Intrusion Tolerance for Networked Systems through Two-Level Feedback Control

Kim Hammar[†] and Rolf Stadler[†]
[†] KTH Royal Institute of Technology, Sweden
Email: {kimham, stadler}@kth.se

***Abstract*—We formulate intrusion tolerance for a system with service replicas as a two-level optimal control problem. On the local level node controllers perform intrusion recovery, and on the global level a system controller manages the replication factor. The local and global control problems can be formulated as classical problems in operations research, namely, the machine replacement problem and the inventory replenishment problem. Based on this formulation, we design TOLERANCE, a novel control architecture for intrusion-tolerant systems. We prove that the optimal control strategies on both levels have threshold structure and design efficient algorithms for computing them. We implement and evaluate TOLERANCE in an emulation environment where we run 10 types of network intrusions. The results show that TOLERANCE can improve service availability and reduce operational cost compared with state-of-the-art intrusion-tolerant systems.**

***Index Terms*—Intrusion tolerance, Byzantine fault tolerance, BFT, intrusion recovery, optimal control, POMDP, MDP, CMDP.**

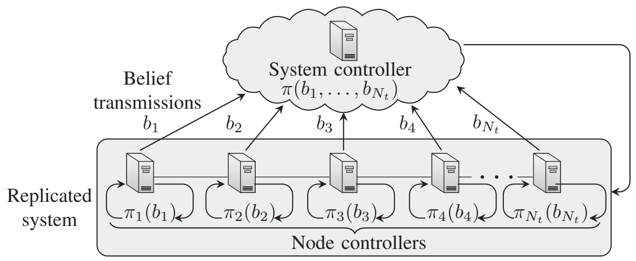


Fig. 1: Two-level feedback control for intrusion tolerance; node controllers with strategies $\pi_1,\ldots,\pi_{N_t}$ compute belief states $b_1,\ldots,b_{N_t}$ and make local recovery decisions; a global system controller with strategy π receives belief states and manages the replication factor N_t.

I. Introduction

As our reliance on online services is growing, there is an increasing demand for reliable systems that provide correct service without disruption. Traditionally, the main cause of disruption in networked systems has been hardware failures and power outages [1], [2]. While tolerance against these types of failures is important, a growing source of disruptions is network intrusion [3].

Intrusions are fundamentally different from hardware failures as an attacker can behave arbitrarily, i.e., Byzantine, which leads to unanticipated failure behavior. Due to the high costs of such failures and the fact that it is impractical to prevent all intrusions, the ability to *tolerate* intrusions becomes necessary [4]. This ability is particularly important for safety-critical applications, e.g., real-time control systems [5]–[10], control-planes for software defined networks [11], SCADA systems [12], [13], and e-commerce applications [14].

We call a system *intrusion-tolerant* if it provides *correct service* while intrusions occur [15]–[17]. The common approach to build intrusion-tolerant systems is to replicate the system over a set of *nodes*, each of which can respond to service requests. Through such redundancy, *compromised* and *crashed* nodes can be substituted by *healthy* nodes.

Current intrusion-tolerant systems are based on three main building blocks: (*i*) a protocol for service replication that tolerates a subset of compromised and crashed nodes; (*ii*) a replication strategy that adjusts the replication factor; and (*iii*) a recovery strategy that determines when to recover potentially compromised nodes [4], [15], [16].

Replication protocols that satisfy the condition in (*i*) are called *Byzantine fault-tolerant* (BFT) and have been studied extensively (see survey [18]). In comparison, few prior works have studied (*ii*) and (*iii*). As a consequence, current intrusion-tolerant systems typically use a fixed replication factor [19]–[21] and rely on inefficient recovery strategies, such as periodic recoveries [19], [22], heuristic rule-based recoveries [16], or manual recoveries by system administrators [23], [24].

In this paper, we address the above limitations and present TOLERANCE, which stands for Two-level recovery and replication control with feedback. TOLERANCE is a control architecture for intrusion-tolerant systems with two levels of control (see Fig. 1). On the local level node controllers perform intrusion recovery, and on the global level a system controller manages the replication factor. The associated control problems can be formulated as two classical problems in operations research, namely, the *machine replacement problem* [25] and the *inventory replenishment problem* [26]. Based on this formulation, we prove structural properties of the optimal control strategies and design efficient algorithms for computing them.

Key benefits of the TOLERANCE architecture are: (*i*) through feedback in terms of alerts and log files, the system can promptly adapt to network intrusions; (*ii*) by using two control levels rather than one, the system can tolerate partial failures and network partitions; and (*iii*) through the connection with classical problems from operations research, the system can leverage well-established control techniques with theoretical guarantees.

To assess the performance of TOLERANCE, we implement it in an emulation environment where we run 10 types of network intrusions. The results show that TOLERANCE can achieve higher service availability and lower operational cost than state-of-the-art intrusion-tolerant systems.

Our contributions can be summarized as follows:

1) We present and evaluate TOLERANCE, a novel control architecture for intrusion-tolerant systems that uses two levels of control to decide when to perform recovery and when to increase the replication factor.
2) We prove properties of the optimal control strategies and design efficient algorithms for computing them.
3) We implement TOLERANCE in an emulation environment and evaluate its performance against 10 types of network intrusions.

Software and data availability. Source code, container images, and a dataset of 6400 intrusion traces are available in the supplementary material [27], [28].

II. The Intrusion Tolerance Use Case

We consider a set of nodes that collectively offer a service to a client population. Each node is segmented into two domains: an application domain, which runs a service replica, and a privileged domain, which runs security and control functions. The replicas are coordinated through a replication protocol that relies on digital signatures and guarantees correct service if no more than f nodes are compromised or crashed simultaneously.

Clients access the service through gateways, which also are accessible to an attacker. The attacker's goal is to intrude on the system and compromise replicas while avoiding detection. We assume that the attacker a) does not have physical access to nodes; b) can not forge digital signatures; and c) can only access the service replicas, not the privileged domains (i.e., we consider the *hybrid failure model* [29]). Apart from these restrictions, the attacker can control a compromised replica in arbitrary, i.e. Byzantine, ways. It can shut it down, delay its service responses, communicate with other replicas, etc.

To prevent the number of compromised and crashed nodes to exceed f, we consider three types of response actions: (*i*) recover compromised nodes; (*ii*) evict crashed nodes from the system; and (*iii*) add new nodes. Each of these actions incurs a cost that must be weighed against the security benefit.

Note that, while we focus on the response actions: eviction, addition, and recovery of nodes in this paper, the use case can be extended to include additional response actions, such as rate limiting [30] and access control [31]. This extension is however beyond the scope of this paper.

III. Background on Intrusion-Tolerant Systems

A. Fault-Tolerant Systems

Research on fault-tolerant systems has almost a century-long history with the seminal work being made by von Neumann [32] and Shannon [33] in 1956. The early work focused on tolerance against hardware failures. Since then the field has broadened to include tolerance against software bugs, operator mistakes, and malicious attacks [1], [2], [34]–[36].

The common approach to build a fault-tolerant service is *redundancy*, whereby the service is provided by a set of replicas. Through such redundancy, compromised and crashed replicas can be substituted by healthy replicas as long as the healthy replicas can coordinate their service responses. This coordination problem is known as the *consensus* problem.

B. Consensus

Consensus is the problem of reaching agreement among distributed nodes subject to failures [37]. The solvability of consensus depends on synchrony and failure assumptions.

The main synchrony assumptions are: (*i*) the *synchronous model*, where there is an upper bound on the communication delay between nodes; (*ii*) the *partially synchronous model*, where an upper bound exists but the system may have periods of instability where the bound is violated; and (*iii*) the *asynchronous model*, where no bound exists [37]–[39].

The main failure assumptions are: (*i*) the *crash-stop failure model*, where nodes fail by crashing; (*ii*) the *Byzantine failure model*, where nodes fail arbitrarily; and (*iii*) the *hybrid failure model*, where nodes fail arbitrarily but are equipped with trusted components that fail by crashing [29], [40].

Deterministic consensus is *not* solvable in the asynchronous model [41, Thm. 1]. In the partially synchronous model, however, consensus *is* solvable with N nodes and $f = \frac{N-1}{2}$ crash-stop failures, $f = \frac{N-1}{3}$ Byzantine failures, and $f = \frac{N-1}{2}$ hybrid failures [42, Thm. 1][40, Thms. 5.8,5.11][43, Thm. 2][44, Thm. 1]. Similarly, consensus *is* solvable in the synchronous model with $f = N-1$ crash-stop failures, $f = \frac{N-1}{2}$ Byzantine failures, and $f = \frac{N-1}{2}$ hybrid failures [40, Thm. 5.2][45, Cor. 14] [46, Thm. 1].

C. Intrusion-Tolerant Systems

Intrusion-tolerant systems extend fault-tolerant systems with intrusion detection, recovery, and response [15], [16], [47], [48]. We call a system *intrusion-tolerant* if it remains secure and operational while intrusions occur [4], [15]. We use the following metrics to quantify intrusion tolerance:

1) *Average time-to-recovery* $T^{(\mathrm{R})}$: the average time from node compromise until recovery starts.
2) *Average availability* $T^{(\mathrm{A})}$: the fraction of time where the number of compromised and crashed nodes is at most f. (We assume the *primary-partition* model [49] to circumvent the CAP theorem [50, Thm. 2].)
3) *Frequency of recoveries* $F^{(\mathrm{R})}$: the fraction of time where recovery occurs.

IV. Intrusion Tolerance through Two-Level Feedback Control

In this section, we describe TOLERANCE: a two-level control architecture for intrusion-tolerant systems (see Fig. 2). It is a distributed system with $N_t \geq 2f+1+k$ *nodes* connected through an authenticated network [51]. Each node runs a *service replica*. The replicas are coordinated through

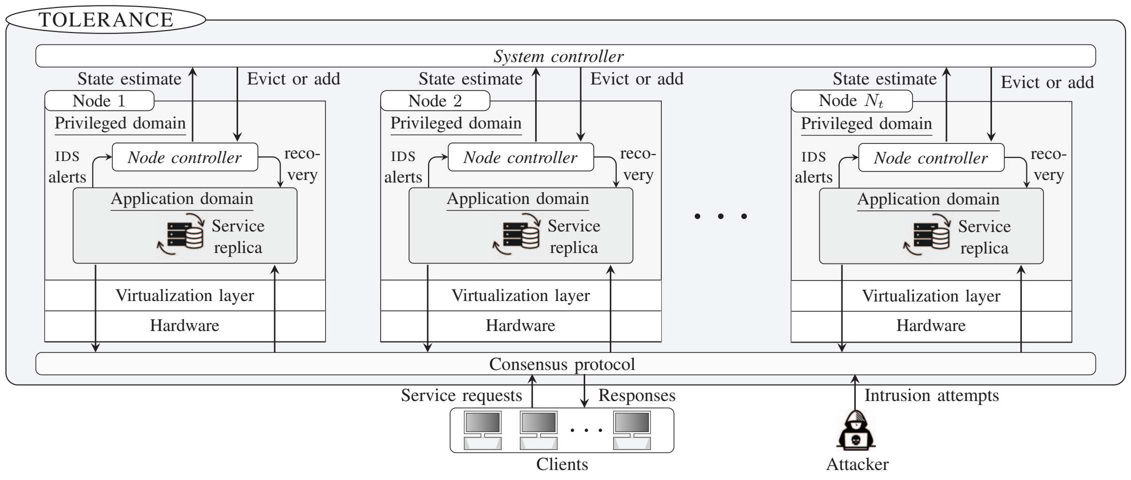


Fig. 2: The TOLERANCE architecture; N_t nodes provide a replicated service to a client population; service responses are coordinated through an intrusion-tolerant consensus protocol; local node controllers decide when to perform recovery and a global system controller manages the replication factor N_t.

a reconfigurable consensus protocol that guarantees *correct service* if no more than f nodes are compromised or crashed simultaneously (e.g., reconfigurable MINBFT [43, §4.2]).

TOLERANCE uses two levels of control: local and global. On the local level, each node runs a *node controller* that monitors the service replica through alerts from an Intrusion Detection System (IDS). Based on these alerts, the controller estimates the replica's state, i.e., whether it is compromised or not, and decides when it should be recovered. As each recovery incurs a cost, the challenge for the controller is to balance the recovery costs against the security benefits. To guarantee correct service, at most k nodes are allowed to recover simultaneously.

