Computers & Security 164 (2026) 104834

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full Length Article

From attack trees to timed stochastic games: A novel intrusion response approach

Tommaso Caiazzi *, Stefano Iannucci, Valerio Marini, Matteo Foschi, Riccardo Torlone

Roma Tre University, Rome, Italy

ARTICLE INFO

Keywords:
Intrusion response
Cyber security
Autonomic computing
Self-protecting systems

ABSTRACT

Most dynamic Intrusion Response Systems (IRSs) use models to characterize the attack patterns and the dynamics of the protected system. They are typically based on some mathematical framework and require a low-level modeling activity that is often difficult and error-prone, even for the experienced end-user. Furthermore, most of the model-based approaches proposed so far do not structurally include the notion of time, which is necessary to model non-instantaneous defense and attack actions. In this paper, we introduce a novel methodology for the automatic generation of IRSs based on Timed Competitive Stochastic Games from augmented Attack-Defense Trees (ADT), a formalism that is commonly used to represent attack patterns and to build IRSs based on a static mapping between attack and response. We formally and empirically prove that: (i) using a static mapping between attack and response or selecting the action with the immediate minimum cost to counter the attack without long-term planning leads to an underestimation of the defense cost; (ii) the total defense cost of a defense policy obtained with an IRS based on the proposed methodology is lower than or equal to the defense cost that can be obtained with an IRS based on static mapping; (iii) not considering time leads to an underestimation of the defense cost. We then perform experiments showing the scalability of the proposed approach in terms of planning time and memory usage.

1. Introduction

Autonomic Computing has been introduced by Kephart et al., in Kephart and Chess (2003), with the identification of four essential types of tasks that must be satisfied for a system to be considered *autonomic*, namely: *self-configuration, self-optimization, self-healing*, and *self-protection*. Most of the work of the research community on autonomic systems has addressed the first three properties (e.g., de Lemos et al., 2013; Psaier and Dustdar, 2011; Weyns, 2021), while self-protection has historically received less attention. More specifically, run-time self-protection can be divided into two macro-steps: Intrusion Detection (ID) and Intrusion Response (IR). The research on the former has been very active, also taking advantage of the century-long scientific literature on time series analysis and anomaly detection. The latter has instead been traditionally overlooked (Yuan et al., 2014), until the last decade. Indeed, to our knowledge, the work proposed by Fisch (1996) in 1996 pioneered the field of IR, by proposing a tool for mapping nine different response sets to as many suspicion levels. Notably, it took almost two decades for the research work in IR to reach a critical mass that allowed the publication of the first taxonomy of Intrusion Response Systems (IRS) in 2007 (Stakhanova et al., 2007). Since then, work on IR considerably sped up, delivering approaches that can be divided into three categories: (i) static mapping, where the appropriate response to an attack is selected according to some information (e.g., the type of the attack or the state of the protected system); (ii) dynamic evaluation, where the response action is selected among a set of candidates, according to some optimization criteria; and (iii) self-evolving, where the response action that is activated upon the verification of some conditions may vary over time, taking into account the evolution of the attack strategies and the defended system.

From another perspective, most of the IR approaches proposed so far rely on some model. Some of them rely on attack models only; others are based on the model of the system that has to be defended only; finally, some of them make use of a model that includes the interactions between the attacker and the defender. In particular, attack models commonly use a graphical representation based on Attack Trees (AT)

* Corresponding author.
E-mail addresses: tommaso.caiazzi@uniroma3.it (T. Caiazzi), stefano.iannucci@uniroma3.it (S. Iannucci), val.marini9@stud.uniroma3.it (V. Marini), mat.foschi@stud.uniroma3.it (M. Foschi), riccardo.torlone@uniroma3.it (R. Torlone).

https://doi.org/10.1016/j.cose.2026.104834
Received 13 May 2025; Received in revised form 2 August 2025; Accepted 12 January 2026
Available online 22 January 2026

or Attack Graphs (AG). Such models intuitively represent the evolution of an attack and, when extended with information regarding possible response actions, like in the case of Attack-Defense Trees (ADT), they can be used to implement elementary IRSs based on the static mapping paradigm.

More sophisticated approaches have been proposed that make use of models that represent the dynamics of a system at run-time, as well as the strategies of the attackers while performing an attack (e.g., An et al., 2018; Iannucci and Abdelwahed, 2018; Yarygina and Otterstad, 2018; Zonouz et al., 2014). Such models are, on the one hand, more detailed and accurate, potentially leading to an improved response action selection; on the other hand, they often still lack time modeling, which is fundamental in attack/defense games. Building a model that incorporates all the aforementioned elements requires specialized mathematical skills, which may hinder the adoption of the IRS.

For these reasons, and considering that ATs are widely used by security professionals (Kordy et al., 2014), we introduce in this work a new methodology, named PANACEA, that, given an (augmented) ADT, automatically generates a timed competitive stochastic game able to provide a timely response to any attack modeled by the ADT. We formally show that, under certain conditions (more details in Section 3), (i) the usage of an IRS based on a static mapping over an ADT always leads to the underestimation of the cost of the defense, due to the lack of long-term time-aware planning; (ii) IRSs based on a static mapping over an ADT produce defense policies whose cumulative cost is higher than or equal to the cost of the policies computed by PANACEA; (iii) not considering time leads to the underestimation of the defense cost.

Experimental results show that, while the state space of the generated stochastic game grows exponentially with the size of the source augmented ADT, memory usage remains practicable even with medium-large source ADTs. Moreover, we introduce an efficient discrete-time modeling technique, inspired by Boyan and Littman (2000), that prevents the generation of an infinite state space and stresses the need for time modeling by empirically showing how the strategies of the attacker and defender may completely change when solving two games that only differ in the execution time of a single action.

PANACEA leverages on ADTool (Kordy et al., 2013), a widely used AT design tool, for building the augmented ADT; and on PRISM-games (Kwiatkowska et al., 2020), a probabilistic model checker tool supporting stochastic multi-player games. Specifically, it automatically converts the specification of an augmented ADT into the PRISM modelling language, enabling the automatic verification of equilibria-based properties. The tool is available with an open source license at Panacea (2026).

The paper is organized as follows: in Section 2 we introduce several formalisms upon which the proposed methodology is built, and we carefully examine the motivations behind this work; in Section 3 we describe the conversion process from augmented ADT to a multi-agent Markov Decision Process (MDP) and provide a formal demonstration to (i), (ii), (iii); in Section 4, experimental results are discussed; related work is presented in Section 5; the research challenges that typically characterize the IR field are examined in Section 6, where hints for future research directions are also provided. Finally, conclusions are drawn in Section 7.

2. Background and motivations

2.1. Background

Attack Trees. An Attack Tree (AT) is a hierarchical representation of possible multi-stage attacks on a system, network, or organization. It comprises a set of nodes, which identify attacker (sub)goals, and edges, which represent the steps to be taken by the attacker to reach the objectives, along with logical operators that indicate whether the attack progression needs multiple sub-objectives to be realized. The root of the AT is the ultimate goal of the attacker, while the leaves usually represent the vulnerabilities that are considered to be the entry point of the attack process. The progression of an attack flows from the leaves to the root of the AT. ATs were first introduced in Schneier (1999) in 1999 and, since then, they have gained extensive acceptance by cybersecurity professionals (Kordy et al., 2014) because they offer a convenient, graphical way to represent security threats. For this reason, several variants of AT have been proposed, among which are the Attack-Defense Tree (ADT Kordy et al., 2011a) and Attack-Response Tree (ART Zonouz et al., 2014). The former extends the concept of AT, which models exclusively the behavior of the attacker, to include information on how to counter or mitigate the ongoing attack; the latter focuses instead on the consequences that attacks have on the victim and on how to mitigate them.

An example of ADT is presented in Fig. 1. Two types of nodes are depicted: attack nodes (red) and defense nodes (green). Several paths can be taken by the attacker to reach the root of the tree, all starting from the leaves $N3$, $N9$, $N10$, $N11$, $N5$. Some attack nodes are connected to defense nodes. For example, the defense node $C3$ is connected to the attack node $N3$, which means that an attack that has reached $N3$ can possibly be countered by action $C3$. Solid edges represent (a set of) actions that the attacker can launch to attempt achieving the next subgoal. The presence of an arc connecting multiple edges incoming to a given node (e.g., $N1$) indicates that an attacker must satisfy *all* the subgoals identified by the children nodes ($N3, N4$ in the example) in order to achieve the goal of the parent node ($N1$).

Markov Decision Process. A Single-Agent Discrete-Time MDP is a stateful probabilistic approach to model the behavior and the runtime dynamics of a system. An MDP is a tuple: $\langle S, A, P, R, \gamma \rangle$, where S represents the state space that an agent can navigate and $s_k \in S$ represents the agent's state at discrete time k. Even if not explicitly considered in the MDP framework, a common practice is to use a factored state representation (Givan et al., 2003), which allows the characterization of each state with a collection of attributes. A is the finite set of actions available to the agent to navigate the state space. Specifically, by executing an action $a_k \in A$ at time k in the current state $s_k \in S$, the agent moves to a successor state $s_{k+1} \in S$. The transition dynamics from the current to the next state are given by the transition probability function P. This function specifies, for each source state $s_k \in S$, for each destination state $s_{k+1} \in S$, and for each action $a_k \in A$, the value $P(s_k, a_k, s_{k+1})$, that is, the probability value that by executing the action a_k in state s_k at time k, the resulting state will be s_{k+1}. Every time an action is executed, the MDP agent is rewarded with a bonus (or penalized with a cost), according to the reward function R. That is, $R_k = R(s_k, a_k, s_{k+1})$ represents the reward that the agent will earn (or the cost that the agent will pay) for executing at time k the action a_k in state s_k and being taken to some state s_{k+1}. Finally, $\gamma \in [0, 1]$ is the discount factor, which specifies how much short-term rewards are preferred over long-term rewards, and is used in the computation of the discounted cumulative reward described below. The overall behavior of the agent is described by a deterministic or stochastic policy π. When π is deterministic, it maps the action a_k that the agent must execute to each s_k. When π is probabilistic, it specifies a probability distribution such that $\pi : S \times A \to [0, 1]$. The objective of the agent is to find a policy π^*, such that, for each k, the discounted reward $R_k = \sum_{j=0}^{\infty} \gamma^j R_{k+j+1}$ is maximized or the discounted cost is minimized.

