Time is money: A temporal model of cybersecurity*

Zoltán Ádám Mann

University of Halle-Wittenberg, Halle, Germany

Abstract. Models of possible attacks and defenses can help assess and improve an organization's cybersecurity posture. However, existing models for analyzing cybersecurity fail to adequately model the temporal aspects of attacks and defenses. For example, game-theoretic models typically assume that the players (here, the attacker and the defender) take turns, and the effect of an action realizes immediately. In reality, attacker and defender actions may start asynchronously at any time and may have very different duration. This has important consequences. For example, whether the defender can successfully thwart an attack may depend on whether the defender can finish implementing a mitigation action before the attack would reach its goal.
This paper proposes a new, formal model of cybersecurity attacks and defense, with a special focus on the time dimension. The model allows reasoning about temporal aspects, such as how the duration of defensive actions impacts their ability to prevent attacks. Based on the proposed formal model, we develop a simulator that supports easy experimentation and what-if analysis with different attack and defense strategies.

Keywords: cybersecurity, security model, game theory, cyberattack

1 Introduction

In a globally connected digital ecosystem, any organization is potentially exposed to cyberattacks from all over the world [13]. Cyberattacks can cause huge harm, from financial losses to reputational damage to loss of human lives. Therefore, effective defense against cyberattacks has become a major priority [1]. Frequent news about successful cyberattacks show that our current defense practices are insufficient, highlighting the need for more research into effective ways for defending against cyberattacks. Effective defense requires the timely detection and mitigation of cybersecurity risks, making automation necessary in the detection and mitigation of such risks [8]. To enable effective machine reasoning on security risks, machine-readable formal models are needed that capture the relevant information about IT systems [9]. In recent years, many different models have been

* This paper was published in the *Proceedings of the 40th IFIP International Conference on ICT Systems Security and Privacy Protection (IFIP SEC)*, pp. 82-96, 2025

devised for cybersecurity risks. Popular families of security models include variants of attack graphs, stochastic (Bayesian, Markovian etc.) models, and game theoretical models [6]. A common deficiency in existing cybersecurity models is the lack of appropriately capturing time. Security models typically have either no explicit notion of time, or only operate with *logical time*, i.e., the notion of events succeeding each other (e.g., an attack moving from one state to the next). What is typically missing is the consideration of *physical time*, i.e., the actual duration of actions (e.g., how long it takes to perform an attack).

Capturing physical time and, in particular, the duration of attacks and of mitigation actions is very important [7]. For example, whether an attack can be foiled could depend on when exactly the attack is detected and whether a mitigation action can be implemented before the attack finishes [16]. Most existing security models do not capture the duration of attack and defense actions and thus do not support reasoning about the implications of these durations.

To address this problem, this paper introduces TIM (Time Is Money), a new formal security model[1] that captures physical time. TIM combines features from several classes of traditional security models. Similar to attack graphs, TIM works on a graph structure, modeling how attacks propagate from node to node in the network. As in stochastic models, TIM features probabilistic events. Inspired by game theoretic models, TIM casts cybersecurity as a game between a defender and a set of attackers. This combination of features makes TIM comprehensive and particularly powerful in modeling cybersecurity.

A formal security model can be used in many ways, e.g., as a sound basis for *simulating* the interplay of attack and defense actions in a system, captured as a model with the given modeling constructs. We propose a simulator called simTIM to simulate TIM models. Performing such simulations and analyzing the results can help identify security issues and improve the security of the system.

The contributions of this paper are as follows. 1. We describe TIM as a formal model, defining the formalisms used to express different aspects relevant for cybersecurity, including physical time. 2. We demonstrate with a running example how TIM can be used to model different elements of realistic cyberattacks and mitigation actions. 3. To operationalize TIM, we present a simulator called simTIM that can simulate attacks and defenses captured as a TIM model.

TIM can become a solid basis for an ecosystem of powerful cybersecurity analysis tools, using physical time in the modeling of attacks and defenses.

2 Requirements

Based on an analysis of typical patterns in cyberattacks and defenses, we first identified a set of important aspects. Our aim is to develop a formal model that incorporates these aspects. This leads to the following list of requirements that a good model for cybersecurity should satisfy:

[1] TIM is a *metamodel*, i.e., a model that defines the constructs that other, more specific models can use. To simplify the language, we refer to TIM as a model.

R1: Systems consist of nodes and links, through which attacks can propagate
R2: Nodes may exhibit different vulnerabilities and host assets of different value
R3: A set of attackers tries to attack the system that a defender tries to defend
R4: Attacks may comprise multiple steps, which can be either local, expanding the attacker's access within a node, or compromising further nodes
R5: Attackers only know the publicly exposed parts of the system initially, but may discover further parts through their activities
R6: Ongoing attacks are not known to the defender, but some adversarial activities might be detected (e.g., by an intrusion detection system)
R7: The defender can implement mitigation actions, which might preclude or thwart certain attacks
R8: Both defender and attacker actions take time and are associated with costs
R9: The defender has finite resources, posing a limitation on the number of defensive actions that can be run in parallel

3 Related work

While several models of cybersecurity have been proposed, few of them incorporate some notion of time. Here we review papers that do incorporate time, and we summarize in Table 1 to what extent they satisfy the requirements of Sec. 2.