The global level includes a *system controller* that collects state estimates from the nodes and adjusts the replication factor N_t. When deciding if N_t should be increased, the controller faces a classical dilemma in reliability theory [52]. On the one hand, it aims at high redundancy to maximize service availability. On the other hand, it does not want an excessively large and costly system.

Since the only task of the system controller is to execute control actions and communicate with the node controllers, it can be deployed on a standard crash-tolerant system, e.g., a RAFT-based system [53]. For this reason, we consider the probability that the system controller crashes to be negligible.

Similar to the VM-FIT and the WORM-IT architectures [19], [20], [29], each node in TOLERANCE is segmented into two domains: a *privileged domain*, which can only fail by crashing, and an *application domain*, which may be compromised by an attacker. The controllers and the IDSs execute in the privileged domain, whereas the service replicas execute in the application domain. The separation between the two domains can be realized in several ways. One option is to use a secure coprocessor to execute the privileged domain (e.g., IBM 4758) [22], [44]. Another option, which does not require special hardware, is to use a security kernel to run the privileged domain, as in the WORM-IT architecture [29]. A third option, used in the VM-FIT architecture [19], is to separate the application domain from the privileged domain using a secure virtualization layer that can be formally verified [54]. TOLERANCE implements the last option for the following reasons: (*i*) virtualization enables efficient recovery of a compromised replica by replacing its virtual container [19], [20]; and (*ii*) virtualization simplifies implementation of software diversification, which reduces the correlation between compromise events across nodes [55].

A. Correctness

We say that a system provides *correct service* if the healthy replicas satisfy the following properties:

Each request is eventually executed. (Liveness)

Each executed request was sent by a client. (Validity)

Each replica executes the same request sequence. (Safety)

Proposition 1. TOLERANCE *provides correct service if the following holds:*

(a) An attacker can not forge digital signatures.
(b) Network links are authenticated and reliable [37, p. 42].
(c) At most k nodes recover simultaneously and at most f nodes are compromised or crashed simultaneously.
(d) $N_t \geq 2f + 1 + k$ at all times t.
(e) The system is partially synchronous [38].

Proof (Sketch). (a)-(b) and the fact that the controllers can only fail by crashing imply the hybrid failure model (§III-B). (c)-(d) state that at least $f + 1 + k$ nodes are healthy at all times t. These properties together with the tolerance threshold $f = \frac{N_t - 1 - k}{2}$ of the consensus protocol (e.g., MINBFT [43, §4.2]) imply (Safety) ([43, Thms. 1–2]). Next, it follows from (e) that the healthy nodes will eventually agree on the response to any service request, which allows to circumvent FLP [41,

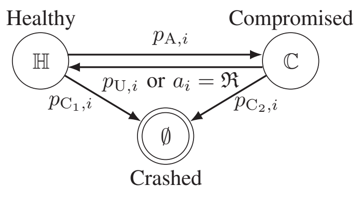


Fig. 3: State transition diagram of node i (2): disks represent states; arrows represent state transitions; labels indicate probabilities and conditions for state transition; self-transitions are not shown.

Thm. 1] and achieve (Liveness). Finally, (Validity) is ensured by the consensus protocol (e.g., MINBFT [43, §4.2]). □

Remark. By appropriate use of cryptographic methods and firewalls, TOLERANCE can be extended to provide confidentiality in addition to (Safety), (Liveness), and (Validity), see e.g., [56], [57]. We leave this extension for future work.

V. Formalizing Intrusion Tolerance As a Constrained Two-Level Control Problem

To guarantee correct service, the controllers in TOLERANCE must ensure that: a) the number of compromised and crashed nodes is at most f, which is achieved by recovery; and b) the number of nodes satisfies $N_t \geq 2f+1+k$, which is achieved by replacing crashed nodes (Prop. 1). In the following, we formulate the problem of meeting these constraints while minimizing operational cost as a control problem with a local and a global level. On the local level, node controllers minimize cost while meeting a), and on the global level, the system controller minimizes cost while meeting b).

Notation. Random variables are denoted by upper-case letters (e.g., X) and their values by lower-case (e.g., x). $\mathbb{P}$ is a probability measure. The expectation of an expression ϕ with respect to X is written as $\mathbb{E}_X[\phi]$. When ϕ includes many random variables that depend on π, we simply write $\mathbb{E}_\pi[\phi]$. $x \sim \phi$ means that x is sampled from ϕ. We use $\mathbb{P}[x]$ as a shorthand for $\mathbb{P}[X = x]$. Calligraphy letters (e.g., $\mathcal{V}$) represent sets. $[\![\cdot]\!]$ is the Iverson bracket. $\lfloor\cdot\rfloor$ is the floor function. Further notation is listed in Table 1.

A. The Local Level: Controlling Intrusion Recovery

The local level consists of N_t node controllers, each of which decides when to perform intrusion recovery. We model this control problem as an instance of the *machine replacement problem* from operations research [25].

Let $\mathcal{N}_t \triangleq \{1, 2, \dots, N_t\}$ be the set of nodes and $\pi_{1,t}, \pi_{2,t}, \dots, \pi_{N,t}$ the corresponding control strategies at time t. Node i has state $s_{i,t} \in \mathcal{S}_\mathrm{N}$ with three values: $\emptyset$ if it is crashed, $\mathbb{C}$ if it is compromised, and $\mathbb{H}$ if it is healthy (see Fig. 3). Controller i takes action $a_{i,t} \in \mathcal{A}_\mathrm{N}$ with two values: $\mathfrak{R}$ is the recovery action and $\mathfrak{W}$ is the wait action. We assume that a recovery action at time t is completed by $t+1$.

Notation(s)	*Description*
$\mathcal{N}_t, N_t, f$	Set of nodes, number of nodes, tolerance threshold (Prop. 1)
$T^{(\mathrm{R})}$	Average time-to-recovery when an intrusion occurs (§III-C)
$F^{(\mathrm{R})}, T^{(\mathrm{A})}$	Frequency of recoveries, average service availability (5)
$T^{(f)}, k$	Time to system failure (Fig. 6), # parallel recoveries (Prop. 1)
$R(t), R_i(t)$	Reliability function for the system and for a node (Fig. 6)
J_i, J	Objectives of node i (5) and the system controller (9)
$\pi_{i,t}, \Pi_\mathrm{N}$	Strategy and strategy space of node i, $\pi_{i,t} \in \Pi_\mathrm{N}$ (6)
$\pi^\star_{i,t}, \alpha^\star_i$	Optimal strategy and threshold for node i (6)
$s_{i,t}, o_{i,t}$	State (1) and observation (3) of node i at time t
$b_{i,t}$	Belief state of node i at time t (4)
$S_{i,t}, O_{i,t}, B_{i,t}$	Random variables with realizations $s_{i,t}$ (1), $o_{i,t}$ (3), $b_{i,t}$ (4)
$a_{i,t}, c_{i,t}$	Action and cost of node controller i at time t (6)
$A_{i,t}, C_{i,t}$	Random variables with realizations $a_{i,t}$ and $c_{i,t}$ (6)
$\mathcal{S}_\mathrm{N}, \mathcal{A}_\mathrm{N}, \mathcal{O}_\mathrm{N}$	State, action and observation spaces of nodes
$\mathfrak{W}, \mathfrak{R} = 0, 1$	The ($\mathfrak{W}$)ait and ($\mathfrak{R}$)ecovery actions (Fig. 3)
$\mathbb{H}, \mathbb{C} = 0, 1$	The ($\mathbb{H}$)ealthy and ($\mathbb{C}$)compromised node states (Fig. 3)
$\emptyset$	The crashed node state (Fig. 3)
$f_{\mathrm{N},i}, Z_i$	Transition (2) and observation (3) functions for node i
$c_\mathrm{N}(s_{i,t}, a_{i,t})$	Cost function for a node (5)
$p_{\mathrm{A},i}$	Probability that node i is compromised (2)
$p_{\mathrm{C}_1,i}$	Probability that node i crashes in the healthy state (2)
$p_{\mathrm{C}_2,i}$	Probability that node i crashes in the compromised state (2)
$p_{\mathrm{U},i}$	Probability that the service replica of node i is updated (2)
Δ_R	Maximum allowed time between node recoveries (6)
$\mathcal{W}, \mathcal{R}$	Wait and recovery sets for node controllers [27]
S_t, s_t	State of the system controller at time t (8), s_t realizes S_t
a_t, c_t	Action and cost of system controller at time t (8)
A_t, C_t	Random variables with realizations a_t, c_t (10)
f_S	Transition function of the system controller (8)
$\mathcal{S}_\mathrm{S}, \mathcal{A}_\mathrm{S}$	State and action spaces of the system controller
$s_{\max}$	Maximum number of nodes (§V-B)
π, Π_S	Strategy and strategy space of the system controller (10)
ϵ_A	Lower bound on the average service availability (10)

TABLE 1: Notation.

The evolution of $s_{i,t}$ can be written as

$$s_{i,t+1} \sim f_{\mathrm{N},i}(\cdot \mid S_{i,t} = s_{i,t}, A_{i,t} = a_{i,t}), \tag{1}$$

where $f_{\mathrm{N},i}$ is a Markovian transition function defined as

$$f_{\mathrm{N},i}(\emptyset \mid \emptyset, \cdot) \triangleq 1 \tag{2a}$$
$$f_{\mathrm{N},i}(\emptyset \mid \mathbb{H}, \cdot) \triangleq p_{\mathrm{C}_1,i} \tag{2b}$$
$$f_{\mathrm{N},i}(\emptyset \mid \mathbb{C}, \cdot) \triangleq p_{\mathrm{C}_2,i} \tag{2c}$$
$$f_{\mathrm{N},i}(\mathbb{H} \mid \mathbb{H}, \mathfrak{R}) \triangleq (1-p_{\mathrm{A},i})(1-p_{\mathrm{C}_1,i}) \tag{2d}$$
$$f_{\mathrm{N},i}(\mathbb{H} \mid \mathbb{H}, \mathfrak{W}) \triangleq (1-p_{\mathrm{A},i})(1-p_{\mathrm{C}_1,i}) \tag{2e}$$
$$f_{\mathrm{N},i}(\mathbb{H} \mid \mathbb{C}, \mathfrak{R}) \triangleq (1-p_{\mathrm{A},i})(1-p_{\mathrm{C}_2,i}) \tag{2f}$$
$$f_{\mathrm{N},i}(\mathbb{H} \mid \mathbb{C}, \mathfrak{W}) \triangleq (1-p_{\mathrm{C}_2,i})p_{\mathrm{U},i} \tag{2g}$$
$$f_{\mathrm{N},i}(\mathbb{C} \mid \mathbb{H}, \mathfrak{W}) \triangleq f_{\mathrm{N},i}(\mathbb{C} \mid \mathbb{H}, \mathfrak{R}) = (1-p_{\mathrm{C}_1,i})p_{\mathrm{A},i} \tag{2h}$$
$$f_{\mathrm{N},i}(\mathbb{C} \mid \mathbb{C}, \mathfrak{R}) \triangleq (1-p_{\mathrm{C}_2,i})p_{\mathrm{A},i} \tag{2i}$$
$$f_{\mathrm{N},i}(\mathbb{C} \mid \mathbb{C}, \mathfrak{W}) \triangleq (1-p_{\mathrm{C}_2,i})(1-p_{\mathrm{U},i}), \tag{2j}$$

where $p_{\mathrm{A},i}$ is the probability that the attacker compromises the node during the time interval $[t, t+1]$, $p_{\mathrm{C}_1,i}$ is the probability that the node crashes in the healthy state, $p_{\mathrm{C}_2,i}$ is the probability that the node crashes in the compromised state, and $p_{\mathrm{U},i}$ is the probability that the node's software is updated. These parameters can be set based on domain knowledge or be obtained through system measurements. In fact, companies such as Google, Meta, and IBM, have documented procedures for estimating such parameters, see e.g., [58], [59].

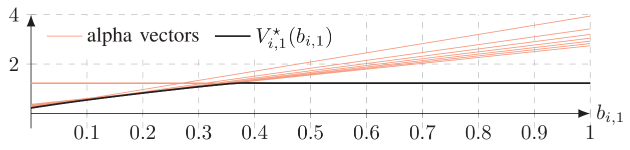


Fig. 4: The optimal value function $V^{\star}_{i,t}(b_{i,t})$ for Prob. 1; the dashed red lines indicate the alpha-vectors [61], [62] and the solid black lines indicate $V^{\star}_{i,t}(b_{i,t})$; the parameters for computing $V^{\star}_{i,t}(b_{i,t})$ are listed in [27, App. E].

