Evaluating Adversary Strategies Through a Security Twin

Fabrizio Baiardi
Dipartimento di Informatica
Università di Pisa, Italy
fabrizio.baiardi@unipi.it

Vincenzo Sammartino
Dipartimento di Informatica
Università di Pisa, Italy
vincenzo.sammartino@phd.unipi.it

Salvatore Ruggieri
Dipartimento di Informatica
Università di Pisa, Italy
salvatore.ruggieri@unipi.it

***Abstract*—Adversary simulation using a security twin represents a paradigm shift in proactive cybersecurity, moving beyond static vulnerability scanning to dynamic, behavioral risk assessment. Unlike standard simulations, a security twin ensures fidelity as the twin is synchronized with the live environment. This paper presents a quantitative comparison of four strategies an attacker may use to build intrusions: *Greedy Value*, *Max Probability*, *Stealth*, and *Random*. The simulations are executed on a security twin of a university network, constructed by the NOTLINE platform. We introduce a novel two-phase Monte Carlo simulation methodology to ensure both the exhaustive discovery of attack paths and the statistical convergence of performance metrics. Our experiments evaluate each strategy under three distinct defensive postures. Results confirm a quantifiable trade-off between the time of an intrusion and detection probability: while aggressive strategies are up to 50% faster, stealth-oriented approaches achieve comparable success rates in high-security environments by evading detection. We further discuss threats to validity related to network scale and modelling abstractions.**

***Index Terms*—Security Twin, Adversary Simulation, Attack Path Analysis, Monte Carlo Methods, Risk Assessment, Cybersecurity Strategies, Security Controls, Intrusion Detection Systems.**

I. INTRODUCTION

The rapid digitalization of critical infrastructure and the convergence of information technology (IT) and operational technology (OT) have largely expanded their cyberattack surface. Traditional security assessments, such as penetration testing and vulnerability scanning, provide only a snapshot of an organization's security posture at a time but they often fail to capture the complex dynamic behavior of advanced persistent threats (APTs) [1], [2]. To address this gap, we propose the adoption of proactive defense mechanisms centred around the concept of *security twin* (ST).

A ST is a live abstract model focused on security properties of a cyber-physical system that supports continuous monitoring, simulation, and analysis [3], [4]. Unlike static models, a ST evolves in real-time with the system it describes, providing a safe environment to simulate even catastrophic cyber-attacks without risking operational downtime [5]. The core utility of such a twin lies in its ability to support *breach and attack simulation* (BAS), where virtual adversaries attempt to compromise the system using known techniques, tactics, and procedures (TTPs) [6], [7].

This paper leverages the NOTLINE platform [8], [9] to build a high-fidelity ST and show how it supports a comparison of four AI-driven adversary strategies. Our contributions include the formalization of these strategies, a robust two-phase simulation methodology, and a quantitative assessment of security controls based on empirical data [10].

II. THE SECURITY TWIN PARADIGM

The adoption of a ST is a technological evolution based upon the digital twin technology [11] driven by the limitations of current cybersecurity assessment methodologies.

A. Definition and Structure

A *digital twin* is defined as a virtual representation that serves as the real-time digital counterpart of a physical object or process. In the context of cybersecurity, a ST applies this paradigm to the security attributes of an IT/OT infrastructure, the target system of the ST.

Structurally, the ST of a system is built upon a dynamic asset inventory modeled as a graph where nodes represent physical or logical assets (e.g., servers, workstations, users, data stores), and edges represent the communication protocols and logical relationships between them. Crucially, this model is not a static map; it is continuously enriched with vulnerability data (e.g., CVEs) and updated in real-time to reflect the changing state of the target system. This ensures that the ST correctly describes the actual status of the target and its components, as well as of the actions an attacker can execute because of this status. This supports accurate behavioral analysis of actual intrusions.

