Open access • Proceedings Article • DOI:10.1145/1853919.1853926

Adversary-driven state-based system security evaluation — Source link

Elizabeth LeMay, Willard L. Unkenholz, Donald Parks, Carol Muehrcke ...+2 more authors

Institutions: University of Illinois at Urbana–Champaign

Published on: 15 Sep 2010 - International Workshop on Security

Topics: Computer security model, Security testing, Covert channel, Security information and event management and Security policy

ADVERSARY-DRIVEN STATE-BASED SYSTEM SECURITY EVALUATION

BY

ELIZABETH ANNE LEMAY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

ABSTRACT

Quantitative metrics can aid decision-makers in making informed trade-off decisions. In system-level security decisions, quantitative security metrics allow decision-makers to compare the relative security of different system configurations.

To produce model-based quantitative security metrics, we have formally defined and implemented the ADversary VIew Security Evaluation (ADVISE) method. Our approach is to create an executable state-based security model of a system and an adversary that represents how the adversary is likely to attack the system and the likely results of such an attack.

In an ADVISE model, attack steps are precisely defined and organized into an attack execution graph, and an adversary profile captures a particular adversary's attack preferences and attack goals. We create executable security models that combine information from the attack execution graph, the adversary profile, and the desired security metrics to produce quantitative metrics data. The ADVISE model execution algorithms use the adversary profile and the attack execution graph to simulate how the adversary is likely to attack the system. The adversary selects the best next attack step by evaluating the attractiveness of several attack steps, considering cost, payoff, and the probability of detection. The attack step decision function compares the attractiveness of different attack steps by incorporating the adversary's attack preferences and attack goals. The attack step decision function uses a state look-ahead tree to recursively compute how future attack decisions influence the attractiveness values of the current attack step options.

To efficiently produce quantitative model-based security metrics, the ADVISE method has been implemented in a tool that facilitates user input of system and adversary data and automatically generates executable models. The tool was used in two case studies that illustrate how to analyze the security of a system using the ADVISE method. The case studies demonstrate the feasibility of ADVISE and provide an example of the type of security analysis that ADVISE enables.

The ADVISE method aggregates security-relevant information about a system and its adversaries to produce a quantitative security analysis useful for holistic system security decisions. System architects can use ADVISE models to compare the security strength of system architecture variants and analyze the threats posed by different adversaries.

Soli Deo Gloria

ACKNOWLEDGMENTS

The work depicted here was performed, in part, with funding from the U.S. Department of Homeland Security under contract FA8750-09-C-0039 with the Air Force Research Laboratory. I particularly wish to thank Douglas Maughan, Director of the Cyber Security Division in the Homeland Security Advanced Research Projects Agency (HSARPA), within the Science and Technology Directorate of the Department of Homeland Security (DHS).

I thank my advisor, Professor William H. Sanders, for his technical advice and his support for this project. He believed in the value of this research project from the beginning and supported the implementation of ADVISE models in the Möbius modeling and simulation tool. I also appreciate his support of my interest in teaching and engineering education while I was a graduate student. I would also like to thank my other committee members, Professor Nikita Borisov, Professor Michael C. Loui, and Professor David M. Nicol, for their questions and insights, especially those during the preliminary examination that guided the final development phase of ADVISE.

I gratefully acknowledge valuable and substantial technical contributions from Willard Unkenholz and Donald Parks, U.S. Department of Defense, during the earliest development of the ADVISE method. It was a pleasure to collaborate with them, and the inclusion of the do-nothing attack step and the three decision criteria are just two examples of their input into what has become the ADVISE method.

The development of ADVISE as a method and a tool was a team effort. Michael Ford built some of the earliest ADVISE models and contributed to the development of ADVISE and the proofs of its optimality. Ken Keefe was substantially responsible for the implementation of ADVISE models within the Möbius tool. Jenny Applequist provided many valuable editorial comments on conference papers and this dissertation.

Other key collaborators on the ADVISE research project include Dr. Carol Muehrcke of Cyber Defense Agency and Bruce Barnett and Michael Dell'Anno of GE. GE provided funding and engaged in valuable discussions about the electric power distribution case study used to test and refine the ADVISE method.

I thank the other PERFORM research group members and my friends and fellow graduate students at University Baptist Church for making my six years in graduate school immeasurably more enjoyable.

I thank my parents, A.J. and Laurel Van Ruitenbeek, for instilling in me the value of education and encouraging me to always do my best. I especially thank my mother, who would not let me give up on finishing graduate school.

I thank my husband, Dr. Michael LeMay, for his unfailing support, love, and encouragement, especially through the rough parts of this project. Michael is the most diligent, conscientious researcher I know, and his perseverance in his own research inspires me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\beta(s)$	A *best next attack step* in state s with a maximal attractiveness value (using a planning horizon of one attack step)
$\beta^N(s)$	A *best next attack step* in state s with a maximal attractiveness value, computed recursively using a planning horizon of N
δ	The *type* of event metric
ϵ	The *event set* for an event metric
λ	The *type* of state metric
σ	The *state indicator function* for a state metric
τ	The *end time* for a state metric or an event metric such that the metric reports on the time period $[0, \tau]$
A	The set of *attack steps* against the system
A_s	The set of *available attack steps* in state s
a_{DN}	The "do-nothing" attack step
$attr(a_i, s)$	The *attractiveness* of available attack step $a_i \in A_s$ in state s (using a planning horizon of one attack step)
$attr^N(a_i, s)$	The *attractiveness* of available attack step $a_i \in A_s$ in state s, computed recursively in a way that evaluates all possible sequences of attack steps of length N that begin with attack step a_i
$B_i(s)$	The *Boolean precondition* for attack step $a_i \in A$ in state s
$C_i(s)$	The *cost* of attempting attack step $a_i \in A$ (regardless of the outcome) in state s
$C_i^N(s)$	The *expected path cost* when the adversary is in state s, attempts attack step $a_i \in A$, and then attempts other attack steps continuing a total of N steps into the future

$D_i(s)$	The *expected detection probability* associated with attempting attack step $a_i \in A_s$ in state s
$D_i(s,o)$	The probability that the attack will be detected when the adversary is in state s and attempts attack step $a_i \in A$, and outcome $o \in O_i$ occurs
$D_i^N(s)$	The *expected path detection* when the adversary is in state s, attempts attack step $a_i \in A$, and then attempts other attack steps continuing a total of N steps into the future
$E_i(s,o)$	The *next-state* that results when the adversary is in state s and attempts attack step $a_i \in A$, and outcome $o \in O_i$ occurs
G	The set of adversary *attack goals* relevant to the system
G_s	The set of *attack goals* the adversary has achieved in state s
K	The set of *knowledge items* relevant to attacking the system
K_s	The set of *knowledge items* that the adversary possesses in state s
$L(\cdot)$	The adversary's *attack skill level function*
$\log(F_i(s))$	The *expected log nondetection* of attack step a_i in state s
$\log(F_i^N(s))$	The *expected path log nondetection* when the adversary is in state s, attempts attack step $a_i \in A$, and then attempts other attack steps continuing a total of N steps into the future
N	The adversary's *planning horizon*, i.e. the number of attack steps into the future the adversary considers when making an attack decision
O_i	The finite set of *outcomes* of attack step $a_i \in A$
$P_i(s)$	The *expected payoff* after the adversary attempts attack step $a_i \in A_s$ in state s
$P_i^N(s)$	The *expected horizon payoff* when the adversary is in state s, attempts attack step $a_i \in A$, and then attempts other attack steps continuing a total of N steps into the future
$Pr_i(s,o)$	The probability that outcome $o \in O_i$ will occur after the adversary attempts attack step $a_i \in A$ in state s
R	The set of *access domains* in the system
R_s	The set of *access domains* that the adversary can access in state s
S	The set of adversary *attack skills* relevant to attacking the system

s_0	The adversary's *initial model state*
$T_i(s)$	The length of *time* required for the adversary to attempt attack step $a_i \in A$ in state s
$U_C(\cdot)$	The adversary's *cost utility function*
$U_D(\cdot)$	The adversary's *detection utility function*
$U_F(\cdot)$	The adversary's *log nondetection utility function*
$U_P(\cdot)$	The adversary's *payoff utility function*
$V(\cdot)$	The adversary's *attack goal value function*
w_C	The adversary's *cost preference weight*
w_D	The adversary's *detection preference weight*
w_F	The adversary's *log nondetection preference weight*
w_P	The adversary's *payoff preference weight*
X	The set of all *model states*

CHAPTER 1

INTRODUCTION

1.1 Motivation for System Security Metrics

Making sound security decisions when designing, operating, and maintaining a complex system is a challenging task. Analysts need to be able to understand and predict how different factors affect the overall system security.

During system design, before the system is built, security analysts usually want to compare the security of multiple proposed system architectures. After a system is deployed, analysts try to determine where security enhancement efforts should be focused by examining how the system is most likely to be successfully penetrated. And when several security enhancement options are being considered, analysts evaluate the relative merits of each.

In each of these scenarios, quantitative security metrics could provide insight on system security and aid security decisions. Quantitative metrics enable ranking of the alternatives to determine the best option. Quantitative assessments of system security are also valuable for risk management trade-off decisions. Quantitative metrics can also help analysts understand the ability of a system to continue operating under attack in an adversarial environment.

Many low-level security metrics exist [3], but the challenge is to produce a quantitative assessment of the security of the system as a whole. This requires an understanding of how the components interact within the system. We propose to construct a security model of the system to facilitate a quantitative holistic security assessment.

An effective security model should contain system information relevant to a security analysis, including the penetrability of security-enforcing devices such as firewalls and the existence

This chapter contains previously published material by E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders [1] and by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [1] is reused by permission of ACM and was published in the *Proceedings of the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010)* `http://doi.acm.org/10.1145/1853919.1853926`. The material from [2] is reused by permission of IEEE and was published in the *Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).*

of possible connection paths between disparate parts of a system. Our method provides a formal structure to aggregate relevant system details in an organized fashion.

An effective security model should also describe the adversaries threatening the system. One key component of our approach is the inclusion of adversary attack behavior models. We assert that meaningful measurements of system security cannot occur in a vacuum devoid of information about the system's adversaries. A system defense should be evaluated in the context of the anticipated opponents.

Different systems face different adversaries. Corporate networks may be attacked by unscrupulous competitors, criminals, script kiddies, or disgruntled employees. Government-owned systems and critical infrastructure systems may attract attacks from nation-state adversaries. To produce meaningful analysis results, security should be analyzed in the context of the specific adversaries likely to attack the system.

For this adversary-behavior-modeling approach, a security model should include profiles of the adversaries that enable predictions about their likely attack behavior. Our method specifies how to generate an adversary profile containing attack goals, attack preferences, and other factors that influence attack behavior decisions. Our method also includes algorithms for making probabilistic statements about attack behavior based on an adversary profile.

1.2 Related Work Motivation

Our method produces holistic quantitative security assessments to aid system-level security design decisions by evaluating alternatives. Other tools already exist for detailed configuration analysis of deployed systems; this is not the purpose of our method.

Our method is not the first instance of model-based security analysis. Attack trees [4] have been used to describe how sets of events can constitute a security compromise. Events are represented as leaf nodes in the attack tree. Events are joined together with AND or OR nodes, and the security compromise is represented as the root node. For example, entering a locked room (the root node security compromise) might be accomplished by (1) using a stolen door key OR (2) picking the door lock (the two leaf nodes connected by an OR node). Attack trees are useful for thinking about multiple ways that an attacker can reach an attack goal. However, attack trees do not contain a notion of time, which prohibits expression of attacks as time-ordered sequences of events.

Attack graphs [5, 6, 7] and privilege graphs [8, 9, 10] extend attack trees by introducing state to the analysis. The nodes in a privilege graph represent privilege states. An attacker starts at one node in the privilege graph and works toward an attack goal by gaining privilege and transitioning to new privilege states. Attack graphs and privilege graphs enable state-based analysis, but they do not consider the different attack goals and attack preferences of individual adversaries. Our ADVISE method extends the attack graph concept by creating executable models driven by the attack preferences of individual adversaries. Thus, an analysis using our method can be customized to reflect the attack behavior of different types of adversaries with different attack objectives, attack preferences, resource levels, attack skill levels, system access, and system knowledge.

Adversary-based analysis is the focus of some other system security analysis techniques. Mission Oriented Risk and Design Analysis (MORDA) [11, 12, 13] was developed by the U. S. National Security Agency (NSA). MORDA assesses system risk by calculating attack scores for a set of system attacks. The scores are based on adversary attack preferences and the impact of the attack on the system mission. Our work in characterizing adversaries was inspired by the MORDA adversary characterization. A version of MORDA is commercially available under the name MIRROR.

The Network Risk Assessment Tool (NRAT) [14] was also developed at NSA. NRAT assesses mission risk by computing the attack competency of potential attackers and the system vulnerability. The computations are performed by examining a set of attributes about the threat actors (adversaries), the attacks, and the information system protection (defense). Each attack analyzed by NRAT is individually analyzed; there is no support for analyzing multiple-step attacks.

Neither NRAT nor MORDA is designed for state-based analysis. The adversarial decision represented in these methods is a one-time selection of a full attack vector. In contrast, our ADVISE method models step-by-step adversarial decisions, in which the outcome of the previous attack step decision impacts the next decision. The state-based nature of our approach enables insight on how attackers are likely to interact with the system as they attempt attacks.

Besides using specialized security analysis methods, security analysts could design a unique security model for each system analysis, but our method introduces a precise, repeatable technique to create state-based security models. Custom models can be expensive and slow to build, but our method makes it possible to use security metrics modeling across a much broader range of projects and project budgets. The precise format of the models also facilitates review by third parties.

1.3 Thesis Statement

There is a need for holistic quantitative security assessments of systems in order to evaluate alternatives to aid system-level security decisions.

It is our thesis that a model formalism, execution algorithm, and analysis methods can be created such that security-relevant information about system components and adversaries can be analyzed to produce a holistic quantitative evaluation of system-level security.

This dissertation contains the following contributions:

- a new adversary-driven, state-based system security evaluation method called ADversary VIew Security Evaluation (ADVISE), which includes
  - a precise adversary characterization formalism that enables simulation of how a particular adversary is likely to attack a system
  - a precise system characterization formalism that describes the possible attack paths into a system and includes security-relevant system details
  - a precise metrics specification formalism
  - a method to create executable models that can produce mission-relevant quantitative security metrics
  - an attack decision function for computing adversary attack decisions
  - an execution algorithm for simulating adversary attack decisions and attack attempt outcomes
- formulation of attack step selection as a Markov decision process,
- analysis of the performance of the attack decision function, and
- two case studies to demonstrate how the ADVISE method can be used to analyze the security of realistic systems.

1.4 ADVISE Method Overview

The approach of the ADVISE method is to create an executable state-based security model of the system. The security model is initialized with information characterizing the system and the adversaries attacking the system. The analyst specifies the security metrics of interest

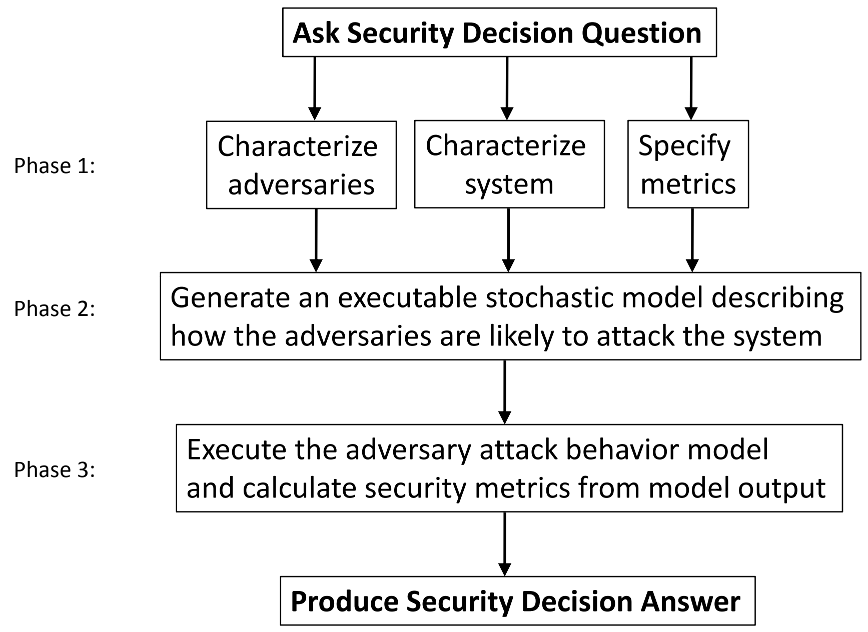

Ask Security Decision Question
Phase 1:
Characterize adversaries
Characterize system
Specify metrics
Phase 2:
Generate an executable stochastic model describing how the adversaries are likely to attack the system
Phase 3:
Execute the adversary attack behavior model and calculate security metrics from model output
Produce Security Decision Answer

Figure 1.1: The ADVISE method produces quantitative model-based system security metrics.

for the system and generates metrics data by computing numerical solutions or running discrete-event simulations of the adversaries attacking the system.

The ADVISE method for system security analysis consists of three main phases. Phase one is the characterization of the system and its adversaries and the specification of the desired security metrics. Phase two is the generation of an executable security model created from the characterization information from phase one. Phase three is the execution of the security model generated in phase two. Quantitative security metrics are produced as model outputs.

Figure 1.1 illustrates how the three phases of the ADVISE method assist security analysts in generating answers to security decision questions by incorporating information about the system, its adversaries, and the security metrics.

The ADVISE method precisely specifies the format of the system and adversary characterization data. The adversary characterization describes the system-specific attack goals of a particular adversary as well as a more general statement on how the adversary prefers to conduct attacks (e.g., risk-averse or risk-tolerant). The adversary characterization also includes ratings of the adversary's skill levels in conducting a variety of types of attacks as well as an initial assessment of what access to the system and knowledge about the system the adversary possesses before beginning any attacks.

The system characterization data in an ADVISE analysis are aggregated into an attack execution graph, which is similar to an attack graph but augmented with additional data needed to create an executable model. Attacks against a system are constructed of chains of small attack steps. Each attack step, if successfully executed, can increase the adversary's access to or knowledge of the system and move him or her closer to achieving attack goals, such as loss of confidentiality of specific data, loss of integrity of specific data, or loss of availability of specific services and/or data within the system. The attack execution graph defines and describes all possible attack steps an adversary could try against the system.

The metrics specification enables the executable model to produce output relevant to the security question posed. The metrics enable an analyst to study the average time for an adversary to reach an attack goal, the most likely attack path to an attack goal, and the average total cost to the adversary to reach an attack goal.

The executable security model simulates the adversary attack behavior. The simulation involves three computations for each attack step: the Boolean expression evaluation of the attack step precondition, the attack step attempt decision, and the attack step attempt outcome. These values are computed using the system and adversary characterization data, as well as current state information provided by the security model.

In that way, the ADVISE method produces mission-relevant (as specified by an analyst) security metrics from the simulation output.

1.5 Dissertation Organization

The remainder of this dissertation defines and explains the ADVISE model formalism and then demonstrates its use in two example system security analyses.

Chapter 2 describes the precise formalisms used for characterizing adversaries and possible attacks against systems. We introduce the attack execution graph and the adversary profile.

Chapter 3 presents the ADVISE model execution algorithm. We introduce the attack step decision function by explaining the decision computation of a short-sighted adversary. Then we explain the recursive decision computation of a long-range-planning adversary. Next we show how a discrete-event simulation algorithm uses the attack step decision function to execute and solve ADVISE models. Chapter 3 also contains a formulation of the attack step decision function as a Markov decision process. Using this formulation, we present several proofs concerning the optimality of the decision function. We then introduce an alternative decision function that we show always produces provably optimal attack decisions. We conclude the chapter with a performance analysis of the ADVISE decision computation.

Chapter 4 describes two types of ADVISE security metrics: state metrics and event metrics. We also propose conditional reporting as a technique to study correlations between different metrics. After discussing how different security analysis objectives require different security metrics, we provide a list of some example security metrics.

Chapter 5 contains two case studies demonstrating the use of the ADVISE method to perform system security analysis. The first case study evaluates the security of two variants of a generic supervisory control and data acquisition (SCADA) system architecture. The second case study examines the security of an electric power distribution system. We study the attack paths of six different adversaries and identify a commonly selected attack step. We also demonstrate how an ADVISE model can be linked with other types of discrete-event simulation models so that we can include a system repair model or a system impact model in the security analysis.

In Chapter 6, we compare ADVISE with other related work. We also discuss the benefits, limitations, and assumptions of ADVISE. In Chapter 7, we conclude with a discussion of possible applications and extensions of the ADVISE method. Appendix A describes the tool implementation of the ADVISE method. This tool was used to run simulations and collect results for the case studies.

The basic analysis philosophy of the ADVISE method was first published as a "fast abstract" at the 2010 International Conference on Dependable Systems and Networks (DSN 2010) [15]. A more detailed description was published as a paper at the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010) [1]. After the tool was implemented, initial case study results were published at the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011) [2]. The QEST 2011 paper also contains a more formal definition of the ADVISE formalism than earlier publications.

The material in the QEST 2011 paper [2] is included in this dissertation in Chapter 1 (specifically, Section 1.2), Chapter 2, Chapter 3 (specifically, Sections 3.1 and 3.2), Chapter 4 (specifically, Sections 4.1, 4.2, and 4.3), Chapter 5 (specifically, Section 5.1) and Chapter 7. Material from the MetriSec 2010 paper [1] is included in Chapter 1 (specifically, Sections 1.1 and 1.4).

CHAPTER 2

ADVISE FORMALISM DEFINITION

A system security analysis using the ADVISE method begins by specifying an attack execution graph (AEG) to represent all potential attack steps against the system and specifying an adversary profile. The ADVISE model formalism collects and organizes specific information about possible attacks and adversaries. This information is then used to automatically generate an executable model that represents how the adversary is likely to attack the system.

2.1 Attack Execution Graph Definition

A security analyst builds an AEG by thinking about attacks in terms of many small attack steps. Each attack step achieves some attack goal or makes progress toward an attack goal by changing the adversary's access to or knowledge of the system. For example, one attack step could be to obtain a user password (gaining knowledge); another attack step could be to log in to an internal network (gaining access); still other attack steps could allow an adversary to crash a server or read proprietary information (achieving attack goals).

An *attack execution graph* is defined by the tuple

$$\langle A, R, K, S, G \rangle, \tag{2.1}$$

where A is the set of *attack steps* against the system, R is the set of *access domains* in the system, K is the set of *knowledge items* relevant to attacking the system, S is the set of adversary *attack skills* relevant to attacking the system, and G is the set of adversary *attack goals* relevant to the system.

In a pictorial representation of an AEG (see Figure 2.1), the attack steps are rectangular boxes, the access domains are squares, the knowledge items are circles, the attack skills are

This chapter contains previously published material by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [2] is reused by permission of IEEE and was published in Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).

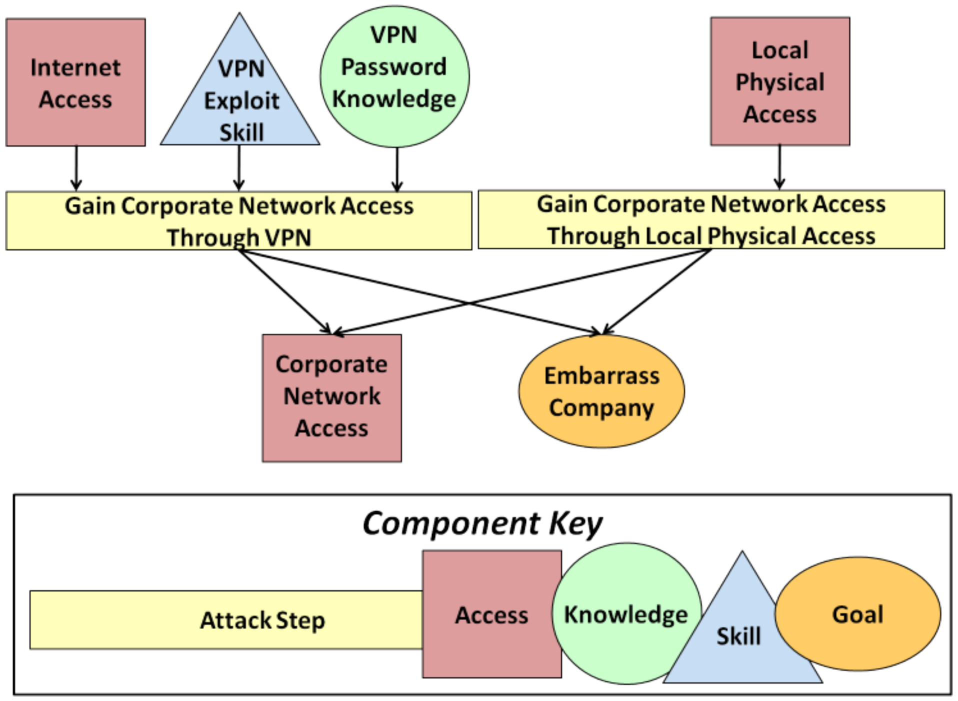


Figure 2.1: An attack execution graph (AEG) represents possible attacks against a system that an adversary can use to reach his or her attack goals.

triangles, and the attack goals are ovals. In this simple AEG, the attack goal is to embarrass the company by gaining corporate network access. There are two attack approaches represented by the two attack steps: "Gain Corporate Network Access Through VPN" and "Gain Corporate Network Access Through Local Physical Access." The objects with arrows pointing towards an attack step are the access, skills, knowledge, or goals that are relevant to determining if the adversary can attempt the attack step. For example, the adversary must first possess "Local Physical Access" before attempting the latter attack step. The objects with arrows pointing away from the attack step are the access, knowledge, or goals that can be affected when an adversary attempts the attack step. For example, either attack step in Figure 2.1 can enable the adversary to obtain "Corporate Network Access" and achieve the "Embarrass Company" goal.

In more complex AEGs, attack steps are joined through the access, knowledge, and goal objects to form a graph-like structure, as shown in the case study AEG (Figure 5.3).

2.2 Attack Step Definition

Each *attack step* $a_i \in A$ in the AEG is defined in such a way that the AEG can be converted later into an executable model. Formally, an attack step a_i is a tuple:

$$a_i = \langle B_i, T_i, C_i, O_i, Pr_i, D_i, E_i \rangle. \tag{2.2}$$

The components of an attack step are defined as functions of model state $s \in X$, where X is the set of all model states (to be defined later).

$B_i : X \to \{True, False\}$ is a Boolean precondition. The precondition evaluates to true if the adversary possesses what is required to attempt the attack step (specific access, knowledge, and/or attack skills) but does not already possess what can be gained by successfully completing this attack step (specific access, knowledge, and/or attack goals). For example, to attempt the attack step "Gain Corporate Network Access Through VPN" in Figure 2.1, the precondition requires that the adversary have Internet access and either the skill to exploit the VPN or knowledge of a VPN account password. The precondition also specifies that the adversary will not attempt this attack step if he or she already possesses "Corporate Network Access."

$T_i : X \times \mathbf{R}^+ \to [0, 1]$ is the length of time required to attempt the attack step. $T_i(s)$ is a random variable defined by a probability distribution function over the positive real numbers.

$C_i : X \to \mathbf{R}^{\geq 0}$ is the cost of attempting the attack step (regardless of the outcome).

O_i is the finite set of outcomes. For many attack steps, the set of outcomes contains two elements: success and failure.

$Pr_i : X \times O_i \to [0, 1]$ is the probability that outcome $o \in O_i$ will occur after the attack step is attempted, where $\sum_{o \in O_i} Pr_i(s, o) = 1$ for all s. System defenses and countermeasures can affect the outcome probabilities.

$D_i : X \times O_i \to [0, 1]$ is the probability that the attack will be detected when outcome $o \in O_i$ occurs.

$E_i : X \times O_i \to X$ is the next-state that results when outcome $o \in O_i$ occurs.

Every AEG contains a "do-nothing" attack step, a_{DN}. The do-nothing attack step represents the option of an adversary to refrain from attempting any active attack against the system. The precondition B_{DN} is always true. The time, T_{DN}, is the period of time that elapses before the adversary reconsiders the decision to do nothing, which depends on the particular adversary. For most AEGs, the cost C_{DN} is zero, the detection probability D_{DN}

is zero, the next-state is the same as the current state ($E_{DN}(s, o) = s$, $\forall s \in X$), there is only one outcome $o \in O_{DN}$, and the probability of that outcome, $Pr_{DN}(s, o)$, is one. The existence of the do-nothing attack step means that, regardless of the model state, there is always at least one attack step in the AEG whose precondition is satisfied.

2.3 Model State Definition

The *model state*, $s \in X$, reflects the progress of the adversary in attacking the system and is defined by

$$s = \langle R_s, K_s, G_s \rangle, \tag{2.3}$$

where $R_s \subseteq R$ is the set of access domains that the adversary can access, $K_s \subseteq K$ is the set of knowledge items that the adversary possesses, and $G_s \subseteq G$ is the set of attack goals the adversary has achieved. X is the finite set of all model states.

2.4 Adversary Profile Definition

The *adversary profile* is defined by the tuple

$$\langle s_0, L, V, P, w_C, w_P, w_D, U_C, U_P, U_D, N \rangle, \tag{2.4}$$

where $s_0 \in X$ is the initial model state; L is the attack skill level function; V is the attack goal value function; P is the payoff value function; w_C, w_P, and w_D are the attack preference weights for cost, payoff, and detection probability, respectively; U_C, U_P, and U_D are the utility functions for cost, payoff, and detection probability, respectively; and N is the planning horizon.

The *initial model state*, s_0, describes the starting point of the adversary's attack. An insider adversary will likely start with more access and knowledge than an outsider.

The *attack skill level function*, $L : S \to [0, 1]$, describes the attack proficiency of the adversary by mapping each attack skill in the AEG to a value in the range $[0, 1]$. A greater skill level is represented by a larger value. To provide a reference for assigning consistent skill level ratings, an analyst-defined rubric could be developed for each attack skill.

The *attack goal value function*, $V : G \to \mathbf{R}^{\geq 0}$, describes the monetary-equivalent value of each attack goal in the AEG from the viewpoint of the adversary. Each attack goal is assigned a nonnegative real number. A more valuable goal is assigned a larger value. The *payoff value function*, $P : X \to \mathbf{R}^{\geq 0}$, describes the total value of all goals achieved in model state $s \in X$. For example, the payoff value of a model state could be the sum of the values of all goals achieved in that state.

The *attack preference weights*, $w_C, w_P, w_D \in [0, 1]$, describe the relative attractiveness of incremental changes in each of the three core criteria that adversaries consider when deciding among their attack options. The weight w_C is the relative attractiveness of decreasing the cost to the adversary in attempting the attack step. The weight w_P is the relative attractiveness of increasing the payoff to the adversary for successfully executing the attack step. The weight w_D is the relative attractiveness of decreasing the probability of being detected by the system during or after attempting the attack step.

The *utility functions* (U_C, U_P, and U_D) map the native values of each attractiveness criterion to a $[0, 1]$ utility scale, where higher utility values represent more desirable values. The *cost utility function*, $U_C : \mathbf{R}^{\geq 0} \to [0, 1]$, maps the monetary value of the attack step cost to its utility according to the adversary. Because lower costs are more desirable (have a higher utility), the cost utility function is a decreasing function. The *payoff utility function*, $U_P : \mathbf{R}^{\geq 0} \to [0, 1]$, maps the monetary value of the attack step payoff to its utility according to the adversary. The payoff utility function is an increasing function. The *detection utility function*, $U_D : [0, 1] \to [0, 1]$, maps the probability of attack step detection to its utility according to the adversary. Because lower detection probabilities are more desirable (have a higher utility), the detection utility function is a decreasing function.

The *planning horizon*, N, is the number of steps into the future the adversary can consider when making an attack decision. (The use of the planning horizon is explained in Section 3.1.2.)

CHAPTER 3

ADVISE EXECUTION ALGORITHM

The ADVISE model formalism enables the creation of executable models. An adversary profile is coupled with an attack execution graph to produce an executable model that represents how the adversary is likely to attack the system. ADVISE executable models can be analyzed using discrete-event simulation.

The executable model includes the initial model state and the functions that govern state transitions. When the model is analyzed using discrete event simulation, the execution algorithm determines the sequence of state transitions that occur during a simulation run. Taking into account the current model state and the adversary's attack preferences, the attack decision function mimics how an adversary chooses the next attack step. The outcome of that attempt (whether the attack step succeeds or fails) determines the next state of the model. This process repeats with the attack decision function choosing the next attack step.

The ADVISE model execution functions consist of the attack step decision function and the attack step outcomes. The decision function produces a deterministic attack decision. Given the same adversary attack preferences (defined in the adversary profile), the same attack execution graph, and the same current model state, the decision function will always select the same next attack step. (The exception is when multiple attack steps have exactly the same attractiveness value; in this case, the decision function selects randomly from among the attack steps with maximal attractiveness.)

In contrast, the attack step outcomes are stochastic. In the attack execution graph, an attack step definition includes the probability of each outcome given that the step is attempted. An attack step outcome is pseudo-randomly generated using those probability distributions. The attack step outcomes determine the sequence of state transitions.

This chapter contains previously published material by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [2] is reused by permission of IEEE and was published in Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).

3.1 Attack Step Selection

The attack step decision function is based on an evaluation of the attractiveness of all the available attack steps from the viewpoint of the adversary and the selection of a most attractive attack step. The decision is based on the current model state s.

First, the decision function checks the precondition, $B_i(s)$, of each attack step, a_i, in the attack execution graph. Recall that the precondition specifies the access, knowledge, and skill that the adversary must possess (as well as the access, knowledge, and skill that the adversary must not possess) to attempt the attack step. Attack steps whose preconditions are satisfied based on the current model state s comprise the set of *available attack steps* A_s:

$$A_s = \{a_i \in A | (B_i(s) = \mathit{True})\}, \tag{3.1}$$

where A is the set of all attack steps.

Next, the decision function evaluates the attractiveness of each available attack step using three decision criteria: the cost of attempting the attack, the expected probability of detection, and the expected payoff in the state reached after the attack. The adversary profile contains attack preference weights that reflect the relative importance of incremental changes in these three decision dimensions.

We begin by explaining the attack step decision function for a short-sighted adversary. This simplified adversary serves to introduce concepts we will extend in our explanation of the long-range-planning adversary.

3.1.1 The Short-Sighted Adversary

In a short-sighted attack, the adversary only considers the immediate next attack steps and the immediate next states that could result from those next steps. The attractiveness, $\mathit{attr}(a_i, s)$, of available attack step $a_i \in A_s$ based on current model state s is a linear combination of the adversary attack preference weights with the data about the attack step:

$$\mathit{attr}(a_i, s) = w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(P_i(s)) + w_D \cdot U_D(D_i(s)), \tag{3.2}$$

where $C_i(s)$ is the cost of attempting attack step a_i from state s, w_C is the adversary preference weight for cost, $P_i(s)$ is the expected payoff after attempting attack step a_i from state s, w_P is the adversary preference weight for payoff, $D_i(s)$ is the expected probability of detection associated with attempting attack step a_i from state s, and w_D is the adversary preference weight for detection. $U_C(\cdot), U_P(\cdot)$, and $U_D(\cdot)$ are the adversary's utility functions for cost, payoff, and detection, respectively.

The adversary's utility functions ($U_C(\cdot), U_P(\cdot)$, and $U_D(\cdot)$) map the native values of each attractiveness criterion to a $[0, 1]$ utility scale, where higher utility values represent more desirable values. The utility functions convert cost, payoff, and detection into an common unit of "utility" so that the weighted average computation (in the attractiveness function) is mathematically valid.

$P_i(s)$, the expected payoff after attempting attack step a_i from state s, is computed as the sum of the payoff values in each possible next-state weighted by the probability of the attack step outcome leading to that next-state:

$$P_i(s) = \sum_{o \in O_i} (P(E_i(s, o)) \cdot Pr_i(s, o)), \tag{3.3}$$

where $P(E_i(s, o))$ is the payoff in next-state $E_i(s, o)$ (as defined in Section 2.4).

