Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2025) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

37th European Modeling & Simulation Symposium (EMSS 2025), held within the 22nd International Multidisciplinary Modeling & Simulation Multiconference (I3M 2025)

A Quantitative Framework for the Validation of Twin-Based Cyber Defense

Fabrizio Baiardi[a,*], Vincenzo Sammartino[a,*]

[a]*Dipartimento di Informatica — Università di Pisa, Largo B. Pontecorvo, 3, 56127 Italy*

Abstract

The validation of cyber defense strategies is a critical challenge that becomes even more critical as we consider the lack of realistic data on intrusions enabled by new attack strategies. This paper presents a framework for continuous validation using security twins. Our approach is founded on creating a high-fidelity digital model of an ICT infrastructure, the security twin, and another model of an adversary, an attacker. We use these twins to apply a Monte Carlo method that runs a number of simulations of the intrusions of the attacker. Using the output of these simulations, we generate and validate an Intrusion Graph, a model that details how vulnerabilities can be exploited to orchestrate intrusions into the infrastructure. Each step in the simulated intrusion is validated through a system of pre- and postconditions, ensuring logical and temporal consistency. The primary advantage of this approach is its non-intrusive nature and this results in a rigorous validation and the generation of high-quality synthetic data without disrupting the operational infrastructure. This validated model serves as a powerful tool for training AI-driven defense agents, evaluating countermeasures, and predicting the impact of emerging threats in a dynamic risk landscape.

Keywords: validation; security twin; adversary emulation; intrusion graph; Monte Carlo simulations; cyber defense

1. Introduction

Validating the robustness of cyber defense mechanisms against sophisticated and evolving threats is a cornerstone of modern cybersecurity. Most traditional validation methods rely on historical data or live penetration testing, which can be disruptive and may not cover the full spectrum of potential attack vectors. A digital twin, a virtual representation of a physical system that mirrors its structure and behavior, offers a powerful alternative to the discovery, analysis, and remediation of intrusions [1–3].

This paper proposes a comprehensive validation framework built around the security twin, an enriched graph-based model specializing in the cybersecurity posture of an ICT infrastructure. Unlike traditional digital twins focused on performance and management optimization, a security twin is designed to model and simulate potential intrusions. This allows us to validate defensive strategies and generate high-fidelity synthetic data for training AI agents without

* Corresponding author.
Email addresses: fabrizio.baiardi@unipi.it (Fabrizio Baiardi), vincenzo.sammartino@phd.unipi.it (Vincenzo Sammartino)

affecting the live system. This process of continuous validation is achieved by persistently gathering information from the target infrastructure in a non-intrusive manner and verifying the twin's accuracy.

The core of our validation methodology is the generation of an Intrusion Graph. This graph is built by simulating intrusions through the interactions with the security twin of an attacker twin that models an intelligent adversary. The logical soundness of each step of an intrusion is enforced through a formal model of pre- and post-conditions. The Intrusion Graph is the culmination of thousands of Monte Carlo simulations, resulting in a probabilistically weighted automaton that details potential attack paths and defensive chokepoints.

A key challenge addressed in this framework is the data drift on intrusion [4–6]. As infrastructures evolve and new vulnerabilities emerge, historical data become obsolete. Our twin-based approach allows for continuous update of the model, ensuring that the validation process remains aligned with the current risk landscape [7, 4, 8–10]. The validated models resulting from these studies provide a baseline for training AI agents to recognize and counter threats effectively.

The remainder of this paper is structured as follows. Section 2 reviews related work on digital twins for cybersecurity and graph-based attack simulation. Section 3 details the formal models of the Security and Attacker Twins. Section 4 defines the structure of the Intrusion Graph, the core output of our methodology. Section 5 describes the Monte Carlo-based process for building this graph. Section 6 outlines the core validation methodologies for both twin and simulated intrusions. Section 7 discusses how the validated model supports advanced cyber defense. Finally, Section 8 concludes the paper and discusses future work.

