The Road Towards Autonomous Cybersecurity Agents: Remedies for Simulation Environments [version before shortening, contains more detailed comparison between CybORG and CYST]

Martin Drašar[1][0000−0002−9623−1756], Ádám Ruman[2][0000−0002−3931−9970], Pavel Čeleda[2][0000−0002−3338−2856], and Shanchieh Jay Yang[3][0009−0004−5503−2082]

[1] Institute of Computer Science, Masaryk University, Brno, Czech Republic
drasar@ics.muni.cz
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic
{ruman, celeda}@fi.muni.cz
[3] Rochester Institute of Technology, Rochester NY, USA jay.yang@rit.edu

Abstract. One of the fundamental challenges in developing autonomous cybersecurity agents (AICA) is providing them with appropriate training environments for skills acquisition and evaluation. Current reinforcement learning (RL) algorithms rely on myriads of training runs to instill proper behavior, and this is reasonably achievable only within a simulated environment. In this paper, we explore the topic of simulation models and environments for RL and compile a set of properties and goals necessary to aid the development of AICA with real-world deployability. We also present an assessment framework to compare simulation models designed for simulating cyberattack scenarios. We examine four existing simulation tools, including a new one by the authors of the paper, and discuss their properties, particularly in terms of deployability, to support RL-based AICA. On the example of complex scenarios, we compare in-depth the two most sophisticated simulation tools and discuss their strengths. The paper thus contributes to the state-of-the-art by introducing a theoretical framework for evaluating RL-focused simulation environments, surveying the currently available environments, and introducing and evaluating a novel and complex simulation environment for next-generation autonomous cybersecurity systems.

Keywords: simulation environments, autonomous decision-making, cybersecurity

1 Introduction

The proliferation of AI-based technologies is likely to have a transformative effect on every aspect of society, cybersecurity not being an exception.

The seemingly unending thirst for workers in cybersecurity is promised to be quenched by autonomous systems driven by AI. Despite all the recent advancements, the genuine autonomy of cybersecurity solutions is an elusive goal.

The reason is multi-faceted, but one of the fundamental challenges is providing appropriate training environments where autonomous systems can be trained and evaluated. Reinforcement learning (RL) methods, which are likely to be a cornerstone of such future systems, rely on myriads of training runs to instill the proper behavior. However, such a volume of repeated executions in real or reasonably complex virtualized environments is infeasible. Simulation, when done properly, provides an instinctive and realistic option to address this problem.

RL-focused simulation environments in the cybersecurity context are currently an under-researched topic. As we demonstrate in this paper, existing works are often approached from an engineering angle, i.e., as a means to evaluate or develop specific RL solutions. The underlying simulation models appear to be ad-hoc developed as a part of the implementation, so it is challenging to do more than superficial comparisons between existing solutions. In this paper, we aim to introduce a more formal theoretical grounding for RL-focused cybersecurity simulation by answering the following research questions:

1. *How can we formalize a description of simulation models to enable comparison between various simulator implementations?*
2. *What are the properties of simulation models and their implementations that facilitate the development of deployable autonomous (especially RL-based) cybersecurity systems?*
3. *How close are we to having a simulation environment that enables the development of deployable autonomous cybersecurity systems?*

The paper is divided into seven sections, which gradually supply answers to these questions. In Section 2, we provide an overview of relevant simulation technologies. In Section 3, we derive the properties of simulation models that are the most conducive to facilitating the development of deployable autonomous cybersecurity systems. In Section 4, we introduce a framework for assessing simulation models, which is built on top of the Cyber Terrain and Capability, Opportunity, Intent (COI) models. In Section 5, we employ this assessment framework for an in-depth analysis of the four most prominent simulation environments. In Section 6, we present an in-depth qualitative comparison between CYST and CybORG [26], which are two of the most feature-rich simulators based on our assessment. We then discuss the specific properties of both simulation environments and their implications. Section 7 summarizes the answers to the research questions and discusses the promising future directions for RL-focused cybersecurity simulation.

2 Related Work

The facilitation of the development of autonomous decision-making based on reinforcement learning requires suitable training environments. These environments have to offer an adequate interface for passing observation spaces and rewards (such as OpenAI Gym [5]) and enable efficient world-building and restoration due to the desired quantity of repetitions.

Contrary to some fields, the complexity of the cybersecurity domain renders abstraction-less training infeasible; thus, simulations, emulations, or their combination are required. Simulations implement high-abstraction models and are, therefore, very lightweight but may suffer from a loss of crucial details, complicating the transfer of learned behavior into the real world. Emulations compromise higher runtime complexity for fewer abstractions.

Existing network simulators, such as OMNet++ [28] and ns-3 [22], are being extended with interfaces for AI/ML interoperability [11, 24]. While they are invaluable for researching autonomous intelligence for network-related tasks, their models' specificity limits their usability in developing more generic and multi-purpose autonomous intelligent cyber-defense agents (AICA) [12]. Security-focused environments are vital for developing AICA.

The DETER Project [15], VINE [8], and SmallWorld [9] provide an emulated environment for cybersecurity experimentation; however, they were not developed with reinforcement learning in mind. Thus, they lack the AI/ML interoperability interface. The same holds for Insight [10], an attack-oriented, pure simulation solution based on modeling system calls and I/O descriptors.

CANDLES [23] and Galaxy [25] were created to support the development and evaluation of evolutionary algorithms. The former is a simulation environment, with simple attack and defense actions, allowing multi-agent rivalry-driven training. The latter is an attack-oriented emulator used to explore enumeration method enhancement.