Models widely used in cybersecurity, such as attack graphs [14], the Cyber Kill Chain[2], or MITRE ATT&CK[3] focus on the succession of steps, without an explicit notion of their duration. Some researchers have extended these models with some relevant notion of time. Hoffmann [5] proposes a stochastic extension to the Cyber Kill Chain, which allows reasoning about the duration of each phase of the chain, as well as the whole duration of the chain. However, that paper does not take into account many important aspects of cybersecurity, such as the network topology and nodes' different characteristics. Outkin et al. [11] proposes a Markovian model based on MITRE ATT&CK, in which the duration of attack steps can be estimated. However, that model ignores the duration of defensive actions, although that aspect can substantially impact the outcomes.

Game theory is often used to model cybersecurity. In the seminal FlipIt approach [12], attacker and defender compete to take control over a resource. The players' aim is to maximize the fraction of time in which they are in control of the resource. FlipIt oversimplifies several aspects of cybersecurity, e.g., it does not support multi-step attacks. Also its handling of time is simplistic, as players' actions are assumed to take effect immediately. Several extensions have been proposed for FlipIt, some of which also extend its handling of time. Merlevede et al. [10] suggest an extension that takes into account the time when gains and damages are realized by discounting future gains and damages. However, that work also suffers from most of FlipIt's oversimplification. Zhang et al. [15] develop a model in which an attacker tries to infect and destruct nodes in a network, while the defender tries to repair infected nodes. Differential equations govern the

[2] https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html
[3] https://attack.mitre.org

Table 1. Overview of the extent to which existing approaches satisfy the requirements of Sec. 2. ✓: requirement is satisfied; (✓): partially satisfied; –: not at all satisfied

Work	R1	R2	R3	R4	R5	R6	R7	R8	R9
Clempner [3]	–	–	✓	–	–	–	✓	(✓)	✓
Farhang & Grossklags [4]	–	–	(✓)	–	–	✓	(✓)	✓	–
Hoffmann [5]	–	–	–	(✓)	(✓)	–	–	(✓)	–
Merlevede et al. [10]	–	–	(✓)	–	–	–	(✓)	(✓)	–
Outkin et al. [11]	–	–	(✓)	(✓)	–	✓	✓	(✓)	–
van Dijk et al. [12]	–	–	(✓)	–	–	–	(✓)	(✓)	–
Zhang et al. [15]	✓	–	(✓)	(✓)	–	–	(✓)	(✓)	–
Żychowski & Mańdziuk [17]	✓	–	(✓)	(✓)	✓	–	(✓)	(✓)	–
TIM (this work)	✓	✓	✓	✓	✓	✓	✓	✓	✓

temporal development of the number of infected, repaired, and destructed nodes. However, the model misses important characteristics of real-world cybersecurity, such as the different vulnerabilities that nodes may exhibit or the unobservability of some nodes and actions. To model and analyze security, Clempner [3] uses Stackelberg Security Games, defined as a continuous-time Markov process, working with physical time. Using reinforcement learning, the approach can dynamically adapt to changes in the environment. However, the approach misses important characteristics of cybersecurity, such as unobservability and vulnerabilities. Żychowski and Mańdziuk [17] work with sequential Stackelberg Security Games, where the defender is assumed to follow a statically chosen strategy which is known to the attacker – an assumption that significantly limits the practical relevance of such models. The authors show how their approach can be applied to an extended version of FlipIt with multiple resources. However, action durations are not modelled. Farhang and Grossklags [4] explicitly consider the time needed by the attacker to carry out an attack and the time needed by the defender to detect and react to attacks. However, their model misses important aspects, such as the propagation of attacks through different nodes in a network. Also, their work makes unrealistic assumptions, e.g., that the attacker performs actions periodically, with a fixed and known periodicity.

4 Proposed model: TIM

In the following subsections, we describe the different constructs of TIM, illustrate them on a running example, and show how they satisfy the above requirements. Table 2 gives an overview of the used mathematical notation.

4.1 System, nodes, links, properties

The considered system S is represented as a directed graph $S = (N, L)$. N is a finite set of *nodes*. Nodes may represent any tangible or intangible assets, such as servers, routers, or software application components. L is a finite set of directed *links*. Each link connects two nodes. The notation $n_1 n_2 \in L$ is used to denote

Table 2. Notation overview

Notation	Description
N / L	Set of all nodes / links in the system
Π	Set of all properties that nodes may have
R_π	Set of possible values of property π
$\pi(n)$	Value of property π for node n
$\widehat{\Pi}$	Set of possible property assignments for a node
$\widehat{\pi}(n)$	List of property assignments of node n
X / $X^{(\text{att})}$	Set of all actors / attackers
$\Omega^{(\text{node})}$ / $\Omega^{(\text{link})}$	Set of access options for nodes / links
$\omega_x(y)$	Access of actor x to node or link y
$\mathcal{A}$	Set of all possible actions
$\mathcal{A}^{(\text{node})}$ / $\mathcal{A}^{(\text{link})}$	Set of all possible node actions / link actions
$\mathcal{A}(x)$	Set of possible actions that actor x may initiate
φ_a / ψ_a	Precondition / postcondition of action a
c_a / d_a / p_a	Cost / duration / success probability of action a
A	Actions being executed
$\mathcal{A}^{(\text{attack})}$	Set of all possible actions of attackers
$\varrho(a, \widehat{\pi}(n))$	Probability that action a applied to node n is detected
$D(a, \widehat{\pi})$ / $G(a, \widehat{\pi})$	One-off damage / gain of action a on a node with properties $\widehat{\pi}$
$\delta(\omega, \widehat{\pi})$ / $\gamma(\omega, \widehat{\pi})$	Damage / gain per time from a node with access ω and properties $\widehat{\pi}$
$D^{\text{total}}_{[0,T]}$	Total damage incurred in the $[0, T]$ time interval
$G_{[0,T]}(x)$	Gain of attacker x in the $[0, T]$ time interval
$C_{x,[0,T]}$	Cost incurred by actor x's actions in the $[0, T]$ time interval
M_x	Maximum number of concurrent actions for actor x