Our Contributions

The main contribution of this work is a non-intrusive framework for the validation of intrusion discovery through twin-based simulation. We build a security twin via passive monitoring and fingerprinting, ensuring zero interference with production systems [9]. We then leverage this twin alongside an attacker model to run Monte Carlo simulations, where simulated actions of an attacker in an intrusion are validated by proper pre- and post-conditions. The aggregation of these simulations produces a validated intrusion graph. This graph not only provides a rich source of synthetic data to train AI agents, but also serves as a comprehensive model for risk analysis, allowing countermeasure evaluation through standard graph metrics and AI-driven techniques. To our knowledge, this is the first work to combine non-intrusive building of digital twins, formal stepwise validation, and a Monte Carlo method to produce a validated and dynamic model of cyberattacks for defence assessment.

2. Related Work

Digital twins have been widely explored in industrial contexts, mainly to support real-time monitoring, optimization, and predictive maintenance [11, 3, 12]. In the cybersecurity domain, digital twins are starting to spread to improve intrusion detection and response mechanisms. For example, recent work has shown the application of twin-based digital models for real-time intrusion detection in industrial environments by simulating distinct attacks, such as impersonation and command injection attacks [13, 14, 2, 1]. This supports the definition of a platform to identify and respond to threats in near real time, improving the security and resilience of the system. Digital twins based on attack graphs have been used to build effective platforms for risk assessment by discovering potential attack paths an attacker might follow and suggesting optimal defensive measures [15–17].

Another important line of research focuses on using graph-based models to simulate cyberattacks. Nyberg and Johnson developed a cyber defense agent that uses reinforcement learning within a simulated environment [18]. The simulation involves attack graphs, which represent the system's vulnerabilities and how an attacker might exploit them. Graph-based approaches also play a crucial role in detecting cyber anomalies by analyzing the generation of graph nodes to identify unusual behavior in network alerts [19]. Some studies, including our previous work, have also explored the integration of security twins with Monte Carlo simulations to generate multiple intrusion scenarios and identify the most likely attack paths [20–24, 4]. These simulations help defenders develop a comprehensive view of potential threats. The challenge of data drift, where historical intrusion data become obsolete, is a critical issue that twin-based approaches aim to solve [4–10].

Although the aforementioned studies lay a strong foundation, our framework distinguishes itself through the synthesis of three key elements. First, we emphasize a non-intrusive construction of the security twin via passive monitoring, ensuring zero impact on production systems [9]. Second, we introduce a formal, stepwise validation for every

simulated action using a strict pre- and post-condition model, which guarantees the logical soundness of every intrusion path. Finally, our use of a Monte Carlo method is not merely for path discovery but for constructing a validated, probabilistically weighted Intrusion Graph. To our knowledge, this is the first work to combine these three aspects to produce a dynamic and rigorously validated model for defense assessment and AI training.

3. Security and Attacker Twins

A security twin, $G_{Inf} = \langle N_{Inf}, E_{Inf} \rangle$, is a graph-based model of an ICT infrastructure designed for security analysis. The node set N_{Inf} models the physical and logical components of the infrastructure, while the edge set E_{Inf} captures their connectivity. Both nodes and edges are enriched with security-relevant attributes, turning the twin into a dynamic model that maps vulnerabilities to the attacker actions they enable. The twin provides the foundation for our rigorous, stepwise simulation and validation process.

3.1. Infrastructure Nodes and Edges

A node $n = n(C_i) \in N_{Inf}$ represents a component C_i (e.g., a server, a container, a code repository, or a firewall). $n(C_i)$ is attributed with a profile that includes:

- **Vulnerabilities ($V(C_i)$):** a set of known vulnerabilities (e.g., CVEs) present in the component's software.
- **Configurations ($S(C_i)$):** key software settings, patch levels, and security policies that affect its behavior.
- **Access Control ($A(C_i)$):** policies defining which other components can interact with C_i and with what permissions.

An edge $\langle n_i, n_j \rangle \in E_{Inf}$ exists if a component C_i can communicate with a component C_j, as governed by network topology and firewall rules. The attributes of an edge include the protocols used (e.g., HTTPS, SSH) and any channel-specific security weaknesses.

3.2. Vulnerabilities and Actions: The Basis for Validation

The security twin maps each vulnerability $v \in V(C_i)$ to a set of potential attacker actions $\{a_1, \ldots, a_m\}$. The formal definition of each action a_k is the cornerstone of the simulation validation and is described by:

- **Precondition ($\boldsymbol{Pre}(a_k)$):** the precise set of privileges and environmental states (e.g., network reachability) the attacker must possess to attempt the action.
- **Postcondition ($\boldsymbol{Post}(a_k)$):** the privileges and the information the attacker gains upon a successful execution of a_k. This defines the resulting state change.
- **Success probability ($\boldsymbol{Pr}(a_k)$):** a value or distribution in $[0, 1]$ representing the likelihood of the action's success, which may depend on the skill of the attacker.
- **Required Resources ($\boldsymbol{Res}(a_k)$):** an estimate of the skill, tools, and time required to execute the action, influencing the attacker's strategic choices.

