Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P7829-cd

A Security Twin to Defeat Intrusions in Cyber Physical Systems

Fabrizio Baiardi
Dipartimento di Informatica, Università di Pisa, Italy. E-mail: fabrizio.baiardi@unipi.it

Salvatore Ruggieri
Dipartimento di Informatica, Università di Pisa, Italy. E-mail: salvatore.ruggieri@unipi.it

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

Cyber risk assessment and management have to face a dynamic risk landscape so that probabilities of interest cannot be estimated using historical data. This paper advocates the adoption of synthetic data generated by combining adversary simulation with digital twin technology. A security twin of a cyber physical system (CPS) extends an inventory of the system with information on current vulnerabilities and attacks. By describing threat agents through other twins, we can supply the twins with a platform that simulates the strategies of threat agents to discover how they exploit vulnerabilities and implement their intrusions. To analyze alternative scenarios, a Monte Carlo approach is adopted that runs multiple independent simulations. This produces an intrusion graph that faithfully can describe rapidly evolving environments and results in more accurate risk management and better resilience of the system in spite of data shift. Initial experimental results support the effectiveness of security twins in accurately modeling intrusions. The synthetic data produced by the simulations can also be used to train AI tools to defend a CPS.

Keywords: security twin, adversary simulation, data shift, synthetic data, AI in cybersecurity, Monte Carlo simulations

1. Introduction

As cyber physical systems (CPSs) become more and more interconnected and pervasive, several threat actors (simply, actors) attempt to orchestrate intrusions to control and manipulate CPS components. To this purpose, actors exploit vulnerabilities in software, hardware, and even humans, such as users and system administrators [6, 4, 20].

The continuous evolutions of actors result in a highly dynamic risk scenario with frequent statistical distribution shifts that prevent the usage of historical data to estimate probabilities of interest, and to train AI models [1] to proactively identify intrusions. This dynamicity, combined with the increasing size and complexity of CPSs, exacerbates the challenges of effectively managing security also because of the lack of formal models and robust methodologies.

Inspired by digital twin technology [9], we advocate the generation of synthetic data through adversary simulations of intrusions. The simulations use executable models, or *digital twins*, of both the target CPS and the actor. The twin of the CPS is a *security twin*, as it abstracts from implementation details and it is focused on security and safety issues. The twin of the actor is an *attacker twin*, that specifies the attacker behavior and its strategies. To produce synthetic data on intrusions, we apply a Monte Carlo method that runs multiple independent simulations of intrusions. The generated data can be analyzed to improve the robustness of the CPS or to train AI-based defensive models.

Twin simulations of intrusions in a CPS can produce a graph, called an intrusion graph, that describes not only intrusions in isolation, but also how they relate each other, e.g., in terms of the shares of exploited vulnerability. An analysis of such a graph help defenders to pinpoint weakness in authentication, in programmable logic controllers (PLC), or in communication protocols. Overall, an intrusion graph provides actionable insights and synthetic data to counteract evolving threats in dynamic environments.

Sect. 2 defines intrusions and their description. Sect. 3 discusses the simulation of intrusions using twins and their representation through an intrusion graph. Sect. 4 analyzes the problem posed by the dynamic risk scenario. Sect. 5 focuses on the analysis of the intrusion graph and how AI can improve it. Sect. 6 describes fidelity, ie how to guarantee that the output of simulations accurately describe intrusions. Sect. 6 presents preliminary experimental results.

2. Executing and Describing Intrusions

We consider intelligent actors with a predefined goal, a set of access rights on the CPS resources they aim to acquire. Each actor has only one goal and actors do not cooperate. An intrusion is a sequence of actions of an actor to reach its goal and control some physical or logical resources [2, 14]. The MITRE ATT&CK Matrix [18] define possible actions of an actor . An actions such a vulnerability scan collects information to discover the CPS components and their potential weaknesses while other actions are attacks that exploit the weaknesses of the target CPS components to gain some access rights. Lastly, a few actions manipulate some components to produce an impact, e.g., a damage. An action may fail because of the environment or for other reasons. Each action is paired with known properties that include the success probability, the execution time, the noise it generates (the probability of being detected by an IDS), and pre- and post-conditions that describe, respectively, the rights to execute the action and the information and the rights it returns if successful. These properties are deduced from the description of the vulnerability in some databases [21, 12]. An intrusion is successful if, after its last action, the actor reaches its goal. To estimate the overall risk, we need to know any intrusion actors can implement and its success probability.

