J. Parada, C. Alcaraz, J. Lopez, J. Caubet and R. Roman , "Digital Twin for Adaptive Adversary Emulation in IIoT Control Networks", *30th European Symposium on Research in Computer Security (ESORICS 2025), Toulouse, France*, vol. 16055, 2025.
http://doi.org/10.1007/978-3-032-07894-0_22
NICS Lab. Publications: https://www.nics.uma.es/publications

Digital Twin for Adaptive Adversary Emulation in IIoT Control Networks

J. Parada[1,2], C. Alcaraz[1], J. Lopez[1], J. Caubet[2], and R. Roman[1]

[1]Computer Science Department, University of Malaga, Spain
[2]Eurecat, Centre Tecnològic de Catalunya, Barcelona, Spain

Abstract

The number of threats in industrial ecosystems is increasing, especially in critical sectors, which have become a particularly lucrative target. These ecosystems have evolved into very complex interconnected systems, driven by the need to adapt to new digitalization and automation trends which extend their attack surface. In addition, the criticality of these systems makes them particularly difficult to test.
For these reasons, this paper covers the application of digital twins as the target of Adversary Emulation for the purpose of improving the security of industrial environments. This is done by involving automated and adaptive adversaries by means of reinforcement learning. Starting from an offensive strategy, these adversaries are able to adapt to the context, attacking the most critical parts of industrial systems. Adversarial attacks are driven by control theory and centrality techniques, providing a safe and efficient way to test critical industrial networks. The proposed methodology also includes the effective training and validation of adversaries by creating a probabilistic model from the analysis of digital twins. The paper provides relevant results on the development of adversarial adversaries and test models, and highlights the importance and opportunities of attack automation in virtualized environments.

1 Introduction

The current cybersecurity landscape continues to grow, especially in the management of potential threats to strategic industrial sectors [1]. This is due to the current need to create hyper-connected environments in which different technologies are incorporated to automate operational processes and intensify digitization of systems. According to Gartner's radar for 2025, there is a specific technological trend [2] that puts Artificial Intelligence (AI) and hybrid and operational computing in the crosshairs of new emerging adversary profiles [3]. All of this, in turn, leads to the need to invest in the protection of industrial environments, which is projected at USD 166.51 billion by 2032, with a Compound Annual Growth Rate (CARG) of 3.67% during the forecast period 2025-2032 [4]. In this respect, recent AI-assisted methods involving advanced prediction and detection have proven to be especially useful for complex and even unclassified attacks [5]. However, defense systems face problems such as the lack of automatic (re)training data, the use of new strategies and technologies by attackers to evade defense systems, and the strong incentives that the latter currently have against specific critical sectors. Some real cases have already witnessed the consequences of possible attacks, such as the advanced ransomware attack against Schneider Electric in January 2025 [6], and the attack on Key Tronic Corporation in May 2024 that caused data leaks and denial of services in the system's primary operations [7].

One way of tackling these situations from a proactive perspective is through Cyber Threat Hunting (CTH). With the support of Cyber Threat Intelligence (CTI), CTH is positioned within the field of active cyber-defense as one of the most relevant areas for dynamic prevention. This area is so important that it already has an investment forecast of USD 7.66 Billon by 2030 with 13.27% CARG for 2025-2030, especially when AI-based analytical prediction approaches are applied to the field of defense [8]. This is the case of Adversary Emulation (AE) as an effective cybersecurity assessment method within CTH to anticipate threat scenarios and provide guarantees of business and operating continuity [9]. This characteristic has also demonstrated in recent studies such as [10], which applies AE in cyber physical systems, and [9] where AE is applied in general-purpose systems to detect potential threats based on offensive techniques, probably designed with advanced and sophisticated attack approaches. However, the active automation of these evaluation methods and their application in very demanding

environments also forces us to explore the use of simulation technologies to emulate attacks without corrupting the integrity and functionality of real systems. Among simulation technologies, the Digital Twin (DT) stands out for its ability to deploy "machines (physical and/or virtual) or computer-based models that are simulating, emulating, mirroring or twinning the life of a physical entity" [11] with high fidelity abstraction [1]. This simulation level, under synchronization and automatic bidirectional communication criteria [12], allows the faithful recreation of scenarios, behaviors and states of the physical counterpart being imitated [1], favoring not only the AE analysis but also feedback to CTH.

Indeed, a DT can help (i) infer security vulnerabilities (probably zero-days) [13, 14, 15, 16], (ii) derive emerging exploits, and (iii) identify new and advanced *modus operandi* and attack strategies without infringing the operation and integrity criteria of its real physical counterpart [1, 17]. If, in addition, we combine this simulation capacity with AE techniques to enrich CTH processes, it is also possible to contribute to the literature with a new prevention mode, as stated in [18]. Unfortunately, there is not enough related work that uses DT for this purpose, and only [18] comes close to what is presented in this study, proving the role of DT for AE and Internet of Things (IoT) scenarios. However, the relevance of Industrial Internet of Things (IIoT) control elements and their susceptibility to attack is not explored.

For that reason, **the main contribution of this paper** is to delve deeper into this particular issue, dynamically attacking and evaluating the main control elements of IIoT-based scenarios. Thus, with the help of a DT, referred to here as *HunTwin*, it is possible to illustrate the behavior of IIoT-type control networks and establish an assessment based on emulated offenders, that will be able to dynamically derive security breaches in the control system in an adaptive way. To automate this process, these adversaries must be equipped with a Reinforcement Learning (RL) model that allows them to autonomously learn the integral nature of the system and its implicit vulnerabilities, especially of the main elements with greater dominance and control within an IIoT environment. The RL model is designed to be easily interpretable, establishing the risk and the most critical zones according to the weights assigned by the system's experience. To train adversaries, *HunTwin* also compiles a significant number of (i) digital models characterizing different IIoT control scenarios and (ii) tests based on offensive games to validate a set of threat scenarios (targeted, hybrid and to the network itself or at node level). To promote efficient learning within the approach, inspired in part by [19], multiple digital twins are deployed within *HunTwin*. These virtual representations are attacked and analyzed to create a probabilistic model of the system on which to train the RL algorithm at an early stage. This model is a simulation which represents the vulnerability of the systems's different parts to adversaries, allowing SW agents to be trained on this model efficiently and faster. During the AE stage, the DT is also applied for the validation of this approximation, testing the knowledge generated by the simulations in a real IIoT environment. This abstraction allows the system to execute the RL systems in large simulations, generating useful results for later risk assessment, intrusion impact measurements or to develop adapted early threat prevention measures.

The paper is organized as follows: Section 2 adds the related work; Section 3 introduces the DT-assisted AE architecture IIoT network deployment process, respecting the properties of structural controllability and extracting maximum control dominance. In Section 4, the adversarial model is introduced together with the necessary conditions to characterize the DT models. Section 5 shows the experimental results, all of them performed at different levels and scales of depth of study in the DT, prioritizing the AE models under an adaptive approach in the RL. Section 6 provides the paper's conclusions and outlines the future work.

2 Related Work

The increasing sophistication of malicious actors has led the cybersecurity industry to take a proactive role in dealing with threats. One of the most popular approaches has been the AE, where the defensive characteristics of a system are evaluated by skilled adversary attackers. Thanks to these offenders, it is possible to analyze attacks that are more complex than a simple vulnerability pentest in order to infer techniques and avoid future attacks. However, the main challenge this method has traditionally faced is the large consumption of resources involved, both computational and human, especially in terms of time. This is why numerous efforts have been made to automate the process of emulating a real adversary as closely as possible. The use of AE in this context has been demonstrated in recent studies [10, 20, 21, 22, 23], which focus on the usefulness of AE in a simple way, showing an offensive approach in IoT and generalist environments. Two studies whose approach is more proximate to this present research are analyzed below.