$D_i(s)$, the expected detection probability after the adversary attempts attack step a_i from state s, is computed similarly:

$$D_i(s) = \sum_{o \in O_i} (D_i(s, o) \cdot Pr_i(s, o)). \tag{3.4}$$

Then, for current model state s, the attack step decision function selects a best next attack step, $\beta(s)$, as an available attack step with the maximal attractiveness value:

$$\beta(s) \in \{a^* \in A_s | attr(a^*, s) = \max_{a_i \in A_s} attr(a_i, s)\}. \tag{3.5}$$

If multiple attack steps have the same maximal attractiveness value, then the attack step decision function uniformly selects one attack step from the set of maximally attractive attack steps.

3.1.2 The Long-Range-Planning Adversary

Equation (3.5) describes an attack decision based on information about the immediate next attack step options and the resulting immediate next states; however, most real adversaries do not attempt multi-step attacks by plunging blindly ahead looking only one step into the

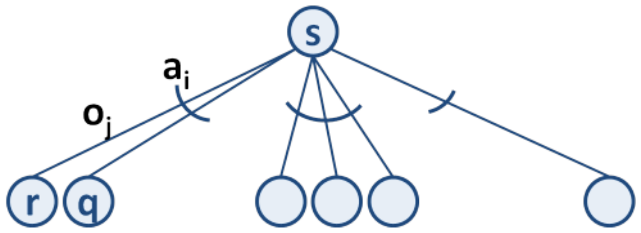


Figure 3.1: A state look-ahead tree (SLAT) explores all possible outcomes of all possible attack steps from the root node (state s) to determine all possible next-states. Each next-state becomes a child node in the SLAT.

future. Modeling a more sophisticated adversary decision requires a long-range-planning attack step decision function. The adversary considers what goals can be achieved using a sequence of attack steps. We modify Equations (3.2)–(3.5) to model the goal-driven attack step selection of a long-range-planning adversary.

The planning horizon, N, is the number of steps into the future the adversary can consider when making an attack decision. The planning horizon is analogous to the number of moves a chess player can think ahead when planning his next move. The chess player must consider what he can directly control (his own future moves), as well as what he cannot control (the possible future moves of his opponent). Similarly, the adversary in our model must consider both what he can control (his own attack step decisions), as well as what he cannot control (the outcome of an attack attempt).

To analyze all possible sequences of attack steps of length N, we introduce the state look-ahead tree (SLAT). The root node of the SLAT is the current state (labeled as state s in Figure 3.1). The child nodes are the possible next states of the parent state. Each available attack step in the parent state produces one or more child nodes. Each possible outcome of an attack step produces one child node. All child nodes produced by the same attack step are grouped together by a hash mark. In Figure 3.1, attack step a_i is an available attack step in state s, and outcome o_j of attack step a_i results in the model state transitioning to state r. Construction of the SLAT continues with the exploration of the available attack steps in the leaf node states of the SLAT and the addition of next-state child nodes that become the new leaf nodes. This tree-building process continues until the leaf nodes are N attack steps distant from the root node. Because the "do-nothing" attack step is an available attack step in any state, we can always build a tree with leaf nodes N attack steps distant from the root node. In the example SLAT in Figure 3.2, the planning horizon, N, is two.

After the top-down construction of the SLAT, the bottom-up, best-next-step analysis

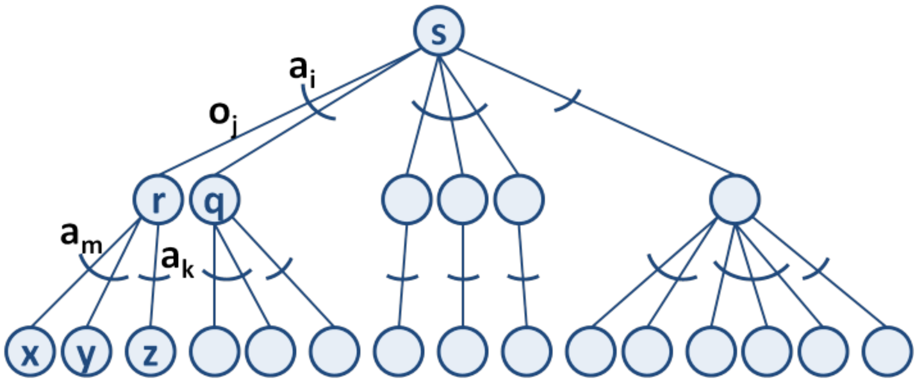


Figure 3.2: Construction of the SLAT continues with the exploration of the available attack steps in the leaf node states of the SLAT and the addition of next-state child nodes until the leaf nodes are N attack steps distant from the root node. Here, $N = 2$.

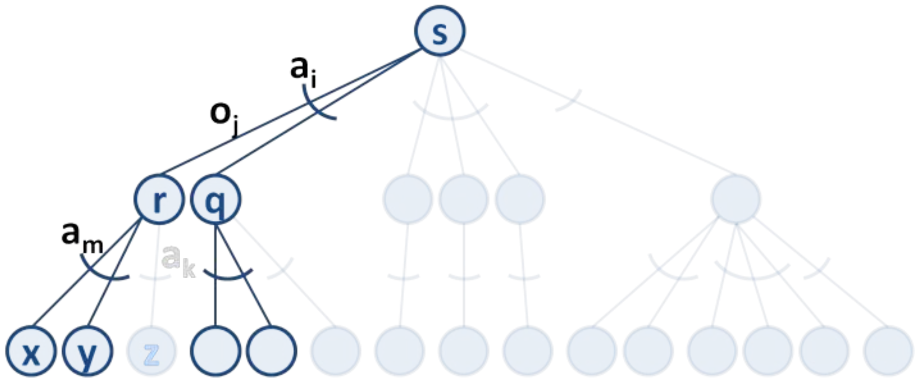


Figure 3.3: After the top-down construction of the SLAT, the bottom-up best next-step analysis prunes off branches with non-maximal attractiveness. For each state in the pruned SLAT, there is only one attack step group left; this attack step is the best next attack step at that state, considering all attack step data between that state and the planning horizon states.

begins. To evaluate the attractiveness of attack step a_i with a planning horizon of two, the decision function needs to consider not only the cost, detection probability, and payoff of a_i itself, but also the cost, detection probability, and payoff of the attack step that will be performed after a_i. Assume that attempting attack step a_i could transition the model state to state r or state q. Considering state r, the decision function needs to evaluate the attractiveness of attack steps a_m and a_k to determine the best attack step in state r. For illustration, let the best attack step from state r be a_m. The cost, detection probability, and payoff of a_m are reported back to be used in the attractiveness calculation for attack step a_i. The other available attack steps (here, a_k) are pruned from the tree. In this way, attack decisions made at the bottom of the SLAT determine which cost, detection probability, and payoff values are factored into the attack decisions higher in the SLAT. A pruned SLAT is shown in Figure 3.3. For each state in the pruned SLAT, there is only one attack step group left; this attack step is the best next attack step at that state taking into consideration all attack step data between that state and the planning horizon states. (The exception is when multiple attack steps have exactly the same attractiveness value; in this case, the decision function selects one randomly from among the attack steps with maximal attractiveness.)

More formally, the best next attack step, $\beta^N(s)$, is uniformly selected from among the available attack steps in model state s with the maximal attractiveness value, where the attractiveness value is computed recursively using a planning horizon of N:

$$\beta^N(s) \in \{a^* \in A_s | attr^N(a^*, s) = \max_{a_i \in A_s} attr^N(a_i, s)\}. \tag{3.6}$$

The attractiveness of an available attack step a_i in state s is computed recursively in a way that evaluates all possible sequences of attack steps of length N that begin with attack step a_i:

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)) + w_D \cdot U_D(D_i^N(s)), \tag{3.7}$$

where $C_i^N(s)$ is the recursively computed *expected path cost*, w_C is the adversary preference weight for cost, $P_i^N(s)$ is the recursively computed *expected horizon payoff*, w_P is the adversary preference weight for payoff, $D_i^N(s)$ is the recursively computed *expected path detection*, and w_D is the adversary preference weight for detection. $U_C(\cdot), U_P(\cdot)$, and $U_D(\cdot)$ are the adversary's utility functions for cost, payoff, and detection, respectively.

The expected horizon payoff is based on the states that can be reached at the edge of the planning horizon; the expected path cost and expected path detection are based on the attack steps between the current state and the states at the edge of the planning horizon.

The expected path cost, $C_i^N(s)$, is the expected sum of attack step costs starting from state s with attack step a_i and continuing a total of N steps into the future:

$$C_i^N(s) = \begin{cases} C_i(s), & \text{when } N = 1, \\ C_i(s) + \sum_{o \in O_i} (C_*^{N-1}(r) \cdot Pr_i(s,o)), & \text{when } N > 1, \end{cases} \tag{3.8}$$

where state $r = E_i(s,o)$ is the next state after starting in state s, attempting attack step a_i, and obtaining outcome o. The definition of state r is also used in Equations (3.9)–(3.13).

The recursive definition for expected path cost requires determining the best next attack step from state r with planning horizon $(N-1)$:

$$C_*^{N-1}(r) \equiv C_k^{N-1}(r) \text{ when } \beta^{N-1}(r) = a_k. \tag{3.9}$$

The expected horizon payoff, $P_i^N(s)$, is the expected payoff in the state reached after attempting attack step a_i and continuing a total of N steps into the future:

$$P_i^N(s) = \begin{cases} \sum_{o \in O_i} (P(E_i(s,o)) \cdot Pr_i(s,o)), & \text{when } N = 1, \\ \sum_{o \in O_i} (P_*^{N-1}(r) \cdot Pr_i(s,o)), & \text{when } N > 1. \end{cases} \tag{3.10}$$

Just as for the expected path cost, the computation of the expected horizon payoff requires determining the best next attack step from state r with planning horizon $(N-1)$:

$$P_*^{N-1}(r) \equiv P_k^{N-1}(r) \text{ when } \beta^{N-1}(r) = a_k. \tag{3.11}$$

The expected path detection, $D_i^N(s)$, is the expected probability of detection at any point during attack step a_i or the other attack steps, continuing a total of N steps into the future.

$$D_i^N(s) = \begin{cases} \sum_{o \in O_i} (D_i(s,o) \cdot Pr_i(s,o)), & \text{when } N = 1, \\ \sum_{o \in O_i} ((1 - (1 - D_i(s,o)) \cdot (1 - D_*^{N-1}(r))) \cdot Pr_i(s,o)), & \text{when } N > 1. \end{cases} \tag{3.12}$$

Just as for the expected path cost and expected horizon payoff, the computation of the expected path detection requires determining the best next attack step from state r with planning horizon $(N-1)$:

$$D_*^{N-1}(r) \equiv D_k^{N-1}(r) \text{ when } \beta^{N-1}(r) = a_k. \tag{3.13}$$

Note that when the planning horizon N is one, Equation (3.7) is equivalent to Equation (3.2).

As N grows, the time needed to perform the recursive attractiveness computation will increase significantly due to exponential growth in the number of states to explore while constructing the SLAT. However, for small values of N, the computation is tractable.

For the practical implementation of the recursive attractiveness computation, the SLAT is constructed and pruned using an in-order traversal. An in-order traversal approach limits the amount of space required for the attractiveness computation because only a small portion of the SLAT is stored in memory at a time.

3.2 Model Simulation

When an ADVISE model is solved using discrete-event simulation, the initial model state is s_0 from the adversary profile, and the state transitions are governed by the attack step decision function $\beta^N(s)$, the outcome probability distributions Pr_i, and the next-state functions E_i. The attack step decision function chooses one attack step; the outcome probability distributions are used to randomly select one outcome of that attack step; and the next-state function of that attack step outcome determines the next state. The model state changes to that next state, and then the state transition process repeats.

During initialization, the simulation time is set to zero. Each attack step attempt advances the simulation clock by an amount of time determined by a random sample from the attack step execution time distribution defined by T_i. The simulation ends when the simulation clock reaches a specified simulation end time τ.

The ADVISE model execution algorithm is shown in Algorithm 3.1. The outcome probability distribution function $Prob_i(\text{State})$ refers to a discrete probability distribution, in which each $o \in O_i$ has probability $Pr_i(\text{State}, o)$, as defined in the attack step definition.

Algorithm 3.1 ADVISE Model Execution

```
1: Time $\Leftarrow 0$
2: State $\Leftarrow s_0$
3: while Time $< \tau$ do
4:     Attack$_i \leftarrow \beta^N$(State)
5:     Outcome $\leftarrow o$, where $o \sim Prob_i$(State)
6:     Time $\leftarrow$ Time + $t$, where $t \sim T_i$(State)
7:     State $\leftarrow E_i$(State,Outcome)
8: end while
```

3.3 Formulation of Attack Step Selection as a Markov Decision Process

ADVISE attack step selection can be thought of as a stochastic security game in which one player (the defense) has a fixed strategy and the other player (the adversary) faces an optimization problem that can be characterized as a Markov decision process. In the language of a stochastic security game, the defense's fixed strategy is reflected in the outcome probabilities for each attack step. The stronger the defense's protection against an attack step, the smaller the adversary's probability of success for that attack step.

Each attack decision of the adversary is a Markov decision process. A Markov decision process is a "model for sequential decision-making under uncertainty," considering "both the outcomes of current decisions and future decision-making opportunities" [16].

In a Markov decision process, a decision is made at each epoch $t \in T$. At each epoch, the system is in a state $s \in S$, and the decision-maker selects an action from the set of allowable actions A_s. The reward function specifies the reward that the decision-maker receives for selecting action $a_i \in A_s$ when the system is in state s. The system state at the next decision epoch is determined by the transition probability function.

The ADVISE adversary attack decision is a finite-horizon Markov decision process defined by the following:

- A set of discrete-time *decision epochs*, $T = \{1, 2, \ldots, M, M + 1\}$, where $M + 1$ is the planning horizon N from the ADVISE formalism definition;
- A finite set of *states*, S, where $s \in S$ is defined by $s = \langle R_s, K_s, G_s \rangle$ from the ADVISE formalism definition;
- A finite set of allowable *actions* in state s, $A_s = \{a_i \in A | (B_i(s) = \textit{True})\}$ from the ADVISE attack step selection;

- The stationary *reward function* (the attractiveness function for $N = 1$), $attr(a_i, s) = w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(P_i(s)) + w_D \cdot U_D(D_i(s))$ from the ADVISE attack step selection; and

- A stationary *transition probability function* defined by $Pr_i : S \times O_i \to [0,1]$, the probability that outcome $o \in O_i$ occurs after the attack step is attempted, where $\sum_{o \in O_i} Pr_i(s, o) = 1$ for all s, and $E_i : S \times O_i \to S$, the next-state that results when outcome $o \in O_i$ occurs (both from the ADVISE attack step definitions).

A Markov decision process together with an optimality criterion forms a Markov decision problem.

3.3.1 Optimal Policies

A *decision rule* specifies how to select an action at a decision epoch. A *policy* specifies the decision rule at all decision epochs. A decision-maker implementing a policy receives a sequence of rewards based on the action selected at each decision epoch and the next-state that results. Because the next-state transition is stochastic, the reward sequence is random. One way to compare random reward vectors is to examine the expected total reward, which is the expected sum of rewards received at each decision epoch. The expected total reward criterion enables a total ordering of all random reward vectors (i.e., all random reward vectors are comparable).

Under the expected total reward optimality criterion, an *optimal policy* is a policy whose expected total reward is greater than or equal to the expected total reward of all possible policies.

The SLAT (introduced in Section 3.1.2) explores all possible outcomes of all possible attack steps from a given state to determine all possible next-states, building a tree to the depth of the planning horizon. A policy selects one attack step at each decision epoch based on the state at that decision epoch, and the other attack steps that are not selected are pruned from the SLAT. Though the attack step selection is deterministic, there is still a stochastic outcome, so several different next-states may be possible. In each possible next-state resulting from the selected action, the policy must then select another next attack step. Thus, a policy equates to the selection of a subtree of the SLAT (using the graph theory definition of a subtree as a subset of the vertices and edges of the full tree). Although each possible policy can be represented as a subtree of the SLAT, not all subtrees represent valid policies.

The number of all possible policy subtrees of a SLAT can be expressed using the recurrence relation

$$Q_N = Y \cdot (Q_{N-1})^Z \tag{3.14}$$

with $Q_1 = Y$, where Y is the number of attack steps available at each state, Z is the number of outcomes of each attack step, and N is the planning horizon. We first consider a SLAT with planning horizon 1. At the root node, the adversary can select one out of the Y available attack steps, so there are exactly Y possible policy subtrees. Hence, $Q_1 = Y$. We next consider a SLAT with a planning horizon of N. At the root node, the adversary chooses one of the Y available attack steps. For each choice of attack step at the root node, there are Z possible outcomes (equivalently, there are Z possible next-states). Each of the next-states is the root node of a policy subtree with a planning horizon of $(N-1)$. A single policy subtree for a SLAT with planning horizon N must first select one out of the Y available attack steps and then, for each possible outcome of that selected attack step, select one policy subtree rooted at the next-state resulting from that outcome. The number of all possible policy subtrees for a SLAT with a planning horizon of $(N-1)$ is expressed as Q_{N-1}. Thus, the number of all possible policy subtrees of a SLAT can be described by the recurrence relation in Equation (3.14). The closed-form solution of this recurrence relation can be expressed as

$$Q_N = Y^{\sum_{i=0}^{N-1} Z^i}. \tag{3.15}$$

As N increases, direct evaluation of the expected total reward of all possible policies quickly becomes intractable.

However, when the reward function has a certain structure (linear additive or exponential multiplicative), a multistage Markov decision problem can be solved as a sequence of single-stage problems using dynamic programming (also called *backwards induction*). Under those conditions, an optimal policy can be found using a recursive scheme that does not require enumeration and evaluation of all possible policies.

Bellman's principle of optimality states that "an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [17].

The Bellman optimality equations give the optimal expected total reward from decision epochs $t, t+1, ..., M$:

$$u_t^*(s_t) = \sup_{a \in A_{s_t}} \{r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j)\}, \text{ for } t = 1, ..., (M-1), \tag{3.16}$$

and

$$u_M^*(s_M) = r_M(s_M), \tag{3.17}$$

where s_t is the state at decision epoch t; A_{s_t} is the set of all allowable actions in state s_t; $r_t(s_t, a)$ is the reward obtained from performing action a in state s_t; S is the set of all possible next-states given state s_t and action a; and $p_t(j|s_t, a)$ is the probability of next-state j given state s_t and action a.

3.3.2 Evaluating the Optimality of the ADVISE Policy

The ADVISE attack step selection process uses a stationary deterministic Markovian decision rule,

$$\beta^N(s) = \underset{a_i \in A_s}{\arg\max}\{\mathit{attr}^N(a_i, s)\}, \tag{3.18}$$

where

$$\mathit{attr}^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)) + w_D \cdot U_D(D_i^N(s)),$$

w_C is the adversary preference weight for cost; $C_i^N(s)$ is the recursively computed *expected path cost* (that is, the expected sum of attack step costs incurred from decision epoch t to decision epoch $t + N - 1$, given state s and action a_i in decision epoch t); $U_C(\cdot)$ is the cost utility function, converting cost to units of utility; w_P is the adversary preference weight for payoff; $P_i^N(s)$ is the recursively computed *expected horizon payoff* (that is, the expected payoff in decision epoch $t+N-1$, given state s and action a_i in decision epoch t); $U_P(\cdot)$ is the payoff utility function, converting payoff to units of utility; w_D is the adversary preference weight for detection; $D_i^N(s)$ is the recursively computed *expected path detection* (that is, the expected probability of detection from decision epoch t to decision epoch $t + N - 1$, given state s and action a_i in decision epoch t); and $U_D(\cdot)$ is the detection utility function, converting detection to units of utility.

We examine the conditions under which application of this decision rule constitutes an optimal policy.

Taken individually, cost is a linear additive reward, detection is a multiplicative exponential reward, and payoff is a linear additive reward in which the additive payoff is nonzero only in the N-th decision epoch. The ADVISE decision rule combines these three components into one function that contains both linear additive and multiplicative exponential rewards. The ADVISE decision rule is provably optimal when the reward function becomes either wholly linear additive or wholly multiplicative exponential.

We first show that the ADVISE decision rule is provably optimal when the reward function becomes wholly linear additive (see Theorem 3.1). We also show that the ADVISE decision rule is provably optimal when the reward function is wholly multiplicative exponential (see Theorem 3.2). Then we provide a counterexample that shows that the ADVISE decision rule does not always implement an optimal decision when the reward function contains both linear additive and multiplicative exponential rewards (see Theorem 3.3). These results motivate us to modify the original function to develop an alternative attractiveness function (see Section 3.4) that combines cost, payoff, and detection criteria in such a way that the decision is always provably optimal. We show that the alternative attractiveness function is always provably optimal (see Theorem 3.4).

Theorem 3.1. *Let $w_D = 0$. Let the cost and payoff utility functions be linear functions of the forms $U_C(x) = m_C x + b_C$ and $U_P(x) = m_P x + b_P$. Then, the ADVISE decision rule (Equation (3.18)) implements an optimal policy.*

Proof. When w_D is zero, the attractiveness function is wholly linear additive:

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)). \tag{3.19}$$

The equations for expected path cost (3.8) and expected horizon payoff (3.10) are substituted into Equation (3.19).

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{when } N = 1, \\ w_C \cdot U_C(C_i(s) + \sum_{o \in O_i} (C_*^{N-1}(r) \cdot Pr_i(s,o))) \\ \quad + w_P \cdot U_P(\sum_{o \in O_i} (P_*^{N-1}(r) \cdot Pr_i(s,o))), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{when } N > 1. \end{cases}$$

Next, we replace the utility function names with their slope-intercept representations: $U_C(x) = m_C x + b_C$ and $U_P(x) = m_P x + b_P$. For the remainder of the proof, we manipulate only the function valid for N greater than one.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot (m_C(C_i(s) + \sum_{o \in O_i}(C_*^{N-1}(r) \cdot Pr_i(s,o))) + b_C) \\ \quad + w_P \cdot (m_P(\sum_{o \in O_i}(P_*^{N-1}(r) \cdot Pr_i(s,o))) + b_P), \\ \hfill \text{when } N > 1. \end{cases}$$

Then we separate the terms.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) + w_C \cdot m_C \cdot (\sum_{o \in O_i}(C_*^{N-1}(r) \cdot Pr_i(s,o))) + w_C \cdot b_C \\ \quad + w_P \cdot m_P \cdot (\sum_{o \in O_i}(P_*^{N-1}(r) \cdot Pr_i(s,o))) + w_P \cdot b_P, \\ \hfill \text{when } N > 1. \end{cases}$$

Because $\sum_{o \in O_i} Pr_i(s,o) = 1$, we can introduce this quantity in the b_C and b_P terms so that we can combine terms more easily later.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad + w_C \cdot m_C \cdot \sum_{o \in O_i}(C_*^{N-1}(r) \cdot Pr_i(s,o)) + w_C \cdot b_C \cdot \sum_{o \in O_i} Pr_i(s,o) \\ \quad + w_P \cdot m_P \cdot \sum_{o \in O_i}(P_*^{N-1}(r) \cdot Pr_i(s,o)) + w_P \cdot b_P \cdot \sum_{o \in O_i} Pr_i(s,o), \\ \hfill \text{when } N > 1. \end{cases}$$

Next, we move the preference weights and the slope and intercept terms inside the summations.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum\limits_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad + \sum\limits_{o \in O_i} (w_C \cdot m_C \cdot C_*^{N-1}(r) \cdot Pr_i(s,o)) + \sum\limits_{o \in O_i} (w_C \cdot b_C \cdot Pr_i(s,o)) \\ \quad + \sum\limits_{o \in O_i} (w_P \cdot m_P \cdot P_*^{N-1}(r) \cdot Pr_i(s,o)) + \sum\limits_{o \in O_i} (w_P \cdot b_P \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

We combine the summation terms into one summation.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum\limits_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad + \sum\limits_{o \in O_i} (((w_C \cdot m_C \cdot C_*^{N-1}(r)) + (w_C \cdot b_C) \\ \qquad\qquad + (w_P \cdot m_P \cdot P_*^{N-1}(r)) + (w_P \cdot b_P)) \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

Recognizing the linear utility functions, we replace them with the function names U_C and U_P.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum\limits_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad + \sum\limits_{o \in O_i} (((w_C \cdot U_C(C_*^{N-1}(r))) \\ \qquad\qquad + (w_P \cdot U_P(P_*^{N-1}(r)))) \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

The attractiveness of the optimal attack step one step in the future is $attr^{(N-1)}(a_\beta, r) = w_C \cdot U_C(C_*^{(N-1)}(r)) + w_P \cdot U_P(P_*^{(N-1)}(r))$, where $r = E_i(s,o)$ is the next-state given state s, attack step a_i, and outcome o; and a_β is the most attractive attack step in state r.

We perform the following substitution.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))), & \\ \qquad\qquad\qquad \text{when } N = 1, & \\ w_C \cdot m_C \cdot C_i(s) + \sum_{o \in O_i} (attr^{(N-1)}(a_\beta, r)) \cdot Pr_i(s,o)), & \\ \qquad\qquad\qquad \text{when } N > 1. & \end{cases} \tag{3.20}$$

Equation (3.20) conforms to Bellman's principle of optimality, namely that regardless of the initial state s, the initial decision $\beta^N(s)$, and the state r resulting from the initial decision, the remaining decisions (beginning with $\beta^{N-1}(r)$) constitute an optimal policy with regard to state r. Therefore, the ADVISE decision rule (Equation (3.18)) implements an optimal policy when w_D is zero and the cost and payoff utility functions are linear. □

Theorem 3.2. *Let $w_C = w_P = 0$. Let the detection utility function be an exponential function of the form $U_D(x) = a \cdot e^{-x}$. Then, the ADVISE decision rule (Equation (3.18)) implements an optimal policy.*

Proof. For the case when w_C and w_P are zero, the attractiveness function is wholly exponential multiplicative:

$$attr^N(a_i, s) = w_D \cdot U_D(D_i^N(s)). \tag{3.21}$$

The equation for expected path detection (3.12) is substituted into Equation (3.21).

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i} (D_i(s,o) \cdot Pr_i(s,o))), & \\ \qquad\qquad\qquad \text{when } N = 1, & \\ w_D \cdot U_D(\sum_{o \in O_i} ((1 - (1 - D_i(s,o)) \cdot (1 - D_*^{N-1}(r))) \cdot Pr_i(s,o))), & \\ \qquad\qquad\qquad \text{when } N > 1. & \end{cases}$$

Next, the exponential detection utility function is represented as $U_D(x) = a \cdot e^{-x}$. For the remainder of the proof, we manipulate only the function valid for N greater than one.

$$attr^N(a_i,s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}((1-(1-D_i(s,o)) \cdot (1-D_*^{N-1}(r))) \cdot Pr_i(s,o))}, \\ \hfill \text{when } N > 1. \end{cases}$$

Then we separate the terms in the exponent.

$$attr^N(a_i,s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) - Pr_i(s,o) \cdot (1-D_i(s,o)) + Pr_i(s,o) \cdot (1-D_i(s,o)) \cdot D_*^{N-1}(r)]}, \\ \hfill \text{when } N > 1. \end{cases}$$

We split the exponent to isolate the $D_*^{N-1}(r)$ term.

$$attr^N(a_i,s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) - Pr_i(s,o) \cdot (1-D_i(s,o))]} \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot (1-D_i(s,o)) \cdot D_*^{N-1}(r)]}, \\ \hfill \text{when } N > 1. \end{cases}$$

Then we simplify the first exponent.

$$attr^N(a_i,s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot (1-D_i(s,o)) \cdot D_*^{N-1}(r)]}, \\ \hfill \text{when } N > 1. \end{cases}$$

We change the second exponential term from the exponential of a sum to the product of exponentials.

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i} (D_i(s,o) \cdot Pr_i(s,o))), & \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot \prod_{o \in O_i} e^{-[Pr_i(s,o) \cdot (1 - D_i(s,o)) \cdot D_*^{N-1}(r)]}, & \text{when } N > 1. \end{cases}$$

We rewrite the second exponent.

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i} (D_i(s,o) \cdot Pr_i(s,o))), & \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot \prod_{o \in O_i} (e^{-D_*^{N-1}(r)})^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, & \text{when } N > 1. \end{cases}$$

Recognizing the exponential detection utility function, we replace it with the function name U_D.

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i} (D_i(s,o) \cdot Pr_i(s,o))), & \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot \prod_{o \in O_i} \left(\frac{U_D(D_*^{N-1}(r))}{a} \right)^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, & \text{when } N > 1. \end{cases}$$

The attractiveness of the optimal attack step one step in the future is $attr^{(N-1)}(a_\beta, r) = w_D \cdot U_D(D_*^{(N-1)}(r))$, where $r = E_i(s, o)$ is the next-state given state s, attack step a_i, and outcome o; and a_β is the most attractive attack step in state r. We perform the following substitution and then simplify.

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \\ \quad \cdot \prod_{o \in O_i} \left(\frac{attr^{(N-1)}(a_\beta, r)}{(w_D \cdot a)} \right)^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, \\ \hfill \text{when } N > 1. \end{cases}$$

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_D \cdot a \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot (w_D \cdot a)^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot (1 - D_i(s,o))]} \\ \quad \cdot \prod_{o \in O_i} (attr^{(N-1)}(a_\beta, r))^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, \\ \hfill \text{when } N > 1. \end{cases}$$

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ (w_D \cdot a)^{\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot e^{-\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \\ \quad \cdot \prod_{o \in O_i} (attr^{(N-1)}(a_\beta, r))^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, \\ \hfill \text{when } N > 1. \end{cases}$$

$$attr^N(a_i, s) = \begin{cases} w_D \cdot U_D(\sum_{o \in O_i}(D_i(s,o) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ \left(\frac{w_D \cdot a}{e}\right)^{\sum_{o \in O_i}[Pr_i(s,o) \cdot D_i(s,o)]} \cdot \prod_{o \in O_i} (attr^{(N-1)}(a_\beta, r))^{[Pr_i(s,o) \cdot (1 - D_i(s,o))]}, \\ \hfill \text{when } N > 1. \end{cases} \tag{3.22}$$

Equation (3.22) conforms to Bellman's principle of optimality, namely that regardless of the initial state s, the initial decision $\beta^N(s)$, and the state r resulting from the initial decision, the remaining decisions constitute an optimal policy with regard to state r. Therefore, the ADVISE decision rule (Equation (3.18)) implements an optimal policy when w_C and w_P are zero and the detection utility function is exponential. $\square$

We now show one instance in which the ADVISE decision rule is nonoptimal when considered from decision epoch $t-1$. In particular, we examine how a combination of linear additive and exponential multiplicative rewards within the attractiveness function can result in nonoptimal decisions.

Theorem 3.3. *Let w_D and w_C both be nonzero. Let the cost utility function be linear of the form $U_C(x) = m_C x + b_C$. Let the detection utility functions be exponential of the form $U_D(x) = a \cdot e^{-x}$. Then, there exist at least one attack execution graph and adversary for which the ADVISE decision rule (Equation (3.18)) does not implement an optimal policy.*

Proof. When linear additive and exponential multiplicative rewards are combined in one decision rule, the attractiveness function does not necessarily produce an optimal decision. We provide a counterexample showing the existence of an attack execution graph and adversary for which the ADVISE decision rule applied at decision epoch t produces a nonoptimal choice when considered from decision epoch $t-1$.

For simplicity, let $w_P = 0$, so that

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_D \cdot U_D(D_i^N(s)).$$

Consider the adversary's attack step decision in state s with two available attack steps a_1 and a_2 and a planning horizon $N = 1$.

Let the attractiveness of the two attack steps be

$$attr^1(a_1, s) = w_C \cdot U_C(C_1(s)) + w_D \cdot U_D(D_1(s)) \tag{3.23}$$

and

$$attr^1(a_2, s) = w_C \cdot U_C(C_2(s)) + w_D \cdot U_D(D_2(s)). \tag{3.24}$$

Let $C_1(s) = c_1$, $C_2(s) = c_1 + c_0$, $D_1(s) = d_1$, and $D_2(s) = d_1 + d_0$, so that Equations (3.23) and (3.24) become Equations (3.25) and (3.26), respectively.

$$attr^1(a_1, s) = w_C \cdot U_C(c_1) + w_D \cdot U_D(d_1). \tag{3.25}$$

$$attr^1(a_2, s) = w_C \cdot U_C(c_1 + c_0) + w_D \cdot U_D(d_1 + d_0). \tag{3.26}$$

Let attack step a_1 be the most attractive attack step choice in state s with planning horizon $N = 1$. Therefore,

$$\begin{aligned}
attr^1(a_1, s) &> attr^1(a_2, s) \\
w_C \cdot U_C(c_1) + w_D \cdot U_D(d_1) &> w_C \cdot U_C(c_1 + c_0) + w_D \cdot U_D(d_1 + d_0) \\
w_C \cdot m_C \cdot c_1 + w_C \cdot b_C + w_D \cdot a \cdot e^{-d_1} &> w_C \cdot m_C \cdot (c_1 + c_0) + w_C \cdot b_C + w_D \cdot a \cdot e^{-(d_1+d_0)} \\
0 &> w_C \cdot m_C \cdot c_0 + w_D \cdot a \cdot e^{-(d_1+d_0)} - w_D \cdot a \cdot e^{-d_1} \\
0 &> w_C \cdot m_C \cdot c_0 + w_D \cdot a \cdot (e^{-(d_1+d_0)} - e^{-d_1}). \quad (3.27)
\end{aligned}$$

Now consider the attack step decision at decision epoch $t - 1$ in state q with available attack step a_3 leading to next-state s. We compute the attractiveness of attack step a_3 with a planning horizon of $N = 2$. Let c_3 and d_3 be the cost and detection, respectively, of attack step a_3.

The ADVISE decision rule dictates that the most attractive attack step in state s, namely a_1, be used in computing the attractiveness of a_3, but we show that there exist instances in which choosing the other attack step a_2 in state s would yield a higher attractiveness value for a_3 in state q. In those instances, the ADVISE decision rule does not implement an optimal policy.

The attractiveness value of a_3 using the ADVISE decision rule (i.e., computed using the most attractive attack step in state s, a_1) is

$$attr^2(a_3, q) = w_C \cdot U_C(c_1 + c_3) + w_D \cdot U_D(1 - (1 - d_1)(1 - d_3)).$$

The attractiveness value of a_3 using the less attractive attack step in state s (a_2) is

$$attr^{2,alt}(a_3, q) = w_C \cdot U_C((c_1 + c_0) + c_3) + w_D \cdot U_D(1 - (1 - (d_1 + d_0))(1 - d_3)).$$

If $attr^1(a_1, s) > attr^1(a_2, s)$ and $attr^{2,alt}(a_3, q) > attr^2(a_3, q)$ both hold true, then the ADVISE decision rule does not implement an optimal policy.

$$\begin{aligned}
attr^{2,alt}(a_3, q) &> attr^2(a_3, q) \\
w_C \cdot U_C((c_1 + c_0) + c_3) + w_D \cdot U_D(1 - (1 - (d_1 + d_0))(1 - d_3)) &> \\
w_C \cdot U_C(c_1 + c_3) + w_D \cdot U_D(1 - (1 - d_1)(1 - d_3)) & \\
w_C \cdot m_C(c_1 + c_0 + c_3) + w_C \cdot b_C + w_D \cdot a \cdot e^{-(1-(1-(d_1+d_0))(1-d_3))} &> \\
w_C \cdot m_C(c_1 + c_3) + w_C \cdot b_C + w_D \cdot a \cdot e^{-(1-(1-d_1)(1-d_3))}. & \qquad (3.28)
\end{aligned}$$

We provide a scenario under which Inequalities (3.27) and (3.28) can both hold true:

Let $w_C = w_D = 0.5$, $m_C = -1$, $b_C = 1$, $a = 1$, $c_0 = 0.5$, $c_1 = c_3 = 0.2$, $d_0 = -0.98$, $d_1 = 0.99$, and $d_3 = 0.1$. Then Inequality (3.28) becomes $0.013 > 0 > -0.366$. This result means that the ADVISE decision rule did not implement an optimal policy in this instance. Thus, we have proven that the ADVISE decision rule does not implement an optimal policy for all attack execution graphs and adversaries when w_D and w_C are both nonzero.

