SUPPLEMENTARY MATERIAL

This document provides supplementary material for the paper "*CSLE: A Reinforcement Learning Platform for Autonomous Security Management*", published at the Machine Learning and Systems (MLSys) conference 2026. It contains the following appendices.

- **Appendix A**: *Management System Frontend*.
  - This appendix provides example screenshots of the management system in CSLE.
- **Appendix B**: *Target System Configurations*.
  - This appendix provides the detailed configurations of the target systems in Fig. 9.
- **Appendix C**: *Hyperparameters*.
  - This appendix provides the hyperparameters that we use to instantiate the reinforcement learning algorithms in the experimental evaluation.
- **Appendix D**: *Simulation Models*.
  - This appendix provides details about the simulation models used in the experimental evaluation.
- **Appendix E**: *Proof of Prop. 8.1*.
  - This appendix provides the proof of Prop. 8.1.
- **Appendix F**: *Example: Observation Mapping*.
  - This appendix provides an illustrative example of how we map log events (e.g., security alerts) to observations that are input to the reinforcement learning strategies.
- **Appendix G**: *Example: Digital twin configuration*.
  - This appendix provides an example configuration of a digital twin in CSLE.
- **Appendix H**: *Artifacts appendix*.
  - This appendix provides details about the artifacts associated with this paper.

A MANAGEMENT SYSTEM FRONTEND

Three screenshots of the web interface to the management system in CSLE are shown in Figs. 13–15. A video demonstration is available at (Hammar, 2023). The backend is implemented as a REST API in Python based on the Flask framework. The frontend is implemented in JavaScript based on the React framework.

B TARGET SYSTEM CONFIGURATIONS

The configuration of the target system in Fig. 9.a is available in Table 5. (Note that each component in Fig. 9 is labeled with an identifier N_i.) Similarly, the configuration of target systems in Fig. 9.b and Fig. 9.c are available in Table 6 and Table 7, respectively.

C HYPERPARAMETERS

The hyperparameters that we use to instantiate the reinforcement learning algorithms are listed in Table 8.

Parameter(s)	*Value(s)*
Discount factor γ	0.99
SPSA (Spall, 1992)	
$c, \epsilon, \lambda, A, a$	$1, 0.101, 0.602, 100, 1$
Rollout (Bertsekas, 2021)	
Rollout horizon	20
Lookahead horizon	1
Monte-Carlo samples	20
PPO (Schulman et al., 2017)	
Learning rate, # hidden layers	$5.148 \cdot 10^{-5}$, 1
# Neurons/layer	64
# Steps between updates	2048
Batch size, discount factor γ	16, 0.99
GAE λ, clip range, entropy coefficient	$0.95, 0.2, 2 \cdot 10^{-4}$
Value coefficient, max gradient norm	0.102, 0.5

Table 8. Hyperparameters used for the experimental evaluation.

D SIMULATION MODELS

In this appendix, we detail the models used to instantiate the simulations for the use cases described in §8.

D.1 Flow Control POMDP

We formulate the flow control use case when the attacker follows a static strategy as a POMDP, which is defined by the following nine-tuple

$$\langle \mathcal{S}, \mathcal{A}, f, r, \gamma, b_1, T, \mathcal{O}, z \rangle. \tag{2}$$

The POMDP evolves in time steps from $t = 1$ to $t = T$, which constitutes one *episode*. The constant $\gamma \in [0, 1]$ is a discount factor, $\mathcal{S}$ is the set of states, and $\mathcal{A}$ is the set of actions. The initial state is drawn from $b_1 \in \Delta(\mathcal{S})$ and $f(s_{t+1} \mid s_t, a_t)$ is the probability of transitioning from state s_t to state s_{t+1} when taking action a_t. The set of observations is denoted by $\mathcal{O}$ and $z(o_t \mid s_t)$ is the *observation function*, where $o_t \in \mathcal{O}$.

ID (s)	*Operating system*	*Services*	*Vulnerabilities*
1	Ubuntu 20	Snort (community ruleset v2.9.17.1), SSH	-
2	Ubuntu 20	SSH, HTTP Erl-Pengine, DNS	Weak password
4	Ubuntu 20	HTTP, Telnet, SSH	Weak password
10	Ubuntu 20	FTP, MongoDB, SMTP, Tomcat, Teamspeak 3, SSH	Weak password
12	Debian Jessie	Teamspeak 3, Tomcat, SSH	CVE-2010-0426, Weak password
17	Debian Wheezy	Apache 2, SNMP, SSH	CVE-2014-6271
18	Debian 9.2	IRC, Apache2, SSH	SQL injection
22	Debian Jessie	ProFTPd, SSH, Apache2, SNMP	CVE-2015-3306
23	Debian Jessie	Apache2, SMTP, SSH	CVE-2016-10033
24	Debian Jessie	SSH	CVE-2015-5602, Weak password
25	Debian Jessie	Elasticsearch, Apache 2, SSH, SNMP	CVE-2015-1427
27	Debian Jessie	Samba, NTP, SSH	CVE-2017-7494
3,11,5-9	Ubuntu 20	SSH, SNMP, PostgreSQL, NTP	-
$13-16,19-21,26,28-31$	Ubuntu 20	NTP, IRC, SNMP, SSH, PostgreSQL	-

Table 5. Configuration of the target system in Fig. 9.a.

ID(s)	*Type*	*Operating system*	*Zone*	*Services*	*Vulnerabilities*
1	Gateway	Ubuntu 20	-	Snort (ruleset v2.9.17.1), SSH, OpenFlow v1.3, Ryu SDN controller	-
2	Gateway	Ubuntu 20	DMZ	Snort (ruleset v2.9.17.1), SSH, OVS v2.16, OpenFlow v1.3	-
28	Gateway	Ubuntu 20	R&D	Snort (ruleset v2.9.17.1), SSH, OVS v2.16, OpenFlow v1.3	-
3,12	Switch	Ubuntu 22	DMZ	SSH, OpenFlow v1.3 , OVS v2.16	-
21, 22	Switch	Ubuntu 22	-	SSH, OpenFlow v1.3, OVS v2.16	-
23	Switch	Ubuntu 22	Admin	SSH, OpenFlow v1.3, OVS v2.16	-
29-48	Switch	Ubuntu 22	R&D	SSH, OpenFlow v1.3, OVS v2.16	-
13-16	Honeypot	Ubuntu 20	DMZ	SSH, SNMP, PostgreSQL, NTP	-
17-20	Honeypot	Ubuntu 20	DMZ	SSH, IRC, SNMP, SSH, PostgreSQL	-
4	App node	Ubuntu 20	DMZ	HTTP, DNS, SSH	CWE-1391
5, 6	App node	Ubuntu 20	DMZ	SSH, SNMP, PostgreSQL, NTP	-
7	App node	Ubuntu 20	DMZ	HTTP, Telnet, SSH	CWE-1391
8	App node	Debian Jessie	DMZ	FTP, SSH, Apache 2, SNMP	CVE-2015-3306
9,10	App node	Ubuntu 20	DMZ	NTP, IRC, SNMP, SSH, PostgreSQL	-
11	App node	Debian Jessie	DMZ	Apache 2, SMTP, SSH	CVE-2016-10033
24	Admin system	Ubuntu 20	Admin	HTTP, DNS, SSH	CWE-1391
25	Admin system	Ubuntu 20	Admin	FTP, MongoDB, SMTP, Tomcat, Teamspeak 3, SSH	-
26	Admin system	Ubuntu 20	Admin	SSH, SNMP, Postgres, NTP	-
27	Admin system	Ubuntu 20	Admin	FTP, MongoDB, SMTP, Tomcat, Teamspeak 3, SSH	CWE-1391
49-59	Compute node	Ubuntu 20	R&D	Spark, HDFS	-
60	Compute node	Debian Wheezy	R&D	Spark, HDFS, Apache 2, SNMP, SSH	CVE-2014-6271
61	Compute node	Debian 9.2	R&D	IRC, Apache 2, SSH	CWE-89
62	Compute node	Debian Jessie	R&D	Spark, HDFS, Teamspeak 3, Tomcat, SSH	CVE-2010-0426
63	Compute node	Debian Jessie	R&D	SSH, Spark, HDFS	CVE-2015-5602
64	Compute node	Debian Jessie	R&D	Samba, NTP, SSH, Spark, HDFS	CVE-2017-7494

Table 6. Configuration of the target system in Fig. 9.b.