Each action is described by a tuple:

<A, IPs, IPd, precond, postcond, information>

where: A is an action [18]. If A is an attack, it also details enabling vulnerabilities. *IPs* and *IPd* are the IP addresses of, respectively, the source node where A is initiated and the target node of A. *precond* and *postcond* are, respectively, the access rights the actor needs to execute A those granted to the actor after performing A. The *information* field is the knowledge A returns.

As an example, assume that A runs a scanner. Then, *IPs* is the address of a node that runs the scanner and *IPd* is the target node. Here, the *information* field includes the vulnerabilities the scanning returns. Here *precond* is empty because the scanning needs no access right. The *postcond* field is empty and the information one includes any discovered vulnerability.

The current state of an actor Ag consists of the access rights and the information it has previously acquired. An action A is *enabled* in a state if Ag owns the rights in its precondition. The strategy of Ag chooses the action it executes at each step among enabled actions. As a consequence, a sequence of actions S is an intrusion of Ag if the first action in S belongs to the *attack surface* of Ag, i.e., it is enabled by Ag legal access rights. Furthermore, the union of the legal access rights of Ag and all those returned by the actions in S before A includes the precondition of A.

An intelligent actor never executes an action that returns information or access rights it already owns. Furthermore, it repeats failed actions only when conditions change (e.g., new information is obtained, vulnerabilities are reassessed, or the action is attempted on another target node).

An intrusion Is is *successful* if, at the end of Is, Ag owns the access rights in its goal. Otherwise, Is fails. A is *useless* in an intrusion if Ag does not need the privileges or information A grants to achieve its goal. Ag executes useless actions because it has not collected enough information on CPS. Ag can build distinct intrusions to the same goal. Their number influences the overall risk and defines the degree of freedom of Ag.

3. Emulating Intrusions using Twins

This section presents the security twin and the actor twin. Next, we discuss adversary simulations.

A twin is a digital model to simulate the entity behaviors, monitor its ongoing status, recognize internal and external complexities, detect abnormal patterns, reflect the entity performance, and

predict future trends []qi2018digital. Both the security twin St(S) and the actor T(Ag) twin are formal models of, respectively, a CPS *S* and the actor *Ag*. They both focus on those attributes to accurately model the behaviors of, respectively, *Ag* and *S* in an intrusion.

3.1. *The Security Twin*

St(S) is an enriched inventory of hardware and software modules of *S* that describes:

a) the hardware nodes and their connections,
b) the software modules with their operations,
c) the mapping of modules onto nodes and their configurations,
d) the accounts on each node, and their rights,
e) any vulnerability of each module or node, the attacks it enables, and their properties,
f) routing and filtering rules,
g) the logical connections among modules the previous rules determine,
h) the subnet, or the endpoint, each intrusion sensor monitors and the probability it detects an intrusion,
i) hierarchical relations, i.e., the one between a hypervisor and the virtual machines it runs,
j) information flows among the modules.

Alternative module configurations may result in distinct vulnerabilities. For each operation of a module of *S* there is an access right *Ag* can acquire. Hierarchical relations and information flows determine dependencies among modules. For example, access rights on a hypervisor allow the control of the virtual machines it manages. Access rights on the source of an information flow enable the manipulation of the values it transmits to the receiver.

Pre- and post-conditions exist also for actions, and they determine how *Ag* strategy maps the current state of *Ag*, i.e., the access rights and the information the previous actions have returned.

St(S) also describes (human) user classes of *S* and pairs each class with access rights and the probability that a user in the class is the victim of an attack implemented by stealing authentication information or by phishing.