Environments specifically targeting reinforcement learning differ not just in abstraction level but also in their fundamental goal.

The FARLAND [17] framework aids the development of AICA utilizing network traffic. It builds upon probabilistic behaviors for action and state spaces. Being a hybrid approach, it leverages a training loop consisting of many simulated learning runs followed by an emulated evaluation round. It also explores the possibility of deceptive adversaries, who can interfere with the learning process of the developed agents.

CyGil [13] builds upon virtualized emulation and leverages the adversary emulation platform CALDERA [16] to define its action space and help collect data for the AI/ML interoperability layer to report observations and rewards to the agents. The environment provides means for single-agent or multi-agent rivalry-based training instances.

Well-known for its use in the CAGE challenge [27], CybORG [26] is a hybrid, multi-agent training environment for AICA development and evaluation. Its simulation part is based on a finite state machine that describes all the possible states of the scenario infrastructure. Agents then transition between these states with the help of predefined actions. Once the simulation loop is finished, the agents are taken into the emulation to be evaluated in a more realistic setting.

Microsoft's take on the topic is CyberBattleSim [14], a purely simulation-based environment with its focus skewed towards attacking techniques. The environment targets experimentation with the post-breach lateral movement phase

of an attack, with the ultimate goal of getting control over as many devices in the environment as possible.

The most recent environment is Yawning Titan [3]. It is purely simulation-based with very high levels of abstraction because it aims to aid the testing of the specific RL algorithms in a simplified cybersecurity context.

3 AICA development's requirements for simulations

The topic of full cybersecurity autonomy has been spearheaded by the NATO IST-152 research task group and later picked up by the AICA-IWG group. These groups attempted a holistic approach to delineate the research domain and identify key research objectives and obstacles. As a result, an informal specification of a reference architecture for autonomous cybersecurity agents was created [12], and thirteen problem domains were identified. These cover infrastructure, architecture, engineering; individual and collective decision-making; stealth and resilience; and societal aspects. Even though simulation represents only a part of one extensive problem domain, it is related to most of its components. Therefore, considering its role, we specify several high-level goals for simulation environments and their models best to facilitate the development of reliably deployable autonomous cybersecurity systems. Simulations should:

- enable the creation of deployable cybersecurity systems;
- enable multiple deployment domains and abstraction levels;
- enable human-in-the-loop;
- enable multi-agent and self-learning systems.

As these goals are rather broad, we extracted several more specific requirements for simulations that – collectively – enable reaching these goals.

General requirements:

- should support a multi-agent mode;
- should be dynamic and developing to track changes in the cybersecurity domain;
- should support different application and deployment domains;
- simulation fidelity shall track action granularity.

Requirements for actions:

- should allow integration of actions with different abstraction levels;
- should enable both agent-environment and agent-agent action types;
- should enable action parametrization;
- should enable action artifacts and side effects specification;
- may allow optional stochasticity.

Behavior control requirements:

- should not have fixed observation and reward systems;
- should provide an interface for RL frameworks.

Requirements for beyond-simulation interfacing:

- should enable the mapping of simulated actions to real-world equivalents;

– should enable the transformation of simulation artifacts to real-world effects;
– should enable integration of simulation and emulation;
– should provide a comprehensive mechanism to handle simulation timing.

These requirements enable assessing how much a simulation model and its implementation can facilitate the development of deployable autonomous cybersecurity systems and in which aspects they have deficiencies. Together with the ontology and actor framework defined later in this work, it enables the evaluation of both the current state and the potential of a simulation model and its implementation.

4 An Assessment Framework for Autonomous Cyber Agent Simulation

This work explores and assesses simulation environments that support attack scenario manifestation with RL-based cyber agents. A simulation is an implementation of an abstract model of a domain in reality (see Figure 1).

To help with the analysis, we define an ontology describing simulation models. This description stems from the Cyber Terrain model [21]. For actors in the simulation, we use and extend the Capability, Opportunity, Intent model, described in [19]. To assess the overall quality of the models, we build upon [4] that presents a qualitative, class-based framework for evaluating specific attributes of abstract representations.

Fig. 1. Levels of abstractions for simulations.

4.1 Ontology for Cyber Terrain Simulation

The high-level architecture of the ontology has five components and a varying quantity of sub-components, as shown in Figure 2.

The Topology Plane This plane describes the physical topology of the infrastructure. It describes the devices in the cyber domain with optional differentiating attributes and the communication channels that connect them. Most often, this plane is realized with the help of a discrete graph structure. The plane can be instantiated as dynamic (new hosts can be added together with connections) or static.

The Logic Plane This plane defines the functionality of the simulation. It describes attributes and nuances critical for the domain and the simulation context. This plane can be further divided into the following sub-planes:

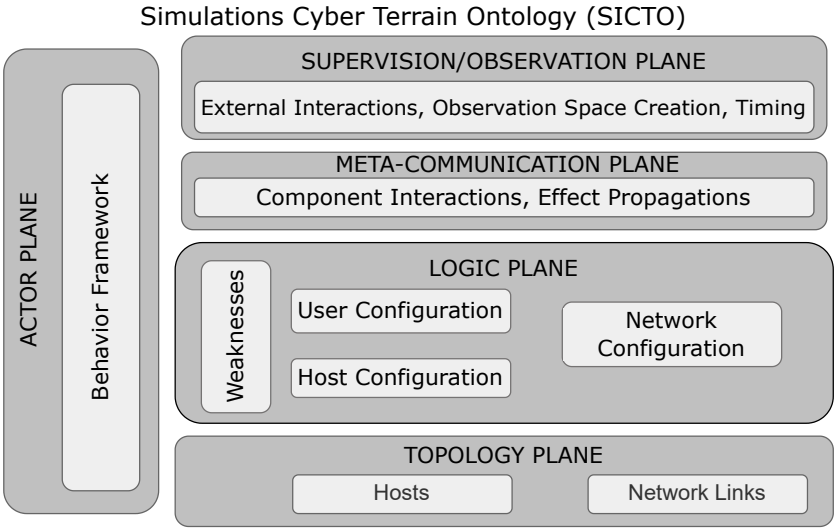

Fig. 2. The SICTO framework for describing simulation environments.