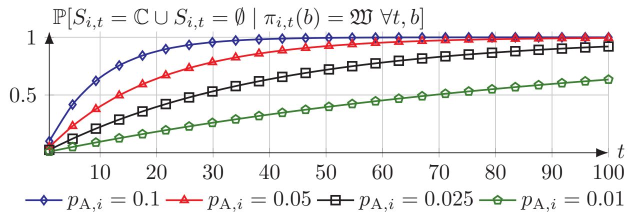


Fig. 5: Probability that a node is compromised ($\mathbb{C}$) or crashed ($\emptyset$) by time-step t if no recoveries occur; the curves relate to $p_{\mathrm{A},i}$ (2); hyperparameters are listed in [27, App. E].

Equations (2a)–(2c) capture the transitions to $\emptyset$, which is an absorbing state [60, Def. 4.4]. (A crashed node that is restarted is considered as a new node in the model.) Next, (2d)–(2g) define the transitions to $\mathbb{H}$, which is reached when the controller takes action $\mathfrak{R}$ (2d)–(2f) or when its software is updated (2g). Lastly, (2h)–(2j) capture the transitions to $\mathbb{C}$, which happen when an intrusion occurs.

It follows from (2) that the number of time-steps until a node fails (crash or compromise) is geometrically distributed (see Fig. 5). Note that (2) applies independently to each node, which means that compromise and crash events are statistically independent across nodes. This independence reflects our assumption that the system is geographically distributed and uses software diversification (§IV).

A node state is hidden from its controller. At time t, controller i observes $o_{i,t} \in \mathcal{O} \subset \mathbb{N}_0$, which is based on the number of IDS alerts received during the time interval $[t-1, t]$, weighted by priority. (While we focus on the IDS alert metric in this paper, alternative sources of metrics can be used; a comparison between different metrics is available in the supplementary material [27, App. H].) $o_{i,t}$ is drawn from the random variable $O_{i,t}$ with distribution

$$Z_i(o_{i,t} \mid s_{i,t}) \triangleq \mathbb{P}\left[O_i = o_{i,t} \mid S_{i,t} = s_{i,t}\right]. \tag{3}$$

Based on its observations, controller i computes the belief state

$$b_{i,t} \triangleq \mathbb{P}[S_{i,t} = \mathbb{C} \mid o_{i,1}, a_{i,1}, \ldots, a_{i,t-1}, o_{i,t}, b_{i,1}], \tag{4}$$

which is a sufficient statistic for $s_{i,t}$ (see [27, App. A]). $\pi_{i,t}$ can thus be defined as $\pi_{i,t} : [0,1] \to \{\mathfrak{W}, \mathfrak{R}\}$.

When selecting the strategy $\pi_{i,t}$, the controller balances two conflicting goals: minimize the average time-to-recovery $T^{(\mathrm{R})}_i$ and minimize the frequency of recoveries $F^{(R)}_i$. The weight $\eta \geq 1$ controls the trade-off between these two objectives, which results in the bi-objective

$$\begin{aligned} \text{minimize } J_i &\triangleq \lim_{T\to\infty} \left[\eta T^{(\mathrm{R})}_{i,T} + F^{(\mathrm{R})}_{i,T}\right] \\ &= \lim_{T\to\infty} \left[\frac{1}{T}\sum_{t=1}^{T} \underbrace{\eta s_{i,t} - a_{i,t}\eta s_{i,t} + a_{i,t}}_{\triangleq c_{\mathrm{N}}(s_{i,t}, a_{i,t})}\right], \end{aligned} \tag{5}$$

where $T^{(\mathrm{R})}_{i,T}$ and $F^{(\mathrm{R})}_{i,T}$ denote the average values at time T, $\mathbb{H}, \mathbb{C} = 0, 1$, $\mathfrak{W}, \mathfrak{R} = 0, 1$, and c_{N} is the cost function.

We define the intrusion recovery problem as that of minimizing J_i (5) subject to a bounded-time-to-recovery (BTR) constraint [63]. Formally,

Problem 1 (Optimal Intrusion Recovery)**.**

$$\begin{aligned} &\underset{\pi_{i,t}\in\Pi_{\mathrm{N}}}{\text{minimize}} && \mathbb{E}_{\pi_{i,t}}\left[J_i \mid b_{i,1} = p_{\mathrm{A},i}\right] && \forall i \in \mathcal{N}_t && \text{(6a)} \\ &\text{subject to} && a_{i,k\Delta_{\mathrm{R}}} = \mathfrak{R} && \forall i, k && \text{(6b)} \\ &&& s_{i,t+1} \sim f_{\mathrm{N},i}(\cdot \mid s_{i,t}, a_{i,t}) && \forall i, t && \text{(6c)} \\ &&& o_{i,t+1} \sim Z_i(\cdot \mid s_{i,t}) && \forall i, t && \text{(6d)} \\ &&& a_{i,t+1} = \pi_{i,t}(b_{i,t}) && \forall i, t, && \text{(6e)} \end{aligned}$$

where $t, k = 1, 2, \ldots$; Π_{N} is the strategy space; $b_{i,1}$ is the initial state distribution of node i; $\mathbb{E}_{\pi_{i,t}}$ denotes the expectation over the random variables $(S_{i,t}, O_{i,t}, A_{i,t}, B_{i,t})_{t\in\{1,2,\ldots\}}$ when following strategy $\pi_{i,t}$; (6b) is the BTR constraint; (6c) is the dynamics constraint; (6d) captures the observations; and (6e) captures the actions. (Remark: Prob. 1 does not include a constraint on the maximum number of parallel recoveries (i.e., k in Prop. 1) as we assume this constraint is enforced by the implementation.)

We say that a *strategy* $\pi^{\star}_{i,t}$ *is optimal* if it solves (6). Figure 4 shows the expected cost of $\pi^{\star}_{i,t}$. We note that $\pi^{\star}_{i,t}$ has threshold structure, as stated below.

Theorem 1. *Assuming*

$$\begin{aligned} &p_{\mathrm{A},i}, p_{\mathrm{U},i}, p_{\mathrm{C}_1,i}, p_{\mathrm{C}_2,i} \in (0,1) && \text{(A)} \\ &p_{\mathrm{A},i} + p_{\mathrm{U},i} \leq 1 && \text{(B)} \\ &\frac{p_{\mathrm{C}_1,i}(p_{\mathrm{U},i}-1)}{p_{\mathrm{A},i}(p_{\mathrm{C}_1,i}-1) + p_{\mathrm{C}_1,i}(p_{\mathrm{U},i}-1)} \leq p_{\mathrm{C}_2,i} && \text{(C)} \\ &Z_i(o_{i,t} \mid s_{i,t}) > 0 \qquad \forall o_{i,t} \in \mathcal{O}, s_{i,t} \in \mathcal{S}_{\mathrm{N}} && \text{(D)} \\ &Z_i \text{ is TP-2 [62, Def. 10.2.1]}, && \text{(E)} \end{aligned}$$

then there exists an optimal strategy $\pi^{\star}_{i,t}$ that solves Prob. 1 for each node i and satisfies

$$\pi^{\star}_{i,t}(b_{i,t}) = \mathfrak{R} \iff b_{i,t} \geq \alpha^{\star}_{i,t} \qquad \forall t, \tag{7}$$

where $\alpha^{\star}_{i,t} \in [0,1]$ is a threshold.

Proof. See the supplementary material [27, App. B]. □

Corollary 1. *The thresholds satisfy $\alpha^{\star}_{i,t+1} \geq \alpha^{\star}_{i,t}$ for $t \in [k\Delta_R, (k+1)\Delta_R]$ and $i \in \mathcal{N}$. As $\Delta_{\mathrm{R}} \to \infty$, all thresholds converge to $\alpha^{\star}_i$, which is time-independent.*

Proof. See the supplementary material [27, App. C]. □

Theorem 1 states that under assumptions generally met in practice, there exists an optimal strategy for each node that performs recovery when the belief (4) exceeds a threshold (7). Further, Cor. 1 says two things: (*i*) the threshold increases as the time until the next periodic recovery decreases; and (*ii*) when there are no periodic recoveries (i.e., when $\Delta_R = \infty$), the threshold is independent of time.

The above statements rely on assumptions A–E. A–C are mild assumptions stating that the attack, crash, and upgrade probabilities are small but non-zero and that the difference $p_{\mathrm{C}_2,i} - p_{\mathrm{C}_1,i} > 0$ is sufficiently large. D is a technical assumption saying that the probability of observing k IDS alerts is non-zero for any $k \in \mathcal{O}$, which generally is true in practice (see §X). E states that the probability of high-priority IDS alerts increases when an intrusion occurs. While E may not always hold, empirical studies suggest that it holds for many types of intrusions [31], [64]. If E does not hold, the threshold strategy in (7) can still achieve near-optimal performance but it is not guaranteed to be optimal.

Theorem 1 and Cor. 1 lead to two important practical benefits. First, the complexity of computing optimal strategies can be reduced by only considering threshold strategies (see §X). Second, the optimal strategies can be efficiently implemented.

B. The Global Level: Controlling the Replication Factor

The global level includes a *system controller* that adjusts the replication factor N_t. At each time t, it receives the belief states $b_{1,t}, \dots, b_{N_t,t}$ from the nodes and decides whether N_t should be increased (see Fig. 1). A node that fails to send $b_{i,t}$ is considered to have crashed and is evicted from the system, which decrements N_t. (Remark: in practice, an evicted node can rejoin the system, but it is then considered as a new node in the model.) We assume that the system controller does not crash, i.e., we assume it is crash-tolerant (§IV).

A large replication factor N_t improves the service availability but increases cost (see Fig. 6). Our goal thus is to find the optimal cost-redundancy trade-off. We model this control problem as an instance of the *inventory replenishment problem* from operations research [26].

We define the state s_t to represent the expected number of healthy nodes at time t. The state space is $\mathcal{S}_{\mathrm{S}} \triangleq \{0, 1, \dots, s_{\max}\}$ and the initial state is $s_1 = N_1$.

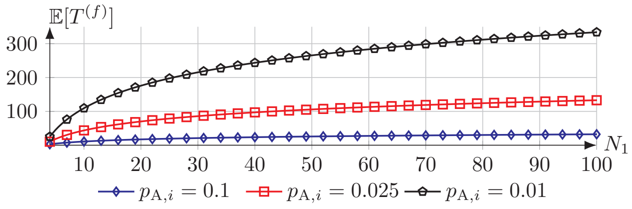
At time t, the controller adds $a_t \in \{0, 1\} \triangleq \mathcal{A}_{\mathrm{S}}$ nodes:

$$s_{t+1} \sim f_{\mathrm{S}}(\cdot \mid S_t = s_t, A_t = a_t), \tag{8}$$

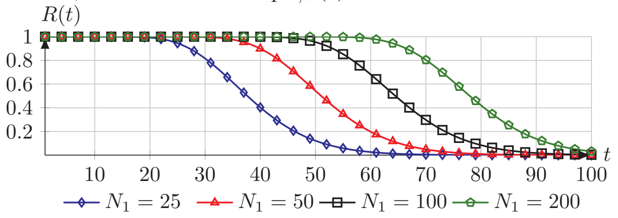
where f_{S} is defined as

$$f_{\mathrm{S}}(s_{t+1} \mid s_t, a_t) \triangleq \mathbb{P}\left[\left\lfloor \sum_{i \in \mathcal{N}_t} 1 - B_{i,t} \right\rceil = s_{t+1} - a_t \right].$$

When selecting the strategy π, the controller balances two conflicting goals: maximize the average service availability



(a) Mean time to failure (MTTF) in function of the number of initial nodes N_1; the curves relate to $p_{\mathrm{A},i}$ (2).



(b) Reliability curves for varying number of initial nodes N_1; The reliability function is defined as $R(t) \triangleq \mathbb{P}[T^{(f)} > t]$.

Fig. 6: Illustration of Prob. 2; $T^{(f)}$ is a random variable representing the time when $N_t < f + k + 1$ with $f = 3$ and $k = 1$ (Prop. 1); hyperparameters and formulas for computing the curves are listed in [27, App. E–F].

