Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

SIMULATION-BASED OPTIMIZATION OF INFORMATION SECURITY CONTROLS: AN ADVERSARY-CENTRIC APPROACH*

Elmar Kiesling

Information and Software Engineering Group
Vienna University of Technology
Favoritenstraße 9-11/188
1040 Vienna, AUSTRIA

Andreas Ekelhart, Bernhard Grill

SBA Research
Favoritenstraße 16
1040 Vienna, AUSTRIA

Christine Strauss

Dept. of Business Administration
University of Vienna
Rossauer Lände 3
1090 Vienna, AUSTRIA

Christian Stummer

Dept. of Business Administration and Economics
Bielefeld University
Universitätsstr. 25
33615 Bielefeld, GERMANY

ABSTRACT

Today, information systems are threatened not only by the opportunistic exploitation of particular technical weaknesses, but increasingly by targeted attacks that combine multiple vectors to achieve the attacker's objectives. Given the complexities involved, identifying the most appropriate measures to counteract the latter threats is highly challenging. In this paper, we introduce a novel simulation-optimization method that tackles this problem. It combines rich conceptual modeling of security knowledge with discrete event simulation and metaheuristic optimization techniques. By simulating attacks, the method infers possible routes of attack and identifies emergent weaknesses while accounting for adversaries' heterogeneous objectives, capabilities, and available modes of entry. The optimization iteratively adapts the system model by means of a genetic algorithm and optimizes its ability to detect ongoing attacks and prevent their successful execution. We describe a prototypical implementation and illustrate its application by means of scenarios for five types of adversaries.

1 INTRODUCTION

Most organizations today have developed a critical dependence on information systems. Securing these systems in order to protect the reputation, profitability, confidence in, and, ultimately, the existence of their organizations has therefore become a major concern that decision-makers have to address. Historically, security investment decisions mainly focused on protecting systems from threats such as worms or viruses, in line with the conception of IT security as purely a technical problem. Although automated attacks that exploit singular technical weaknesses still remain relevant, their importance today is eclipsed by targeted attacks that combine different approaches and coordinate the effects of multiple steps to reach particular goals. In this context, security can no longer be viewed as purely a technical issue, but has to be conceived

as an emergent property that involves interdependent aspects such as network security, software security, physical security, organizational security, and so forth.

In the light of the increasingly complex nature of information security problems and the large array of available physical, technical, operational, and organizational controls that aim to improve it (e.g., virus scanners, firewalls, intrusion detection and prevention systems, access control systems, encryption, patch management, security awareness training), decision-makers struggle to identify the most appropriate means to counteract the threats their organizations face.

In this context, it is fundamental to gain an understanding of potential adversaries, their motivations, capabilities, resources, trust status and objectives because these characteristics determine their attack campaign and, ultimately, the risk they pose to an information system. Hence, these aspects must be considered in security analyses and decisions because without addressing the question "secure from whom?", the term "security" itself is essentially meaningless (Schneier 2000).

In this paper, we introduce an approach that is based on an adversary-centric view and combines modeling and simulation-optimization techniques. It relies on (i) a model of the system to be optimized, (ii) an abstract causal model of attack actions and their effects, and (iii) an explicit model of threat agents and their behavior.

Our work is related to the attack graph concepts introduced in Ammann et al. (2002) and Sawilla and Ou (2008), but follows a new direction. Whereas existing approaches are based on the idea of enumerating the entire set of possible paths to build a complete attack graph, we conceive the discovery of attack paths as a dynamic process. In this respect, we follow some ideas for state-space modeling and simulation from the literature. Related work includes advanced formalisms for dynamic security simulations, such as Generalized Stochastic Petri Nets (Dalton et al. 2006) and Boolean Logic Driven Markov Processes (BDMP) (Piétre-Cambacédès and Bouissou 2010). In contrast to these contributions, our approach simulates human adversaries as agents that make active decisions in attacking a system, deliberately exploiting dynamic interactions of vulnerabilities. Furthermore, we introduce metaheuristic methods to optimize information systems and enable decision-makers to study how its security may be improved (e.g., by adding physical, technical, operational, and organizational security controls) while trading off multiple objectives (e.g., effectiveness against different types of adversaries, cost, risk, awareness of attacks etc.).

The paper is organized as follows: In section 2, we introduce our approach for attack modeling and simulation. Section 3 is concerned with meta-heuristic techniques for the optimization of security control sets. Section 4 illustrates the practical usefulness of the approach by means of optimizations for sample scenarios. We close with concluding remarks and an outlook on future research in Section 5.

2 ATTACK SIMULATION

Figure 1 provides a high-level overview of the proposed attack modeling and simulation framework. We will explain the constituent components in the following sections.

2.1 Knowledge base

The knowledge base (KB) consists of (i) an attack and control model and (ii) a system model. It captures the knowledge required to simulate attacks and preserves it in a well-structured and reusable manner. Our implementation uses the declarative logic programming language Prolog to model and reason on the system environment.