In [9], a model of CTH via AE is proposed, showing a promising proactive approach rather than relying on traditional ones such as penetration testing, firewalls, and Security Information and Event Management (SIEM) systems. This emulation consists of two main phases: an initial phase in which the system is fed with information from CTI, as well as threat reports, information from blogs and forums from which it extracts the Techniques Tactics and Procedures (TTP); and a second phase comprising the execution of the AE, as well as the construction and validation of a hypothesis with generated data,

which are fundamental parts of a CTH. In a nutshell, the proposed model is an effective method compared to typical countermeasures based on known threats, but also an efficient hunting resource compared to more traditional and less automated techniques, such as the creation of red teams.

The AE system presented in [18] is based on DT as an offensive target. MITRE Caldera is used for automated AE tests, where one compromised device attacks another target. This paper demonstrates how DTs can be a crucial tool for securing IoT networks. However, a system that takes into account adaptability, and in particular the structural vulnerabilities of large IIoT systems for its propagation, lateral movement techniques, and TTP in general, could further enrich the contribution that has been made.

In respect to the recent literature shown in Table 1, *HunTwin* is comparable in terms of external TTP sources. Furthermore, it presents adaptability to the context in which it is developed by having Artificial Intelligence (AI) models that learn from the IIoT network structural controllability characteristics, explained in Section 3.2. On the other hand, considering the infrastructure used, containers are lighter and more efficient compared to other heavier virtualization systems such as Virtual Machine (VM), without losing realism in the simulation, since it is a digital model synchronized with a cyber-physical environment. Taking advantage of this computational efficiency, *HunTwin* simulates adversary emulations over large target networks, assuming large networks such as topologies with more than 50 targets. This makes it possible to develop more sophisticated techniques taking advantage of complex network scenarios and industrial characteristics.

3 DT-Assisted Adversarial Emulation Architecture

Figure 1 illustrates the system architecture discussed in this section. *HunTwin* is based on three main Layers (L): IIoT-based physical layer (**L1**), orchestrator layer (**L2**), and DT layer (**L3**). This division allows to isolate all layers in different systems, enhancing independent scalability and security by functionality.

3.1 Layered HunTwin for security assessment

L1 comprises all the physical IIoT elements of the real world, and is in charge of periodically sending synchronization data of current states to L2. This synchronization is performed by two components in a bidirectional flow with L2. The first component is the *Network Change Listener*, which listens for changes in the real physical world and notifies changes through the *Synchronization Bridge* in L2. The second component is the *Network Updater*, which is able to receive updated data from L2 and update the real world infrastructure accordingly. Additionally, there is a group called *CTI & External Sources* that is not part of L1. It consists of the CTI sources which serve to feed the TTP of the AE models.

In contrast, L2 is responsible for computing received data and synchronizing DTs to their respective counterparts. To do this, L2 relies on a central component called *Synchronization Bridge*, which allows layers to subscribe and emit changes produced in their models. The other element of L2 is the AE module, which has two main responsibilities: the first is to conceptualize the deployment of a graph characterizing the industrial features of real control systems (e.g., controllers, SCADA, etc). These controllability characteristics define the dominance and control between the different nodes which is a feature of IIoT. This conceptual model is the *Simulated Control Graph*, obtained and synchronized through the *Synchronization Bridge*. The second responsibility is to invoke a set of adversarial software agents (hereinafter illustrated as set A) to attack the threat scenario and verify security gaps. These adversaries have two parts: an RL model acting from L2 as the brain and a real adversary in L3 as the body. The adversary model acts with the *Simulated Control Graph* as an interface to the *Synchronization Bridge*, perturbing the information coming from the cyber-physical world and adding itself to the model. This will be reflected in the DT (L3), therefore enabling the model to attack it. The adversary model communicates with the DT-adversary in L3 as a Command and Control system, and reads the response by analyzing the *Simulated Control Graph* state. This graph limits the actions that the adversarial model is able to perform in the DT, preventing it from performing illegal actions, such as instantiating itself on all nodes simultaneously.

Finally, L3 contemplates the virtualization of the DTs, simulating the properties of the physical world and performing the adversarial tasks. In order to obtain real-time information on adversarial actions, L3 also incorporates, in addition to the *Synchronization Bridge*, a sniffer capable of extracting network packets to later enable subsequent analysis. This allows, in conjunction with the stats generated by the *Simulated Control Graph*, building probabilistic models based on network behavior, as performed in Section 5.1.

3.2 Control Theory-based Environments

This section introduces DTs as a real time identical copy of IIoT networks. The main challenge of digital twins is that their production is technically expensive due to industrial high fidelity. To solve this, an algorithm for the generation of industrial control networks that emulates the structure and organization of such networks is designed. This algorithm also uses images of known sys-

Table 1: Some Adversary Emulation Researches

Ref	Year	Industry	TTP sources	Infrastructure	DT	CTI	AI agents	Approach	Large Net
[9]	2021	General	CTI, Forums, Blogs	VM	✗	✓	✗	CTH	✗
[18]	2023	IoT	MITRE ATT&ACK	VM	✓	✓	✗	Offensive	✗
[10]	2020	CPS	Internal Model	Mathematical	✗	✗	✓	Offensive	✗
[20]	2022	IoT	RouterSploit	VM	✗	✓	✗	Offensive	✗
[21]	2023	General	MITRE ATT&ACK	Cloud	✗	✓	✗	CTH	✗
[22]	2024	General	Own Model	VM	✗	✓	✗	Anti-detection	✗
[23]	2024	General	MITRE ATT&CK	Cloud	✗	✓	✓	Deception	✗
HunTwin	2025	Industrial	MITRE ATT&CK, MITRE CVE, ExploitDB, VulDB, Metasploit	Containers	✓	✓	✓	CTH	✓

tems that have vulnerabilities, which are distributed and replicated throughout the environment according to their functionality, e.g. routers, switches or linux systems.

To characterize this type of scenarios and identify attack targets with higher dominance a network topology that follows a power-law [24] must be first be prepared. Under this assumption, structural controllability rules and power dominance must be reinforced. To do this, it is also necessary to identify among the network elements, hereinafter shown as graph $G(V, E)$, the nodes with the most control that satisfy the rules of control theory [25] and power dominance [26]. These characteristics are reflected in $OR1$ and $OR2$, two types of groups calculated following the controllability conditions for the entire G network [26]:

- *OR1*: A vertex v_i of degree D observes itself and all its neighbors.
- *OR2*: If a vertex V_i of degree $D \geq 2$ is observed and has adjacent $(D-1)$ observed vertices, the last vertex becomes observed.