B. Automated Reverse Engineering of the Infrastructure

One of the primary challenges in securing modern systems is the discrepancy between the "As-Designed" architecture (what administrators think exists) and the "As-Built" reality (what actually exists). Documentation is rarely up-to-date, and shadow IT or ad-hoc configuration changes introduce unknown risks. In our framework, the high-fidelity ST is built on the NOTLINE platform [8]. By passively ingesting traffic, network telemetry, configuration files, and system logs, the platform reconstructs bottom-up the logic and topology of the target system. This process infers relationships, dependencies, and effective access control lists that may not be explicitly documented. Consequently, a ST does not assume arbitrary

network features, but rather represents the output of a reverse engineering process of the actual security attributes of the target system. This is crucial for identifying the "drift" between security policy and operational reality, i.e. between As-Designed and As-Built. The methodology we propose is target-agnostic; it can be applied to any ST that can export a graph-based representation of the infrastructure.

C. Overcoming the Limitations of Static Assessment

Traditional vulnerability assessments and penetration tests are inherently static. They provide a "snapshot" of the target system security at a single moment in time. However, modern systems are highly dynamic: configurations change, new devices connect, and new vulnerabilities are disclosed daily. A static report is often obsolete by the time it is delivered. Instead, a ST is built through the continuous data collection and reverse engineering process, such as the one of the NOTLINE platform, and it evolves in lockstep with the target system, allowing continuous rather than periodic risk assessment. This transforms security validation from a quarterly compliance exercise into a real-time operational capability [12], [13].

D. The "Safe-to-Fail" Environment

A critical barrier to effective security testing is the risk of operational disruption. Security teams cannot unleash destructive malware (e.g., ransomware, wipers) or aggressive denial-of-service attacks on a production network to test resilience, as this would cause unacceptable downtime and financial loss. A ST offers a "safe-to-fail" environment. It decouples the simulation of cyber-attacks from the target system. Using a ST, Red Teams can execute destructive scenarios with high-impact—such as wiping databases or shutting down SCADA controllers—to observe their cascading effects on the target system without any risk to live business operations.

E. Addressing Complexity and Interdependency

In modern cyber-physical systems (CPSs), the interaction between components is often non-linear. A minor vulnerability in a low-value IoT device can serve as a pivot point in an intrusion to compromise a high-value Domain Controller. These complex attack paths usually span multiple subnets and protocols. They are too intricate for human analysts to visualise, as they require sequences of actions by the attacker. The graph-based nature of a ST (as implemented in NOTLINE) naturally models these interdependencies in the target system. By reverse-engineering the transitive trust relationships between assets, the ST allows adversary simulations to discover attack paths of intrusions, or "kill chains", that would otherwise remain undetected until a breach occurs.

III. RELATED WORK

The transition from static vulnerability assessment to dynamic risk quantification is a central theme in research. Our framework intersects with three distinct but converging streams of literature: formal attack modeling, automated breach simulation, and the application of digital twin paradigms to security operations.

A. From Static Attack Graphs to Dynamic Models

The foundational approach to modeling network security is based on attack graphs (AG) and Attack Trees [14]. Early works established the formalism for mapping the sequences of exploits an attacker can use to penetrate a network. Although AGs provide a rigorous way to visualize potential attack paths, they suffer from two significant limitations: the state-space explosion problem and the lack of temporal dynamics [15], [16]. In large-scale networks, the number of possible states grows exponentially, making the generation and analysis of full AGs computationally prohibitive. Furthermore, traditional AGs are inherently static; they assume the attackers knows any intrusion and the network is a frozen snapshot.

While recent Probabilistic AGs (PAGs) incorporate CVSS-based likelihoods, they still fail to capture adversary behavior. Our work addresses this by simulating an agent to discover executable actions and exploit paths based on system vulnerabilities. By integrating distinct strategies (e.g., Greedy vs. Stealth), we transform the static AG into a dynamic tool for behavioral analysis.

B. BAS and Automated Red Teaming

The industrial need to validate security controls has led to the BAS market. These tools automate the execution of known threat vectors to test if firewalls, EDRs, and SIEMs are configured correctly. This is paralleled by research into Automated Red Teaming. Applebaum et al. [17] and others have explored the use of planning algorithms to automate the generation of intrusions [18], [19], [20].