Attack and control model The attack and control model formally specifies how systems may be attacked (and secured) by means of *attack patterns*. Each pattern is applicable in a particular context under specific preconditions. We define actions as clauses with a set of preconditions in the body:

```
action_sqlInjection(Attacker, DbServer):-
        technicalSkillLevel(Attacker, TechnicalSkillLevel),
        TechnicalSkillLevel >= 1,
        owned(Attacker, AttackHost),
```

```
connected(AttackHost, WebServer, httpProtocol, httpPort),
connected(WebServer, DbServer, dbProtocol, dbPort),
usesDbSoftware(WebServer, DbServer),
not(owned(Attacker, DbServer)).
```

This example defines the preconditions required for an SQL injection attack. Prolog responds to the query *action_sqlInjection* by returning all variable bindings for *Attacker* and *DbServer* which satisfy the subgoals, based on the current facts in the KB. To each attack action, we assign (i) costs, (ii) execution time, (iii) base probability for successful execution, and (iv) the maximum number of possible attempts, which are attributes required at simulation execution time.

Postconditions define possible outcomes of an attack action, i.e., they specify state transitions that occur during the simulation. Basic outcome types are successful attacks and failed attacks. Successful execution of an attack action, could, for example, allow the attacker to gain root access to the target computer, while a failed attempt could take the target computer offline. Formally, we define both types as execution rules that assert a change in the KB:

```
exec_success_action_sqlInjection(Attacker, DbServer):-
        assert(owned(Attacker, DbServer)).
```

Invoking this rule with valid bindings from the previous query asserts the fact that the attacker has access to the DB server, which may in turn affect the results of subsequent queries. The attack and control model also contains preventive and detective security controls that may be applied to certain system elements to increase the difficulty of attacks or enable the system to detect ongoing attacks, respectively.

```
control_properties(control_ids, detective, false, null, max, delayed, 1000, ids, hostGroup).
```

This example defines an intrusion detection control *ids* and the associated properties, which determine (i) if the control type is *detective* or *preventive*, (ii) whether the control is visible to an attacker, (iii) the outcome of the control (*stop* or *null*) (iv) the aggregation type that determines the effect of multiple controls on an asset (*min*, *max*, *cumulate*), (v) the control response type (*immediate* or *delayed*), (vi) the amount of the detection delay, (vii) the type of assets that implement this control, and (viii) the type of assets that this control can be applied to. A particular control instance *ids*1 with associated effectiveness in the context of an action as well as specific costs can be defined as follows:

```
control_effectiveness(ids1, action_sqlInjection, 0.80).
ids(ids1, 7500).
```

System Model To reason about the security of a particular information system, it is necessary to model its constituent components and establish relations between them. To this end, we model specific knowledge about the system to be protected, but separate it from the abstract attack knowledge. While a

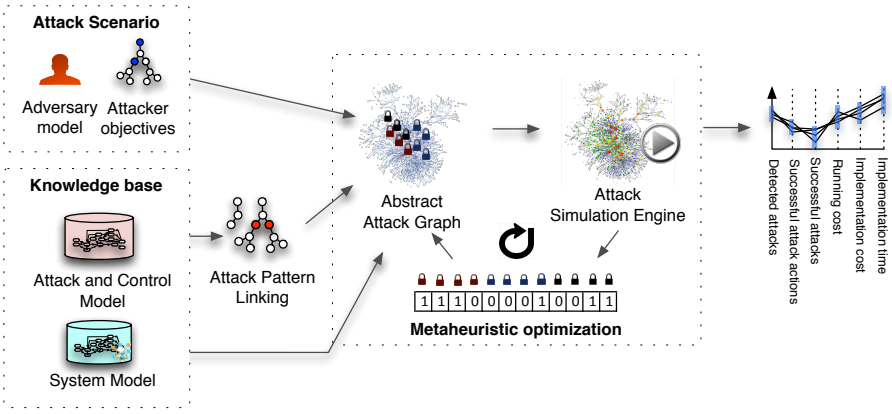

Figure 1: Optimization framework overview

domain specific attack model can be reused by all organizations facing similar threats, the system model is organization specific, and hence, kept in a separate file. In a corporate information system context, this model may contain a set of tangible and intangible assets, including hardware components, networks, data, employees, policies etc. To illustrate how an asset is modeled and controls are associated with it, consider the previously given example of an intrusion detection control. First, an instance of *host* is added to the system model. Second, the *installed* relation is used to link a candidate intrusion detection software asset to the host. Furthermore, we can state that the *dbServers_host_1* stores the project database *projectDb1*, and that the server is part of the host group *dbServerHosts*. *workstationHosts* can access the *dbServerHosts* over HTTP. In the same way, additional infrastructure knowledge is modeled.

```
host(dbServers_host_1).
installed(dbServers_host_1, ids1).
stores(dbServers_host_1, projectDb1).
inHostGroup(dbServers_host_1, dbServerHosts).
hacl(workstationHosts, dbServerHosts, httpProtocol, httpPort).
```

2.2 Attack Scenario

The attack simulation leverages and combines the knowledge embodied in the attack, control and system models by executing dynamic attacks for given threat scenarios. Each threat scenario consists of an adversary model and a particular objective, which is formally specified as a target condition describing a desired system state (e.g., attacker has access to *db2*). We formally specify an adversary type with associated characteristics. This characteristics include initial access (we may, for example, specify that the *employee* adversary type has access privileges on *workstationHosts* and is member of the *workstationUserGroup*), behavioral attributes (risk aversion, propensity to alternate between different attack strategies etc.), and resources (time). Adversaries are also associated with a technical skill level that determines whether certain advanced actions are available. An organization can define relevant adversary types according to their threat profile.