3.2.1. Example: CI/CD Pipeline Poisoning

Consider an intrusion where the attacker targets a Continuous Integration/Continuous Deployment (CI/CD) pipeline.

- **Component and Vulnerability:** the target is a build server C_i (e.g., Jenkins), with a vulnerability, v_j, that enables unauthenticated users to trigger build jobs via an exposed API endpoint.
- **Action (a_k):** "Poisoned Pipeline Execution". The attacker triggers a build for a project, injecting a malicious command into a build parameter.
- ***Pre***(a_k)**:**

  - **Network Access:** the attacker can reach the build server's API endpoint from their current position in the network.
  - **Information:** the attacker must know the project name and the specific parameter that can be manipulated.

- ***Post***(a_k)**:**
  - **Privilege Escalation:** the malicious command executes with the permissions of the build server process, often providing access to sensitive credentials (e.g., cloud API keys, SSH keys) stored as secrets within the pipeline.
  - **Lateral Movement:** using stolen credentials, the attacker gains access to other critical components, such as source code repositories or production databases.

3.3. Attacker Twin: Modeling the Adversary

While G_{Inf} models the target, the Attacker Twin, Tw_A, models the adversary's decision-making logic. It encapsulates the attacker's goals, skill level, and strategic preferences, often aligned with frameworks like the MITRE ATT&CK Matrix [25]. The core function of Tw_A is to select the next action to execute based on its current state (known information, held privileges) and its ultimate goal (e.g., data exfiltration).

A novel approach is to implement the attacker's logic using a Large Language Model (LLM). In this paradigm, the simulation state (current intrusion node, available actions, and goal) is fed to the LLM as a prompt. The LLM's response dictates the next action, allowing for the emulation of more complex and less predictable adversarial behaviors that go beyond predefined scripts. This LLM-driven twin can adapt its tactics mid-simulation, providing a more realistic adversary for training AI defense agents.

4. Intrusion Graph: Modelling Intrusions

For simplicity, we assume that each attacker does not cooperate with other attackers. The Intrusion Graph, $\mathcal{G}(Tw_A, G_{Inf})$, is the primary artifact of our methodology, representing a validated and aggregated model of the actions an attacker A (described by the twin Tw_A) may execute against an infrastructure Inf (modeled by G_{Inf}). It is not a mere collection of paths but a formal probabilistic automaton, where each state transition has been generated through adversary simulation and then validated. Formally, the graph:

$$\mathcal{G}(Tw_A, G_{Inf}) = \langle N, E \rangle$$

is a labeled, directed, and acyclic multigraph. Each node $n \in N$ describes a validated state of an intrusion (defined by the attacker's acquired privileges and information), and each edge $e \in E$ denotes a validated state transition driven by one or more attacker actions.

The graph is constructed by merging the outputs of numerous Monte Carlo simulations. Since different simulation runs can yield distinct intrusion paths due to probabilistic action outcomes, the resulting graph intrinsically models this stochasticity. This provides a comprehensive, probabilistically weighted view of the threat landscape, suitable for implementation within a graph database for efficient querying and analysis.

We define the following notation:

- P is the set of all possible privileges.
- I is the set of all possible information items about the target infrastructure.
- $\mathcal{A}$ is the set of all possible attacker actions.
- $\mathbb{T}$ is the time domain (e.g., timestamps).

4.1. Nodes: Intrusion States

A node $n \in N$ represents a distinct, validated state of an intrusion by attacker A. It is uniquely identified by a tuple:

$$n = \langle P_n, I_n \rangle \quad \text{where } P_n \subseteq P \text{ and } I_n \subseteq I$$

These sets denote, respectively, the privileges held and the information gathered by the attacker upon reaching this state. The tuple $\langle P_n, I_n \rangle$ serves as the unique key for the node, representing what the attacker *can do* and *knows* at a specific point. Temporal information about when this state was reached in distinct simulations is stored as metadata associated with the incoming edges.