$\square$

The implication of this result is that, for some attack decisions, the ADVISE decision rule does not select the most attractive next attack step. There exists another policy that selects a more attractive next attack step than the ADVISE decision rule. We use the result of Theorem 3.3 as motivation for exploring an alternative always-optimal attractiveness function in the next section.

3.4 An Alternative Attractiveness Function

In the original attractiveness function, the three decision criteria are attack step cost, attack goal payoff, and probability of detection. When the adversary makes decisions using a planning horizon greater than one, the cost and payoff are computed as linear additive rewards. However, the detection probability is computed as a multiplicative reward. When linear additive and exponential multiplicative rewards are combined in one decision rule, the attractiveness function does not necessarily produce an optimal decision.

To obtain an alternative attractiveness function that always produces mathematically optimal decisions, we replace the detection probability criterion with a "log nondetection" criterion. Nondetection, $F_i(s, o)$, refers to the probability that the adversary is not detected after being in state s, attempting attack step a_i, and obtaining outcome o. The probability of nondetection is one minus the probability of detection. The expected log nondetection of attack step a_i when the adversary is in state s, written $\log(F_i(s))$, is computed as the logarithm of the geometric mean of the nondetection probabilities in each possible attack step outcome weighted by the probability of that attack step outcome:

$$\log(F_i(s)) = \sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)). \tag{3.29}$$

Note that the logarithm of the weighted geometric mean is equivalent to the weighted arithmetic mean of the logarithms of the individual values.

3.4.1 Alternative Attractiveness Function Definition

When the detection decision criterion is replaced by the "log nondetection" decision criterion, the attractiveness of available attack step a_i in state s is computed using a modified equation:

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)) + w_F \cdot U_F(\log(F_i^N(s))), \tag{3.30}$$

where $C_i^N(s)$ is the recursively computed *expected path cost*, w_C is the adversary preference weight for cost, $P_i^N(s)$ is the recursively computed *expected horizon payoff*, w_P is the adversary preference weight for payoff, $log(F_i^N(s))$ is the recursively computed *expected path log nondetection*, and w_F is the adversary preference weight for log nondetection. The expected path cost, $C_i^N(s)$, was previously defined in Equation (3.8), and the expected horizon payoff, $P_i^N(s)$, was previously defined in Equation (3.10). $U_F(\cdot)$ is the adversary's utility function for log nondetection.

The expected path log nondetection, $log(F_i^N(s))$, is the expected log nondetection during attack step a_i and the other attack steps continuing a total of N steps into the future.

$$\log(F_i^N(s)) = \begin{cases} \sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)), & \\ \qquad\qquad\qquad\qquad \text{when } N = 1, \\ \sum_{o \in O_i} ((\log(F_i(s,o)) + \log(F_*^{N-1}(r))) \cdot Pr_i(s,o)), & \\ \qquad\qquad\qquad\qquad \text{when } N > 1, \end{cases} \tag{3.31}$$

where state $r = E_i(s,o)$ is the next state after the adversary starts in state s, attempts attack step a_i, and obtains outcome o.

The computation of the expected path log nondetection requires determining the best next attack step from state r with planning horizon $(N-1)$:

$$\log(F_*^{N-1}(r)) \equiv \log(F_k^{N-1}(r)) \text{ when } \beta^{N-1}(r) = a_k. \tag{3.32}$$

3.4.2 Utility Functions for an Optimal Attractiveness Function

The adversary's utility functions ($U_C(\cdot), U_P(\cdot)$, and $U_F(\cdot)$) map the native values of each attractiveness criterion to a $[0,1]$ utility scale, where higher utility values represent more desirable values. The utility functions convert cost, payoff, and log nondetection into a common unit of "utility" so that the weighted average computation (in the attractiveness function) is mathematically valid.

We must construct appropriate utility functions to ensure the optimality of the attractiveness computation. It is sufficient for the utility functions to be linear. The linear cost utility function has a negative slope, and the linear utility functions for payoff and nondetection have a positive slope.

The following utility functions are sufficient (but not necessary) for constructing an optimal attractiveness computation. The domain of the cost utility function is $[0, C_{max} \cdot N]$, where C_{max} is the maximum single-attack-step cost and N is the planning horizon. Equation (3.33) gives a mathematical formula for the cost utility function.

$$U_C(c) = \frac{-c}{C_{max} \cdot N} + 1. \tag{3.33}$$

The domain of the payoff utility function is $[0, P_{max}]$, where P_{max} is the maximum payoff value of a single model state. Equation (3.34) gives a mathematical formula for the payoff utility function.

$$U_P(p) = \frac{p}{P_{max}}. \tag{3.34}$$

The domain of the log nondetection utility function is $[N \cdot \log(F_{min}), 0]$, where F_{min} is the minimum single-attack-step nondetection probability. Note that F_{min} must be nonzero. Equation (3.35) gives a mathematical formula for the log nondetection utility function.

$$U_F(f) = \frac{-f}{N \cdot \log(F_{min})} + 1. \tag{3.35}$$

The range for all utility functions is $[0,1]$.

3.4.3 Optimality of the Alternative Attractiveness Function

We now examine the optimality of the ADVISE policy when it incorporates the alternative attractiveness function as described in Section 3.4.1 and linear utility functions as described in Section 3.4.2.

We use the same MDP definition given in Section 3.3, except we use a different reward function. The stationary reward function is now defined as $attr(a_i, s) = w_C \cdot U_C(C_i(s)) + w_P \cdot U_P(P_i(s)) + w_F \cdot U_F(\log(F_i(s)))$.

The revised ADVISE attack step selection process uses a stationary deterministic Markovian decision rule,

$$\beta^N(s) = \underset{a_i \in A_s}{\arg\max}\{attr^N(a_i, s)\}, \tag{3.36}$$

where

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)) + w_F \cdot U_F(\log(F_i^N(s))),$$

where w_C is the adversary preference weight for cost; $C_i^N(s)$ is the recursively computed *expected path cost* (that is, the expected sum of attack step costs incurred from decision epoch t to decision epoch $t+N-1$, given state s and action a_i in decision epoch t); $U_C(\cdot)$ is the cost utility function, which converts cost to units of utility; w_P is the adversary preference weight for payoff; $P_i^N(s)$ is the recursively computed *expected horizon payoff* (that is, the expected payoff in decision epoch $t+N-1$, given state s and action a_i in decision epoch t); $U_P(\cdot)$ is the payoff utility function, which converts payoff to units of utility; w_F is the adversary preference weight for log nondetection; $log(F_i^N(s))$ is the recursively computed *expected path log nondetection* (that is, the expected log nondetection from decision epoch t to decision epoch $t+N-1$, given state s and action a_i in decision epoch t); and $U_F(\cdot)$ is the log nondetection utility function, which converts log nondetection values into units of utility.

We examine the conditions under which application of this revised decision rule constitutes an optimal policy.

Taken individually, cost, payoff, and log nondetection are linear additive rewards. For payoff, the additive payoff is nonzero only in the Nth decision epoch. The ADVISE decision rule combines these three components (cost, payoff, and log nondetection) into one function that contains all three types of linear additive rewards. The ADVISE decision rule is provably optimal when the utility functions meet the sufficient conditions.

Theorem 3.4. *Let the cost, payoff, and log nondetection utility functions be linear functions of the forms $U_C(x) = m_C x + b_C$, $U_P(x) = m_P x + b_P$, and $U_F(x) = m_F x + b_F$, respectively. Then, the alternative ADVISE decision rule (Equation (3.36)) implements an optimal policy.*

Proof. The alternative attractiveness function is defined as

$$attr^N(a_i, s) = w_C \cdot U_C(C_i^N(s)) + w_P \cdot U_P(P_i^N(s)) + w_F \cdot U_F(\log(F_i^N(s))). \tag{3.37}$$

The equations for expected path cost (3.8), expected horizon payoff (3.10), and expected path log nondetection (3.31) are substituted into Equation (3.37).

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s, o) \cdot (P(E_i(s, o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s, o)) \cdot Pr_i(s, o))), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } N = 1, \\ w_C \cdot U_C(C_i(s) + \sum_{o \in O_i} (C_*^{N-1}(r) \cdot Pr_i(s, o))) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} (P_*^{N-1}(r) \cdot Pr_i(s, o))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} ((\log(F_i(s, o)) + \log(F_*^{N-1}(r))) \cdot Pr_i(s, o))), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } N > 1. \end{cases}$$

Next, we replace the utility function names with their slope-intercept representations: $U_C(x) = m_C x + b_C$, $U_P(x) = m_P x + b_P$, and $U_F(x) = m_F x + b_F$. For the remainder of the proof, we manipulate only the function valid for N greater than one.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o\in O_i} Pr_i(s,o)\cdot(P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o\in O_i}(\log(F_i(s,o))\cdot Pr_i(s,o))), \\ \hfill \text{when } N=1, \\ w_C \cdot (m_C(C_i(s) + \sum_{o\in O_i}(C_*^{N-1}(r)\cdot Pr_i(s,o))) + b_C) \\ \quad +w_P \cdot (m_P(\sum_{o\in O_i}(P_*^{N-1}(r)\cdot Pr_i(s,o))) + b_P) \\ \quad +w_F \cdot (m_F(\sum_{o\in O_i}((\log(F_i(s,o)) + \log(F_*^{N-1}(r)))\cdot Pr_i(s,o))) + b_F), \\ \hfill \text{when } N>1. \end{cases}$$

Then we separate the terms.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o\in O_i} Pr_i(s,o)\cdot(P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o\in O_i}(\log(F_i(s,o))\cdot Pr_i(s,o))), \\ \hfill \text{when } N=1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_C \cdot m_C \cdot (\sum_{o\in O_i}(C_*^{N-1}(r)\cdot Pr_i(s,o))) \\ \quad +w_C \cdot b_C \\ \quad +w_P \cdot m_P \cdot (\sum_{o\in O_i}(P_*^{N-1}(r)\cdot Pr_i(s,o))) \\ \quad +w_P \cdot b_P \\ \quad +w_F \cdot m_F \cdot (\sum_{o\in O_i}(\log(F_i(s,o))\cdot Pr_i(s,o)))) \\ \quad +w_F \cdot m_F \cdot (\sum_{o\in O_i}(\log(F_*^{N-1}(r))\cdot Pr_i(s,o))) \\ \quad +w_F \cdot b_F, \\ \hfill \text{when } N>1. \end{cases}$$

Because $\sum_{o\in O_i} Pr_i(s,o) = 1$, we can introduce this quantity in the b_C, b_P, and b_F terms so that we can combine terms more easily later.

$$attr^N(a_i,s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o))), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_F \cdot m_F \cdot (\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)))) \\ \quad +w_C \cdot m_C \cdot \sum_{o \in O_i} (C_*^{N-1}(r) \cdot Pr_i(s,o)) \\ \quad +w_C \cdot b_C \cdot \sum_{o \in O_i} Pr_i(s,o) \\ \quad +w_P \cdot m_P \cdot \sum_{o \in O_i} (P_*^{N-1}(r) \cdot Pr_i(s,o)) \\ \quad +w_P \cdot b_P \cdot \sum_{o \in O_i} Pr_i(s,o) \\ \quad +w_F \cdot m_F \cdot \sum_{o \in O_i} (\log(F_*^{N-1}(r)) \cdot Pr_i(s,o)) \\ \quad +w_F \cdot b_F \cdot \sum_{o \in O_i} Pr_i(s,o), \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{when } N > 1. \end{cases}$$

Next, we move the preference weights and the slope and intercept terms inside the summations.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_F \cdot m_F \cdot (\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)))) \\ \quad + \sum_{o \in O_i} (w_C \cdot m_C \cdot C_*^{N-1}(r) \cdot Pr_i(s,o)) \\ \quad + \sum_{o \in O_i} (w_C \cdot b_C \cdot Pr_i(s,o)) \\ \quad + \sum_{o \in O_i} (w_P \cdot m_P \cdot P_*^{N-1}(r) \cdot Pr_i(s,o)) \\ \quad + \sum_{o \in O_i} (w_P \cdot b_P \cdot Pr_i(s,o)) \\ \quad + \sum_{o \in O_i} (w_F \cdot m_F \cdot \log(F_*^{N-1}(r)) \cdot Pr_i(s,o)) \\ \quad + \sum_{o \in O_i} (w_F \cdot b_F \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

We combine the summation terms into one summation.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_F \cdot m_F \cdot (\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)))) \\ \quad + \sum_{o \in O_i} (((w_C \cdot m_C \cdot C_*^{N-1}(r)) + (w_C \cdot b_C) \\ \qquad\qquad +(w_P \cdot m_P \cdot P_*^{N-1}(r)) + (w_P \cdot b_P) \\ \qquad\qquad +(w_F \cdot m_F \cdot \log(F_*^{N-1}(r))) + (w_F \cdot b_F)) \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

Recognizing the linear utility functions, we replace them with the function names U_C, U_P, and U_F.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_F \cdot m_F \cdot (\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)))) \\ \quad + \sum_{o \in O_i} (((w_C \cdot U_C(C_*^{N-1}(r))) \\ \qquad\quad +(w_P \cdot U_P(P_*^{N-1}(r))) \\ \qquad\quad +(w_F \cdot U_F(\log(F_*^{N-1}(r))))) \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases}$$

The attractiveness of the optimal attack step one step in the future is $attr^{(N-1)}(a_\beta, r) = w_C \cdot U_C(C_*^{(N-1)}(r)) + w_P \cdot U_P(P_*^{(N-1)}(r)) + w_F \cdot U_F(\log(F_*^{N-1}(r))$, where $r = E_i(s,o)$ is the next-state given state s, attack step a_i, and outcome o; and a_β is the most attractive attack step in state r.

We perform the following substitution.

$$attr^N(a_i, s) = \begin{cases} w_C \cdot U_C(C_i(s)) \\ \quad +w_P \cdot U_P(\sum_{o \in O_i} Pr_i(s,o) \cdot (P(E_i(s,o)))) \\ \quad +w_F \cdot U_F(\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o))), \\ \hfill \text{when } N = 1, \\ w_C \cdot m_C \cdot C_i(s) \\ \quad +w_F \cdot m_F \cdot (\sum_{o \in O_i} (\log(F_i(s,o)) \cdot Pr_i(s,o)))) \\ \quad + \sum_{o \in O_i} (attr^{(N-1)}(a_\beta, r)) \cdot Pr_i(s,o)), \\ \hfill \text{when } N > 1. \end{cases} \tag{3.38}$$

Equation (3.38) conforms to Bellman's principle of optimality, namely that regardless of the initial state s, the initial decision $\beta^N(s)$, and the state r resulting from the initial decision, the remaining decisions (beginning with $\beta^{N-1}(r)$) constitute an optimal policy with regard to state r. Therefore, the alternative ADVISE decision rule (Equation (3.36)) implements an optimal policy. $\square$

3.5 Performance Analysis

We conduct a performance analysis to explore the scalability of the ADVISE analysis method. As the size of an ADVISE model increases, the execution runtime for each decision computation increases. This runtime increase is due to the increase in the size of the SLAT that is traversed during the recursive attractiveness computation.

The factors that can affect the execution runtime of the decision computation are the planning horizon N, the total number of attack steps in the attack execution graph, the number of available attack steps in each state, and the number of outcomes per attack step. We examine each of these factors individually, holding all other factors constant.

This analysis is performed using the implementation of ADVISE as a new atomic model formalism within the Möbius modeling and simulation tool. The Möbius framework supports multiple input formalisms and multiple solution techniques [18]. Appendix A describes the tool implementation of ADVISE within Möbius. Each data point reported in this performance analysis is the Möbius-reported CPU runtime for 100,000 simulation runs. The simulation is set to terminate such that the simulation performs one attack decision computation per run.

For an implementation of the ADVISE decision computation based on an in-order traversal of the SLAT, the memory space requirement is linear with the planning horizon, so space is not a practical constraint.

3.5.1 Basic Model

The execution runtime is highly dependent on the model parameters, so we use an ADVISE model with a regular structure for the performance analysis. The basic model is shown in Figure 3.4. The basic model contains eight attack steps, including seven standard attack steps plus the do-nothing attack step (not shown). The attack steps in Figure 3.4 are labeled from top to bottom: Step_1, . . . , Step_7. Step_8 is the do-nothing attack step that is not visible. The attack execution graph for the basic model also contains eight access domains and seven attack goals. The access domains in Figure 3.4 are named from top to bottom:

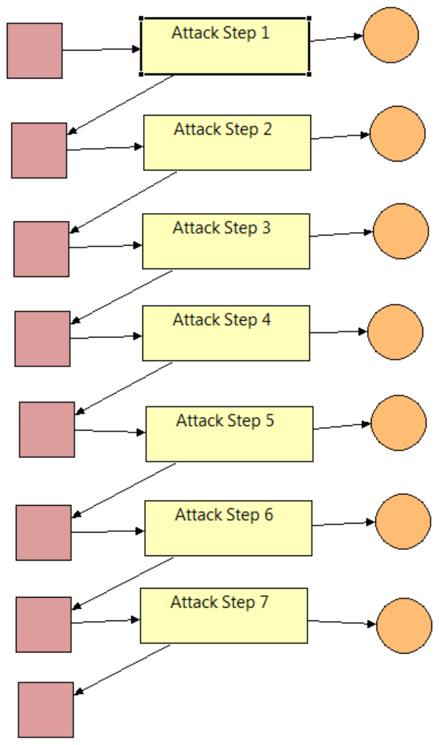


Figure 3.4: Attack execution graph for the basic model.

Access_1, . . . , Access_8. The attack goals in Figure 3.4 are named from top to bottom: Goal_1, . . . , Goal_7. To simplify the performance analysis, attack skills and knowledge items are not included in any of the performance analysis models because their presence does not impact the size of the SLAT.

All the attack steps in the attack execution graph possess a similar structure. For each attack step Step_M ∈ {Step_1, . . . , Step_8}, the attack cost is 10, and the attack execution time is 49. Note that cost and time are put into the model as unitless quantities; the person who constructs the model must take care that all input values are in the same cost unit (dollars, millions of dollars, etc.) and the same time unit (minutes, hours, days, etc.).

The precondition for the do-nothing attack step is always true. The precondition for all other attack steps is of the following form: (Access_M == 1) and (Access_(M+1) == 0). I.e., the adversary must possess Access_M and not possess Access_(M+1).

Each attack step has two outcomes. Outcome 1 has an outcome probability of 0.5 and a detection probability of 0.5. The effects of Outcome 1 are of the following form: Access_(M+1) = 1; Goal_M = 1. I.e., the adversary gains Access_(M+1) and achieves Goal_M.

Outcome 2 also has an outcome probability of 0.5 and a detection probability of 0.5. There are no effects for Outcome 2.

The same adversary profile is used for all performance analysis models. The adversary preference weights are as follows: the cost preference weight is 0, the detection preference weight is 0, and the payoff preference weight is 1. The adversary possesses no skills and no initial knowledge. The adversary's initial access consists of Access_1 only. The attack goals each have a payoff value of 100.

The regular structure of this attack execution graph means that the number of available attack steps in each state in the SLAT is two (the do-nothing attack step and one other). The one exception is the model state in which there is only one available attack step. The adversary possesses all the access domains and has achieved all the attack goals, and the only attack step available is the do-nothing attack step. However, this state does not appear in the SLAT we analyze because that state cannot be reached in the first attack decision with the planning horizon depths that we use in the analysis. The regular structure of this attack execution graph also means that the number of outcomes per attack step is always two.

When we modify the attack execution graph to increase the number of available attack steps in each state or the number of outcomes per attack step, we do so in a way that maintains the regular structure of the attack execution graph.

3.5.2 Planning Horizon

The adversary's planning horizon N affects model execution time because the planning horizon determines the depth of the SLAT.

Figure 3.5 shows the structure of the full (unpruned) SLAT for the basic model with a planning horizon of two. Each state has two available attack steps, and each attack step has two outcomes. The ADVISE attack decision function traverses the SLAT to select the most attractive next attack step from a given state.

Increasing the adversary's planning horizon increases the execution time exponentially, as shown in Figure 3.6. The curve is relatively flat for small N because simulation overhead dominates the execution time. For large N, the planning horizon significantly impacts the execution time.

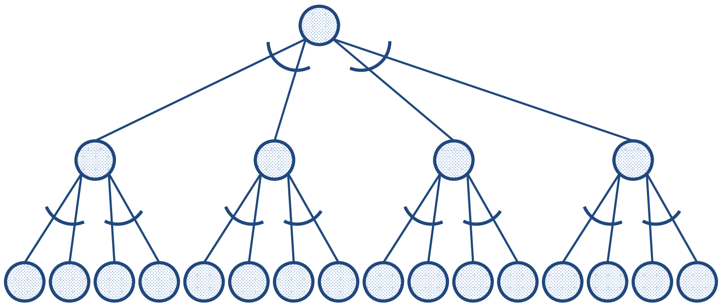


Figure 3.5: Structure of a state look-ahead tree for the basic model (when $N = 2$).

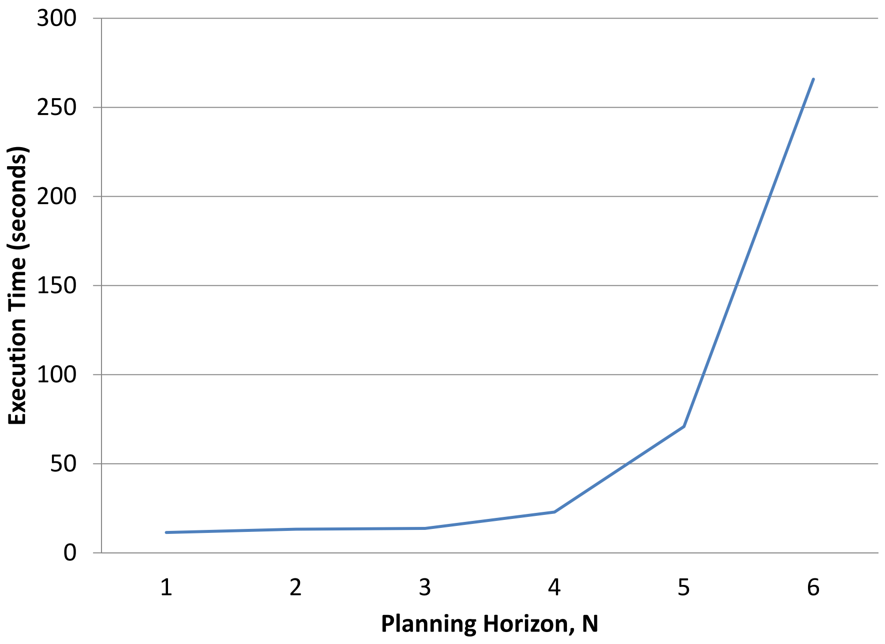


Figure 3.6: Execution time for the basic model (for 100,000 runs).

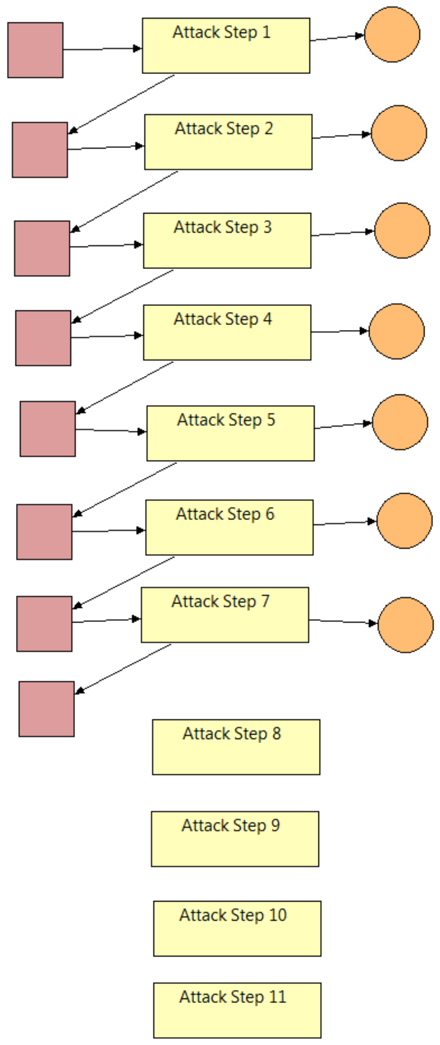


Figure 3.7: Attack execution graph for the model with 12 attack steps.

3.5.3 Total Number of Attack Steps

To study the effect of the number of attack steps in the attack execution graph on the execution time, we add attack steps to the basic model. To isolate the effect of the total number of attack steps and the effect of the number of attack steps available to the adversary at each state, the additional attack steps are assigned a precondition that is always false. The attack execution graph for the basic model contains eight attack steps. We also measure the execution time for attack execution graphs with 12 and 16 attack steps, shown in Figures 3.7 and 3.8, respectively.

The total number of attack steps does not impact the size of the SLAT. However, the

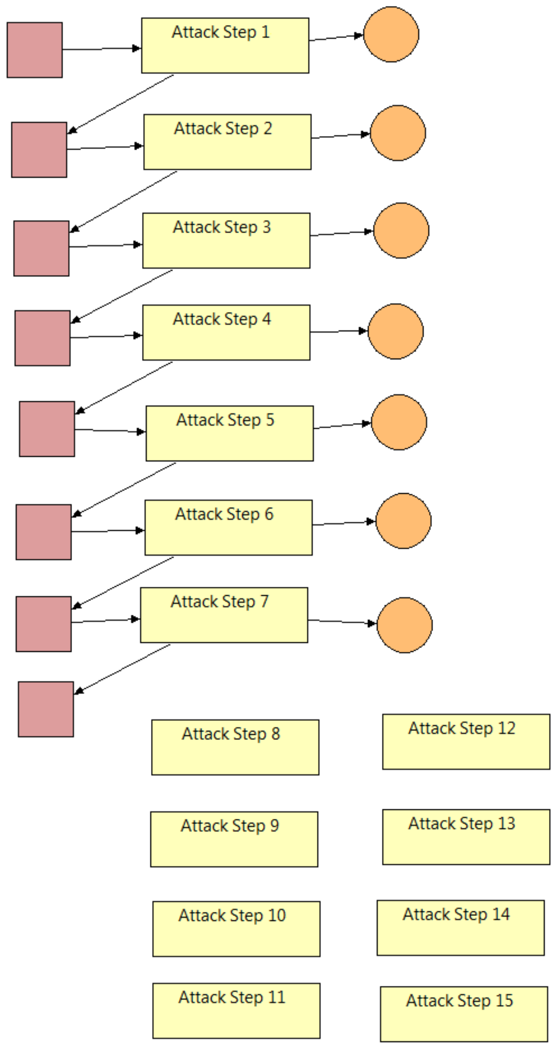


Figure 3.8: Attack execution graph for the model with 16 attack steps.

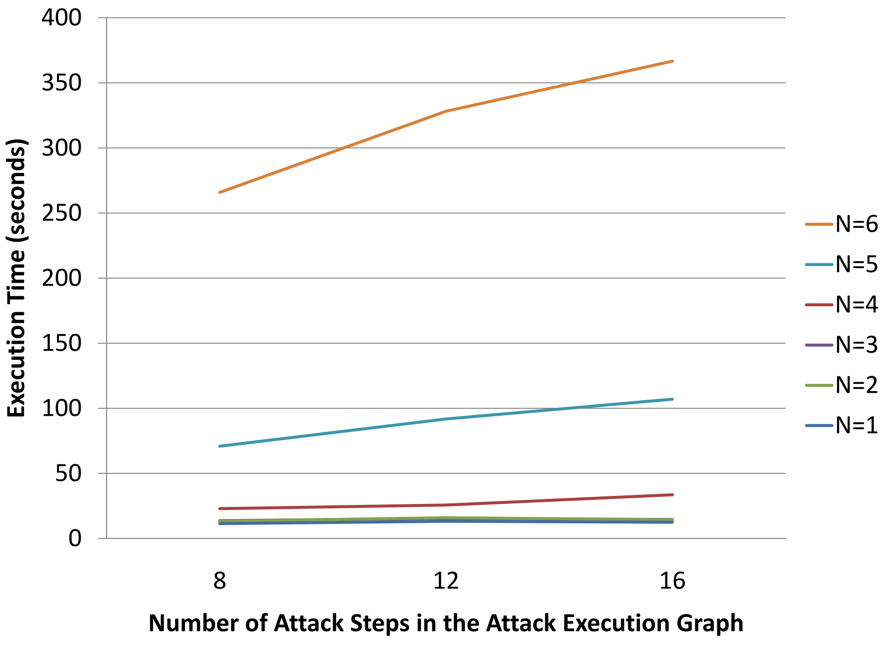


Figure 3.9: Execution time for models with 8, 12, or 16 attack steps (for 100,000 runs) (N is the planning horizon).

construction of the SLAT involves checking the precondition of each attack step in the attack execution graph to determine which attack steps are available (i.e., which attack steps have a true precondition). Therefore, an increase in the total number of attack steps increases the execution time linearly with respect to the number of states traversed during the SLAT construction. Thus, the total number of attack steps in the attack execution graph contributes to the execution time in a relatively minor way compared to the other factors we study. The execution times for attack execution graphs with 8, 12, and 16 attack steps are shown in Figure 3.9.

3.5.4 Number of Available Attack Steps in Each State

Next, we study the effect on execution time of the number of available attack steps in each state in the SLAT. For this analysis, we use the model with 16 attack steps (shown in Figure 3.8) with some changes. In the previous analysis, all eight additional attack steps had a false precondition so that the number of available attack steps in each state remained at two. For this analysis, we first change the precondition of one of the additional attack

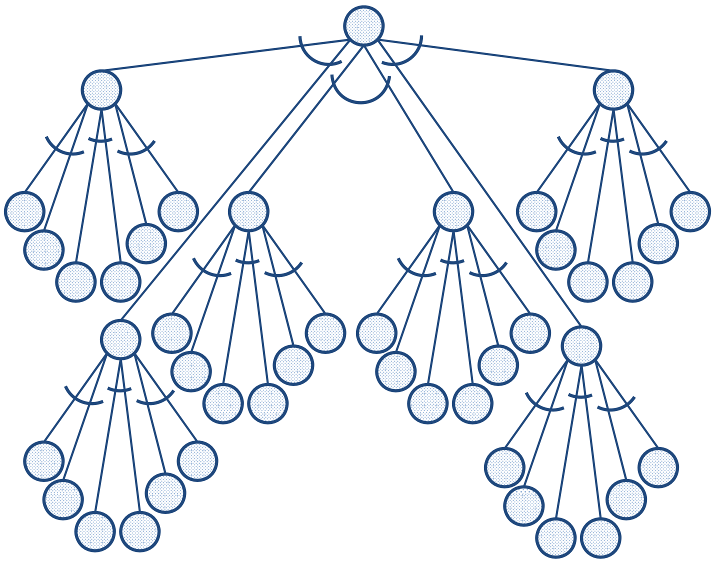


Figure 3.10: Structure of a state look-ahead tree for the model with three available attack steps in each state (when $N = 2$).

steps to always be true. As a result, there are three available steps at each state in the SLAT, as shown in Figure 3.10. Then we change the precondition of one more of the additional attack steps to always be true. Then, there are four available steps at each state in the SLAT. We note that none of the additional attack steps have effects on the model state.

Increasing the number of available attack steps in each state in the SLAT exponentially increases the size of the SLAT. Thus, the time required to traverse all the states in the SLAT increases exponentially, causing the execution time to increase exponentially, as shown in Figure 3.11.

3.5.5 Number of Outcomes Per Attack Step

Finally, we study the effect on execution time of the number of outcomes per attack step. In the basic model, each attack step has two outcomes. For this analysis, we use the model with eight attack steps (shown in Figure 3.4) with some changes. We increase the number of outcomes per attack step to three and to four. The SLAT for the case of three outcomes per attack step is shown in Figure 3.12.

As the number of outcomes per attack step increases, the size of the SLAT increases

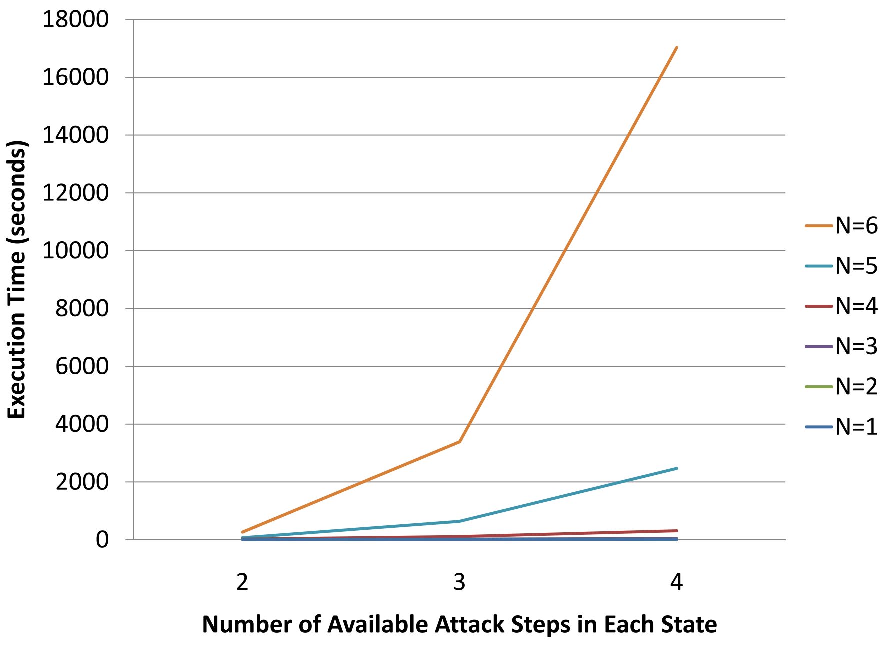


Figure 3.11: Execution time for the models with two, three, or four available attack steps in each state (for 100,000 runs) (N is the planning horizon).

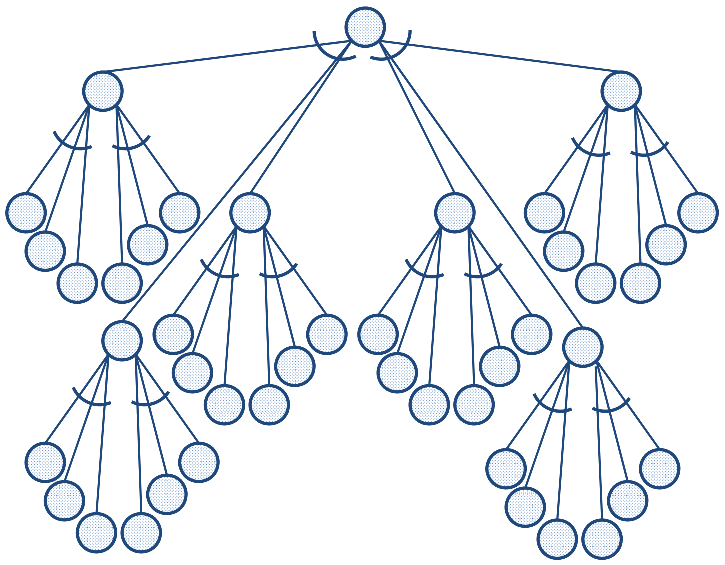


Figure 3.12: Structure of a state look-ahead tree for the model with three outcomes per attack step (when $N = 2$).