2.3 Simulation Execution

Adversaries' choices regarding their course of action, the outcome of individual attack actions, and the detection of attacks are probabilistic aspects in the simulation. Therefore, it is necessary to perform multiple replications with varying sets of random seeds for each attack scenario to capture the inherent variability and uncertainty. For each replication, the simulation executes a schedule of discrete events and records various outcome variables which can later be analyzed and aggregated. This flexible framework can capture complex causal relations and timing interactions.

The types of events used in the simulation are illustrated in Fig. 2. *Action selection* events schedule *Action Start Events* using the adversary's behavioral model; upon execution, these events determine the effective duration of the action (which is influenced by factors such as difficulty, adversary skills, preventive controls etc.) and schedule an *Action End Event* accordingly. Detection events may be triggered when assets associated with a detective control are being attacked; an *Attacker Stopped Event* may be scheduled by terminating detective controls. Finally, a *Target Condition Reached Event* is scheduled in case the attacker has reached the target condition. The mechanisms invoked through these events, i.e., *action selection*, *action execution*, and *detective control response*, are explained in the following paragraphs.

Action selection We associate adversaries with a behavioral model that iteratively selects attack actions based on (i) individual adversary characteristics, (ii) the attacker's general knowledge about possible routes of attack, and (iii) the outcomes of prior attack actions. This behavioral model is based on the idea that adversaries have general security knowledge, but not necessarily complete specific knowledge about the system being attacked. The general knowledge about possible routes of attack is embodied in an abstract attack graph for the pursued target condition. This graph may be interpreted as an adversary's mental map of how actions can be combined to reach a particular outcome. The simulation applies this

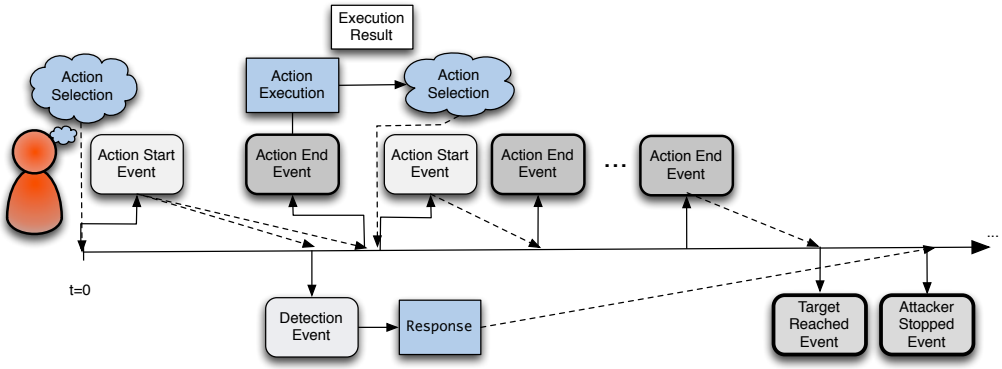

Figure 2: Discrete event types and scheduling of events

abstract knowledge by querying the knowledge base for valid asset assignments for all preconditions of an abstract action in the context of the modeled system. This yields a set of concrete action instances $\overline{A}$ that can be executed against particular assets. We denote abstract actions by a and concrete action instances (i.e., abstract actions with assigned variables) by $\overline{a}$. In our application example, we use the behavioral model specified in Algorithm 1 for all internal and external adversaries and assign varying parameters for $w_{distance}$, $w_{success}$ and $w_{detection}$. For the other behavioral parameters, we set uniform values for all adversary types ($p_{continueNew} = 0.9$, $p_{retry} = 0.5$, $p_{alternativesNoNew} = 0.7$, and $p_{alternativesFailed} = 0.3$).

According to this model, adversaries alternate between following a chosen attack path (i.e., a sequence of abstract attack actions) and trying new tactics (i.e., choosing new entry points and attack paths). We assume that adversaries minimize expected effort, i.e., they tend to launch attacks "close" to the target if possible. This idea is implemented in the $choose(\overline{A})$ procedure of Algorithm 1, which chooses among preselected sets of action instances. It calculates the shortest path distances $d(a,t)$ in terms of cumulative effort required between each abstract attack action a and the target condition t using Dijkstra's algorithm (Dijkstra 1959). Using the longest abstract distance found as a reference, it calculates the relative distance for all candidate actions based on their position in the abstract graph.