4.2. Edges: Validated State Transitions

An edge $e \in E$ connects two states, n_j and n_k, and represents the set of validated actions that caused the transition from n_j to n_k across all simulations. Each edge is defined by a tuple:

$$e_{jk} = \langle n_j, n_k, \text{Occ}(e_{jk}) \rangle$$

where $\text{Occ}(e_{jk})$ is a multiset (or "bag") of occurrences. Each occurrence is a pair:

$$(a, t) \in \mathcal{A} \times \mathbb{T}$$

Here, a is the specific action by the attacker, and t is the timestamp of its execution in a given simulation. The multiset $\text{Occ}(e_{jk})$ therefore aggregates every recorded instance of a transition between n_j and n_k.

This unified structure elegantly models both successful and failed actions:

4.2.1. Successful Actions

A transition between two **distinct** nodes ($n_j \neq n_k$) represents one or more successful actions. For example, if executing action a_1 at time t_1 in one simulation moves the attacker from state n_j to n_k, then the pair (a_1, t_1) is added to the multiset $\text{Occ}(e_{jk})$.

4.2.2. Failed Actions

A **self-loop** ($n_j = n_k$) represents one or more failed actions. If the attacker attempts action a_2 from state n_j at time t_2 and fails (thus not leaving the state n_j), the pair (a_2, t_2) is added to the multiset of the self-loop edge, $\text{Occ}(e_{jj})$. The cardinality of $\text{Occ}(e_{jj})$ for a given action provides a direct measure of its failure rate from that specific state.

4.3. State Transition Validity

For an edge $e_{jk} = \langle n_j, n_k, \text{Occ}(e_{jk}) \rangle$ to be considered valid, every occurrence $(a, t) \in \text{Occ}(e_{jk})$ must satisfy the following formal conditions, where $n_j = \langle P_j, I_j \rangle$ and $n_k = \langle P_k, I_k \rangle$:

1. **Precondition Sufficiency:** The attacker's privileges in the source node, P_j, must satisfy the preconditions required to execute action a.
2. **Postcondition Consistency:** The execution of action a from state n_j must result precisely in state n_k. This implies that the change in privileges ($P_k \setminus P_j$) and information ($I_k \setminus I_j$) should exactly match the validated result of the action a.

This rigorous definition ensures that every path in the intrusion graph represents a causally sound and empirically supported sequence of adversarial actions, making the graph a reliable foundation for predictive security analysis.

5. Building the Intrusion Graph

An intrusion graph is built by applying a Monte Carlo method that runs a large number of independent twin-based simulations. Each simulation represents a single validation experiment that yields a partial intrusion graph. These partial graphs are subsequently merged to produce the final, comprehensive model. A simulation ends when the attacker's goal is reached or a predefined time bound T_M is exceeded.

Figure 1 summarizes the key phases of this validation and calibration process. The security twin and an adversary model drive the Monte Carlo simulations. Each simulation generates potential "simulated attack paths": sequences of actions and resulting states that represent one possible intrusion. Then these paths are systematically analyzed and validated. The output is a refined, calibrated model that provides feedback to improve the twin's accuracy, ensuring that the predictive model remains current.

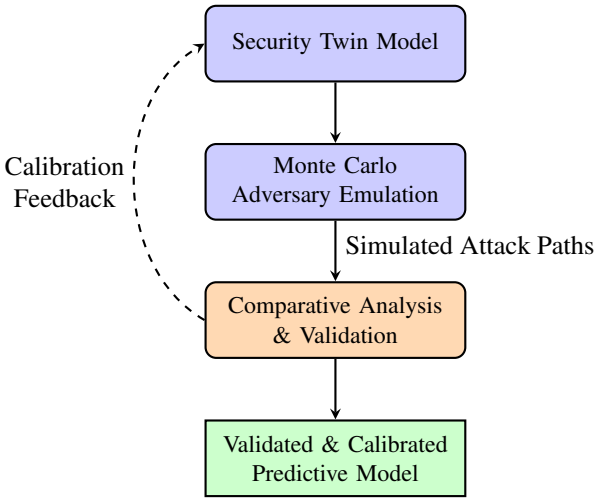

Fig. 1. Flowchart of the validation and calibration process. The Security Twin model drives Monte Carlo simulations to generate intrusion paths. Analysis of these paths provides feedback to calibrate and refine the predictive model continuously.