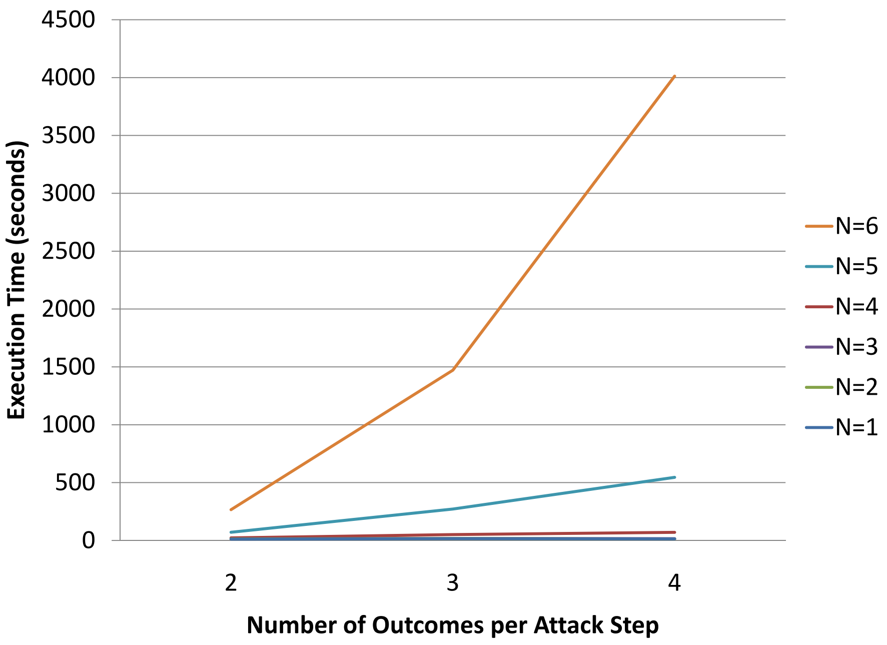


Figure 3.13: Execution time for the models with two, three, or four outcomes per attack step (for 100,000 runs) (N is the planning horizon).

exponentially. Thus, the time required to traverse all the states in the SLAT increases exponentially, causing the execution time to increase exponentially, as shown in Figure 3.13.

In summary, the execution time of the ADVISE adversary attack decision grows exponentially with respect to increases in the planning horizon, the number of available attack steps in a state, and the number of outcomes per attack step. For increases in the total number of attack steps in an attack execution graph, the execution time increases linearly with respect to the number of states traversed during the SLAT construction.

CHAPTER 4

ADVISE METRICS

The purpose of building ADVISE models is to generate quantitative metrics that provide insight on the security strength of a particular system against a particular adversary. Metrics can be generated by running discrete-event simulations of the adversary attacking the system.

In general, metrics generated by discrete-event simulation can take many forms. The Möbius modeling and simulation tool provides a general metrics capability called reward variables [19]. To introduce analysts to the type of analysis possible with ADVISE models and to facilitate rapid ADVISE model analysis, we specify a particular format for ADVISE metrics that was inspired by the reward variables implemented in Möbius.

There are two types of ADVISE metrics: state metrics and event metrics. State metrics analyze the model state. Event metrics analyze events, namely state changes, attack step attempts, and attack step outcomes. For each metric, there are three required parts: time, metric type, and the state or event indicator function. Conditional reporting, which is optional, requires a condition expression.

4.1 State Metrics

State metrics take the form

$$\langle \tau, \lambda, \sigma \rangle, \tag{4.1}$$

where τ is the end time such that the metric reports on the time period $[0, \tau]$, $\lambda \in \{\mathit{EndProb}, \mathit{AvgTime}\}$ specifies the type of state metric, and σ is the state indicator function. *EndProb* is the ending state occupancy probability metric, i.e., the probability that the model state s at time τ is a state of interest ($\sigma(s) =$ True). *AvgTime* is the average time metric, i.e., the average amount of time during the time interval $[0, \tau]$ spent in a model state s such that

This chapter contains previously published material by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [2] is reused by permission of IEEE and was published in Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).

$\sigma(s) = \text{True}$.

The state indicator function, σ, returns a True value for model states of interest:

$$\sigma : X \rightarrow \{True, False\}, \tag{4.2}$$

where X is the set of all model states. Recall that the model state $s \in X$ is defined by the tuple $\langle R_s, K_s, G_s \rangle$. The state indicator function separates states of interest from those not of interest. For example, a state metric could measure the probability that the model is in a state s at time $\tau = 200$ in which goal $g_1 \in G$ has been achieved (i.e., $g_1 \in G_s$).

4.2 Event Metrics

Event metrics take the form

$$\langle \tau, \delta, \epsilon \rangle, \tag{4.3}$$

where τ is the end time such that the metric reports on the time period $[0, \tau]$, $\delta \in \{$*Freq, ProbOcc*$\}$ specifies the type of event metric, and ϵ is the event set. *Freq* is the frequency metric, i.e., the number of occurrences of events in ϵ during the time interval $[0, \tau]$. *ProbOcc* is the probability of occurrence metric, i.e., the probability that one or more of the events in ϵ occur at least once during the time interval $[0, \tau]$.

The event set ϵ is specified as the union of eight sets:

$$\epsilon = A_\epsilon \cup O_\epsilon \cup R_+ \cup R_- \cup K_+ \cup K_- \cup G_+ \cup G_-, \tag{4.4}$$

where $A_\epsilon \subseteq A$ is the set of attack steps whose execution constitutes an event; $O_\epsilon \subseteq \{\bigcup_{a_i \in A} O_i\}$ is the set of attack step outcomes whose execution constitutes an event; R_+, K_+, and G_+ are the sets of access domains, knowledge items, and attack goals, respectively, whose addition to the model state $\langle R_s, K_s, G_s \rangle$ constitutes an event; and R_-, K_-, and G_- are the sets of access domains, knowledge items, and attack goals, respectively, whose removal from the model state $\langle R_s, K_s, G_s \rangle$ constitutes an event. For simple metrics, all but one of these sets may be empty. For example, an event metric could measure the frequency with which the adversary attempts attack step a_i in the interval $[0, \tau]$. In this example, $\epsilon = \{a_i\}$ because a_i is the only event of interest.

4.3 Conditional Reporting

Conditional reporting enables analysts to study correlations between different metrics. Typically, metrics data are generated from many simulation runs and averaged across all runs. With conditional reporting, only the simulation runs that meet a specified condition are included in the average. The conditional expression includes a state or event metric, an equality or inequality relation, and a value. For example, one conditional expression could be that the frequency of attempts of attack step a_i is greater than four during time interval $[0, \tau]$. When a conditional expression is linked with a second state or event metric, correlations can be measured directly. One example of a metric with conditional reporting is the following: if, for this simulation run, the frequency of attack attempts for attack step a_i is greater than four, then use the data from this run to compute the ending state occupancy probability of a model state in which attack goal g_j is achieved. This metric enables analysts to examine the correlation between frequent attempts of attack step a_i and the achievement of attack goal g_j. Algorithm 4.1 provides one method for producing conditional reporting of simulation results.

Algorithm 4.1 Conditional Reporting of Simulation Results

```
Input: NumRuns
1: X ← 0
2: while X < NumRuns do
3:     ⟨ConditionalExpression, Metric⟩ ← Output from a new simulation run
4:     if ConditionalExpression = TRUE then
5:         Report Metric
6:         X ← X + 1
7:     end if
8: end while
```

4.4 System Security Analysis Using ADVISE Metrics

The ADVISE method enables security analysts to specify customized metrics relevant to the security objectives of a particular system. The metrics specification determines what output is collected from the model. The particular security decision determines which security metrics are needed.

The ADVISE method produces quantitative metrics that are intended for relative comparisons rather than absolute measurements. Model output can be difficult to validate as an absolute measurement, but relative assessments are often sufficient for supporting security decisions. A system assessment can be either system-focused or adversary-focused.

A system-focused security assessment compares the security strength of different system configurations. An initial assessment is performed on the baseline system. Then the system configuration is modified, the security assessment is repeated, and the results are compared with the baseline system assessment. Configuration changes can affect the system architecture (i.e., the arrangement and connection of components) or the individual components (e.g., firewall settings or the VPN software version). Security analysts can choose to test the security of the system against a single adversary or multiple different adversaries.

Metrics for a system-focused security assessment may include the average time for the adversary to achieve a particular attack goal or the attack path (i.e., series of attack steps) most likely to be used to achieve a particular attack goal. Other variants are possible, such as the probability that an attack goal is achieved by time t or the attack goal that is most likely to be achieved first when there are multiple attack goals.

An adversary-focused security assessment examines how changes in the characterization of an adversary or differences between adversaries affect the apparent security strength of the system. Typically, the system configuration remains constant as the adversary characteristics are varied. The adversary's attack skill levels, attack goals, attack preferences, initial system access, and/or initial system knowledge may be varied.

When multiple adversaries are considered, metrics may measure which adversary is able to achieve a particular attack goal in the shortest average time or which attack path is most likely. An assessment may reveal that different adversaries are likely to choose different attack paths into a system based on their different strengths and attack preferences. Metrics may also measure the total average cost for an adversary to achieve a particular attack goal.

In either type of analysis, the metrics assess different aspects of the average time for an adversary to reach an attack goal, the mostly likely attack path to an attack goal, and the average total cost for an adversary to reach an attack goal.

For example, if a security objective of the system is to protect the availability of wireless-networked devices, then a security analyst could conduct a system-focused ADVISE security assessment and examine how modifying the wireless security settings and the device security settings impacts the average time required for an adversary to make the device unavailable.

To determine the relative advantage of an insider adversary over an external adversary, the security analyst can perform an adversary-focused security assessment. The insider adversary would possess more initial system access (such as local physical access to a substation) and more initial system knowledge (such as wireless network passwords) than an external adversary would. The analyst can study the differences between the two adversaries in the average total attack cost and the average time to make the substation device unavailable. After determining the magnitude of the insider threat, the security analyst can also use the model to assess the effectiveness of security mechanisms intended to protect against the insider threat.

4.5 Example Security Metrics

Different security metrics are useful for measuring different aspects of system security. The ADVISE security analysis method enables the generation of many types of security metrics.

Metrics can assess the attack goals an adversary can achieve:

- The probability that a particular attack goal is achieved at least once during the analysis time interval $[0, \tau]$,
- The average time until a particular attack goal is first achieved (given that the goal is achieved during the analysis time interval), and
- The frequency with which a particular attack goal is achieved during the analysis time interval.

The frequency metric only makes sense for an ADVISE system model that is coupled with a dynamic defense model. The dynamic defense model can detect compromise and return the ADVISE system model to an uncompromised state. In that situation, the adversary could compromise the system multiple times within some analysis time interval.

Metrics can provide insight on common or likely targets:

- The probability that a particular attack step is attempted,
- The frequency of attempts for a particular attack step, and
- The probability of a particular attack step outcome given that the attack step is attempted.

In particular, metrics can give insight on the preferred attack path of an adversary:

- Given that a particular goal is achieved, the probability that a particular attack step is attempted during that same simulation run, and

- Given that a particular goal is achieved, the probability that a particular attack step outcome occurs during that same simulation run.

Metrics can provide insight on the criticality of certain components:

- If a particular attack step outcome occurs, the probability that a particular attack goal is achieved during the same simulation run, and
- If a particular attack step outcome does not occur, the probability that a particular attack goal is achieved during the same simulation run.

Metrics can aid in impact assessment:

- The probability that some combination of attack goals is achieved that represents a higher-level system service measure, and
- The probability that the system meets a particular security objective (e.g., the adversary does not have access to the internal network) for some period of time.

The ADVISE metric specification enables security analysts to capture a wide range of security-relevant measurements from ADVISE models.

CHAPTER 5

CASE STUDIES

The ADVISE method was implemented as a new atomic model type in the Möbius modeling tool [18], which we used to perform two case studies. The case studies demonstrate how the quantitative security metrics produced by ADVISE can aid system design and provide insight on system security. Appendix A describes the implementation of the ADVISE method within the Möbius modeling framework.

System design often involves trade-off decisions involving cost, response time, reliability, security, and other performance metrics. Often, improving one aspect of the system performance (e.g., reliability) negatively impacts another (e.g., cost). Quantitative security metrics enable system architects to make informed trade-off decisions by quantifying the change in security when the system design is modified.

The first case study analyzes the security of a generic SCADA system. The second case study is a more detailed look at the security of an electric power distribution system. Both studies consider several types of adversaries.

5.1 Generic SCADA System

For the first case study, we use ADVISE to evaluate the security of two variants of a supervisory control and data acquisition (SCADA) system architecture. We measure the security of the system relative to five different adversaries: a nation-state, a terrorist organization, a lone hacker, a disgruntled employee, and a disgruntled system administrator.

This chapter contains previously published material by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [2] is reused by permission of IEEE and was published in Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).

5.1.1 SCADA System Architectures

SCADA systems use a centralized control center to remotely monitor and control devices located at distant field sites. SCADA systems are used to control electric power grids, water distribution and wastewater collection systems, and oil and natural gas pipelines. We analyze two SCADA system architectures presented in Figures 5-1 and 5-3 of the National Institute of Standards and Technology Guide to Industrial Control Systems Security (NIST SP 800-82) [20].

Both SCADA system architectures consist of a corporate network and a control network separated by a firewall. The corporate network is also connected to the Internet through another firewall. In one architecture (see Figure 5.1), the data historian is located in the corporate network. In the other architecture (see Figure 5.2), a demilitarized zone (DMZ) is added between the control and corporate networks, and the data historian is moved to the DMZ network. A data server is also placed in the DMZ to mediate communication between the data historian and machines in the corporate and control networks. In the non-DMZ architecture, machines in the corporate and control networks can communicate with the data historian directly.

The AEG that represents possible attacks against the non-DMZ architecture is shown in Figure 5.3. This AEG consists of 18 attack steps, eight access domains, four knowledge items, two attack skills, and five attack goals. The AEG for the DMZ architecture (shown in Figure 5.4) contains two more attack steps and one more access domain.

5.1.2 Adversaries

We consider five distinct adversaries in our analysis of the two architectures. Each has a unique profile. The Nation-State adversary is well-funded, has a low tolerance for detection, and possesses high skill levels. The Lone Hacker possesses a high technological skill level, and gaining payoff and avoiding detection are equally important. The Terrorist Organization favors attacks with large payoff values and pays little attention to detection risk. However, limited skills may make some attack steps unavailable or unattractive. The Disgruntled Employee and Disgruntled Administrator have limited resources but start the attack with an insider advantage. The Disgruntled Administrator possesses more technical skill than the Disgruntled Employee. Table 5.1 lists the attack preference weights for each adversary. For this analysis, the planning horizon (N) is four for all adversaries.

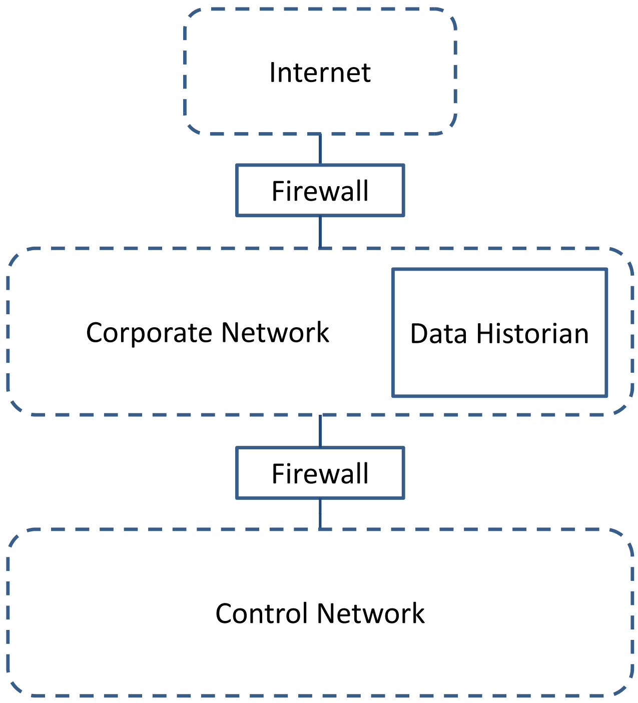


Figure 5.1: SCADA system architecture without a DMZ.

Table 5.1: Attack Preference Weights for Five Adversaries

Adversary	***Cost***	***Payoff***	***Detection***
Nation-State	0.01	0.40	0.59
Lone Hacker	0.20	0.40	0.40
Terrorist Organization	0.05	0.80	0.15
Disgruntled Employee	0.40	0.50	0.10
Disgruntled Administrator	0.40	0.50	0.10

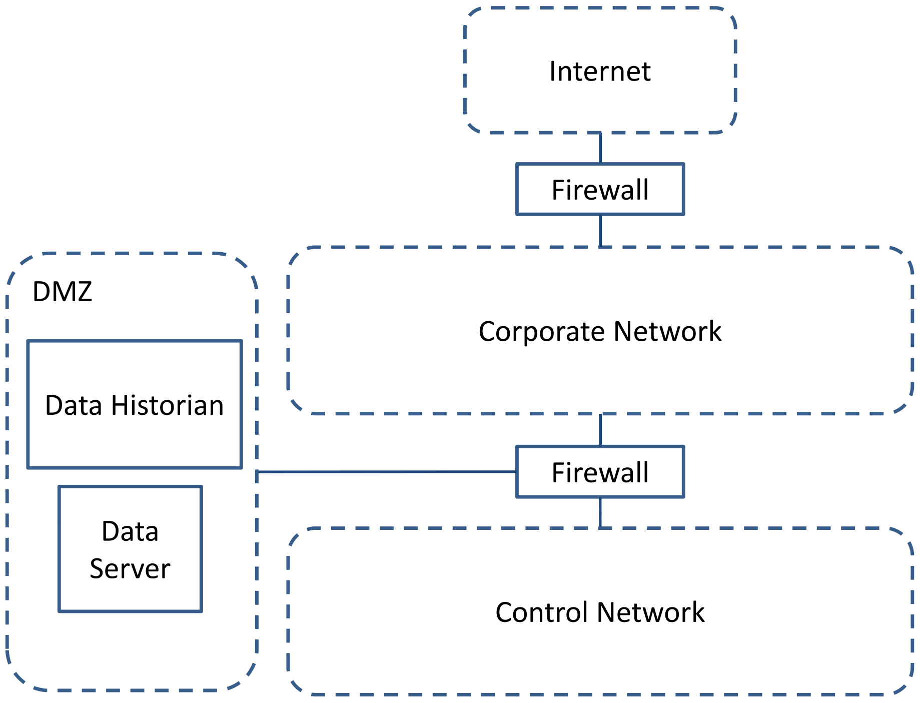


Figure 5.2: SCADA system architecture with a DMZ.

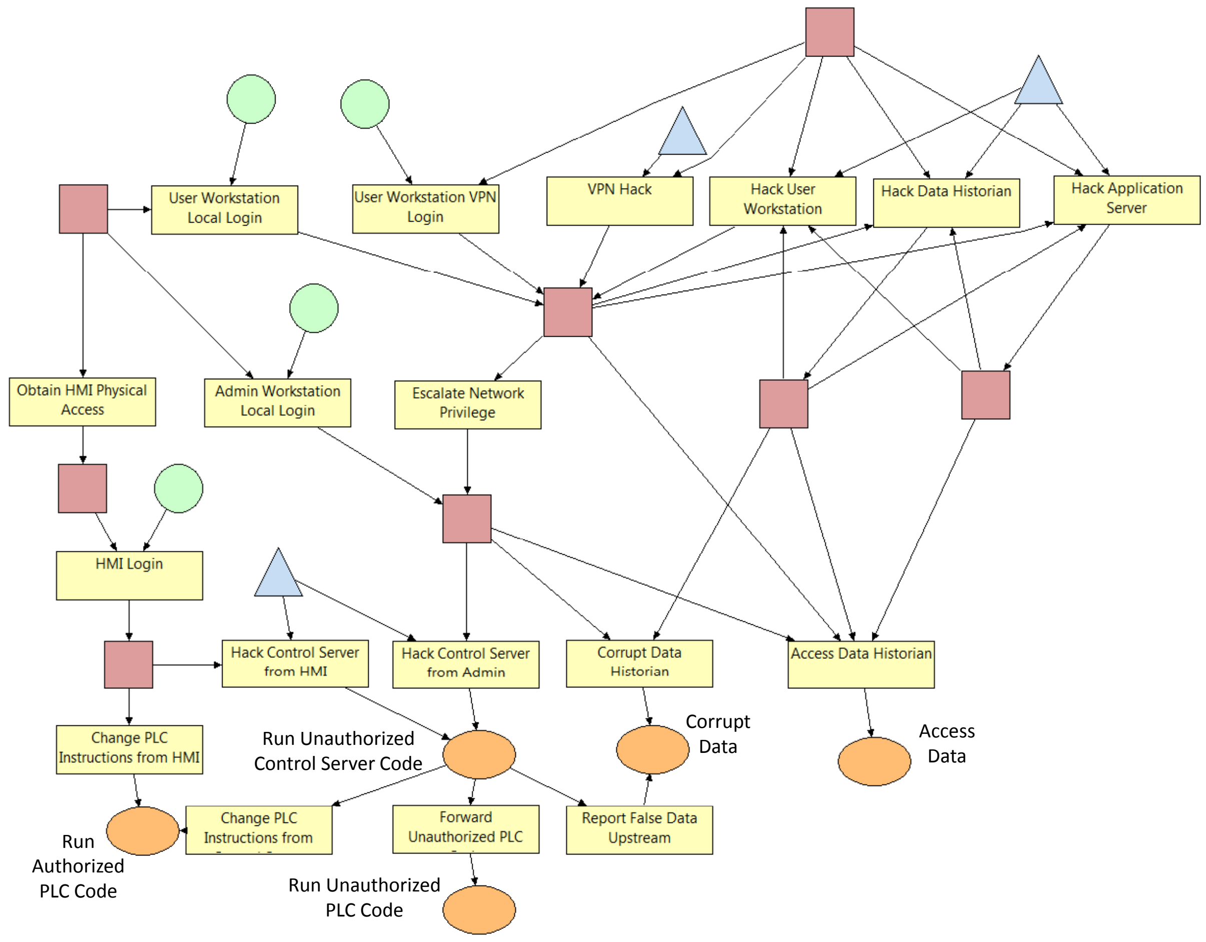

User Workstation Local Login
User Workstation VPN Login
VPN Hack
Hack User Workstation
Hack Data Historian
Hack Application Server
Obtain HMI Physical Access
Admin Workstation Local Login
Escalate Network Privilege
HMI Login
Hack Control Server from HMI
Hack Control Server from Admin
Corrupt Data Historian
Access Data Historian
Change PLC Instructions from HMI
Run Unauthorized Control Server Code
Corrupt Data
Access Data
Change PLC Instructions from
Forward Unauthorized PLC
Report False Data Upstream
Run Authorized PLC Code
Run Unauthorized PLC Code

Figure 5.3: The attack execution graph for the non-DMZ SCADA architecture contains multiple paths to the labeled attack goals.

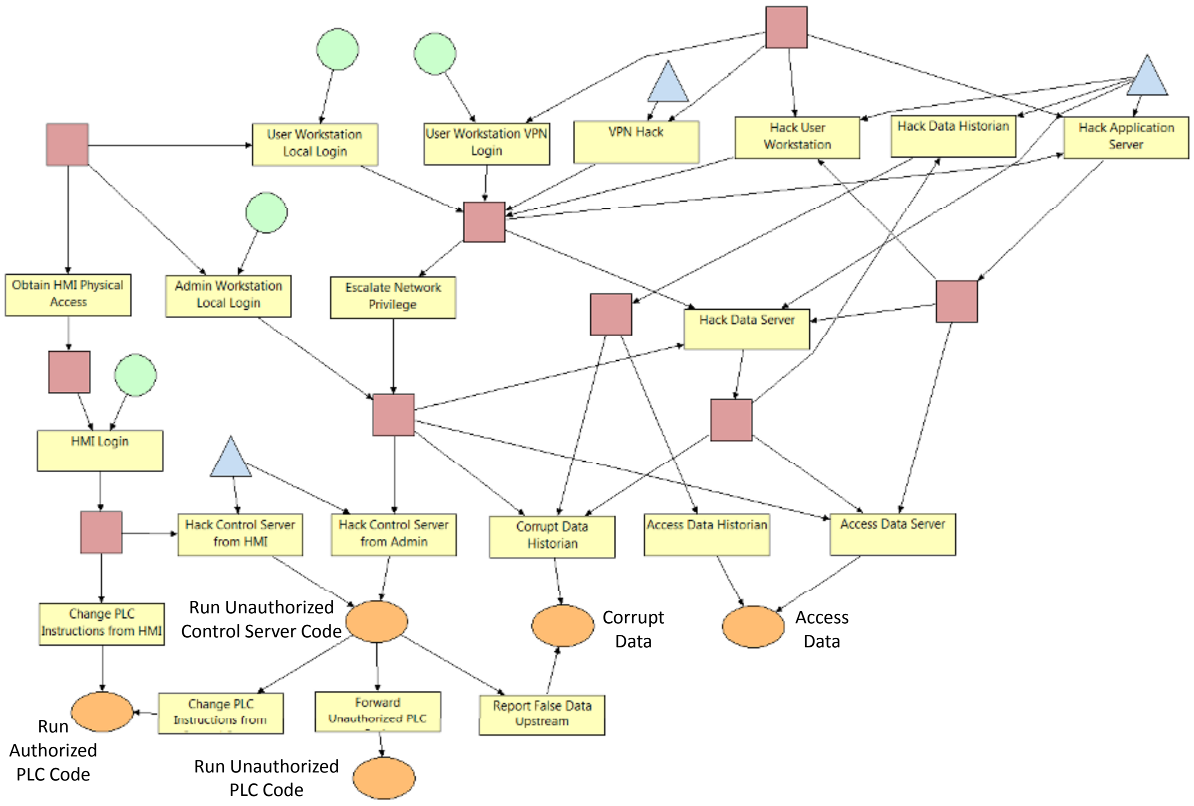

User Workstation Local Login
User Workstation VPN Login
VPN Hack
Hack User Workstation
Hack Data Historian
Hack Application Server
Obtain HMI Physical Access
Admin Workstation Local Login
Escalate Network Privilege
Hack Data Server
HMI Login
Hack Control Server from HMI
Hack Control Server from Admin
Corrupt Data Historian
Access Data Historian
Access Data Server
Change PLC Instructions from HMI
Run Unauthorized Control Server Code
Corrupt Data
Access Data
Run Authorized PLC Code
Change PLC Instructions from
Forward Unauthorized PLC
Report False Data Upstream
Run Unauthorized PLC Code

Figure 5.4: The attack execution graph for the DMZ SCADA architecture is slightly larger than the one for the non-DMZ SCADA architecture because there are two additional attack steps (*Hack Data Server* and *Access Data Server*) to access the DMZ.

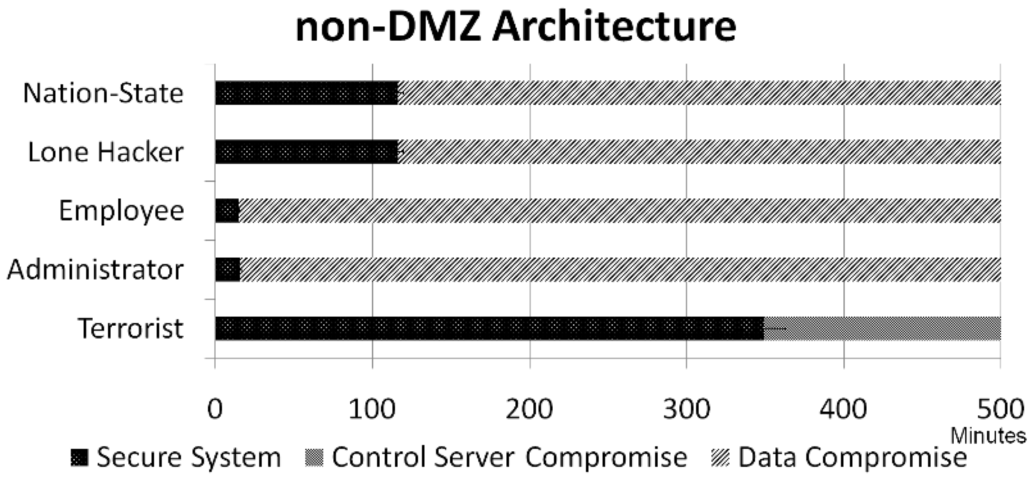


Figure 5.5: This graph shows the average time during $[0, 500 \text{ min}]$ that the non-DMZ system is in a secure state. All adversaries other than the Terrorist Organization are attempting to access data on the data historian. The Terrorist Organization is attempting to run unauthorized code on the control server.

5.1.3 Metrics

To assess how quickly the adversaries can achieve each attack goal $g \in G$, we measure the average amount of time during the time interval $[0, 500 \text{ min}]$ spent in a model state in which goal g has been achieved.

Formally, we specify a state metric for each attack goal $g \in G$: $\langle \tau, \lambda, \sigma \rangle$, where $\tau = 500$, $\lambda = AvgTime$, and $\sigma(s) =$ True, if $g \in G_s$; False, otherwise.

5.1.4 Results and Analysis

Different adversaries (each with different attack preferences, attack goals, and initial states) can attack the same system and achieve different results. When the system is modified, the adversaries can respond differently to changes.

For each adversary, we ran discrete-event simulations of our ADVISE executable models and measured the average time during $[0, 500 \text{ min}]$ that the system was in a secure state. Figures 5.5 and 5.6 reveal how each adversary fared in attacking the DMZ and non-DMZ SCADA architectures. The two attack goals of interest are compromising data on the data historian and compromising the control server on the control network. The figures contain 95% confidence intervals.

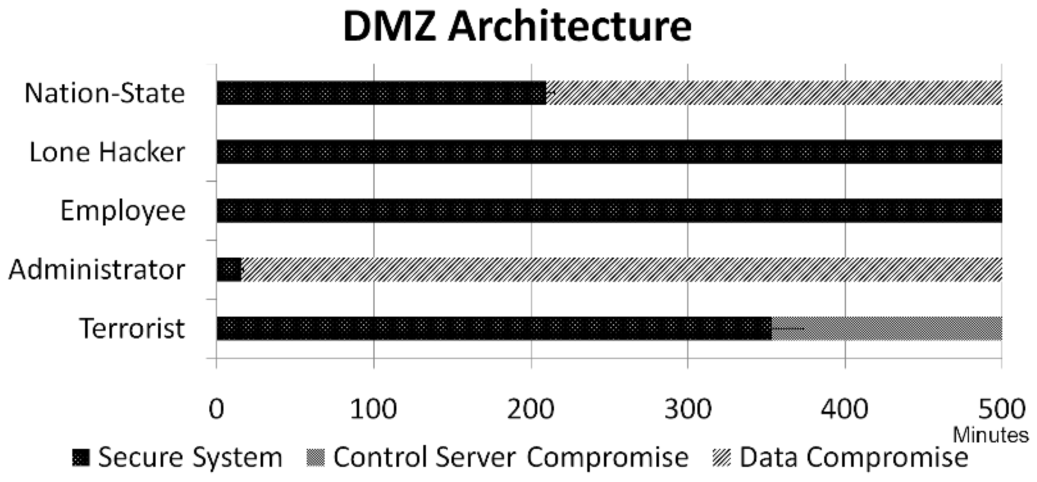


Figure 5.6: This graph shows the average time during $[0, 500 \text{ min}]$ that the DMZ system is in a secure state. All adversaries other than the Terrorist Organization are attempting to access data on the data historian. The Terrorist Organization is attempting to run unauthorized code on the control server.

In both architectures, the Nation-State compromises data, but it takes longer to compromise the DMZ architecture because more attack steps are required to accomplish the compromise when the data historian is stored in the DMZ.

The Lone Hacker only compromises data in the non-DMZ architecture. For the Lone Hacker, the increased number of attack steps to compromise data in the DMZ architecture is enough to deter him or her from attempting the attack. In this case, the "do-nothing" attack step is more attractive than any of the other attack steps. When the "do-nothing" attack step is the most attractive, the system enjoys a form of system protection based on deterrence.

Both the Nation-State and the Disgruntled Employee compromise data in the non-DMZ architecture, but the Disgruntled Employee achieves the data compromise much more quickly due to his or her insider status. Logging on to the corporate network is much faster than hacking in from the Internet.

However, the Disgruntled Employee does not attempt the data compromise attack against the DMZ architecture. The Disgruntled Employee lacks the technical attack skills needed to access the data historian when it is in the DMZ.

Unlike the Disgruntled Employee, the Disgruntled Administrator does have the technical attack skills to gain access to data in the DMZ, so the DMZ architecture is no more secure than the non-DMZ architecture against this threat. (Make sure you can trust your system administrators!)

The addition of the DMZ also does not affect the Terrorist Organization's attack against the control server because the Terrorist Organization's path to the control server on the control network does not change when the DMZ is added. The DMZ does not address the issue of control server compromise.

In summary, the case study shows that the DMZ SCADA architecture offers better protection than the non-DMZ architecture against data compromise by a Nation-State, a Lone Hacker, or a Disgruntled Employee, but a Disgruntled Administrator remains unimpeded by the DMZ. Also, the DMZ does not impact the ability of the Terrorist Organization to compromise the control server in the control network. In this case, adding one security defense mechanism does not protect against all types of adversaries and all types of compromise. The system architects may want to reconsider their design. The quantitative security metrics produced by ADVISE can aid system design by enabling such insights on system security.

5.2 Electric Power Distribution System

The second case study is a more in-depth security analysis using the ADVISE method. We examine the security of an electric power distribution system, considering attacks from several types of adversaries.[1] We examine how changes in the system impact how adversaries decide to attack the system. We also study how a targeted security enhancement impacts adversary attack decisions. In particular, the purpose of this analysis is to gain insight on the likely paths chosen by different adversaries and the relative speed of their attacks.

5.2.1 System Description

The electric power infrastructure consists of three main parts: electricity generation, transmission, and distribution. Electric power generation converts other sources of energy into electricity using nuclear power plants, fossil fuel power plants, hydroelectric dams, and wind farms. Electric power transmission carries electricity from the generation sources to substations. Transmission carries electricity long distances at very high electric voltages. At the

[1]Disclaimer: The system architectures used in this study are realistic, but component and system vulnerabilities were fabricated. As a result, the quantitative results are not an accurate assessment of an actual electric power distribution system.

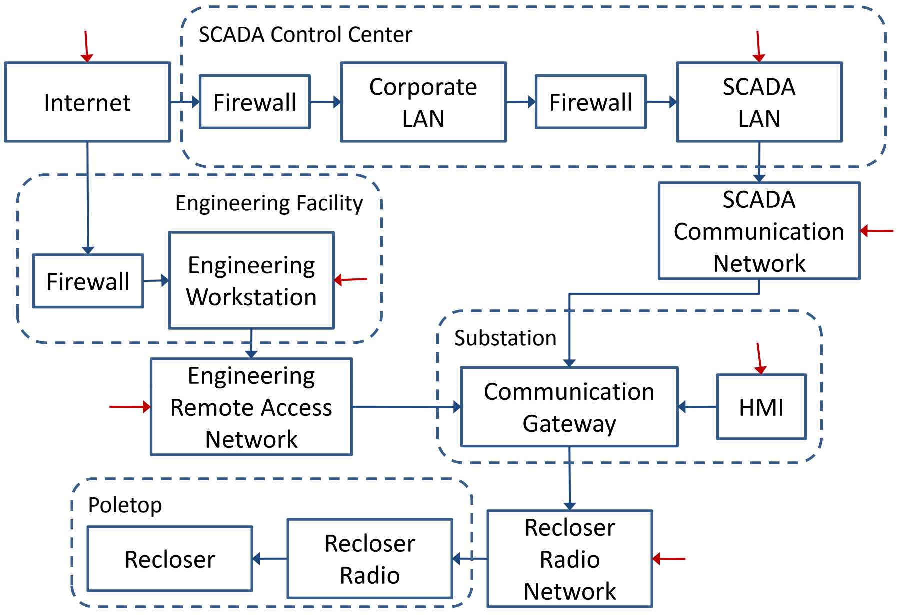


Figure 5.7: The SCADA architecture of an electric power distribution system contains many different components. The diagram emphasizes the connections between different types of components. A real system would contain multiple substations and multiple poletop units. The arrows in the diagram represent the locations of possible attacks against the system.

substations, the electric power is transformed to lower voltages. Then, the electric power distribution system carries the electricity from the substations to the electric meters on individual homes and businesses.