ID(s)	*Operating system*	*Background services*	*Vulnerabilities*
1	Debian 9.2	Apache2	CWE-89
2	Debian Jessie	FTP	CVE-2015-3306
3	Ubuntu 20	SSH, Spark	CWE-1391
4	Debian Jessie	Phpmailer	CVE-2016-10033
5	Debian Wheezy	Nginx	CVE-2014-6271
6	Debian Jessie	SSH, GRPC	CWE-1391, CVE-2010-0426
7	Debian Jessie	SSH, Spring boot	CVE-2015-5602, CWE-1391
8	Debian Jessie	PostgreSQL, Samba	CVE-2017-7494

Table 7. Configuration of the target system in Fig. 9.c. (The number of replicas is dynamically scalable. When starting a new replica, its configuration is selected randomly from the table.)

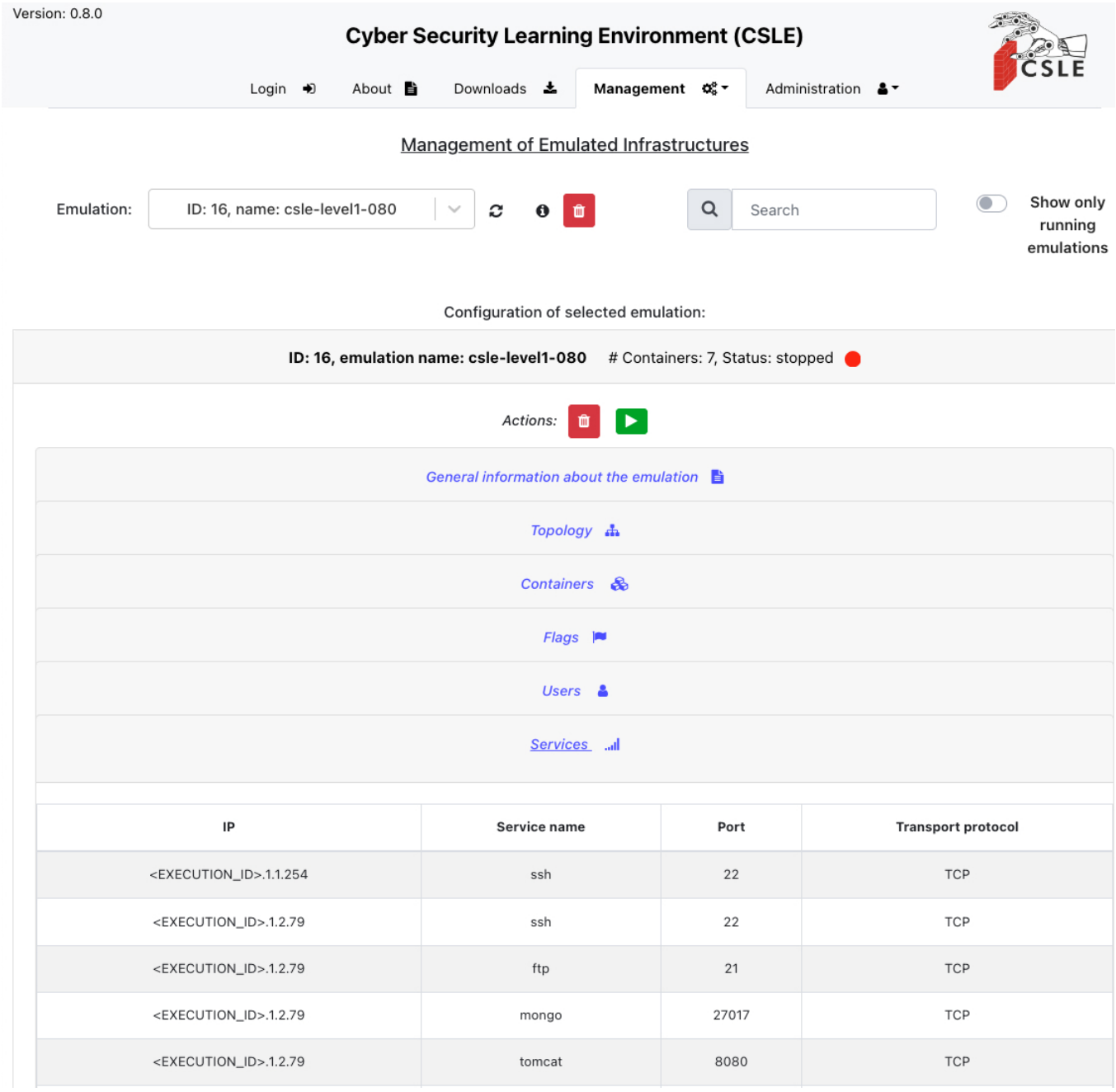

Figure 13. Screenshot of the web interface to the management system in CSLE. The figure shows the page for configuring emulations (i.e., digital twins). In addition to this page, the web interface allows viewing reinforcement learning experiments, debugging learned security strategies, real-time monitoring of digital twins, management of simulations, and integration with large language models.

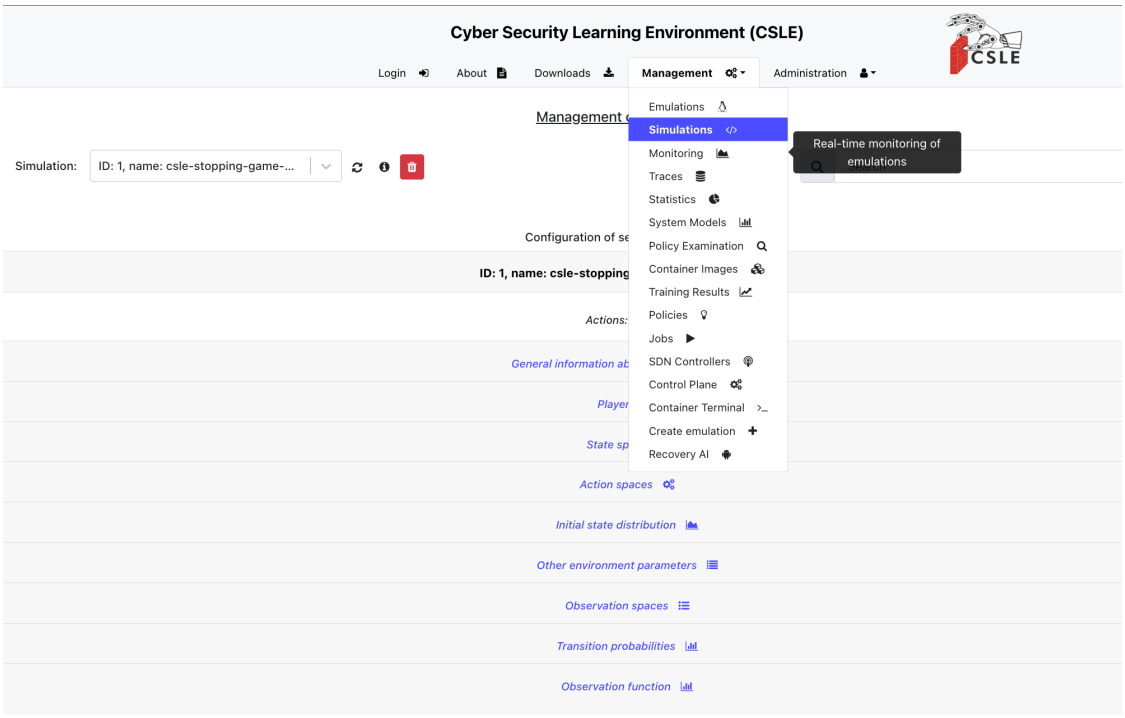

Figure 14. Screenshot of the web interface to the management system in CSLE. The figure shows the list of available pages.