5.1. Initial Setup

Each simulation begins by creating a partial intrusion graph, $\mathcal{G}(Tw_A, G_{Inf})_y$. This graph is initialized with a single root node representing the initial state of the attacker A:

$$N_0 = N\langle P_0, I_0, \{0\}\rangle$$

where:

- P_0 is the initial privileges the attacker A holds.
- I_0 is the initial information A owns, such as knowledge about system configurations or known vulnerabilities.

By convention, every intrusion simulation starts at time $t = 0$.

5.2. Running One Simulation

A simulation unfolds as a sequence of steps. At each step, the attacker's strategy selects an action to execute. An action $a_k \in Tw_A$ is considered valid only if its preconditions $\mathrm{Pre}(a_k)$ are satisfied by the current intrusion state and its complexity is within the attacker's capability. This serves as the primary validation check for each step. If the action is valid, its outcome is determined probabilistically based on $\Pr(a_k)$. A successful action transitions the simulation to a new state where the postconditions $\mathrm{Post}(a_k)$ hold. If it fails, the failure is recorded by adding a self-loop, and the attacker's strategy determines the next move, such as retrying the action or choosing an alternative path. We assume an attacker never repeats a previously successful action.

5.3. Mapping a Simulation into a Graph

The sequence of actions in a single simulation is mapped into a partial intrusion graph $\mathcal{G}(Tw_A, G_{Inf})_y$.

5.3.1. Successful Execution of an Action

When an attacker successfully executes an action a_w from a state $n\langle P_j, I_j\rangle$, a new node $n\langle P_k, I_k\rangle$ is added to the graph to represent the new intrusion state. The transition is captured by an edge:

$$e_{jk} = \langle n\langle P_j, I_j\rangle,\ n\langle P_k, I_k\rangle, \{a_w\}, \{t_w\}\rangle$$

In this transition, the action a_w updates the attacker's state, resulting in the new privilege set P_k and information set I_k. The edge timing bag, $T(e_{jk})$, records the time t_w required to execute a_w.

5.3.2. Failed Action

A failed action results in a self-loop on the current node, indicating that the intrusion state has not changed:

$$e_{jj} = \langle n\langle P_j, I_j\rangle,\ n\langle P_j, I_j\rangle, \{a_w\}, \{t_{jj}\}\rangle$$

This records the failure of a_w at time t_{jj}, preserving an accurate timeline of events. The following failures in the same state update the action and time bags of this edge.

5.3.3. Information Collection

An action may only involve reconnaissance to prepare for future attacks. This is modeled as a transition where privileges remain unchanged, but the attacker's knowledge grows:

$$e_{jk} = \langle n\langle P_j, I_j\rangle,\ n\langle P_j, I_k\rangle, \{a_w\}, \{t_{jk}\}\rangle$$

Here, P_j is unchanged, but the information set expands from I_j to I_k. This can represent the discovery of new vulnerabilities in $V(C_k)$, the identification of new potential targets, the detection of misconfigured components, or the recognition of paths for a lateral movement. The time taken for this gathering action is recorded in the edge's time bag.

5.3.4. Timing Constraints

The time required to execute each action, drawn from the Security Twin, is recorded in $T(e_{jk})$, the time bag of the corresponding edge. This ensures the temporal accuracy of a simulated intrusion. The collection of times paired with the edges of the graph enables reconstruction and analysis of complete timelines of intrusions, including the identification of critical paths based on the speed of execution. This validation step ensures that the timing of each simulated action is coherent with the twin.

5.4. Monte Carlo Method

The Monte Carlo method involves the execution of a large number of independent simulations. Each simulation run, or "rollout," samples a possible trajectory through the state space, guided by the attacker's strategy and the probabilistic success of actions [26]. Even with a single attacker model, different simulations can yield distinct intrusion paths due to random events such as failures of actions and updates of the strategy. By running thousands of simulations, we can effectively explore the vast space of potential action sequences and build a statistically robust model of the threat landscape.

5.5. Graph Merging

After all simulations are complete, the resulting partial graphs $(\mathcal{G}(Tw_A, G_{Inf})_1, \ldots, \mathcal{G}(Tw_A, G_{Inf})_n)$ are merged into $\mathcal{G}(Tw_A, G_{Inf})$, a single, comprehensive intrusion graph.