However, the distinction between "replay" and "simulation" is critical because several existing BAS solutions simply replay recorded attack scripts. If the target system environment deviates, even slightly, from the script assumptions, the intrusion fails. In contrast, our approach utilizes autonomous agents capable of decision-making and that execute actions that depend upon the information on the security status of the target system as reflected in the ST. These adversaries adapt their actions to the current vulnerabilities and defences, offering a more realistic assessment of the target resilience.

C. Digital Twins and Security Twins

The concept of digital twin, originally defined by Fuller et al. [3] for manufacturing and aerospace, has only recently been adapted for cybersecurity by the definition of ST. A ST is distinct from a standard network description because of its accuracy and, most important, its synchronization with the physical assets in the target system. While simulations may consider an old description, an ST is synchronized with the live environment [21], [22] and offers a more updated description.

Current literature on STs often focuses on Cyber-Physical Systems (CPS) and Industrial Control Systems (ICS), where the cost of physical testing is prohibitive. [8], [23] have

extended this concept to IT infrastructures, emphasizing the automated construction of a ST through passive network monitoring. This paper builds upon that foundation by applying the NOTLINE platform not just as a static repository of asset data, but as a computational substrate for Monte Carlo experiments. In this way, a ST becomes an active predictive engine.

IV. METHODOLOGY

To ensure the reproducibility and validity of our comparative analysis, we developed an experimental framework that integrates the construction pipeline of a high-fidelity ST, a formal mathematical model for adversary actions, and a two-phase statistical simulation engine to emulate the behavior of an adversary and to handle the stochastic nature of its actions.

A. NotLine: Building the Security Twin

In our framework, the NOTLINE platform builds and continuously updates the ST through an automated, continuous, non-intrusive pipeline:

1) **Passive Network Telemetry:** The platform ingests raw network traffic (PCAP) and configuration files. It applies deep packet inspection [24] to discover active hosts, open ports, and service banners in the system without generating scanning noise that could alter the system state.
2) **Knowledge Graph Generation:** The platform maps the ingested data into a directed property graph $G = (V, E)$. Here, V represents entities (hosts, subnets, users, services), and E represents relationships (physical connections, logical trust relationships, firewall permissions).
3) **Vulnerability and Control Overlay:** The ST is enriched with threat intelligence. Identified software versions are correlated against the NVD [25] to pair each graph node with its vulnerabilities. This determines the actions an adversary has available and their success probabilities. Simultaneously, security controls (firewall ACLs, IDS signatures) are modeled as constraints on the edges among nodes.

As the NOTLINE platform continuously collects data and updates the ST, new simulations can be run to check if any known or unknown update to the target system changes the overall risk. Due to the time of adversary simulations, The status of the ST does not change during a simulation.

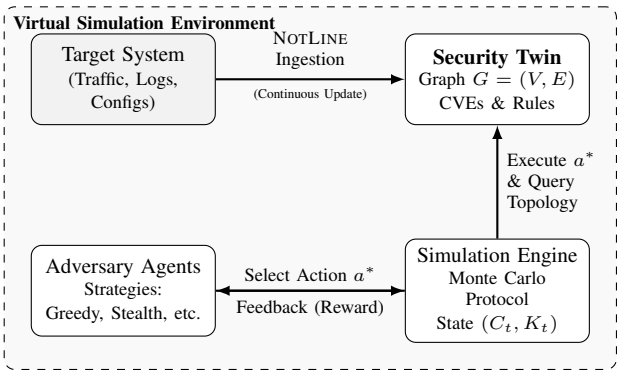

Fig. 1. The experimental architecture.

B. Formalizing the Adversary Model

The engine executes an adversary simulation as a discrete-time stochastic process on the graph G.