Several optimal and suboptimal algorithms for solving MDPs have been proposed (e.g., Bellman, 1957; Kearns et al., 2002; Kocsis and Szepesvári, 2006; Li and Littman, 2008), but one of the most commonly used remains the Value Iteration (VI) algorithm (Bellman, 1957) because of its simplicity. It is based on the concept of *state-value* function $V_\pi(s_k) = \mathbb{E}_\pi[R_k | s_k]$, that is, the expected reward achievable by the agent starting from the state s_k and following policy π. The base step of the algorithm is to assign an initial random state-value V^0 to all states and then to execute the iterative refinement process described in Boutilier (1996):

$$V^{i+1}(s_k) = \max_{a_k \in A} R(s_k) + \gamma \sum_{s_{k+1} \in S} P(s_k, a_k, s_{k+1}) V^i(s_{k+1}) \quad (1)$$

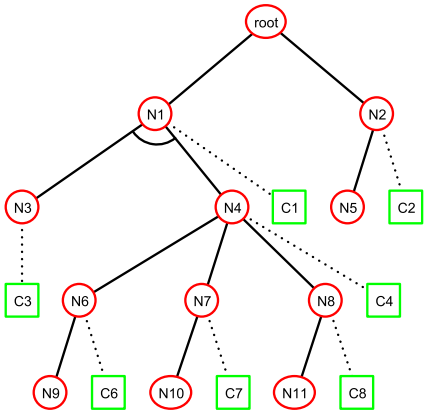

Fig. 1. Example attack-defense tree.

The sequence of functions V^i converges to the optimal value V^* in the limit and provides the expected maximum reward obtainable by following the optimal policy π^* from state s_k.

Multi-Agent Discrete-Time Markov Decision Process (MA-MDP). A MA-MDP, also known as *Stochastic Game*, is an extension of the Single-Agent MDP. A MA-MDP with n agents is defined by the tuple $\langle S, A_1, \dots, A_n, P, R_1, \dots, R_n, \gamma \rangle$. With this formulation, different agents have possibly different sets of available actions and can have potentially different reward functions. According to the definition of reward function, the agents could cooperate to reach a common objective or could behave selfishly, i.e., trying to achieve personal goals at the expense of the other agents. The former case is described in detail in Boutilier (1996), while the competitive case is described in Busoniu et al. (2008). MA-MDPs are the foundations for the realization of the stochastic games aimed at providing a response planning methodology, as described in Section 3.

2.2. Motivations and goal

MDPs are effective, but difficult to model. The modeling of MDPs offers a powerful framework for capturing the dynamics of decision-making in complex and uncertain environments. However, since it is not widely recognized as a standard methodology for protecting computer systems, it could be difficult for a system administrator to adopt an IR technique that requires the manual design of a MDP.

Another drawback of an approach based on MDP is that it is difficult to find the right granularity at which the model should be created. Indeed, the number of state variables and actions that may be considered in the creation of the model of the system that has to be defended and/or of the behavior of the attacker could be overwhelming, leading easily to the design of over-sophisticated models that, in turn, may lead to a degradation in terms of the planning time and scalability.

ADTs cannot consider long-term planning. On the other hand, while ADTs are widely recognized as a standard framework for designing IRSs, they are based on a static mapping mechanism that associates a (set of) response actions to a given system state. As a consequence, they could produce suboptimal policies because they do not consider long-term planning, contrarily to MDPs, as shown in Section 3.2.

For example, consider the ADT represented in Fig. 1. Should the attacker reach node $N8$, the defender could respond triggering the response action $C8$. However, while this could stop the attacker from proceeding further to $N4$, it does not prevent the attacker from trying other entry points, such as, $N10$ or $N9$. Following the same logic, should the attacker reach $N7$, $C7$ would be executed to counter it and, similarly, $C6$ would be executed to counter a possible attacker reaching $N6$.

The main limitation of ADTs and static mapping between attacks and responses is that it is, by nature, reactive; that is, it does not consider the potential countermoves of the attacker. As a consequence, a defense approach based on ADT will always be one step behind the attacker. Although this is intuitively suboptimal, we would like to provide here an initial quantitative example of how a proactive approach could improve the defense strategy. In particular, consider now the same ADT of Fig. 1 and assume that the cost for executing $C6$, $C7$, and $C8$ is equal to 1 unit each, while the cost for executing $C4$ is equal to 2 units. It is clear that, if the attacker reaches $N8$, as in the example above, and the likelihood of the attacker trying other entry points is high enough, the execution of $C4$ should be preferred to the execution of $C8$, $C7$, and $C6$ because it has a lower cost and proactively stops any other move the attacker may attempt on the subtree rooted at $N4$. Planning the execution of $C4$ cannot be done with the information provided in the ADT alone.

Key-idea: building MDPs starting from ADTs. For the reasons above, we introduce a novel technique that allows the automatic construction of MDPs from augmented ADTs. The rationale is that the system administrator could leverage existing ADTs and, at the same time, take advantage of the long-term planning capabilities of MDPs. The advantages of the proposed approach are manifold: first, it leverages an existing and established formalism that is presently widely used to model attack and defense strategies. Second, it leverages a solid mathematical framework that allows both single-agent and multi-agent planning, thus providing an effective way to simulate best-case, worst-case, and average-case attack and response scenarios; third, the proposed modeling technique, unlike most of the existing work, supports the concept of time, which is paramount to obtain an accurate simulation of attack and response; fourth, by utilizing existing ADTs, it seamlessly reflects the granularity at which the system administrator has already modeled the system, in turn limiting potential scalability issues. In what follows, we provide an in-depth description of works that are closely related to our approach. Other related works are broadly discussed in Section 5.

RRE: A Game-Theoretic Intrusion Response and Recovery Engine. In Zonouz et al. (2014), the authors propose a hierarchical IRS that employs *engines* on local machines to solve the IR problem locally and a global solver if none of the solutions calculated locally satisfy the security policy. Similar to our work, competitive games are used at both levels. However, they are formulated as sequential Stackelberg games, in which the IRS assumes the role of the leader, and the attacker assumes the role of the follower. However, we argue that this model does not correctly represent the dynamics of an IR game. Indeed, due to the non-negligible duration of the actions during the game, the roles of leader and follower may swap over time. Another point of similarity with our work is the automatic conversion from an AT-based representation to a Competitive MDP. In our work, we use an extension of ADT, which models the progress of an attack; in Zonouz et al. (2014), an ART is used, which models the level of compromise. Unlike ADTs, in ARTs there are no attack nodes, and there is no representation of a possible multi-stage attack. For this reason, attack actions are derived only from effects indicated in the leaf nodes of the ART. As a consequence, the resulting automatically generated game will not support multi-stage attacks.

Model-Based Response Planning Strategies for Autonomic Intrusion Protection. To address the need for timed games, in our previous work (Iannucci and Abdelwahed, 2018), we introduced a two-agent IR modeling technique that supports timed actions, empirically demonstrating how the consideration of time could have a huge impact on the simulation results. In such work, we made use of a Multi-Agent discrete MDP (MA-MDP) to model the interactions between an attacker agent and a defender agent making decisions upon a shared state space, where each state was partially visible to the agents. Time synchronization between the agents is managed using two timers representing the attacker and the defender state-in-time. Every action executed by any agent increases its timer by the expected duration of the executed action, and an agent is allowed to execute the next action only if its state-in-time is lower than or equal to the state-in-time of the other agent. The experimental results show that, while the introduction of timed actions

Table 1
Symbol summary.

Symbol	Description
S	State space of the MDP
A	Set of all actions
S_τ	Target states where execution stops
A_α	Attacker's action set
A_δ	Defender's action set
V_ρ	Pre-condition nodes
V_α	Attacker action nodes
V_δ	Defender action nodes
$V = V_\rho \cup V_\alpha \cup V_\delta$	Set of R-ADT nodes
$\rho \in V_\rho$	Precondition node
$\alpha \in V_\alpha$	Attack action node
$d \in V_\delta$	Defense action node
E	Set of directed edges of the R-ADT
$p \in P$	Attack path
$p\|_s \in P$	Attack path starting from node ρ_s
$\mathbb{V}_{\rho,\alpha} : V_\rho \to \{T, F\}$	Evaluation function of attacker pre-conditions
$\mathbb{V}_{\rho,\delta} : P \to \{T, F\}$	Evaluation function of defender pre-conditions
$\mathbb{V}_\alpha : V_\alpha \to A_\alpha$	Mapping function from attack nodes to attack actions
$\mathbb{V}_\delta : V_\delta \to A_\delta$	Mapping function from defense nodes to defense actions
$R(\mathbb{V}_\delta(d))$	Reward (cost) of defense action d

is paramount to accurately modeling the IR problem, the introduction of time-based synchronization and multi-agent modeling increases the size of the model, leading to planning times that are hardly manageable without the usage of heuristics, potentially leading to suboptimal strategies (more details of a state-space pruning heuristic for IR can be found in Iannucci et al. (2018)). In particular, we argue that the way time is modeled in Iannucci and Abdelwahed (2018) is inefficient because it leads to potentially infinite state spaces.