In this analysis, we study the security of the electric power distribution system, considering both physical attacks against the distribution system equipment and cyber attacks against the control network. Figure 5.7 provides an overview of the system components and connections relevant for our analysis. In the diagram, the arrows between components indicate possible attack paths, and the arrows pointing into components indicate possible attack entry points. The system diagram is based on Figure 11.7 in a book published by Schweitzer Engineering Laboratories [21] and discussions with researchers at GE Global Research [22].

The substations in the distribution system are controlled remotely by operators at the SCADA control center. The operators send commands from the SCADA LAN through the SCADA communication network to the communication gateway in each substation. The SCADA LAN is connected to the corporate LAN, and the corporate LAN is also connected to the Internet. Firewalls should be configured to disallow Internet traffic to reach the SCADA LAN. The substation can also be controlled locally by the human-machine interface (HMI); this local control option allows technicians to change the settings as needed. A third access vector into the substation is from engineering workstations connecting through the engineering remote access network. This special access allows engineers to service substations remotely.

In addition to substations, the distribution system includes lines, circuit breakers, and auto-reclosers. When a circuit breaker detects a fault on a line carrying electricity, the circuit breaker opens and stops the flow of electricity. At preset intervals, an auto-recloser repeatedly attempts to close the circuit breaker and determine if the fault condition has been corrected. If the fault has been cleared, then the circuit breaker is closed. If the fault remains, then the circuit breaker remains open until a technician removes the cause of the fault and closes the circuit breaker. We use the term "recloser" to refer to a circuit breaker and auto-recloser together.

When a radio is added to the recloser-unit, commands can be relayed remotely from the substation to the recloser, and sensor data can be sent from the recloser radio back to the substation. This remote communication between substations and reclosers enables a faster response time to problems but also opens up the system to more attack vectors. It is reasonable to question whether the benefits of installing recloser radios outweigh the risks. By measuring the security of a system without and with recloser radios, we can better answer this question by being better informed about the magnitude of possible risks.

5.2.2 Possible Attacks Against the System

Adversaries attacking the electric power distribution system may have different attack goals. Some adversaries want to temporarily disrupt electricity service. Others want to damage the distribution system equipment and cause longer service outages. Still others want to surreptitiously install malicious software on the control network or the individual pieces of equipment. The malicious software could provide the adversary with backdoor access or other unauthorized access or control of the system.

We build an attack execution graph (shown in Figure 5.8) to represent the possible sequences of attack steps that an adversary could execute in order to achieve one or more of those attack goals. Figure 5.9 shows how the attack execution graph in Figure 5.8 corresponds with the architecture diagram in Figure 5.7.

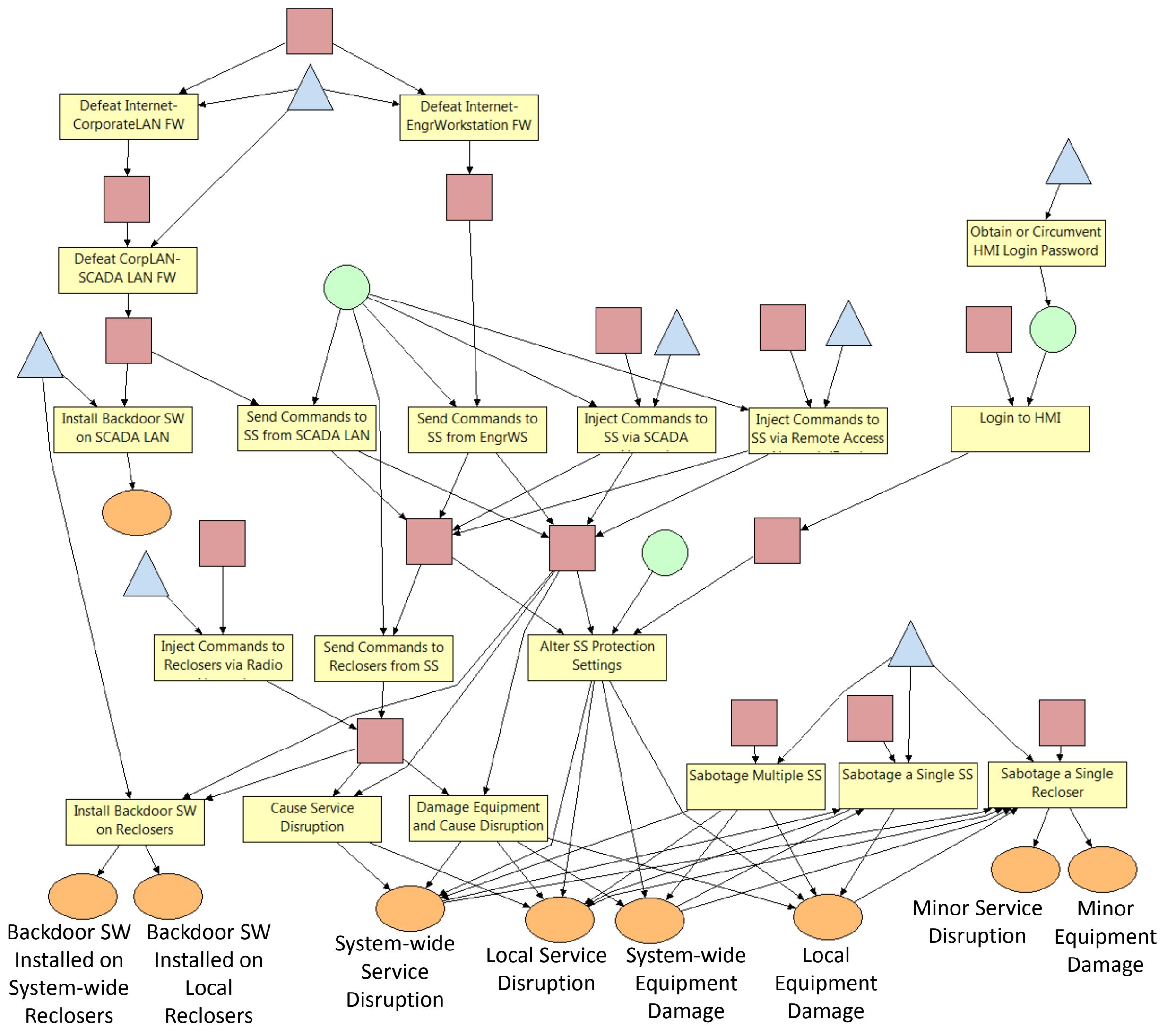

Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Obtain or Circumvent HMI Login Password
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption
Backdoor SW Installed on System-wide Reclosers
Backdoor SW Installed on Local Reclosers
System-wide Service Disruption
Local Service Disruption
System-wide Equipment Damage
Local Equipment Damage
Minor Service Disruption
Minor Equipment Damage

Figure 5.8: Attack execution graph for an electric power distribution system (with attack goals labeled).

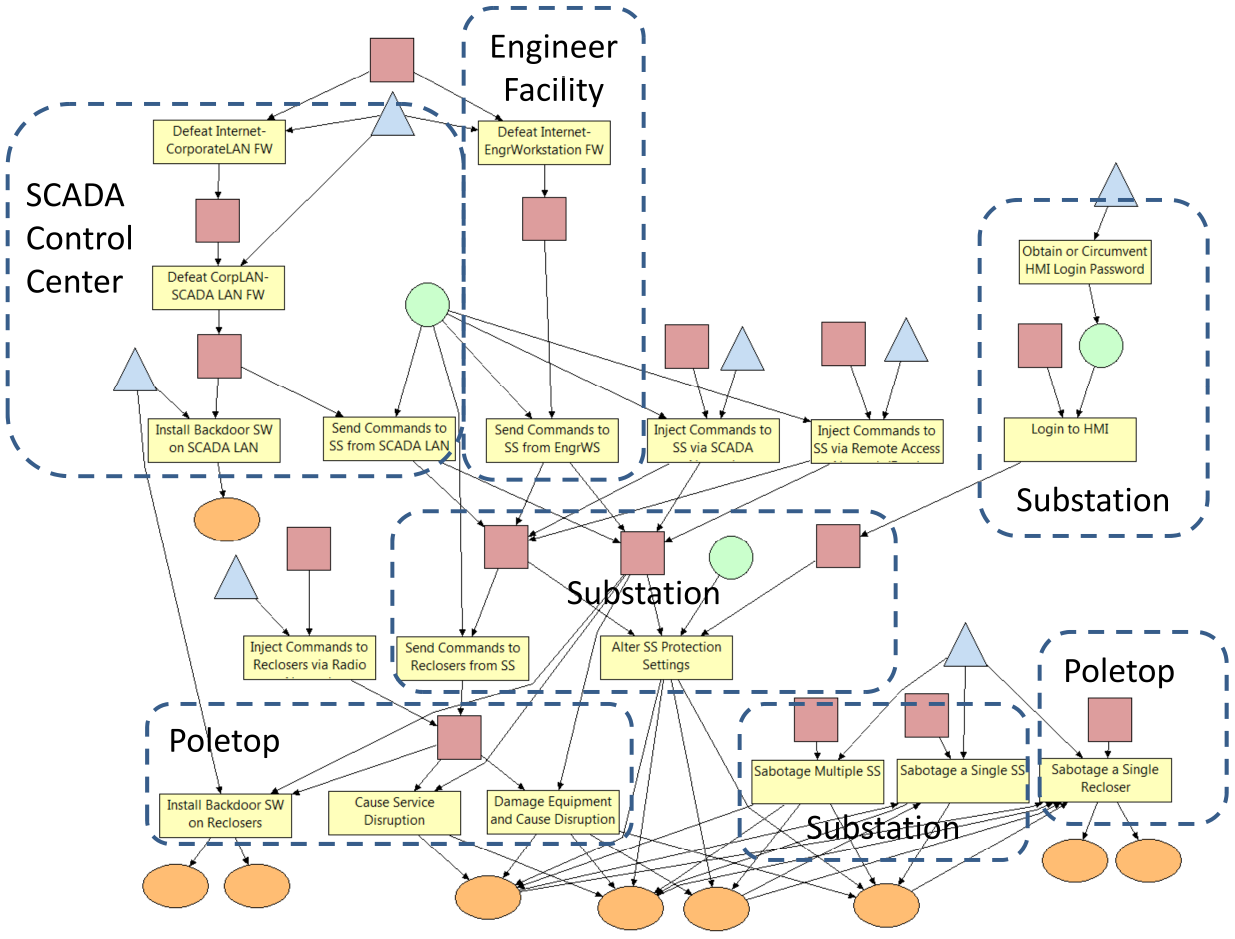

Engineer Facility
SCADA Control Center
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Obtain or Circumvent HMI Login Password
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Substation
Substation
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Poletop
Poletop
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Substation

Figure 5.9: Attack execution graph for an electric power distribution system with SCADA architecture components labeled.

The attack execution graph contains the following attack steps, described informally:

- *DefeatInternetCorporateLANFW*: An adversary with Internet access and skill in attacking firewalls can defeat the firewall between the Internet and the corporate LAN and thereby gain unauthorized access to the corporate LAN.

- *DefeatCorpLANSCADALANFW*: An adversary with corporate LAN access and skill in attacking firewalls can defeat the firewall between the corporate LAN and the SCADA LAN and thereby gain unauthorized access to the SCADA LAN.

- *InstallBackdoorSWonSCADALAN*: An adversary with SCADA LAN access and skill in backdoor software can install malicious backdoor software on the SCADA LAN.

- *DefeatInternetEngrWorkstationFW*: An adversary with Internet access and skill in attacking firewalls can defeat the firewall between the Internet and the engineering workstation and thereby gain unauthorized access to the engineering workstation.

- *SendCommandstoSSfromSCADALAN*: An adversary with SCADA LAN access and knowledge of SCADA protocols can send unauthorized commands from the SCADA LAN to the substations and thereby gain control of the substation communication gateway.

- *SendCommandstoSSfromEngrWS*: An adversary with engineering workstation access and knowledge of SCADA protocols can send unauthorized commands from the engineering workstation to the substations and thereby gain control of the substation communication gateway.

- *InjectCommandstoSSviaSCADANetwork*: An adversary with physical access to the SCADA communication network, the skills to perform traffic analysis and packet injection on the SCADA communication network, and knowledge of SCADA protocols can inject unauthorized commands to the substation via the SCADA communication network and thereby gain control of the substation communication gateway.

- *InjectCommandstoSSviaRemoteAccess*: An adversary with access to the engineering remote access network, the skills to perform traffic analysis and packet injection on the engineering remote access network, and knowledge of SCADA protocols can inject unauthorized commands to the substation via the engineering remote access network and thereby gain control of the substation communication gateway.

- *ObtainorCircumventHMILoginPassword*: An adversary with skill in attacking passwords can obtain or circumvent the HMI login password.

- *LogintoHMI*: An adversary with physical access to the HMI and knowledge of the HMI login password can gain control of the HMI in the substation.

- *InjectCommandstoReclosersViaRadio*: An adversary with physical access to the recloser radio network and skill in traffic analysis and packet injection on the recloser radio network can inject unauthorized commands and gain control of the reclosers.

- *SendCommandstoReclosersfromSS*: An adversary with control of the substation communication gateway (i.e., able to send unauthorized commands to the substation) and knowledge of SCADA protocols can send unauthorized commands from the substation to the reclosers and thereby gain control of the reclosers.

- *AlterSSProtectionSettings*: An adversary with control of either the substation HMI or the substation communication gateway as well as knowledge of substation protection settings can alter the settings in such a way as to damage distribution equipment and disrupt electricity service. If the adversary is physically at a substation using the HMI, then the effects are limited to the local service area. If the adversary has remote control over multiple substation communication gateways, then the effects are system-wide.

- *InstallBackdoorSWonReclosers*: An adversary with control of reclosers and skill in backdoor software can install malicious backdoor software on the reclosers. If the adversary has remote control over multiple substation communication gateways, then the software is installed on reclosers throughout the distribution system. Otherwise, the software is installed only on local reclosers (all the reclosers connected to one substation in the distribution system).

- *CauseServiceDisruption*: An adversary with control of the reclosers can cause an electricity service disruption. If the adversary has remote control over multiple substation communication gateways, then the service disruption is system-wide. Otherwise, the service disruption is limited to the local service area of one substation.

- *DamageEquipmentandCauseDisruption*: An adversary with control of the reclosers can damage distribution equipment and cause an electricity service disruption. If the adversary has remote control over multiple substation communication gateways, then the service disruption and damage are system-wide. Otherwise, the service disruption and damage are limited to the local service area.

- *SabotageMultipleSS*: An adversary with physical access to multiple substations and skill in physical sabotage can sabotage multiple substations, causing system-wide equipment damage and electricity service disruptions.
- *SabotageaSingleSS*: An adversary with physical access to a single substation and skill in physical sabotage can sabotage a single substation, causing local equipment damage and electricity service disruptions in the local service area of one substation.
- *SabotageaSingleRecloser*: An adversary with physical access to a single recloser and skill in physical sabotage can sabotage a single recloser, causing minor equipment damage and electricity service disruptions.

Each attack step is formally defined according to the ADVISE formalism described in Chapter 2. Here we provide the details for the *DefeatCorpLANSCADALANFW* attack step. The attack cost is defined as 10 cost units. The attack execution time is normally distributed with the mean dependent on the adversary's level of skill in attacking firewalls, and the variance is equal to the mean. The distribution's mean is defined as a function of the firewall attack skill level, as shown in Figure 5.10. (Note that because the time must always be non-negative, and the density of a normal distribution is distributed over all real numbers, the normal distribution is truncated at zero for the execution time distribution.) The attack step precondition is that the adversary must have access to the corporate LAN, not have access to the SCADA LAN, and possess a firewall attack skill level greater than zero.

There are two possible outcomes of the *DefeatCorpLANSCADALANFW* attack step: Success and Failure. The probability of each outcome depends on the firewall attack skill level of the adversary. If the adversary has a firewall attack skill level less than 0.7, then the probability of the Success outcome is zero. Otherwise, the probability of the Success outcome is equal to the firewall skill level value. Recall that skill levels are values in the range $[0, 1]$. Since there are only two outcomes for this attack step, the probability of the Failure outcome is one minus the probability of the Success outcome.

The probability of detection also depends on the adversary's firewall attack skill level. For the Success outcome, if the adversary's firewall attack skill level is greater than or equal to 0.9, then the probability of detection is 0.01; otherwise, the probability of detection is 0.1.

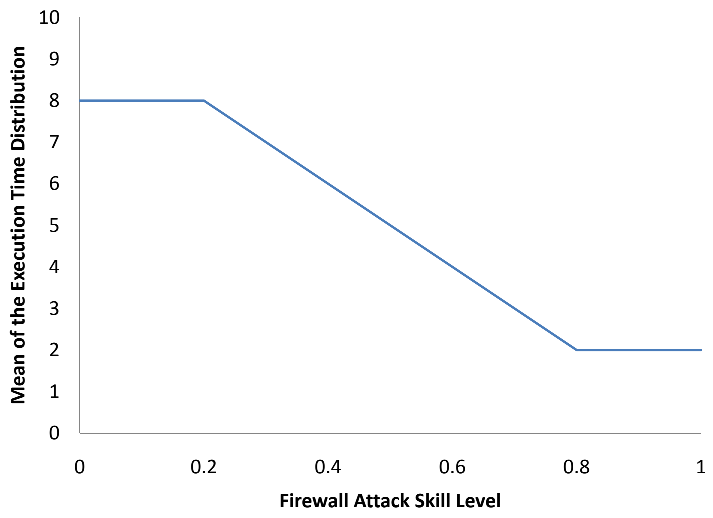


Figure 5.10: For the *DefeatCorpLANSCADALANFW* attack step, the attack execution time distribution's mean is a function of the firewall attack skill level.

For the Failure outcome, if the adversary's firewall attack skill level is greater than or equal to 0.9, then the probability of detection is 0.05; otherwise, the probability of detection is 0.5.

The effect of the Success outcome on the model state is that the adversary gains SCADA LAN access. The Failure outcome has no effects on the model state. Recall that the ADVISE formalism does not restrict the effect of the attack attempts on model state. Outcome effects can add or take away from the model state any combination of access, knowledge, and/or goals achieved.

Examining the details for the *DefeatCorpLANSCADALANFW* attack step has demonstrated how ADVISE can handle attack step definitions dependent on characteristics of the adversary. The attack step definition can also contain dependencies on the model state.

5.2.3 Adversary Characterization

For this analysis, we consider six different adversaries: a Foreign Government, a Hacker, a Hostile (Terrorist) Organization, an Insider Engineer, an Insider SCADA Operator, and an Insider Technician. As we introduce the adversary characterizations used for this analysis, we discuss similarities and differences among the adversaries.

Attack Decision Parameters

Table 5.2 provides the core decision parameters for the adversaries. Recall that the preference weights are used in the adversary attack decision to evaluate the relative attractiveness of attack steps, and the planning horizon is the number of steps into the future the adversary will consider when evaluating the attractiveness of the possible next attack steps. For this analysis, all adversaries have the same planning horizon.

The Foreign Government adversary is very well-funded, so cost is not a factor in the attack decisions considered in this attack execution graph. Given the payoff scale used for this attack execution graph (a payoff of 1000 has utility one), the Foreign Government has equal preference weights for detection and payoff. Of all the adversaries, the Foreign Government places the most significance on the detection probability when making an attack decision.

The Hacker is resourced-constrained, so the cost preference weight is nonzero given the cost scale used here (a cost of 100 has utility zero). The Hacker has equal preference weights for detection and payoff.

The Hostile Organization does consider attack cost when making attack decisions but does not weight it as heavily as the Hacker does. The Hostile Organization is driven by payoff more than any of the other adversaries.

The three Insiders share the same preference weights. They share the same cost preference weight as the Hacker, since all four are resource-constrained individuals for whom attack cost can be a deterrent. The three Insiders have a relatively high payoff preference weight and a relatively low detection preference weight.

Attack Skill Levels

Table 5.3 contains the skill level values for each adversary. The Foreign Government can attack only from a remote location and therefore does not have the skills that require physical proximity to the electric power distribution system. However, the Foreign Government is highly competent at the skills that it does possess. The Hacker has all the skills except the physical sabotage skill. The Hacker is generally very skilled in cyber attacks, although the Hacker's skill level is not as high for the specialized traffic analysis and injection skills. The Insider Engineer is very familiar with traffic sent in the distribution system, so the Insider Engineer is most competent at traffic analysis and injection. To a lesser degree, the Insider Engineer possesses the other attack skills. The Insider Operator and Insider Technician possess only the physical sabotage skill.

Table 5.2: Adversary Characterization: Decision Parameter Values

Decision Parameters	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
Planning Horizon	6	6	6	6	6	6
Cost Preference Weight	0.2	0.2	0.2	0	0.05	0.2
Detection Preference Weight	0.1	0.1	0.1	0.5	0.2	0.4
Payoff Preference Weight	0.7	0.7	0.7	0.5	0.75	0.4

Table 5.3: Adversary Characterization: Skill Level Values for Each Attack Skill

Attack Skill	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
Firewall Attack Skill	0.3	0	0	0.9	0.7	0.9
SCADA Network Traffic Analysis and Injection Skill	0.5	0	0	0	0.7	0.7
Engr Network Traffic Analysis and Injection Skill	0.5	0	0	0.9	0.7	0.7
Recloser Radio Traffic Analysis and Injection Skill	0.5	0	0	0	0.7	0.7
Backdoor SW Skill	0.3	0	0	0.9	0.7	0.9
Physical Sabotage Skill	0.3	0.3	0.3	0	0.9	0
Password Attack Skill	0.3	0	0	0.9	0.7	0.9

Initial Access

Table 5.4 indicates which access domains each adversary can already access at the start of the attack simulation. All adversaries start with Internet access. The Insider Engineer starts with additional access to the engineering workstation. The Insider Operator starts with access to the SCADA LAN and the Corporate LAN. The Foreign Government and the Hacker have the most limited physical access to the electric power distribution system.

Initial Knowledge

Table 5.5 shows which adversaries possess each type of knowledge at the start of the attack simulation. All adversaries possess some SCADA protocol knowledge. All adversaries except the Hacker have knowledge about the substation protection settings. Only the Insider Engineer and Insider Technician know the substation HMI login password.

Payoff Values for Each Attack Goal

Table 5.6 shows how much each adversary values each attack goal. The Foreign Government is interested only in installing backdoor software, with an emphasis on installing software on the SCADA LAN. The Hacker is most interested in installing backdoor software but is also interested in disrupting electricity service. The Hostile Organization is interested entirely in service disruption and equipment damage. The Insider Engineer wants to install backdoor software or disrupt electricity service. The Insider Operator and Insider Technician want to disrupt electricity service. These attack goals will drive the decisions of the adversaries.

Utility Conversion Functions

The same utility conversion functions were used for all adversaries. These utility functions were also used in the previous SCADA case study in Section 5.1.

For the current example, the cost utility function is intended to convert attack costs from the domain of $[0, 100]$ cost units to the range of $[0, 1]$ utility units. Lower attack step costs have higher utility, so a cost of zero has a utility of one. Higher costs have lower utility, so a cost of 100 has a utility of zero. Equation 5.1 gives the complete mathematical formula for the cost utility function. Figure 5.11 shows the shape of the cost utility curve.

Table 5.4: Adversary Characterization: Initial Access

Access	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
Internet Access	yes	yes	yes	yes	yes	yes
Corporate LAN Access	no	yes	no	no	no	no
SCADA LAN Access	no	yes	no	no	no	no
Engr Workstation Access	yes	no	no	no	no	no
Access to Engr Remote Access Network	yes	no	yes	yes	yes	yes
Control of HMI in SS	no	no	no	no	no	no
Control of SS Gateway	no	no	no	no	no	no
Control of Multiple SS Gateways	no	no	no	no	no	no
Control of All Local Reclosers	no	no	no	no	no	no
Physical Access - SCADA Network	yes	no	yes	no	yes	no
Physical Access - HMI	yes	no	yes	no	no	no
Physical Access - Recloser Radio Network	yes	no	yes	no	yes	no
Physical Access - One Recloser	yes	yes	yes	no	yes	no
Physical Access - One SS	yes	yes	yes	no	yes	no
Physical Access - Multiple SS	yes	yes	yes	no	yes	no

Table 5.5: Adversary Characterization: Initial Knowledge

Knowledge	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
SCADA Protocol Knowledge	yes	yes	yes	yes	yes	yes
SS Protection Settings Knowledge	yes	yes	yes	yes	yes	no
HMI Login Password	yes	no	yes	no	no	no

Table 5.6: Adversary Characterization: Payoff Values for Each Attack Goal

Attack Goal	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
Backdoor SW - SCADA LAN	500			600		500
Backdoor SW - Local Reclosers	50			100		50
Backdoor SW - System-wide Reclosers	100			300		100
Local Service Disruption	25	200	200		25	25
Minor Service Disruption	50	700	700		50	50
System-wide Service Disruption	275	100	100		275	275
Local Equipment Damage					50	
Minor Equipment Damage					100	
System-wide Equipment Damage					500	

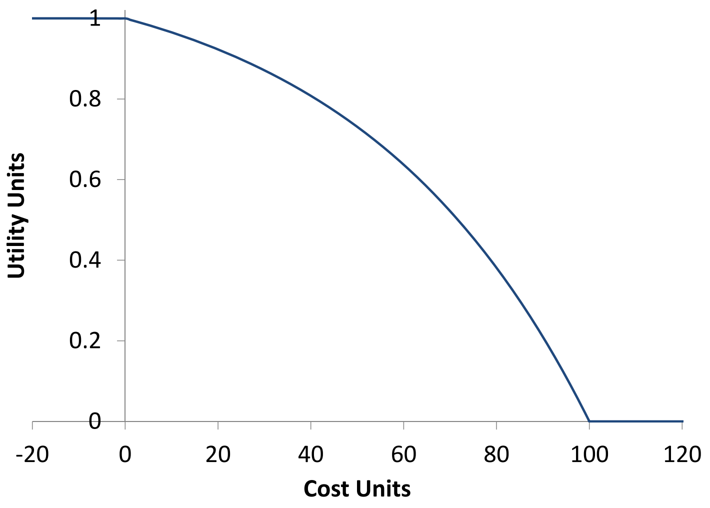


Figure 5.11: The cost utility function converts cost values from traditional cost units (e.g., dollars) to values on the utility scale $[0, 1]$.

$$U_C(c) = \begin{cases} 1, & \text{when } c < 0, \\ \dfrac{e^2 - e^{(c/50)}}{e^2 - 1}, & \text{when } 0 \le c \le 100, \\ 0, & \text{when } c > 100. \end{cases} \tag{5.1}$$

The payoff utility function is intended to convert payoff from the domain of $[0, 1000]$ payoff units to the range of $[0, 1]$ utility units. Lower payoff values have lower utility, so a payoff of zero has a utility of zero. Higher payoff values have higher utility, so a payoff of 1000 has a utility of one. Equation 5.2 gives the complete mathematical formula for the payoff utility function. Figure 5.12 shows the shape of the payoff utility curve.

$$U_P(p) = \begin{cases} 0, & \text{when } p < 0, \\ \dfrac{e^{10/3} - \frac{e^{10/3}}{e^{p/300}}}{e^{10/3} - 1}, & \text{when } 0 \le p \le 1000, \\ 1, & \text{when } p > 1000. \end{cases} \tag{5.2}$$

The detection utility function converts detection from the domain of $[0, 1]$ probability values to the range of $[0, 1]$ utility units. Lower detection probabilities have higher utility, so a detection of zero has a utility of one. Higher detection probabilities have lower utility, so a detection of one has a utility of zero. Equation 5.3 gives the complete mathematical formula for the detection utility function. Figure 5.13 shows the shape of the detection utility curve.

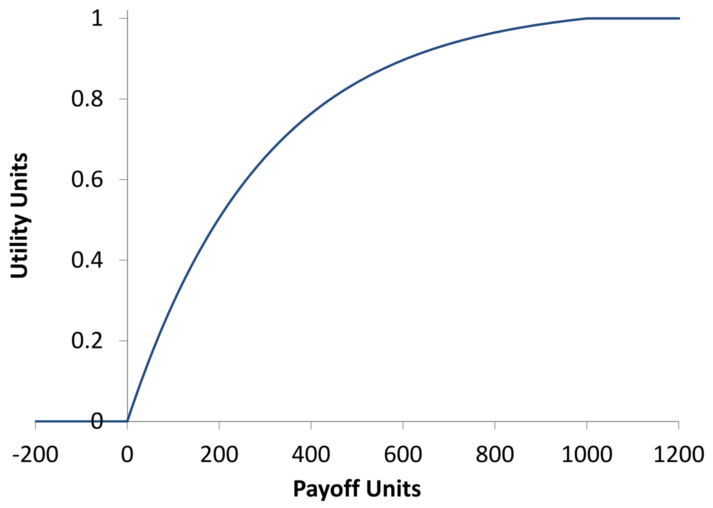


Figure 5.12: The payoff utility function converts payoff values from traditional payoff units (e.g., dollars) to values on the utility scale $[0, 1]$.

$$U_D(d) = \frac{1 - \frac{e^2}{e^{2d}}}{1 - e^2}. \tag{5.3}$$

By adjusting the utility conversion functions and the attack preference weights, we can customize the ADVISE decision rule to reflect a wide variety of adversary decision priorities. However, to generate provably optimal attack decisions using the alternative ADVISE decision rule, the utility conversion functions must be linear over their appropriate domains.

5.2.4 Metrics

The purpose of this analysis is to gain insight on the likely paths chosen by different adversaries and the relative speed of their attacks, so we choose our metrics accordingly.

To study the preferred attack path of each adversary, we measure the average number of attempts per attack step. Attack steps with a nonzero number of attempts are on the preferred attack path of that adversary. Also, a high average number of attempts can indicate an attack step that an adversary tries several times before being successful.

To determine which goals the adversary is both willing and able to achieve, we measure the probability that an adversary has achieved the attack goal at the end of the simulation time interval (50 hours). This measurement is collected for each attack goal.

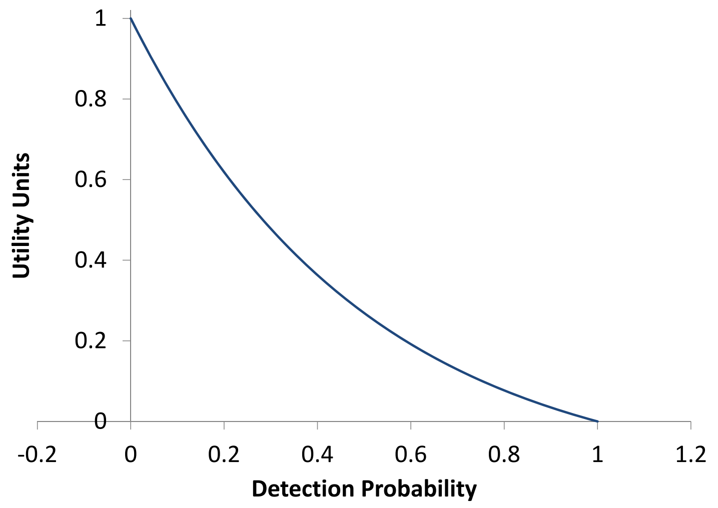


Figure 5.13: The detection utility function converts detection probabilities to values on the utility scale $[0, 1]$.

To study the speed of the adversary's attack, we measure the average time-to-achieve-goal for each attack goal for which the end-probability measurement is one or very close to one.

5.2.5 Results and Analysis

Having fully specified the attack execution graph, the adversary profiles, and the security metrics, we now analyze and discuss the simulation results.

The ADVISE model is simulated using two system configurations. One configuration is an electric power distribution system with recloser radios that enable communication between substations and reclosers. The second configuration is a system without recloser radios. For each configuration, we individually simulate each of the six adversaries attacking the system. All experiments are simulated for 200 runs, which is sufficient to produce results with 95% confidence intervals that are within $\pm 10\%$ of the sample mean. The time values in the model can be interpreted in time units of hours. The cost units can be interpreted as dollars.

We first show a visualization of the expected attack path of each adversary for each of the two system configurations (with and without recloser radios). Then we individually examine the attack of each adversary, referring to the tables of measurements collected from the simulations.

Figures 5.14 and 5.15 show how the six adversaries choose to attack the system. The

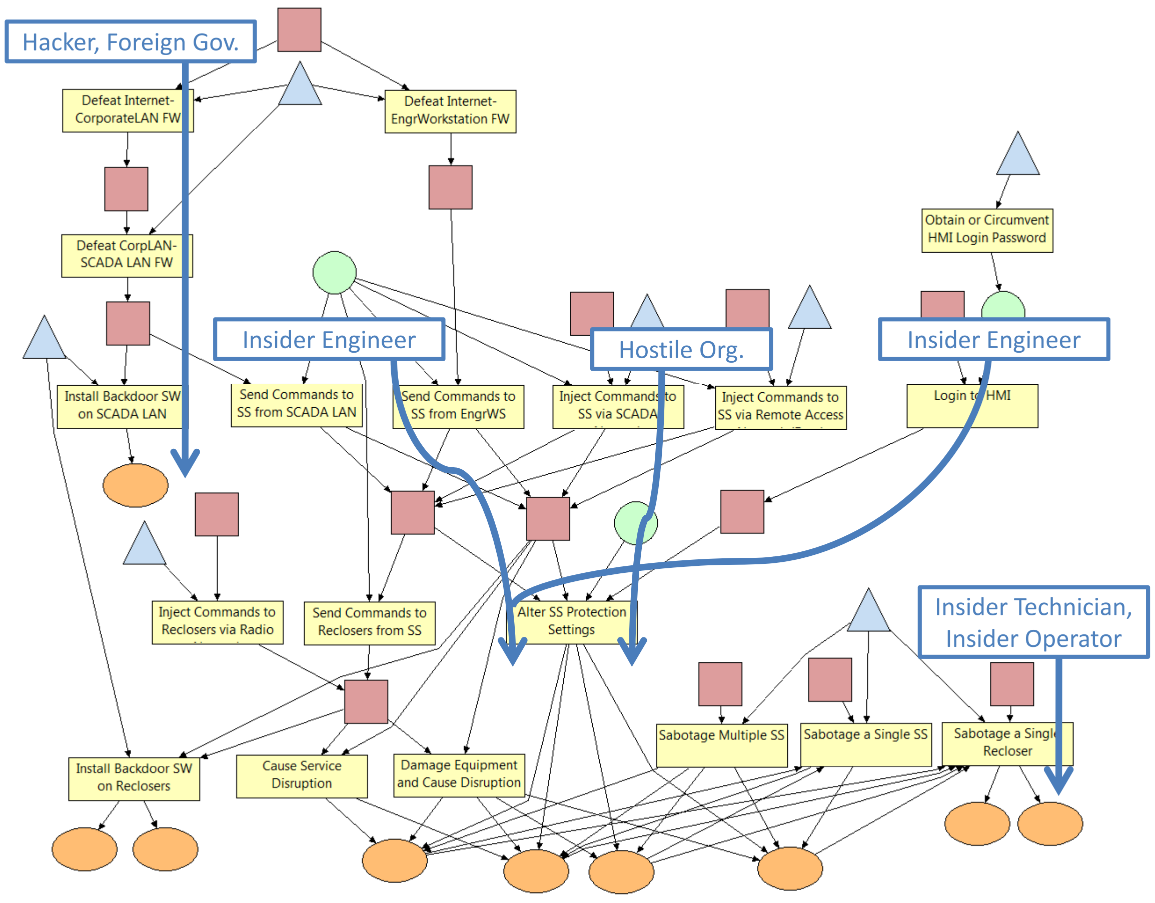

Hacker, Foreign Gov.
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Obtain or Circumvent HMI Login Password
Insider Engineer
Hostile Org.
Insider Engineer
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Insider Technician, Insider Operator
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption

Figure 5.14: Adversaries' preferred attack paths (without recloser radios).

preferred attack paths are overlaid on the attack execution graph shown earlier in Figure 5.8. The expected attack path visualizations were constructed by noting which attack steps were attempted at least once in the time interval $[0, 50]$ hours. A bold line was drawn through those steps to indicate the progression of the adversary's attack. Notice that when the addition of recloser radios opens up a new attack vector, that vector is attractive enough that the Hacker, the Hostile Organization, the Insider Engineer, and the Insider Operator change their preferred attack paths. However, the Insider Technician and the Foreign Government keep the same preferred paths.