a link from node $n_1 \in N$ to node $n_2 \in N$. Π is a finite set of node *properties*. Every property $\pi \in \Pi$ is a function $\pi : N \to R_\pi$, where R_π is the set of possible values of the property. For a property $\pi \in \Pi$ and a node $n \in N$, $\pi(n) \in R_\pi$ is the value of the property for the given node. Every R_π contains the special element undef; $\pi(n) = \mathsf{undef}$ means that π is not defined for n. Let $\widehat{\Pi} = \times(R_\pi : \pi \in \Pi)$ denote the set of possible assignments of properties for a node. For a node $n \in N$, $\widehat{\pi}(n) = (\pi(n) : \pi \in \Pi) \in \widehat{\Pi}$ is the list of property assignments of n.

Running example. Fig. 1 shows a simple example system, comprising two nodes connected by a link. Node n_1 is a web server, running version 5.7.0 of the Apache Tapestry web application framework (https://tapestry.apache.org/),

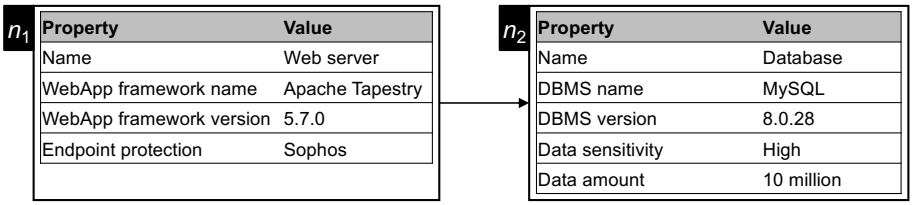

Fig. 1. Example system. Undefined properties are not shown

and using Sophos endpoint protection software (https://www.sophos.com/en-us/products/endpoint-antivirus). Node n_2 is a database server, running Oracle MySQL (https://www.mysql.com/) database management system (DBMS), version 8.0.28. The database contains 10 million sensitive data records. This is a simple example of a typical two-tier system, in which the web application running on node n_1 uses data from the database on n_2 to serve client queries.

Link to requirements. Nodes and links can be used to satisfy **R1**, while properties can be used to satisfy **R2**.

4.2 Actors and their access to nodes and links

X denotes the set of actors that may perform actions on the system. X consists of a defender, denoted as $\mathsf{defender} \in X$, and 0 or more attackers forming the set $X^{(\mathrm{att})} = X \setminus \{\mathsf{defender}\}$. Different actors may have different possibilities for accessing different parts of the system. For an actor $x \in X$, this is formalized as follows. For a node $n \in N$, let $\omega_x(n) \in \Omega^{(\mathrm{node})} = \{\mathsf{none}, \mathsf{visible}, \mathsf{user}, \mathsf{admin}\}$ denote the access of actor x to node n. The option none means that the actor is not aware of the given node, while $\mathsf{visible}$ means that the actor knows basic information, such as the IP address, of the node, allowing the actor to, for example, send ping requests to the node. The options user and admin mean that the actor has user-level and administrator-level access to the node, respectively. For a link $l \in L$, let $\omega_x(l) \in \Omega^{(\mathrm{link})} = \{\mathsf{none}, \mathsf{visible}\}$ denote the access of actor x to link l. Here, none means that the actor is unaware of the link, while $\mathsf{visible}$ means that the actor is aware of the link.

Running example. Let $X = \{\mathsf{defender}, \mathsf{attacker}\}$. Node n_1 is exposed over the internet, and thus is visible to the attacker ($\omega_{\mathsf{attacker}}(n_1) = \mathsf{visible}$), but the attacker has no access to the other node ($\omega_{\mathsf{attacker}}(n_2) = \mathsf{none}$) or the link ($\omega_{\mathsf{attacker}}(n_1 n_2) = \mathsf{none}$). The defender is assumed to have the highest level of access to both nodes ($\omega_{\mathsf{defender}}(n_1) = \omega_{\mathsf{defender}}(n_2) = \mathsf{admin}$) and to the link ($\omega_{\mathsf{defender}}(n_1 n_2) = \mathsf{visible}$).

Link to requirements. The set of actors can be used to model the defender and the attackers, as required by **R3**. The actors' access to nodes can be modeled through the $\omega_x(n)$ values, thus satisfying **R5**.

4.3 Possible actions

$\mathcal{A}$ is the set of possible actions. There are two types of actions: node actions and link actions. Thus, $\mathcal{A} = \mathcal{A}^{(\mathrm{node})} \cup \mathcal{A}^{(\mathrm{link})}$, where $\mathcal{A}^{(\mathrm{node})}$ is the set of possible node actions, $\mathcal{A}^{(\mathrm{link})}$ is the set of possible link actions, and $\mathcal{A}^{(\mathrm{node})} \cap \mathcal{A}^{(\mathrm{link})} = \emptyset$. For an actor $x \in X$, $\mathcal{A}(x) \subseteq \mathcal{A}$ denotes the set of actions that x may initiate.