3. Intrusion response modeling

3.1. MDP-based modeling

Single-Agent IR Modeling. We extend the basic MDP framework by introducing: a *termination function* τ and a *pre-condition function* ψ. The termination function $\tau : S \to \{true, false\}$ is used to define the subset $S_\tau = \{s \in S | \tau(s) = true\}$ of the target states, which represents the set of states where the agent stops its execution because the system is considered secure or irrecoverably compromised; the pre-condition function $\psi : S \times A \to \{true, false\}$ is instead used to define whether an action $a \in A$ is executable in state $s \in S$. The objective of the MDP agent is to drive the system from an initial state $s \notin S_\tau$ to a target state $s' \in S_\tau$ such that the path between s and s' minimizes the cost. It is worth noting that this formulation can be used to model either a defender agent or an attacker agent. In the former case, A will contain defense actions, and S_τ will represent the subset of states in which the system is considered safe; in the latter case, A will contain attack actions, and S_τ will represent the subset of states in which the system is irrecoverably compromised.

Multi-Agent IR Formulation. The IR formulation of a single agent captures the dynamics of the underlying system and can be used to plan optimal long-term policies to defend the system against an attack. However, even if long-term policies generally outperform short-term policies (Iannucci and Abdelwahed, 2016), an IRS built on such a model is not able to anticipate, and thus prevent, a possible multistep attack because the model does not include the attacker behavior. The competitive multi-agent IR formulation introduces a proactive defense mechanism by describing the system and its dynamics when subject to control actions executed by both the defender and the attacker. Knowing what actions are available to the attacker enables the planning of proactive long-term response policies that are aware of the strategy of the attacker and that are thus potentially able to prevent an attack escalation. Therefore, while the single-agent formulation is suitable to respond to zero-day attacks, the multi-agent formulation can take advantage of prior knowledge about the attacker's behavior to build a comprehensive model that considers the interactions of the attacker and defender.

In this model, each attack is characterized by two factors: (i) an attack action, based on preconditions that can specify dependencies on other system attributes; and (ii) the effects that the attack has on the system attributes. For simplicity and without loss of generality, in what follows, we describe the extensions needed to realize a two-agent (an attacker and a defender) stochastic game. The same considerations can be applied to games with an arbitrary number of agents. The two-agent model inherits all the characteristics of the single-agent model and extends (i) the set of state attributes, (ii) the set of available actions, which is now per-agent, and (iii) the reward function, which is now per-agent. It is worth noting that the multi-agent IRS design cannot generally be thought of as a zero-sum game, because attacker and defender could have arbitrary reward functions that are not necessarily one the opposite of the other.

We argue that, differently from a single-agent game, where the execution time of the actions may or may not be part of the model and could be possibly included in the reward function (e.g., to minimize the time-to-protection), in an IR multi-agent game, the execution time of the actions must be a structural part of the model due to the need for synchronization between agents. In fact, while several approaches to IR that have been proposed leverage sequential games, mostly using generic turn-based games (Iannucci and Abdelwahed, 2018; Emami-Taba et al., 2015; Kundu and Ghosh, 2014) or Stackelberg games (Zonouz et al., 2014; Tang et al., 2024; Bilinski et al., 2019), they do not correctly capture the dynamics of attacks and responses in a system. Indeed, in a real scenario, each agent has its own execution thread, which could be affected somehow by actions executed by other agents. For this reason, we include in the multi-agent IR formulation the notion of time, with a modeling approach similar to the Time-Based MDP that has been introduced in Boyan and Littman (2000), where a continuous variable representing the time was added in the structure of the MDP state. However, the introduction of a continuous variable in the state structure could lead to a state space explosion that would make the IR problem hard to be solved in a reasonable time. For this reason, with some loss of precision, we model time with a discrete variable $t \in \mathbb{Z}$ that represents the time difference between the two agents. In the initial state of the game $t = 0$. The variable is then increased (decreased) by the execution time of an attack (defense) action. The attacker (the defender) can run the next action if $t \leq 0$ ($t \geq 0$). Using this variable, it is possible to model time in a sequential game, where at each stage any given agent can either execute a new attack (or defense) action, or a *no operation (nop)* action if it is waiting for the previous action to be completed. For example, suppose that initially $t = 0$ and that the attacker executes an action a_α with execution time $t(a_\alpha) = 5$. After the execution of action a_α, $t = 5$. At this point, before executing the next action, the attacker has to wait for one or more actions a_δ of the defender totaling at least 5 time units.

Other options to effectively model time in a MDP include the usage of Semi-MDPs (Ibe, 2013) (SMDP). A Semi-MDP is a MDP in which the agent, after having decided which action to take at a given state, waits a certain amount of time, called *holding time* and denoted as $H(s_k, a_k, s_{k+1})$. Unlike the proposed approach, which models time as a deterministic-valued component, this formulation has the advantage of being more expressive, because it allows the definition of the holding time as either a probability mass function (for discrete time modeling) or a probability distribution function (for continuous time modeling). When the time between decision epochs is exponentially distributed, the SMDP is also called Continuous-Time MDP (CTMDP). However, while on the one hand modeling the distribution of the holding time could model the system dynamics in a more accurate way than only using mean execution time values, on the other hand it increases both the modeling effort and the computational complexity, requiring to specify and validate time distributions and possibly leading to uncountable state representations.

3.2. Converting an attack-defense tree to a stochastic game

Augmenting the ADTs. ATs and ADTs do not express a specific semantic for nodes and edges. Indeed, their meaning is left to the designer, who can sometimes assume a node to be an action, a system attribute, or a vulnerability. While on the one hand, the lack of predefined semantics guarantees the maximum flexibility, because ATs and ADTs can be fine-tuned to the needs of the specific system and system administrator, on the other hand, it can be seen as a limitation when they have to be used programmatically. Moreover, neither ATs nor ADTs support the notion of *reward*, which is necessary for cost-aware planning. For this reason, we introduce here an extension of the ADT formalism, hereafter referred to as *Reward ADT* (*R-ADT*). Unlike an ADT, an R-ADT enforces a predefined structure, where the nodes can be either of type *attacker action* or *defender action* or *pre-condition*.

An example of an R-ADT is shown in Fig. 2. Nodes labeled $N*$ represent preconditions necessary for the attackerâ€™actions, denoted by $A*$. The $D*$ nodes correspond to defense actions that, once executed, block the associated attacker actions sharing the same precondition node. The R-ADT is derived from the ADT in Fig. 1, preserving its logical structure while enforcing a more organized layout. In this structure, different system states (pre-conditions) can be reached only by executing specific actions, which themselves depend on certain preconditions. Starting from the leaves, the semantics of the ADT nodes are evaluated to identify actions and their corresponding preconditions (e.g., system attributes and vulnerabilities). The transformation process is then straightforward, following two basic rules: (i) an action node can have one or more precondition nodes as children; (ii) an action node must have exactly one precondition node as its parent, representing a component of the new system state resulting from the action's execution.

More formally, let $T = (V, E)$ be a R-ADT, with $V = V_\rho \cup V_\alpha \cup V_\delta$ representing the set of nodes and E representing the set of the edges, where V_ρ is the subset of attack preconditions, V_α is the subset of attack actions, and V_δ is the subset of defense actions.

We define:

- $\mathbb{V}_{\rho,\alpha} : V_\rho \to \{true, false\}$ as the attacker evaluation function for pre-condition nodes which, given a precondition node, returns a boolean value representing whether the precondition is met;
- An attack path $p = [\rho_1, \alpha_1, \rho_2, \alpha_2, \dots, \rho_{G-1}, \alpha_{G-1}, \rho_G] \in P$ where $\rho_i, \in V_\rho$, $\alpha_i \in V_\alpha, i \in [1, G]$, and G is the number of pre-conditions that can be activated from ρ_1 to the goal;
- $\mathbb{V}_{\rho,\delta} : P \to \{true, false\}$ as the defender evaluation function for pre-condition nodes, which, given a path p, returns a boolean value representing whether the precondition is met. In particular, we define:

  $$\mathbb{V}_{\rho,\delta}(p) = true \iff \exists \rho_i \in p.\ \mathbb{V}_{\rho,\alpha}(\rho_i) = true$$

  In other words, the satisfaction of any precondition in a given attack path p causes all the defense actions along that path to be enabled. We would like to point out that this definition does not require the first active precondition to be a leaf, accounting thus for attacks that have not been detected at the initial stages and for attacks that have not been properly modeled, and for which the initial detection led to the activation of a non-leaf pre-condition function. However, this is without loss of generality because the initial pre-condition of the path could be a leaf.
- $\mathbb{V}_\alpha : V_\alpha \to A_\alpha$ is the mapping function for attack actions, which maps attack nodes to attacker actions A_α;
- $\mathbb{V}_\delta : V_\delta \to A_\delta$ is the mapping function for defense actions, which maps defense nodes to defender actions A_δ.

Edges in the R-ADT represent relationships within connecting nodes. Given any two nodes $a, b \in V$, we denote with $a \xrightarrow{e} b$ the *parent-child* relationship between a and b, where $e \in E$ is the connecting edge, b is the parent node and a is the child node. The semantics of the edge depends on the pair of nodes it connects. Assume $a \xrightarrow{e} b$:

- with $a \in V_\rho$ and $b \in V_\alpha$, e is an association between the precondition $\mathbb{V}_\rho(a)$ and the attack action $\mathbb{V}_\alpha(b)$ that can be launched when the precondition is met;
- when $a \in V_\alpha$ and $b \in V_\rho$, e is an association between the attack action $\mathbb{V}_\alpha(a)$ and the precondition of the next attack in the chain $\mathbb{V}_\rho(b)$, which will be set to *true* if the attack is successful, determining in other words the post-condition of the attack action $\mathbb{V}_\alpha(a)$;
- with $a \in V_\delta$ and $b \in V_\rho$, e is an association between response action $\mathbb{V}_\delta(a)$ and the pre-condition $\mathbb{V}_{\rho,\delta}(p)$, where $p \in P$ is any path for which $b \in p$ holds, indicating that response action $\mathbb{V}_\delta(a)$ can be executed when precondition $\mathbb{V}_{\rho,\delta}(p)$ is met.