Actions $\mathcal{A}$ The defender has two actions: (S)top and (C)ontinue. The action space is thus $\mathcal{A} = \{\mathsf{S}, \mathsf{C}\}$. We encode S with 1 and C with 0 to simplify the formal description below. Each stop is associated with a flow control action, and the objective is to decide the optimal times for stopping. The number of stops the defender must execute to prevent an intrusion is $L \geq 1$, which is a predefined parameter of our use case. The number of stop actions remaining is denoted by $l \in \{1, \ldots, L\}$.

States $\mathcal{S}$ The state s_t is 0 if no intrusion occurs and 1 if an intrusion is ongoing. The terminal state $\emptyset$ is reached after the defender takes the final stop action. The state space is thus $\mathcal{S} = \{0, 1, \emptyset\}$. The initial state is $s_1 = 0$. Hence, $b_1 \in \Delta(\mathcal{S})$ is the degenerate distribution $b_1(0) = 1$.

Observations $\mathcal{O}$ The defender has a partial view of the system and observes $o_t = (\Delta x_t, \Delta y_t, \Delta z_t)$, where Δx_t, Δy_t, and Δz_t are bounded counters that denote the number of severe IDS alerts, warning IDS alerts, and login attempts generated during time step t, respectively.

Transition function $f_l(s' \mid s, a)$ We model the start of an intrusion by a Bernoulli process $(Q_t)_{t=1}^{T}$, where $Q_t \sim \text{Ber}(p)$ is a Bernoulli random variable with $p > 0$. The first occurrence of $Q_t = 1$ defines the intrusion start time I, which thus is geometrically distributed.

Consequently, we can define the transition function as

$$f_1(\emptyset \mid \cdot, 1) = f_l(\emptyset \mid \emptyset, \cdot) = 1, \tag{3a}$$
$$f_l(0 \mid 0, a) = 1 - p, \qquad \text{if } l - a > 0, \tag{3b}$$
$$f_l(1 \mid 0, a) = p, \qquad \text{if } l - a > 0, \tag{3c}$$
$$f_l(1 \mid 1, a) = 1, \qquad \text{if } l - a > 0, \tag{3d}$$

where $a \in \mathcal{A}$.[2][3] All other state transitions occur with probability 0. Equation (3a) defines the transitions to the terminal state $\emptyset$, which is reached when the *final* stop action is taken (i.e., when $l = 1$ and $a = 1$). If (3a) is not applicable, i.e., if the system does not reach the terminal state, then the transitions are defined by (3b)-(3d). Equation (3b) captures the case where no intrusion occurs; (3c) specifies the case when the intrusion starts; and (3d) describes the case where an intrusion is in progress. Note that the intrusion state $s = 1$ is absorbing until L stop actions have been taken.

Observation function $z(o_t \mid s_t)$ We estimate the distributions of IDS alerts and login attempts using data from the digital twin. At the end of every 30s interval on the twin, we collect the metrics Δx, Δy, Δz, which contain the alerts and login attempts that occurred during the interval. We use $M = 21,000$ i.i.d. samples to compute the empirical distribution $\widehat{z}(\cdot \mid s_t)$ as an estimate of $z(\cdot \mid s_t)$, where $\widehat{z} \overset{\text{a.s.}}{\to} z$ as $M \to \infty$ (Glivenko & Cantelli, 1933). Figure 16

[2]Recall that we encode $(\mathsf{S}, \mathsf{C}) = (1, 0)$; hence $l_{t+1} = l_t - a_t$.
[3]We define $p = 0.01$ for the evaluation reported in the paper.

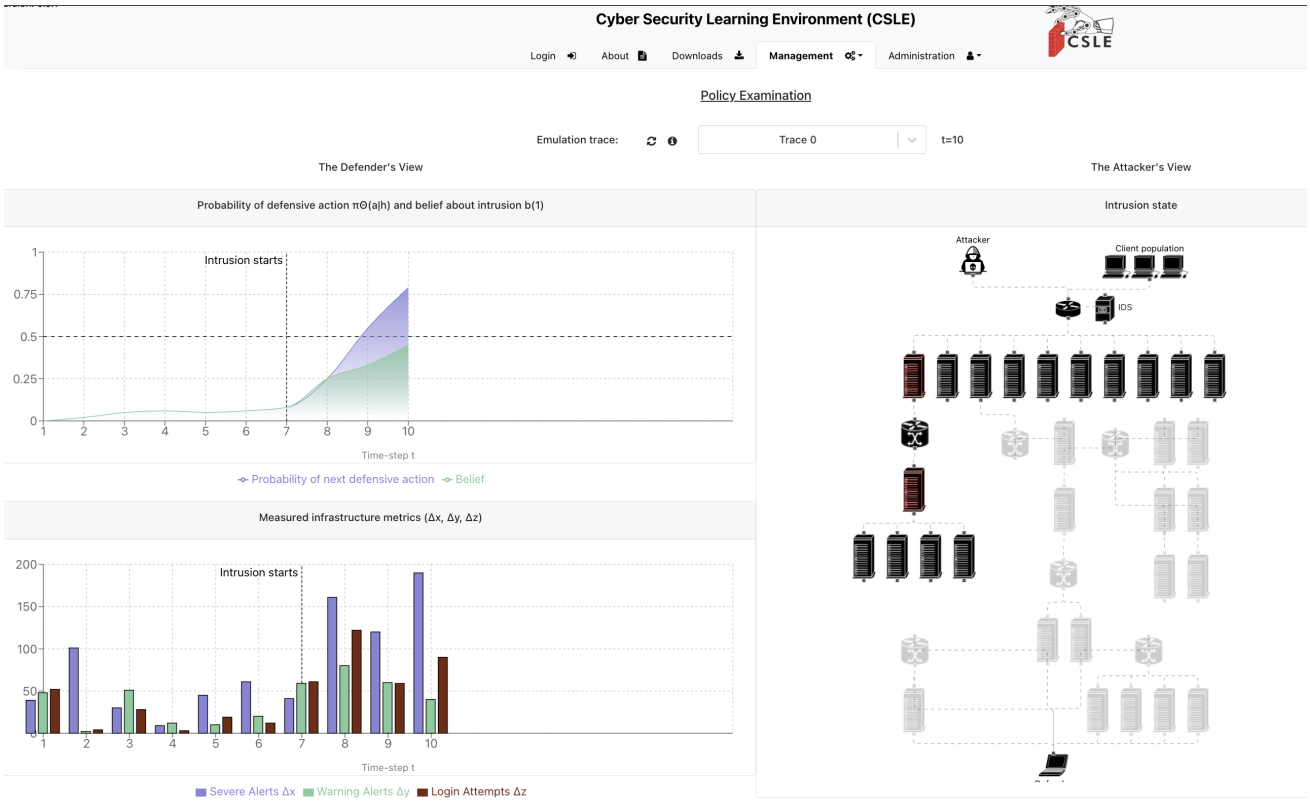

Figure 15. Screenshot of the web interface of the strategy debugger in CSLE, which allows the user to interactively step through a POMDP episode. The left panel shows the defender's view, with infrastructure statistics updated in real time. The right panel shows the attacker's view, which consists of partial knowledge of the system under attack.