An action $a \in \mathcal{A}$ is represented as a tuple $a = (\varphi_a, \psi_a, c_a, d_a, p_a)$. φ_a is the precondition, which is an SMT (satisfiability modulo theories) formula [2] on the set V_{φ_a} of variables. For a node action, V_{φ_a} contains the properties of the node and the actor's access to the node. For a link action, V_{φ_a} contains the properties of the start and end node of the link and the actor's access to the start and end node of the link. ψ_a is the postcondition, consisting of a set of assignments. Each

assignment assigns to a variable in V_{ψ_a} a value from its respective domain. For a node action, V_{ψ_a} contains the properties of the node and the actor's access to the node. For a link action, V_{ψ_a} contains the properties of the end node of the link and the actor's access to the end node of the link. $c_a \in \mathbb{R}_{\geq 0}$ is the cost of performing action a, $d_a \in \mathbb{R}_{\geq 0}$ is the time it takes to perform action a, and $p_a \in [0, 1]$ is the success probability of action a. The next subsection explains how actions are executed, which also defines the semantics of the above elements.

Running example. Apache Tapestry version 5.7.0 is affected by an unauthenticated remote code execution vulnerability[4]. A potential attack exploiting this vulnerability to gain administrator access to a web server is formalized as an action $a_{\mathsf{Tapestry}} \in \mathcal{A}^{(\text{node})} \cap \mathcal{A}(\mathsf{attacker})$. The pre- and postconditions are:

$$\begin{aligned}\varphi_{a_{\mathsf{Tapestry}}} = (&\text{“WebApp framework name”} = \text{“Apache Tapestry”} \wedge \\ &\text{“WebApp framework version”} \in \{5.4.5, 5.5.0, 5.6.2, 5.7.0\} \wedge \\ &\omega_x(n) \neq \mathsf{none}), \\ \psi_{a_{\mathsf{Tapestry}}} = (&\omega_x(n) = \mathsf{admin}).\end{aligned}$$

Performing this attack incurs costs of $c_{a_{\mathsf{Tapestry}}} = 300$ USD, takes about $d_{a_{\mathsf{Tapestry}}} =$ 1 day, and has a success probability of $p_{a_{\mathsf{Tapestry}}} = 0.2$.

Link to requirements. In line with **R4**, an attack can be mapped to one or more actions in TIM. Node actions can model privilege escalation. Link actions can model compromising further nodes, allowing the propagation of attacks (**R1**). Vulnerabilities (**R2**) are mapped to preconditions of attack actions, as shown in the example above. The c_a and d_a attributes of an action model its costs and duration, satisfying **R8**.

4.4 Execution of actions

A is the set of actions that have already started but not finished yet. Elements in A are tuples $(x, a, y, t_{\text{start}})$, where $x \in X$ is the actor that started the action, $a \in \mathcal{A}(x)$ is a possible action of x, $y \in N \cup L$ is the node or link to which the action is applied (if $a \in \mathcal{A}^{(\text{node})}$, then $y \in N$; otherwise, $y \in L$), and $t_{\text{start}} \in \mathbb{R}_{\geq 0}$ is the time when the action was started. At time $t \in \mathbb{R}_{\geq 0}$, the action is in A if and only if it has already been started but not finished yet, i.e., $t_{\text{start}} \leq t < t_{\text{start}} + d_a$.

At any point in time $t \in \mathbb{R}_{\geq 0}$, any actor $x \in X$ can select an action $a \in \mathcal{A}(x)$, and initiate the execution of a, provided that the necessary conditions are met (see below). The actor also specifies the node (if a is a node action) or the link (if a is a link action) to which the action should be applied. The actor's access to the given node or link must not be none. The execution starts if the precondition φ_a of a is met. If a is a node action that should be applied to node n, then this means that φ_a is true for the properties of n and x's access to n. On the other hand, if a is a link action that should be applied to link $n_1 n_2$, then this means that φ_a is true for the properties of n_1 and n_2 and x's access to n_1 and n_2.

If the precondition is met, the tuple $(x, a, y, t_{\text{start}})$ is put in A, where x is the actor that started the action, a is the action, y is the node (if a is a node

[4] https://nvd.nist.gov/vuln/detail/CVE-2021-27850

action) or link (if a is a link action) to which the action is applied, and t_{start} is the starting time of the action. In addition, the cost c_a is incurred for actor x.

If at any time in the $[t_{\text{start}}, t_{\text{start}} + d_a)$ interval, the value of φ_a becomes false, then the action aborts, and is removed from A. Otherwise, at time $t_{\text{start}} + d_a$, the tuple $(x, a, y, t_{\text{start}})$ is removed from A. With probability p_a, the action successfully terminates; with probability $1 - p_a$, the action terminates without success. If the action successfully terminates, then the postcondition ψ_a is applied. That is, if a is a node action applied to node $n \in N$, then properties of n and x's access to n are set according to ψ_a. If a is a link action applied to link $n_1 n_2 \in L$, then properties of n_2 and x's access to n_2 are set according to ψ_a.

