ways to build attack trees, which are related to our anti-goal refinement framework. Morais et al. advocate to first build the overall attack, and then identify the violated security properties and the security mechanisms to be exploited, respectively, and finally model the concrete attack actions [3]. Paul proposes a layer-per-layer approach to generate skeletons of attack trees using information comes from system architecture, risk assessment study, and related security knowledge base [19]. However, these approaches do not capture the attacker's malicious intentions and cannot analyze attack strategies as we define them.

Apart from the attack trees, attack graphs are another way of representing attack scenarios. An attack graph shows all paths through a system that end in a state where an attacker achieves his malicious intentions. Phillips and Swiler first use attack graphs to analyze network security [20]. Due to the homogeneous settings of machines in the network, the states of machines (nodes in the attack graph) and the atomic attacks to machines (transitions in the attack graph) are able to be enumerated. As such, it is possible to fully automate the generation of attack graphs using a comparatively simple attack strategy. Take the approach of Sheyener et al., for example: an attacker starts from a machine with the root permission, he then iteratively detects a new machine in the network, logs into that machine, and gets the root permission of that machine until reaching his target machine [21]. In a recent study, Beckers et al. propose to apply the attack graph approach to analyze social engineering attacks, where the states of people are modeled as nodes and social engineering attacks are captured as transitions between nodes [22]. However, the attack graph approach only applies to systems that have simple and homogeneous components, and is therefore inappropriate for security analysis of complex socio-technical systems that have heterogeneous components, such as people, software, and hardware.

8 Conclusions

In this paper, we argue that analyzing attack strategies is an efficient and systematic way of identifying all potential attack scenarios, which are essential for performing security analysis from an attacker's viewpoint. As such, we examine three real attack scenarios to understand how attackers elaborate their malicious intentions, from which we summarize five refinement patterns. Based on these refinement patterns, we further propose an anti-goal refinement framework for systematically generate attack strategies from an attacker's viewpoint. Finally, we evaluate our proposal with another scenario of credit card theft.

In the future, we plan to seamlessly integrate the anti-goal refinement framework into our holistic attack modeling and analysis framework [6, 7] using the attacker's viewpoint as part of the holistic design of secure systems. Next, we aim to implement the anti-goal refinement patterns into formal logic inference rules using Datalog and extend the tool MUSER [15] to support the semi-automatic application of the framework. Finally, with the tool support, we aim to further evaluate our approach with more real attack scenarios.

Acknowledgements Trento authors are supported by the ERC advanced grant 267856, titled "Lucretius: Foundations for Software Evolution". Jennifer Horkoff is supported by an ERC Marie Skodowska-Curie Intra European Fellowship (PIEF-GA-2013-627489), and by a Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship (Sept. 2014 - Aug. 2016).

References

1. Schneier, B.: Attack trees. Dr. Dobb's journal **24**(12) (1999) 21–29
2. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. Requirements Engineering **10**(1) (2005) 34–44

3. Morais, A., Hwang, I., Cavalli, A., Martins, E.: Generating attack scenarios for the system security validation. Networking science **2**(3-4) (2013) 69–80
4. Shostack, A.: Threat Modeling: Designing for Security. John Wiley & Sons (2014)
5. Lamsweerde, A.V.: Elaborating security requirements by construction of intentional anti-models. In: ICSE. (2004) 148–157
6. Li, T., Paja, E., Mylopoulos, J., Horkoff, J., Beckers, K.: Holistic security requirements analysis: An attacker's perspective. In: Requirements Engineering Conference (RE), 2015 IEEE 23nd International, to be published (2015)
7. Li, T., Horkoff, J., Beckers, K., Paja, E., Mylopoulos, J.: A holistic approach to security attack modeling and analysis. In: Proceedings of the Eighth International i* Workshop, to be published (2015)
8. Li, T., Horkoff, J.: Dealing with security requirements for socio-technical systems: A holistic approach. In: CAiSE'14. (2014)
9. Mitnick, K.D., Simon, W.L.: The art of deception: Controlling the human element of security. John Wiley & Sons (2011)
10. Skoudis, E., Liston, T.: Counter hack reloaded: a step-by-step guide to computer attacks and effective defenses. Prentice Hall Press (2005)
11. Jureta, I., Borgida, A., Ernst, N., Mylopoulos, J.: Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In: Proc. of RE'10. (2010) 115–124
12. Scandariato, R., Wuyts, K., Joosen, W.: A descriptive study of microsofts threat modeling technique. Requirements Engineering **20**(2) 163–180
13. ISO, I., Std, I.: Iso 27005: 2011. Information technology–Security techniques–Information security risk management. ISO (2011)
14. Lund, M.S., Solhaug, B., Stølen, K.: Model-driven risk analysis: the CORAS approach. Springer Science & Business Media (2010)
15. Li, T., Horkoff, J., Mylopoulos, J.: A prototype tool for modeling and analyzing security requirements from a holistic viewpoint. In: the CAiSE'14 Forum at the 26th International Conference on Advanced Information Systems Engineering. (2014)
16. Moore, A.P., Ellison, R.J., Linger, R.C.: Attack modeling for information security and survivability. Technical report, CMU-SEI-2001-TN-001. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST (2001)
17. Barnum, S., Sethi, A.: Attack patterns as a knowledge resource for building secure software. In: OMG Software Assurance Workshop: Cigital. (2007)
18. Letier, E., Van Lamsweerde, A.: Agent-based tactics for goal-oriented requirements elaboration. In: Proceedings of the 24th International Conference on Software Engineering, ACM (2002) 83–93
19. Paul, S.: Towards automating the construction & maintenance of attack trees: a feasibility study. arXiv preprint arXiv:1404.1986 (2014)
20. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: Proceedings of the 1998 workshop on New security paradigms, ACM (1998) 71–79
21. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on, IEEE (2002) 273–284
22. Beckers, K., Krautsevich, L., Yautsiukhin, A.: Analysis of social engineering threats with attack graphs. In: Data Privacy Management, Autonomous Spontaneous Security, and Security Assurance. Springer (2015) 216–232