It is worth noting that, differently from the attack actions, which can be executed only when their respective pre-conditions are satisfied, defense actions can be executed when *any* of the pre-conditions in a given path is satisfied. This assumption enables the consideration of proactive responses, as illustrated in the example of Fig. 1. Table 1 summarizes the main symbols used in the model.

One of the limitations of ADTs is that, in terms of IR, they cannot consider alternative actions in a given attack stage, because they do not support any mechanism to evaluate the quality of a response. For this reason, they can be considered as a mere extension of a static attack-response mapping, with the advantage of being based on a multi-stage model of the attack instead of a flat representation. Nonetheless, it is common to consider multiple response actions for a given attack stage. For this reason, an R-ADT allows the specification of a *per-action reward* $R(\mathbb{V}_\delta(d)) \in \mathbb{R}, d \in V_\delta$, that is, a score that can be associated to an action to quantify its quality with respect to the attack node it is connected to.

Unfortunately, while R-ADTs allow the evaluation of multiple defense actions when a given pre-condition is met, they still do not support long-term reasoning or, in other words, answering questions like: *is there any combination of defense actions, not necessarily associated to the node representing the current stage of the attack, that could yield a reward that is greater than the reward obtainable with the execution of the presently best rewarding action?*

To systematically answer this question, we propose an automatic conversion of the R-ADT into a multi-agent MDP. The conversion is carried out under the assumption that the reward function is defined as strictly positive and strictly monotonic, that is:

$$\forall d \in V_\delta.\ R(\mathbb{V}_\delta(d)) > 0 \tag{2}$$

and

$$\forall d_1, d_2 \in V_\delta\ \forall \rho_1, \rho_2 \in V_\rho\ \ d_1 \to \rho_1, d_2 \to \rho_2 \tag{3}$$

$$\rho_1 \to^* \rho_2 \Rightarrow R(\mathbb{V}_\delta(d_1)) < R(\mathbb{V}_\delta(d_2)) \tag{4}$$

where $\rho_1 \to^* \rho_2$ indicates that there is a subpath of any length from ρ_1 to ρ_2.

In the Intrusion Response field, it is common to consider the reward function as a cost function that must be minimized. Thus, in what follows, we will consider the minimization of a cost function. Assuming that the cost function is defined as strictly positive means that there are no defense actions that can be executed for free. The rationale is that any defense action, no matter how fast, cheap, or with low impact on the defended system, would not be executed if the system is not under attack because it would not provide any benefit.

Furthermore, we argue that it is reasonable to consider the cost function to be strictly monotonic, given a path p. Indeed, a path represents an escalation that an attacker can conduct against the protected system, which leads to an increasing level of compromise, and thus to more impactful actions that must be executed to defend the system. Should we relax the assumption, it could be possible to define, as an extreme case, a single defense action with the lowest cost that is actionable at the root of the tree, making the solution of the problem trivial, as it would always be possible to defend the system using the same action with minimal cost regardless of the stage of the attack. However, this is clearly in contrast with the idea that attacks become more serious stage after stage.

We would like to point out, however, that the monotonicity assumption is not a mathematical requirement for the validity of the proposed approach, as Propositions 1–3, described later in this section, would still hold.

Translating the R-ADT to MDP. As described in Section 2.1, a two-agent stochastic game M is a tuple $M = \langle S, A_\alpha, A_\delta, P, R_\alpha, R_\delta, \gamma \rangle$. In what follows, we describe the process to obtain M from the R-ADT.

First, a factored representation is used, where as many state variables are created as the number of precondition nodes in the R-ADT. For simplicity, but without loss of generality, we consider boolean state variables. Let $\vartheta_\alpha = \{v_{\alpha,1}, v_{\alpha,2}, \dots, v_{\alpha,|V_\alpha|}\}$ be the set of the attacker attributes and $\vartheta_\delta = \{v_{\delta,1}, v_{\delta,2}, \dots, v_{\delta,|V_\delta|}\}$ be the set of the defender attributes. The state space of the MDP is then given by:

$$S = \{(v_{\alpha 1}, v_{\alpha 2}, \dots, v_{\alpha|V_\alpha|}, v_{\delta 1}, v_{\delta 2}, \dots, v_{\delta|V_\delta|}) | \\ \forall i \; \forall j \; v_{\alpha,i} \in \{T, F\}, v_{\delta,j} \in \{T, F\}\}$$

Second, the stochastic game action sets A_α and A_δ are mapped respectively to V_α and V_δ through $\mathbb{V}_\alpha$ and $\mathbb{V}_\delta$. Third, the transition probability function $P : S \times A_\alpha \cup A_\delta \times S \to \mathbb{R}$ is mapped to $P : S \times \mathbb{V}_\alpha \cup \mathbb{V}_\delta \times S \to \mathbb{R}$.

Suppose now an MDP has been derived from an R-ADT.

Proposition 1. *The realization of a static mapping IRS based on the R-ADT alone leads to an underestimation of the defense costs.*

Proof. Let $\rho_s \in V_\rho$ be the highest satisfied precondition along path p, that is, $p = [\rho_1, \alpha_1, \dots, \rho_s, \alpha_s, \dots, \alpha_{G-1}, \rho_G]$. We indicate with $p|_s$ the subpath of p starting from ρ_s. Let D be the subset of defense actions that are available on that path: $D = \{\mathbb{V}_\delta(d) | \mathbb{V}_{\rho,\delta}(p|_s)\}$. The selection of the response action $\mathbb{V}_\delta(d)^*$ that minimizes the cost is trivial: $\mathbb{V}_\delta(d)^* = argmin(R(\Delta)), \Delta \in D$. We want to show that such value, which is the estimated cost incurred in the execution of the immediate optimal action, as calculated using the R-ADT alone, is always lower than or equal to the optimal state value $V(s)^*$.[1] Since, by definition $V(s) \leq V(s)^*$:

$$R(\mathbb{V}_\delta(d)^*) \leq V(s) \Rightarrow R(\mathbb{V}_\delta(d)^*) \leq V(s)^* \tag{5}$$

First, we need to find a map from d to s. To this end, recall that, by design, $\mathbb{V}_{\rho,\delta}(p)$ represents the subset of satisfied pre-conditions given a path p, and that D, as previously defined, represents the set of defense actions that can be executed. In the MDP context, this is equivalent to the identification of the subset of states $S_D \subseteq S$ where the actions $\Delta \in D$ are activable: $S_D = \{s \in S | V_{\rho,\delta}(p)\}$. Thus, if $\gamma = 0$, and therefore only the optimal immediate action is considered:

$$V(s) = \min_{\delta \in A_\delta} R(s) = R(\mathbb{V}_\delta(d)^*) \tag{6}$$

If $\gamma > 0$, using Eq. (1), it is trivial to prove that $V(s)_{\gamma=0} \leq V(s)_{\gamma>0}$. □

Proposition 2. *The total cost of execution of an optimal policy calculated using the IRS based on the stochastic game is always lower than or equal to the total execution cost of multiple single-action policies calculated by repeatedly selecting the action at minimum cost using an R-ADT.*

Proof. We compare the long-term optimal policy π^* from state s of the defender agent obtained by optimally solving the stochastic game to the short-term, greedy policy π_g obtained by consecutively selecting the action at minimum cost, reasoning on the R-ADT from state s.

By definition, the expected discounted cumulative cost obtainable from state s in an MDP whose reward function has to be minimized is given by:

$$V^*(s) = \min_\pi \mathbb{E}^\pi [\sum_{t=0}^{\infty} \gamma^t R(s_k, a_k) | s_0 = s] \tag{7}$$

The greedy policy π_g is one of the possible policies, and therefore $V^{\pi_g} \geq V^*(s)$ must hold. □

[1] Recall that the state value $V(s)^*$ of a state s is the expected cumulative reward that an agent can obtain starting from state s and following the optimal policy π^*

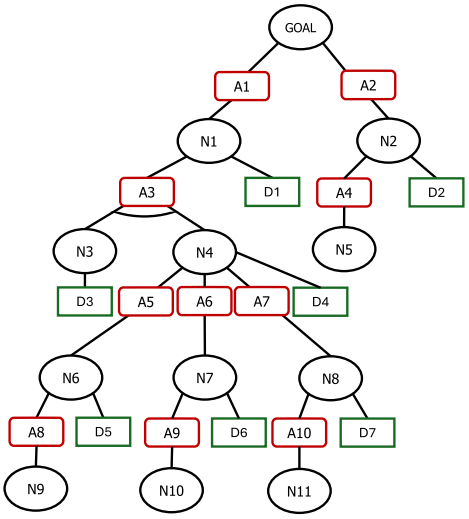

Fig. 2. Example of reward attack-defense tree.

Proposition 3. *The optimal policy obtained by an IRS based on a stochastic game without considering time yields a discounted cumulative cost that is lower than the discounted cumulative cost of the optimal policy obtained by solving the game considering timing aspects.*

Proof. The stochastic game with timing inherits all the elements of the stochastic game without timing, while at the same time it extends the state attributes to include time management, as discussed in Section 3.1, and tightens the action preconditions based on the newly introduced state attribute. By monotonicity of constrained optimization, the optimal value under a stricter constraint set cannot be better than under a relaxed one (Nocedal and Wright, 2006). □

Example of translation. To illustrate how the translation process works, we focus on the R-ADT depicted in Fig. 2, omitting temporal aspects for simplicity. PANACEA parses the tree structure, creating a distinct state variable for each precondition node. Following the tree's topology, the system links states via actions, assigning roles (*i.e.*, attacker or defender), costs, and execution times based on the R-ADT metadata. This results in a system state defined by 12 variables (including the goal), where the attacker has access to 10 different actions and the defender to 7. Supposing each state variable can assume only binary values (*e.g.* `active` or `inactive`), the resulting MDP comprises approximately $2^{12} = 4096$ states. In general, we assume that the pre-conditions associated with leaf nodes are always satisfied, as they represent exploitable vulnerabilities in the monitored system that are publicly accessible. To complete the formulation, we only need to define the reward function (*e.g.*, minimizing costs) according to the specific use case. At this stage, it becomes possible to solve the MDP by computing the optimal defense policy (*i.e.* the best path from the current state to a secure one) for each state of the system.