This process aggregates the findings of all the simulations. Nodes that appear frequently in attack paths point out intermediate states that intrusions share. Paths with high edge multiplicity or large action-bags highlight the most common routes exploited by attackers, identifying key areas for defensive investment. Similarly, nodes with frequent failed actions (self-loops) point to well-defended but heavily targeted areas that require continuous monitoring for intrusion detection.

5.5.1. Node Merging

Some nodes in several partial graphs represent the same intrusion state because they share the identical privilege set P_j and information set I_j. These nodes are collapsed into a single one in the final graph. The time bags associated with the nodes in the various graphs are merged. For example, if a node $n\langle P_j, I_j\rangle$ in a graph $\mathcal{G}_f$ has a time bag $\{T_f, T_h\}$ and the same node in a graph $\mathcal{G}_g$ has a time bag $\{T_g\}$, the merged node will have the time bag $T_f \cup T_g = \{T_f, T_h, T_g\}$.

5.5.2. Edge Merging

Edges in several partial graphs that connect the same pair of (now merged) nodes are themselves merged. The action and time bags of the resulting edge are the union of the bags from the original edges. Initially, an edge e_{jk} in a single simulation has singleton bags: $A(e_{jk}) = \{a_w\}$ and $T(e_{jk}) = \{t_w\}$. When merging two such edges from different simulations:

$$e_{jk}^{(1)} = (n\langle P_j, I_j\rangle,\ n\langle P_k, I_k\rangle, A_{jk1}, T_{jk1}),$$

$$e_{jk}^{(2)} = (n\langle P_j, I_j\rangle,\ n\langle P_k, I_k\rangle, A_{jk2}, T_{jk2})$$

the resulting merged edge is:

$$e_{jk}^{\text{merged}} = (n\langle P_j, I_j\rangle,\ n\langle P_k, I_k\rangle,\ A_{jk1} \cup A_{jk2},\ T_{jk1} \cup T_{jk2})$$

This merging process creates a probabilistically weighted model of all validated attack paths, providing actionable intelligence to train AI agents and prioritize defensive countermeasures.

6. Core Validation Methodologies

This section details our approach to validation, which occurs at two levels: validating the security twin against the real infrastructure, and validating the simulated intrusions generated from the twin.

6.1. Twin Validation

A security twin G_{Inf} must accurately and completely represent the target infrastructure. Our primary validation method relies on the continuous monitoring of the live system to verify that no observed event contradicts the information encoded in the twin [9]. For example, if module A sends a message to module B in the target system, the corresponding connection should exist in the twin. We also implement reverse checks, such as using vulnerability scanners on a live component to check that the findings match the vulnerabilities G_{Inf} lists for that component.

Another validation technique is a multilevel detail analysis. This involves a layered validation where we "zoom in" on specific parts of the twin to verify its accuracy at progressively finer levels of detail—from high-level network topology down to individual software configurations [3, 12]. This hierarchical approach ensures the model is consistent with reality at all relevant scales.

6.2. Intrusion Validation

Assuming G_{inf} is validated, we also have to validate the sequence of actions the simulations generate. This ensures that every path in the final intrusion graph, $\mathcal{G}(Tw_A, G_{Inf})$, represents a realistic and logically sound attack path. This validation is enforced through a set of constraints that should hold for each step of each simulation:

Sequence Consistency: Each action in a sequence must satisfy three core constraints:

- **Privilege:** The attacker needs the necessary access rights to attempt the action.
- **Information:** The attacker needs the required information (for example, a specific vulnerability) to execute the action.
- **Reachability:** A path exists in G_{Inf} that enables the attacker to reach the target component.

Sequence Feasibility and Alignment:

- Every action of an attacker is enabled by a vulnerability, a misconfiguration, or a permission defined in G_{Inf}. The twin should record the corresponding information.

- The sequence of actions should not violate the network topology and the configuration of the components.

Temporal Consistency: The execution time of each action in the sequence should be consistent with the timing parameters in the security twin. This ensures that the simulated intrusion timeline is realistic.

6.3. False Positives and False Negatives

This multilayered validation process is designed to eliminate false positives that occur if an intrusion path in a simulation is not feasible in the real system. By ensuring that every simulated step adheres to strict consistency and feasibility rules, we ensure that the resulting intrusion graph is a reliable model for defense analysis.