- **State Space (S):** The state S_t at time t is defined by the tuple (C_t, K_t), where $C_t \subseteq V$ is the set of compromised nodes (assets where the attacker has code execution rights) and $K_t \subseteq G$ is the adversary's current knowledge of the topology of the network of the target system. Initially, C_0 contains only the entry point, and K_0 is limited to public information.
- **Probabilistic Outcomes:** Each action a is paired with a tuple $\langle P_{succ}(a), P_{det}(a), Cost(a)\rangle$. P_{succ} is derived from CVSS exploitability. P_{det} represents the probability of triggering an alert. We model the IDS as a probabilistic filter where noisy actions (e.g., port scanning) have high P_{det}, while stealthy actions (e.g., lateral movement via valid credentials) have low P_{det}.
- **Action Space (A):** At each step, the engine determines $A(S_t)$, the feasible actions of the adversary as enabled by its information and access rights. Alternative actions include *Scanning* (expanding K_t), *Exploitation* (expanding C_t), or *Lateral Movement*. Then, the adversary strategy selects and executes an action in $A(S_t)$ and determines its success. Successful actions increase the amount of information and the access rights of an attacker.

An attack path describes an intrusion, i.e. all the actions an agent executes in a simulation that reaches a goal. For each path, the time it takes is also known. Success depends on the goal of the agent that is on the resources in the target system the agent aims to control.

C. Adversary Strategies

The core of our comparative analysis lies in the decision logic the simulation engine applies to select the next action $a^* \in A(S_t)$. In this paper, we considered four distinct heuristics strategy:

1) Greedy Value Strategy (High Risk, High Reward): This models an aggressive attacker motivated by immediate gain. Each node n in the target system is assigned a business value $V(n)$. Under a worst-case assumption, the agent knows this value and it calculates a Return on Investment (ROI) metric for each node it can reach. It selects the action that maximizes the ratio of potential value with respect to estimated time cost, disregarding detection risk.

$$a^* = \underset{a \in A(S_t)}{\operatorname{argmax}} \left(\frac{V(\text{target}(a))}{Cost(a)} \right) \tag{1}$$

This logic mimics ransomware operators who prioritize high-value assets (databases, backups) to maximize extortion leverage quickly.

2) Max Probability Strategy (Technological Determinism): It represents a risk-averse but technically proficient attacker. Instead of business value, this agent prioritizes the reliability

of the exploit. Using the CVSS Exploitability Sub-score as a proxy for P_{succ}, the agent chooses the path of least resistance.

$$a^* = \underset{a \in A(S_t)}{\operatorname{argmax}}(P_{succ}(a)) \tag{2}$$

The equation describes an attacker who prioritize reliable exploits and actions.

3) Stealth Strategy (The APT Approach): This models an adversary operating under a "detection budget", i.e. that maintains a cumulative noise counter η. For every action in $A(S_t)$, it evaluates the associated detection probability $P_{det}(a)$. The optimization goal is to reach a goal while keeping η below a critical threshold τ_{IDS}.

$$\text{minimize} \sum_{t=0}^{T} P_{det}(a_t) \quad \text{subject to Target} \in C_T \tag{3}$$

In practice, this agent will bypass a fast, noisy exploit (e.g., a brute-force attack) in favour of a slow, silent technique (e.g., pass-the-hash), effectively trading time for survival.

4) Random Strategy (Baseline): The random agent selects $a \in A(S_t)$ using a uniform distribution, with a slight weighting factor towards unvisited nodes to prevent infinite loops. This serves as a stochastic baseline to evaluate the other strategies.

D. Statistical Simulation Protocol

Given the stochastic nature of exploit success and IDS detection, a single simulation run is statistically not significant. We devised a two-phase execution protocol to ensure robust results from our simulation engine.

a) Phase 1: Path Discovery (Saturation): The first objective is to map the "Attack Surface" by identifying all feasible attack paths of the agent. Our engine runs simulations in batches. Let $\mathcal{P}_k$ be the set of unique successful attack paths of an agent that the engine discovers after batch k. We define a saturation condition where the simulation halts only when no simulation discovers a new path for a predefined window of iterations W:

$$|\mathcal{P}_k| - |\mathcal{P}_{k-1}| = 0 \quad \forall k \in [i, i+W] \tag{4}$$

This gives some assurance we have explored the tails of the probability distribution, finding even rare, low-probability attack vectors.

b) Phase 2: Metric Convergence (Stabilization): After identifying attack paths, the engine focuses on stabilizing quantitative metrics such as *Average Time to Compromise* (TTC). To this purpose, it computes the running mean μ and standard deviation σ of the TTC. The engine runs further simulations until the width of the 95% confidence interval for the mean is within a 5% relative error margin.

$$\frac{1.96 \cdot \sigma}{\sqrt{N}} \leq 0.05 \cdot \mu \tag{5}$$

This stopping criterion ensures that the differences between strategies are statistically significant and not artefacts of random variance. Both the confidence level and the relative error can be tuned according to the target system and the required accuracy.