4. Evaluation

The evaluation is divided into two parts: firstly, we investigate the relationship between the R-ADT size and the corresponding MDP size; secondly, we demonstrate the potential of PANACEA by applying our methodology to a simple example.

Testing Environment. We performed the experiments on a server equipped with an Intel® Xeon® Gold 6126 @ 2.60GHz and 500 GB of RAM. The conversion script for translating R-ADTs into MDPs is written in Python. To solve MDPs, we rely on Prism-Games (Kwiatkowska et al., 2020), a probabilistic model checker for stochastic multi-player games.

Reproducibility. All the code needed to run our Panacea prototype and to reproduce the experiments is available at Panacea (2026).

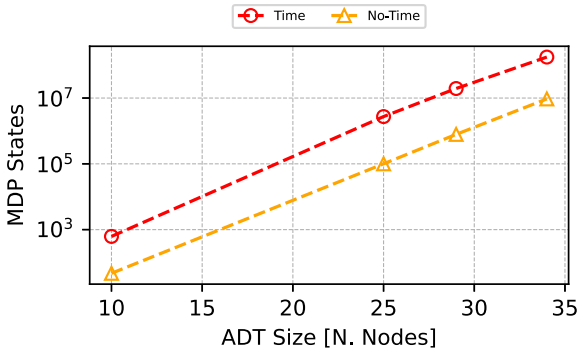

Fig. 3. R-ADT size (number of nodes) over MDP size (number of states).

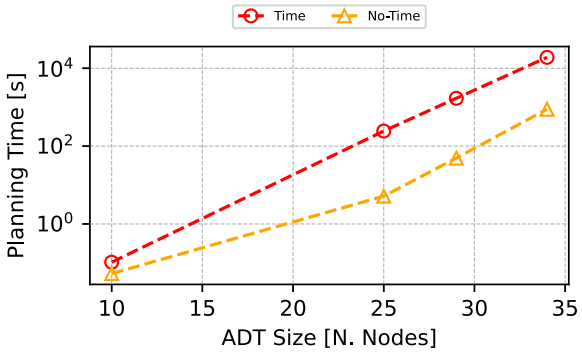

Fig. 4. Planning time (seconds) over R-ADT size (number of nodes).

4.1. Investigating the scalability of PANACEA

Since the aim of this section is to only investigate the scalability of PANACEA, for these experiments, we decided to use arbitrary R-ADTs where the names of actions and labels are generic, and the costs and times of actions have no relations to the actions themselves.

The aim is to answer the following questions:

S1 *"How much does the R-ADT size affect the MDP size?"*
S2 *"How much does the R-ADT size affect the planning time?"*
S3 *"How much does the R-ADT size affect the memory usage during the MDP resolution?"*

S1: MDP states increase exponentially over the number of nodes in the R-ADT. To measure how the size of the R-ADTs affects the number of states of the corresponding MDPs, we measure the size of the MDPs starting from different R-ADTs. Fig. 3 shows the results. The x-axis represents the R-ADT size in terms of the number of nodes, while the y-axis depicts the MDP states on a logarithmic scale. We measure the MDP size in two situations, while not considering time (yellow line with triangle markers) and while considering it (red line with circle markers). The plot shows that the number of MDP states increases exponentially with the R-ADT size in both scenarios. However, the increase is more pronounced in the scenario incorporating time, suggesting that incorporating time into the model significantly increases the complexity and the state space of the MDP. Overall, the figure illustrates that the R-ADT size has a substantial impact on the MDP size, with the effect being more significant when time is considered.

S2: planning time increases exponentially over the size of the R-ADT. To confirm the findings of S1, we also measure how the planning time is affected by the R-ADT size. Fig. 4 shows how the planning time (y-axis) varies as the size of the R-ADT (x-axis) increases. As expected, since the number of states in the MDP increases exponentially, the planning time follows a similar trend. This is true both in the scenario incorporating time (red line) and in the one without it (yellow line). Considering this result, it is worth noting that in the proposed deployment model of PANACEA, the optimal solution to the MDP can be found offline and that the state-action table of the MDP can be queried online in $O(1)$.

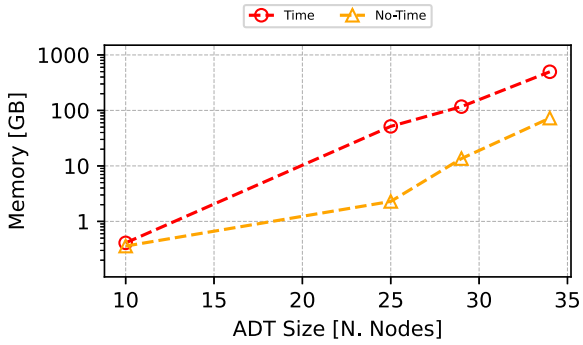

Fig. 5. RAM usage (in GB) needed for the MDP resolution over the R-ADT size (number of nodes).

Table 2
Nodes, software, and their vulnerabilities.

Node	Software	Vulnerability
Web Server	Apache2 - Version: 2.4.49	OS Command Injection (CVE-2021-42013)
Database	MySQL - Version: 8.0.27	Heap-based Buffer Overflow (CVE-2023-38545)

S3: the memory usage needed for the resolution increases exponentially with the R-ADT size. To conclude our analysis of the impact of the R-ADTs size on the scalability of PANACEA, we also investigate the effect on the memory needed for the resolution of the relative MDPs. Fig. 5 shows how the memory needed during the MDP resolution (y-axis) scales over the number of nodes in the R-ADT (x-axis), revealing a trend consistent with that of S1. Indeed, both the resolutions incorporating time (red line) and the one without it (yellow line) show memory values that grow exponentially as the size of the R-ADT increases.

4.2. Use case: Protecting a system with PANACEA

In this section, we assume to act as a network administrator who wants to build an IRS based on PANACEA. The goal is to protect a simple system composed of a switch and two servers, one acting as a web server (running Apache2) and the other as a database (running MySQL).

Table 2 shows, for each node, the version of the running software and the associated vulnerabilities we found on CVE (MITRE, 2022). From these vulnerabilities, we derive the R-ADT depicted in Fig. 6.[2] The cost for each action and the time needed to execute them is reported in Table 3. In this example, values are assigned only to illustrate the use case. We envision an automatic approach based on integrating state-of-the-art frameworks for managing attack and defense phases (*e.g.* Caldera Corporation, 2024, CybORG Cyber operations research gym, 2022) in an emulated environment. The basic idea is to reproduce attack and defense actions in the emulation (the process can be automated) to gather statistics that better estimate costs and times. A discussion on this aspect is presented in Section 6.

For this part, the aim is to answer the following questions:

E1 *"Is the long-term planning of PANACEA truly effective?"*
E2 *"How much does introducing time into the model affect costs?"*

E1: the MDP model outperforms the R-ADT one. To empirically verify that our implementation of PANACEA outperforms R-ADT-based models

[2] Note that both the vulnerabilities and the R-ADT are not intended to be exhaustive, and are used only to show the features of PANACEA.

Table 3
Available actions for each agent.

Agent	Action	Cost	Time (seconds)
Attacker	Web Recon	5	1
	Path Traversal	20	2
	Get Files	30	2
	Get Login Data	10	2
	Buffer Overflow	40	2
	Exfiltrate Data	50	2
Defender	Disable CGI Scripts	45	1
	Update Apache	10	20
	Reconfigure Apache	50	2
	Change Files Permissions	60	2
	Encrypt Files	150	5
	Change Credentials	200	2
	Disable SOCKS5 Proxy	120	1

Table 4
Initial state of the MDP.

Attribute	Value
Data Exfiltrated	0
Access To Reverse Shell	0
Access to MySQL	0
Web Recon Successful	0
Access to Execute Arbitrary Code	0
Access To Sensitive Files	0
Misconfigured Apache	1
Web Server Publicly Exposed	1
CGI Scripts Enabled	1
SOCKS5 Proxy Enabled	1
Vulnerable Apache Version	1
Unencrypted Files	1

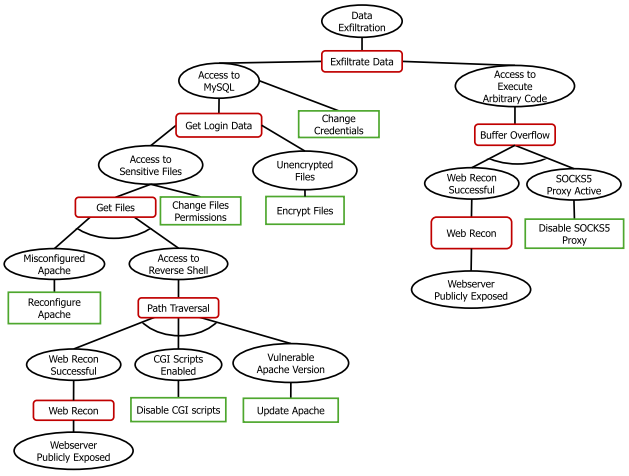

Fig. 6. R-ADT derived from the vulnerabilities of Table 2.