Once the environment and control characteristics have been calculated, the next step is to identify the vertices with the highest betweenness centrality, since most of the control passes through them. To do so, the process classifies vertices into different communities using the *Clauset-Newman-Moore* greedy modularity maximization algorithm [27] to find the network's most modularized partitions. This calculation is then used to analyze the different malicious behaviors in small interconnected groups. This is especially interesting in networks that follow a power-law distribution, since they do not contain a large level of interconnection, apart from the zones with the highest centrality. In addition, a criterion involving control characteristics and network centrality $\forall v \in V$ must be defined as a control requirement. To do so, the betweenness centrality of each vertex is calculated. Then, we define the *Scope of Compromise* value for each vertex, which represents a numerical value of how much impact it has to lose a vertex v_i to an attacker, calculated from Equation 2. This function mixes betweenness centrality, where σ_{mn} is the number of shortest paths from node m to node n, $\sigma_{mn}(v_i)$ those paths that also pass through v_i, and structural controllability-based vertex weighting w_i. These weights depend on whether $v_i \in OR1$, $v_i \in OR2$ or $v_i \notin OR1 \cup OR2$ as shown in Equation 1.

$$w_i = \begin{cases} p_1 & \text{if } v_i \in OR1 \\ p_2 & \text{if } v_i \in OR2 \quad \forall v_i \in V \quad \text{where} \quad p_1 > p_2 > p_3. \\ p_3 & \text{otherwise} \end{cases} \tag{1}$$

$$com(v_i) = \sum_{m \neq v_i \neq n} \frac{\sigma_{mn}(v_i)}{\sigma_{mn}} \cdot w_i \forall v_i \in V \tag{2}$$

Once the mathematical framework of the topology has been established, including the calculations related to controllability characteristics, the next step is the deployment and digitization of the system. Digital models are generated for each G node, the main information being the copy of the image to be virtually created. In these experiments, images of vulnerable devices are also established to be exploited by agents. Furthermore, some nodes incorporate deception strategies in order to intensify the security testing. Specifically, the position and quantity of these nodes are added randomly, never exceeding a presence of more than 10% of $|V|$ in order to intensify the defense against agents, making topology learning essential to breach the system. The difference between standard and deception nodes is that the deception nodes have no value in quantifying the damage caused to the infrastructure. In other words, these nodes are ignored by the *Scope of Compromise* calculation function in Equation 3.

$$R(a,t) = \frac{\sum_{i=0}^{|D(a,t)|} com(v_i) \forall v_i \in \{\forall v \in V : (a,v) \in M(t)\}}{\sum_{j=0}^{|V|} com(v_j) \forall v_j \in V} \tag{3}$$

Finally, for the development of an agent's intelligence, the objective function uses the same parameters as the

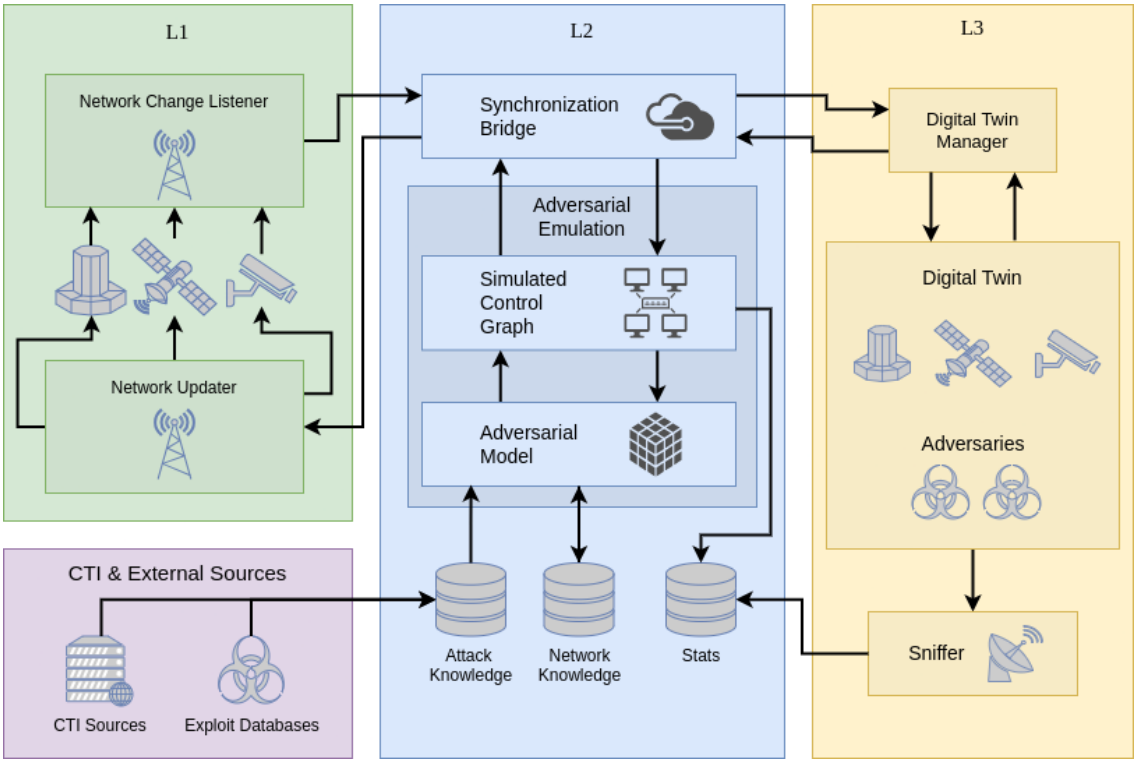


Figure 1: HunTwin layered architecture: L1, L2 and L3

Scope of Compromise calculation, but summing the compromise index of all vertices. This assumes that the malicious agent has compromised the system by exceeding a certain threshold calculated by Equation 3, which aggregates the *Scope of Compromise* of breached vertices by the attacker, establishing a ratio with respect to the total network.

4 DT-assisted Adversary Model

To formalize the DT and agents-supported emulation, we define each match scenario as an undirected multigraph of type $G = (V, E)$, where V represents the set of vertices illustrating IIoT devices, including those related to control, and E comprises the edges representing the connections between IIoT devices without isolation. Edges are represented as a tuple of two vertices and an index (k), since between two vertices there can be more than one edge representing different physical links $e = (v_i, v_j, k)$. Notice that $\forall\ v \in V$ and $\forall\ e_n \in E$ there is no static allocation of weights because the DT represents illustrations that may vary with the update of its physical counterpart.

4.1 Adversary and Reward Model Formalization

In the agent execution formal system, training and validation matches comprise the following set of variables, conditions and rules:

Time: All the games start at instant $t = 0$ and end at an instant t defined by the starting conditions. An instant t is equivalent to executing an action by $\forall a \in A$.

Players: Agents $(a \in A)$ deployed over G are initially randomly assigned to one $v \in V$. This characteristic makes the assignment $(a \rightarrow v)$ variables for each match, increasing the number of scenario combinations and avoiding initial disadvantageous situations for a same player. The dominance of vertices by agents is represented by pairs $(a, v) \in M(t)$, where t is a moment and the function M returns a set with all dominance tuples in the instance t.

Teams: Agents $(a \in A)$ only belong to a team represented by the set z, which, in turn, belongs to the set of teams represented as Z such that $Z = z_i, ..., z_j$ and $z = ai, ...a_j$. This property also follows the conditions given by Equation 4.

$$A = \bigcup_{n=0}^{|Z|} z_n \wedge |A| = \sum_{n=0}^{|Z|} |z_n| \tag{4}$$

Node state: This criterion is based on two possible states. The first state is that an $v \in V$ is dominated by an $a \in A$ (regardless of its z) in an instant t, which means that given $M(t)$ there exists the tuple (a, v). The other state is a neutral state, meaning that the node has not been dominated by any other agent. Node states also follow the restrictions shown in Equation 5.

$$|\forall v \in V, \forall t \in \mathbb{N}_0 : (a, v) \in M(t)| \in \{1, 0\} \quad (5)$$

Movement: An agent's movement is equivalent to *an attempt to join* a $v \in V$ not dominated by an $a \in A$, resulting in adding (a, v) to $M(t+1)$. As the action indicates, this is only an attempt within the game, mainly because the node itself has defensive capabilities against intrusion; even when an $a \in A$ attempts to dominate a $v \in V$, this may result in a failed move.