The procedure then calculates individual weights $w_{\overline{a}}$ for each action instance $\overline{a}$ based on $\overline{a}$'s success and detection probabilities $p_{success}(\overline{a})$ and $p_{detection}(\overline{a})$ as well as the relative distance $d^{rel}_{\overline{a}}$ to the target condition. Procedure $weightedChoice(\overline{A}, W)$ then determines the final choice by selecting from the candidate action instances with probabilities proportional to the calculated weights W.

After the initial action has been selected, subsequent choices are dependent upon the result of the previously executed action p. If the previous action was successful and new action instances $\overline{N}$ have become available as a consequence, the adversary will choose among them with probability $p_{continueNew}$ and choose among all actions otherwise (lines 9-15).

If the previous action was successful, but did not yield any new action instances, then there are two options (lines 16-21). With probability $p_{alternativesNoNew}$, the adversary will choose among action instances that have become available as a consequence of the same action as $\overline{p}$, i.e., those returned by $getAlternatives(\overline{a})$. Otherwise, the adversary will search for a new entry point by selecting from all available actions. In case the previous action $\overline{p}$ was not successful, the adversary will retry with probability p_{retry}. Otherwise, it will either choose a neighboring action instance (with probability $p_{alternativesFailed}$) or search for a new entry point among all available actions. At each step during an attack, the adversary will chose available action instances directly if their outcome fulfills the target condition (lines 1-6).

Action execution As noted above, each action execution is triggered by an *Action End Event*. Upon execution of such an event, we determine (i) whether the executed action was successful, (ii) which actions are no longer available, (iii) which new actions have become available, and (iv) whether the attack target has been reached. The result then drives the selection of the following action.

Algorithm 1 Attacker behavior model

Input: available actions $\overline{A}$, previous action p, new actions $\overline{N}$
Output: selected action

```
1:  procedure SELECTACTION(Ā, p, N̄)
2:      for a ∈ Ā do
3:          if FULLFILLSTARGETCOND(ā) then
4:              return ā
5:          end if
6:      end for
7:      if p = null then
8:          CHOICE(Ā)
9:      else if WASSUCCESSFUL(p̄) then
10:         if |N| > 0 then
11:             if p_continueNew > RANDOM() then
12:                 return CHOOSE(N̄)
13:             else
14:                 return CHOOSE(Ā)
15:             end if
16:         else if p_altNoNew > RANDOM() then
17:             S̄ ← GETALTERNATIVES(p)
18:             return CHOOSE(S̄)
19:         else
20:             return CHOOSE(Ā)
21:         end if
22:     else if p_retry > RANDOM() then return p
23:     else if p_altFailed > RANDOM() then
24:         S̄ ← GETALTERNATIVES(p)
25:         return CHOOSE(S̄)
26:     else
27:         return CHOOSE(Ā)
28:     end if
29: end procedure
```

```
1: procedure CHOOSE(Ā)
2:     for ā ∈ Ā do
3:         d_ā^rel ← d(a,t) / (max(d(a,t))+1)
4:         W_ā ← p_success(ā)^w_success (1 − p_detection(ā))^w_detection (1 − d_ā^rel)^w_distance
5:     end for
6:     return WEIGHTEDCHOICE(Ā, W)
7: end procedure
```

Detective control response Detective controls define two possible outcomes when attacks are detected: (i) *Stop*, i.e., the simulation run terminates, or (ii) a defined path to a postcondition (modeled in the abstract attack graph) is executed. These postconditions might change property values in the system model. For example, when an intrusion detection system blocks the adversary's source IP it inhibits certain attack actions but potentially also enables new attack actions (e.g., IPS reaction leads to service disruption).

3 CONTROL OPTIMIZATION

The simulation model can support security decision-makers through an iterative process that consists in modeling a system with a set of particular security controls in place, running the simulation to assess the joint effectiveness of all implemented controls, and interpreting the results. Although this process yields valuable insights, discovering interactions between various individual security controls and optimizing them by altering the system configuration manually until satisfactory results can be achieved is difficult and tedious. Simulation optimization techniques can alleviate these problems. For an introduction and review of techniques, cf. Tekin and Sabuncuouglu (2004) and Fu et al. (2005).

Our problem is characterized by a discrete, finite decision space (system designs), multiple objectives (cost, prevention of attacks, detection of attacks etc.), and, due to the complex relationship between system configurations and outcome variables, a multi-modal response surface. Multi-objective genetic algorithms are a particularly promising global optimization technique for this problem class. Hence, we introduce the concept of a *control portfolio* characterized by a *genotype string* of binary indicator variables. Each variable

represents a control-asset pair (e.g., *logPolicy1 - dbServerHosts*) that takes the value 1 if the respective control is applied to the asset and 0 otherwise.

To evaluate a control portfolio, we initialize the system by applying controls to assets according to its genotype. We then simulate a number of attacks with varying random seeds and record the outcomes using various metrics. These metrics can be aggregated across simulation runs (e.g., *sum*, *min*, *max*, *average*, *median*) and used as optimization objectives. Because we consider multiple objectives (cost, effectiveness of controls in preventing and detecting attacks etc.) simultaneously, the optimization does not generally result in a single "best" control portfolio, but typically in a set of (proposed) *(Pareto-)efficient* portfolios. Each proposed efficient portfolio is nondominated, i.e., there is no other portfolio with equal or better values for all objectives and a strictly better value in one of the objectives.