(satisfying Preposition 2), we conducted a test starting from the R-ADT of Fig. 6, comparing three systems: (*i*) a R-ADT-based model which always selects the available action with the lowest cost (R-ADT LC); (*ii*) a R-ADT-based model which always selects the available action at lowest distance from the root, blocking the most dangerous path to the attacker goal (R-ADT LD); (*iii*) a MDP-based model created from the R-ADT (*i.e.*, PANACEA). To compare results, all the formulations are based on a multi-agent game where agents compete to reach their goal, performing actions in turn. The goal of the attacker is to reach the terminal state, represented by the root of the R-ADT in tree-based models and by the equivalent state in the MDP for PANACEA. The defender's goal is to block all the paths through the root when an attack is ongoing, represented in the MDP by all the states from which the attacker cannot reach its goal. In the MDP, the agents' objective function minimizes the costs of the performed actions. The reward function is defined as: $R(a_i) = c_i$, where

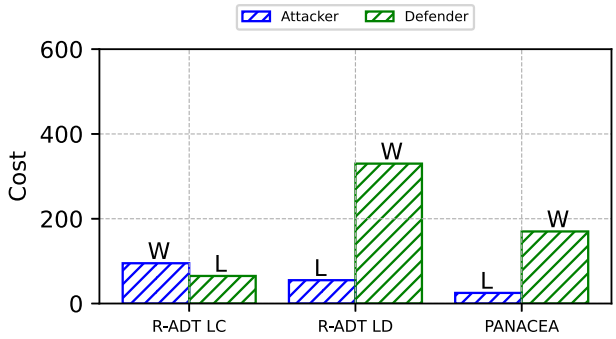

Fig. 7. Cost paid by different models without considering time.

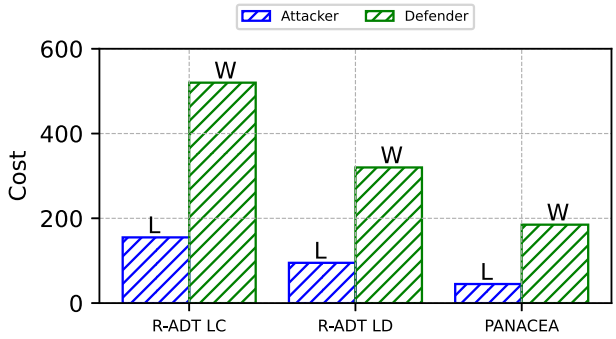

Fig. 8. Cost paid by different models while considering time.

$a_i \in A$, $A = A_\alpha \cup A_\delta$, and c_i represents the cost associated to action i. The initial state for all formulations is depicted in Table 4.

Fig. 7 depicts the results showing, for each model, the outcome of the game (*i.e.* marking with a `W` or a `L` the bar of the winner and loser agent, respectively) and the cost paid by each agent (y-axis) for the optimal resolution. As expected, the results respect Preposition 2, with R-ADT models that always pay a higher cost with respect to PANACEA. Specifically, in the R-ADT LC case, the defender agent, deciding to execute the lowest cost action in the left subtree (*i.e.* Update Apache), permits the attacker to reach the goal by selecting actions on the right subtree (*i.e.* Web Recon and Buffer Overflow). In the R-ADT LD model, instead, the defender action, selecting actions at a lower distance from the root (*i.e.* `Disable SOCKS5 Proxy`, `Encrypt File`, and `Change File Permissions`) wins the game, but pays a higher cost with respect to PANACEA, which is able to determine the best actions to perform to win the game (*i.e.* `Disable SOCKS5 Proxy`, and `Reconfigure Apache`). Fig. 9 graphically illustrates the PANACEA resolution, showing the traversed states, the actions performed, and the costs paid by each agent.

E2: introducing time does not imply better rewards. To understand the impact of time, we performed the same experiment as described in E1, incorporating the time notion explained in Section 3. The execution time for each action is detailed in Table 3. Fig. 8 shows the cost incurred by the models. Firstly, they continue to satisfy Preposition 2, with the MDP-based formulation outperforming the R-ADT-based alternatives. Secondly, introducing the time element refines the model, enabling a more accurate representation of the environment. In accordance with Proposition 3, this results in higher costs, with the MDP-based formulation incurring greater costs for both agents in the time-aware scenario compared to the time-agnostic one.

To conclude our analysis of the impact of time, we tested the sensitivity of the model by varying the execution time of a single action. We evaluated the MDP-based model from previous experiments using three different formulations: one that does not include time (`NO-TIME` in the following), one that includes time and uses the values reported in Table 3 (`TIME-A` in the following), and one where the execution time of

Table 5
Comparison of MDP formulations.

Formulation	States Traversed	Actions Performed	Attacker Cost	Defender Cost	Timer	Winner
NO-TIME	5	4	25	175	N/A	Defender
TIME-A	15	5	45	190	2	Defender
TIME-B	20	5	95	65	4	Attacker

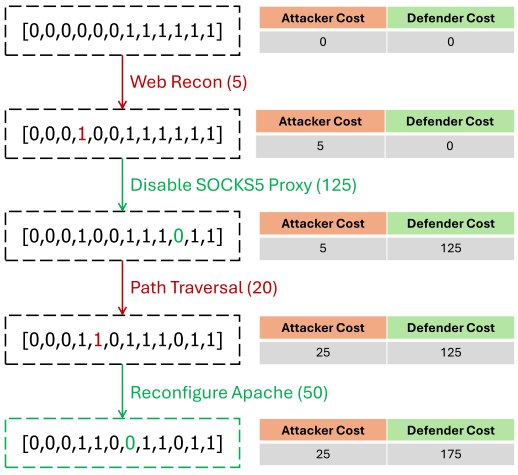

Fig. 9. Optimal strategy for the model not including time.

the defender's action `Disable SOCKS5 Proxy` is increased to 4 (`TIME-B` in the following). As usual, the initial state is the one in Table 4.

Fig. 9 shows the optimal policy for the `NO-TIME` formulation. For reasons of space, results for the others are summarized in Table 5, which compares the formulations in terms of the number of states traversed, actions performed, costs incurred by each agent, timer value (where applicable), and winning agent. As expected, the introduction of time results in a higher number of traversed states. However, it is noteworthy that changing the cost of just one action can completely alter the outcome of the game. These insights highlight the importance of designing and implementing methodologies and tools that can assist administrators in better estimating costs and times, which we leave as future work.

5. Related work

As widely discussed in other works (e.g., Nespoli et al., 2017; Shameli-Sendi et al., 2014), most of the IR approaches build upon some model to make predictions. Specifically, most of the work of the last decade is based on some *Attacker Model*, e.g. *Attack Graphs* (as in Chung et al., 2013; Foo et al., 2005; Shameli-Sendi and Dagenais, 2015; Poolsappasit et al., 2011) or *Attack Trees*, widely discussed in Section 2.1. However, while modeling the behavior and/or the possible paths that can be undertaken by an attacker could enable automated reasoning on which countermeasure to take, it does not offer any protection capability against *zero-day* attacks (Bilge and Dumitraş, 2012) for which, by definition, there are no available models. For this reason, some of the research on IR shifted to the design of an IRS based on the *Defender Model*. An analogy can be drawn with anomaly based IDSs which, instead of using signatures for the identification of known attacks, are trained to recognize when the behavior of the protected system is anomalous. Similarly, approaches based on the model of the defender characterize the normal behavior of the managed system, and the impact that any response action could have on it. To this end, several modeling techniques have been proposed, ranging from *Service Dependency Graph* (Kheir et al., 2009; Shameli-Sendi et al., 2018) to MDP (Iannucci and Abdelwahed, 2016). Despite leveraging the behavioral model of the defended system could offer an advantage against zero-day attacks (Hughes et al., 2022; Iannucci and Abdelwahed, 2016), it could offer a less specialized protection against those that are known.

For this reason, some works consider a *Joint Model* of the attacker and of the defender (An et al., 2018; Yarygina and Otterstad, 2018; Manshaei et al., 2013). In this context, several formalisms have been proposed. A wide range of works introduces extensions of the AT (Kordy et al., 2014) such as the Attack-Countermeasure Tree (Roy et al., 2010, 2012) and the Attack-Defense Tree (Kordy et al., 2011b). Moreover, aside from RRE (Zonouz et al., 2014), already discussed in Section 2.2, other works propose the AT translation into more sophisticated models to further analyze and formally verify the correctness of ATs. In Lounis and Ouchani (2021), the authors propose a stochastic framework for analyzing ATs through Continuous Time Markov Chains. The authors of Kumar et al. (2015), instead, introduce an optimization framework that translates ATs into priced timed automata. Contextually, authors propose an AT model where leaves are augmented with various parameters such as cost and time, similar to that proposed by PANACEA. Anyway, all these works are unable to exploit the long-term planning advantage offered by PANACEA.

Many works propose agent-based approaches extending a MDP. In this area, Iannucci and Abdelwahed (2018), Emami-Taba et al. (2015), Kundu and Ghosh (2014) propose stochastic game formulations to accurately model the environment and the dynamics between agents, while Zonouz et al. (2014), Tang et al. (2024), Bilinski et al. (2019) propose Stackelberg games variants. However, all these works suffer from a complex modelling phase with respect to PANACEA.

6. Future directions and research challenges

Estimating execution times and costs. Experiment E2 demonstrates that modeling execution time (and associated costs) significantly impacts system outcomes. Currently, PANACEA does not provide any mechanism to estimate these values, leaving the responsibility entirely to the system administrator. Estimating costs is a complex task that depends heavily on the specific use case and requires in-depth knowledge of the organization in question. Designing a generic model capable of producing reliable estimates across different contexts is particularly challenging, as organizations with similar IT infrastructures may have completely different business goals and consequently, very different cost structures. As for time estimation, a potential direction for future work is the development of a framework to support system administrators in estimating the execution time of actions. As discussed in Section 4.2, the idea is to leverage an emulation-based digital twin of the production infrastructure, integrating state-of-the-art tools for simulating attack and defense phases, such as Caldera (Corporation, 2024) and CybORG (Cyber operations research gym, 2022). This approach would provide system administrators with a realistic testing environment that replicates the production infrastructure, enabling them to assess vulnerabilities and implement defense strategies. After defenses are deployed, tools like Caldera can be used to build automated pipelines for triggering defense actions and monitoring their execution.