False negatives, namely the valid attack paths that the intrusion graph does not describe, can still occur if the Monte Carlo simulations do not explore some paths due to the time constraint T_M. However, the probability of missing these paths decreases as the number of simulations increases. For any value F_n, we can run a number of simulations that result in a false negative probability lower than F_n, thus achieving complete coverage of critical attackers.

7. Leveraging the Validated Model for Cyber Defense

The validation of the security twin and its associated intrusion graph marks a critical transition from passive representation to an active cyber-defense engine. The utility of this high-fidelity model lies not only in the generation of synthetic data for AI training as it also provides the foundational intelligence for a proactive and ultimately autonomous defense paradigm. By leveraging the intrusion graph, security operations can shift from a reactive, alert-driven methodology to one that is predictive and adversarially focused. The system can continuously simulate potential intrusions according to the current status of the infrastructure. This results in the preemptive identification and mitigation of critical vulnerabilities as determined by their contextualised risk according to the unique attack surface rather than through a system-independent score such as the CVSS score.

A truly innovative application lies in the use of the model for *proactive cyber deception*. The intrusion graph can precisely identify the most probable paths an attacker might take. In addition to hardening these paths, the model can drive the automated deployment of high-interaction decoys and honeypots at key pivot points. These deceptive assets are designed to lure adversaries, feeding misinformation while capturing invaluable intelligence on their tools, techniques, and procedures (TTPs) without exposing production systems to risk. This transforms the defense posture from a static wall to a sentient, adaptive fabric that actively engages and misdirects threats.

Furthermore, the model serves as the kernel for *autonomous remediation and optimal countermeasure synthesis*. When a new zero-day vulnerability is disclosed or a novel attack strategy emerges, the model can run thousands of Monte Carlo simulations in seconds to determine its potential impact on the target system. More importantly, it can programmatically evaluate a spectrum of available countermeasures—such as micro-segmentation, credential rotation, or function deprecation—and recommend the optimal response. This recommendation is not based on a static playbook, but on a dynamic cost-benefit analysis that weighs the security gain against potential operational disruption, thus ensuring maximum resilience with minimal business friction. This approach paves the way for a self-healing infrastructure, where defenses adapt at machine speed to an ever-evolving threat landscape, moving far beyond human-in-the-loop limitations.

8. Conclusion and Future Work

This paper has presented a framework for the validation of cyber defenses that integrates security and attacker twins with Monte Carlo simulations. By focusing on a rigorous, multilayered validation process, validating the twin against the live system, and validating each simulated intrusion step via pre- and post-conditions, we produce a high-fidelity Intrusion Graph. The validation step eliminates false positives and provides a reliable foundation for analyzing infrastructure weaknesses, predicting attack paths, and training the next generation of AI-driven defense systems in a non-intrusive manner.

We believe the proposed validation framework represents a significant step in creating a more resilient and adaptive cyber defense but, despite its promise, it faces certain limitations. The computational complexity of Monte Carlo simulations requires a careful balance between model accuracy and efficiency. The fidelity of the security twin is also constrained by real-world blind spots such as shadow IT and incomplete vulnerability intelligence. Finally, our current

model simplifies the success probabilities of actions using discrete values, whereas real-world probabilities are often complex distributions.

Future work will tackle these challenges by investigating adaptive simulation methods to control complexity, incorporating more dynamic data sources to enhance twin accuracy, and crafting advanced probabilistic models to more effectively capture the intricacies of real-world cyberattacks.

References

[1] Y. Wang, Z. Su, S. Guo, M. Dai, T. H. Luan, Y. Liu, A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects, arXiv preprint arXiv:2301.13350 (2023).

[2] J. Zhou, S. Zhang, M. Gu, Revisiting digital twins: Origins, fundamentals, and practices, Frontiers of Engineering Management 9 (2022) 668–676. doi:10.1007/s42524-022-0216-2.

[3] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches, Springer, 2017, pp. 85–113.

[4] F. Baiardi, S. Ruggieri, V. Sammartino, AI-enabled Cybersecurity using Synthetic Data, in: Digital Twins Ecosystems and Applications, PerCom 2025, 2025.
URL https://www.computer.org/csdl/proceedings-article/percom-workshops/2025/355300a140/27FQOCPGpwY/

[5] A. Kuppa, N. Le-Khac, Learn to adapt: Robust drift detection in security domain, arXiv preprint arXiv:2206.07581 (2022).