- *Host Configurations* – details about operating systems and their components, services, processes, executables, and data.
- *User Configurations* – details about user accounts, authentication methods, memory, and userspace isolation (including access to data).
- *Network Configurations* – addressing, communication rules, traffic shaping capabilities.
- *Weaknesses* – particularly crucial for actors, these describe deviations from the ideal, secure operation flow.

The Meta-Communication Plane While the name might suggest a connection with network communication, this plane describes the underlying (and possibly hidden from RL observations) signaling between the components defined in other planes. This signaling is exceptionally crucial when we want to have lightweight mechanisms for propagating the effects of events in the domain, and the propagation mechanism itself is irrelevant for AICA.

The Supervision/Observation Plane This plane is critical for creating observation spaces for RL. While the quality and detail of this plane can not elevate the overall quality of the simulation (above the threshold defined by the other components), an inadequate design can be detrimental.

This plane specifies the timing of the environment to control the interactions between active components of the model. It also provides an interface for external interactions, which are useful for monitoring, human interfacing, and allowing cooperation with external services.

The Actor Plane The actors introduce state-changing events into the simulation. Actors are defined by their behavioral model, a more complex primitive

described in Section 4.2. Well-defined actors are critical, as AICA can be seen as their decision-making guide.

4.2 Actor Behavior Framework and Types

The COI framework [19] provides a four-layer description of actor behavior; they are:

- *Intent* – the existential goal of the actor.
- *Opportunities* – the domain of events that can be invoked by this actor.
- *Capabilities* – atomic predicates limiting the actor's opportunities in real-time.
- *Preferences* – prioritization inside opportunity equivalence classes based on secondary intents.

We extend this framework with one additional layer: sophistication. Sophistication helps with the cost and risk assignment of taking specific opportunities, effectively providing another layer of long-term filtering. Thus, the **COPSI** framework offers a way to express the time and financial effectiveness of the actors.

In the case of AICA development, **COPSI** attributes might not be fixed (e.g., be loosely constrained or entirely volatile). Moreover, allowing alternatives might benefit a simulator (as it enables deeper tailoring of AICA properties).

When analyzing a simulation environment, we will use **COPSI** to describe what the environment allows.

While many options exist for actor intent, we can define four actor types on a high level. The four types, by no means exhaustive, are the following:

Adversaries The adversary intends to compromise data or service availability, integrity, or confidentiality. It is the most dynamic actor regarding capabilities, as they generally start with a minimal set and need to acquire new ones to accomplish their goal(s). Their opportunities are actions they can take as steps to reach their goal. Possible secondary intents defining preferences might be stealth, cost, complexity, etc. Sophistication can be represented by vague classes or structurally (e.g., based on NIST SP 800-30 [20]).

Defenders Defending actors aim to thwart and ultimately eliminate the possibility of attackers achieving their goals without disrupting standard operations. While their capability set and opportunities are almost static, their preferences might change throughout the scenario as it evolves. Their sophistication is heavily influenced by the technologies they are built upon.

Benign Participants Users, administrators. They intend to mimic human activity and react to events happening in the simulation (such as clicking email links, agreeing to a UAC prompt, etc.) Their capabilities are mostly predetermined but can be extended due to events invoked by other actors. Their preferences and sophistication can result from security education, role in the company, etc.

The Fates Encompassing the invocation of sporadic events and the pattern and impulse-based flow of common phenomena in the domain, these entities cover the events out of the control of other actor types.

4.3 Metrics to Measure Quality

The framework provided by MITRE [4] defines qualitative metrics for representations' comprehensiveness and concreteness (C&C).

- The classes for concreteness are: *abstract*, *notional*, *representative*, and *fully realized*.
- The classes for comprehensiveness are: *fragmentary*, *partially specified*, and *fully specified*.

The metrics are defined for adversary characteristics, adversary behavior, technical environments, operational architectures, and defender actions. Table 1 shows how we map the various simulation model components to the MITRE's C&C perspectives.

Model Component	MITRE Perspective
Adversarial actor	Attack vectors Attack actions Adversary characteristics
Defending actors' opportunities	Defender actions
Topology plane Logic plane Meta-Communication plane	Technical architecture
Logic plane - Weaknesses	Technical vulnerabilities

Table 1. MITRE C&C mapping to model components.

Assessment of concreteness and comprehensiveness helps us with assessing the fitness of the environment for RL-based AICA development because:

- the more concrete and comprehensive the attacker and defender models are, the more tailorable and capable will the trained AICA be;
- with more concrete and comprehensive technical architecture and vulnerabilities, information used for autonomous decision-making can be more nuanced.