To avoid confusion, we point out that a preferred "path" generally is not a linear path but is instead a tree in which branches diverge when different outcomes of an attack step put

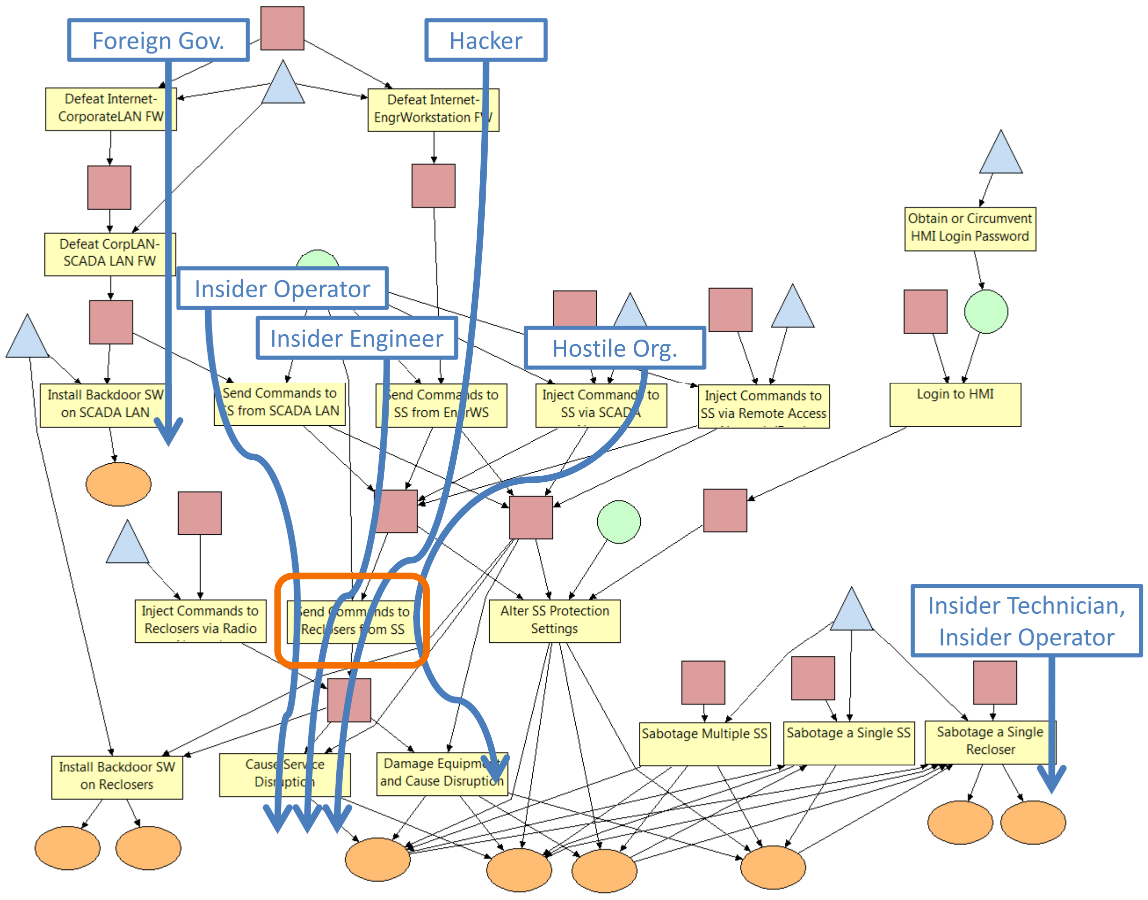

Foreign Gov.
Hacker
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Obtain or Circumvent HMI Login Password
Defeat CorpLAN-SCADA LAN FW
Insider Operator
Insider Engineer
Hostile Org.
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Insider Technician, Insider Operator
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption

Figure 5.15: Adversaries' preferred attack paths (with recloser radios).

the model into different model states. Then the adversary must pursue the most attractive path forward given whatever the new model state is. For the attack execution graph we are studying here, most of the Failure outcomes have no effect on the model state, so the preferred next step is to try the same step again until it succeeds. Thus, the preferred paths are mostly linear for this model. The exception is when an adversary pursues and reaches one attack goal and then pursues and reaches a second attack goal. Recall that an attacker may have more than one attack goal, and an attacker will continue attempting attack steps as long as the attack steps are more attractive than the do-nothing attack step.

We now examine the attack results for each adversary individually.

In both system configurations, the Foreign Government achieves the attack goal of installing backdoor software on the SCADA LAN. The preferred attack path is from the Internet to the corporate LAN to the SCADA LAN. The Foreign Government chooses the same attack approach with and without recloser radios, and the speed of the attack is the same (about five hours on average) (see Table 5.7). The Foreign Government demonstrates effective attack execution in that the number of attempts per attack step is approximately 1.1 for each of the three attack steps that the Foreign Government executes (see Table 5.8).

Without recloser radios in the electric power distribution system, the Hacker's preferred path is the same as the Foreign Government's. The Hacker achieves the goal of installing backdoor software on the SCADA LAN with the same speed and the same number of attempts per attack step as the Foreign Government. The reason is that the Hacker and the Foreign Government are attempting the same attack path and have the same skill levels relevant to achieving the SCADA LAN backdoor software goal.

However, the Hacker has a different preferred attack path when there are recloser radios in the system. Then the Hacker's preferred path is from the Internet to the engineering workstation to the substation communication gateway (see Table 5.9), from which the Hacker can send commands from the substation to the reclosers via the recloser radios. The Hacker's attack goal is to disrupt system-wide service, which is accomplished in a little under four hours on average (see Table 5.10).

We can contrast the Hacker with the Foreign Government and ask why the Foreign Government does not switch its preferred attack path like the Hacker does when recloser radios are introduced into the system. One main difference between these two adversaries is their different attack goal priorities. The Hacker is interested in installing backdoor software and disrupting service, but the Foreign Government is interested only in installing backdoor software. The recloser radios make it easier for adversaries to disrupt service, but they do not

Table 5.7: Average Time (in Hours) to Achieve Each Attack Goal (without Recloser Radios)

Attack Goal	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
Local Service Disruption	-	-	-	-	-	-
Minor Service Disruption	-	-	-	-	0.10	0.10
System-wide Service Disruption	-	-	14.92	2.03	-	-
Backdoor SW - SCADA LAN	5.17	5.17	-	-	-	-
Backdoor SW - Local Reclosers	-	-	-	-	-	-
Backdoor SW - System-wide Reclosers	-	-	-	-	-	-
Local Equipment Damage	-	-	-	0.92	-	-
Minor Equipment Damage	-	-	-	-	0.10	0.10
System-wide Equipment Damage	-	-	14.92	2.03	-	-

Table 5.8: Number of Attempts per Attack Step in the Time Interval [0, 50 Hours] (without Recloser Radios)

Attack Step	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
DefeatInternetCorporateLANFW	1.09	1.09	0	0	0	0
DefeatCorpLANSCADALANFW	1.08	1.08	0	0	0	0
InstallBackdoorSWonSCADALAN	1.07	1.07	0	0	0	0
DefeatInternetEngrWorkstationFW	0	0	0	0	0	0
SendCommandstoSSfromSCADALAN	0	0	0	0	0	0
SendCommandstoSSfromEngrWS	0	0	0	1.11	0	0
InjectCommandstoSSviaSCADANetwork	0	0	1.40	0	0	0
InjectCommandstoSSviaRemoteAccess	0	0	0	0	0	0
ObtainorCircumventHMILoginPassword	0	0	0	0	0	0
LogintoHMI	0	0	0	1.02	0	0
InjectCommandstoReclosersViaRadio	0	0	0	0	0	0
SendCommandstoReclosersfromSS	0	0	0	0	0	0
AlterSSProtectionSettings	0	0	1.10	2.21	0	0
InstallBackdoorSWonReclosers	0	0	0	0	0	0
CauseServiceDisruption	0	0	0	0	0	0
DamageEquipmentandCauseDisruption	0	0	0	0	0	0
SabotageMultipleSS	0	0	0	0	0	0
SabotageaSingleSS	0	0	0	0	0	0
SabotageaSingleRecloser	0	0	0	0	1.22	1.22

Table 5.9: Number of Attempts per Attack Step in the Time Interval [0, 50 Hours] (with Recloser Radios)

Attack Step	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
DefeatInternetCorporateLANFW	1.09	0	0	0	0	0
DefeatCorpLANSCADALANFW	1.08	0	0	0	0	0
InstallBackdoorSWonSCADALAN	1.07	0	0	0	0	0
DefeatInternetEngrWorkstationFW	0	1.13	0	0	0	0
SendCommandstoSSfromSCADALAN	0	0	0	0	1.10	0
SendCommandstoSSfromEngrWS	0	1.09	0	1.08	0	0
InjectCommandstoSSviaSCADANetwork	0	0	1.52	0	0	0
InjectCommandstoSSviaRemoteAccess	0	0	0	0	0	0
ObtainorCircumventHMILoginPassword	0	0	0	0	0	0
LogintoHMI	0	0	0	0	0	0
InjectCommandstoReclosersViaRadio	0	0	0	0	0	0
SendCommandstoReclosersfromSS	0	1.14	1.11	1.09	1.12	0
AlterSSProtectionSettings	0	0	0	0	0	0
InstallBackdoorSWonReclosers	0	0	0	0	0	0
CauseServiceDisruption	0	1.07	0	1.12	1.10	0
DamageEquipmentandCauseDisruption	0	0	1.08	0	0	0
SabotageMultipleSS	0	0	0	0	0	0
SabotageaSingleSS	0	0	0	0	0	0
SabotageaSingleRecloser	0	0	0	0	1.20	1.22

Table 5.10: Average Time (in Hours) to Achieve Each Attack Goal (with Recloser Radios)

Attack Goal	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
Local Service Disruption	-	-	-	-	-	-
Minor Service Disruption	-	-	-	-	0.45	0.10
System-wide Service Disruption	-	3.42	16.02	0.94	1.08	-
Backdoor SW - SCADA LAN	5.17	-	-	-	-	-
Backdoor SW - Local Reclosers	-	-	-	-	-	-
Backdoor SW - System-wide Reclosers	-	-	-	-	-	-
Local Equipment Damage	-	-	-	-	-	-
Minor Equipment Damage	-	-	-	-	0.45	0.10
System-wide Equipment Damage	-	-	16.02	-	-	-

really affect the path to the attack goal that the Foreign Government cares about the most.

In both system configurations, the Hostile Organization begins the attack by injecting commands to the substation via the SCADA communication network (see Tables 5.8 and 5.9). In the system configuration without recloser radios, the Hostile Organization chooses to alter the protection settings on substation equipment to cause system-wide equipment damage and service outages. In the system configuration with recloser radios, the Hostile Organization enters the network the same way but then diverts from the original preferred path and sends commands from multiple substations to multiple reclosers. The Hostile Organization then chooses to cause system-wide equipment damage and service outages. In both system configurations, the Hostile Organization pursues and achieves the same attack goals even though a different attack path is used.

The Hostile Organization takes a longer time (approximately 15 or 16 hours total on average) to achieve its attack goals than the Hacker or the Foreign Government. The longer time is due to the longer execution time for injecting commands on the SCADA network, which requires physical presence near the target. Also, injecting commands on the SCADA network is a tricky attack step to execute correctly, as evidenced by the relatively high number of attempts at that attack step (see Tables 5.8 and 5.9).

Without recloser radios in the electric power distribution system, the preferred path of the Insider Engineer is a two-pronged attack (see Table 5.7). First, the Insider Engineer logs in to a substation HMI and alters the substation protections in such a way as to cause local equipment damage and service disruption. This is done first because it is the most attractive attack when considering detection, cost, and payoff along with the Insider Engineer's attack preference weights. Then, the Insider Engineer sends commands from the engineering workstation to multiple substations. Via this second attack path, the Insider Engineer causes system-wide equipment damage and service disruption. This attack was performed second because it was more attractive than doing nothing, even if it was not as attractive as the attack path to the first attack goal. Due to the unique nature of the Insider Engineer's attack, the Insider Engineer performs the attack step to alter the substation protection settings at least twice during every attack. However, because the two prongs of the attack originate in different places, the effects of the attack step are different (different attack goals are achieved). This illustrates some of the subtleties of the ADVISE attack decision function as well as the versatility allowed in defining ADVISE attack steps.

When recloser radios are introduced, the Insider Engineer decides that an attack path using the recloser radios is more attractive than the HMI attack vector. The new attack

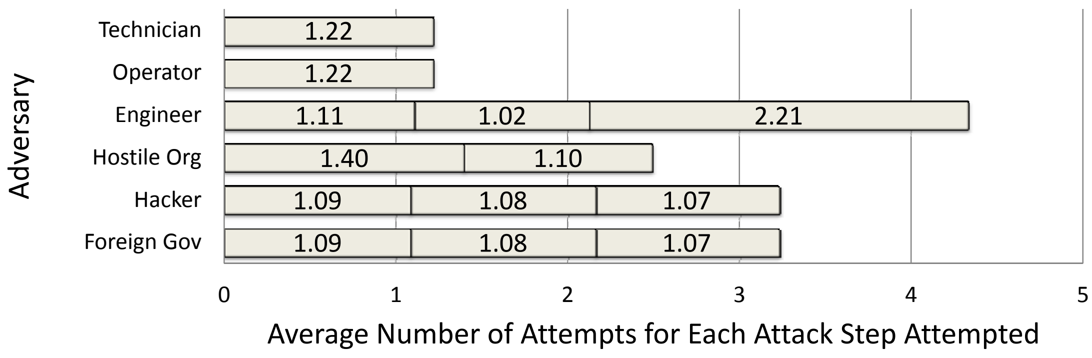


Figure 5.16: In the electric power distribution system *without* recloser radios, adversaries attempt one or more attack steps to achieve their goals. This stacked bar graph shows the average number of attempts per attack step. This graph shows the data from Table 5.8.

path allows the Insider Engineer to send commands from the engineering workstation to the substation and then to the reclosers. After gaining control over system-wide reclosers, the Insider Engineer causes system-wide service disruptions. With this new attack path, the Insider Engineer is able to cause system-wide effects more quickly.

In the system configuration without recloser radios, both the Insider Operator and the Insider Technician prefer to sabotage a single recloser. Among all the attack steps available to them, this attack step best suits their attack preferences. It is a very quick, one-step attack path.

When recloser radios are added to the system, the Insider Technician keeps the same preferred attack path. However, the Insider Operator now also executes another attack against the system by using SCADA LAN access to send commands from the SCADA LAN to the substation and then to the reclosers. After gaining control over system-wide reclosers, the Insider Operator causes system-wide service disruptions.

We put the attack data from Tables 5.8 and 5.9 in graph format to compare the attack strategies of different adversaries. Figures 5.16 and 5.17 illustrate how the adversaries differ in the number of attack step attempts they need in order to achieve their attack goals. The technician uses a single attack step to achieve the attack goal, but that step must be attempted 1.22 times on average to achieve the attack goal successfully. In contrast, the foreign government uses a three-step attack but needs only 1.08 attempts on average to achieve an attack step. An adversary with a high probability of success when attempting an attack step will need to attempt the attack fewer times on average to successfully complete it.

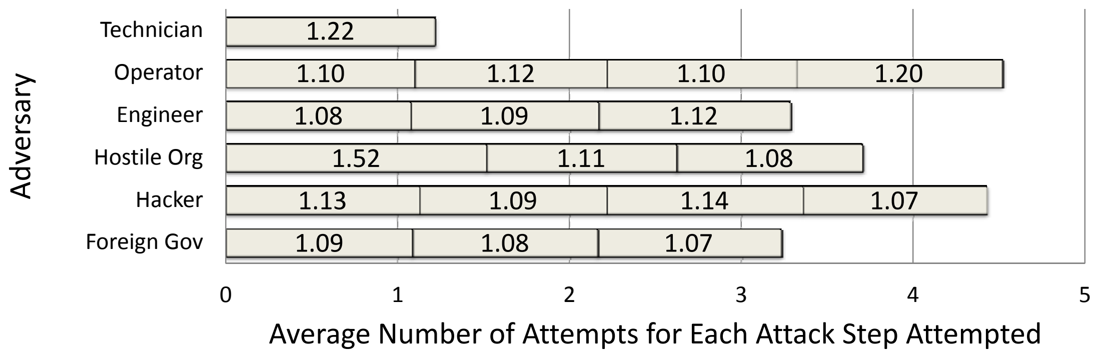


Figure 5.17: In the electric power distribution system *with* recloser radios, adversaries attempt one or more attack steps to achieve their goals. This stacked bar graph shows the average number of attempts per attack step. This graph shows the data from Table 5.9.

We also put the attack data from Tables 5.7 and 5.10 in graph format. Figures 5.18 and 5.19 show the average times for an adversary to achieve an attack goal without and with recloser radios, respectively. The attack speed depends heavily on which attack goal is achieved. An attack goal takes longer for the adversary to achieve if the attack path contains more attack steps, more repeated attempts of attack steps (caused by a lower probability of success), and/or slower attack steps (as defined by the attack step time distributions). When recloser radios are added to the system, some adversaries pursue different attack goals, and that shift in goals affects the average time needed to achieve their attack goals. Other adversaries pursue the same attack goals but use different attack paths, and that change affects their attack speeds.

By comparing the simulated attacks of the adversaries before and after the recloser radios are added to the system, we can see that the recloser radios provide an attractive attack vector whose existence convinces the Hacker, the Hostile Organization, the Insider Engineer, and the Insider Operator to deviate from their original attack path. From Figure 5.15 (showing the configuration containing recloser radios), we can identify the attack step to send commands from substations to reclosers as a commonly selected attack step (four of the six adversaries use it).

A logical reaction to this insight might be to modify the system to decrease the adversary's probability of success and thus lower the attractiveness of this attack step. However, we must then confront the question of how good the defense must be to sufficiently lower the attractiveness of this attack step.

To study this new defense strength question, we modify the parameters of our original

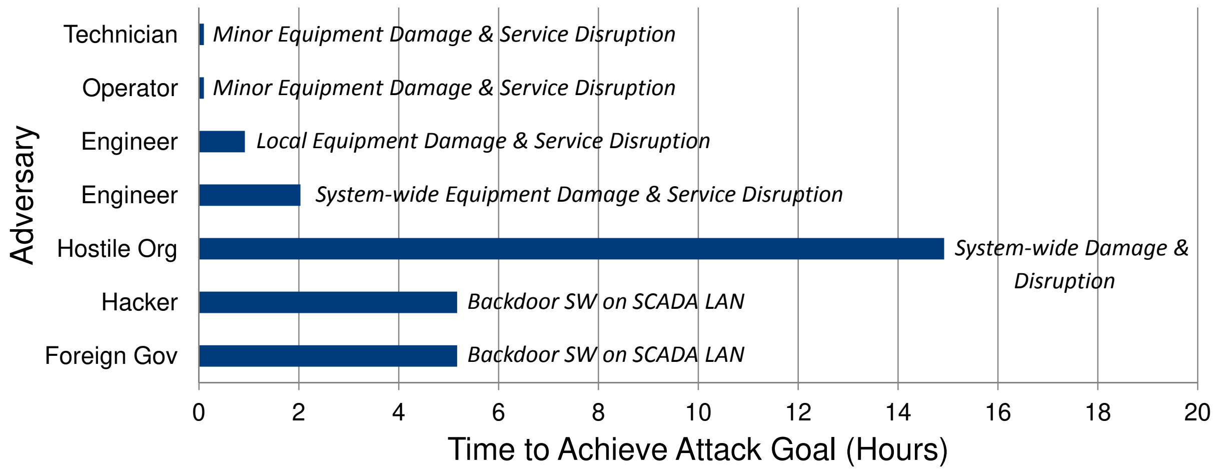


Figure 5.18: In the electric power distribution system *without* recloser radios, adversaries have different attack speeds. The attack goals are labeled. The engineer appears twice. This graph shows the data from Table 5.7.

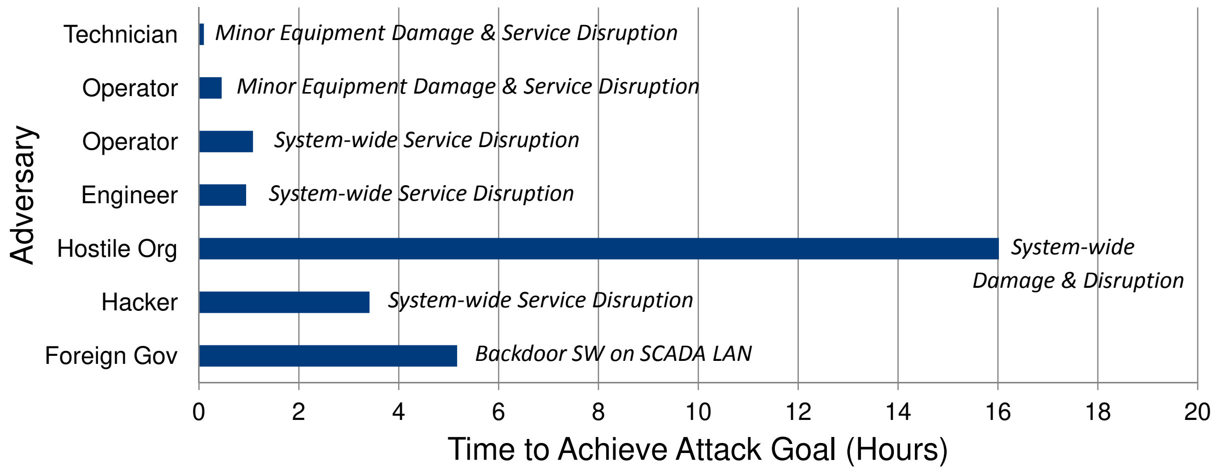


Figure 5.19: In the electric power distribution system *with* recloser radios, adversaries have different attack speeds. The attack goals are labeled. The insider operator appears twice. This graph shows the data from Table 5.10.

model and run additional simulations. We vary the probability of success for the *SendCommandstoReclosersfromSS* attack step from 0.2 to 0.9 in 0.1 increments. We examine the simulation results to determine the highest probability of success at which each adversary would stop using the *SendCommandstoReclosersfromSS* attack step as part of an attractive, preferred attack path.

For the Insider Engineer and the Insider Operator, that probability is 0.3, and for the Hacker and the Hostile Organization, that probability is 0.4. These are rough approximations, but they still provide some insight into the level of security required in order to measurably improve the security of the system.

5.2.6 Repair Model

ADVISE models can be linked with other types of discrete-event simulation models by sharing state variables. This capability enables us to build auxiliary models to model other aspects of the system not captured in an ADVISE model. For example, we can construct a repair model in which system compromises are detected and the system is restored to an uncompromised state after the repair period is complete. The repair model is implemented using a stochastic activity network (SAN) model [23].

In our example, the adversary decision explores state changes based only on the ADVISE model state changes and on the SAN model state changes connected with instantaneous activities that are fired when the ADVISE model changes state. (This is due to the current implementation of ADVISE in the Möbius tool.) As a result, the adversary may make different attack decisions with a repair model than without one.

Other implementation policies for composing ADVISE and SAN models are possible. One alternative is that the adversary attack decision is based solely on the ADVISE model state. In this case, the adversary decision is calculated as if the SAN repair model did not exist. A second alternative policy is that the adversary's attack decision is informed by not only the instantaneous state changes in the SAN repair model but also the timed (delayed) state changes in the SAN repair model.

Figure 5.20 shows a SAN model representing the repair of the minor equipment damage and minor service disruption caused when one recloser is sabotaged. The repair time is exponential with mean 6 hours because a repair technician must visit the location to repair the damaged equipment and restore electricity service. Note that this repair model does nothing to prevent the attack from reoccurring, so as soon as the model state becomes

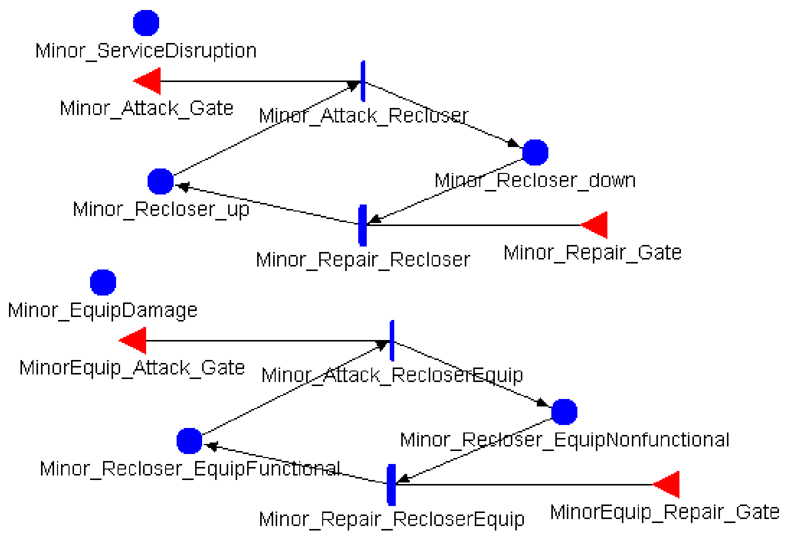


Figure 5.20: Repair model for minor equipment damage and minor service disruption.

uncompromised, the adversary will again start attacking.

We study the case of the Insider Technician, who decides to sabotage a single recloser. The Insider Technician causes minor equipment damage and minor service disruption in 0.1 hours on average. Without the repair model, the Insider Technician attempts the recloser sabotage attack step 1.22 times on average during the time interval [0, 50 hours].

After the repair model is added, the Insider Technician must reattack the recloser each time it is repaired. With the repair model, the Insider Technician attempts the recloser sabotage attack step 10.6 times on average during the time interval [0, 50 hours]. During that same time interval, the recloser was in a damaged state 90.8% of the time, compared with 99.8% of the time without the repair model. Although in this instance the repair model is overpowered by a quick-working adversary (the average time-to-compromise is 0.1 hours, and the average time-to-repair is 6 hours), this example still demonstrates how repair models can interact with ADVISE models.

5.2.7 Impact Model

In addition to the repair model, we can also link an ADVISE model to an impact model to help gauge the severity and the impact of system compromises on external entities. For example, Figure 5.21 shows a SAN model representing how service disruptions in the electric power distribution system are likely to affect Acme Business. (SAN models are formally

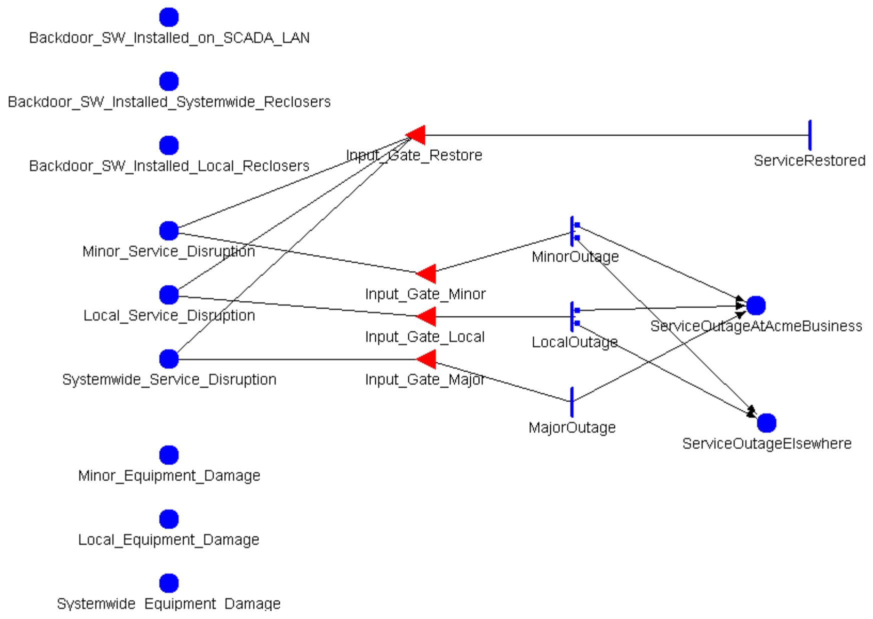


Figure 5.21: Impact model for service disruption.

defined in [24].) The impact model functions as follows: when there is a minor service disruption, Acme Business has a 1% chance of an electricity outage; when there is a local service disruption, Acme Business has a 10% chance of an electricity outage; and when there is a major service disruption, Acme Business has a 100% chance of an electricity outage.

Continuing our study of the Insider Technician with the repair model from the previous section, we add the impact model and run additional simulations. The simulation results report that Acme Business experiences an electricity outage for approximately 1% of the time interval [0, 50 hours].

An impact model enables us to analyze the severity of a compromise from the perspective of those who depend on or defend the system. For example, an impact model can be used to assess the degradation in the operation of the system due to adversary attacks. Impact models can also simulate the cascading effects of adversary attacks.

5.2.8 Using the Alternative Attractiveness Function

The case study results reported up to this point were generated with the original attractiveness function (Equation (3.7)). We now present results from the electric power distribution system case study using the alternative attractiveness function given in Section 3.4.

The analysis using the alternative attractiveness function uses the same system description described in Section 5.2.1, the same possible attacks against the system described in Section 5.2.2 (with one minor modification, noted below), and the same metrics described in Section 5.2.4. The analysis uses a set of six adversaries similar to those described in Section 5.2.3: a Foreign Government, a Hacker, a Hostile (Terrorist) Organization, an Insider Engineer, an Insider SCADA Operator, and an Insider Technician. The adversaries in the new analysis use different attack preference weights and utility functions.

Table 5.11 provides the new preference weights for the adversaries' attack decisions. Recall that the preference weights are used in conjunction with the utility functions to evaluate the relative attractiveness of attack steps during the adversary attack decision.

The new utility functions are linear so that the attack decisions produced using the alternative ADVISE decision rule are provably optimal (see Theorem 3.4). For this case study, the same utility conversion functions are used for all adversaries.

The cost utility function converts attack costs from the domain of $[0, 60000]$ cost units to the range of $[0, 1]$ utility units. Equation (5.4) gives the mathematical formula for the cost utility function.

$$U_C(c) = \frac{-c}{60000} + 1. \tag{5.4}$$

The payoff utility function converts payoff from the domain of $[0, 1000]$ payoff units to the range of $[0, 1]$ utility units. Equation (5.5) gives the mathematical formula for the payoff utility function.

$$U_P(p) = \frac{p}{1000}. \tag{5.5}$$

The log nondetection utility function converts log nondetection values from the domain of $[-18, 0]$ to the range of $[0, 1]$ utility units. Equation (5.6) gives the mathematical formula for the log nondetection utility function.

$$U_F(f) = \frac{-f}{6 \cdot \log 0.001} + 1. \tag{5.6}$$

To avoid having a logarithm of zero in the decision computation, we slightly modified the attack execution graph so that the probability of detection is never exactly one. Specifically, the attack step *AlterSSProtectionSettings* outcome with probability of detection 1.0 in the original attack execution graph was changed to have a probability of detection 0.999. That

Table 5.11: Adversary Decision Parameter Values for Alternative Attractiveness Function

Decision Parameters	***Engineer***	***Operator***	***Technician***	***Foreign Gov***	***Hostile Org***	***Hacker***
Cost Preference Weight	0.7	0.7	0.7	0	0.2	0.5
Detection Preference Weight	0.29	0.29	0.29	0.99	0.79	0.49
Payoff Preference Weight	0.01	0.01	0.01	0.01	0.01	0.01

was the only change we made to the attack execution graph for this analysis.

5.2.9 Results and Analysis

As in the original analysis, the ADVISE model was simulated using two system configurations. One configuration was an electric power distribution system with recloser radios that enabled communication between substations and reclosers. The second configuration was a system without recloser radios. For each configuration, we individually simulated each of the six adversaries attacking the system. All experiments were simulated for 100 runs, and the results were averaged across the 100 runs. The time values in the model can be interpreted in time units of hours. The cost units can be interpreted as dollars.

We first discuss the results generated using the alternative attractiveness function. Later we will compare these results with the results from the original model. We begin with a visualization of the expected attack path of each adversary for each of the two system configurations (with and without recloser radios).

Figures 5.22 and 5.23 show how the six adversaries choose to attack the system. The preferred attack paths are overlaid on the attack execution graph shown earlier in Figure 5.8. We constructed the expected attack path visualizations by noting which attack steps were attempted at least once in the time interval $[0, 50]$ hours. A bold line is shown through those steps to indicate the progression of the adversary's attack. Notice that when the addition of recloser radios opens up a new attack vector, that vector is attractive enough that the Foreign Government, the Hostile Organization, the Insider Engineer, and the Insider Operator change their preferred attack paths. However, the Insider Technician and the Hacker keep the same preferred paths.

Tables 5.12 and 5.13 (and the corresponding graphs in Figures 5.24 and 5.25) provide data on the average time for an adversary to achieve an attack goal without and with recloser radios, respectively. The attack speed depends heavily on which attack goal is achieved. An attack goal takes longer for the adversary to achieve if the attack path contains more attack steps, more repeated attempts of attack steps (caused by a lower probability of success), and/or slower attack steps (as defined by the attack step time distributions). When recloser radios are added to the system, some adversaries pursue different attack goals, and that shift in goals affects the average time needed to achieve their attack goals. Other adversaries pursue the same attack goals but use different attack paths, and that change affects their attack speeds.

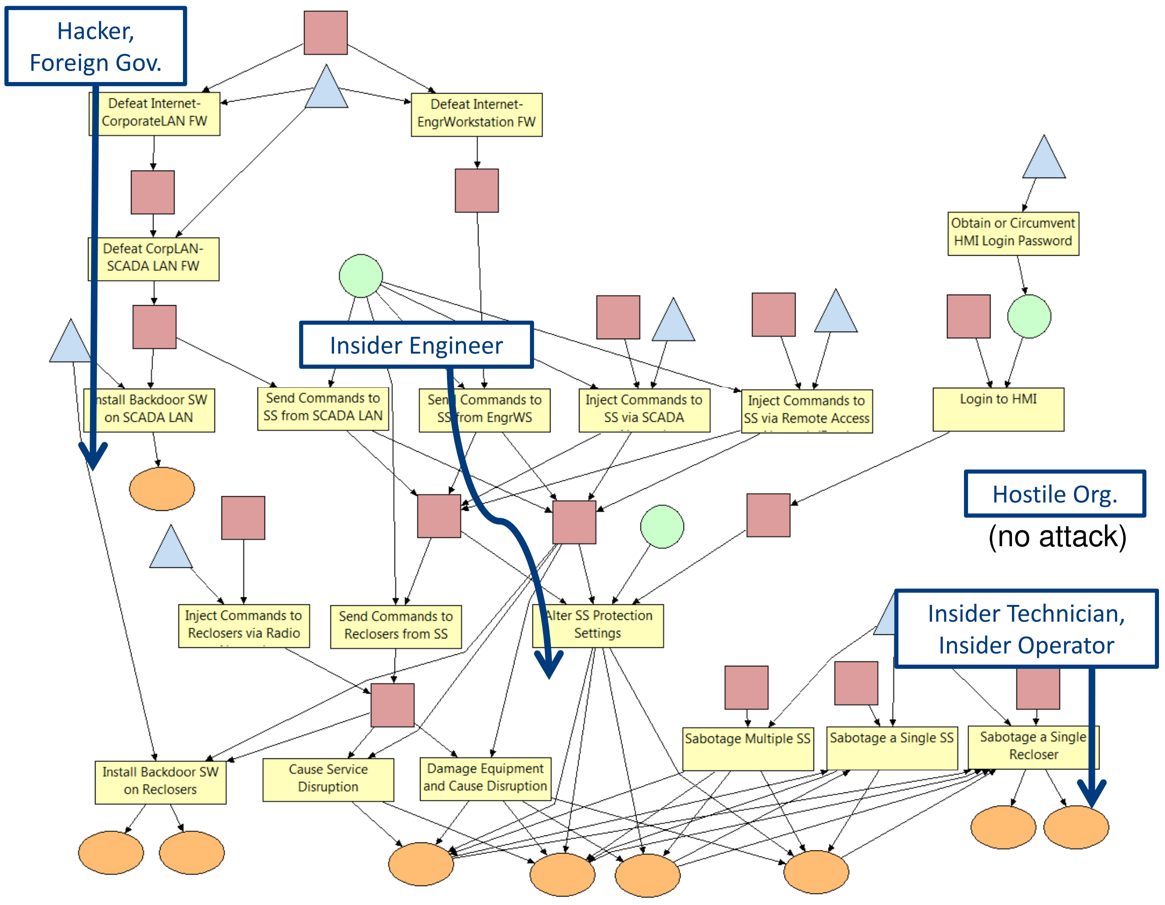

Hacker, Foreign Gov.
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Obtain or Circumvent HMI Login Password
Insider Engineer
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Hostile Org.
(no attack)
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Insider Technician, Insider Operator
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption

Figure 5.22: Adversaries' preferred attack paths (without recloser radios) using alternative attractiveness function.