$T^{(\mathrm{A})}$ and minimize the number of nodes s_t. We model this bi-objective as follows.

$$\text{minimize } J \triangleq \lim_{T \to \infty} \left[\frac{1}{T} \sum_{t=1}^{T} s_t \right] \tag{9}$$

subject to $T^{(\mathrm{A})} \geq \epsilon_{\mathrm{A}}$, which can be expressed as (Prop. 1)

$$\lim_{T \to \infty} \left[\frac{1}{T} \sum_{t=1}^{T} [\![s_t \geq f + 1]\!] \right] \geq \epsilon_{\mathrm{A}},$$

where ϵ_{A} is the lower bound on service availability.

Given (9) and the Markov property of s_t (8), we define π as a function $\pi : \mathcal{S}_{\mathrm{S}} \to \Delta(\mathcal{A}_{\mathrm{S}})$, where $\Delta(\mathcal{A}_{\mathrm{S}})$ is the set of probability distributions over $\mathcal{A}_{\mathrm{S}}$. Based on this definition, we formulate the problem of controlling the replication factor as

Problem 2 (Optimal Replication Factor)**.**

$$\begin{aligned} \underset{\pi \in \Pi_{\mathrm{S}}}{\text{minimize}} \quad & \mathbb{E}_\pi \left[J \mid s_1 = N_1 \right] && \text{(10a)} \\ \text{subject to} \quad & \mathbb{E}_\pi \left[T^{(\mathrm{A})} \right] \geq \epsilon_{\mathrm{A}} && \text{(10b)} \\ & s_{t+1} \sim f_{\mathrm{S}}(\cdot \mid s_t, a_t) \quad \forall t && \text{(10c)} \\ & a_{t+1} \sim \pi(\cdot \mid s_t) \quad \forall t, && \text{(10d)} \end{aligned}$$

where Π_{S} is the strategy space; s_1 is the initial state; $\mathbb{E}_\pi$ denotes the expectation of the random variables $(S_t, A_t)_{t=1,2,\dots}$ under strategy π; (10b) is the availability constraint; (10c) is the dynamics constraint; and (10d) captures the actions.

We say that a *strategy* $\pi^\star$ *is optimal* if it solves (10). We note that (10) states a Constrained Markov Decision Problem (CMDP) [65], which leads to the following result.

Theorem 2. *Assuming*

$$\exists \pi \in \Pi_{\mathrm{S}} \text{ such that } \mathbb{E}_{\pi}\left[T^{(\mathrm{A})}\right] \geq \epsilon_{\mathrm{A}} \tag{A}$$

$$f_{\mathrm{S}}(s' \mid s, a) > 0 \tag{B}$$

$$\sum_{s'=s}^{s_{\max}} f_{\mathrm{S}}(s' \mid \hat{s}+1, a) \geq \sum_{s'=s}^{s_{\max}} f_{\mathrm{S}}(s' \mid \hat{s}, a) \tag{C}$$

$$\sum_{s'=s}^{s_{\max}} f_{\mathrm{S}}(s'|\hat{s},1) - f_{\mathrm{S}}(s'|\hat{s},0) \text{ is increasing in } s \tag{D}$$

for all $(s, \hat{s}, a) \in \mathcal{S}_{\mathrm{S}} \times \mathcal{S}_{\mathrm{S}} \times \mathcal{A}_{\mathrm{S}}$.

Then there exist two strategies π_{λ_1} *and* π_{λ_2} *that satisfy*

$$\pi_{\lambda_1}(s_t) = 1 \iff s_t \leq \beta_1 \qquad \forall t, s_t \in \mathcal{S}_{\mathrm{S}} \tag{11}$$

$$\pi_{\lambda_2}(s_t) = 1 \iff s_t \leq \beta_2 \qquad \forall t, s_t \in \mathcal{S}_{\mathrm{S}} \tag{12}$$

and an optimal strategy $\pi^\star$ *that satisfies*

$$\pi^\star(s_t) = \kappa\pi_{\lambda_1}(s_t) + (1-\kappa)\pi_{\lambda_2}(s_t) \qquad \forall t, s_t \in \mathcal{S}_{\mathrm{S}} \tag{13}$$

for some probability $\kappa \in [0,1]$, *where* λ_1, λ_2 *are Lagrange multipliers and* β_1, β_2 *are thresholds.*

Proof. See the supplementary material [27, App. D]. □

Theorem 2 states that under assumptions A-D, there exists an optimal strategy that can be written as a mixture of two threshold strategies. A ensures that Prob. 2 is feasible. This assumption can be met by tuning ϵ_{A} (10b). B–D are mild assumptions which generally hold in practice. B states that the probability of several simultaneous intrusions or recoveries is non-zero; C conveys that a large number of healthy nodes at current time increases the probability of having a large number of healthy nodes in the future; and D states that the transition probabilities are tail-sum supermodular [62, Eq. 9.6].

VI. Computing Optimal Control Strategies

Algorithm 1: Parametric optimization for optimal recovery strategy.

```
1 Input: Problem 1, a node $i \in \mathcal{N}$, and a parametric optimizer PO.
2 Output: A near-optimal recovery strategy $\hat{\pi}_{i,\boldsymbol{\theta},t}$ (6).
3 Algorithm
4     if $\Delta_{\mathrm{R}} < \infty$, $\mathrm{d} \leftarrow \Delta_{\mathrm{R}} - 1$, else $\mathrm{d} \leftarrow 1$
5     $\Theta \leftarrow [0,1]^{\mathrm{d}}$
6     $\pi_{i,\boldsymbol{\theta},t}(b_t) \triangleq \begin{cases} \mathfrak{R} & \text{if } b_t \geq \boldsymbol{\theta}_k \text{ where } k = \max[t, \mathrm{d}] \\ \mathfrak{W} & \text{otherwise} \end{cases} \quad \forall \boldsymbol{\theta} \in \Theta, t \geq 1$
7     $J_{i,\boldsymbol{\theta}} \leftarrow \mathbb{E}_{\pi_{i,\boldsymbol{\theta},t}}[J_i]$ where $J_i$ is defined in (5)
8     $\hat{\pi}_{i,\boldsymbol{\theta},t} \leftarrow \mathrm{PO}(\Theta, J_{i,\boldsymbol{\theta}})$
9     return $\hat{\pi}_{i,\boldsymbol{\theta},t}$
```

The time complexity of Prob. 1 (optimal intrusion recovery) is in the complexity class PSPACE-hard [66, Thm. 6]. (Recall that P $\subseteq$ NP $\subseteq$ PSPACE.) The time complexity of Prob. 2 (optimal replication factor), on the other hand, is polynomial [67, Thm. 1][65, Thm. 4.3].

To manage the high time complexity of Prob. 1, we exploit Thm. 1 and parameterize $\pi^\star_{i,t}$ with a finite number of thresholds. Given this parametrization, we formulate Prob. 1 as a parametric optimization problem, which can be solved with standard optimization algorithms, e.g., stochastic approximation [68]. Algorithm 1 contains the pseudocode of our solution.

Algorithm 2: Linear program for optimal replication strategy.

```
1 Input: Problem 2 and a linear programming solver LP.
2 Output: An optimal replication strategy $\pi^\star$ (10).
3 Algorithm
4     Solve (14) using LP; let $\rho^\star$ denote the solution of (14) and define
          $\pi^\star(a|s) \triangleq \frac{\rho^\star(s,a)}{\sum_{s \in \mathcal{S}_{\mathrm{S}}} \rho^\star(s,a)} \qquad \forall s \in \mathcal{S}_{\mathrm{S}}, a \in \mathcal{A}_{\mathrm{S}}$
          $\underset{\rho}{\text{minimize}} \sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} s\rho(s,a)$   (14a)
          $\text{subject to } \rho(s,a) \geq 0 \quad \forall s \in \mathcal{S}_{\mathrm{S}}, a \in \mathcal{A}_{\mathrm{S}}$   (14b)
          $\sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) = 1$   (14c)
          $\sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) = \sum_{s' \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s',a) f_{\mathrm{S}}(s'|s,a) \ \forall s \in \mathcal{S}_{\mathrm{S}}$   (14d)
          $\sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) [\![ s_t \geq f+1 ]\!] \geq \epsilon_{\mathrm{A}}$   (14e)
      return $\pi^\star$
```

We solve Prob. 2 with Alg. 2, which leverages the linear programming formulation of CMDPs in [65, Thm. 4.3]. It takes a linear programming solver as input, formulates Prob. 2 as a linear program (14), and solves it using the provided solver (line 5) [62, Thm. 6.6.1]. (Remark: The correctness of Alg. 2 is independent of Thm. 2 and therefore solves Prob. 2 even if the assumptions of Thm. 2 do not hold.)

A. Numerical Evaluation

We evaluate Algs. 1–2 on instantiations of Probs. 1–2 with different values of Δ_{R} (6b). Hyperparameters are listed in the supplementary material [27, App. E]. The computing environment for the evaluation is a server with a 24-core INTEL XEON GOLD 2.10 GHz CPU and 768 GB RAM.

For each instantiation of Prob. 1 (optimal intrusion recovery), we run Alg. 1 with four optimization algorithms: Simultaneous Perturbation Stochastic Approximation (SPSA) [68, Fig. 1], Bayesian Optimization (BO) [71, Alg. 1], Cross Entropy Method (CEM) [69, Alg. 1], and Differential Evolution (DE) [70, Fig. 3]. We compare the results with that of two baselines: Incremental Pruning (IP) [73, Fig. 4], which is a dynamic programming algorithm, and Proximal Policy Optimization (PPO) [72, Alg. 1], a reinforcement learning algorithm. The results are shown in Table 2 and Figs. 7–9.

We observe in the first three rows of Table 2 that most of the algorithms that utilize Thm. 1 find near-optimal recovery strategies for all Δ_{R}. By contrast, IP becomes computationally intractable as $\Delta_{\mathrm{R}} \to \infty$ (bottom row of Table 2).

The convergence times are shown in Figs. 7–8. We observe that CEM, BO, DE, and PPO find near-optimal strategies within an hour of computation, whereas SPSA does not converge. This is probably due to a poor selection of hyperparameters.

Lastly, Fig. 9 shows the performance of Alg. 2. We see that Alg. 2 solves Prob. 2 (optimal replication factor) within 2 minutes for systems with up to 2048 nodes.

Method	$\Delta_{\mathrm{R}}=5$ Time (min)	$\Delta_{\mathrm{R}}=5$ J_i (5)	$\Delta_{\mathrm{R}}=15$ Time (min)	$\Delta_{\mathrm{R}}=15$ J_i (5)	$\Delta_{\mathrm{R}}=25$ Time (min)	$\Delta_{\mathrm{R}}=25$ J_i (5)	$\Delta_{\mathrm{R}}=\infty$ Time (min)	$\Delta_{\mathrm{R}}=\infty$ J_i (5)
CEM [69, Alg. 1]	**1.04**	$\mathbf{0.12 \pm 0.01}$	**8.84**	$\mathbf{0.17 \pm 0.06}$	**14.48**	0.19 ± 0.08	11.81	0.16 ± 0.01
DE [70, Fig. 3]	2.35	$\mathbf{0.12 \pm 0.03}$	8.98	$\mathbf{0.17 \pm 0.01}$	15.45	$\mathbf{0.18 \pm 0.02}$	22.68	0.16 ± 0.01
BO [71, Alg. 1]	29.18	$\mathbf{0.12 \pm 0.02}$	62.57	$\mathbf{0.17 \pm 0.05}$	90.26	$\mathbf{0.18 \pm 0.12}$	9.07	$\mathbf{0.15 \pm 0.06}$
SPSA [68, Fig. 1]	10.78	0.18 ± 0.01	88.35	0.58 ± 0.40	123.85	0.77 ± 0.48	**4.20**	0.20 ± 0.02
PPO [72, Alg. 1]	28.20	0.18 ± 0.01	30.01	0.19 ± 0.02	30.33	0.21 ± 0.07	28.95	$0.21 + \pm 0.09$
IP [73, Fig. 4]	11.11	**0.12**	237.06	**0.17**	743.73	**0.18**	> 10000	not converged

TABLE 2: Solving Prob. 1 (optimal intrusion recovery) using Alg. 1 (upper rows) and baselines (lower rows); columns represent Δ_{R}; subcolumns indicate the computational time (left) and the average cost (right); numbers indicate the mean and the 95% confidence interval based on 20 random seeds.