Simulation environments are pivotal in shaping the choice domain over which AICA does its decision-making. The shape of this domain is given by its size and structure. The domain can be flat (all options are equal) or hierarchical. A hierarchical structure can be best described with the example of an adversarial AICA, which makes sequential choices over the MITRE ATT&CK® framework's TTP scheme (first choosing a tactic, then the target, followed by a technique allowing it, and lastly, a specific procedure). A flat domain would be an immediate choice from the procedures over a fixed target. Thus the *choice domain shape* is an additional property we use for assessment. Unfortunately, this domain has to be textually described, as no unified qualitative ladder can be defined.

5 Comparison of Existing Simulations' Models

In this section, we use the proposed framework to analyze existing simulation environments structurally. The analysis helps us assess these simulators' limits and what they excel at. This, in turn, helps us spot possible neglected or underdeveloped fragments in these environments and serves as an excellent pointer for where to improve.

We analyze the **Yawning Titan** [3], **CyberBattle Sim** [14], **CybORG** [26] and **CYST** [7] simulation environments as they are the four most promising candidates for RL. CYST is an environment by the authors of this paper.

5.1 Yawning Titan

This simulation environment aims to test new RL algorithms in a simplified cybersecurity setting. Thus the model is very high level(for example the only attribute of a host is its compromitation status) to limit the size of the explorable state.

This model design is most suitable for exploring the "territorial contest" in the cybersecurity domain. The model's simplicity results in a two-level adversarial choice domain of target selection and deciding whether to use a guaranteed attack. The defensive choice domain is similar, with target selection as the top layer and then action selection from a small-size sub-domain. Thus, this simulator is not suitable for the development of full-fledged AICA.

Actors and their Events The simulator has only two types of actors, defensive and adversarial. The adversarial actor, based on COPSI (4.2), looks like:

- *Intent*: Maximizing territorial control over the network hosts.
- *Opportunities*: Attacking a host with either known techniques or zero-day exploits.
- *Capabilities*: A set of hosts accessible from already controlled ones. Availability of a zero-day exploit.
- *Preferences*: Limitedly configurable.
- *Sophistication*: A numerical skill level that affects the success of attacking a host based on its vulnerability score compared to this value. Time-to-develop of zero-day exploits that allow always-successful attacks.

The defensive actors, based on the COPSI framework, can be characterized as:

- *Intent*: Minimizing the attacker's control in the network.
- *Opportunities*: An action set consisting of: *reducing the vulnerability of a host*, *scanning the network for compromised hosts*, *regaining a compromised node*, *resetting a node to its original state*, *deploying deceptive hosts (analogy with honeypots)*, *isolating a host from the network and lifting this isolation.*
- *Capabilities*: Nonexistent.
- *Preferences*: Nonexistent.
- *Sophistication*: Probability of successfully discovering compromised hosts.

5.2 CyberBattleSim

Microsoft's model has much more granularity than Yawning Titan. It is designed to examine autonomous intelligence in more complex cybersecurity scenarios. It allows CTF-like scenarios (gathering specific data) or an availability disruption campaign.

CyberBattleSim's model is more fitting for training higher-level adversarial decision-making with a limited, hierarchical choice domain. The hierarchy consists of a layer of target selection, then a layer of high-level action selection, followed by a layer of vulnerability selection.

Actors and their Events CyberBattleSim allows for adversarial and defensive actors. The adversarial actor, represented by COPSI, is:

- *Intent*: Customizable.
- *Opportunities*: The action set consisting of *scanning for vulnerabilities*, *exploiting a local vulnerability*, *exploiting a remote vulnerability*, and *signing-on to a machine*.
- *Capabilities*: A set of accessible hosts and level of control over them. Available vulnerabilities on the target.
- *Preferences*: Can be customized by tweaking the action to RL reward relation.
- *Sophistication*: Can be customized by tweaking the action to RL reward relation.

And the defensive counterpart:

- *Intent*: Countering the attacker.
- *Opportunities*: An action set consisting of *node reimaging*, *overriding firewall rules*, *starting and stopping services*.
- *Capabilities*: Nonexistent.
- *Preferences*: Can be customized in the probabilistic model.
- *Sophistication*: Can be customized in the probabilistic model.

5.3 CybORG

CybORG aims to provide an environment for creating AICA with real-world usability. It is a hybrid simulation-emulation environment, but in this paper, we are strictly interested in the simulation part. CybORG views the simulation as a state machine with states representing the overall state of the infrastructure, thus mappable to our ontology.

CybORG allows for training more realistic and multi-purpose AICA, as it provides for a two-level hierarchical choice domain (a sizable action selection layer followed by an action parametrization layer – including target selection) for both defensive and adversarial AICA.

Actors and their Events Besides adversarial and defensive actors, CybORG allows a very simple benign type actor with an action set consisting of connecting to a host, sending data, ping scanning, and port scanning or a subset of it.

The adversarial actor, via COPSI is:

- *Intent*: Customizable.
- *Opportunities*: The action set is a freely chosen subset of a vast option set. It consists of common exploit tactics and techniques from well-known tools such as Metasploit.
- *Capabilities*: A set of accessible hosts and level of control over them. Logic plane configuration values encoding action applicability.
- *Preferences*: Customizable by tweaking reward computation.
- *Sophistication*: Customizable by tweaking reward computation.

The defensive actor:

- *Intent*: Customizable.
- *Opportunities*: A freely chosen subset of a provided set of typical defender actions and techniques sampled from tools such as Velociraptor.
- *Capabilities*: Nonexistent.
- *Preferences*: Customizable by tweaking reward computation.
- *Sophistication*: Customizable by tweaking reward computation.