Computation-wise, this simulation-optimization problem is particularly challenging, mainly due the combinatorial size of the decision space (2^n where n is the number of control-asset combinations), the expensive simulation-based evaluation procedure, and the need to account for multiple optimization objectives. Exact solutions (i.e., complete sets of all efficient portfolios for a given number of simulation replications) can only be obtained for very small problem instances through complete enumeration. To tackle larger, more practically relevant problem sizes, we therefore resort to meta-heuristic solution procedures. A key advantage of these approaches is their versatility and ability to optimize large combinatorial problems without a priori assumptions about their structure. For our optimization problem at hand, we experimented with genetic algorithms. These population-based approaches are inspired by nature and optimize problems implicitly through evolutionary mechanisms. They evolve a population of individuals (control portfolios) iteratively by evaluating their fitness (assessing objectives), selecting fit individuals (control portfolios), and performing crossover and mutation operations on the selected individuals to generate offspring. The binary genotype strings generated in this process are evaluated using the simulation, which performs a number of replications and returns aggregate objective values used for assessing the "fitness" of the respective control portfolio.

To evaluate algorithms, we so far conducted preliminary experiments for small problem instances (search space 2^{12}) with the NSGA-II (Deb et al. 2000) and SPEA-II (Zitzler et al. 2002) algorithms. We evaluated solution quality and convergence characteristics by comparing results to that of a complete enumeration and found that both algorithms performed reasonably well. Using the standard parameters recommended in the original papers, NSGA-II performed slightly better and on average identified approximately 65% of all efficient solutions within covering the first 20% of the search space. We therefore used NSGA-II for experiments with larger problem sizes, which we discuss in the following section. We expect that customization and parameter tuning will yield significant performance improvements.

4 APPLICATION EXAMPLE

We illustrate the practical application of our tool by means of a hypothetical, but realistic example for which we perform simulation and optimization experiments.

4.1 Implementation

The attack simulation model and optimization algorithms used were implemented in Java. We use the scheduling mechanisms provided by MASON (Luke et al. 2004), a fast discrete-event simulation core which also provides the pseudo-random numbers used is in the simulation, generated by a fast Mersenne Twister implementation. The underlying knowledge base was implemented in SWI-Prolog (Wielemaker et al. 2012) and accessed in Java via JPL (Singleton, P. and Dushin, F. and Wielemaker, J.). The optimization on top of the simulation core was implemented using Opt4J (Lukasiewycz et al. 2011), a flexible framework for implementing and testing meta-heuristics.

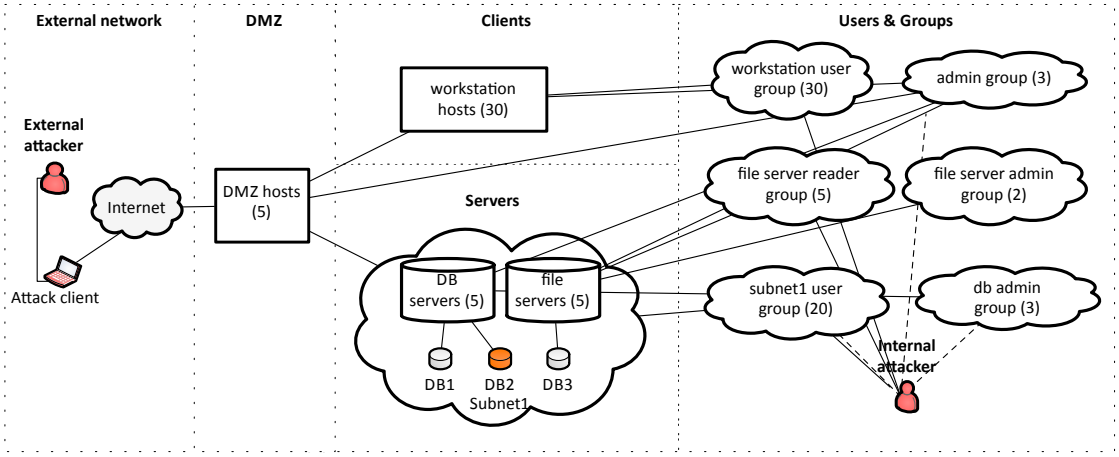

Figure 3: Scenario system model overview

4.2 Domain Description

First, we created a general attack model based on an existing penetration testing catalogue. This model comprises attack patterns (e.g., brute force, SQL injection, social attack, spearfish attack, keylogger, and backdoor installation) and legitimate actions that can be used maliciously (e.g., connecting to another computer or accessing data by username and password). To account for attacks that exploit previously unknown vulnerabilities, we added "zero-day" attack actions. Successful execution of these actions gives the adversary access to the attacked host. Existing controls such as antivirus software are largely ineffective against these kinds of attacks.

Next, we enriched the model with security control definitions. Controls include, for example, antivirus software, patches for specific software, intrusion detection systems, log policies, security training.