V. Experimental Results

To validate our framework, we executed a campaign of Monte Carlo simulations on a ST modeling a typical university departmental network. It is important to note that while the ST supports real-time updates, for the purpose of this comparative analysis, we froze the twin's state at t_0. This ensures that any change in performance is only due to the adversary strategies and not to changing network conditions during the experiment. A firewall partitions the target system into 5 distinct subnets (DMZ, Student Wi-Fi, Admin LAN, Research Lab, Data Center) with a total of 60 active hosts and over 250 services. An IDS monitors network interactions. In the following, we neglect the details of the IDS detection strategies and only focus on their success probability. The goal of an intrusion for all adversary agents is to compromise the Domain Controller in the Data Center and exfiltrate a specific "flag" file.

NotLine and the simulation engine runs on a dedicated high-performance computing cluster to ensure an acceptable performance of the engine. For each strategy-scenario pair, i.e. for each ST and agent, we applied the two-phase protocol in Section IV executing over 12,000 simulations.

A. Scenario 1: Baseline (Firewall and IDS Active)

This scenario represents the "Gold Standard" of defense, with strict network segmentation rules enforced by firewalls and a signature-based intrusion detection system (IDS) monitoring traffic for known exploit patterns.

TABLE I
Strategy Performance: Baseline (FW + IDS)

Strategy	Success (%)	Avg. Time (h)	Avg. Steps
Greedy Value	11.4	28.65	5.33
Max Probability	13.2	45.20	7.82
Stealth	13.2	65.41	9.15
Random	1.0	35.99	6.02

Detailed Analysis: The baseline results highlight the efficacy of layered defenses in this system. The *Greedy Value* strategy, despite its aggressive logic, achieves a relatively low success rate of 11.4%. Analysis of the failed traces reveals that this agent frequently triggers high-severity IDS alerts early in the intrusion because of noisy port scans or brute-force attempts that lead to automated blocking. However, when it succeeds, it is the fastest (28.65 hours), due to the exploitation of direct, high-risk paths.

The *Stealth* strategy shows superior resilience, matching the highest success rate (13.2%) of *Max Probability*, but with a steep operational cost: an average time of 65.41 hours. The 130% increase in the time of an intrusion compared to the Greedy agent may be due to the agent's priorities, which force it to "sleep" between actions to decay the noise counter and to choose circuitous routes that bypass monitored chokepoints. The *Random* strategy's 1.0% success rate serves as a control, confirming that the network structure prevents accidental compromise.

B. Scenario 2: IDS Disabled (Firewall Only)

This scenario deactivates the IDS and the associated automated blocking rules, leaving only the static Access Control Lists (ACLs) of the firewalls. This simulates an environment with perimeter security but no internal visibility. This shows how the framework supports a what-if approach to compare alternative security mechanisms [26].

TABLE II
STRATEGY PERFORMANCE: NO IDS

Strategy	Success (%)	Avg. Time (h)	Avg. Steps
Greedy Value	23.2	12.50	3.95
Max Probability	23.4	18.99	4.62
Stealth	**23.5**	26.10	**5.03**
Random	16.1	20.23	4.69

Detailed Analysis: As expected, the freezing of the IDS strongly improves adversary performance. Success rates across all intelligent strategies nearly doubled, converging around 23%. This shows that about half of the intrusions in the Baseline scenario were thwarted by dynamic detection rather than static segmentation.

Most significantly, the *Time to Compromise* collapses. The *Greedy* agent reduces its time to just 12.50 hours. Even the *Stealth* agent, which no longer receives "high noise" feedback from the environment, can optimise its time from 65.41 to 26.10 hours. Hence, the IDS adds roughly a 40-hour delay to a sophisticated attack.