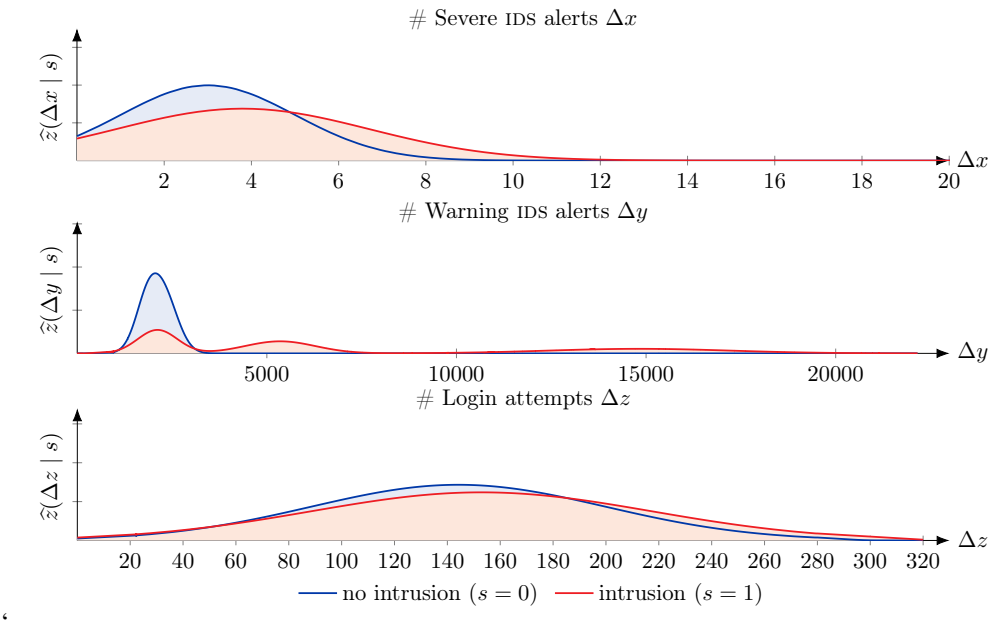

Figure 16. Estimated (smoothed) distributions of severe IDS alerts Δx (top row), warning IDS alerts Δy (middle row), and login attempts Δz (bottom row) for the flow control POMDP. The distributions are estimated based on measurements from the digital twin.

shows some of the estimated (smoothed) distributions. The distributions during normal operation and intrusion overlap. However, the distributions during intrusions tend to have more probability mass at larger values of Δx, Δy, and Δz.

Reward function $r(s, a)$ The objective is to maintain service on the infrastructure while preventing a possible intrusion. Therefore, we define the reward function to give the maximum reward if the defender maintains service until the intrusion starts and then prevents the intrusion by taking L stop actions. The reward per time step $r(s, a)$ is parameterized by the reward that the defender receives for stopping an intrusion ($R_{\text{st}} > 0$), the reward for maintaining service ($R_{\text{sla}} > 0$), and the loss of being intruded ($R_{\text{int}} < 0$):

$$r(\emptyset, \cdot) = 0, \tag{4a}$$

$$r(s, \mathsf{C}) = R_{\text{sla}} + \frac{sR_{\text{int}}}{L}, \qquad s \in \{0, 1\}, \tag{4b}$$

$$r(s, \mathsf{S}) = \frac{sR_{\text{st}}}{L}, \qquad s \in \{0, 1\}. \tag{4c}$$

Equation (4a) states that the reward in the terminal state is zero. Equation (4b) states that the defender receives a positive reward (R_{sla}) for maintaining service and a loss ($\frac{R_{\text{int}}}{L}$) for each time step that it is under intrusion. Lastly, (4c) indicates that each stop incurs a cost by interrupting service (i.e., no R_{sla}) and possibly a reward ($\frac{R_{\text{st}}}{L}$) if it affects an ongoing intrusion.

Time horizon T The time horizon T is a random variable that indicates the time $t > 1$ when the terminal state $\emptyset$ is reached.

Theoretical analysis A detailed theoretical analysis of this POMDP can be found in (Hammar & Stadler, 2022).

D.2 Flow Control Markov Game

We formulate the flow control use case when the attacker follows a dynamic strategy as a partially observed zero-sum stochastic game (Markov game). The game follows similar dynamics as the POMDP defined above and has two players: the (D)efender and the (A)ttacker. In the following, we describe the components of the game, its evolution, and the players' objectives.

Time horizon T The time horizon $T > 1$ is a random variable representing the time when the attacker stops its intrusion or is prevented, depending on which event occurs first.

State space $\mathcal{S}$ The game has three states: $s_t = 0$ if no intrusion occurs, $s_t = 1$ if an intrusion is ongoing, and $s_T = \emptyset$ if the game has ended. Hence, $\mathcal{S} = \{0, 1, \emptyset\}$. The initial state is $s_1 = 0$. Therefore, the initial state distribution $b_1 \in \Delta(\mathcal{S})$ is the degenerate distribution $b_1(0) = 1$.

Action spaces $\mathcal{A}_{\text{k}}$ Each player $\text{k} \in \mathcal{N}$ can invoke two actions: (S)top and (C)ontinue. The action spaces are thus $\mathcal{A}_{\text{D}} = \mathcal{A}_{\text{A}} = \{\mathsf{S}, \mathsf{C}\}$. Executing action S triggers a change in the game, while action C is a passive action. The attacker can invoke the stop action twice: the first to start the intrusion and the second to terminate it. The defender can invoke the stop action $L \geq 1$ times. Each invocation corresponds to a defensive action against a possible intrusion. The number of stop actions remaining to the defender at time t is known to both players and is denoted by $l_t \in \{1, \ldots, L\}$. Using the encoding $(\mathsf{S}, \mathsf{C}) = (1, 0)$, we can write $l_{t+1} = l_t - a_t^{(\text{D})}$, where $a_t^{(\text{D})}$ is the defender action at time t. At each time step, the attacker and the defender simultaneously choose their actions $a_t = (a_t^{(\text{D})}, a_t^{(\text{A})})$, where $a_t^{(\text{k})} \in \mathcal{A}_{\text{k}}$.

Observation space $\mathcal{O}$ The attacker has complete observability and knows the game state, the defender's actions, and the defender's observations. In contrast, the defender has a limited set of observations $o_t \in \mathcal{O}$, where $\mathcal{O}$ is a finite set. In our use case, o_t relates to the weighted sum of IDS alerts triggered during time step t.