Open-source testbed. Historically, the development of sophisticated approaches to IR has been hindered by the lack of standard testbeds (Montemaggio et al., 2020). Indeed, differently from research on ID, which can rely on several reference datasets (e.g., UGR'16 Maciá-Fernández et al., 2018, CIC-IDS-2017 Sharafaldin et al., 2018), most of which were published in the last decade, research on IR must leverage a

reference system for a fair comparison of the state-of-the-art techniques. Unfortunately, the lack of such a system led researchers to make use of custom-built systems, either simulated, emulated, or real, usually with little to no critical comparison with respect to other approaches in the literature. This lack of a reference system is due to the absence of a modern, flexible, and open-source testbed. One of the most recent surveys on this topic (Ukwandu et al., 2020) reports indeed that only 3 out of the 48 reviewed testbeds (namely, INSALATA Herold et al., 2017, Softgrid Gunathilaka et al., 2016, and Arizona CWR Acwr, 2022) are released with an open-source license. However, INSALATA seems to be currently unmaintained; Softgrid has been specifically designed for the evaluation of cybersecurity solutions for power grids; Arizona CWR is instead a collection of remotely hosted capture-the-flag challenges, with an attack-oriented perspective, making it impossible to integrate it with an IRS. An interesting recent work in this direction is represented by CybORG (Cyber operations research gym, 2022), a cybersecurity environment for testing and training autonomous agents. Supporting both simulation and emulation (on Amazon Web Services cloud), CybORG provides a high-fidelity environment to verify agents' behavior on a virtual network. However, it cannot be considered a standard framework, as relying on a vendor-specific cloud can be problematic for researchers due to potential limitations on accessibility, increased costs, and the lack of flexibility to deploy the framework on alternative infrastructures.

Response selection. Several challenges remain to be addressed, among which: (i) the curse of dimensionality and (ii) the need for self-evolving approaches that can follow the non-stationary nature of computer systems. Several studies have been proposed to mitigate (i), which depends on the underlying technique used for building the response strategy. For example, approaches based on MDP could benefit from exact techniques (e.g., Hansen and Bowman, 2020; Iannucci et al., 2018), or from approximate techniques (e.g. Kocsis and Szepesvári, 2006; Iannucci et al., 2018) for state space pruning. Other metaheuristics that have been employed are genetic algorithms (e.g., Fessi et al., 2009) and Particle Swarm Optimization (e.g., Karami and Guerrero-Zapata, 2015). However, to the best of our knowledge, none of them addresses (ii). A promising technique to address both (i) and (ii) is deep reinforcement learning. This field of research received a strong development impulse with the publication of a paper in 2013, presenting a Q-Learning implementation with neural networks, known as Deep Q-Networks (DQN Mnih et al., 2013), which was able to learn how to play Atari games using visual frames as the sole input. This problem could not have been solved with traditional reinforcement learning, due to the size of the state space (256^{33600} states). For this reason, more recently, several approaches to IR based on DQN have been proposed (e.g., Iannucci et al., 2020; Zolotukhin et al., 2020), because they exploit the generalization capabilities of the underlying neural networks to avoid the need for a full state space exploration (thus addressing (i)), which is instead needed in order for traditional reinforcement learning algorithms, such as Q-Learning, to converge[3] Another characteristic of approaches based on deep reinforcement learning is that their formulation is based on a multi-armed bandit problem, which intrinsically requires some exploration step, and that in turn enables the underlying neural networks to automatically evolve, thus addressing (ii). On the other hand, the same characteristic constitutes a limitation because any form of exploration undertaken by a fully trained agent could lead to a suboptimal response strategy.

Another open problem is the design of an integrated technique for both, attack and defense modeling. This problem is manyfold, and regards aspects pertaining to, among the others: (i) techniques for the definition of the models; (ii) frameworks for the formulation of the mathematical problem; (iii) performance (e.g., execution time, scalability, memory consumption, etc.) of the mathematical tool chosen for the solution of the problem. Regarding (i), information on known attacks can be found in several online sources, such as, the Common Vulnerabilities and Exposure (CVE) database (MITRE, 2022), maintained by MITRE, and the National Vulnerability Database (NVD) (NIST, 2022), maintained by NIST. Both the databases are synchronized, and, depending on the specific vulnerability, its NVD entry could include information on how to mitigate or fix it. One open challenge is how to merge the CVE and NVD data regarding attack mitigation, which is largely unstructured, into model libraries on top of which some reasoning can be carried out. The same applies to typical system components (e.g., in the domain of enterprise systems: web servers, database servers, and so forth) for which, to the best of our knowledge, standard behavioral models do not exist. Regarding aspect (ii), several techniques could be adopted for the formulation of the mathematical decision-making problem, ranging from formulations based on static mapping (e.g, Fisch, 1996), cost-based ordering (e.g., Chen et al., 2014), to game theory (e.g., Zonouz et al., 2014), and other meta-heuristics as described above. Each of the aforementioned techniques comes with its own toolbox in terms of algorithms for solving the underlying optimization problem. The challenge is to find the proper toolbox, considering the representativeness of the model, the effectiveness of the solution of the optimization problem, and the performance of the chosen algorithm. Indeed, as of today, achieving near real-time planning time and good scalability is still an open challenge.

7. Conclusions

In this work, we proposed a new methodology, named PANACEA, to automatically generate timed competitive stochastic games from augmented ADTs. The goal of the work is to provide an easy-to-adopt framework for training and developing autonomous agents that are able to handle attack and defense phases. Starting from a well-known security framework such as attack trees, PANACEA overcomes the intrinsic challenges deriving from the complex modeling phase of MDPs, while at the same time benefiting from long-term planning. The evaluation, after illustrating insights about the scalability, presents how to apply our novel methodology to a use case, showing the effectiveness of PANACEA.

CRediT authorship contribution statement

Tommaso Caiazzi: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Stefano Iannucci:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Valerio Marini:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation; **Matteo Foschi:** Writing – original draft, Software, Methodology; **Riccardo Torlone:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Funding acquisition.

Data availability

Code and data are public. I have shared the link to the code/data at the attach file step.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Stefano Iannucci reports financial support was provided by Government of Italy. Riccardo Torlone reports financial support was provided by Government of Italy. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

[3] It is worth noting however that DQN does not guarantee convergence (Wang and Ueda, 2022). Some more recent studies (Avrachenkov et al., 2021; Wang and Ueda, 2022) address this issue and propose variants of DQN that provably converge.

Acknowledgment

This work has been partially supported by the PRIN 2022 project 2022Y45XE3 *Panacea: A Model-Based Framework for Self-Protecting Systems* funded by Ministero dell'Università e della Ricerca (MUR) and by Progetto Integrato 2.1 "Cyber Security dei sistemi energetici" funded by Ministero dell'Ambiente e della Sicurezza Energetica (MASE).

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.cose.2026.104834.

References

acwr, 2022. Arizona cyber warfare range. https://www.azcwr.org/

An, X., Lin, F., Xu, S., Miao, L., Gong, C., 2018. A novel differential game model-based intrusion response strategy in fog computing. Secur. Commun. Netw., 2018 (1) 1821804. https://doi.org/10.1155/2018/1821804

Avrachenkov, K.E., Borkar, V.S., Dolhare, H.P., Patil, K., 2021. Full gradient DQN reinforcement learning: a provably convergent scheme. In: Modern Trends in Controlled Stochastic Processes. Springer.

Bellman, R.E., 1957. Dynamic Programming. Princeton University Press.

Bilge, L., Dumitraş, T., 2012. Before we knew it: an empirical study of zero-day attacks in the real world. In: 2012 ACM Conference on Computer and Communications Security.

Bilinski, M., Ferguson-Walter, K., Fugate, S., Gabrys, R., Mauger, J., Souza, B., 2019. You only lie twice: a multi-round cyber deception game of questionable veracity. In: Alpcan, T., Vorobeychik, Y., Baras, J.S., Dán, G. (Eds.), Decision and Game Theory for Security. Springer International Publishing, Cham.

Boutilier, C., 1996. Planning, learning and coordination in multiagent decision processes. In: 6th Conference on Theoretical Aspects of Rationality and Knowledge. Morgan Kaufmann Publishers Inc.

Boyan, J.A., Littman, M.L., 2000. Exact solutions to time-dependent MDPs. Proceedings of the 14th International Conference on Neural Information Processing Systems, MIT Press. Cambridge, MA, USA. 982–988. NIPS'00.

Busoniu, L., Babuska, R., De Schutter, B., 2008. A comprehensive survey of multiagent reinforcement learning, IEEE Tran. Syst. Man Cybern. Part C Appl. Rev. 38 (2) 156–172. https://doi.org/10.1109/TSMCC.2007.913919

Chen, Q., Abdelwahed, S., Erradi, A., 2014. A model-based validated autonomic approach to self-protect computing systems. IEEE Internet Things J. 1 (5) 446–460. https://doi.org/10.1109/JIOT.2014.2349899

Chung, C.-J., Khatkar, P., Xing, T., Lee, J., Huang, D., 2013. NICE: network intrusion detection and countermeasure selection in virtual network systems. IEEE Trans. Dependable Secure Comput. https://doi.org/10.1109/TDSC.2013.8

Corporation, M., 2024. Mitre caldera: a scalable, automated adversary emulation platform. https://github.com/mitre/caldera

Cyber operations research gym, 2022. Cyber operations research gym. https://github.com/cage-challenge/CybORG. Created by Maxwell Standen, David Bowman, Son Hoang, Toby Richer, Martin Lucas, Richard Van Tassel, Phillip Vu, Mitchell Kiely, KC C., Natalie Konschnik, Joshua Collyer.

Emami-Taba, M., Amoui, M., Tahvildari, L., 2015. Strategy-aware mitigation using Markov games for dynamic application-layer attacks. In: IEEE 16th International Symposium on High Assurance Systems Engineering.

Fessi, B.A., BenAbdallah, S., Hamdi, M., Boudriga, N., 2009. A new genetic algorithm approach for intrusion response system in computer networks. In: 2009 IEEE Symposium on Computers and Communications.

Fisch, E.A., 1996. Intrusion Damage Control and Assessment: A Taxonomy and Implementation of Automated Responses to Intrusive Behavior. Ph.D. thesis. Texas A&M University.