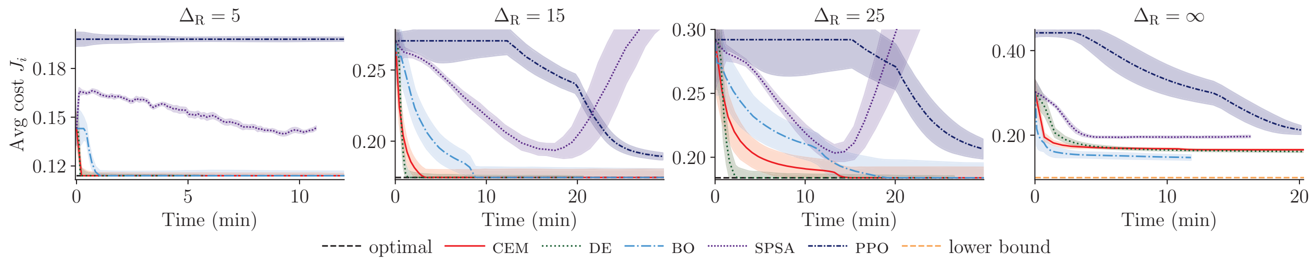


Fig. 7: Convergence curves of Alg. 1 for Prob. 1 (optimal intrusion recovery); the curves show the mean value from evaluations with 20 random seeds and the shaded areas indicate the 95% confidence interval divided by 10.

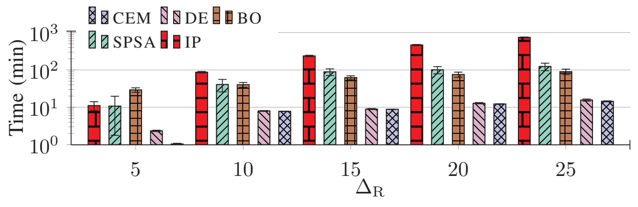


Fig. 8: Mean compute time to solve Prob. 1 for different algorithms and values of Δ_{R}; the error bars indicate the 95% confidence interval based on 20 measurements.

Server	*Processors*	RAM (GB)
1, R715 2U	two 12-core AMD OPTERON	64
2, R715 2U	two 12-core AMD OPTERON	64
3, R715 2U	two 12-core AMD OPTERON	64
4, R715 2U	two 12-core AMD OPTERON	64
5, R715 2U	two 12-core AMD OPTERON	64
6, R715 2U	two 12-core AMD OPTERON	64
7, R715 2U	two 12-core AMD OPTERON	64
8, R715 2U	two 12-core AMD OPTERON	64
9, R715 2U	two 12-core AMD OPTERON	64
10, R630 2U	two 12-core INTEL XEON E5-2680	256
11, R740 2U	1 20-core INTEL XEON GOLD5218R	32
12, SUPERMICRO 7049	2 TESLA P100, 1 16-core INTEL XEON	126
13, SUPERMICRO 7049	4 RTX 8000, 1 24-core INTEL XEON	768

TABLE 3: Specifications of the physical nodes.

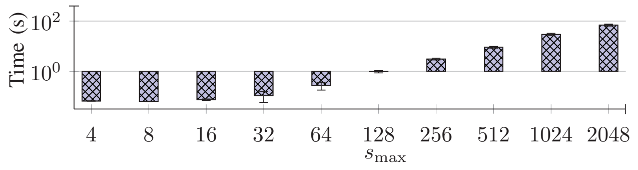


Fig. 9: Mean compute time to solve Prob. 2 (optimal replication factor) with Alg. 2; error bars indicate the 95% confidence interval based on 20 measurements.

VII. Testbed Implementation of TOLERANCE

We implement TOLERANCE as a proof-of-concept on a testbed. The implementation includes three layers.

A. The Physical Layer

The physical layer contains a cluster with 13 nodes connected through an Ethernet network. Specifications of the nodes can be found in Table 3.

Nodes communicate via message passing over authenticated channels. Each node runs (*i*) a service replica in a Docker container [74]; (*ii*) a node controller (§IV); and (*iii*) the SNORT IDS with ruleset v2.9.17.1 [75].

B. The Virtualization Layer

Each service replica runs a web service [76]. The service offers two deterministic operations: (*i*) a *read operation*, which returns the current state of the service; and (*ii*) a *write operation*, which updates the state. To coordinate these operations, replicas run reconfigurable MINBFT [43, §4.2]. The throughput of our implementation of MINBFT is shown in Fig. 14. The source code and a description of our implementation is available in the supplementary material [27, App. G].

Clients access the service by issuing requests that are sent to all replicas. Each request has a unique identifier that is digitally signed. After sending a request, the client waits for a quorum of $f+1$ identical replies with valid signatures [43]. (Remark: a quorum is necessary to guarantee that the response is correct since the client does not know which replicas are compromised (Prop. 1).)

C. The Control Layer

Node controllers collect IDS alerts and decide when to recover service replicas. When a replica is recovered, it starts with a new container and its state is initialized with the (identical) state from $f+1$ other replicas [20], [77].

The system controller is implemented by a crash-tolerant system that runs the RAFT protocol [53]. When it decides to evict or add a node, it triggers a view change in MINBFT. Figure 12.a illustrates the strategy of the system controller.

VIII. Evaluation of TOLERANCE

In this section, we evaluate our implementation of TOLERANCE and compare it with state-of-the-art intrusion-tolerant systems.

A. Evaluation Setup

An evaluation run evolves in time-steps of 60 seconds. It starts with N_1 nodes from Table 3, each of which runs a service replica (see Table 4). At each time-step, one or more replicas may be recovered by the node controllers and a new node may be added by the system controller. When a replica is recovered, its container is replaced with a container selected randomly from the list in Table 4. Similarly, when a new node is added, a node from the list in Table 3 is started.

Each replica has one or more vulnerabilities that can be exploited by the attacker using the steps listed in Table 6. After compromising a replica, the attacker randomly chooses between: a) participating in the consensus protocol; b) not participating; and c) participating with randomly selected messages.

Replicas are interconnected through Gbit/s connections with 0.05% packet loss (emulated with NETEM [78]). They receive a stream of service requests, which are sent by a client over 100 Mbit/s connections with 0.1% packet loss.

To emulate IDS events for a realistic system, each replica runs a set of background services in addition to the replicated service (see Table 5). These background services are consumed by a population of background clients, who arrive with a Poisson rate $\lambda = 20$ and have exponentially distributed service times with mean $\mu = 4$ time-steps.

All parameters for the evaluation are listed in the supplementary material [27, App. E] except for Z_i (3), which we estimate with the empirical distribution $\widehat{Z}_i$ (see Fig. 10). We compute $\widehat{Z}_i$ based on $M = 25,000$ samples [27], knowing that $\widehat{Z}_i \overset{\text{a.s.}}{\to} Z_i$ as $M \to \infty$ (Glivenko-Cantelli theorem).

In practice, $\widehat{Z}_i$ may be implemented using any statistical intrusion detection method (e.g., anomaly detection [23]). Similarly, the model parameters (e.g., the probability that a node crashes) can be defined based on domain knowledge or based on system measurements [58], [59].

Replica ID	*Operating system*	*Vulnerabilities*
1	UBUNTU 14	FTP weak password
2	UBUNTU 20	SSH weak password
3	UBUNTU 20	TELNET weak password
4	DEBIAN 10.2	CVE-2017-7494
5	UBUNTU 20	CVE-2014-6271
6	DEBIAN 10.2	CWE-89 on DVWA [79]
7	DEBIAN 10.2	CVE-2015-3306
8	DEBIAN 10.2	CVE-2016-10033
9	DEBIAN 10.2	CVE-2010-0426, SSH weak password
10	DEBIAN 10.2	CVE-2015-5602, SSH weak password

TABLE 4: Containers running the service replicas.

Background services	*Replica ID(s)*
FTP, SSH, MONGODB, HTTP, TEAMSPEAK	1
SSH, DNS, HTTP	2
SSH, TELNET, HTTP	3
SSH, SAMBA, NTP	4
SSH	5, 7, 8, 10
DVWA, IRC, SSH	6
TEAMSPEAK, HTTP, SSH	9

TABLE 5: Background services of the service replicas.

B. Baseline Control Strategies

We compare the control strategies of TOLERANCE with those used in current intrusion-tolerant systems, whereby we choose three baseline strategies: NO-RECOVERY, PERIODIC and PERIODIC-ADAPTIVE. The first baseline, NO-RECOVERY, does not recover or add any nodes, which corresponds to the strategy used in traditional intrusion-tolerant systems, such as RAMPART [24] and SECURE-RING [81]. The second baseline, PERIODIC, recovers nodes every Δ_{R} time-steps but does not add any new nodes. This is the strategy used in most of the intrusion-tolerant systems proposed in prior work, including PBFT [22], VM-FIT [19], [20], WORM-IT [29], PRRW [82], [83], MAFTIA [84], RECOVER [85], SCIT [86], [87], COCA [88], SPIRE [21], ITCIS-PRR [89], CRUTIAL [90], SBFT [91], BFT-SMART [92], UPRIGHT [93], and SKYNET [94]. The third baseline, PERIODIC-ADAPTIVE, recovers nodes every Δ_{R} time-steps and adds a node when $o_{i,t} \geq 2\mathbb{E}[O_t]$ (3), which approximates the heuristic strategies used in [95], SITAR [96], ITSI [97], and ITUA [98], [99].

Replica ID	*Intrusion steps*
1	TCP SYN scan, FTP brute force
2	TCP SYN scan, SSH brute force
3	TCP SYN scan, TELNET brute force
4	ICMP scan, exploit of CVE-2017-7494
5	ICMP scan, exploit of CVE-2014-6271
6	ICMP scan, exploit of CWE-89 on on DVWA [79]
7	ICMP scan, exploit of CVE-2015-3306
8	ICMP scan, exploit of CVE-2016-10033
9	ICMP scan, SSH brute force, exploit of CVE-2010-0426
10	ICMP scan, SSH brute force, exploit of CVE-2015-5602

TABLE 6: Steps to compromise service replicas.

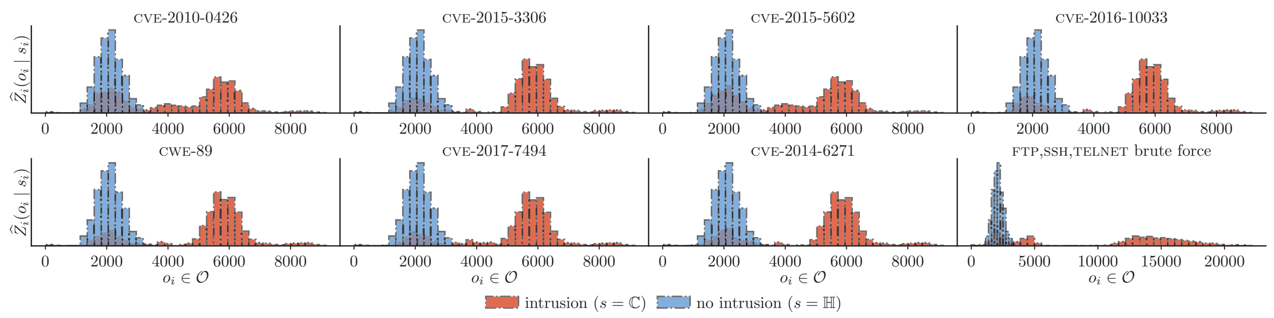


Fig. 10: Empirical distributions $\widehat{Z}_1(\cdot \mid s_i), \ldots, \widehat{Z}_{10}(\cdot \mid s_i)$ as estimates of $Z_1, \ldots, Z_{10}$ (3) for the containers in Table 4.