Transition function $f_l(s' \mid s, a^{(\text{D})}, a^{(\text{A})})$ At each time step t, a transition from s_t to s_{t+1} occurs with probability $f_l(s_{t+1} \mid s_t, a_t^{(\text{D})}, a_t^{(\text{A})})$, where f_l is defined as

$$f_{l>1}(0 \mid 0, \mathsf{S}, \mathsf{C}) = f_l(0 \mid 0, \mathsf{C}, \mathsf{C}) = 1, \tag{5a}$$

$$f_{l>1}(1 \mid 1, \cdot, \mathsf{C}) = f_l(1 \mid 1, \mathsf{C}, \mathsf{C}) = 1 - \phi_l, \tag{5b}$$

$$f_{l>1}(1 \mid 0, \cdot, \mathsf{S}) = f_l(1 \mid 0, \mathsf{C}, \mathsf{S}) = 1, \tag{5c}$$

$$f_{l>1}(\emptyset \mid 1, \cdot, \mathsf{C}) = f_l(\emptyset \mid 1, \mathsf{C}, \mathsf{C}) = \phi_l, \tag{5d}$$

$$f_{l=1}(\emptyset \mid \cdot, \mathsf{S}, \cdot) = f_l(\emptyset \mid \emptyset, \cdot, \cdot) = f_l(\emptyset \mid 1, \cdot, \mathsf{S}) = 1. \tag{5e}$$

All other state transitions have probability 0. Equations (5a)–(5b) define the probabilities of the recurrent transitions $0 \to 0$ and $1 \to 1$. The game stays in state 0 with probability 1 if the attacker selects action C and $l_t - a_t^{(\text{D})} > 0$. Similarly, the game stays in state 1 with probability $1 - \phi_l$ if the attacker chooses action C and $l_t - a_t^{(\text{D})} > 0$. Here, ϕ_l denotes the probability that the defender stops the intrusion, which is a parameter of the use case. The intrusion can be stopped at any time step, either because the attacker terminates the intrusion or as a consequence of previous stop actions by the defender, i.e., the effect of a defensive action is non-immediate. We assume that ϕ_l increases with each stop action that the defender takes.

Equation (5c) captures the transition $0 \to 1$, which occurs when the attacker chooses action S and $l_t - a_t^{(\text{D})} > 0$. (5d)–(5e) define the probabilities of the transitions to the terminal state $\emptyset$, which is reached in three cases: (*i*) when $l_t = 1$

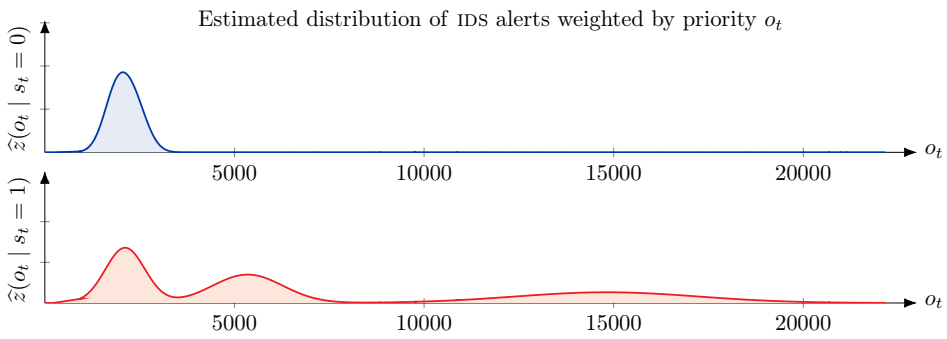

Figure 17. Fitted Gaussian mixture models of z when no intrusion occurs ($s_t = 0$) and during intrusion ($s_t = 1$) for the flow control Markov game.

and the defender takes the final stop action S (i.e., when $l_t - a_t^{(\mathrm{D})} = 0$); (*ii*) when the intrusion is stopped by the defender with probability ϕ_l; and (*iii*) when $s_t = 1$ and the attacker terminates the intrusion ($a_t^{(\mathrm{A})} = \mathsf{S} = 1$).

Reward function $r_l(s, a^{(\mathrm{D})}, a^{(\mathrm{A})})$ At time step t, the defender receives the reward $r_t = r_l(s_t, a_t^{(\mathrm{D})}, a_t^{(\mathrm{A})})$ and the attacker receives the reward $-r_t$. The reward function is parameterized by the defender's reward for stopping an intrusion ($\mathrm{R_{st}} > 0$), its cost of taking a defensive action ($\mathrm{R_{cost}} < 0$), and its cost while an intrusion occurs ($\mathrm{R_{int}} < 0$), as defined below.

$$r_l(\emptyset, \cdot) = 0, \tag{6a}$$
$$r_l(1, \cdot, \mathsf{S}) = 0, \tag{6b}$$
$$r_l(0, \mathsf{C}, \cdot) = 0, \tag{6c}$$
$$r_l(0, \mathsf{S}, \cdot) = \frac{\mathrm{R_{cost}}}{l_t}, \qquad l_t \in \{1, 2, \ldots, L\}, \tag{6d}$$
$$r_l(1, \mathsf{S}, \mathsf{C}) = \frac{\mathrm{R_{st}}}{l_t}, \qquad l_t \in \{1, 2, \ldots, L\}, \tag{6e}$$
$$r_l(1, \mathsf{C}, \mathsf{C}) = \mathrm{R_{int}}. \tag{6f}$$

Equations (6a)–(6b) state that the reward is zero in the terminal state and when the attacker terminates an intrusion. Equation (6c) states that the defender incurs no cost when no attack occurs and it does not take a defensive action. Equation (6d) indicates that the defender incurs a cost when taking a defensive action if no intrusion is ongoing. Equation (6e) states that the defender receives a reward when taking a stop action while an intrusion occurs. Lastly, (6f) indicates that the defender incurs a cost for each time step during which an intrusion occurs.

Observation function z We estimate the observation function z based on data from the digital twin. Specifically, at the end of every time step in the digital twin, i.e., at the end of each 30s interval, we collect the number of IDS alerts with priorities 1–4 that occurred during the time step, where priorities 1–4 refer to the Snort priorities "very low", "low", "medium", and "high", respectively (Roesch, 1999)[4]. We do so for $23,000$ time steps, which provides us with a dataset to estimate the distribution of IDS alerts. Using this dataset, we apply expectation-maximization (Dempster et al., 1977) to fit Gaussian mixture distributions $\widehat{z}(\cdot \mid 0)$ and $\widehat{z}(\cdot \mid 1)$ as estimates of $z(\cdot \mid 0)$ and $z(\cdot \mid 1)$, which represent the true observation distributions in the target infrastructure.

Figure 17 shows the fitted models over the discrete observation space $\mathcal{O} = \{0, 1, \ldots, 22000\}$. We note that $\widehat{z}(\cdot \mid 0)$ and $\widehat{z}(\cdot \mid 1)$ are (discretized) Gaussian mixtures with one and three components, respectively. Both mixtures have the most probability mass within 0–5000. The distribution $\widehat{z}(\cdot \mid 1)$ also has substantial probability mass at larger values.

Theoretical analysis A detailed theoretical analysis of this game can be found in (Hammar & Stadler, 2024b).

D.3 Network Segmentation Markov Game

We formulate the network segmentation use case when the attacker follows a dynamic strategy as a partially observed zero-sum stochastic game (Markov game). The game has two players: the (D)efender and the (A)ttacker.

The game is played on an IT infrastructure with application servers connected by a communication network that is segmented into zones. Overlaid on this physical infrastructure is a virtual infrastructure with a tree structure that includes *nodes*, which collectively offer services to clients. A service is modeled as a *workflow*, which comprises a set of interdependent nodes. A dependency between two nodes reflects information exchange through service invocations.

[4]Note that according to Snort's terminology (Roesch, 1999), 1 is the highest priority. We invert the labeling in our model for convenience.

In the following, we describe the components of the game, its evolution, and the players' objectives.

Infrastructure We model the virtual infrastructure as a (finite) directed graph $\mathcal{G} = \langle \{\text{gw}\} \cup \mathcal{V}, \mathcal{E} \rangle$. The graph has a tree structure rooted at the gateway gw. Each node $i \in \mathcal{V}$ has three state variables. Variable $v_{i,t}^{(\text{R})}$ represents the reconnaissance state. We have $v_{i,t}^{(\text{R})} = 1$ if the attacker has discovered the node, 0 otherwise. Variable $v_{i,t}^{(\text{I})}$ represents the intrusion state. We have $v_{i,t}^{(\text{I})} = 1$ if the attacker has compromised the node, 0 otherwise. Lastly, variable $v_{i,t}^{(\text{Z})}$ indicates the zone in which the node resides. We call a node *active* if it is functional as part of a workflow (denoted $\alpha_{i,t} = 1$). Due to a defender action (e.g., a shutdown), a node $i \in \mathcal{V}$ may become inactive ($\alpha_{i,t} = 0$). The active state is determined by $v_{i,t}^{(\text{Z})}$, i.e., $\alpha_{i,t}$ is a function of $v_{i,t}^{(\text{Z})}$.