Cost: Movements do not have a fixed cost. In case of failure in the dominance of a $v \in V$, the cost is given by the loss of the instant t to perform a successful action. Moreover, the difficulty of dominance of a $v \in V$ is not given by v itself, but by the edge $e \in E$ along which the movement takes place. In other words the difficulty is given by attacking $v_i \in V$ through $(v_j, v_i, k) \in E$.

Objective: The objective is given by a function that evaluates whether an $a \in A$, for the end of an instant t has violated the system or not. This function takes into account z_i and the dominance state of all $(a, v) \in M(T)$. Each DT has its own target function since the violation characteristics depend on the physical counterpart it represents.

Reward: It is processed at the end of each match and $\forall a \in A$. There are three types of rewards: *Agent reward*, when $a \in A$ individually wins the match; *Team reward*, when $a \in A$ has not achieved its individual goal but $a \in z \in Z$ and z is the winning team. *Defeat reward*, which involve a defeat of a and the z to which a belongs, in which case a is negatively rewarded.

The execution rules and updates of the system are shown in the Algorithm 1. This algorithm contains two nested loops that iterate A and t, leaving its cyclomatic complexity as shown in Equations **??** and 7. On the other hand, the complexity of *update_state* and *has_won* functions are linear to the growth of $|V|$. The complexity of the *next* function grows logarithmically with respect to $|V|$, although its spatial complexity is analyzed in more detail in Section 4.2.

$$O(main) = t \cdot |A| \cdot (O(next) + O(update\_state) + O(has\_won)) \quad (6)$$

$$O(main) = t \cdot |A| \cdot (2|V| + Ln(|V|)) \quad (7)$$

4.2 Adversary Learning Process Modeling

Taking into account the system's flexibility to perform different AI models, this article follows the approach of launching adversarial attackers under RL criteria, and particularly under the *Q-learning* decision policy such as in Equation 8. In this equation, s is the current state of the game, b is the action to perform and Q is a function that returns the learned quality of a b given s.

Algorithm 1 Simulation loop for the agent execution

Input: 1. A graph $G(V, E)$
Input: 2. A set of Agents A
Input: 3. S_i, where S_i is the strategy of the agent a_i
Input: 4. A function `has_won` that given an a_i and a game state gs returns whether a_i has won
Input: 6. A function `next` that receives `gs`, an a_i, and S_i and returns a (a_i, e_i) to attack
Input: 7. A function `update_state` that receives `gs`, an a_i, and (v_i, e_i) and returns the new `gs`
Input: 8. The initial state of the game `game_state`
Input: 9. Game run time t

```
W ← ∅; it ← 0
while LENGTH(W) == 0 and it < t do
    it ← it + 1
    for all a_i ∈ A do
        (v, e) ← next(gs, a_i, S_{A_i})
        gs ← update_state(gs, a_i, (v, e))
        if has_won(A_i, current_state) then
            W ← W ∪ {A_i}
        end if
    end for
end while
return W
```

When applying *Q-learning* policies, a scalability-related challenge is the need to assess the size of the case set to be stored. In a typical approach shown by Equations 8 and 9, the possession of each node by $a \in A$ at an instant t is stored as s, and the decision to be made for each $v \in V$ is independently evaluated, leading to an exponential spatial complexity as shown in Equation 10. This type of strategy is valid in small games such as *tic tac toe* with fewer possible combinations or even in *HunTwin* with small graphs. However, bearing in mind that the match scenarios considered reach up to $|V| = 100$, this complexity is unacceptable, not only because of memory usage but also because of the number of matches that have to be played in order to develop an AI model that has experienced enough casuistry to cover the whole spectrum of possibilities.

$$A(s) = \underset{b}{\operatorname{argmax}}\ Q(s, b) \quad (8)$$

$$s = |\{(a_j, v) \mid v_i \in V,\ a_i \in z_i,\ a_j \in z_i,\ (a_i, v) \in M(t)\}| \quad (9)$$

$$O(f) = 2^{|V|} \quad (10)$$

This is why the approach has been changed for a simpler one, adapted to the type of context to be executed. The foundation of the approach of the designed attack model is inspired by Napoleonic warfare strategies and *Blitzkrieg* [28]. Napoleon's strategy consisted of deep flanking operations in which the enemy's rear was enveloped, with commands advancing along different routes to the point of attack. These characteristics make it ideal for industrial control environments, taking into account the industrial network characteristics detailed in Section 3.2. The main focus of this attack is to avoid battles of attrition, emphasizing fast and effective moves; specifically, to surround the enemy's central command area

quickly in order to paralyze it and isolate its defenses from the outside.

However, the design of this attack model is more sophisticated than just applying this strategy, in which case we would be ignoring the particular characteristics of industrial network topologies. The idea in this case is to model the agent's knowledge so that the former can learn for itself how to apply it, learning from the network's weakness and evaluating the critical points the strategy talks about.

$$A(s) = \underset{b}{\operatorname{argmax}}\ Q(s,b) \quad s = \{P(a,t), L(a)\} \tag{11}$$

$$P(a_i,t) = |\{\forall a_j : a_j \in z_k \wedge a_i \in z_k \wedge \exists (a_j,v) \in M(t)\}| \tag{12}$$

$$L(a) = v : (a,v) \in M(0) \tag{13}$$

$$O(f) = O(P) * O(L) = (|V|-1)^2 * (|z|-1) \tag{14}$$

In order to perform this Napoleonic-based strategy, as shown in Equation 11, the policy is decomposed into two functions, P and L, which correspond to Equations 12 and 13, respectively. Function L contains the initial position of each $a \in A$, allowing the model to establish optimal attack routes based on each agent's initial position. On the other hand, P consists of enumerating the number of $a \in A$ which belong to the same $z \in Z$. This way, adversaries can learn to distribute themselves over different routes and positions in G. These functions cause agents to not interfere with each other and can change the attack strategy as z decreases. Validation of whether this knowledge modeling translates into this type of strategy is tested in Section 5.2. Moreover, thanks to the simplification of the knowledge system, the spatial complexity of the algorithm decreases to quadratic, as shown in Equation 14.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{15}$$

The evaluation of victory is based on whether the agent owns a number of $v \in V$ which exceed a certain *Scope of Compromise* threshold. This threshold is given by *Scope of Compromise* calculations, shown at Equation 3 in Section 3.2. This assessment is used to support the reinforcement policy defined in Equation 15, applying a positive reward to opponents when they win and a negative reward otherwise. This equation allows adversaries to learn based on long-term knowledge, optimizing the decisions that lead to victory. The function updates the quality values of each movement since the last movement performed, propagating backwards. Furthermore, some typical RL parameters like learning and exploration rate are present, and are configured when designing the agent's strategy apart from the DT *Scope of Compromise* calculation.

5 Experimental cases and discussion

This section comprises three main experimental studies: (i) Node-level AE (**E1**), (ii) network-level AE (**E2**), and (iii) hybrid (**E3**, combining E1 and E2). E1 consists in creating probabilistic models by attacking DTs to create simulations on which to train the adversaries. On the other hand, E2 consists of training the adversaries in a simulated graph, in which they can learn about the topology and its weaknesses. Finally, E3 consists of running the trained agents on the DT to validate that the generated knowledge is applicable to a real system.

5.1 E1: Node-level AE vectors

E1 consists of studying the network's nodes individually, without taking into account the relationships they have at the network level. In this experiment, attack vectors are executed against the nodes in order to perform a probabilistic model that is able to estimate the ability to breach the system by the adversaries.