C. Scenario 3: Firewall Disabled (IDS Active)

The final scenario removes network segmentation to simulate a "flat" network with an active IDS. This is a further what-if test to evaluate whether detection can compensate for poor segmentation.

TABLE III
STRATEGY PERFORMANCE: NO FIREWALL (IDS ACTIVE)

Strategy	Success (%)	Avg. Time (h)	Avg. Steps
Greedy Value	13.5	15.02	4.03
Max Probability	**14.6**	20.48	5.49
Stealth	**14.6**	51.00	7.92
Random	6.3	18.88	4.07

Detailed Analysis: By removing the firewall, we reduce the average number of steps of attack paths. However, it does not significantly improve success rates if the IDS remains active. The success rates (approx. 14.6%) are only marginally better than the Baseline (13.2%).

Crucially, the *Stealth* strategy is still slow (51.00 hours). Even in a flat network, the agent has to move cautiously to avoid detection. This suggests that the baseline high time is mainly due to IDS evasion, rather than navigating the resulting network. The *Greedy* agent, while faster than in the Baseline, still suffers because of a high detection rate, reinforcing the observation that speed is a liability in monitored environments.

D. Statistical Convergence and Stability

A contribution of our methodology is the verification of statistical stability. In Phase 1 (Path Discovery), we observed that the discovery of new unique attack paths follows a logarithmic decay. For the *Random* strategy, path discovery saturates after approximately 4,500 iterations, whereas the *Greedy* and *Max Probability* strategies saturate after 800 iterations, showing that they deterministically converge on a few optimal routes.

In Phase 2 (Metric Refinement), we monitored the standard error of the *Average Time* metric. The *Greedy* strategy shows the highest variance ($\sigma = 12.4h$), requiring a larger sample size ($N > 2000$) to achieve the desired 95% confidence interval. In contrast, the *Stealth* strategy results in lower variance ($\sigma = 4.1h$), suggesting that its behavior, while slow, is highly predictable. This confirms that our averages are statistically robust estimates, rather than artefacts of limited sampling.

E. Sensitivity Analysis: Vulnerability Distribution

To evaluate the dependence of our findings on specific vulnerability instances, we implemented a sensitivity analysis by perturbing high- and low-severity CVEs between intermediate "pivot" nodes and the target assets to assess whether the vulnerability locations influence the agent's performance. Severity labels (e.g., Low, Medium, High, Critical) refer to the CVSS [27] base score, a standard scale to assess the technical severity of vulnerabilities.

Our results show a high degree of stability. The deviations in key performance metrics were minimal, with a maximum variation of approximately 0.73% for the *Success Rate* and 0.85% for the *Average Time to Compromise* across all strategies. These marginal fluctuations suggest that, within a complex network topology, the macroscopic structural constraints (e.g., segmentation, reachability) and active defense mechanisms are more influential on adversary performance than the detailed attributes of individual vulnerabilities.

VI. DISCUSSION

Our results offer a multifaceted view of the adversarial landscape. Beyond the raw performance metrics, they reveal fundamental dynamics on the interaction between attacker intent and defensive posture. This section analyzes the "Speed-Stealth-Success" trilemma, quantifies the operational impact of security controls, and offers suggestions for security strategies.

A. The Speed-Stealth-Success Trilemma

Our results support the hypothesis of a critical operational trade-off that governs adversary decision-making: no single strategy can simultaneously maximize speed, stealth, and success rate in a defended environment. The *Greedy Value* strategy, while theoretically the most efficient in terms of path length (averaging 5.33 steps), suffers from a severe "aggression penalty." Its disregard for noise leads to rapid detection and blocking by the IDS. This confirms that in high-security environments, speed is often inversely proportional

to success probability unless the attacker possesses zero-day exploits to bypass detection.

Conversely, the *Stealth* strategy shows the concept of a "Stealth Tax." In fact, it achieves the highest success rate (13.2%) in the baseline scenario, but it pays a high cost in terms of time, requiring an average of 65.41 hours to reach the goal. This is more than double the time of the Greedy agent and the 37-hour difference represents the operational cost of evasion. For defenders, this metric is critical as it defines the window of opportunity for response. If the Security Operations Center (SOC) cannot detect and contain a threat within this 65-hour window, the stealthy attacker is statistically likely to succeed.