Workflows We model a workflow $w \in \mathcal{W}$ as a subtree $\mathcal{G}_w = \langle \{\text{gw}\} \cup \mathcal{V}_w, \mathcal{E}_w \rangle$ of the infrastructure graph. Workflows do not overlap except for the gateway, which belongs to all workflows.

Attacker At each time t, the attacker takes an action $a_t^{(\text{A})}$, which is defined as the composition of the local actions on all nodes $a_t^{(\text{A})} = (a_{1,t}^{(\text{A})}, \ldots, a_{|\mathcal{V}|,t}^{(\text{A})}) \in \mathcal{A}_\text{A}$, where the set $\mathcal{A}_\text{A}$ is finite. A local action is either a null action (denoted with $\perp$) or an offensive action. An offensive action on a node i may change the reconnaissance state $v_{i,t}^{(\text{R})}$ or the intrusion state $v_{i,t}^{(\text{I})}$. A node i can only be compromised if it is discovered, i.e., if $v_{i,t}^{(\text{R})} = 1$. We express this constraint as $a_t^{(\text{A})} \in \mathcal{A}_\text{A}(s_t^{(\text{A})})$.

The attacker state $s_t^{(\text{A})} = \big(v_{i,t}^{(\text{R})}, v_{i,t}^{(\text{I})}\big)_{i \in \mathcal{V}} \in \mathcal{S}_\text{A}$ evolves as

$$s_{t+1}^{(\text{A})} \sim f_\text{A}\big(\cdot \mid s_t^{(\text{A})}, a_t^{(\text{A})}, a_t^{(\text{D})}\big), \tag{7}$$

where $a_t^{(\text{D})}$ represents the defender action at time t, as defined below.

Defender At each time t, the defender takes action $a_t^{(\text{D})}$, which is defined as the composition of the local actions on all nodes $a_t^{(\text{D})} = (a_{1,t}^{(\text{D})}, \ldots, a_{|\mathcal{V}|,t}^{(\text{D})}) \in \mathcal{A}_\text{D}$, where the set $\mathcal{A}_\text{D}$ is finite. A local action is either a defensive action or a null action $\perp$. Each defensive action $a_{i,t}^{(\text{D})} \neq \perp$ leads to $s_{i,t+1}^{(\text{A})} = (0, 0)$ and may affect $v_{i,t+1}^{(\text{Z})}$.

The defender state $s_t^{(\text{D})} = \big(v_{i,t}^{(\text{Z})}\big)_{i \in \mathcal{V}} \in \mathcal{S}_\text{D}$ evolves as

$$s_{t+1}^{(\text{D})} \sim f_\text{D}\big(\cdot \mid s_t^{(\text{D})}, a_t^{(\text{D})}\big). \tag{8}$$

Clients Clients consume infrastructure services by accessing workflows. We model client behavior through stationary stochastic processes, which affect the observations available to the attacker and the defender. That is, the clients are implicitly modeled by the observation function z, as defined next.

Observability and strategies At each time t, the defender and the attacker both observe $o_t = \big(o_{1,t}, \ldots, o_{|\mathcal{V}|,t}\big) \in \mathcal{O}$, where $\mathcal{O}$ is finite[5]. The observation o_t is drawn from the random vector $O_t = (O_{1,t}, \ldots, O_{|\mathcal{V}|,t})$ whose marginal distributions $z_{O_1}, \ldots, z_{O_{|\mathcal{V}|}}$ are stationary and conditionally independent given $s_{i,t} = (s_{i,t}^{(\text{D})}, s_{i,t}^{(\text{A})})$. (Note that z_{O_i} depends on the traffic generated by clients.) As a consequence, the joint conditional distribution $z(o \mid s)$ is given by

$$z\big(o \mid s\big) = \prod_{i=1}^{|\mathcal{V}|} z_{O_i}\big(o_i \mid s_i\big) \quad \forall o \in \mathcal{O}, s \in \mathcal{S}_\text{A} \times \mathcal{S}_\text{D}. \tag{9}$$

The sequence of observations and states at times $1, \ldots, t$ forms the histories

$$h_t^{(\text{D})} = (b_1^{(\text{D})}, s_1^{(\text{D})}, a_1^{(\text{D})}, o_2, \ldots, a_{t-1}^{(\text{D})}, s_t^{(\text{D})}, o_t) \in \mathcal{H}_\text{D},$$
$$h_t^{(\text{A})} = (b_1^{(\text{A})}, s_1^{(\text{A})}, a_1^{(\text{A})}, o_2, \ldots, a_{t-1}^{(\text{A})}, s_t^{(\text{A})}, o_t) \in \mathcal{H}_\text{A},$$

where $s^{(\text{D})} \sim b_1^{(\text{D})}$ and $s^{(\text{A})} \sim b_1^{(\text{A})}$ are the initial state distributions.

Based on their respective histories, the defender and the attacker select actions according to their strategies. The defender's behavior strategy is defined as $\pi_\text{D} \in \Pi_\text{D} = \mathcal{H}_\text{D} \to \Delta(\mathcal{A}_\text{D})$ and the attacker's behavior strategy is defined as $\pi_\text{A} \in \Pi_\text{A} = \mathcal{H}_\text{A} \to \Delta(\mathcal{A}_\text{A})$.

Defender Objective When selecting the strategy π_D, the defender must balance two conflicting objectives: maximizing the workflow utility towards its clients and minimizing the cost of intrusion. The weight $\eta \geq 0$ controls the trade-off between these two objectives, which results in the bi-objective

$$J = \sum_{t=1}^{\infty} \gamma^{t-1} \left(\sum_{w \in \mathcal{W}} \sum_{i \in \mathcal{V}_w} \underbrace{\eta u_{i,t}^{(\text{W})}}_{\text{workflow utility}} - \underbrace{c_{i,t}^{(\text{I})}}_{\text{intrusion cost}} \right), \tag{10}$$

where $\gamma \in [0, 1)$ is a discount factor, $c_{i,t}^{(\text{I})} = v_{i,t}^{(\text{I})} + c^{(\text{A})}(a_{i,t}^{(\text{D})})$ is the intrusion cost associated with node i at time t[6], and $u_{i,t}^{(\text{W})}$ expresses the workflow utility associated with node i at time t.

[5] In our use case, $o_{i,t}$ relates to the number of intrusion alerts associated with node i.

[6] $c^{(\text{A})}$ is a non-negative function that represents the operational costs of defender actions.

Markov Game When the game starts at $t = 1$, $s_1^{(\mathrm{D})}$ and $s_1^{(\mathrm{A})}$ are sampled from $b_1^{(\mathrm{D})}$ and $b_1^{(\mathrm{A})}$, respectively. A play of the game proceeds in time steps $t = 1, 2, \ldots$. At each time t, the defender observes $h_t^{(\mathrm{D})}$ and the attacker observes $h_t^{(\mathrm{A})}$. Based on these histories, both players select actions according to their respective strategies, i.e., $a_t^{(\mathrm{D})} \sim \pi_{\mathrm{D}}(\cdot \mid h_t^{(\mathrm{D})})$ and $a_t^{(\mathrm{A})} \sim \pi_{\mathrm{A}}(\cdot \mid h_t^{(\mathrm{A})})$. As a result of these actions, five events occur at time $t+1$: (*i*) o_{t+1} is sampled from z; (*ii*) $s_{t+1}^{(\mathrm{D})}$ is sampled from f_{D}; (*iii*) $s_{t+1}^{(\mathrm{A})}$ is sampled from f_{A}; (*iv*) the defender receives the reward $r(s_t, a_t^{(\mathrm{D})})$; and (*v*) the attacker receives the reward $-r(s_t, a_t^{(\mathrm{D})})$, where the reward function r is defined by the expression within brackets in (10).