The first phase of this attack vector consists of performing a scan of all exposed services and vulnerabilities that each node has. The variability of this experiment consists of executing different forms of scanning, such as TCP SYN, UDP Scan, etc., all present in [29], involving characteristics such as the method to use, types of packets to be sent, sizes and intervals of sending, among others. The second phase consists of cross-checking the information obtained by the scan with CTI sources, in which it tries to find vulnerabilities and the corresponding exploits capable of breaching the system. These CTI sources consist mainly of MITRE CVE [30] and VulDB [31]. The third phase consists in executing different exploits from the Exploitdb [32] and Metasploit [33], and TTPs from MITRE ATT&CK [34] in order to control the system. Exploits and TTPs are mixed and concatenated until the agent is able to perform damage to the node and can see the rest of the connections and links to which they propagate. These CTI sources allow to perform real attacks and execute realistic malicious behaviors.

A large number of tests under different conditions and interfaces are performed per image in order to prepare the automatic generation of the control network graph. With these statistics, a ratio is established for each software image approximating the proportion of how many times these elements are breached by adversaries. Thanks to this ratio, fast training of the agents by means of a control graph with a large number of matches

can be performed, as shown in Experiment E2, without having to wait for the execution of a real DT.

5.2 E2.1: Net-level AE Vectors

The next experiments are the simulation of massive matches between different agents to enable them to learn about the network's design. An offensive z_i is configured and must take advantage of these elements to attack the centrality and targets in $OR1$ and $OR2$, further exploiting weaknesses in their design. To improve the efficiency of E2 and to favor the training of agents at an early stage, a simulated control graph is used instead of attacking the DT, as shown in the previous experiment. Moreover, E2 addresses two teams, A and B. Both teams are formed, respectively, of attackers and defenders with the main purpose of competing against each other and finding the fastest way to achieve their goal. Team A is based on an RL model already discussed in Section 4. Team B is based on a deterministic algorithm that handles non-dominated nodes propagating through the nodes that are reachable. The idea is that Team B represents a recovery system with a predefined logic and no learning capabilities.

Experiments are conducted on different G, with 10, 20, 50 and 100 nodes, in order to explore the performance of RL. Regarding the games, a total of 10 DTs peer node numbers have been generated, with a total of 100,000 matches for each one. Namely, there is a final total of 1 million matches for each type of scenario, limiting the number of actions to 500 moves per agent. Figure 2 depicts the E2 results.

The results obtained on the smallest size graphs, $|V| = 10$ and $|V| = 20$, show a limited performance by the attacking team, unable to overcome a winning ratio higher than that of the defending team. Although there is some dominance of the attacking team, especially in $|V| = 20$, both teams have an approximated winning ratio close to 50%. Based on these results, we conclude the following. On the one hand, victories tend to rely on the initial advantage provided by the assigned starting positions due to the simplicity of the networks. On the other hand, the deviation produced in favor of the attackers in $|V| = 20$ stems from the fact that there is an incipient complexity in the network, which makes the strategic differences between vertices more pronounced.

The results provided by $|V| = 50$ lead to the following conclusions. There is a clear dominance of Team A in the distribution of victories. This is because the network's size is already considerable, with more central and connected zones and more remote isolated zones. The anomaly of games with a clear tendency to the absence of winners is remarkable. This is due to a peculiar network topology with a star-shaped subgraph that has a large number of vertices compared to the general topography. With this interesting topology, there are many possibilities that both teams start the match at one of these points. This in turn, translates into a Nash equilibrium [35] between them to conquer the center, which is the only means of escape from this star but also the only one to be defeated if the opponent passes through it. Moreover, the results obtained by $|V| = 100$ are the most promising. Although the distribution of victories is similar to $|V| = 50$ in tendency, in this case it has greatly increased the number of victories of the attacking team, decreasing victories by the defender team. Complex networks represent an advantage for adversarial attackers, because having more nodes and more connections in the central zones allows them to learn to evade defenses easily without wasting time, and at the same time prioritizing a greater number of critical nodes.

The results of these experiments suggest promising outcomes. In networks with certain complexity and size, the adversaries are able to take advantage of network features by means of some strategies that will be discussed in Section 5.3. Although these results do not accurately represent that agents will be able to breach the system, they provide an approximation of how well they will perform in this type of environment. They are also useful to train the AI of agents before running them in a real environment or DT.

5.3 E2.2 Agent Behavior Analysis

Given the predominance of victories of the attacking teams, one additional result that can be extracted from analyzing the previous experiment is the identification of the nodes most relevant to the RL algorithm. This can be done by observing node weights, as shown in Figure 3 and in the Appendix 8 for all topologies. These topologies are depicted, respectively, with different mathematical representations showing how high the normalized qualities of vertices are through heat graphs. The representation of Figure 3.a corresponds to the arithmetic mean of the quality per node. The problem with this image is that is difficult to translate into an attacking path to see its short and long-term objectives.

In order to analyze short and long-term thinking, a normalized decay function shown in Equation 16 is applied to the different representations. This is based on the initial node from which agents start. In consequence, the assigned qualities have a decay, as they are closer from the initial position, thus pondering especially where all paths converge. This is shown in Figure 3.b, where it can be observed that the relevance falls on the nodes with more centrality, resulting in an initial priority. This criterion is also applied in reverse, setting a negative α parameter in order observe the nodes that are farther from the initial position. This can be seen in the Figure 3.c, where the objective is clearly to spread to the outer zones without completely renouncing centrality.

$$Q(a, v_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{q_i}{1 + \alpha \, d_G(L(a), v_i)} \tag{16}$$

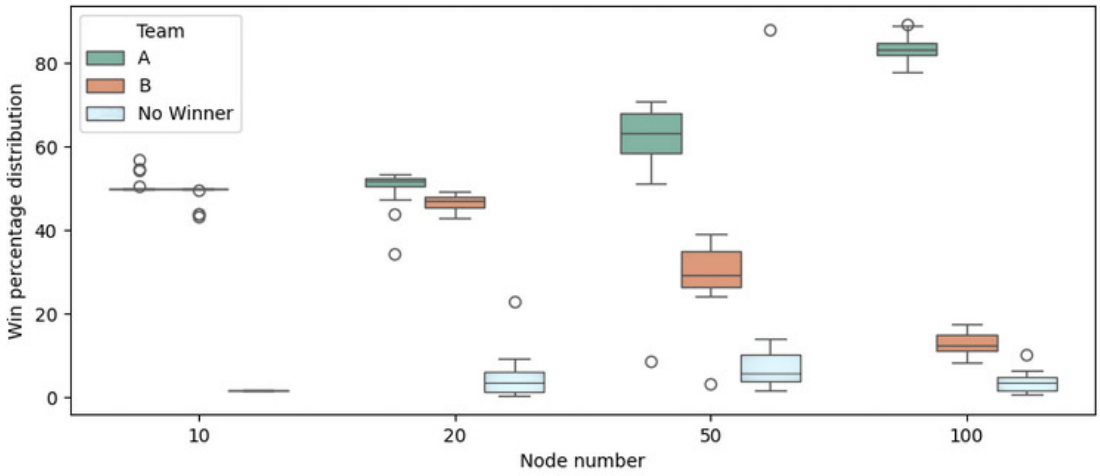
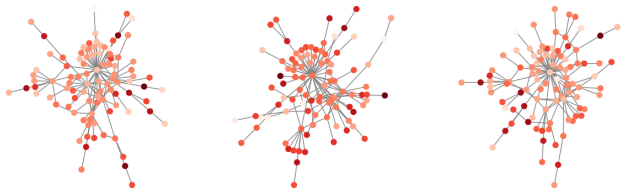


Figure 2: Distribution of percentage of victories by team.