Control strategy	$\Delta_R = 15$			$\Delta_R = 25$			$\Delta_R = \infty$		
	$T^{(A)}$	$T^{(R)}$	$F^{(R)}$	$T^{(A)}$	$T^{(R)}$	$F^{(R)}$	$T^{(A)}$	$T^{(R)}$	$F^{(R)}$
				$N_1 = 3$					
TOLERANCE	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.43 \pm 0.09}$	0.09 ± 0.01	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.43 \pm 0.09}$	0.09 ± 0.01	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.43 \pm 0.09}$	0.09 ± 0.01
NO-RECOVERY	0.08 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.08 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.08 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC	0.97 ± 0.01	6.06 ± 1.16	0.065 ± 0.01	0.93 ± 0.01	8.64 ± 1.48	0.04 ± 0.01	0.08 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC-ADAPTIVE	0.95 ± 0.02	5.42 ± 0.93	0.05 ± 0.01	0.94 ± 0.02	6.57 ± 1.01	0.03 ± 0.01	0.09 ± 0.04	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
				$N_1 = 6$					
TOLERANCE	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.47 \pm 0.07}$	0.07 ± 0.01	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.47 \pm 0.07}$	0.07 ± 0.01	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.47 \pm 0.07}$	0.07 ± 0.01
NO-RECOVERY	0.16 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.16 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.16 ± 0.06	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC	0.98 ± 0.01	5.96 ± 1.16	0.065 ± 0.01	0.95 ± 0.02	8.13 ± 1.48	0.04 ± 0.01	0.16 ± 0.03	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC-ADAPTIVE	$\mathbf{0.99 \pm 0.01}$	5.02 ± 0.34	0.06 ± 0.01	0.97 ± 0.02	6.16 ± 0.54	0.03 ± 0.01	0.17 ± 0.03	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
				$N_1 = 9$					
TOLERANCE	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.44 \pm 0.05}$	0.07 ± 0.01	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.44 \pm 0.05}$	0.07 ± 0.01	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.44 \pm 0.05}$	0.07 ± 0.01
NO-RECOVERY	0.17 ± 0.04	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.17 ± 0.04	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$	0.17 ± 0.04	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC	0.99 ± 0.00	5.37 ± 0.34	0.06 ± 0.01	0.98 ± 0.01	7.74 ± 0.51	0.04 ± 0.01	0.17 ± 0.04	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$
PERIODIC-ADAPTIVE	$\mathbf{1.00 \pm 0.00}$	4.44 ± 0.25	0.06 ± 0.01	0.99 ± 0.01	6.01 ± 0.39	0.04 ± 0.01	0.18 ± 0.02	$10^3 \pm 0.00$	$\mathbf{0.00 \pm 0.00}$

TABLE 7: Comparison between TOLERANCE and the baselines (§VIII-B); columns indicate values of Δ_R; subcolumns represent performance metrics; row groups relate to the number of initial nodes N_1; numbers indicate the mean and the 95% confidence interval from evaluations with 20 random seeds.

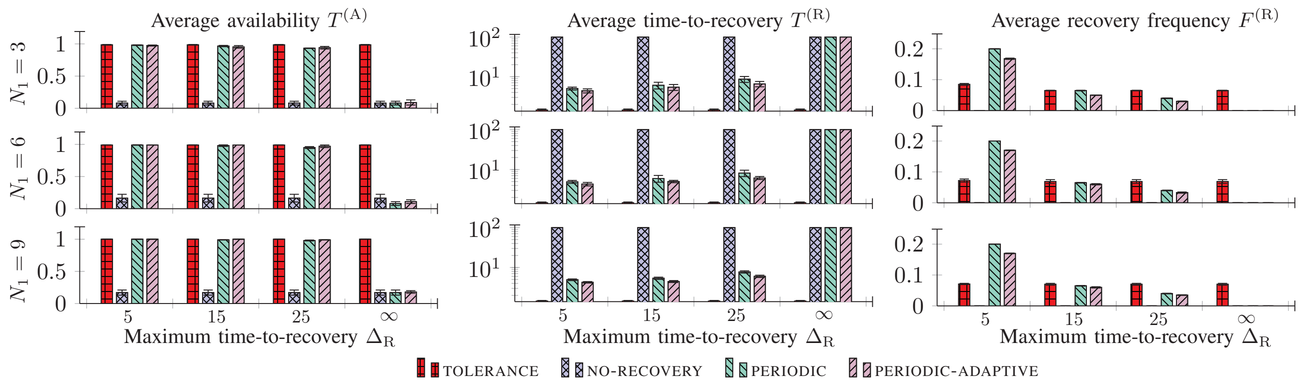


Fig. 11: Comparison between TOLERANCE and the baselines (§VIII-B); columns represent performance metrics; x-axes indicate values of Δ_R; rows relate to the number of initial nodes N_1; bars show the mean and error bars indicate the 95% confidence interval from evaluations with 20 random seeds.

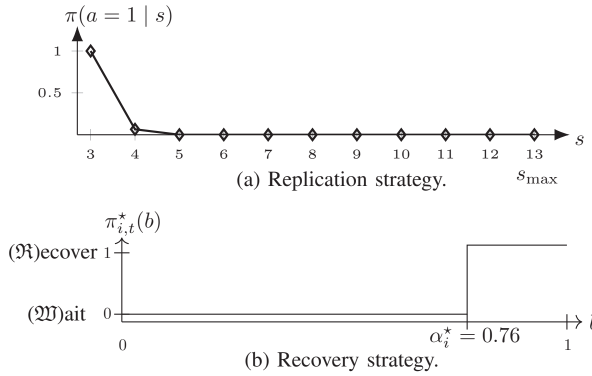


Fig. 12: Illustration of the replication and recovery strategies for $\Delta_{\mathrm{R}} = \infty$, $N_1 = 6$, and $f = 1$.

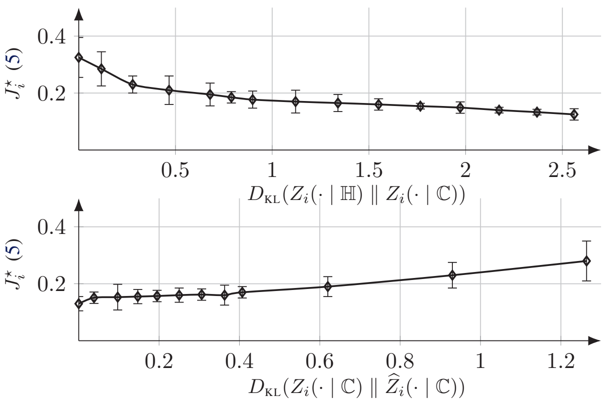


Fig. 13: Optimal recovery cost (5) in function of the accuracy of the intrusion detection model Z_i (3); D_{KL} refers to the Kullback-Leibler (KL) divergence [80].

C. Evaluation Results

The results are summarized in Fig. 11 and Table 7. The control strategies are illustrated in Fig. 12 and the sensitivity of the controllers with respect to the intrusion detection model Z_i (3) is shown in Fig. 13.

The red bars in Fig. 11 relate to TOLERANCE. The blue, green, and pink bars relate to the baselines. The leftmost column in Fig. 11 shows the average availability for different values of Δ_{R}. We observe that TOLERANCE has close to 100% service availability in all of the cases we studied. By contrast, NO-RECOVERY has close to 0% availability. The availability achieved by PERIODIC and PERIODIC-ADAPTIVE is in-between; they perform similar to TOLERANCE when Δ_{R} is small (i.e., when recoveries are frequent) and similar to NO-RECOVERY when $\Delta_{\mathrm{R}} \to \infty$. We note that increasing N_1 from 3 to 9 doubles the availability of NO-RECOVERY but has a negligible impact on the availability of TOLERANCE, PERIODIC, and PERIODIC-ADAPTIVE.

The second leftmost column in Fig. 11 shows the average time-to-recovery $T^{(\mathrm{R})}$. We observe that $T^{(\mathrm{R})}$ of TOLERANCE is an order of magnitude smaller than that of PERIODIC and PERIODIC-ADAPTIVE and two orders of magnitude smaller than that of NO-RECOVERY. This result demonstrates the benefit of feedback control, which allows the system to react promptly to intrusions.

Finally, the rightmost column in Fig. 11 shows the average frequency of recoveries $F^{(\mathrm{R})}$. We note that $F^{(\mathrm{R})}$ of TOLERANCE is about the same as PERIODIC and PERIODIC-ADAPTIVE when $\Delta_{\mathrm{R}} = 15$. Interestingly, when $\Delta_{\mathrm{R}} = 5$, TOLERANCE has both a smaller $F^{(\mathrm{R})}$ and a smaller $T^{(\mathrm{R})}$ than PERIODIC and PERIODIC-ADAPTIVE.

D. Discussion of the Evaluation Results

The key findings from our evaluation of the TOLERANCE architecture can be summarized as follows:

(i) TOLERANCE can achieve a lower time-to-recovery and a higher service availability than state-of-the-art intrusion-tolerant systems (Table 7, Fig. 11). The BTR constraint (6b) guarantees that TOLERANCE never has a worse time-to-recovery than current systems. The performance of TOLERANCE depends on the accuracy of the intrusion detection model $\widehat{Z}_i$ (see Fig. 13 and Fig. 10).

(ii) The solutions to both control problems of the TOLERANCE architecture have threshold properties (Thms. 1–2, Cor. 1), which enable efficient computation of optimal strategies (Figs. 7–9, Algs. 1–2).

(iii) The benefit of using an adaptive replication strategy as opposed to a static strategy is mainly prominent when node crashes are frequent (see Fig. 12 and cf. the results of PERIODIC and PERIODIC-ADAPTIVE in Fig. 11.)

While the results demonstrate clear benefits of TOLERANCE compared to current intrusion-tolerant systems, TOLERANCE has two drawbacks. First, the performance of TOLERANCE depends on the accuracy of the intrusion detection model $\widehat{Z}_i$ (see Fig. 10 and Fig. 13). This means that practical deployments of TOLERANCE require a statistical intrusion detection model for estimating the probability of intrusion (4). This model can be realized in many ways. It can for example be based on anomaly detection methods or machine learning techniques. Further, the detection model can use different types of data sources, e.g., log files, IDS alerts, threat intelligence sources, etc. Our proof-of-concept implementation of TOLERANCE uses the SNORT IDS as the data source and obtains the distribution of IDS alerts using maximum likelihood estimation, which allows to compute the probability of intrusion (4).

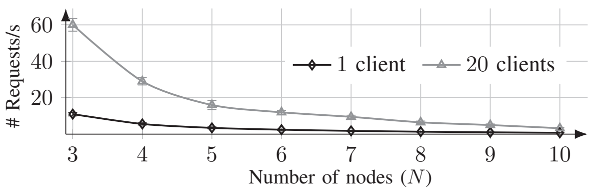


Fig. 14: Average throughput of our implementation of MINBFT; error bars indicate the 95% confidence interval based on 1000 samples.

Second, TOLERANCE is vulnerable to an attack where a large amount of false IDS alerts trigger excess recoveries. Analysis of such attacks requires a game-theoretic treatment, whereby problems 1–2 are modified to take into account how an attacker may exploit the control strategies (e.g., minimax problem formulations [100]). We plan to investigate such problem formulations in future work (see §X).

IX. Related Work

Intrusion tolerance is studied in several broad areas of research, including: Byzantine fault tolerance [18], dependability [101], [102], reliability [103], survivability [104], and cyber resilience [4], [105]–[109]. This research effort has led to many mechanisms for implementing intrusion-tolerant systems, such as: intrusion-tolerant consensus protocols [18], [91]–[93], [101]–[103], [110]–[112], software diversification schemes [77], geo-replication schemes [113], cryptographic mechanisms [56], [57], and defenses against denial of service [30]. These mechanisms provide the foundation for TOLERANCE, which adds automated recovery and replication control.

While TOLERANCE builds on all of the above works, we limit the following discussion to explain how TOLERANCE differs from current intrusion-tolerant systems and how it relates to prior work that uses feedback control.

A. Intrusion-Tolerant Systems

Existing intrusion-tolerant systems include PBFT [22], ZYZZYVA [114], HQ [115], HOTSTUFF [111], VM-FIT [19], [20], WORM-IT [29], PRRW [82], RECOVER [85], SCIT [86], [87], COCA [88], [95], SPIRE [21], ITCIS-PRR [89], CRUTIAL [90], UPRIGHT [93], BFT-SMART [92], SBFT [91], SITAR [96], ITUA [98], [99], MAFTIA [84], ITSI [97], and SKYNET [94]. All of them are based on intrusion-tolerant consensus protocols and support recovery, either directly or indirectly through external recovery services, like PHOENIX [116]. TOLERANCE differs from these systems in two main ways.

First, TOLERANCE uses feedback control to decide when to perform intrusion recovery. This contrasts with all of the referenced systems, which either use periodic or heuristic recovery schemes. (PRRW, RECOVER, CRUTIAL, SCIT, SITAR, ITSI, and ITUA can be implemented with feedback-based recovery but they do not specify how to implement such recovery strategies.)