5.4 CYST

CYST aims to achieve the goals presented in Section 3. CYST is built on the message-passing discrete event-processing paradigm and is freely available [6]. Like CybORG, it is a hybrid simulation-emulation environment, enabling concurrent training of multi-agent systems. It supports extensive customization using a comprehensive API, which also enables dynamic changes in the simulation environment during simulation runs. It also enables on-the-fly transformations of simulation messages to other representations (such as flow or packet traces) and using these representations to interact with systems outside simulations (such as IDS systems). This way, it sidesteps the issue of reimplementing realistic cybersecurity services within a simulation and enables mixed simulation-emulation training of agents in a more realistic environment. Unlike the previous environments, CYST does not provide state space representations and reward computations – these are intrinsic to agents, and they must construct them from received messages.

Actors and their Events *CYST* enables the creation of actions with arbitrary semantics, so the actors are not limited in their capabilities by their assigned role. The effect of these actions is expressed through the use of API, which enables modifying the entire simulation.

The adversarial actor, via COPSI is:

- *Intent*: Customizable.
- *Opportunities*: The action set is a freely chosen subset of various action sets. The current implementation supports the Action-Intent Framework [18] and CYST-specific actions. MITRE ATT&CK framework implementation is on the roadmap.

- *Capabilities*: Capabilities are expressed through the possession of various in-simulation artifacts, such as authentication or authorization tokens, and network sessions.
- *Preferences*: Intrinsic to agents' implementation. Not expressible in the environment.
- *Sophistication*: Intrinsic to agents' implementation. Not expressible in the environment.

The defensive actor:

- *Intent*: Customizable.
- *Opportunities*: The action set is a freely chosen subset of various action sets. The current implementation supports only a limited subset of the MITRE D3FENSE framework. The rest is on the roadmap.
- *Capabilities*: Capabilities are expressed through the possession of various in-simulation artifacts, such as authentication or authorization tokens, and network sessions.
- *Preferences*: Intrinsic to agents' implementation. Not expressible in the environment.
- *Sophistication*: Intrinsic to agents' implementation. Not expressible in the environment.

5.5 Structured Comparison

To summarize the commonalities and differences of the analyzed simulation models in a concise way, we compiled Table 4, which uses the *SICTO* framework. Furthermore, in Table 2 and Table 3, we compare some attributes of the models based on the MITRE C&C assessment suite.

Property	Y.Titan	C.BattleSim	CybORG	CYST
Adversary characteristics	Abstract	Abstract	Abstract	Full. real.*
Attack vectors	Abstract	Notional	Notional	Notional
Attack actions	Abstract	Abstract	Full. real.	Full. real.*
Defender actions	Notional	Notional	Full. real.	Full. real.*
Technical architecture	Abstract	Notional	Represent.	Represent.
Technical vulnerabilities	Abstract	Represent.	Represent.	Represent.

* - depends on the external implementation but allows up to the class.

Table 2. Concreteness comparison of model properties.

6 In-depth comparison of CybORG and CYST

According to Table 4, it can be seen that there are two simulators that are conducive to creating real-world deployable AICA – CybORG and CYST. Both are hybrid simulation-emulation environments, and their level of abstraction is low enough to map to real-world tools and processes. According to the SICTO

Property	Y.Titan	C.BattleSim	CybORG	CYST
Adversary characteristics	Fragmentary	Fragmentary	Fragmentary	Full. spec.*
Attack vectors	Fragmentary	Part. spec.	Part. spec.	Part. spec.
Attack actions	Fragmentary	Part. spec.	Full. spec.	Full. spec.*
Defender actions	Part. spec.	Part. spec.	Full. spec.	Full. spec.*
Technical architecture	Fragmentary	Part. spec.	Part. spec.	Part. spec.
Technical vulnerabilities	Fragmentary	Part. spec.	Part. spec.	Part. spec.

* - depends on the external implementation but allows up to the class.

Table 3. Comprehensiveness comparison of model properties.

Property	***Yawning Titan***	***CyberBattleSim***	***CybORG***	***CYST***
Overall Abstraction	Very high	High	Moderately low	Moderately low
Topology				
Dynamic changes	Allowed	Not supported	Allowed	Allowed
Representation	Unidirected graph	Non-directed graph	Non-directed graph	Unidirected graph
Logic				
Network				
Rule direction	Bidirectional	Bidirectional	Bidirectional	Bidirectional
Rule granularity	Per-host	Per-protocol	Per-protocol	Per-protocol
Additional capabilities	None	None	None	Jitter, traffic shaping
Hosts				
OS	Not supported	Not supported	Available	Via software components
Software	Not supported	List of network services	Processes and executables	Active and passive services
Software properties	Not supported	State, downtime penalty	Owners, identities	Message handling function
Data	Not supported	Not supported	Files	Service data
Extendability	No	Yes, with boolean expressions	No	With custom message handling
Users				
Account granularity	Not supported	Privilege levels, per-device	Accounts, groups, per-device	Accounts, per-service
Credentials	Not supported	Primary, per-service	Primary, per-account	Primary, MFA
Authorizations	Not supported	Not supported	Not supported	Yes, with federated auth.
Remote access control	Not supported	Per-service	Per-device	Per-service
Local access control	Not supported	Not supported	Files only	Not yet available
Weaknesses				
Realism	Low	Medium	High	High
Representation	Host vulnerability score	Applicable adversary actions	Applicable adversary actions	Enablers *
Applicability guard	Attacker skill value	Host attribute prerequisites	Host attribute prerequisites	Service attribute prerequisites
Additional action attributes	None	Cost, success probability	None	None
Meta-communication				
Event invocation	Supervision intervention	Supervision intervention	Supervision intervention	Via messaging
Event propagation	Supervision intervention	Supervision intervention	Supervision intervention	Via messaging
Supervision & Observation				
Observation space	Provided	Provided	Provided	Not provided
Timing	Sequential, turn-taking	Sequential	Sequential	Concurrent
Reward computation	Provided	Provided	Provided	Not provided
Multiagent support	No	No	Yes	Yes

* – CWE [1], CVE [2], misconfigurations, bad-practices

Table 4. Comparison of simulation models' attributes structured by the *SICTO* framework.

framework, their difference is minimal, with CYST being more customizable and supporting some sophisticated features at the expense of not providing state representations and reward computations.