[6] Cybersecurity and Infrastructure Security Agency (CISA), Compendium of cisa tech investigations 2024 (2024).

[7] F. Baiardi, V. Sammartino, S. Ruggieri, A security twin to defeat intrusions in cyber physical systems, in: ESREL SRA-E 2025, 2025.
URL https://rpsonline.com.sg/proceedings/esrel-sra-e2025/pdf/ESREL-SRA-E2025-P7829.pdf

[8] F. Baiardi, S. Ruggieri, V. Sammartino, Anticipating disasters through a security twin, in: ARES - DOD 2024 International Workshop on Dynamics of Disasters: Hybrid Threats, Vienna, Austria, 2024.

[9] F. Baiardi, V. Sammartino, S. Ruggieri, Not intrusive production of synthetic data on ict intrusions, in: SynDAiTE: Synthetic Data for AI Trustworthiness and Evolution - Workshop at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2025), 2025.

[10] F. Baiardi, V. Sammartino, S. Ruggieri, Graph-based cyber defence using a security twin, in: 29th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2025), 2025.

[11] K. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine 51 (11) (2018) 1016–1022.

[12] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital twin: Enabling technologies, challenges and open research, IEEE Access 8 (2020) 108952–108971.

[13] S. A. Varghese, A. Dehlaghi Ghadim, A. Balador, Z. Alimadadi, P. Papadimitratos, Digital twin-based intrusion detection for industrial control systems, in: 2022 IEEE Int. Conf. on Pervasive Computing and Communications Workshops, IEEE, 2022, pp. 611–617.

[14] C. Alcaraz, et al., Digital twin: A comprehensive survey of security threats, NICS Lab. Technical Reporthttps://www.nics.uma.es/wp-content/papers/Alcaraz2022b.pdf (2022).

[15] F. Baiardi, F. Tonelli, Twin based continuous patching to minimize cyber risk, European Journal for Security Research 6 (2021) 211–227.

[16] T. Zhao, E. Foo, H. Tian, A digital twin framework for cyber security in cyber-physical systems, arXiv preprint arXiv:2204.13859 (2022).

[17] Wavestone Risk Insight, Security twins: A new security & trust guarantee for connected devices (2020).

[18] J. Nyberg, P. Johnson, Learning automated defense strategies using graph-based cyber attack simulations, in: Proc. 2023 Workshop on Security Operation Center Operations and Construction, 2023. doi:10.14722/wosoc.2023.23006.

[19] A. Palladino, C. Thissen, Cyber anomaly detection using graph-node role-dynamics, ArXiv abs/1812.02848 (2018). doi:10.1145/3306195.3306198.

[20] S. Cui-xia, Network security risk analysis based on simulation attacks, Transactions of Beijing Institute of Technology (2008).

[21] M. Masi, G. Sellitto, H. Aranha, T. Pavleska, Securing critical infrastructures with a cybersecurity digital twin, Software and Systems Modeling 22 (2023) 689–707. doi:10.1007/s10270-022-01075-0.

[22] Sen, N. Bleser, A. Ulbig, Digital twin for evaluating detective countermeasures in smart grid cybersecurity, in: 2023 IEEE Inter. Confe. on Commu., Control, and Computing Tech. for Smart Grids, 2023, pp. 1–6.

[23] Y. Jiang, W. Wang, J. Ding, X. Lu, Y. Jing, Leveraging digital twin technology for enhanced cybersecurity in cyber-physical production systems, Future Internet 16 (4) (2024) 134. doi:10.3390/fi16040134.

[24] K. Ramesh, S. Gvk, M. Rao, J. L. Bapat, D. Das, Digital twin based what-if simulation of security attacks in smart irrigation systems, in: 2024 IEEE Int. Conf. on Electronics, Computing and Communication Technologies (CONECCT), 2024, pp. 1–6.

[25] B. Strom, A. Applebaum, D. Miller, K. Nickels, C. Pennington, S. Rayburn, MITRE ATT&CK™: Design and philosophy, MITRE, 2020.

[26] K. Buczynski, A. Khan, Cyber security and monte carlo simulation, https://sectara.com/news/cyber-security-and-monte-carlo-simulation/ (2020).