Second, TOLERANCE uses an adaptive replication strategy. In comparison, all of the referenced systems use static replication strategies except SITAR, [95], ITUA, and ITSI, who implement adaptive replication based on time-outs and static rules as opposed to feedback control. The benefit of feedback control is that it allows the system to adapt promptly to intrusions, not having to wait for a time-out.

B. Intrusion Response through Feedback Control

Intrusion response through feedback control is an active area of research that uses concepts and methods from various emergent and traditional fields. Most notably from reinforcement learning (see examples [31], [64], [100], [109], [117]–[125]), control theory (see examples [104], [126]–[132]), causal inference (see example [133]), game theory (see examples [134]–[139]), natural language processing (see example [140]), evolutionary computation (see example [141]), and general optimization (see examples [142], [143]). While these works have obtained promising results, none of them consider the integration with intrusion-tolerant systems as we do in this paper. Another drawback of the existing solutions is that many of them are inefficient and lack safety guarantees. Finally, and most importantly, nearly all of the above works are limited to simulation environments and it is not clear how they generalize to practical systems. In contrast, TOLERANCE is practical: it can be integrated with existing intrusion-tolerant systems, it satisfies safety constraints, and it is computationally efficient.

X. Conclusion and Future Work

This paper presents TOLERANCE: a novel control architecture for intrusion-tolerant systems that uses two levels of control to decide when to perform recovery and when to increase the replication factor. These control problems can be formulated as two classical problems in operations research, namely, the machine replacement problem and the inventory replenishment problem. Using this formulation, we prove that the optimal control strategies have threshold structure (Thms. 1–2, Cor. 1) and we design efficient algorithms for computing them (Algs. 1–2, Table 2). We evaluate TOLERANCE in an emulation environment where we run 10 types of network intrusions. The results demonstrate that TOLERANCE improves service availability and reduces operational cost when compared with state-of-the-art intrusion-tolerant systems in the scenarios we studied (Fig. 11, Table 7). The improvement of TOLERANCE with respect to current systems comes at the expense of a training phase, where we first fit an intrusion detection model (Fig. 10) and then train the controllers based on this model (Figs. 7–9).

We plan to continue this work in several directions. First, we will improve our implementation by integrating a high-performance consensus protocol, e.g., HOTSTUFF-M [44]. Second, we intend to extend our control-theoretic model of the TOLERANCE architecture to a game-theoretic model, which allows us to study defenses against dynamic attackers. Third, we plan to investigate methods for online learning of intrusion detection models and online adaptation of control strategies.

References

[1] A. Aviziens, "Fault-tolerant systems," *IEEE Transactions on Computers*, vol. C-25, no. 12, pp. 1304–1312, 1976.

[2] F. Cristian, "Understanding fault-tolerant distributed systems," *Communications of the ACM*, vol. 34, no. 2, pp. 56–78, 1991.

[3] W. economic forum, "Global cybersecurity outlook 2022," 2022, https://www.weforum.org/publications/global-cybersecurity-outlook-2022/.

[4] A. A. Ganin, E. Massaro, A. Gutfraind, N. Steen, J. M. Keisler, A. Kott, R. Mangoubi, and I. Linkov, "Operational resilience: concepts, design and analysis," *Scientific Reports*, vol. 6, no. 1, p. 19540, 2016.

[5] A. Babay, J. Schultz, T. Tantillo, S. Beckley, E. Jordan, K. Ruddell, K. Jordan, and Y. Amir, "Deploying intrusion-tolerant scada for the power grid," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 328–335.

[6] B. Wolff and P. Scheffers, “Dms-r, the brain of the iss: 10 years of continuous successful operation in space,” 2012, proceedings of DASIA 2012 DAta Systems In Aerospace, held on 14-16 May 2012.

[7] J. Sims, “Redundancy management software services for seawolf ship control system,” in *Proceedings of IEEE 27th International Symposium on Fault Tolerant Computing*, 1997, pp. 390–394.

[8] J. H. Lala, *A Byzantine resilient fault tolerant computer for nuclear power plant applications*. United States: IEEE Service Center, 1986.

[9] J. Wensley, L. Lamport, J. Goldberg, M. Green, K. Levitt, P. Melliar-Smith, R. Shostak, and C. Weinstock, “Sift: Design and analysis of a fault-tolerant computer for aircraft control,” *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1240–1255, 1978.

[10] J. Lala and R. Harper, “Architectural principles for safety-critical real-time applications,” *Proceedings of the IEEE*, vol. 82, no. 1, pp. 25–40, 1994.

[11] E. Sakic, N. Deric, and W. Kellerer, “Morph: An adaptive framework for efficient and byzantine fault-tolerant sdn control plane,” *IEEE J.Sel. A. Commun.*, vol. 36, no. 10, p. 2158–2174, oct 2018.

[12] A. Nogueira, M. Garcia, A. Bessani, and N. Neves, “On the challenges of building a bft scada,” in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 163–170.

[13] J. Kirsch, S. Goose, Y. Amir, D. Wei, and P. Skare, “Survivable scada via intrusion-tolerant replication,” *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 60–70, 2014.

[14] J. Soikkeli, G. Casale, L. Muñoz-González, and E. C. Lupu, “Redundancy planning for cost efficient resilience to cyber attacks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 1154–1168, 2023.

[15] Y. Deswarte, L. Blain, and J.-C. Fabre, “Intrusion tolerance in distributed computing systems,” in *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society, 1991, pp. 110–110.

[16] P. E. Veríssimo, N. F. Neves, and M. P. Correia, “Intrusion-tolerant architectures: Concepts and design,” in *Architecting Dependable Systems*, R. de Lemos, C. Gacek, and A. Romanovsky, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 3–36.

[17] F. Wang, R. Uppalli, and C. Killian, “Analysis of techniques for building intrusion tolerant server systems,” in *IEEE Military Communications Conference, 2003. MILCOM 2003.*, vol. 2, 2003, pp. 729–734 Vol.2.

[18] T. Distler, “Byzantine fault-tolerant state-machine replication from a systems perspective,” *ACM Comput. Surv.*, vol. 54, no. 1, feb 2021.

[19] T. Distler, R. Kapitza, and H. Reiser, “State transfer for hypervisor-based proactive recovery of heterogeneous replicated services,” *FERS-Mitteilungen*, vol. 29, no. 1, 2011.

[20] H. P. Reiser and R. Kapitza, “Hypervisor-based efficient proactive recovery,” in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, 2007, pp. 83–92.

[21] A. Babay, T. Tantillo, T. Aron, M. Platania, and Y. Amir, “Network-attack-resilient intrusion-tolerant scada for the power grid,” in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 255–266.

[22] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, p. 398–461, nov 2002.

[23] A. Fuchsberger, “Intrusion detection systems and intrusion prevention systems,” *Inf. Secur. Tech. Rep.*, vol. 10, no. 3, p. 134–139, Jan. 2005.

[24] M. K. Reiter, “The rampart toolkit for building high-integrity services,” in *Theory and Practice in Distributed Systems*, K. P. Birman, F. Mattern, and A. Schiper, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 99–110.

[25] B. A. Kalymon, “Machine replacement with stochastic costs,” *Management Science*, vol. 18, no. 5, pp. 288–298, 1972.

[26] W. A. Donaldson, “Inventory replenishment policy for a linear trend in demand – an analytical solution,” *Operational Research Quarterly (1970-1977)*, vol. 28, no. 3, pp. 663–670, 1977.

[27] K. Hammar and R. Stadler, “Supplementary Material for ”Intrusion Tolerance for Networked Systems Through Two-Level Feedback Control”,” Dec. 2023, https://doi.org/10.5281/zenodo.10215950, https://arxiv.org/abs/2404.01741.

[28] CSLE, “Cyber security learning environment,” 2023, documentation: https://limmen.dev/csle/, traces: https://github.com/Limmen/csle/releases/tag/v0.4.0, source code: https://github.com/Limmen/csle, installation: https://www.youtube.com/watch?v=l_g3sRJwwhc.

[29] M. Correia, N. F. Neves, L. C. Lung, and P. Veríssimo, “Worm-it – a wormhole-based intrusion-tolerant group communication system,” *Journal of Systems and Software*, vol. 80, no. 2, pp. 178–197, 2007.

[30] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, “Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods,” *IEEE Access*, vol. 7, pp. 51 691–51 713, 2019.

[31] K. Hammar and R. Stadler, “Intrusion prevention through optimal stopping,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2333–2348, 2022.

[32] J. von Neumann, “Probabilistic logics and synthesis of reliable organisms from unreliable components,” in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956, pp. 43–98.

[33] E. Moore and C. Shannon, “Reliable circuits using less reliable relays,” *Journal of the Franklin Institute*, vol. 262, no. 3, pp. 191–208, 1956.

[34] A. Avizienis, “Design methods for fault-tolerant navigation computers,” Tech. Rep., 1969.

[35] A. Avižienis, “Design of fault-tolerant computers,” in *Proceedings of the November 14-16, 1967, fall joint computer conference*, 1967, pp. 733–743.

[36] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, jul 1982.

[37] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming*, 2nd ed. Springer Publishing Company, Incorporated, 2011.

[38] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the presence of partial synchrony,” *J. ACM*, vol. 35, no. 2, p. 288–323, apr 1988.

[39] D. Dolev, C. Dwork, and L. Stockmeyer, “On the minimal synchronism needed for distributed consensus,” *J. ACM*, vol. 34, no. 1, p. 77–97, jan 1987.

[40] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2004.

[41] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *J. ACM*, vol. 32, no. 2, p. 374–382, apr 1985.

[42] G. Bracha and S. Toueg, “Asynchronous consensus and broadcast protocols,” *J. ACM*, vol. 32, no. 4, p. 824–840, oct 1985.

[43] G. T. dos Santos Veronese, “Intrusion tolerance in large scale networks,” Ph.D. dissertation, 2010, universidade de Lisboa.

[44] S. Yandamuri, I. Abraham, K. Nayak, and M. K. Reiter, “Communication-efficient bft protocols using small trusted hardware to tolerate minority corruption,” Cryptology ePrint Archive, Paper 2021/184, 2021, https://eprint.iacr.org/2021/184.

[45] J. Katz and C.-Y. Koo, “On expected constant-round protocols for byzantine agreement,” *Journal of Computer and System Sciences*, vol. 75, pp. 445–462, 09 2006.

[46] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, “Synchronous byzantine agreement with expected o(1) rounds, expected communication, and optimal resilience,” in *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers*. Berlin, Heidelberg: Springer-Verlag, 2019, p. 320–334.

[47] Y. Deswarte, L. Blain, and J.-C. Fabre, “Intrusion tolerance in distributed computing systems,” in *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, 1991, pp. 110–121.

[48] K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B. Muthusamy, “Characterizing intrusion tolerant systems using a state transition model,” in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX’01*, vol. 2, 2001, pp. 211–221 vol.2.

[49] K. P. Birman, “Building secure and reliable network applications,” in *Worldwide Computing and Its Applications*, T. Masuda, Y. Masunaga, and M. Tsukamoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 15–28.

[50] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” *SIGACT News*, vol. 33, no. 2, p. 51–59, jun 2002.

[51] D. Dolev and H. R. Strong, “Authenticated algorithms for byzantine agreement,” *SIAM Journal on Computing*, vol. 12, no. 4, 1983.

[52] R. Barlow, F. Proschan, and L. Hunter, *Mathematical Theory of Reliability*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1996.

[53] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC'14. USA: USENIX Association, 2014, p. 305–320.

[54] M. Dam, R. Guanciale, N. Khakpour, H. Nemati, and O. Schwarz, "Formal verification of information flow security for a simple arm-based separation kernel," in *:*, 2013, qc 20131218.

[55] M. Garcia, A. Bessani, I. Gashi, N. Neves, and R. Obelheiro, "Os diversity for intrusion tolerance: Myth or reality?" in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, 2011, pp. 383–394.

[56] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin, "Confidential byzantine Fault-Tolerance." Boston, MA: USENIX Association, Dec. 2002.

[57] R. Padilha and F. Pedone, "Belisarius: Bft storage with confidentiality," in *2011 IEEE 10th International Symposium on Network Computing and Applications*, 2011, pp. 9–16.

[58] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2010.

[59] J. Meza, T. Xu, K. Veeraraghavan, and O. Mutlu, "A large scale study of data center network reliability," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 393–407.

[60] R. Douc, E. Moulines, P. Priouret, and P. Soulier, *Markov Chains*, ser. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2018.