To provide a more in-depth comparison between those two simulators, we analyze them in terms of their intended users, i.e., what it entails to develop agents in their contexts. For CybORG, we use as a reference point the second CAGE challenge [27]. For CYST, there is not yet a published testing setup, thus, we introduce it in the following text. We then compare the required tasks for both environments to draw observations of their capabilities and strong points. Both CybORG and CYST provide open access to their code, so the interested reader is free to build on this text and follow with environment investigation in their specific niche.

Environment and scenario setup
The *CybORG* environment is described in-depth here [27], so we provide only a quick summary. The goal of the user is to develop a defense agent that can withstand attackers with different strategies. The topology of the scenario consists of three subnetworks, and the attacker always starts from one computer in one of the subnetworks. The vulnerability of hosts is specified on a per-port basis.

The *CYST* environment is tailored toward the training of attack agents. The infrastructure and kill chain are depicted in Figures 4 and 3, in the appendix. The scenario contains two routers and is partitioned into four isolated subnetworks with additional routing rules enabling, for example, limited communication between operational servers and DMZ. These networks are populated with a bare minimum number of machines, which are all critical for executing the complete kill chain. The kill chain consists of seven alternating reconnaissance and exploitation phases. While the reconnaissance is not strictly necessary, it is included as it better mimics what an attacker would have to do.

The same infrastructure is used also for emulation via Docker, and agents can transition between those two environments transparently. However, as CybORG does not support emulation for the second CAGE challenge, we are omitting this aspect from the comparison.

Agent's observations and actions
CybORG provides observations in response to agents' actions. Thanks to premade wrappers, they can be directly used with e.g., OpenAI Gym environment. These observations simulate that a defender has monitoring services deployed on each host in the infrastructure, thus providing observations matching the objective state. There are 4+7 actions (monitor, analyze, remove, restore, and service decoys) parametrized by hostname.

CYST provides neither the observation space nor rewards for agents' actions. The reasoning is that agents dependent on external observations and their rewards would not be able to progress with their training in the emulated environment, where this information is generally inaccessible. Agents are then forced to form their belief state from the incoming messages and develop reward systems according to their goals.

Agents are given a set of actions they can execute to progress within the infrastructure. These actions are scanning, password brute-forcing, session creation, credentials extraction, and data extraction. These actions are not stochastic to provide a more realistic experience for trained agents. They work only if the agent satisfies all action conditions, such as correctly adding an exploit or an authentication token.

Discussion
It is exciting and reassuring that both simulation environments are gravitating to similar goals and are using similar approaches, despite being developed in isolation. Table 5 summarises the strong points of both CYST and CybORG, which may be critical when deciding which solution to use. Properties that are

essentially the same between both are left out, as they can either be extrapolated from the model description, or they are discussed concerning achieving the goals mentioned above.

It can be said that, in general, many of the functionalities described in Table 5 can be implemented by either of the simulators provided there is enough incentive. However, it is likely that the implementation of the properties related to interfacing with humans or services besides simulation would require considerable changes to the CybORG simulation model and implementation. The same is likely for agent-agent interaction and non-singular actions.

CYST	CybORG
Infrastructure & Logic	
Network traffic shaping.	Service and OS knowledge base.
Modeling the traffic.	Modeling OS.
Support for complex authentication and authorization.	Host level information down to PID and files and their permissions.
Supervision, Actors & Agents	
Unbounded action and observation spaces.	Provides global and local observations.
Complex action parametrization to mimic real-world actions tailored for RL.	Integrated rewards.
Non-singular action handling.	Rich action space.
Transaction support for faster training.	
Agent-agent interaction in addition to agent-environment.	
External & Miscellaneous	
Strong focus on deployability.	Ready wrappers and interfaces for OpenAI.
Maximizing extensibility, stand-alone packages, usable as a library, and plugin support.	
Integration with outside running services.	
Human-machine interface.	

Table 5. Strong features of CYST and CybORG.

7 Conclusion

In this paper, we discussed the current landscape of simulation environments for training RL-based cybersecurity solutions. We proposed a theoretical framework to assess existing simulation models systematically and compared three recent works. Through this study, we answer the three originally posed research questions as follows:

1. *How can we formalize a description of simulation models to enable comparison between various simulator implementations?*

In Section 4, we present an assessment framework built on the Cyber Terrain ontology, an extended Capability, Opportunity, Intent model, and a quality metric by MITRE. We use this framework in Sections 5 and 6 to analyze the four advanced RL-focused simulation environments, including a newly introduced CYST simulation framework.