Running example. The attacker can apply action a_{Tapestry} to node n_1. Since the precondition is fulfilled, the action can start. If the action successfully terminates, the attacker's access to node n_1 is elevated from visible to admin.

Link to requirements. The execution of actions satisfies **R4**, as well as the propagation of attacks stipulated by **R1**. Executing actions incurs costs and takes a predefined amount of time, satisfying **R8**. In line with **R7**, the defender may execute actions that change the properties of nodes in such a way that the precondition of some attacker actions is not fulfilled, thus precluding future attacks or interrupting ongoing attacks.

4.5 Detection of malicious activities

An action may have consequences on the system – e.g., changing an attribute of a node – that other actors may notice if they have access to the affected part of the system. However, we assume that the fact that an action a is being performed by an actor is not visible to other actors. (An actor with administrator privileges might be able to observe some activities of other actors, but may still not be able to tell if those activities are part of an attack / defense action a. For example, the administrator may be able to see that a user runs a script called "script.sh", without being able to tell that this script is performing a specific attack.) The only exception is if the defender has put specific monitoring measures (e.g., an intrusion detection system) in place to detect malicious actions.

Let $\mathcal{A}^{(\text{attack})} = \cup\{\mathcal{A}(x) : x \in X^{(\text{att})}\}$ denote the set of possible actions of attackers. Let $\varrho : \mathcal{A}^{(\text{attack})} \times \widehat{\Pi} \to [0, 1]$ denote the *detection probability function*. Let $x \in X^{(\text{att})}$ be an attacker that starts an action $a \in \mathcal{A}(x)$ at time t_{start}, applying it to a node $n \in N$ or to a link $n'n \in L$. In both cases, $\varrho(a, \widehat{\pi}(n))$ is the probability that the defender detects sometime between t_{start} and $t_{\text{start}} + d_a$ that a is being executed. If the execution of the action is detected, the detection time follows a random distribution with cumulative distribution function F_a with $F_a(0) = 0$ and $F_a(1) = 1$: for any $0 \leq t \leq d_a$, the probability that the execution of the action is detected by $t_{\text{start}} + t$ is $F_a(t/d_a) \cdot \varrho(a, \widehat{\pi}(n))$.

Running example. While the a_{Tapestry} attack is executing on node n_1, the Sophos endpoint protection software may be able to detect the suspicious activity. $\varrho(a_{\mathsf{Tapestry}}, \widehat{\pi}(n_1)) = 0.8$, meaning that the probability of detecting the attack sometime during its execution is 80%. Note that the detection probability depends not only on the attack, but also on properties of the node that the attack

is performed on – in this example, the fact that Sophos endpoint protection is deployed to the node has major influence on the detection probability.

Link to requirements. In accordance with **R6**, ongoing attacks are generally unknown to the defender. Nevertheless, the defender might succeed in detecting some activities of the attacker (from which the defender might potentially be able to infer some of the aims pursued by the attacker).

4.6 Dynamic behavior

At time $t = 0$, the model starts with an initial set of nodes, links, property assignments, actors, access of actors to nodes and links, possible actions, and with $A = \emptyset$. As time goes by, all these may change, either because of an action of an actor, or something else. E.g., a node may disappear because of a hardware failure, a new attacker may appear, or a new vulnerability is discovered, leading to a new possible action. If necessary for the sake of clarity, we put the time in a left-side subscript to denote the value of a variable at that point in time. E.g., ${}_tN$ denotes the set of nodes at time t. We omit this subscript when no distinction between different points in time is needed.

At any time, actors can execute actions as described in Sec. 4.4. Actors are free to choose when they execute actions. They may do this in a time-driven manner (e.g., deciding periodically what to do) or in an event-driven manner (e.g., starting the next action when the previous action finished).

By default, actors do not know about each other's actions. They may see the results of other actors' actions through changes in the system, e.g., through changed properties. The defender may detect attackers' activities through the mechanism described in Sec. 4.5. If there are multiple attackers, they might communicate with each other through mechanisms outside the model.

A TIM model can operate for an arbitrarily long time. It may or may not reach a stable state in which no actor can, or is willing to, perform any further actions. Since the execution of actions and the detection of attacks is probabilistic, the model's operation is in general not deterministic.

Running example. We show two possible scenarios. In the first scenario, the attacker first gains administrator access to n_1 using the a_{Tapestry} attack. Using the administrator access to node n_1, the attacker can perform a second node action on n_1, a port scan (or simply using a network tool like netstat), leading to the discovery of link n_1n_2 and of node n_2. Using a vulnerability of the MySQL database[5], the attacker can go on to compromise node n_2 remotely from n_1 using a link action, gaining administrator access to n_2 and thus also to the sensitive data stored in the database on n_2.

In the second scenario, the defender detects the attack on node n_1 as described in Sec. 4.5. Performing a vulnerability scan and realizing that n_2 (along with the sensitive data stored on that node) is at risk, the defender applies the node action of upgrading MySQL on n_2 to a newer version. This precludes the attacker from compromising n_2.

[5] https://nvd.nist.gov/vuln/detail/CVE-2022-21457

Link to requirements. The dynamic behavior of TIM is related to many of the requirements. Especially the propagation of attacks, starting from exposed endpoints, through multiple steps toward other nodes (**R4** and **R5**) is clearly linked, as exemplified by the first scenario above. The defender's mitigation possibilities (**R7**) are also related and illustrated by the second scenario.