Finally, we modeled the organization and its IT infrastructure by creating instances of concepts such as *Host*, *Subnet*, *Data*, and *User*, as well as relations between these concepts (e.g., *Host stores Data*, *Host uses Software*, *User in UserGroup*). For this illustrative scenario, a synthetic example system model was generated automatically. It contains 30 hosts, 5 web servers, 5 database servers, 30 employees, and 3 administrators. Fig. 3 gives an overview of the generated system (details such as connections between systems, installed software etc. have been omitted for the sake of clarity).

4.3 Experimental Setup

The primary goal of the proposed optimization method is to facilitate an understanding of how a system may be protected against different types of adversaries. For our illustrative application, we defined five types of adversaries summarized in Table 1. We distinguish between external and internal adversaries. The latter already have some level of access to the system. External adversaries, by contrast, are assigned an attack host connected to the Internet only. In this case, the demilitarized zone (DMZ) is the main entry point to the company's internal network.

We further assigned heterogeneous preference weights, time budgets and other parameters that shape adversaries' simulated behavior. An advanced persistent threat (APT), for example, has a large time budget, the highest technical skill level, and is highly risk-averse (i.e., has a strong preference for avoiding detection). An unskilled external attacker, on the other hand, is less patient and hence has a low time budget, a low technical skill level, and is primarily focused on successfully executing attack actions and is less concerned about being detected. The adversary type Employee has access to the internal workstation hosts but no technical skill level whatsoever. In all simulation experiments, the target condition for all adversaries is to access the data set *db2*.

Table 1: Adversary types

adversary type	timeBudget (sec.)	$w_{detection}$	$w_{success}$	$w_{distance}$	access
Employee	150000	0.45	0.25	0.30	workstation hosts
Administrator	300000	0.50	0.20	0.30	all hosts and servers
Skilled External	200000	0.30	0.40	0.30	attack client
Unskilled External	100000	0.30	0.40	0.30	attack client
Advanced Persistent Threat	10000000	0.50	0.20	0.30	attack client

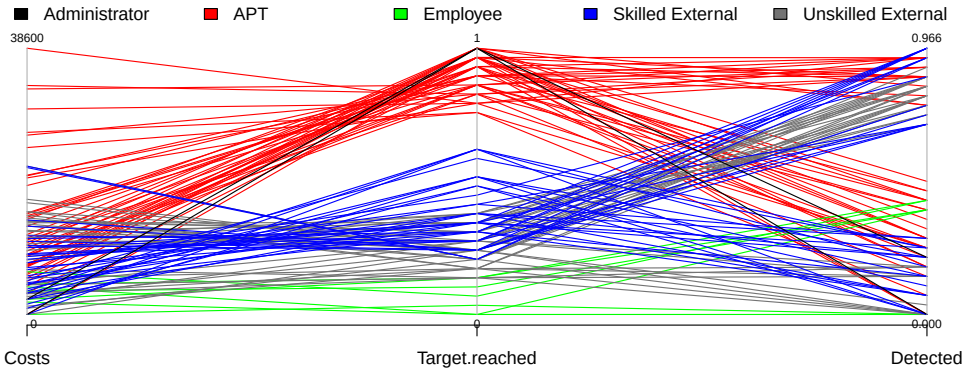

Figure 4: Objective values of proposed efficient portfolios

In the experiments, we used *minimize costs*, *minimize successful attacks*, and *maximize detection of attacks* as optimization objectives. We performed 25 simulation replications for each portfolio for 250 generations using NSGA-II with recommended standard parameters (cf. Deb et al. 2000).

4.4 Results

All optimization scenarios were executed on a 2x3Ghz Xeon machine and took between 92 minutes (Admin scenario) and 31 hours (APT scenario) to complete, depending on the complexity of the respective scenario. The genetic algorithm identified 40 efficient portfolios for the advanced persistent threat scenario, 33 for the skilled external attacker, and 33 for the unskilled external attacker. Compared to these external threat sources, the number of portfolios was much lower for internal attackers – 2 for the administrator scenario and 7 for the employee scenario. Fig. 4 visualizes the portfolios and their objective values.

First focusing on the administrator, we see that this type of adversary always reaches the target, which is expected given that he/she already has all required access privileges. For this type of attacker, there are no effective preventive controls available, but applying a log policy on the hosting *dbServerHosts* at least raises the detection rate to approximately 20%.

An employee as an attacker on the other hand succeeds in obtaining access to the target data set only in 13,7% of simulation runs at best. Table 2 summarizes the prevalence of controls in efficient portfolios relative to the theoretical maximum, i.e., no. efficient portfolios x no. assets that the control can be applied to.

Even though the employee has access to internal systems, he cannot access the hosting servers directly and lacks the skills for technical attacks and must solely rely on social attacks. Accordingly, a control portfolio that includes security trainings (*train1*, *train2*, *train3*) for the database (dbAdminGroup) and server administrators (*adminGroup*) turns out as highly effective against this type of attacker.