2. *What are the properties of simulation models and their implementations that facilitate the development of deployable autonomous (especially RL-based) cybersecurity systems?*
   In Section 3, we compile a set of properties for an appropriate simulation model. These properties are formulated from the general goals extracted from the pioneering works on autonomous intelligent cyber defense agents (AICA) by NATO IST-152 and AICA-IWG members.
3. *How close are we to having a simulation environment that enables the development of deployable autonomous cybersecurity systems?*
   In Section 6, we compare the usage between CYST and CybORG, and based on this, we compile a list of strong points for both simulation environments. We then argue that both environments gravitate towards the appropriate properties introduced in Section 3. However, both solutions still need to be considerably extended to enable the training of future deployable autonomous cybersecurity systems.

This paper contributes to the state-of-the-art in several ways. Introducing and applying the assessment framework provides a systematic approach to developing and assessing RL-focused cybersecurity simulators. Formulating goals and properties of appropriate simulation models sets a clear path toward capable RL-focused simulation models and environments that others can follow. And finally, by presenting a new simulation environment CYST, it introduces technological diversity to existing simulators.

7.1 Future Work

The properties of an appropriate simulation model described in Section 3 indicate the areas the research community should focus on. In addition, we see several areas that will require concentrated research and development efforts to support future deployable AICA better. These include:

- Development of environments for orchestration of simulation runs to enable at-scale training of AICA.
- Mechanisms for procedural generation of cyber terrains to provide variability to agents' training.
- High-fidelity implementation of industry-proven attack and defense action sets to increase the realism of simulations.

References

1. Common weakness enumeration. https://cwe.mitre.org, accessed: 2023-6-28
2. Cve-website. https://www.cve.org, accessed: 2023-6-28
3. Andrew, A., Spillard, S., Collyer, J., Dhir, N.: Developing optimal causal cyber-defence agents via cyber security simulation (2022). https://doi.org/10.48550/ARXIV.2207.12355
4. Bodeau, D., Graubart, R., Heinbockel, W.: Mapping the cyber terrain: Enabling cyber defensibility claims and hypotheses to be stated and evaluated with greater rigor and utility. Tech. rep., The MITRE Corporation., Bedford, MA, USA (2013), https://www.mitre.org/sites/default/files/publications/mapping-cyber-terrain-13-4175.pdf
5. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016). https://doi.org/10.48550/ARXIV.1606.01540
6. Drasar, M.: Cyst (2023), https://pypi.org/project/cyst-core/
7. Drasar, M.: Cyst api documentation (2023), https://muni.cz/go/cyst/
8. Eskridge, T.C., Carvalho, M.M., Stoner, E., Toggweiler, T., Granados, A.: Vine: A cyber emulation environment for mtd experimentation. In: Proceedings of the Second ACM Workshop on Moving Target Defense. p. 43–47. MTD '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2808475.2808486
9. Furfaro, A., Piccolo, A., Parise, A., Argento, L., Saccà, D.: A cloud-based platform for the emulation of complex cybersecurity scenarios. Future Generation Computer Systems **89**, 791–803 (2018). https://doi.org/10.1016/j.future.2018.07.025
10. Futoransky, A., Miranda, F., Orlicki, J., Sarraute, C.: Simulating cyber-attacks for fun and profit. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. Simutools '09, ICST, Brussels, BEL (2009). https://doi.org/10.4108/ICST.SIMUTOOLS2009.5773
11. Gawłowicz, P., Zubow, A.: Ns-3 meets openai gym: The playground for machine learning in networking research. In: Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. p. 113–120. MSWIM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3345768.3355908
12. Kott, A., Théron, P., Drašar, M., Dushku, E., LeBlanc, B., Losiewicz, P., Guarino, A., Mancini, L., Panico, A., Pihelgas, M., Rzadca, K.: Autonomous intelligent cyber-defense agent (aica) reference architecture. release 2.0 (2019)
13. Li, L., Fayad, R., Taylor, A.: Cygil: A cyber gym for training autonomous agents over emulated network systems (2021). https://doi.org/10.48550/ARXIV.2109.03331
14. Microsoft: Cyberbattlesim (2021), https://github.com/microsoft/cyberbattlesim, created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei.
15. Mirkovic, J., Benzel, T.V., Faber, T., Braden, R., Wroclawski, J.T., Schwab, S.: The deter project: Advancing the science of cyber security experimentation and test. In: 2010 IEEE International Conference on Technologies for Homeland Security (HST). pp. 1–7. IEEE, Waltham, MA, USA (2010). https://doi.org/10.1109/THS.2010.5655108
16. MITRE: Caldera: A scalable, adversary emulation platform (2022), https://caldera.mitre.org