B. Defense-in-Depth: Spatial vs. Temporal Controls

A comparison of the scenarios supports a quantitative evaluation of the priority of different security mechanisms along distinct dimensions. The removal of the firewall (Scenario 3) does not drastically reduce the *Time to Compromise* for the Stealth strategy, but it increases the number of available paths. This further confirms that firewalls primarily act as *spatial* constraints, reducing both attack surface and attack paths. This funnels the attackers into predictable chokepoints.

On the other hand, the removal of the IDS (Scenario 2) strongly reduces the time of an intrusion, with the Stealth strategy's time dropping from 65.41h to 26.10h. This suggests that a significant value of an IDS is not merely blocking, but imposing a *temporal* constraint on the adversary. By forcing the attacker to move "low and slow" to avoid threshold-based alerts, the IDS buys valuable time for human analysts. Hence, while the firewall limits *where* attackers can go, the IDS limits *how fast* they can move. Consequently, removing either component degrades the system's resilience non-linearly, supporting the principle that Defense-in-Depth is multiplicative rather than linear.

C. The Failure of Static Risk Metrics

A significant finding is the discrepancy between static vulnerability scores and actual simulation results. The *Max Probability* strategy, which relies solely on CVSS exploitability scores, was often outperformed by the *Stealth* strategy in terms of survival, or the *Greedy* strategy in terms of raw speed in undefended scenarios. This highlights the limitations of traditional vulnerability management. A vulnerability with a high CVSS score on a deep, segmented node might be less risky because it is more complex to reach than a medium-severity vulnerability on a pivotal edge node. The simulation reveals that *reachability* and *detectability* are as critical as *exploitability*. Therefore, risk prioritization should be based on a dynamic "Attack Path Analysis" rather than isolated CVE scores, shifting the focus from vulnerability ranking to contextual risk.

D. Operational Implications for Defense and Offense

For Blue Teams, the Stealth strategy's success against active IDS necessitates a shift toward User and Entity Behavior Analytics to detect low and slow patterns. Red Teams should adopt a diverse portfolio: Greedy runs to test automated blocking and Stealth runs to challenge human analysts. Additionally, the *Random* strategy's non-zero success rate proves the fragility of security through obscurity. Red Teams should use stochastic simulations to uncover these unlikely but possible paths before engaging the live environment.

E. Threats to Validity

We acknowledge several limitations in this study. *Scalability:* The experiments were conducted on a medium-sized university network. While the results are statistically robust for the target system, distinct dynamics might emerge in hyperscale cloud environments or flat OT systems. *Strategy Adaptability:* The current adversary agents are static in their strategy (e.g., a Greedy agent remains Greedy). In reality, sophisticated actors often adapt their strategy mid-campaign.

VII. Conclusion and Future Work

This paper has presented a comparative analysis of adversary strategies using a high fidelity ST. We used a two-phase simulation methodology to improve the statistical reliability of our findings within the scope of the modeled environment. Experimental results show that while aggressive strategies are faster, stealth-oriented approaches are significantly more robust in defended environments. We also quantified the critical role of IDS in forcing attackers to expend time—a crucial resource for defenders.

Future work will focus on three key areas. The first one is the implementation of Reinforcement Learning agents that can switch strategies dynamically during an intrusion based on feedback from the environment (e.g., switching from Stealth to Greedy if detection is suspected). Then, we will integrate honeypots to measure their effectiveness in confusing the "Max Probability" and "Greedy" strategies. Lastly, we will enable the ST to automatically select and simulate actions such as patching or firewall updates.

Acknowledgements. Research partly funded by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme.

References

[1] V. Sammartino, "A framework for proactive cyber-resilience: Non-intrusive modeling for autonomous defense," in *DS-RT 2025*, 2025.

[2] C. Zhou et al., "Network digital twin: Concepts and reference architecture," Internet Engineering Task Force, Internet-Draft, Mar. 2024, 25 pp.