As expected, the advanced persistent threat has the highest success rate of all external attackers. Due to its proficiency and budget, a wide range of attacks and attack paths are available, and hence, a large number of security controls is present in most portfolios. Patching vulnerable systems, deploying antivirus

Table 2: Relative prevalence of controls in proposed efficient portfolios

adversary type	av1	av2	ids1	ids2	patch	log	train1	train2	train3
Administrator	0	0	0	0	0	0.1	0	0	0
Employee	0	0	0	0	0	0.2	0.0238	0.0714	0
Skilled external	0.1484	0.0387	0.1032	0.0194	0.0903	0.1548	0.0430	0.0160	0
Unskilled external	0.2242	0	0.1394	0.0667	0.1273	0.1333	0.02024	0.0707	0
APT	0.1484	0.0387	0.1032	0.0194	0.0903	0.1548	0.0430	0.0160	0

software, and performing security trainings are the most common preventive controls in efficient portfolios. For the *dmzHosts* in particular, *av2* – which in our model offers a higher detection rate at higher costs than *av1* – is frequently included in efficient portfolios. The highest level of security training for the *adminGroup* only occurs in portfolios for this attacker type. Even with high investments in security controls, however, this attacker can rarely be stopped from reaching the target.

Optimization results for the skilled external attacker were similar, even though this type of adversary succeeded less frequently. Accordingly, efficient portfolios tended to include less expensive controls. Against an unskilled external attacker, intrusion detection systems proved effective and showed a high detection rate. To prevent email attacks (including backdoors and keyloggers), security training, especially for workstation users (*workstationUserGroup*) were included in many portfolios.

5 CONCLUSIONS

In this paper, we introduced a novel simulation-based method for assessing and optimizing the security of complex information systems. Our approach combines abstract attack knowledge and a model of the system to be protected with discrete event simulation and genetic algorithms for multi-criteria optimization. The method identifies efficient sets of security controls to prevent and detect attacks. This flexible approach facilitates the modeling of complex causal and temporal interactions as well as the proper differentiation between different types of adversaries. It not only determines whether or not there are possible paths to reach a target condition for different types of adversaries, but also performs probabilistic analyses on which combinations of controls are likely to detect attacks and prevent their successful execution. Rather than assessing individual security controls separately, this integrated evaluation recognizes that the combined effectiveness of security controls is generally not cumulative, but highly context-dependent and the result of complex interactions. By explicitly modeling adversaries' behavior using abstract attack graphs as mental maps, we can generate attack paths dynamically through simulation. This is a salient feature because the construction of complete graphs of all possible attacks on a particular systems is typically infeasible for large networked environments.

From a practical perspective, the promising results of our simulation-optimization experiments suggest that the proposed approach is viable and can be usefully applied to study and optimize real-world information systems. An important advantage of the proposed approach in this context is the separation of abstract attack knowledge and knowledge about the particular information system to be protected. Because of this separation, only general knowledge on generic attack patterns needs to be modeled by a security domain expert, whereas organization that want to leverage the expert's formalized knowledge only need to model their own information system to do so.

The development of such a comprehensive attack knowledge base is ongoing work. It necessitates the definition of modeling conventions and sets of common concepts that can be used across security subdomains. Once formally implemented, an extensive set of attack patterns would support a broad view on security and help to discover attack paths that can otherwise not be easily identified.

Since the proposed framework rests fundamentally on adequate modeling of adversaries' behavioral patterns, we are also in the process of further refining and validating behavioral mechanisms. In this context, we draw upon and extend the existing body of literature (e.g., Liu et al. 2005, Sallhammar et al. 2005).

Customization and parameter tuning to increase the performance of the optimization are also important areas of future work in order to overcome the significant computational challenges involved in optimizing larger problem instances.

Ultimately, we aim to not only present decision-makers with lists of efficient portfolios of security controls for each adversary type, but to provide interactive decision support for the selection of a final set of security controls to implement. To this end, we will implement mechanisms for assessing the effectiveness of control portfolios against multiple adversary types and apply suitable interactive visualizations to explore the space of efficient solutions (e.g., those described in Neubauer et al. 2006 or Gettinger et al. 2013).

ACKNOWLEDGMENTS

The work presented in this paper is developed within the project "MOSES3" funded by the Austrian Science Fund (FWF): P23122-N23. The research is carried out at Secure Business Austria, a COMET K1 program competence center supported by FFG - Austrian Research Promotion Agency.

REFERENCES

Ammann, P., D. Wijesekera, and S. Kaushik. 2002. "Scalable, graph-based network vulnerability analysis". In *Proceedings of the 9th ACM conference on Computer and communications security*, 217–224: ACM.

Dalton, G. C., R. F. Mills, J. M. Colombi, and R. A. Raines. 2006. "Analyzing attack trees using generalized stochastic petri nets". In *IEEE Information Assurance Workshop*, 116–123.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2000. "A fast elitist multi-objective genetic algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197.

Dijkstra, E. W. 1959. "A note on two problems in connexion with graphs". *Numerische Mathematik* 1 (1): 269–271.