4.7 Optimization objectives

A successful attack leads to damage for the defender and a gain for the attacker. These two may differ. E.g., in a successful ransomware attack, the attacker's gain is the ransom, while the damage for the defender may include, beyond the paid ransom, also the costs for managing the situation, for upholding operations with alternative means, for recovering from the incident, and reputational damage.

In terms of the temporal characteristics of damage/gain, we differentiate between two behaviors, as in [8]. *One-off*: the successful attack leads to one-time damage/gain. E.g., a data breach leads to a one-off fine for the defender and a one-off income for the attacker from selling the stolen data. *Time-proportional*: the successful attack leads to a misbehavior, and damage/gain is incurred as long as the misbehavior persists. E.g., a denial-of-service attack makes a service unavailable; the longer the service is unavailable, the higher the damage/gain.

To model the different types of damage and gain, we introduce four functions. $D, G : \mathcal{A}^{(\text{attack})} \times \widehat{\Pi} \to \mathbb{R}_{\geq 0}$ capture one-off damage and gain, respectively, while $\delta, \gamma : \Omega^{(\text{node})} \times \widehat{\Pi} \to \mathbb{R}_{\geq 0}$ capture time-proportional marginal damage and gain, respectively. For an attacker $x \in X^{(\text{att})}$, action $a \in \mathcal{A}(x)$, and a node $n \in N$, $D(a, \widehat{\pi}(n))$ is the amount of one-off damage that x generates by successfully performing attack a on node n, given n's properties $\widehat{\pi}(n)$. Similarly, $G(a, \widehat{\pi}(n))$ is the one-off gain of x from successfully performing attack a on node n. For time-proportional damage/gain, $\delta(\omega_x(n), \widehat{\pi}(n))$ is the amount of damage that x generates via node n per unit time, and $\gamma(\omega_x(n), \widehat{\pi}(n))$ is the gain of x from n per unit time. Note that δ and γ do not depend on a specific attack, but rather on the situation – the attacker's access to the node and the properties of the node – resulting from one or more attack(s).

The total damage created by all attackers in the whole system in the $[0, T]$ time interval is computed as follows. Let $0 < t_1 < t_2 < \ldots < t_k < T$ be the points in time at which at least one property of at least one node or the access of at least one attacker to at least one node changed. In addition, let $t_0 = 0$ and $t_{k+1} = T$. For $0 \leq i \leq k$, the properties of any node as well as the access of any actor to the node remain constant in the $[t_i, t_{i+1})$ interval, which we denote by ${}_{t_i}\widehat{\pi}(n)$ and ${}_{t_i}\omega_x(n)$, respectively. Let $0 < \tau_{x,1} \leq \tau_{x,2} \leq \ldots \leq \tau_{x,\ell_x} \leq T$ be the points in time at which attacker x successfully finished an attack, let $a_{x,j}$ be the attack that x successfully finished at time $\tau_{x,j}$ and let $n_{x,j}$ be the node on which

that attack successfully finished. Then, the total damage in $[0,T]$ is

$$D^{\text{total}}_{[0,T]} = \sum_{n\in N}\sum_{x\in X^{(\text{att})}}\sum_{i=0}^{k}\delta\left({}_{t_i}\omega_x(n), {}_{t_i}\widehat{\pi}(n)\right)\cdot(t_{i+1}-t_i) + \\ + \sum_{x\in X^{(\text{att})}}\sum_{j=1}^{\ell_x} D\left(a_{x,j}, {}_{\tau_{x,j}}\widehat{\pi}(n_{x,j})\right).$$

Similarly, the gain of attacker $x \in X^{(\text{att})}$ in $[0,T]$ is

$$G_{[0,T]}(x) = \sum_{n\in N}\sum_{i=0}^{k}\gamma\left({}_{t_i}\omega_x(n), {}_{t_i}\widehat{\pi}(n)\right)\cdot(t_{i+1}-t_i) + \sum_{j=1}^{\ell_x} G\left({}_{\tau_j}\omega_x(n), {}_{\tau_j}\widehat{\pi}(n_{x,j})\right).$$

For an actor $x \in X$, the cost incurred by x's actions in the $[0,T]$ time interval is $C_{x,[0,T]} = \sum_{a\in A_{x,[0,T]}} c_a$, where $A_{x,[0,T]}$ is the multiset[6] of actions initiated by x in the $[0,T]$ time interval.

The objective of the defender is to minimize $D^{\text{total}}_{[0,T]} + C_{\mathsf{defender},[0,T]}$. The objective of an attacker $x \in X^{(\text{att})}$ is to maximize $G_{[0,T]}(x) - C_{x,[0,T]}$.

Running example. In the first scenario described in Sec. 4.6, assume that the attacker executes the a_{Tapestry} attack twice (the first attack is unsuccessful), thus compromising node n_1 (i.e., obtaining admin access to n_1) after two days and incurring 600 USD costs. After 5 further days, the attacker manages to compromise also n_2, incurring further 800 USD for attack execution. Compromising n_1 leads to neither damage nor gain. However, compromising n_2 that stores large amounts of sensitive data allows the attacker to gain a one-off ransom of 100,000 USD. The defender incurs no costs for action execution, but suffers a one-off damage of 150,000 USD (paying the ransom plus additional recovery costs). Thus, the overall balance is $100{,}000 - 600 - 800 = +98{,}600$ USD for the attacker and $-150{,}000$ USD for the defender.