Theoretical analysis A detailed theoretical analysis of this game can be found in (Hammar & Stadler, 2023).

D.4 Replication Control MDP

We formulate the replication control use case when the attacker follows a static strategy as the following MDP.

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

Actions The action $a_t = 1$ means that a new replica is added to the system; $a_t = 0$ otherwise. Hence, the action space is $a_t \in \{0, 1\} = \mathcal{A}$.

Transition probabilities The state s_t evolves as

$$s_{t+1} \sim f(\cdot \mid s_t, a_t). \tag{11}$$

We estimate the conditional probability distribution f based on measurements from the digital twin. A subset of the estimated conditional distributions is illustrated in Fig. 18.

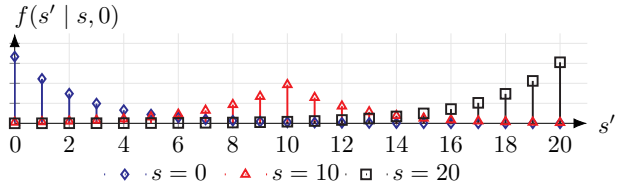

Figure 18. Transition function for the replication control MDP.

Objective Increasing the replication factor s_t improves service availability $T^{(\mathrm{A})}$ but increases cost. ($T^{(\mathrm{A})}$ is the fraction of time steps where service is available.) The goal of the controller is thus to find the optimal cost-redundancy trade-off, i.e., to minimize

$$J = \lim_{T \to \infty} \left[\sum_{t=1}^{T} \frac{a_t}{T} \right], \quad \text{subject to } T^{(\mathrm{A})} \geq \epsilon_{\mathrm{A}}, \tag{12}$$

where ϵ_{A} is the chosen lower bound on service availability. For instance, if $\epsilon_{\mathrm{A}} = 0.999$, then at most 8.4 hours of service disruption per year is allowed.

Theoretical analysis A detailed theoretical analysis of this MDP can be found in (Hammar & Stadler, 2024a).

D.5 Replication Control Markov Game

The Markov game model of the replication control use case follows the same model as the MDP, except that the attacker's strategy is dynamic. Specifically, the attacker can control which nodes in the system to attack, which causes them to be compromised with probability $p_{\mathrm{A}} = 0.01$. Hence, the attacker's strategy influences the transition function in (11). The objective of the attacker is diametrically opposed to the controller, i.e., it is a zero-sum game.

Theoretical analysis A detailed theoretical analysis of this game can be found in (Hammar & Stadler, 2025).

D.6 Recovery Control POMDP

The recovery control use case involves a networked system with K service replicas. We formulate this use case as the following POMDP.

States Each replica has two states: 1 (compromised) or 0 (safe), i.e., $s = (s^1, \ldots, s^K)$ where $s^l \in \{0, 1\}$. Compromises occur randomly over time and incur operational costs. The transition probabilities are defined as follows. If replica l is compromised ($s^l = 1$), then it remains so until recovery is applied ($a^l = 1$), at which point the state s^l is set to 0. Otherwise, the probability that it becomes compromised is $\min\{0.2(1 + \mathcal{N}_l(s)), 1\}$, where $\mathcal{N}_l(s)$ is the number of compromised neighbors of replica l in the network.

Actions An action is defined as a vector $a = (a^1, \ldots, a^K)$, where each a^l determines whether to recover component l ($a^l = 1$) or take no action ($a^l = 0$). The goal is to determine an optimal recovery strategy $\pi^\star$ that balances security requirements against recovery costs.

Observations Intrusion detection systems generate security alerts $o = (o^1, \ldots, o^K)$ that provide partial indications of the replicas' states. We define the observation distribution as

$$p(o \mid s, a) = \prod_{l=1}^{K} p(o^l \mid s^l), \quad \text{for all } o \in \mathcal{O}, s \in \mathcal{S}, a \in \mathcal{A},$$

where each $p(o^l \mid s^l)$ is estimated based on measurements from the digital twin. A subset of the estimated distributions is shown in Fig. 19.

Rewards We define the reward function as

$$r(s,a) = -\sum_{l=1}^{K} \left(\overbrace{2s^l(1-a^l)}^{\text{intrusion cost}} + \overbrace{a^l(1-s^l)}^{\text{recovery cost}} \right), \quad (13)$$

i.e., negative rewards are incurred for unmitigated intrusions ($s^l = 1$) and unnecessary recovery actions ($a^l = 1$ and $s^l = 0$).

Theoretical analysis A detailed theoretical analysis of this POMDP can be found in (Hammar et al., 2025).

E Proof of Proposition 8.1

The proof follows the same chain of reasoning as the proof of the simulation lemma in (Kearns & Singh, 2002). For notational brevity, we use $\boldsymbol{\pi}(s)$ as a shorthand for $(\pi_{\mathrm{D}}(s), \pi_{\mathrm{A}}(s))$. We start by expanding the difference $|\tilde{J}_{\boldsymbol{\pi}}(s) - J_{\boldsymbol{\pi}}(s)|$ as

$$\begin{aligned}
&|\tilde{J}_{\boldsymbol{\pi}}(s) - J_{\boldsymbol{\pi}}(s)| \\
&= \Bigg| r(s,\boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) \tilde{J}_{\boldsymbol{\pi}}(s') - \Bigg(\\
&\quad r(s,\boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} f(s' \mid s, \boldsymbol{\pi}(s)) J_{\boldsymbol{\pi}}(s') \Bigg) \Bigg| \\
&= \Bigg| \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) \tilde{J}_{\boldsymbol{\pi}}(s') - \\
&\quad \gamma \sum_{s' \in \mathcal{S}} f(s' \mid s, \boldsymbol{\pi}(s)) J_{\boldsymbol{\pi}}(s') \Bigg| \\
&= \Bigg| \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) \tilde{J}_{\boldsymbol{\pi}}(s') - \\
&\quad \gamma \sum_{s' \in \mathcal{S}} f(s' \mid s, \boldsymbol{\pi}(s)) J_{\boldsymbol{\pi}}(s') + \\
&\quad \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) J_{\boldsymbol{\pi}}(s') - \\
&\quad \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) J_{\boldsymbol{\pi}}(s') \Bigg| \\
&= \Bigg| \gamma \sum_{s' \in \mathcal{S}} \tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) \left(\tilde{J}_{\boldsymbol{\pi}}(s') - J_{\boldsymbol{\pi}}(s') \right) + \\
&\quad \gamma \sum_{s' \in \mathcal{S}} \left(\tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) - f(s' \mid s, \boldsymbol{\pi}(s)) \right) J_{\boldsymbol{\pi}}(s') \Bigg| \\
&\leq \gamma \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty + \\
&\quad \left| \gamma \sum_{s' \in \mathcal{S}} \left(\tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) - f(s' \mid s, \boldsymbol{\pi}(s)) \right) J_{\boldsymbol{\pi}}(s') \right| \\
&\overset{(a)}{\leq} \gamma \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty + \\
&\quad \gamma \sum_{s' \in \mathcal{S}} \left| \left(\tilde{f}(s' \mid s, \boldsymbol{\pi}(s)) - f(s' \mid s, \boldsymbol{\pi}(s)) \right) \right| \frac{\beta}{1-\gamma} \\
&\leq \gamma \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty + \frac{\gamma\alpha\beta}{1-\gamma},
\end{aligned}$$

where (a) follows because $|J_{\boldsymbol{\pi}}(s)| \leq \sum_{t=0}^{\infty} \gamma^t \beta = \frac{\beta}{1-\gamma}$ and the fact that $|ab| = |a||b|$ (we use the triangle inequality to move the absolute value inside the sum). Since this upper bound holds for any state s, we have

$$\begin{aligned}
&\|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty \leq \gamma \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty + \frac{\gamma\alpha\beta}{1-\gamma} \\
&\implies \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty - \gamma \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty \leq \frac{\gamma\alpha\beta}{1-\gamma} \\
&\implies (1-\gamma) \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty \leq \frac{\gamma\alpha\beta}{1-\gamma} \\
&\implies \|\tilde{J}_{\boldsymbol{\pi}} - J_{\boldsymbol{\pi}}\|_\infty \leq \frac{\gamma\alpha\beta}{(1-\gamma)^2}. \quad \square
\end{aligned}$$