17. Molina-Markham, A., Miniter, C., Powell, B., Ridley, A.: Network environment design for autonomous cyberdefense (2021). https://doi.org/10.48550/ARXIV.2103.07583
18. Moskal, S., Yang, S.J.: Cyberattack action-intent-framework for mapping intrusion observables. CoRR **abs/2002.07838** (2020), https://arxiv.org/abs/2002.07838
19. Moskal, S., Yang, S.J., Kuhl, M.E.: Cyber threat assessment via attack scenario simulation using an integrated adversary and network modeling approach. The Journal of Defense Modeling and Simulation **15**(1), 13–29 (2018). https://doi.org/10.1177/1548512917725408
20. NIST: Guide for conducting risk assessments. Tech. Rep. SP 800-30 Rev. 1, U.S. Department of Commerce, Washington, D.C. (2012). https://doi.org/10.6028/NIST.SP.800-30r1
21. Raymond, D., Cross, T., Conti, G., Nowatkowski, M.: Key terrain in cyberspace: Seeking the high ground. In: 2014 6th International Conference On Cyber Conflict (CyCon 2014). pp. 287–300. IEEE, Tallinn, Estonia (2014). https://doi.org/10.1109/CYCON.2014.6916409
22. Riley, G.F., Henderson, T.R.: The ns-3 Network Simulator, pp. 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_2
23. Rush, G., Tauritz, D.R., Kent, A.D.: Coevolutionary agent-based network defense lightweight event system (candles). In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. p. 859–866. GECCO Companion '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2739482.2768429
24. Schettler, M., Buse, D.S., Zubow, A., Dressler, F.: How to train your its? integrating machine learning with vehicular network simulation. In: 2020 IEEE Vehicular Networking Conference (VNC). pp. 1–4. IEEE, New York, NY, USA (2020). https://doi.org/10.1109/VNC51378.2020.9318324
25. Schoonover, K., Michalak, E., Harris, S., Gausmann, A., Reinbolt, H., Tauritz, D.R., Rawlings, C., Pope, A.S.: Galaxy: A network emulation framework for cybersecurity. In: 11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18). USENIX Association, Baltimore, MD (Aug 2018), https://www.usenix.org/conference/cset18/presentation/schoonover
26. Standen, M., Lucas, M., Bowman, D., Richer, T.J., Kim, J., Marriott, D.: Cyborg: A gym for the development of autonomous cyber agents (2021), https://doi.org/10.48550/ARXIV.2108.09118
27. The Technical Cooperation Program: TTCP CAGE Challenge 2 (2022), https://github.com/cage-challenge/cage-challenge-2
28. Varga, A.: OMNeT++, pp. 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_3

A Appendix – Scenario Infrastructure and Killchain

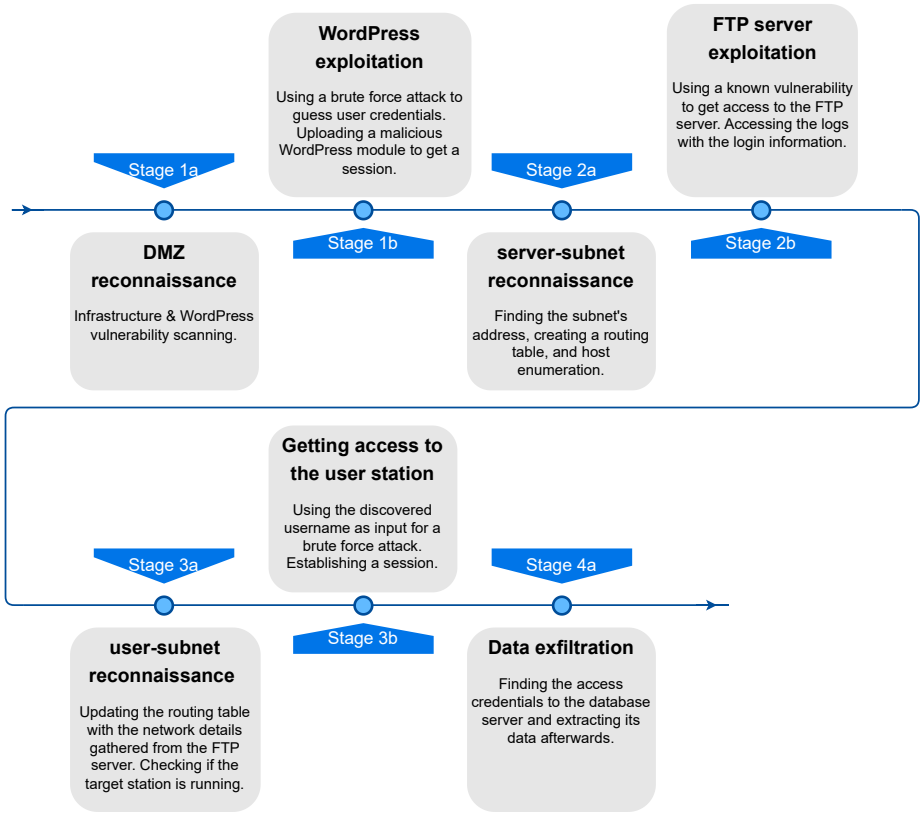

Fig. 3. The kill chain of the scenario.

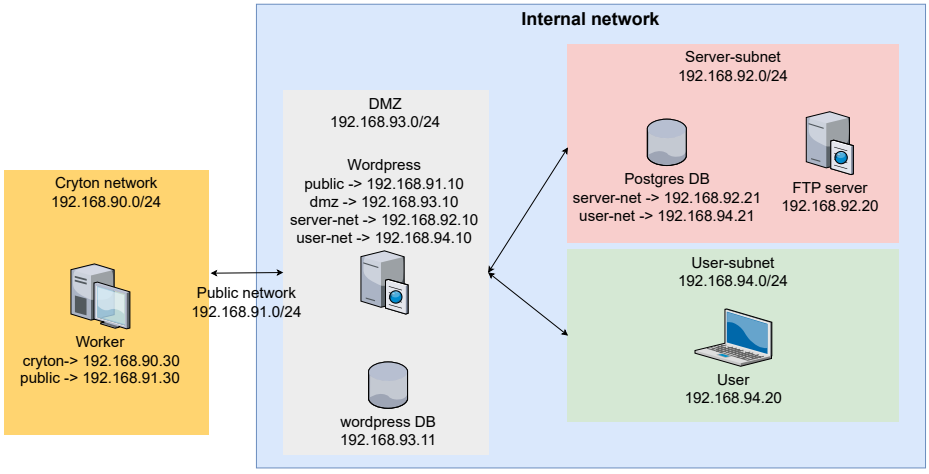

Fig. 4. The infrastructure of the scenario.