In the second scenario, the attacker again incurs 600 USD costs in compromising n_1. There is no vulnerability on n_2 to exploit, thus no further costs are incurred for the attacker, and there is also no gain, leading to an overall balance of -600 USD. For the defender, upgrading MySQL costs 500 USD, and there are no further costs nor damages. Thus, the defender's balance is -500 USD.

Link to requirements. The optimization objectives drive attacker and defender behavior, as required by **R3** and **R7**. The cost of actions, as stipulated by **R8**, is taken into account by both attacker and defender objectives.

4.8 Capacity constraints

For an actor $x_0 \in X$, let $M_{x_0} \in \mathbb{R}^+ \cup \{\infty\}$ be the maximum number of concurrent actions that x_0 can execute. If $M_{x_0} = \infty$, then there is no constraint on the number of concurrent actions that x_0 can execute. Otherwise, the constraint $|\{(x,a,y,t_{\text{start}}) \in A : x = x_0\}| \leq M_{x_0}$ must hold at all times. When an actor tries to start a new action (cf. Sec. 4.4), this condition is also checked, and the action can only be started if it does not lead to a violation of this constraint.

[6] If an action was invoked by the actor m times, then it occurs m times in this multiset.

Running example. We make the conservative assumption that the attacker has unlimited resources ($M_{\mathsf{attacker}} = \infty$), while the defender's capacity is strictly limited ($M_{\mathsf{defender}} = 1$). This assumption does not impact the scenarios described before, as it does not constrain the attacker, and the defender only performs a single action. It should be noted though that while the defender is busy upgrading MySQL, it would not be able to deal with other risks, should anything else arise.

Link to requirements. The construct of capacity constraints satisfies **R9**.

5 Simulation

We present simTIM, a proof-of-concept implementation of a simulator to simulate TIM models. simTIM is written in Python and is publicly available[7].

Some parts of the implementation are straight-forward. Nodes, links, actors, and actions are implemented by corresponding classes in simTIM's object-oriented design. Specific action types are sub-classes of an abstract Action class. Access types for nodes and links are implemented as Enum classes. Node properties are implemented as dictionaries with key–value pairs of type string.

The challenging part is the implementation of the temporal behavior of TIM. TIM actors could start an action at any time; also the detection of an attack could happen at any time. To create a manageable implementation of this behavior, we use the concept of discrete-event simulation. The finish of an action is considered an event. The currently running (i.e., started but not yet finished) actions are kept in a queue, sorted according to their finish time. When an action of actor x finishes, x is notified about this (also including whether the action finished successfully or not), allowing x to start new actions in an event-driven manner. This makes sense because an actor can only observe their own actions, not the actions of other actors. To also allow time-triggered actions, simTIM introduces a special action called WaitAction, which finishes successfully with probability 1 after a given amount of time. Just as with any other action, when a WaitAction of actor x finishes, x is notified about this, allowing x to start new actions. This way, an actor can schedule actions in a time-driven way, e.g., at regular times, or at a specified amount of time after another action finished.

Also the detection of attacks (Sec. 4.5), is implemented using a special action type called DetectAction. When an attack action is launched, simTIM decides probabilistically, according to the detection probability $\varrho(a, \widehat{\pi}(n))$, if the attack will be detected. If yes, the cumulative distribution function F_a is used to determine randomly after how much time the action will be detected. If the attack is to be detected in t time, simTIM creates a DetectAction belonging to the defender, which finishes successfully with probability 1 after t time. When this happens, the defender is notified about the attack, in accordance with the behavior described in Sec. 4.5.

Different attacker and defender strategies can be implemented in simTIM by specifying what new actions the actors start upon the finish of various actions

[7] https://github.com/zoltanmann/simTIM

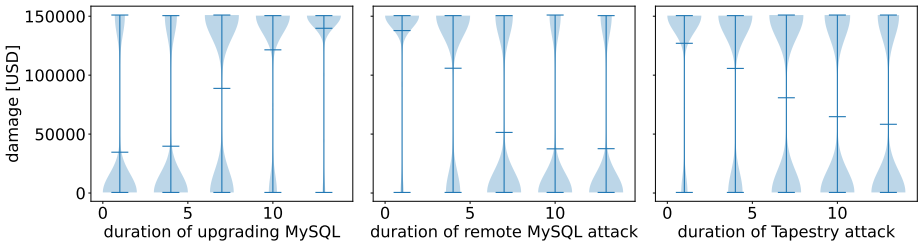

Fig. 2. Violin plots showing the distribution and the mean of the damage, depending on the duration of various actions (in hours). The default duration is 4 hours each for the two attack actions and 8 hours for the defensive action.

(including the finish of a WaitAction and, for the defender, of a DetectAction). Since the simulation is probabilistic, it makes sense to re-run it several times and collect statistics of the resulting metrics (see Sec. 4.7) to assess how successful the attacker and defender strategies are. For example, we implement the strategies sketched in Sec. 4.6. The attacker tries to first perform the Tapestry attack on n_1, then a port scan to find n_2, and finally the MySQL attack to compromise n_2 and gain access to the sensitive data. If the defender detects the Tapestry attack on n_1, it tries to upgrade MySQL on n_2 to protect the sensitive data. The resulting distribution of the damage, for different durations of various actions, is shown in Fig. 2. As shown in the first plot, when the defensive action of upgrading MySQL is fast (taking 1 to 4 hours), the defender can thwart the attack in most cases; but as the defensive action becomes more time-consuming, the attack succeeds more and more often, leading to much higher average damage. The two other plots show that increasing the duration of the attack actions significantly lowers the probability of a successful attack and thus also the average damage. The effect of increasing the duration of the Tapestry attack is less pronounced in this regard than that of increasing the duration of the remote MySQL attack; this is because increasing the duration of the Tapestry attack also tends to delay the detection of the attack by the defender. These results show once more the importance of the time dimension in cybersecurity.