F Example: Observation Mapping

Consider the flow control use case (Fig. 9.a). Assume that the attacker attempts a brute-force SSH login against a server in the target system. The mapping from raw events in the digital twin to observations that are input to the security strategy proceeds in three stages.

Stage 1 - raw log events The attacker launches an SSH brute-force attack against a server with IP 172.31.2.10. This produces raw log entries on the target host, e.g., in `/var/log/auth.log`:

```
Mar 12 09:14:01 srv2 sshd[4821]:
Failed password for root from
172.31.1.42 port 49822 ssh2
Mar 12 09:14:02 srv2 sshd[4821]:
Failed password for root from
172.31.1.42 port 49822 ssh2
...
```

Simultaneously, the network traffic is captured by the monitoring agent, which records packet-level statistics (e.g., flow byte counts and connection durations).

Stage 2 - Detection alerts The Snort IDS, running on the same network segment, matches the traffic against its ruleset and produces alerts:

```
[**] [1:2001219:20] ET SCAN Potential
SSH Brute Force [**]
[Priority: 2] 03/12-09:14:05.003
172.31.1.42:49822 -> 172.31.2.10:22
```

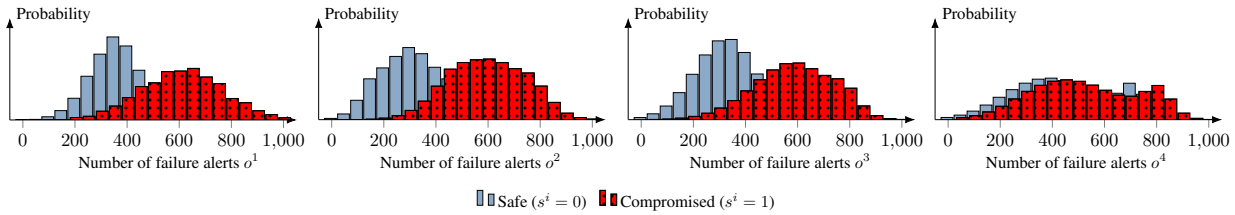

Figure 19. Empirical observation distributions for the recovery control use case based on measurements from the digital twin.

```
[Classification: Attempted Information Leak]

[**] [1:2003068:7] ET SCAN Potential
SSH Login Attempt [**]
[Priority: 2] 03/12-09:14:18.112
172.31.1.42:49830 -> 172.31.2.10:22
[Classification: Misc Attack]
```

The monitoring agent on the host pushes both the raw metrics and the IDS alerts to the Kafka event bus. A data pipeline then consumes these events, aggregates them over a fixed monitoring interval (e.g., 15 seconds), and writes the results to the metastore.

Stage 3 - Mapping to observations At the end of each monitoring interval, the aggregated data is transformed into a numerical observation that serves as input to the security strategy learned through reinforcement learning. For the flow control POMDP, the observation at time step t is:

$$o_t = (\text{number of severe alerts}, \text{number of warning alerts}, \text{number of login events}).$$

For the example above, supposing the monitoring interval covers the attack window, one realization might be:

$$o_t = (1, 9, 4).$$

This observation is then used to update the defender's belief state b_t, which is input to the security strategy π_{D}.

G Example: Digital Twin Configuration

A complete configuration of a digital twin of the system in Fig. 9.a is available at https://github.com/Kim-Hammar/csle/blob/master/emulation-system/envs/090/level_9/config.py.

H Artifact Appendix

H.1 Abstract

All results presented in this paper are fully reproducible using open-source software and data. To enable independent verification of our results and encourage future research, we release a complete set of artifacts that allow the community to build upon our work without additional engineering effort.

Specifically, we provide the following artifacts:

- The source code of CSLE, our reinforcement learning platform for autonomous security management. Available at: https://github.com/Kim-Hammar/csle.
- Pre-built Docker images for creating digital twins. Available at: https://hub.docker.com/u/kimham.
- A video that demonstrates how to install CSLE. Available at: https://www.youtube.com/watch?v=l_g3sRJwwhc.
- A video that demonstrates the web interface of CSLE. Available at: https://www.youtube.com/watch?v=iE2KPmtIs2A.
- A dataset of attack statistics generated with CSLE. Available at: https://github.com/Kim-Hammar/csle/releases/download/v0.4.0/statistics_dataset_14_nov_22_json.zip.
- Platform documentation. Available at: https://github.com/Kim-Hammar/csle/blob/master/releases (PDF version) and https://kim-hammar.github.io/csle/ (web version).

H.2 Artifact check-list (meta-information)

- **Dataset:** https://github.com/Kim-Hammar/csle/releases/download/v0.4.0/statistics_dataset_14_nov_22_json.zip.
- **Publicly available?:** Yes, https://github.com/Kim-Hammar/csle and https://hub.docker.com/u/kimham.
- **Code licenses (if publicly available)?:** CC-BY-SA 4.0.
- **Data licenses (if publicly available)?:** CC-BY-SA 4.0.
- **Archived:** Yes, https://doi.org/10.5281/zenodo.18869003.

H.3 Description

H.3.1 How delivered

The code, data, and documentation are available on GitHub: https://github.com/Kim-Hammar/csle. The video demonstrations are available on YouTube: https://www.youtube.com/watch?v=l_g3sRJwwhc and https://www.youtube.com/watch?v=iE2KPmtIs2A. The Docker images are available on DockerHub: https://hub.docker.com/u/kimham.

H.3.2 Hardware dependencies

Minimum hardware requirements to run CSLE are: 16GB memory (RAM), 1 CPU, and 50GB disk space.

H.3.3 Software dependencies

Docker, Python, PostgreSQL, Prometheus, cAdvisor, Grafana, Node, and NPM. A comprehensive list of software dependencies is available in the platform documentation: https://github.com/Kim-Hammar/csle/blob/master/releases.

H.3.4 Datasets

A dataset of attack statistics generated with CSLE is available at: https://github.com/Kim-Hammar/csle/releases/download/v0.4.0/statistics_dataset_14_nov_22_json.zip.

H.4 Installation

The installation is automated through the Ansible playbooks available at: https://github.com/Kim-Hammar/csle/tree/master/ansible.

H.5 Evaluation and expected result

To validate the availability of our artifacts, perform the following steps: (*i*) access the code, documentation, and data at https://github.com/Kim-Hammar/csle; (*ii*) access the video demonstrations at https://www.youtube.com/watch?v=l_g3sRJwwhc and https://www.youtube.com/watch?v=iE2KPmtIs2A; and (*iii*) access the Docker images at https://hub.docker.com/u/kimham.