Fu, M., F. Glover, and J. April. 2005. "Simulation optimization: a review, new developments, and applications". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. Kuhl, N. Steiger, F. Armstrong, and J. Joines, 83–95. Orlando, FL: IEEE Press.

Gettinger, J., E. Kiesling, C. Stummer, and R. Vetschera. 2013. "A comparison of representations for discrete multi-criteria decision problems". *Decision Support Systems* 54 (2): 976–985.

Liu, P., W. Zang, and M. Yu. 2005. "Incentive-based modeling and inference of attacker intent, objectives, and strategies". *ACM Transactions on Information and System Security* 8 (1): 78–118.

Lukasiewycz, M., M. Glaß, F. Reimann, and J. Teich. 2011. "Opt4J: A modular framework for meta-heuristic optimization". In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, 1723–1730. New York: ACM.

Luke, S., C. Cioffi-Revilla, L. Panait, and K. Sullivan. 2004. "MASON: A new multi-agent simulation toolkit". In *2004 SwarmFest Workshop*.

Neubauer, T., C. Stummer, and E. Weippl. 2006. "Workshop-based multiobjective security safeguard selection". In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES 2006)*, 366–373. Vienna, Austria.

Piétre-Cambacédès, L., and M. Bouissou. 2010. "Beyond attack trees: dynamic security modeling with Boolean logic driven Markov processes (BDMP)". In *European Dependable Computing Conference (EDCC 2010)*, 199–208.

Sallhammar, K., B. E. Helvik, and S. J. Knapskog. 2005. "Incorporating attacker behavior in stochastic models of security". In *Proceedings of the International Conference on Security and Management (SAM '05)*, 79–85.

Sawilla, R. E., and X. Ou. 2008. "Identifying Critical Attack Assets in Dependency Attack Graphs". In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, 18–34: Springer.

Schneier, B. 2000. *Secrets & Lies: Digital Security in a Networked World*. New York, NY: Wiley.

Singleton, P. and Dushin, F. and Wielemaker, J. "JPL: A bidirectional Prolog/Java interface". Accessed June 14, 2013. http://www.swi-prolog.org/packages/jpl/.

Tekin, E., and I. Sabuncuouglu. 2004. "Simulation optimization: A comprehensive review on theory and applications". *IIE Transactions* 36 (11): 1067–1081.

Wielemaker, J., T. Schrijvers, M. Triska, and T. Lager. 2012. "SWI-Prolog". *Theory and Practice of Logic Programming* 12 (Special Issue 1-2): 67–96.

Zitzler, E., M. Laumanns, and L. Thiele. 2002. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm". In *Evolutionary Methods for Design, Optimisation and Control*, edited by K. Giannakoglou, D. Tsahalis, K. Papailiou, and T. Fogarty. International Center for Numerical Methods in Engineering.

AUTHOR BIOGRAPHIES

ELMAR KIESLING is a researcher in the Information and Software Engineering Group at the Vienna University of Technology, Austria. He is also affiliated with SBA Research, an industrial research center for IT security. His research interests include risk and information security management, agent-based modeling and simulation, complex systems, optimization via simulation, and decision support systems. He holds a Ph.D. in Management Science from the University of the Vienna, Austria. His email address is elmar.kiesling@tuwien.ac.at.

ANDREAS EKELHART is a researcher at SBA Research and holds a Ph.D. in Computer Science from the Vienna University of Technology. His research interests include semantic applications and applied concepts of IT security, with a focus on information security risk management. He is a member of the International Information Systems Security Certification Consortium (ISC2). His email address is aekelhart@sba-research.org.

BERNHARD GRILL received a bachelor's degree in Computer Science from the UAS Technikum Vienna and is currently enrolled in the Master program "Software Engineering and Internet Computing" at the Vienna University of Technology. His research interests inlcude windows malware, penetration testing, and secure software development. His email address is bgrill@sba-research.org.

CHRISTINE STRAUSS is Head of the Electronic Business Group at the University of Vienna, Austria. She holds a PhD in Computer Science from the University of Zurich. Prof. Strauss was also head of a research group at EC3, an electronic commerce competence center in Vienna, Austria. Her current research focuses on economic aspects in security, security portfolios, integration of security into business strategies, security and controlling, sustainable security, evaluation of risk scenarios, efficiency and sustainability in security, quantitative models in security, and decision support. Her email address is christine.strauss@univie.ac.at.

CHRISTIAN STUMMER holds the Chair of Innovation and Technology Management at the Department of Business Administration and Economics at Bielefeld University, Germany. He has served as an associate professor at the University of Vienna, Austria, as the head of a research group at the Electronic Commerce Competence Center (EC3) at Vienna, and as a visiting professor at the University of Texas at San Antonio, United States. His research focuses on multiobjective decision support, combinatorial optimization, and (agent-based) simulation with applications in various fields such as in innovation management, marketing, health care management, or IT management. Prof. Stummer is a member of the editorial boards of the EURO Journal on Decision Processes and the Central European Journal of Operations Research; he has published two books, more than thirty papers in reviewed journals, and numerous other works. His email address is christian.stummer@uni-bielefeld.de.