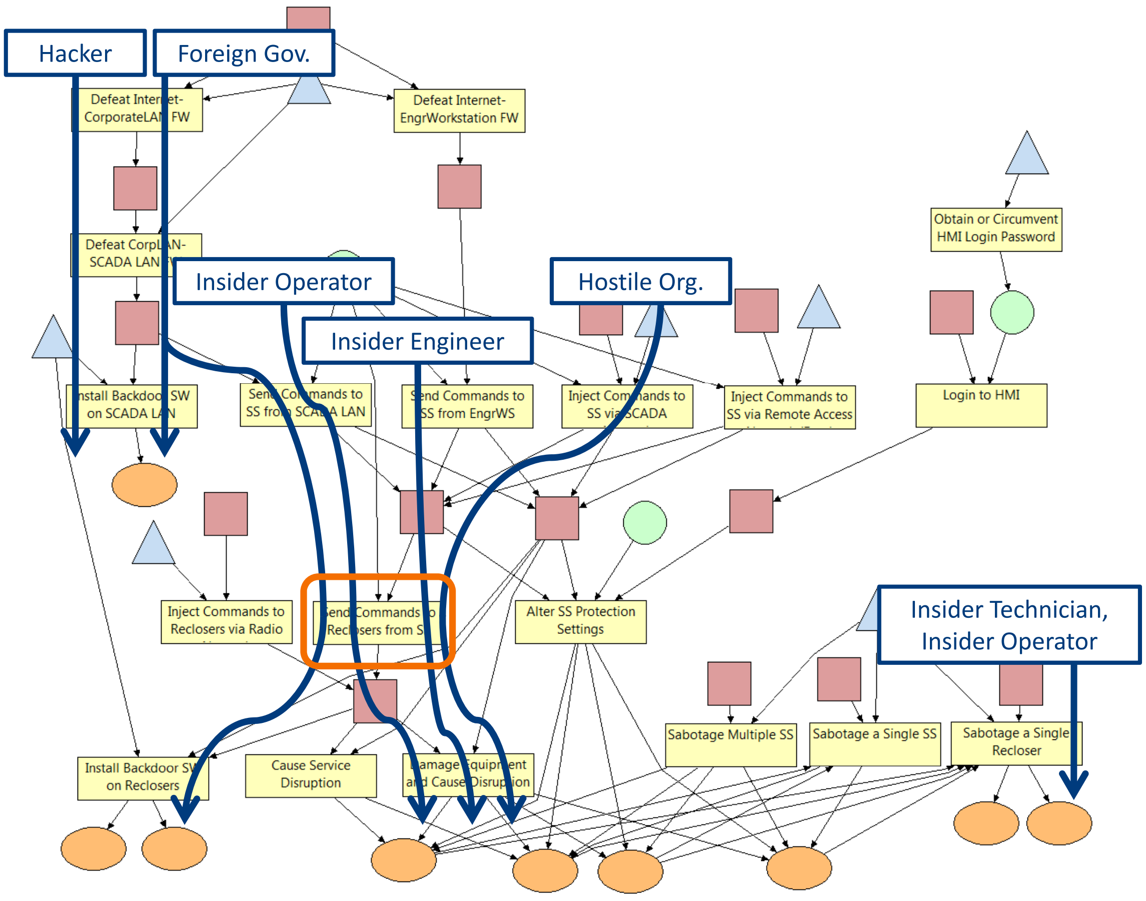

Hacker
Foreign Gov.
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Insider Operator
Hostile Org.
Insider Engineer
Obtain or Circumvent HMI Login Password
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Insider Technician, Insider Operator
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption

Figure 5.23: Adversaries' preferred attack paths (with recloser radios) using alternative attractiveness function.

Table 5.12: Average Time (in Hours) to Achieve Each Attack Goal (without Recloser Radios) Using Alternative Attractiveness Function

Attack Goal	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
Local Service Disruption	-	-	-	-	-	-
Minor Service Disruption	-	-	-	-	0.09	0.09
System-wide Service Disruption	-	-	-	1.11	-	-
Backdoor SW - SCADA LAN	5.26	5.26	-	-	-	-
Backdoor SW - Local Reclosers	-	-	-	-	-	-
Backdoor SW - System-wide Reclosers	-	-	-	-	-	-
Local Equipment Damage	-	-	-	-	-	-
Minor Equipment Damage	-	-	-	-	0.09	0.09
System-wide Equipment Damage	-	-	-	1.11	-	-

Table 5.13: Average Time (in Hours) to Achieve Each Attack Goal (with Recloser Radios) Using Alternative Attractiveness Function

Attack Goal	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
Local Service Disruption	-	-	-	-	-	-
Minor Service Disruption	-	-	-	-	0.10	0.09
System-wide Service Disruption	-	-	16.30	0.95	1.06	-
Backdoor SW - SCADA LAN	5.46	5.26	-	-	-	-
Backdoor SW - Local Reclosers	6.72	-	-	-	-	-
Backdoor SW - System-wide Reclosers	6.72	-	-	-	-	-
Local Equipment Damage	-	-	-	-	-	-
Minor Equipment Damage	-	-	-	-	0.10	0.09
System-wide Equipment Damage	-	-	16.30	0.95	1.06	-

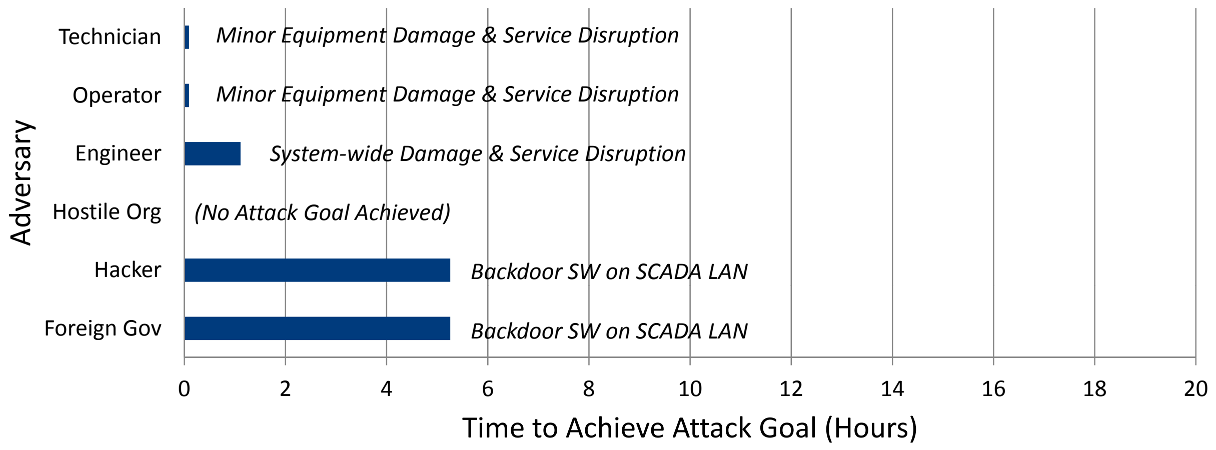


Figure 5.24: In the electric power distribution system *without* recloser radios, adversaries have different attack speeds. The attack goals are labeled. For these results, the adversary uses the alternative attractiveness function to make attack decisions. The hostile organization does not achieve any attack goals. This graph shows the data from Table 5.12.

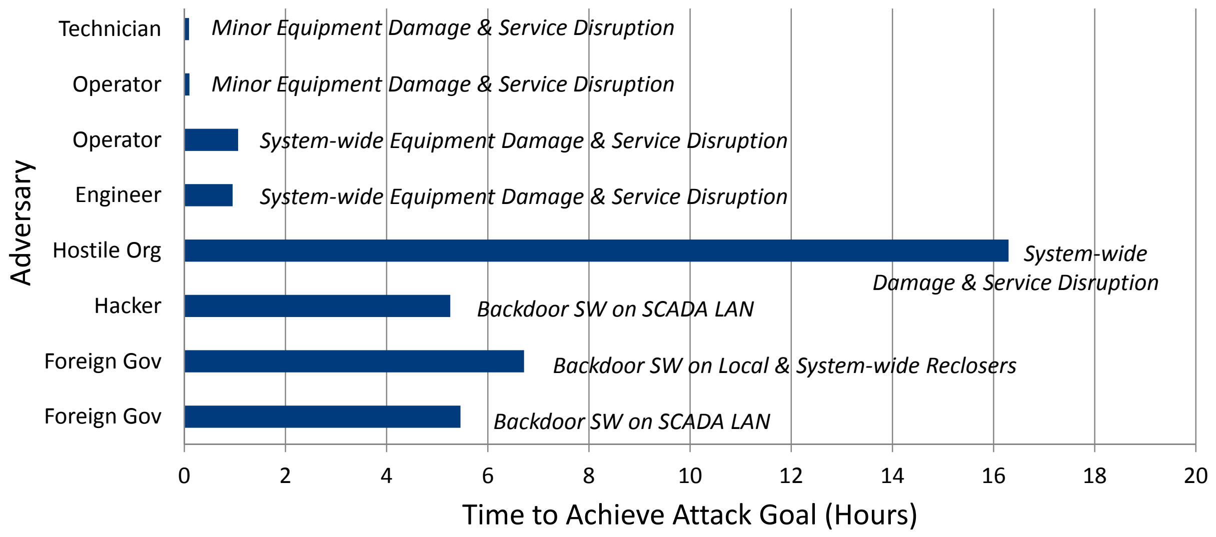


Figure 5.25: In the electric power distribution system *with* recloser radios, adversaries have different attack speeds. The attack goals are labeled. For these results, the adversary uses the alternative attractiveness function to make attack decisions. The insider operator and foreign government appear twice. This graph shows the data from Table 5.13.

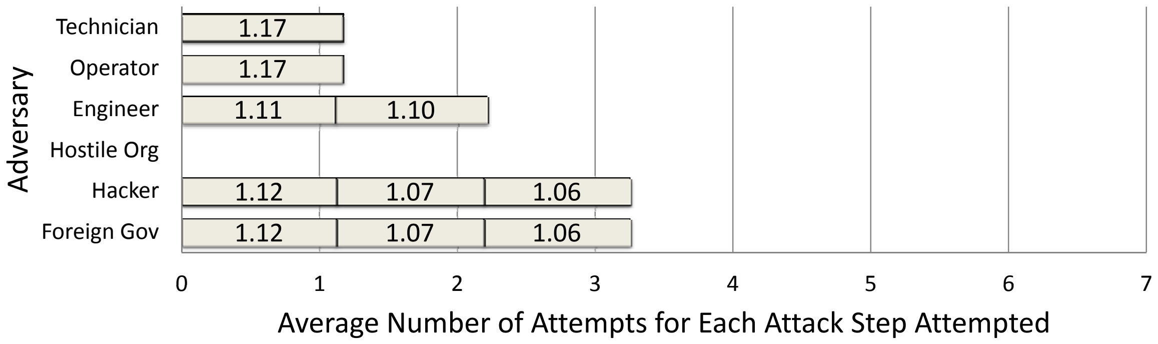


Figure 5.26: In the electric power distribution system *without* recloser radios, adversaries attempt one or more attack steps to achieve their goals. This stacked bar graph shows the average number of attempts per attack step when the adversary uses the alternative attractiveness function to make attack decisions. This graph shows the data from Table 5.14.

Tables 5.14 and 5.15 (and the corresponding graphs in Figures 5.26 and 5.27) provide data on how the adversaries differ in the number of attack step attempts they need to achieve their attack goals. An adversary with a high probability of success when attempting an attack step will need to attempt the attack fewer times on average to successfully complete it.

As expected, the results generated using the alternative attractiveness function are similar to the results generated using the original attractiveness function. For the system configuration without recloser radios, the Hacker, Foreign Government, Insider Operator, and Insider Technician achieve the same attack goals and follow the same attack paths as in the original results. The remaining two adversaries are less aggressive in their attacks. When using the alternative attractiveness function, the Insider Engineer follows a subset of the attack path reported in the original results and thus achieves only a subset of the attack goals achieved in the original results. The Hostile Organization chooses to attempt no attack at all when using the alternative attractiveness function.

For the system configuration that does include recloser radios, the Hostile Organization and the Technician using the alternative attractiveness function achieve the same attack goals and follow the same attack paths as in the original results. When using the alternative attractiveness function, the Insider Engineer and Insider Operator follow attack paths and achieve attack goals that are similar to those in the original results. In both sets of results, they choose to use the newly available recloser radio attack step. The Hacker pursues a

Table 5.14: Number of Attempts per Attack Step in the Time Interval [0, 50 Hours] (without Recloser Radios) Using Alternative Attractiveness Function

Attack Step	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
DefeatInternetCorporateLANFW	1.12	1.12	0	0	0	0
DefeatCorpLANSCADALANFW	1.07	1.07	0	0	0	0
InstallBackdoorSWonSCADALAN	1.06	1.06	0	0	0	0
DefeatInternetEngrWorkstationFW	0	0	0	0	0	0
SendCommandstoSSfromSCADALAN	0	0	0	0	0	0
SendCommandstoSSfromEngrWS	0	0	0	1.11	0	0
InjectCommandstoSSviaSCADANetwork	0	0	0	0	0	0
InjectCommandstoSSviaRemoteAccess	0	0	0	0	0	0
ObtainorCircumventHMILoginPassword	0	0	0	0	0	0
LogintoHMI	0	0	0	0	0	0
InjectCommandstoReclosersViaRadio	0	0	0	0	0	0
SendCommandstoReclosersfromSS	0	0	0	0	0	0
AlterSSProtectionSettings	0	0	0	1.10	0	0
InstallBackdoorSWonReclosers	0	0	0	0	0	0
CauseServiceDisruption	0	0	0	0	0	0
DamageEquipmentandCauseDisruption	0	0	0	0	0	0
SabotageMultipleSS	0	0	0	0	0	0
SabotageaSingleSS	0	0	0	0	0	0
SabotageaSingleRecloser	0	0	0	0	1.17	1.17

Table 5.15: Number of Attempts per Attack Step in the Time Interval [0, 50 Hours] (with Recloser Radios) Using Alternative Attractiveness Function

Attack Step	***Foreign Gov***	***Hacker***	***Hostile Org***	***Engineer***	***Operator***	***Technician***
DefeatInternetCorporateLANFW	1.12	1.12	0	0	0	0
DefeatCorpLANSCADALANFW	1.09	1.07	0	0	0	0
InstallBackdoorSWonSCADALAN	1.04	1.06	0	0	0	0
DefeatInternetEngrWorkstationFW	0	0	0	0	0	0
SendCommandstoSSfromSCADALAN	1.11	0	0	0	1.09	0
SendCommandstoSSfromEngrWS	0	0	0	1.05	0	0
InjectCommandstoSSviaSCADANetwork	0	0	1.52	0	0	0
InjectCommandstoSSviaRemoteAccess	0	0	0	0	0	0
ObtainorCircumventHMILoginPassword	0	0	0	0	0	0
LogintoHMI	0	0	0	0	0	0
InjectCommandstoReclosersViaRadio	0	0	0	0	0	0
SendCommandstoReclosersfromSS	1.16	0	1.13	1.11	1.07	0
AlterSSProtectionSettings	0	0	0	0	0	0
InstallBackdoorSWonReclosers	1.07	0	0	0	0	0
CauseServiceDisruption	0	0	0	0	0	0
DamageEquipmentandCauseDisruption	0	0	1.06	1.14	1.14	0
SabotageMultipleSS	0	0	0	0	0	0
SabotageaSingleSS	0	0	0	0	0	0
SabotageaSingleRecloser	0	0	0	0	1.26	1.17

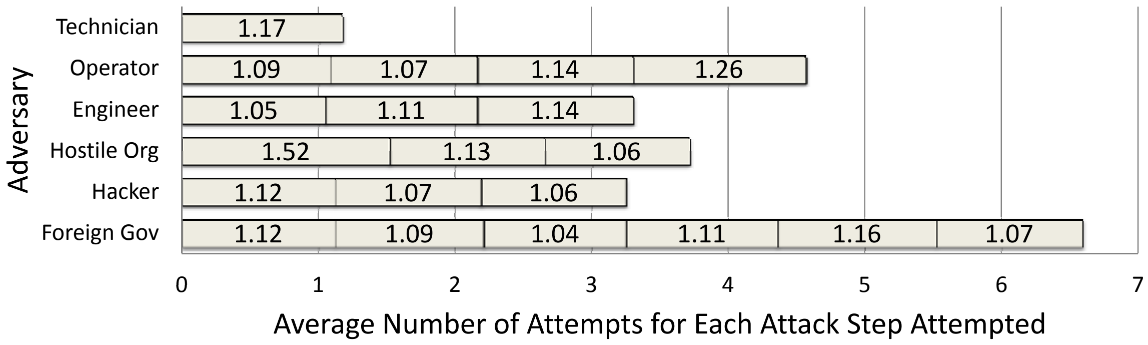


Figure 5.27: In the electric power distribution system *with* recloser radios, adversaries attempt one or more attack steps to achieve their goals. This stacked bar graph shows the average number of attempts per attack step when the adversary uses the alternative attractiveness function to make attack decisions. This graph shows the data from Table 5.15.

different attack goal when using the alternative attractiveness function and follows a less aggressive attack path. In contrast, the Foreign Government attacks more aggressively when using the alternative attractiveness function. In the new results, the Foreign Government installs backdoor software on the reclosers as well as on the SCADA LAN.

The original results helped us identify the attack step *SendCommandstoReclosersfromSS* as a commonly selected attack step. (Four of the six adversaries use it.) We note that the same insight can be gleaned from the new results. Using the alternative attractiveness function, four of the six adversaries use that same *SendCommandstoReclosersfromSS* attack step when it is available. These results indicate that the recloser radios provide an attractive attack vector.

In summary, the alternative attractiveness function with other minor modifications to the model generates results and insights similar to those from the original model. The proof of Theorem 3.4 shows that the alternative attractiveness function used with appropriate utility functions is provably optimal. The similarity of the new case study results to the original results suggests that the alternative attractiveness function may be able to produce results that are not only mathematically optimal but also reasonable.

5.3 Summary

The case studies in this chapter demonstrate that the ADVISE method is usable in realistic system security analysis situations. The studies illustrate how the ADVISE method can aid security analysts in producing a holistic quantitative evaluation of system-level security.

The ADVISE method enables analysts to gain valuable insight on system security. Results from ADVISE can help identify critical points in system security defenses and can aid system owners and administrators in deciding how to spend effort and money most effectively to measurably improve the system's defenses against adversaries.

One premise of the ADVISE analysis method is that the adversaries are rational decision-makers who weigh the cost of attempting the attacks and the risk of detection against the payoff value of the attack goals they hope to achieve. Given multiple possible attack vectors, they will choose paths that best meet their attack objectives and suit their attack preferences.

For hardened systems with high detection risk and high attack cost for the adversaries relative to the payoff, rational adversaries may decide that the system is not worth attacking. Systems containing attack targets with particularly low payoff values to adversaries may also not be worth attacking, even with relatively low detection and attack cost. This fact implies that low-attack-value targets may not need military-grade security, especially given practical budget constraints.

We argue that a system does not require perfect security (if such a thing exists); it just needs security that is good enough to defend its assets against the threats that it faces.

CHAPTER 6

RELATED WORK

ADVISE is a method to produce holistic quantitative security assessments to aid system-level security design decisions by evaluating alternatives. As stated in Chapter 1, ADVISE was motivated by aspects of both model-based security analysis (such as attack graphs and privileges graphs) and adversary-based analysis (such as MORDA and NRAT).

We compare ADVISE with the two closest alternative system security evaluation methods. Unlike ADVISE, these other security analysis methods use network scanners to gather input about the system. That means that a system must be operational before a security analysis can be conducted using those analysis methods. In contrast, ADVISE can be used for design decisions before the system is deployed or before network changes are implemented. The security of several configuration options can be analyzed before one is chosen for deployment.

6.1 Comparison with TVA and CAULDRON

Researchers at George Mason University developed the Topological Vulnerability Analysis (TVA) tool [25] to automatically generate attack graphs based on input from a network vulnerability scanner. The researchers developed a database of vulnerabilities that specifies a precondition and postcondition for each vulnerability. The system information from the scanner and the vulnerability information from the database are combined with information about the starting state and goal state of the attacker to create an attack graph. The analysis of the attack graph finds a minimum set of conditions such that the attacker can reach the attack goal.

To improve the TVA tool by adding visualization capabilities, the researchers at George Mason University developed a tool called Combinatorial Analysis Utilizing Logical Dependencies Residing on Networks (CAULDRON) [26], which is now available as a commercial tool.

CAULDRON is similar to ADVISE in that there is a forward and backward traversal of attack paths towards the attack goal. On the forward traversal, CAULDRON finds paths to the attack goals. On the backward traversal, CAULDRON prunes off irrelevant parts. This forward and backward traversal of possible attack paths is similar to the construction of a SLAT in the ADVISE method.

CAULDRON differs from ADVISE in that a CAULDRON analysis deals with only one attack goal per attack graph analysis. An ADVISE analysis can study how an adversary with multiple attack goals decides to attack a system. Also, CAULDRON assumes monotonicity in that an attacker never backtracks or loses ground during an attack. ADVISE does not impose that restriction. Allowing backtracking enables ADVISE to model the possibility that an attack step attempt may fail catastrophically, placing the adversary in a state very different from the state before the attempt or the state resulting from a successful attack attempt.

The most substantial difference between CAULDRON and ADVISE is that CAULDRON performs a reachability analysis (which attack paths are possible?), while ADVISE performs a likelihood analysis (which attack paths are likely?). Also, because ADVISE models include time, ADVISE can give insight on the speed of an attack.

6.2 Comparison with NetSPA and GARNET

Researchers at MIT Lincoln Laboratory developed a system called NETwork Security and Planning Architecture (NetSPA) [27] to generate attack graphs more efficiently than previous methods did. They achieve this efficiency by building multiple-prerequisite graphs. There are three types of nodes in a multiple-prerequisite graph. State nodes describe the attacker's access on a host. Edges point from state nodes to prerequisite nodes to indicate which prerequisites a state gives an attacker. Then, edges point from prerequisite nodes to vulnerability instance nodes to indicate which prerequisites are needed to exploit a vulnerability. The exploitation of a vulnerability instance provides the attacker with more states.

NetSPA receives input data from a network vulnerability scanner and vulnerability databases. NetSPA also computes reachability, determining how an attacker can maximally exploit a system's defenses. NetSPA assumes monotonicity in attack progress and does not analyze the time required to execute an attack.

To facilitate the analysis of attack graphs generated by NetSPA, an interactive visualization tool called Graphical Attack Graph and Reachability Network Evaluation Tool (GARNET) [28] was developed.

NetSPA does not include noncyber attacks, such as physical attacks and social engineering attacks. Social engineering is when attackers manipulate people in order to obtain sensitive information or unauthorized access. In contrast, ADVISE is a general formalism for representing attacks that can accommodate physical, social, and cyber attacks in a single attack execution graph. That flexibility enables ADVISE to examine attack paths that cross the boundaries between those three areas. NetSPA also differs from ADVISE in many of the same ways that CAULDRON does, including the attack monotonicity assumption and the lack of attack time analysis.

6.3 Benefits of ADVISE

While tools like CAULDRON and NetSPA perform detailed configuration analyses of deployed systems, ADVISE focuses on analyzing architectural-level vulnerabilities, in both hypothetical and implemented systems.

One benefit of ADVISE is that using it does not require the large overhead of developing databases of the preconditions and postconditions of thousands or hundreds of thousands of vulnerabilities. The ADVISE formalism does not dictate the level of abstraction used to represent the system and the attacks against the system. The analyst can describe attacks in terms of large or small granularity.

Another benefit of ADVISE is that it is not limited to cyber-based attacks. ADVISE can help analyze the security of any type of attack that can be represented as a sequence of attacks in which the attacker gains access and knowledge and achieves attack goals. This includes physical attacks and social engineering attacks.

ADVISE is an advancement over other system security analysis methods because ADVISE models how the characterization of different adversaries impacts their attack decisions. ADVISE enables analysts to consider how adversary attack preferences influence attack decisions. ADVISE also contains the notion of the "do-nothing" attack step, whereby the attractiveness of not executing any attack step is weighed against the attractiveness of executing the other attack steps.

The inclusion of attack time in an ADVISE simulation analysis is also significant in that it enables ADVISE to report metrics on the speed of an adversary's attack.

Not least among the benefits of ADVISE is that the exercise of modeling system security using ADVISE encourages system security analysts to think carefully about possible attacks. Previously unstated assumptions about system security become codified in the model so that the assumptions can be peer-reviewed by subject matter experts. The construction of an ADVISE model can bring together the expertise of many different subject matter experts (adversary characterization and behavior experts, attack experts, system architecture experts, and others). Including the expertise of all relevant subject matter experts promotes well-informed security decisions.

The adversary characterization ideas used in ADVISE were first explored in [15]. The early development of the ADVISE method is described in [1]. As the method has matured and been implemented as a tool, many parts have been modified. The most recent publication of the ADVISE method is [2], which supersedes [1].

6.4 Limitations of ADVISE

The ADVISE method is useful for synthesizing details about adversaries and system vulnerabilities into a quantitative holistic system security assessment. However, the method may not be ideal in all situations. The ADVISE method requires a large quantity of input data about the system and its adversaries. When some information is not known exactly, subject matter experts are often able to give a reasonable estimate that is sufficient for analysis purposes. However, when the appropriate subject matter experts either do not exist or are not available, then the ADVISE method is not recommended. When the input data set is incomplete, it may still be possible to make some assumptions and then use ADVISE to obtain results dependent on those assumptions that are beneficial to decision-makers.

The ADVISE method is also not intended for detailed configuration analysis, such as a firewall rule check. Other tools already exist that provide real-time network configuration analysis. ADVISE is intended to provide insight on the security aspects of hypothetical system architectures before they are built.

6.5 Assumptions of ADVISE

The ADVISE system security evaluation method is based on some assumptions about the nature of attacks, adversaries, and adversary attack decisions.

About attacks, we assume that attacks against systems can be described as sequences of individual attack steps. We assume that the security state of the system under attack can be adequately represented by a model state consisting of access, knowledge, and attack goals.

We also assume independence between the probabilities of different attack steps (or, at least, we assume that the dependence can be fully expressed through the model state).

About adversaries, we assume static adversary preferences. The analysis is over a short enough time period that desperation and other preference-changing factors do not affect the adversary. We also assume static adversary skill levels. We assume that the adversary has perfect knowledge of the system attack options (up to the planning horizon depth), including the probability of success.

About attack decisions, we assume that adversary attack decisions are based on attack preferences in three areas: cost, detection, and payoff. The choice of these particular attack preference areas was based on the three main things that drive attack decisions, according to subject matter experts: cost, probabilistic risk, and probabilistic payoff. We assume that the three criteria accurately capture the decision criteria of real adversaries. We assume that multiple-objective decision analysis (MODA) is applicable, so that we can consider decisions as being based on the linear combination of those three criteria. Finally, we assume that that planning horizon can be expressed as a number of attack steps, not an amount of attack execution time.

CHAPTER 7

CONCLUSIONS

Security was once considered a binary quantity (the system is either secure or not), but we now recognize that levels of security are more accurately regarded as a continuum. At one end of the continuum, a system is considered perfectly secure because it is impervious to compromise (an optimistic view, to be sure!). At the other end, a system is considered completely insecure because all systems are susceptible to compromise given a formidable enough adversary (a pessimistic, worst-case view). Real-world systems fall somewhere between these extremes, with some systems being arguably more secure than others. Security metrics enable analysts to quantify the differences in security between systems. Which system is more secure? Is it more secure by a lot or by a little?

Security metrics enable analysts to compare different system architectures on the basis of their security strength. These types of security metrics are inherently predictive metrics that ideally should provide insight on how adversaries are likely to attack the system. In this way, security metrics can help analysts identify and correct the weak points in a system's defense.

To produce model-based quantitative security metrics, we have formally defined and implemented the ADversary VIew Security Evaluation (ADVISE) method. In an ADVISE model, attack steps are precisely defined and organized into an attack execution graph, and an adversary profile captures a particular adversary's attack preferences and attack goals. We create executable security models that combine information from the attack execution graph, the adversary profile, and the desired security metrics to produce quantitative metrics data. The ADVISE model execution algorithms use the adversary profile and the attack execution graph to simulate how the adversary is likely to attack the system. The adversary selects the best next attack step by evaluating the attractiveness of several attack steps, considering cost, payoff, and the probability of detection. The attack step decision function

This chapter contains previously published material by E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke [2]. The material from [2] is reused by permission of IEEE and was published in Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011).

compares the attractiveness of different attack steps by incorporating the adversary's attack preferences and attack goals. The attack step decision function uses a state look-ahead tree to recursively compute how future attack decisions influence the attractiveness values of the current attack step options.

System architects can use ADVISE to compare the security strength of system architecture variants and analyze the threats posed by different adversaries. The practical application of ADVISE has been demonstrated in two case studies. One case study examined the security of two variants of a SCADA system architecture against attacks from five types of adversaries. The other case study analyzed the security of an electric power distribution system. These case studies demonstrated how the quantitative security metrics produced by ADVISE can aid system design and provide insight on system security.

7.1 Extensions and Applications

The ADVISE method can be extended and applied in many ways.

7.1.1 Increasing the Usability of the ADVISE Method and Tool

One limitation of the current ADVISE method is the need to gather a large volume of information to populate an ADVISE model. The development of libraries to store and efficiently reuse adversary profile data and attack step data would help increase the usability of the ADVISE method. When a security analyst is assessing the security of multiple similar systems facing the same adversaries, the adversary characterization data could be reused and modified as needed to fit the specific system. Analysts who construct ADVISE security models for many similar systems could also benefit from the development of libraries of attack step templates. The attack step templates would include default values for the input parameters.

The current ADVISE method and tool are suitable for use by security modeling experts. Further work is necessary to make the ADVISE method and tool widely accessible to and usable by a broader range of users. One option is to implement a simplified user interface for the ADVISE tool that divides the modeling task between domain experts and security experts. The domain experts would use their domain knowledge to construct simple system block diagrams to represent connections between components in the system of interest. Security experts would populate a database specifying the possible attack steps against each type of component. Then the block diagram would be automatically combined with the attack step data about the components to generate an attack execution graph.

7.1.2 Combining ADVISE Models with Other Types of Models

Another way to extend the ADVISE method is to combine ADVISE models with other types of models. This extension would make it possible to study the interactions between models. For example, the adversary-perspective ADVISE model could be combined with the user-perspective Multiple-Asymmetric-Utility System Model [29] to study how adversary actions and user actions can interact and affect system security. The Möbius abstract functional interface facilitates combining multiple smaller models into one larger model using shared state variables.

7.1.3 Alternative Analysis Techniques

This thesis focused on simulation-based analysis techniques for ADVISE models, but other nonsimulation analysis techniques are possible. One example is reachability analysis. However, attack graphs already do reachability analysis, so it is difficult to justify the time it would take to build a sophisticated attack execution graph solely for reachability analysis when an attack graph will do.

Another possible nonsimulation analysis technique is to report the preferred attack path (or preferred attack tree, if the preferred path diverges based on the outcome of an attack step attempt). This preferred attack path/tree is equivalent to a pruned SLAT whose root node is the initial model state. Since the existing ADVISE algorithms can already generate the preferred attack path/tree, the future work is in the design of a good visualization and numerical report of the preferred attack path/tree.

7.1.4 Exploring Applications Beyond Cyber Security

One final area of future work is the application of the ADVISE method beyond cyber security evaluation. The concepts represented in an attack execution graph (access domains, knowledge items, attack skills, and attack goals) are all present in physical security. For example, an analyst could construct attack execution graphs modeling possible attacks against border control or aviation security.

Whatever the application domain, the ADVISE method can be used to obtain a high-level, holistic, quantitative system security assessment by aggregating low-level security-relevant information.

7.2 Future Work

More work is needed to validate security metrics as accurate predictive measures of system security. Security analysts need to know if they are measuring the right things to gauge the security strength of a system. Metrics validation efforts would involve the collection and analysis of large quantities of security data, including both predictive metrics and incidence reports.

However, public release of security data is a sensitive issue for private companies. Companies may be more likely to allow a trusted third party to aggregate and analyze the security data than to release the data publicly. One possible trusted third party would be an insurance company offering computer security insurance. In exchange for releasing their security data to the insurance company (and paying an appropriate insurance premium), the companies would receive insurance against catastrophic security compromises. Actuarial scientists already have experience calculating car, home, and life insurance rates based on known risk factors. When security risk factors can be reliably measured, computer security insurance can become common, with lower premiums for systems with better security. Insurance companies have strong profit incentives to determine accurately which security practices actually lower the likelihood of compromise and to encourage the use of such practices.

Security metrics are important for rational business decisions involving system security. Decision-makers commonly use performance metrics, reliability metrics, and cost metrics to inform their decisions. Security metrics should become another standard part of well-informed decisions.

APPENDIX A

TOOL IMPLEMENTATION OF ADVISE

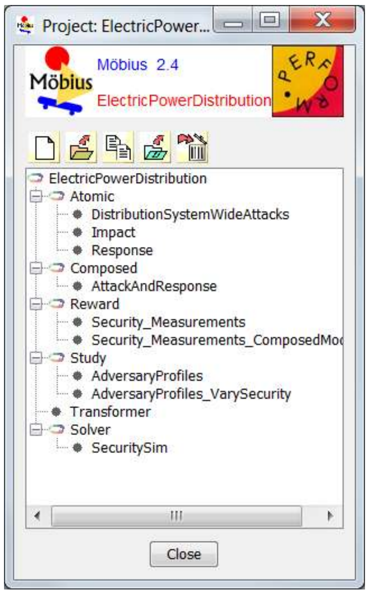
Möbius developer Ken Keefe led the implementation of the ADVISE model formalism as an atomic model inside the Möbius modeling framework and tool [18]. The enhanced Möbius tool now provides a graphical user interface to ease the collection of ADVISE input data and automates the generation and solution of ADVISE executable models.

A.1 ADVISE Model Inputs

The tool work flow consists of model creation, metrics specification, and study definitions, followed by model generation and execution. The project manager window (see Figure A.1) displays a list of all the parts of one project.

For model creation, the ADVISE atomic model editor was added to the Möbius tool. The tool's graphical user interface allows security analysts to create attack execution graphs on a canvas and then enter the attack step data needed for the ADVISE method (see Figure A.2). The palette of components (Access, Knowledge, Skill, Attack Step, Goal, Connection) is on the left-hand side. The user selects the component type and then adds new components on the canvas. The connection feature allows the user to connect access, knowledge, skill, and goal components to attack step components. When a component is selected, the details about the component appear on the right-hand portion of the screen. For access, knowledge, skill, and goal components, the user can modify the name of that component. For the attack step component, the user can modify the name and specify all the data required for an ADVISE attack step: attack cost, attack execution time, preconditions, and outcomes.