Foo, B., Wu, Y.-S., Mao, Y.-C., Bagchi, S., Spafford, E., 2005. ADEPTS: a daptive intrusion response using attack graphs in an e-commerce environment. In: 2005 International Conference on Dependable Systems and Networks. IEEE. https://doi.org/10.1109/DSN.2005.17

Givan, R., Dean, T., Greig, M., 2003. Equivalence notions and model minimization in Markov decision processes. Artif. Intell., 147 (1) 163–223. Planning with Uncertainty and Incomplete Information. https://doi.org/10.1016/S0004-3702(02)00376-4

Gunathilaka, P., Mashima, D., Chen, B., 2016. Softgrid: a software-based smart grid testbed for evaluating substation cybersecurity solutions. In: 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy.

Hansen, E., Bowman, T., 2020. Improved vector pruning in exact algorithms for solving POMDPs. In: Conference on Uncertainty in Artificial Intelligence. PMLR.

Herold, N., Wachs, M., Dorfhuber, M., Rudolf, C., Liebald, S., Carle, G., 2017. Achieving reproducible network environments with INSALATA. In: IFIP International Conference on Autonomous Infrastructure, Management and Security. Springer, Cham.

Hughes, K., McLaughlin, K., Sezer, S., 2022. A model-free approach to intrusion response systems. J. Inf. Secur. Appl.

Iannucci, S., Abdelwahed, S., 2016. A probabilistic approach to autonomic security management, In: 13th IEEE International Conference on Autonomic Computing (ICAC), 157–166.

Iannucci, S., Abdelwahed, S., 2018. Model-based response planning strategies for autonomic intrusion protection. ACM Trans. Auton. Adapt. Syst.

Iannucci, S., Abdelwahed, S., Montemaggio, A., Hannis, M., Leonard, L., King, J., Hamilton, J., 2018. A model-integrated approach to designing self-protecting systems. IEEE Trans. Software Eng. https://doi.org/10.1109/TSE.2018.2880218

Iannucci, S., Cardellini, V., Barba, O.D., Banicescu, I., 2020. A hybrid model-free approach for the near-optimal intrusion response control of non-stationary systems. Future Gener. Comput. Syst.

Ibe, O., 2013. Markov processes for stochastic modeling. Newnes.

Karami, A., Guerrero-Zapata, M., 2015. A hybrid multiobjective RBF-PSO method for mitigating dos attacks in named data networking. Neurocomputing, 151, 1262–1282. https://doi.org/10.1016/j.neucom.2014.11.003

Kearns, M., Mansour, Y., Ng, A.Y., 2002. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Mach. Learn. https://doi.org/10.1023/A:1017932429737

Kephart, J.O., Chess, D.M., 2003. The vision of autonomic computing. IEEE Comput.

Kheir, N., Debar, H., Cuppens-Boulahia, N., Cuppens, F., Viinikka, J., 2009. Cost evaluation for intrusion response using dependency graphs. In: 2009 International Conference on Network and Service Security. IEEE.

Kocsis, L., Szepesvári, C., 2006. Bandit based Monte-Carlo planning. In: Machine Learning: ECML 2006. Springer. https://doi.org/10.1007/11871842_29

Kordy, B., Kordy, P., Mauw, S., Schweitzer, P., 2013. ADTool: security analysis with attack–defense trees. In: International Conference on Quantitative Evaluation of Systems. Springer.

Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P., 2011a. Foundations of attack–defense trees. In: Formal Aspects of Security and Trust: 7th International Workshop, FAST 2010, Pisa, Italy, September 16–17, 2010. Revised Selected Papers 7. Springer.

Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P., 2011b. Foundations of attack–defense trees. In: Degano, P., Etalle, S., Guttman, J. (Eds.), Formal Aspects of Security and Trust. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kordy, B., Pièrre-Cambacédès, L., Schweitzer, P., 2014. DAG-based attack and defense modeling: don't miss the forest for the attack trees. Comput. Sci. Rev., 13-14, 1–38. https://doi.org/10.1016/j.cosrev.2014.07.001

Kumar, R., Ruijters, E., Stoelinga, M., 2015. Quantitative attack tree analysis via priced timed automata. In: Sankaranarayanan, S., Vicario, E. (Eds.), Formal Modeling and Analysis of Timed Systems. Springer International Publishing, Cham.

Kundu, A., Ghosh, S.K., 2014. Game theoretic attack response framework for enterprise networks. In: International Conference on Distributed Computing and Internet Technology. Springer.

Kwiatkowska, M., Norman, G., Parker, D., Santos, G., 2020. PRISM-games 3.0: stochastic game verification with concurrency, equilibria and time. In: Proc. 32nd International Conference on Computer Aided Verification (CAV'20). Springer. .

de Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., et al., 2013. Software engineering for self-adaptive systems: a second research roadmap. In: Software Engineering for Self-Adaptive Systems II. Springer.

Li, L., Littman, M.L., 2008. Prioritized Sweeping Converges to the Optimal Value Function. Technical Report. Rutgers University. https://doi.org/10.7282/T3TX3JSX

Lounis, K., Ouchani, S., 2021. Modeling attack-defense trees' countermeasures using continuous time Markov chains. In: Software Engineering and Formal Methods. SEFM 2020 Collocated Workshops. Springer International Publishing, Cham.

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Therón, R., 2018. UGR '16: a new dataset for the evaluation of cyclostationarity-based network IDSs. Comput. Secur.

Manshaei, M.H., Zhu, Q., Alpcan, T., Bacşar, T., Hubaux, J.-P., 2013. Game theory meets network security and privacy. Association for Computing Machinery. New York, NY, USA. 45 (3). https://doi.org/10.1145/2480741.2480742

MITRE, 2022. Common vulnerabilities and exposures. https://cve.mitre.org/

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing Atari with deep reinforcement learning. CoRR http://arxiv.org/abs/1312.5602

Montemaggio, A., Iannucci, S., Bhowmik, T., Hamilton, J., 2020. Designing a methodological framework for the empirical evaluation of self-protecting systems. In: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion.

Nespoli, P., Papamartzivanos, D., Mármol, F.G., Kambourakis, G., 2017. Optimal countermeasures selection against cyber attacks: a comprehensive survey on reaction frameworks. IEEE Commun. Surv. Tutorials, 20, 1361–1396. https://api.semanticscholar.org/CorpusID:44093728

NIST, 2022. National vulnerability database. https://nvd.nist.gov/

Nocedal, J., Wright, S.J., 2006. Numerical Optimization. Springer.

Panacea, 2026. Panacea. https://github.com/Marini97/PANACEA

Poolsappasit, N., Dewri, R., Ray, I., 2011. Dynamic security risk management using Bayesian attack graphs. IEEE Trans. Dependable Secure Comput.

Psaier, H., Dustdar, S., 2011. A survey on self-healing systems: approaches and systems. Computing .

Roy, A., Kim, D.S., Trivedi, K.S., 2010. Act: attack countermeasure trees for information assurance analysis. In: 2010 INFOCOM IEEE Conference on Computer Communications Workshops. https://doi.org/10.1109/INFCOMW.2010.5466633

Roy, A., Kim, D.S., Trivedi, K.S., 2012. Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees. Secur. Commun. Netw. https://doi.org/10.1002/sec.299

Schneier, B., 1999. Attack trees. Dr. Dobb's J.

Shameli-Sendi, A., Cheriet, M., Hamou-Lhadj, A., 2014. Taxonomy of intrusion risk assessment and response system, Comput. Secur. 45, 1–16. https://doi.org/10.1016/j.cose.2014.04.009

Shameli-Sendi, A., Dagenais, M., 2015. ORCEF: Online response cost evaluation framework for intrusion response system. J. Netw. Comput. Appl.

Shameli-Sendi, A., Louafi, H., He, W., Cheriet, M., 2018. Dynamic optimal countermeasure selection for intrusion response system. IEEE Trans. Dependable Secure Comput. https://doi.org/10.1109/TDSC.2016.2615622

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. 4th International Conference on Information Systems Security and Privacy.

Stakhanova, N., Basu, S., Wong, J., 2007. A cost-sensitive model for preemptive intrusion response systems. In: 21st International Conference on Advanced Information Networking and Applications.

Tang, Y., Sun, J., Wang, H., Deng, J., Tong, L., Xu, W., 2024. A method of network attack-defense game and collaborative defense decision-making based on hierarchical multi-agent reinforcement learning. Comput. Secur. https://doi.org/10.1016/j.cose.2024.103871

Ukwandu, E., Farah, M. A.B., Hindy, H., Brosset, D., Kavallieros, D., Atkinson, R., Tachtatzis, C., Bures, M., Andonovic, I., Bellekens, X., 2020. A review of cyber-ranges and test-beds: current and future trends. Sensors.

Wang, Z.T., Ueda, M., 2022. A convergent and efficient deep Q network algorithm. In: International Conference on Learning Representations. https://openreview.net/forum?id=OJm3HZuj4r7

Weyns, D., 2021. An Introduction to Self-Adaptive Systems. John Wiley & Sons, Hoboken, NJ, USA.

Yarygina, T., Otterstad, C., 2018. A game of microservices: automated intrusion response. In: IFIP International Conference on Distributed Applications and Interoperable Systems. Springer.

Yuan, E., Esfahani, N., Malek, S., 2014. A systematic survey of self-protecting software systems. ACM Trans. Auton. Adapt. Syst. https://doi.org/10.1145/2555611

Zolotukhin, M., Kumar, S., Hämäläinen, T., 2020. Reinforcement learning for attack mitigation in SDN-enabled networks. In: 6th IEEE Conference on Network Softwarization.

Zonouz, S.A., Khurana, H., Sanders, W.H., Yardley, T.M., 2014. RRE: a game-theoretic intrusion response and recovery engine. IEEE Trans. Parallel Distrib. Syst. https://doi.org/10.1109/TPDS.2013.211