[61] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[62] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.

[63] N. Gandhi, E. Roth, B. Sandler, A. Haeberlen, and L. T. X. Phan, "Rebound: Defending distributed systems against attacks with bounded-time recovery," in *Proceedings of the Sixteenth European Conference on Computer Systems*, ser. EuroSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 523–539.

[64] K. Hammar and R. Stadler, "Learning near-optimal intrusion responses against dynamic attackers," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[65] E. Altman, *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[66] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Math. Oper. Res.*, vol. 12, p. 441–450, Aug. 1987.

[67] L. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.

[68] J. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, 1998.

[69] R. J. Moss, "Cross-entropy method variants for optimization," 2020, https://arxiv.org/abs/2009.09043.

[70] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[71] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[72] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, 2017, http://arxiv.org/abs/1707.06347.

[73] A. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable markov decision processes," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 54–61.

[74] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, p. 2, 2014.

[75] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. USA: USENIX Association, 1999, p. 229–238.

[76] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Comput. Surv.*, vol. 22, no. 4, p. 299–319, dec 1990.

[77] M. Garcia, A. Bessani, and N. Neves, "Lazarus: Automatic management of diversity in bft systems," in *Proceedings of the 20th International Middleware Conference*, ser. Middleware '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 241–254.

[78] S. Hemminger, "Network emulation with netem," *Linux Conf*, 2005.

[79] T. D. team, "Damn vulnerable web application (dvwa)," 2023, https://github.com/digininja/DVWA.

[80] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[81] K. Kihlstrom, L. Moser, and P. Melliar-Smith, "The securering protocols for securing group communication," in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, vol. 3, 1998, pp. 317–326 vol.3.

[82] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Resilient intrusion tolerance through proactive and reactive recovery," in *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, 2007, pp. 373–380.

[83] ——, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 4, pp. 452–465, 2010.

[84] R. Stroud, I. Welch, J. Warne, and P. Ryan, "A qualitative analysis of the intrusion-tolerance capabilities of the maftia architecture," in *International Conference on Dependable Systems and Networks, 2004*, 2004, pp. 453–461.

[85] P. Sousa, A. N. Bessani, and R. R. Obelheiro, "The forever service for fault/intrusion removal," in *Proceedings of the 2nd Workshop on Recent Advances on Intrusiton-Tolerant Systems*, ser. WRAITS '08. New York, NY, USA: Association for Computing Machinery, 2008.

[86] Y. Huang and A. Sood, "Self-cleansing systems for intrusion containment," in *2002 Workshop on Self-Healing, Adaptive and Self-Managed Systems*, 2002.

[87] Q. L. Nguyen and A. Sood, "Designing scit architecture pattern in a cloud-based environment," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2011, pp. 123–128.

[88] L. Zhou, F. B. Schneider, and R. Van Renesse, "Coca: A secure distributed online certification authority," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, p. 329–368, nov 2002.

[89] P. Sousa, A. N. Bessani, W. S. Dantas, F. Souto, M. Correia, and N. F. Neves, "Intrusion-tolerant self-healing devices for critical infrastructure protection," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009, pp. 217–222.

[90] A. Bessani, P. Sousa, M. Correia, N. Neves, and P. Verı́ssimo, "The crutial way of critical infrastructure protection," *Security & Privacy, IEEE*, vol. 6, pp. 44 – 51, 01 2009.

[91] G. Golan Gueta *et al.*, "Sbft: A scalable and decentralized trust infrastructure," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 568–580.

[92] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.

[93] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riche, "Upright cluster services," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, ser. SOSP '09, New York, NY, USA, 2009, p. 277–290.

[94] T. Freitas, J. Soares, M. E. Correia, and R. Martins, "Skynet: a cyber-aware intrusion tolerant overseer," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 2023, pp. 111–116.

[95] A. Saidane, V. Nicomette, and Y. Deswarte, "The design of a generic intrusion-tolerant architecture for web servers," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 45–58, 2009.

[96] S. Bryant and F. Wang, "Aspects of adaptive reconfiguration in a scalable intrusion tolerant system," *Complexity*, vol. 9, no. 2, 2003.

[97] D. O'Brien, R. Smith, T. Kappel, and C. Bitzer, "Intrusion tolerance via network layer controls," in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, 2003, pp. 90–96 vol.1.

[98] P. Pal, P. Rubel, M. Atighetchi, F. Webber, W. H. Sanders, M. Seri, H. Ramasamy, J. Lyons, T. Courtney, A. Agbaria, M. Cukier, J. Gossett, and I. Keidar, "An architecture for adaptive intrusion-tolerant applications," *Software: Practice and Experience*, vol. 36, no. 11-12, pp. 1331–1354, 2006.

[99] S. Singh, M. Cukier, and W. Sanders, "Probabilistic validation of an intrusion-tolerant replication system," in *2003 International Conference on Dependable Systems and Networks, 2003. Proceedings.*, 2003.

[100] K. Hammar and R. Stadler, "Scalable learning ofintrusion response through recursive decomposition," in *Decision and Game Theory for Security*, J. Fu, T. Kroupa, and Y. Hayel, Eds. Cham: Springer Nature Switzerland, 2023, pp. 172–192.

[101] A. Goyal, P. Shahabuddin, P. Heidelberger, V. F. Nicola, and P. W. Glynn, "A unified framework for simulating markovian models of highly dependable systems," *IEEE Transactions on Computers*, vol. 41, no. 1, pp. 36–51, 1992.

[102] A. Burns and A. M. Lister, "A framework for building dependable systems," *The Computer Journal*, vol. 34, no. 2, pp. 173–181, 1991.

[103] K. P. Birman, "The process group approach to reliable distributed computing," *Communications of the ACM*, vol. 36, no. 12, 1993.

[104] O. P. Kreidl and T. M. Frazier, "Feedback control applied to survivability: a host-based autonomic defense system," *IEEE Transactions on Reliability*, vol. 53, pp. 148–166, 2004.

[105] M. Guo and P. Bhattacharya, "Diverse virtual replicas for improving intrusion tolerance in cloud," in *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, ser. CISR '14, New York, NY, USA, 2014, p. 41–44.

[106] A. Kott, M. J. Weisman, and J. Vandekerckhove, "Mathematical modeling of cyber resilience," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*. IEEE, nov 2022.

[107] A. Kott and I. Linkov, *Cyber Resilience of Systems and Networks*, 1st ed. Springer Publishing Company, Incorporated, 2018.

[108] J. E. Ellis, T. W. Parker, J. Vandekerckhove, B. J. Murphy, S. Smith, A. Kott, and M. J. Weisman, "An experimentation infrastructure for quantitative measurements of cyber resilience," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*. IEEE, nov 2022.

[109] Y. Huang, L. Huang, and Q. Zhu, "Reinforcement learning for feedback-enabled cyber resilience," *Annual Reviews in Control*, 2022.

[110] T. Distler, C. Cachin, and R. Kapitza, "Resource-efficient byzantine fault tolerance," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2807–2819, 2016.

[111] M. Yin *et al.*, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, ser. PODC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 347–356.

[112] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 564–577, 2011.

[113] M. Khan and A. Babay, "Toward intrusion tolerance as a service: Confidentiality in partially cloud-based bft systems," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 14–25.

[114] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. L. Wong, "Zyzzyva: speculative byzantine fault tolerance." in *SOSP*, T. C. Bressoud and M. F. Kaashoek, Eds. ACM, 2007, pp. 45–58.

[115] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira, "Hq replication: A hybrid quorum protocol for byzantine fault tolerance," in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, ser. OSDI '06. USA: USENIX Association, 2006.

[116] T. Tran, F. Nawab, P. Alvaro, and O. Arden, "Unstick yourself: Recoverable byzantine fault tolerant services," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023, pp. 1–9.

[117] B. Li, H. Wang, and G. Feng, "Adaptive hierarchical intrusion tolerant model based on autonomic computing," in *2008 International Conference on Security Technology*, 2008, pp. 137–141.

[118] J. Janisch, T. Pevný, and V. Lisý, "Nasimemu: Network attack simulator & emulator for training agents generalizing to novel scenarios," 2023, https://arxiv.org/abs/2305.17246.

[119] K. Hammar and R. Stadler, "Finding effective security strategies through reinforcement learning and Self-Play," in *International Conference on Network and Service Management (CNSM 2020)*, Izmir, Turkey, 2020.

[120] ——, "Learning intrusion prevention policies through optimal stopping," in *International Conference on Network and Service Management (CNSM 2021)*, Izmir, Turkey, 2021, https://arxiv.org/pdf/2106.07160.pdf.

[121] ——, "An online framework for adapting security policies in dynamic it environments," in *18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 359–363.

[122] T. Li, G. Peng, Q. Zhu, and T. Başar, "The confluence of networks, games, and learning a game-theoretic framework for multiagent decision making over networks," *IEEE Control Systems Magazine*, vol. 42, no. 4, pp. 35–67, 2022.

[123] T. Kunz, C. Fisher, J. L. Novara-Gsell, C. Nguyen, and L. Li, "A multiagent cyberbattlesim for rl cyber operation agents," 2023.

[124] R. K. L. Ko, "Cyber autonomy: Automating the hacker- self-healing, self-adaptive, automatic cyber defense systems and their impact to the industry, society and national security," 2020.

[125] M. Foley, M. Wang, Z. M, C. Hicks, and V. Mavroudis, "Inroads into autonomous network defence using explained reinforcement learning," 2023.

[126] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. USA: Wiley & Sons, 2004.

[127] E. Miehling, M. Rasouli, and D. Teneketzis, *Control-Theoretic Approaches to Cyber-Security*. Springer, 2019, pp. 12–28.

[128] A. Teixeira, K. C. Sou, H. Sandberg, and K. H. Johansson, "Secure control systems: A quantitative risk management approach," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 24–45, 2015.

[129] O. P. Kreidl, "Analysis of a markov decision process model for intrusion tolerance," in *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2010, pp. 156–161.

[130] D. Armstrong, S. Carter, G. Frazier, and T. Frazier, "Autonomic defense: Thwarting automated attacks via real-time feedback control," *Complexity*, vol. 9, no. 2, pp. 41–48, 2003.

[131] P. Dash, G. Li, Z. Chen, M. Karimibiuki, and K. Pattabiraman, "Pidpiper: Recovering robotic vehicles from physical attacks," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 26–38.

[132] K. Hammar, T. Li, R. Stadler, and Q. Zhu, "Automated security response through online learning with adaptive conjectures," 2024, https://arxiv.org/abs/2402.12499.

[133] A. Andrew, S. Spillard, J. Collyer, and N. Dhir, "Developing optimal causal cyber-defence agents via cyber security simulation," in *Proceedings of the ML4Cyber workshop, ICML 2022, Baltimore, USA, July 17-23, 2022*. PMLR, 2022.

[134] T. Alpcan and T. Basar, *Network Security: A Decision and Game-Theoretic Approach*, 1st ed. USA: Cambridge University Press, 2010.

[135] M. Tambe, *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*, 1st ed. USA: Cambridge University Press, 2011.

[136] C. J. Fung and R. Boutaba, *Intrusion Detection Networks - A Key to Collaborative Security*. CRC Press, 2013.

[137] L. Buttyan and J.-P. Hubaux, *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. USA: Cambridge University Press, 2007.

[138] P. Lau, W. Wei, L. Wang, Z. Liu, and C.-W. Ten, "A cybersecurity insurance model for power system reliability considering optimal defense resource allocation," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4403–4414, 2020.

[139] S. Zonouz *et al.*, "Rre: A game-theoretic intrusion response and recovery engine," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, 2009.

[140] M. Rigaki, O. Lukáš, C. A. Catania, and S. Garcia, "Out of the cage: How stochastic parrots win in cyber security environments," 2023, https://arxiv.org/abs/2308.12086.

[141] U.-M. O'Reilly and E. Hemberg, "An artificial coevolutionary framework for adversarial ai," in *AAAI Fall Symposium: ALEC*, 2018.

[142] Y. Zhang, L. Wang, and Y. Xiang, "Power system reliability analysis with intrusion tolerance in scada systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 669–683, 2016.

[143] H. Al-Hamadi and I.-R. Chen, "Redundancy management of multipath routing for intrusion tolerance in heterogeneous wireless sensor networks," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 189–203, 2013.