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[4] F. Baiardi and V. Sammartino, "From digital twins to ai agents: A synthetic data paradigm for next-generation cybersecurity," in *Artificial Intelligence in Cybersecurity: Unlocking the Power of Large Language Models*, CRC Press, 2026.

[5] F. Baiardi, S. Ruggieri, and V. Sammartino, "Anticipating Disasters through a Security Twin," in *ARES - DOD 2024*, 2024.

[6] V. Sammartino, F. Baiardi, and S. Ruggieri, "A Security Twin to Defeat Intrusions in Cyber Physical Systems," in *ESREL SRA-E 2025*, 2025.

[7] B. Strom et al., *Mitre att&ck™: Design and philosophy*, 2020.

[8] F. Baiardi, V. Sammartino, and S. Ruggieri, "Graph-based cyber defence using a security twin," in *29th DS-RT*, 2025.

[9] F. Baiardi, V. Sammartino, and S. Ruggieri, "Notline: A non-intrusive automated platform to build a digital twin," in *2025 DS-RT*, IEEE, 2025, pp. 1–8.

[10] F. Baiardi and V. Sammartino, "Simulation-powered cybersecurity: Real-time risk assessment via non-intrusive security twin," 2026.

[11] D. Lehner et al., "Digital twin platforms: Requirements, capabilities, and future prospects," *IEEE Software*, vol. 39, no. 2, pp. 53–61, 2018.

[12] F. Baiardi and V. Sammartino, "A quantitative framework for the validation of twin-based cyber defense," *Procedia Computer Science*, vol. 274, pp. 721–730, 2025.

[13] F. Baiardi and V. Sammartino, "Quantifying the impact of cvss score ordering on attack paths," EAI GOODTECHS 2026, 2026.

[14] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proc. 2002 IEEE Symp. on Security and Privacy*, IEEE, 2002, pp. 273–284.

[15] K. Kaynar, "A taxonomy and systematic review of data-driven attack graph generation and analysis," *Journal of Information Security and Applications*, vol. 27, pp. 1–21, 2016.

[16] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "Security metrics: A attack graph based approach," *Handbook of Information and Communication Security*, pp. 1–26, 2014.

[17] A. Applebaum, J. Baker, D. Beck, and M. Haase, "Attack flows — beyond atomic behaviors," *MITRE Engenuity*, 2022.

[18] J. Schwartz and H. Kurniawati, "Autonomous penetration testing using reinforcement learning," in *arXiv preprint arXiv:1905.05965*, 2019.

[19] M. Ghanem and T. M. Chen, "Reinforcement learning for intelligent penetration testing," in *2020 Second International Conference on Transdisciplinary AI (TransAI)*, IEEE, 2020, pp. 185–192.

[20] M. Standen, M. Lucas, D. Kim, D. Bowman, T. Richer, and J. Kim, "Cyborg: A gym for the development of autonomous cyber agents," in *arXiv preprint arXiv:2108.09118*, 2021.

[21] S. Suhail, R. Hussain, R. Jurdak, and C. S. Hong, "The role of digital twins in the security of critical infrastructure systems," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 094–13 106, 2022.

[22] V. Damjanovic-Behrendt, "A digital twin-based privacy enhancement mechanism for the automotive industry," in *2018 International Conference on Intelligent Systems (IS)*, IEEE, 2018, pp. 272–279.

[23] F. Baiardi, S. Ruggieri, and V. Sammartino, "AI-enabled Cybersecurity using Synthetic Data," in *Digital Twins Ecosystems and Applications, PerCom 2025*, 2025.

[24] L. Deri, M. Martinelli, and A. Cardigliano, "Ndpi: Open-source high-speed deep packet inspection," in *2014 IWCMC*, IEEE, 2014, pp. 1–6.

[25] NIST, *National vulnerability database*, 2024.

[26] F. Baiardi and V. Sammartino, "Quantifying resilience of cyber-physical systems to zero-day threats: A digital twin-based what-if analysis," ESREL 2026, 2026.

[27] The MITRE Corporation, *Cve*, 2024.