(a) (b) (c)

Figure 3: Node weights with different representations

Analyzing the superimposed results of all the experiments in Figure 4, there is a positive tendency to value the centrality of both $OR1$ and $OR2$ within the network communities. On the other hand, there is no clear difference between $OR1$ and $OR2$ node qualities, as the attack is performed on both of them. However, when observing the same analysis on the whole network, a negative tendency is seen towards $OR1$ nodes with higher betweenness centrality. This implies that there is a tendency to attack $OR2$ nodes, avoiding $OR1$ nodes with higher centrality, which produces an enveloping effect on them. As a conclusion of these experiments, RL based agents are able to learn from the characteristics and weaknesses of IIoT networks. Not only have they been able to perform the strategy described in Section 4, but have also been able to apply it in the structural controllability context, demonstrating system adaptability.

Finally, following the results obtained and shown in Figure 4, two individual analyses of performed simulations can be found in Appendix 2. In these figures, the pattern in which the quality of $OR2$ is increasing in centrality while in the case of $OR1$ it is decreasing can be observed more clearly. Taking into account that the topology follows the power-law distribution, the enveloping effect on $OR1$ and the prioritization of $OR2$ is visible. Moreover, non-overlapping results cause the gradient to be more pronounced, making this conclusion even clearer.

5.4 E3: DT-assisted Hybrid AE

E3 intends to validate E1 and E2 in a single DT-supported IIoT environment. This experiment is executed on the DT, entailing the assembly of DT with the IIoT structural controllability theory. The most significant results of the previous experiment are taken for this test, namely those performed on industrial networks with $|V| = 100$. For this purpose, E3 launches agents and hybrid-target attacks under one topology used in Section 5.2. The results of the experiments designed for $|V| = 100$, with a total of 100 matches, are of 73% for Team A and 27% for Team B. It should be remembered that this was an approximation, so a lower result is justified. However, results are relatively similar to those obtained in E2. Note that a high multi-node DT execution combined with a smaller number of experiments increases the probability of less accurate approximation. Furthermore, the usage of 100 virtualized nodes with all communications and agents running over them implies a large computational burden which may lead to slightly different results. Despite these setbacks, the results are clearly close to the target and the modeled evidence, demonstrating the success of these experiments and of the methodology.

6 Conclusion

This paper has demonstrated the usefulness of DT technology to illustrate IIoT-based scenarios, making it an ideal adversarial target without impacting real infrastructure. In order to conceptualize virtual scenarios, a simulated probabilistic model has been defined, delimiting the offensive rules to be applied within the DT and

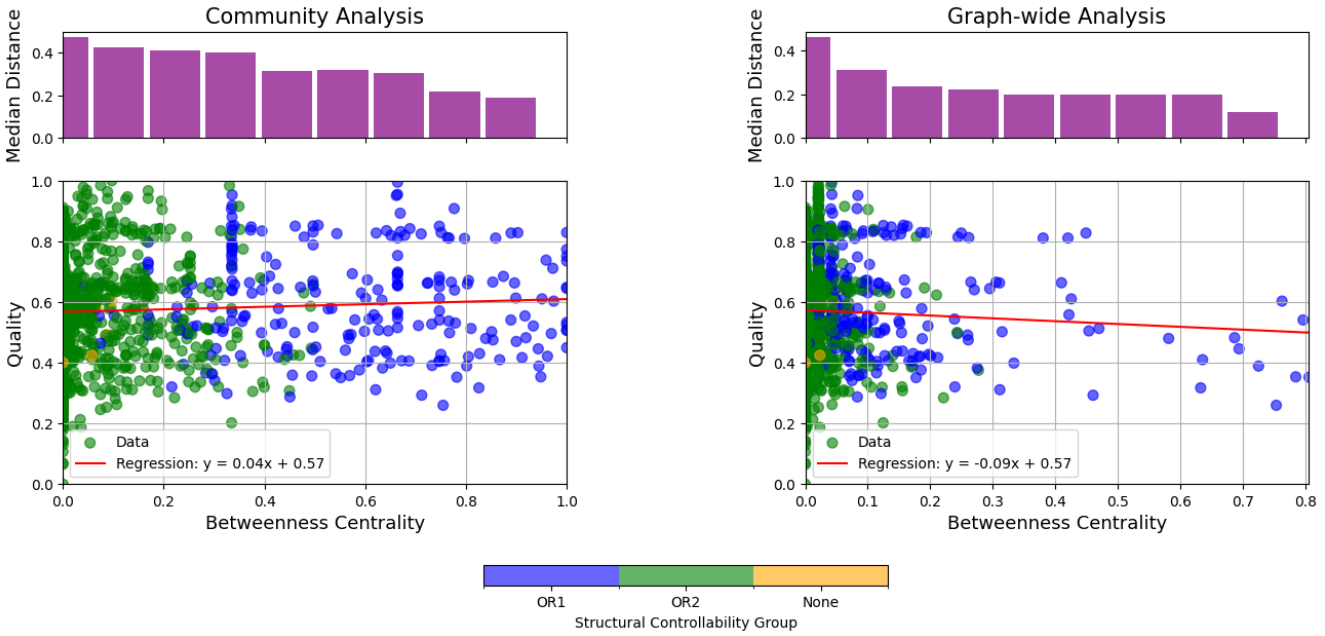


Figure 4: RL weight analysis by community and graph-wide

adapt AE to reality through simulation. The framework allows launching a set of attack strategies and adapting them to the context of the infrastructure using artificial intelligence, evaluating the most critical parts of the system in an autonomous manner. In addition, a methodology for the execution of automated AE actions has also been established. This methodology considers both the suitability of the performed strategy at the network level, as well as an initial performance-oriented approach. The paper describes the development of an innovative AE methodology that uses a DT combined and adapted to control theories to improve the security and prevention of industrial environments. As future improvements, the proposed approach shows limitations in emulating more complex strategies or behaviors as Advanced Persistence Threats that may be the subject of further research.

7 Acknowledgments

The first author is a fellow of Eurecat's "Vicente López" PhD grant program. Likewise, the work has been partially supported by SecTwin 5.0 (TED20 21-129830B-I00) funded by the Ministerio de Ciencia e Innovación, Agencia Estatal de Investigación (10.13039/501100011033), and EU "NextGenerationEU"/Plan de Recuperación, Transformación y Resiliencia; was also partially supported by the AIAS project AIAS (101131292) funded by the European Union under HORIZON-MSCA-2022-SE-01, and the 5G+TACTILE_4 Project (NEXTGENERATION.UE, Spanish UNICO 5G I+D) under Grant TSI-063000-2021-26.

References

[1] Cristina Alcaraz and Javier Lopez. Digital twin: A comprehensive survey of security threats. *IEEE Communications Surveys & Tutorials*, 24(3):1475–1503, 2022.

[2] Gartner. Gartner Top 10 Strategic Technology Trends for 2025. `https://www.gartner.com.au/en/articles/top-technology-trends-2025`, 2025.

[3] Dietmar PF Möller. Cyberattacker profiles, cyberattack models and scenarios, and cybersecurity ontology. In *Guide to cybersecurity in digital transformation: Trends, methods, technologies, applications and best practices*, pages 181–229. Springer, 2023.

[4] Skyquest. Critical Infrastructure Protection Market Size, Share, and Growth Analysis. `https://www.skyquestt.com/report/critical-infrastructure-protection-market`, 2024.

[5] Lip Yee Por, Zhen Dai, Siew Juan Leem, Yi Chen, Jing Yang, Farid Binbeshr, Koo Yuen Phan, and Chin Soon Ku. A systematic literature review on the methods and challenges in detecting zero-day attacks: Insights from the recent crowdstrike incident. *IEEE Access*, 2024.