A separate tab in the same editor window enables the input of the adversary profile data (see Figure A.3). The profile includes the adversary's name, planning horizon, and attack preference weights for cost, payoff, and detection. The user must also provide input values for the adversary's skill levels, initial access, initial knowledge, and goal payoff valuation. The tool populates lists of all the skill, access, knowledge, and goal components present in



ElectricPowerDistribution
- Atomic
  - DistributionSystemWideAttacks
  - Impact
  - Response
- Composed
  - AttackAndResponse
- Reward
  - Security_Measurements
  - Security_Measurements_ComposedMo
- Study
  - AdversaryProfiles
  - AdversaryProfiles_VarySecurity
- Transformer
- Solver
  - SecuritySim

Figure A.1: Screenshot from the ADVISE tool (project manager).

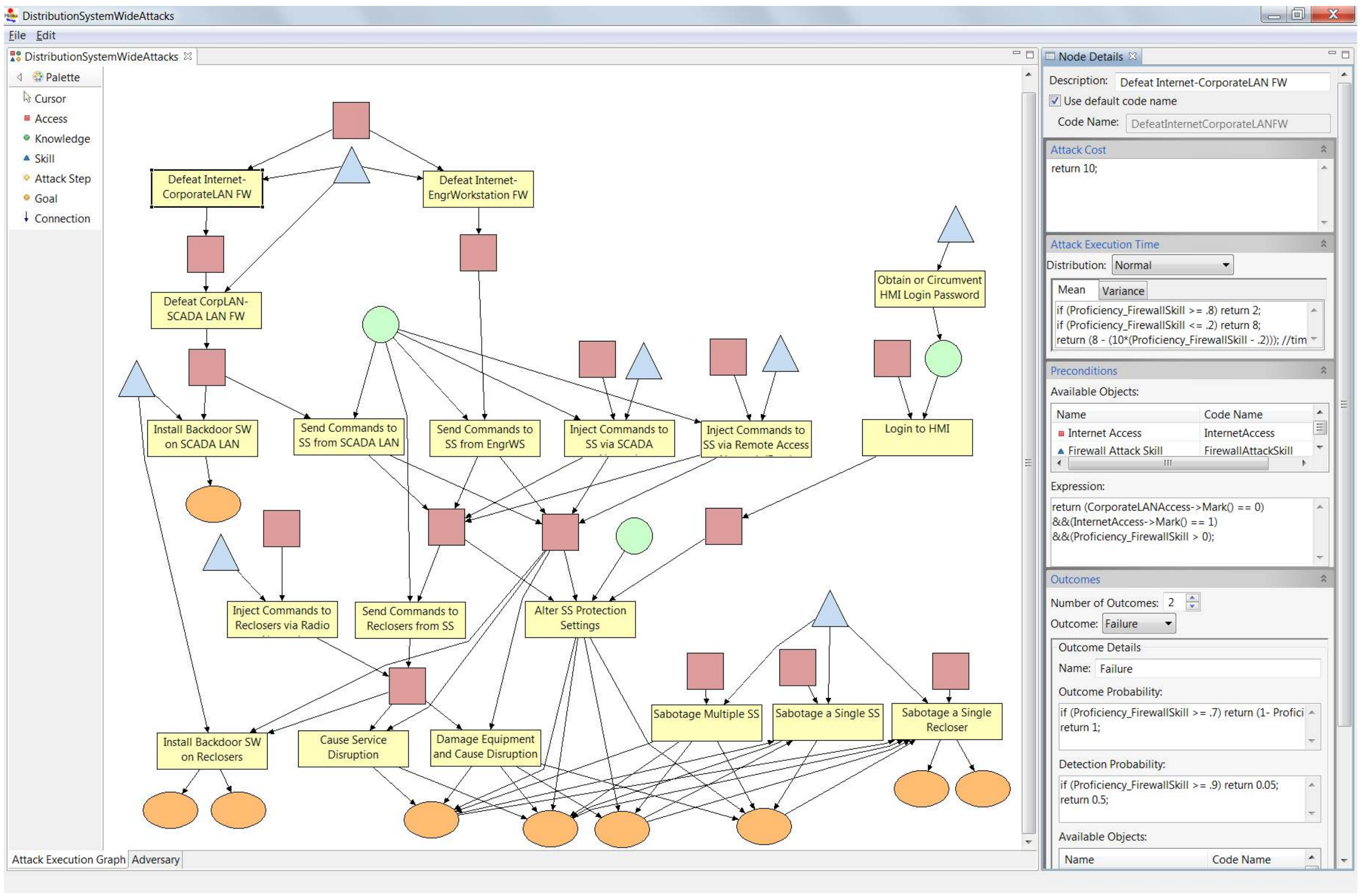
the attack execution graph. The user selects from those master lists to create the adversary profile.

For metrics specification, the Möbius tool contains a reward variable editor (see Figure A.4). Users can add metrics (called *reward variables* in Möbius) by giving a reward function and the times at which the metric is to be computed. For example, a metric specification could tell the Möbius simulation to record the probability that an adversary has achieved a particular attack goal at a particular time point.

For study definitions, global variables used elsewhere in the model are assigned values in the study editor (see Figure A.5). An experiment is defined by the values it assigns to the global variables. For manipulating large numbers of global variables and experiments, the tool provides a way to export and import the variable values from the tool in a spreadsheet format. The study editor was already part of the Möbius tool before the implementation of the ADVISE model formalism.

A.2 ADVISE Model Generation and Execution

The simulation editor (see Figure A.6) allows the user to set up the simulation parameters and then compile all the user-input data into executable simulation models. The pre-existing


DistributionSystemWideAttacks
File Edit
DistributionSystemWideAttacks
Palette
Cursor
Access
Knowledge
Skill
Attack Step
Goal
Connection
Defeat Internet-CorporateLAN FW
Defeat Internet-EngrWorkstation FW
Defeat CorpLAN-SCADA LAN FW
Obtain or Circumvent HMI Login Password
Install Backdoor SW on SCADA LAN
Send Commands to SS from SCADA LAN
Send Commands to SS from EngrWS
Inject Commands to SS via SCADA
Inject Commands to SS via Remote Access
Login to HMI
Inject Commands to Reclosers via Radio
Send Commands to Reclosers from SS
Alter SS Protection Settings
Sabotage Multiple SS
Sabotage a Single SS
Sabotage a Single Recloser
Install Backdoor SW on Reclosers
Cause Service Disruption
Damage Equipment and Cause Disruption
Attack Execution Graph
Adversary
Node Details

Description: Defeat Internet-CorporateLAN FW

☑ Use default code name

Code Name: DefeatInternetCorporateLANFW

Attack Cost

```
return 10;
```

Attack Execution Time

Distribution: Normal

Mean | Variance

```
if (Proficiency_FirewallSkill >= .8) return 2;
if (Proficiency_FirewallSkill <= .2) return 8;
return (8 - (10*(Proficiency_FirewallSkill - .2))); //tim
```

Preconditions

Available Objects:

Name	Code Name
🟥 Internet Access	InternetAccess
▲ Firewall Attack Skill	FirewallAttackSkill

Expression:

```
return (CorporateLANAccess->Mark() == 0)
&&(InternetAccess->Mark() == 1)
&&(Proficiency_FirewallSkill > 0);
```

Outcomes

Number of Outcomes: 2

Outcome: Failure

Outcome Details

Name: Failure

Outcome Probability:

```
if (Proficiency_FirewallSkill >= .7) return (1- Profici
return 1;
```

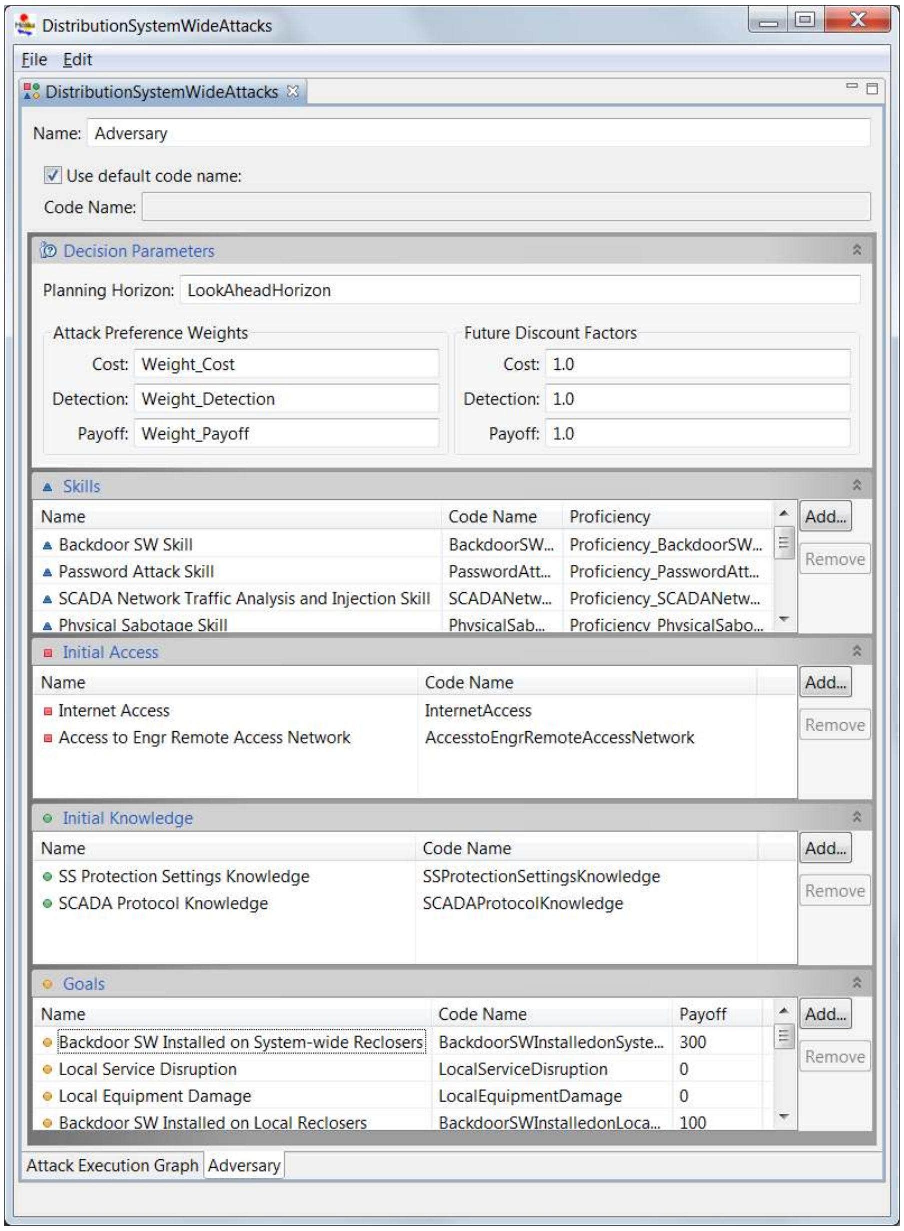
Detection Probability:

```
if (Proficiency_FirewallSkill >= .9) return 0.05;
return 0.5;
```

Available Objects:

Name	Code Name

Figure A.2: Screenshot from the ADVISE tool (attack execution graph editor).



DistributionSystemWideAttacks

File Edit

DistributionSystemWideAttacks

Name: Adversary

☑ Use default code name:

Code Name:

Decision Parameters

Planning Horizon: LookAheadHorizon

Attack Preference Weights		Future Discount Factors	
Cost:	Weight_Cost	Cost:	1.0
Detection:	Weight_Detection	Detection:	1.0
Payoff:	Weight_Payoff	Payoff:	1.0

Skills

Name	Code Name	Proficiency
Backdoor SW Skill	BackdoorSW...	Proficiency_BackdoorSW...
Password Attack Skill	PasswordAtt...	Proficiency_PasswordAtt...
SCADA Network Traffic Analysis and Injection Skill	SCADANetw...	Proficiency_SCADANetw...
Physical Sabotage Skill	PhysicalSab...	Proficiency_PhysicalSabo...

Initial Access

Name	Code Name
Internet Access	InternetAccess
Access to Engr Remote Access Network	AccesstoEngrRemoteAccessNetwork

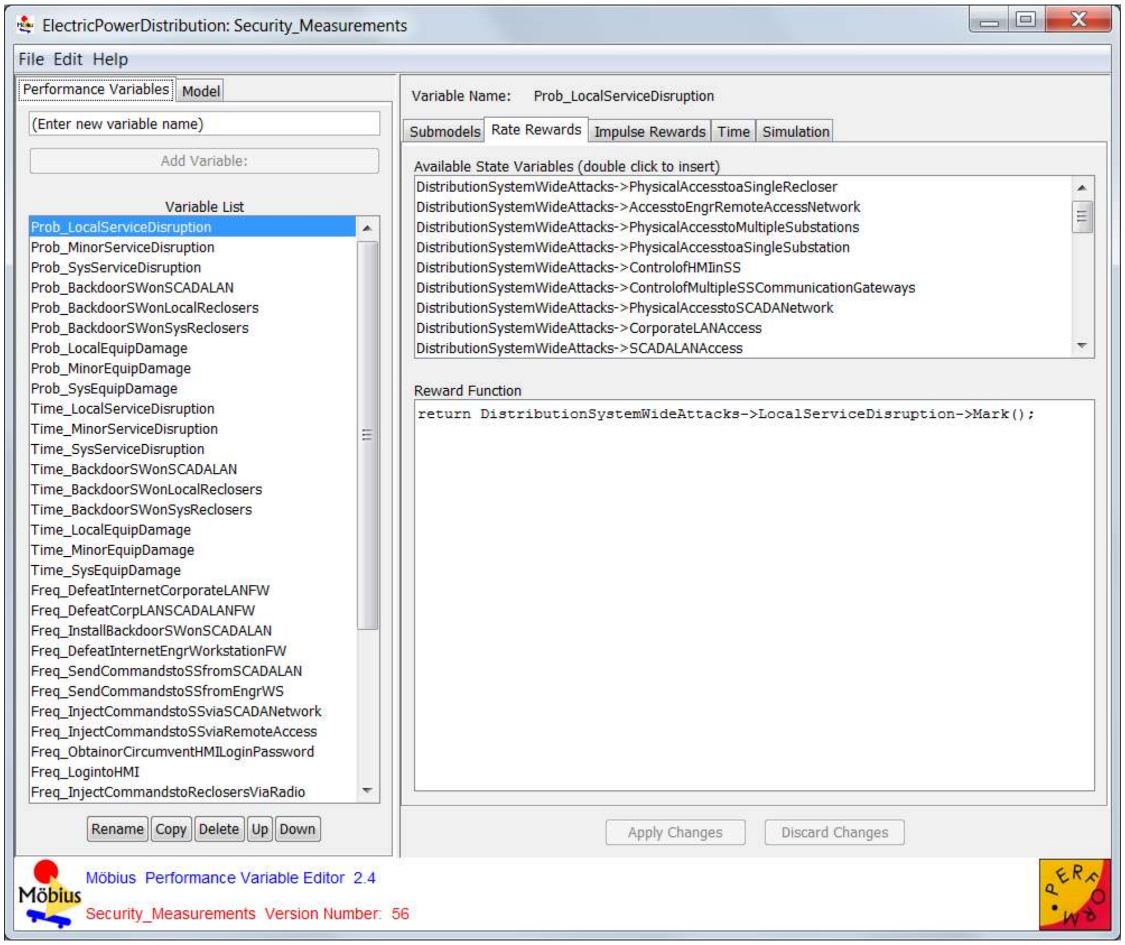
Initial Knowledge

Name	Code Name
SS Protection Settings Knowledge	SSProtectionSettingsKnowledge
SCADA Protocol Knowledge	SCADAProtocolKnowledge

Goals

Name	Code Name	Payoff
Backdoor SW Installed on System-wide Reclosers	BackdoorSWInstalledonSyste...	300
Local Service Disruption	LocalServiceDisruption	0
Local Equipment Damage	LocalEquipmentDamage	0
Backdoor SW Installed on Local Reclosers	BackdoorSWInstalledonLoca...	100

Figure A.3: Screenshot from the ADVISE tool (adversary profile editor).



Variable Name: Prob_LocalServiceDisruption

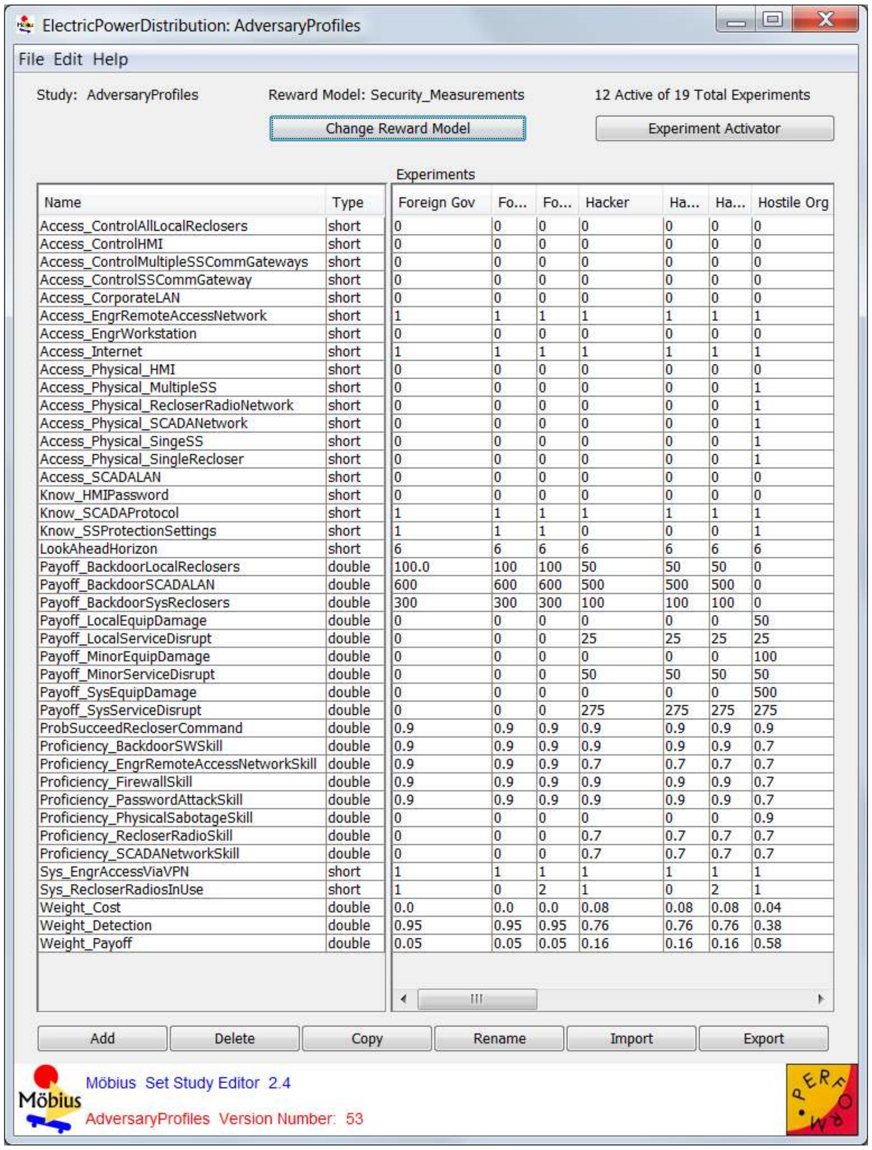
Variable List:
Prob_LocalServiceDisruption
Prob_MinorServiceDisruption
Prob_SysServiceDisruption
Prob_BackdoorSWonSCADALAN
Prob_BackdoorSWonLocalReclosers
Prob_BackdoorSWonSysReclosers
Prob_LocalEquipDamage
Prob_MinorEquipDamage
Prob_SysEquipDamage
Time_LocalServiceDisruption
Time_MinorServiceDisruption
Time_SysServiceDisruption
Time_BackdoorSWonSCADALAN
Time_BackdoorSWonLocalReclosers
Time_BackdoorSWonSysReclosers
Time_LocalEquipDamage
Time_MinorEquipDamage
Time_SysEquipDamage
Freq_DefeatInternetCorporateLANFW
Freq_DefeatCorpLANSCADALANFW
Freq_InstallBackdoorSWonSCADALAN
Freq_DefeatInternetEngrWorkstationFW
Freq_SendCommandstoSSfromSCADALAN
Freq_SendCommandstoSSfromEngrWS
Freq_InjectCommandstoSSviaSCADANetwork
Freq_InjectCommandstoSSviaRemoteAccess
Freq_ObtainorCircumventHMILoginPassword
Freq_LogintoHMI
Freq_InjectCommandstoReclosersViaRadio

Available State Variables (double click to insert)
DistributionSystemWideAttacks->PhysicalAccesstoaSingleRecloser
DistributionSystemWideAttacks->AccesstoEngrRemoteAccessNetwork
DistributionSystemWideAttacks->PhysicalAccesstoMultipleSubstations
DistributionSystemWideAttacks->PhysicalAccesstoaSingleSubstation
DistributionSystemWideAttacks->ControlofHMIinSS
DistributionSystemWideAttacks->ControlofMultipleSSCommunicationGateways
DistributionSystemWideAttacks->PhysicalAccesstoSCADANetwork
DistributionSystemWideAttacks->CorporateLANAccess
DistributionSystemWideAttacks->SCADALANAccess

Reward Function

```
return DistributionSystemWideAttacks->LocalServiceDisruption->Mark();
```

Figure A.4: Screenshot from the ADVISE tool (metrics specification editor).



Study: AdversaryProfiles

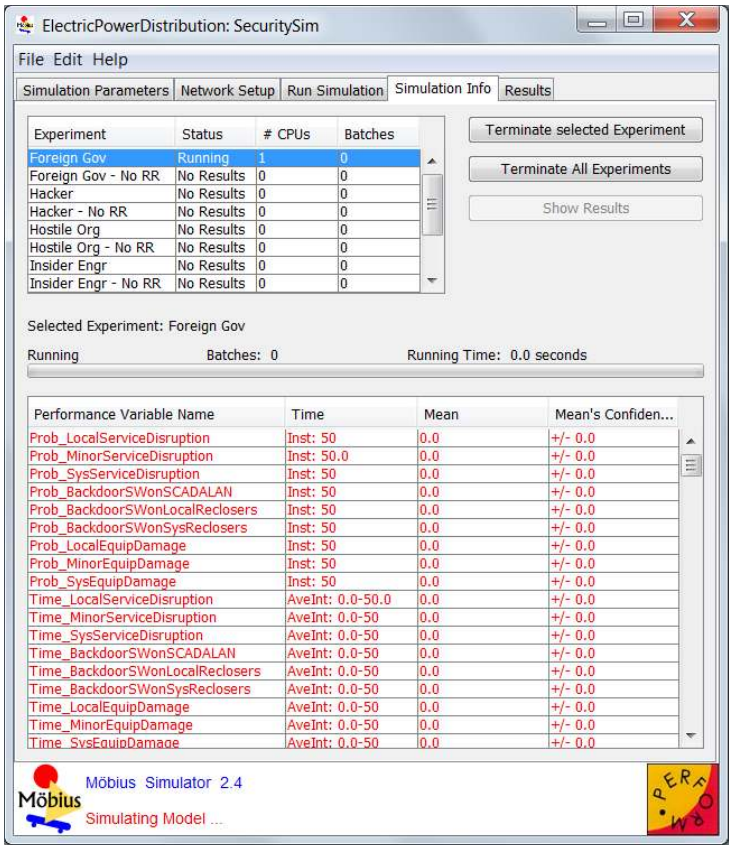
Reward Model: Security_Measurements

12 Active of 19 Total Experiments

AdversaryProfiles Version Number: 53

Name	Type	Foreign Gov	Fo...	Fo...	Hacker	Ha...	Ha...	Hostile Org
Access_ControlAllLocalReclosers	short	0	0	0	0	0	0	0
Access_ControlHMI	short	0	0	0	0	0	0	0
Access_ControlMultipleSSCommGateways	short	0	0	0	0	0	0	0
Access_ControlSSCommGateway	short	0	0	0	0	0	0	0
Access_CorporateLAN	short	0	0	0	0	0	0	0
Access_EngrRemoteAccessNetwork	short	1	1	1	1	1	1	1
Access_EngrWorkstation	short	0	0	0	0	0	0	0
Access_Internet	short	1	1	1	1	1	1	1
Access_Physical_HMI	short	0	0	0	0	0	0	0
Access_Physical_MultipleSS	short	0	0	0	0	0	0	1
Access_Physical_RecloserRadioNetwork	short	0	0	0	0	0	0	1
Access_Physical_SCADANetwork	short	0	0	0	0	0	0	1
Access_Physical_SingeSS	short	0	0	0	0	0	0	1
Access_Physical_SingleRecloser	short	0	0	0	0	0	0	1
Access_SCADALAN	short	0	0	0	0	0	0	0
Know_HMIPassword	short	0	0	0	0	0	0	0
Know_SCADAProtocol	short	1	1	1	1	1	1	1
Know_SSProtectionSettings	short	1	1	1	0	0	0	1
LookAheadHorizon	short	6	6	6	6	6	6	6
Payoff_BackdoorLocalReclosers	double	100.0	100	100	50	50	50	0
Payoff_BackdoorSCADALAN	double	600	600	600	500	500	500	0
Payoff_BackdoorSysReclosers	double	300	300	300	100	100	100	0
Payoff_LocalEquipDamage	double	0	0	0	0	0	0	50
Payoff_LocalServiceDisrupt	double	0	0	0	25	25	25	25
Payoff_MinorEquipDamage	double	0	0	0	0	0	0	100
Payoff_MinorServiceDisrupt	double	0	0	0	50	50	50	50
Payoff_SysEquipDamage	double	0	0	0	0	0	0	500
Payoff_SysServiceDisrupt	double	0	0	0	275	275	275	275
ProbSucceedRecloserCommand	double	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Proficiency_BackdoorSWSkill	double	0.9	0.9	0.9	0.9	0.9	0.9	0.7
Proficiency_EngrRemoteAccessNetworkSkill	double	0.9	0.9	0.9	0.7	0.7	0.7	0.7
Proficiency_FirewallSkill	double	0.9	0.9	0.9	0.9	0.9	0.9	0.7
Proficiency_PasswordAttackSkill	double	0.9	0.9	0.9	0.9	0.9	0.9	0.7
Proficiency_PhysicalSabotageSkill	double	0	0	0	0	0	0	0.9
Proficiency_RecloserRadioSkill	double	0	0	0	0.7	0.7	0.7	0.7
Proficiency_SCADANetworkSkill	double	0	0	0	0.7	0.7	0.7	0.7
Sys_EngrAccessViaVPN	short	1	1	1	1	1	1	1
Sys_RecloserRadiosInUse	short	1	0	2	1	0	2	1
Weight_Cost	double	0.0	0.0	0.0	0.08	0.08	0.08	0.04
Weight_Detection	double	0.95	0.95	0.95	0.76	0.76	0.76	0.38
Weight_Payoff	double	0.05	0.05	0.05	0.16	0.16	0.16	0.58

Figure A.5: Screenshot from the ADVISE tool (study editor).



Experiment	Status	# CPUs	Batches
Foreign Gov	Running	1	0
Foreign Gov - No RR	No Results	0	0
Hacker	No Results	0	0
Hacker - No RR	No Results	0	0
Hostile Org	No Results	0	0
Hostile Org - No RR	No Results	0	0
Insider Engr	No Results	0	0
Insider Engr - No RR	No Results	0	0

Selected Experiment: Foreign Gov

Running Batches: 0 Running Time: 0.0 seconds

Performance Variable Name	Time	Mean	Mean's Confiden...
Prob_LocalServiceDisruption	Inst: 50	0.0	+/- 0.0
Prob_MinorServiceDisruption	Inst: 50.0	0.0	+/- 0.0
Prob_SysServiceDisruption	Inst: 50	0.0	+/- 0.0
Prob_BackdoorSWonSCADALAN	Inst: 50	0.0	+/- 0.0
Prob_BackdoorSWonLocalReclosers	Inst: 50	0.0	+/- 0.0
Prob_BackdoorSWonSysReclosers	Inst: 50	0.0	+/- 0.0
Prob_LocalEquipDamage	Inst: 50	0.0	+/- 0.0
Prob_MinorEquipDamage	Inst: 50	0.0	+/- 0.0
Prob_SysEquipDamage	Inst: 50	0.0	+/- 0.0
Time_LocalServiceDisruption	AveInt: 0.0-50.0	0.0	+/- 0.0
Time_MinorServiceDisruption	AveInt: 0.0-50	0.0	+/- 0.0
Time_SysServiceDisruption	AveInt: 0.0-50	0.0	+/- 0.0
Time_BackdoorSWonSCADALAN	AveInt: 0.0-50	0.0	+/- 0.0
Time_BackdoorSWonLocalReclosers	AveInt: 0.0-50	0.0	+/- 0.0
Time_BackdoorSWonSysReclosers	AveInt: 0.0-50	0.0	+/- 0.0
Time_LocalEquipDamage	AveInt: 0.0-50	0.0	+/- 0.0
Time_MinorEquipDamage	AveInt: 0.0-50	0.0	+/- 0.0
Time_SysEquipDamage	AveInt: 0.0-50	0.0	+/- 0.0

Figure A.6: Screenshot from the ADVISE tool (simulation interface).

state-based model simulation and solution capabilities of the Möbius tool are used to execute the model and record the desired metrics data. The results are written to text and csv files.

The discrete-event simulation consists of states and state changes (also called *actions*). The model state (formally defined in Section 2.3) reflects the progress of the adversary in attacking the system. The state variables include the set of access domains that the adversary can access, the set of knowledge items that the adversary possesses, and the set of attack goals the adversary has achieved. The initial model state is derived from the adversary profile.

Actions cause the model state to change. In an ADVISE model, the state changes are due to adversary attack decisions and attack step outcomes. When the simulation begins, the model is put in the initial state. Then the adversary makes an attack decision and chooses the best next attack step to attempt. The simulation selects one outcome of that attack step stochastically using the outcome probability distributions defined in the attack execution graph. Based on the attack step outcome, the model state is updated. For example, if the attack step outcome was that the adversary had successfully logged on to an internal network, the state change might include adding that internal network access to the set of access domains that the adversary can access.

For each simulation run, the model begins in the initial state. Then the adversary makes an attack decision, an outcome results, and the state is updated. The process repeats, and the simulation continues until the end time or ending condition is reached. As the simulation executes, the metrics data requested by the user are collected and saved to a file. The simulator executes many simulation runs and averages the metrics data across all runs.

A.3 Conclusion

The ADVISE tool implementation using the Möbius modeling framework provides a practical way to use the ADVISE concepts and theory to build models and analyze systems.

The ADVISE tool implementation was used to complete the case studies in Chapter 5 and the execution time performance analysis in Section 3.5. During the summer of 2011, a select group of people from academia, government, and industry participated in an alpha trial of the ADVISE tool implementation. To aid new users of the tool, the ADVISE team developed an extensive tutorial and a users' manual.

REFERENCES

[1] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. H. Sanders, “Adversary-driven state-based system security evaluation,” in *Proceedings of the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010).* New York, NY: ACM, September 15, 2010, pp. 5:1–5:9.

[2] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke, “Model-based security metrics using ADversary VIew Security Evaluation (ADVISE),” in *Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011)*, September 5-8, 2011, pp. 191–200.

[3] A. Jaquith, *Security Metrics: Replacing Fear, Uncertainty, and Doubt.* Upper Saddle River, NJ: Addison-Wesley Professional, 2007.

[4] B. Schneier, *Secrets and Lies: Digital Security in a Networked World.* New York, NY: John Wiley & Sons, 2004.

[5] O. M. Sheyner, “Scenario graphs and attack graphs,” Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 2004.

[6] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, “Automated generation and analysis of attack graphs,” in *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P 2002).* Washington, D.C.: IEEE Computer Society, 2002, p. 273.

[7] L. Wang, A. Singhal, and S. Jajodia, “Toward measuring network security using attack graphs,” in *Proceedings of the 2007 ACM Workshop on Quality of Protection (QoP 2007).* New York, NY: ACM, 2007, pp. 49–54.

[8] M. Dacier and Y. Deswarte, “Privilege graph: An extension to the typed access matrix model,” in *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS 1994).* London, UK: Springer-Verlag, 1994, pp. 319–334.

[9] M. Dacier, Y. Deswarte, and M. Kaâniche, “Models and tools for quantitative assessment of operational security,” in *Information systems security: facing the information society of the 21st century.* London, UK: Chapman & Hall, Ltd., 1996, pp. 177–186.

[10] R. Ortalo, Y. Deswarte, and M. Kaâniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 633–650, 1999.

[11] S. Evans and J. Wallner, "Risk based security engineering through the eyes of the adversary," in *Proceedings of the 2005 IEEE Workshop on Information Assurance.* United States Military Academy, West Point, NY, June 2005, pp. 158–165.

[12] S. Evans, D. Heinbuch, E. Kyle, J. Piorkowski, and J. Wallner, "Risk-based systems security engineering: Stopping attacks with intention," *IEEE Security & Privacy*, vol. 2, no. 6, pp. 59–62, 2004.

[13] D. Buckshaw, G. Parnell, W. Unkenholz, D. Parks, J. Wallner, and O. S. Saydjari, "Mission oriented risk and design analysis of critical information systems," *Military Operations Research*, vol. 10, no. 2, pp. 19–38, 2005.

[14] B. Whiteman, "Network Risk Assessment Tool (NRAT)," *IAnewsletter*, vol. 11, no. 1, pp. 4–8, Spring 2008.

[15] E. Van Ruitenbeek, K. Keefe, W. H. Sanders, and C. Muehrcke, "Characterizing the behavior of cyber adversaries: The means, motive, and opportunity of cyberattacks," in *40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Supplemental (DSN 2010)*, June 28-July 1, 2010, pp. 17–18.

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York, NY: John Wiley & Sons, Inc., 1994.

[17] R. Bellman, *Dynamic Programming.* Princeton, NJ: Princeton University Press, 1957.

[18] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster, "The Möbius framework and its implementation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 956–969, Oct. 2002.

[19] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability, and performability," in *Dependable Computing and Fault-Tolerant Systems*, A. Avizienis, H. Kopetz, and J. Laprie, Eds. Vienna, Austria: Springer-Verlag, 1991, vol. 4, pp. 215–237.

[20] K. Stouffer, J. Falco, and K. Scarfone, "Guide to Industrial Control Systems (ICS) Security," National Institute of Standards and Technology Special Publication 800-82, Gaithersburg, MD, September 2008.

[21] H. J. A. Ferrer and E. O. Schweitzer, III, Eds., *Modern Solutions for Protection, Control, and Monitoring of Electric Power Systems.* Pullman, WA: Schweitzer Engineering Laboratories, Inc., 2010.

[22] B. Barnett and M. J. Dell'Anno, Personal Communications, April 2010–June 2011.

[23] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *Lectures on Formal Methods and Performance Analysis, Lecture Notes in Computer Science no. 2090*, E. Brinksma, H. Hermanns, and J. P. Katoen, Eds. Berlin, Germany: Springer, 2001, pp. 315–343.

[24] W. H. Sanders, "Construction and solution of performability models based on stochastic activity networks," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1988.

[25] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," in *Managing Cyber Threats: Issues, Approaches and Challenges*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. New York, NY: Springer, 2005, ch. 9.

[26] S. OHare, S. Noel, and K. Prole, "A graph-theoretic visualization approach to network risk analysis," in *Proceedings of the 5th International Workshop on Visualization for Computer Security (VizSec 2008)*, J. Goodall, G. Conti, and K.-L. Ma, Eds. Berlin, Germany: Springer-Verlag, September 2008, pp. 60–67.

[27] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proceedings of the 22nd Annual Computer Security Applications Conference.* Washington, D.C.: IEEE Computer Society, 2006, pp. 121–130.

[28] L. Williams, R. Lippmann, and K. Ingols, "Garnet: A graphical attack graph and reachability network evaluation tool," in *Proceedings of the 5th International Workshop on Visualization for Computer Security (VizSec 2008).* Berlin, Germany: Springer-Verlag, 2008, pp. 44–59.

[29] D. Eskins and W. H. Sanders, "The multiple-asymmetric-utility system model: A framework for modeling cyber-human systems," in *Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems (QEST 2011)*, September 5-8, 2011, pp. 233–242.