6 Conclusions and future work

Existing cybersecurity models do not offer adequate support for reasoning about time, although the duration of attacker and defender actions may have significant impact. This paper introduced TIM, the first model to remedy this problem. TIM combines a game-theoretic model of attacker and defender actions with a model of the network to be defended. TIM can model the timing implications of multi-step attacks, including both lateral movement across nodes and privilege escalation within nodes. As shown in Table 1, TIM satisfies all requirements of Section 2, thus significantly going beyond what is possible using state-of-the-art cybersecurity modeling approaches. This way, TIM enables sound reasoning about time aspects of cyber attacks and defense mechanisms. We demonstrated

the use of TIM by applying it to a small but realistic example. In addition, we presented simTIM to simulate TIM models and showed how it can be used to statistically analyze and visualize the impact of the duration of attacker and defender actions on the damage created by attacks, demonstrating the usefulness of both TIM and simTIM.

Further theoretical work may determine optimal defense strategies in the TIM model for different types of systems. Further empirical work may compare different attack and defense strategies using simTIM. Both types of work help gain a better understanding of what strategies work best in what situation.

References

1. Ayed, D., Dragan, P.A., Félix, E., Mann, Z.Á., Salant, E., Seidl, R., Sidiropoulos, A., Taylor, S., Vitorino, R.: Protecting sensitive data in the cloud-to-edge continuum: The FogProtect approach. In: 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid). pp. 279–288. IEEE (2022)
2. Barrett, C., Tinelli, C.: Satisfiability modulo theories. In: Handbook of model checking, pp. 305–343. Springer (2018)
3. Clempner, J.B.: Learning attack-defense response in continuous-time discrete-states Stackelberg Security Markov games. Journal of Experimental & Theoretical Artificial Intelligence **36**(7), 1239–1260 (2024)
4. Farhang, S., Grossklags, J.: When to invest in security? Empirical evidence and a game-theoretic approach for time-based security. In: 16th Annual Workshop on the Economics of Information Security (WEIS) (2017)
5. Hoffmann, R.: Stochastic model of the simple cyber kill chain: cyber attack process as a regenerative process. In: 19th International Conference on Computer Information Systems and Industrial Management (CISIM). pp. 355–365. Springer (2020)
6. Husák, M., Komárková, J., Bou-Harb, E., Čeleda, P.: Survey of attack projection, prediction, and forecasting in cyber security. IEEE Communications Surveys & Tutorials **21**(1), 640–660 (2018)
7. Keller, C., Mann, Z.Á.: Towards understanding adaptation latency in self-adaptive systems. In: Service-Oriented Computing–ICSOC 2019 Workshops. pp. 42–53. Springer (2020)
8. Mann, Z.Á.: Urgency in cybersecurity risk management: toward a solid theory. In: IEEE 37th Computer Security Foundations Symposium (CSF). pp. 589–602 (2024)
9. Mann, Z.Á., Kunz, F., Laufer, J., Bellendorf, J., Metzger, A., Pohl, K.: RADAR: Data protection in cloud-based computer systems at run time. IEEE Access **9**, 70816–70842 (2021)
10. Merlevede, J., Johnson, B., Grossklags, J., Holvoet, T.: Time-dependent strategies in games of timing. In: 10th International Conference on Decision and Game Theory for Security (GameSec). pp. 310–330. Springer (2019)
11. Outkin, A.V., Schulz, P.V., Schulz, T., Tarman, T.D., Pinar, A.: Defender policy evaluation and resource allocation with MITRE ATT&CK evaluations data. IEEE Transactions on Dependable and Secure Computing **20**(3), 1909–1926 (2023)
12. Van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: FlipIt: The game of "stealthy takeover". Journal of Cryptology **26**, 655–713 (2013)
13. Woods, D.W., Böhme, R.: SoK: Quantifying cyber risk. In: IEEE Symposium on Security and Privacy (SP). pp. 211–228. IEEE (2021)

14. Zeng, J., Wu, S., Chen, Y., Zeng, R., Wu, C.: Survey of attack graph analysis methods from the perspective of data and knowledge processing. Security and Communication Networks **2019**(1), 2031063 (2019)
15. Zhang, H., Mi, Y., Liu, X., Zhang, Y., Wang, J., Tan, J.: A differential game approach for real-time security defense decision in scale-free networks. Computer Networks **224**, 109635 (2023)
16. Zmiewski, S.S., Laufer, J., Mann, Z.Á.: Automatic online quantification and prioritization of data protection risks. In: 17th International Conference on Availability, Reliability and Security (ARES). p. art. nr. 7 (2022)
17. Żychowski, A., Mańdziuk, J.: Evolution of strategies in sequential security games. In: Proceedings of the 20th AAMAS Conference. pp. 1434–1442 (2021)