[6] Pierluigi Paganini. Schneider Electric CACTUS Ransomware Attack. https://securityaffairs.com/158320/data-breach/schneider-electric-cactus-ransomware-attack.html, 2021.

[7] Kaspersky ICS CERT. Q2 2024: A Brief Overview of the Main Incidents in Industrial Cybersecurity. https://ics-cert.kaspersky.com/publications/reports/2024/11/08/q2-2024-a-brief-overview-of-the-\main-incidents-in-industrial-cybersecurity/, 2024.

[8] Research and Markets. Threat Hunting Market by Component, Threat Type, Deployment Mode, Industry Verticals - Global Forecast 2025-2030. https://www.researchandmarkets.com, 2024.

[9] Abdul Basit Ajmal, Munam Ali Shah, Carsten Maple, Muhammad Nabeel Asghar, and Saif Ul Islam. Offensive security: Towards proactive threat hunting via adversary emulation. *IEEE Access*, 9:126023–126033, 2021.

[10] Arnab Bhattacharya, Thiagarajan Ramachandran, Sandeep Banik, Chase P Dowling, and Shaunak D Bopardikar. Automated adversary emulation for cyber-physical systems via reinforcement learning. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[11] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE access*, 7:167653–167671, 2019.

[12] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline*, 51(11):1016–1022, 2018.

[13] Manolya Atalay and Pelin Angin. A digital twins approach to smart grid security testing and standardization. In *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, pages 435–440. IEEE, 2020.

[14] Mitra Pooyandeh and Insoo Sohn. A time-stamp attack on digital twin-based lithium-ion battery monitoring for electric vehicles. In *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 499–502. IEEE, 2024.

[15] Antonio João Gonçalves de Azambuja, Tim Giese, Klaus Schützer, Reiner Anderl, Benjamin Schleich, and Vilson Rosa Almeida. Digital twins in industry 4.0 – opportunities and challenges related to cyber security. *Procedia CIRP*, 121:25–30, 2024. 11th CIRP Global Web Conference (CIRPe 2023).

[16] Zhe Hou, Qinyi Li, Ernest Foo, Jin Song Dong, and Paulo De Souza. A digital twin runtime verification framework for protecting satellites systems from cyber attacks. In *2022 26th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 117–122. IEEE, 2022.

[17] Matthias Eckhart and Andreas Ekelhart. Towards security-aware virtual environments for digital twins. In *Proceedings of the 4th ACM workshop on cyber-physical system security*, pages 61–72, 2018.

[18] Ewout Willem van der Wal and Mohammed El-Hajj. Securing networks of iot devices with digital twins and automated adversary emulation. In *2022 26th International Computer Science and Engineering Conference (ICSEC)*, pages 241–246. IEEE, 2022.

[19] Cristina Alcaraz and Javier Lopez. Protecting digital twin networks for 6g-enabled industry 5.0 ecosystems. *IEEE Network Magazine*, 37(2):302–308, 2023.

[20] Shady Shahin and Hassan Soubra. An iot adversary emulation prototype tool. In *2022 5th International Conference on Information and Computer Technologies (ICICT)*, pages 7–12. IEEE, 2022.

[21] Saeid Sheikhi and Panos Kostakos. Cyber threat hunting using unsupervised federated learning and adversary emulation. In *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 315–320. IEEE, 2023.

[22] Vittorio Orbinato, Marco Carlo Feliciano, Domenico Cotroneo, and Roberto Natella. Laccolith: Hypervisor-based adversary emulation with anti-detection. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[23] Michael Kouremetis, Dean Lawrence, Ron Alford, Zoe Cheuvront, David Davila, Benjamin Geyer, Trevor Haigh, Ethan Michalak, Rachel Murphy, and Gianpaolo Russo. Mirage: cyber deception against autonomous cyber attacks in emulation and simulation. *Annals of Telecommunications*, pages 1–15, 2024.

[24] II Power Laws. Generating network topologies that obey power laws.

[25] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Parameterized power domination complexity. *Information Processing Letters*, 98(4):145–149, 2006.

[26] Cristina Alcaraz, Estefanía Etchevés Miciolino, and Stephen Wolthusen. Multi-round attacks on structural controllability properties for non-complete random graphs. In *Information Security: 16th International Conference, ISC 2013, Dallas, Texas, November 13-15, 2013, Proceedings*, pages 140–151. Springer, 2015.

[27] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[28] Hal Brands. *The new makers of modern strategy: From the ancient world to the digital age*. Princeton University Press, 2023.

[29] Gordon Lyon. Nmap network scanning - port scanning techniques. `https://nmap.org/book/man-port-scanning-techniques.html`, 2025.

[30] The MITRE Corporation. Common vulnerabilities and exposures (cve). `https://www.cve.org/`, 2025.

[31] VulDB. Vuldb - the vulnerability database. `https://vuldb.com/`, 2025.

[32] Offensive Security. Exploit database (exploit-db). `https://www.exploit-db.com/`, 2025.

[33] Rapid7. Metasploit framework - penetration testing software. `https://metasploit.com/`, 2025.

[34] MITRE Corporation. Mitre att&ck framework. `https://attack.mitre.org/`, 2025.

[35] David M Kreps. Nash equilibrium. In *Game theory*, pages 167–177. Springer, 1989.

8 Appendix

APPENDIX 1: Figure 5 shows the results of all the topologies generated in the experiments of Section 5.2 with $|V| = 100$, as well as the normalized arithmetic mean of all the RL weights.

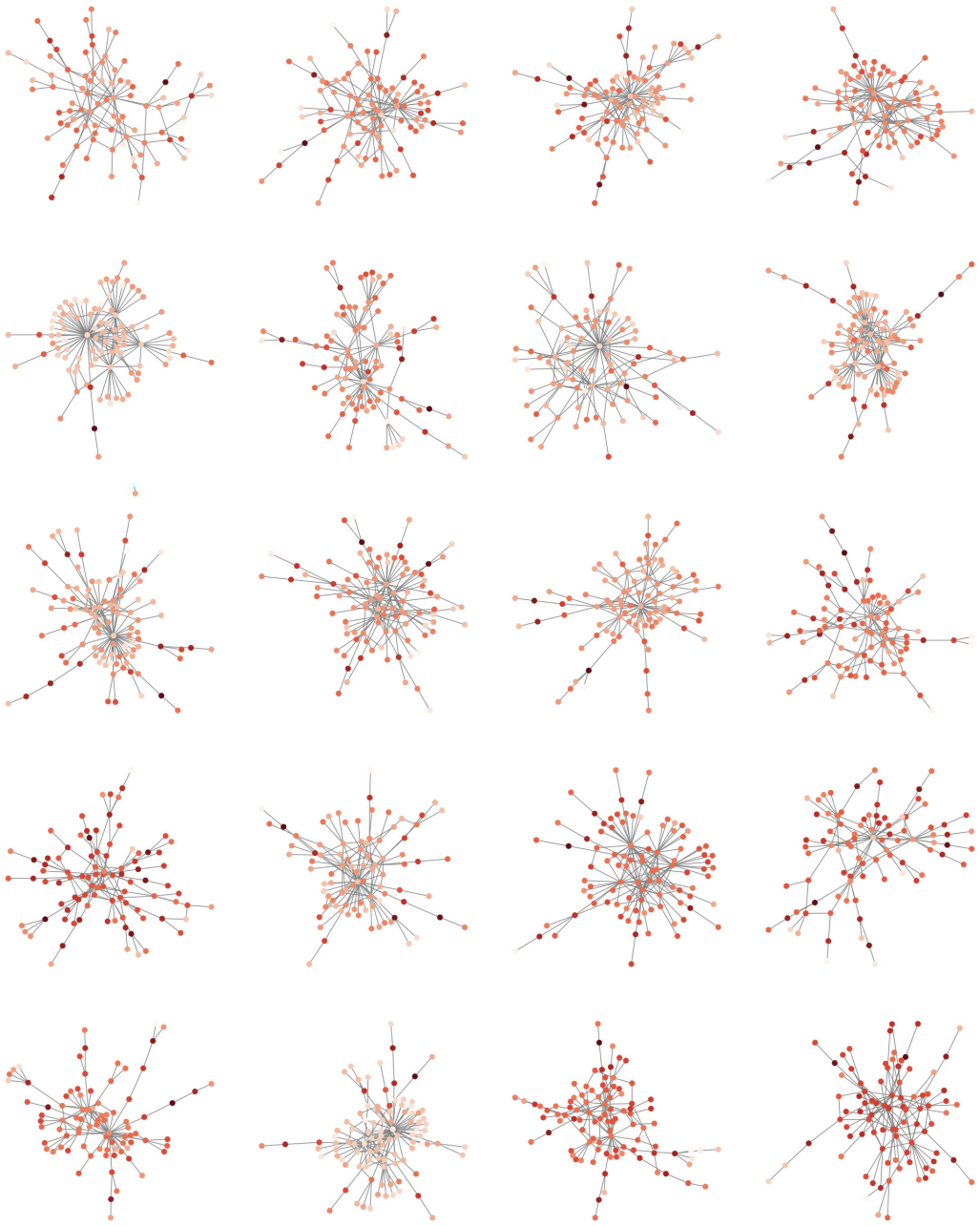


Figure 5: Topology of the DTs and normalized average RL weights.

APPENDIX 2: Figures 6 and 7 show a comparative analysis of centrality, median distance, the structural controllability group to which they belong and a regression with their trend as a function of the quality assigned by the RL algorithm.

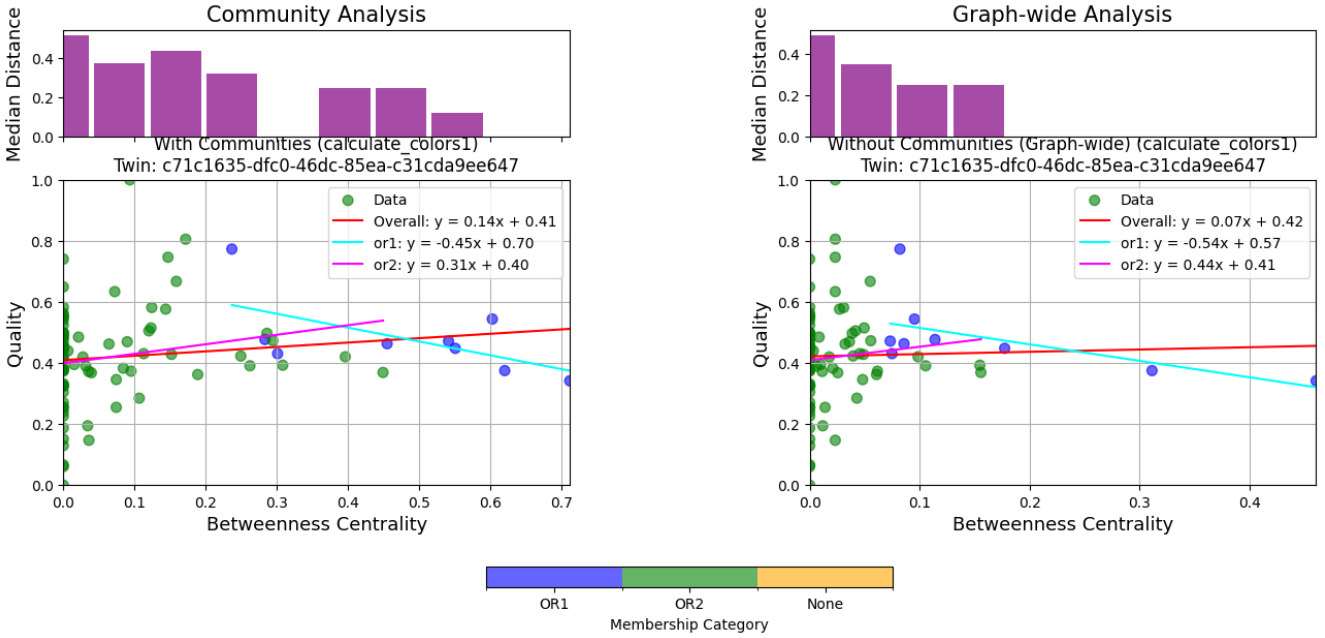


Figure 6: Centrality analysis over controllability groups 1

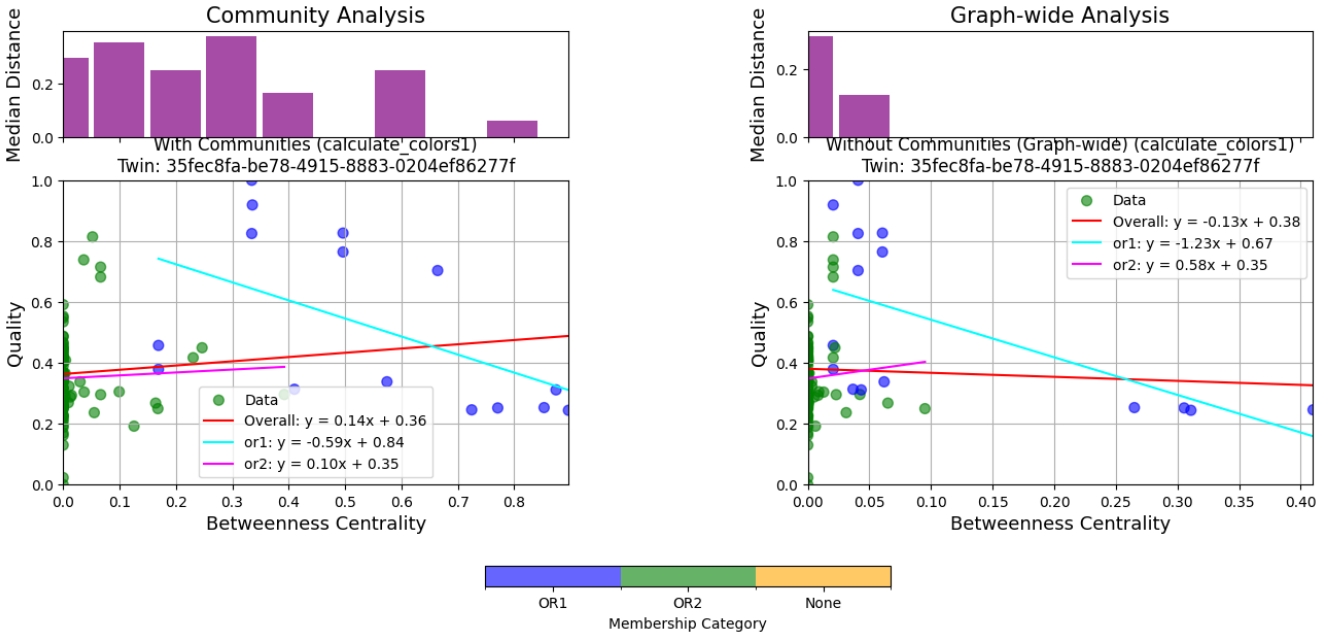


Figure 7: Centrality analysis over controllability groups 2