The starting point to build St(S) is an inventory of *S* with information on nodes, connections, and configurations. The inventory is built by inventory management tools and it is enriched with information on vulnerabilities that a vulnerability scans returns. Alternative security twins of *S* differ in the number of details on the modules, the granularity of operations and the corresponding access rights. Further details concern the behavior of the components. For example, if St(S) neglects intrusion sensors, any analysis using it returns worst-case results because it cannot model intrusion detection. A more accurate twin can model the detection and the resulting failure of an intrusion. However, to minimize the simulation overhead, a security twin never describes in full detail the CPS, its inputs, and its computations.

3.2. *The Actor Twin*

T(Ag) describes the strategy of *Ag*, its initial access rights and information, and its goal. The actor attack surface includes any attack *Ag* may implement to start an intrusion.

The strategy of *Ag* [2] maps *Ag*'s current state, its goal, and the target *S* into the action to execute. The status of *Ag* includes the access rights and the information it has collected. If the strategy returns an action that is an attack, it also returns a target module on a target node and the vulnerability to exploit. Several alternative strategies are possible. For example, a strategy may prefer actions to collect information and select an attack only when it cannot collect further information. Instead, another strategy may execute an attack as soon as the actor owns the privileges in the attack precondition. This may speed up the escalation at the expense of useless attacks. Some strategies include social engineering attacks, while others only exploit vulnerabilities in the CPS modules.

The state of *Ag* also includes a memory that records the last actions and their result (success or failure). A small memory implies that *Ag* forgets its failures quickly and may repeat an action even after many failures.

3.3. *The Intrusion Graph*

A simulation is a coevolution of the security twin and the actor twin, starting from the actor's attack

surface. It consists of a sequence of steps, in discrete time where each step applies the actor strategy, executes the action it returns, and determines its success or failure using the corresponding success probability in the security twin. Each step updates the actor's status if an action is successful. A step also considers the reaction of the CPS to the action. For example, an action fails if a sensor detects the action. In turn, this may result in the failure of the intrusion.

A simulation is successful when the actor reaches a goal. It fails when no action can be selected, or a predefined number of steps have been executed without reaching the goal, or a predefined threshold on the simulated time has been reached, or the CPS detects an attack. The sequence of actions of a successful simulation leads to a successful intrusion, see Sect. 2.

We represent intrusions of *Ag* against *S* through an intrusion graph Int(Ag, S) where each node represents a distinct state of *Ag* and each arc is labeled by the action that enables the corresponding state transition. Self-arcs represent failed action. Each simulation of *Ag* against *S* produces a sequence of nodes, i.e., a path, and distinct simulations produce a set of paths that are merged into Int(Ag, S) by mapping into the same node those that in distinct paths represent the same status. All the paths start from the node that represents the legal status of *Ag*, i.e., the status when the intrusion has not started yet. Two nodes of Int(Ag, S) may be connected by several arcs with distinct labels. Not all intrusion paths necessarily end in a successful state where *Ag* achieves its goal. Paths may also end due to failure, such as the inability to select further actions, or because the maximum allowed runtime has been reached.

4. Intrusion and Data Shifts

Effective management of risk due to *Ag* requires information on the intrusions it can implement, the vulnerabilities they exploit, and their success probability. This information cannot always be discovered using data collected on past intrusions against the same or similar CPSs because of data shifts, namely because the underlying probability distributions change too quickly.

As an example, historical data for assessing the risk due to *Ag* quickly become obsolete if:

- *Ag* adopts new, previously unseen strategies,
- new vulnerabilities are disclosed, altering intrusion dynamics,
- the CPS deploy new applications, changing the attack surface,
- logical or physical connections are modified, affecting network paths,
- nodes are added or removed, impacting overall risk.

The reason for obsolescence is that these events may result in new intrusions and/or change the success probabilities of intrusions. In other words, these events produce a shift in the distribution of the data that affects the probabilities of interest. Consequently, it is not convenient or effective to use historical data to train an AI model [13, 11]. In more detail, the new behaviors of *Ag* may change both its actions or the sequence of actions in its intrusions. For example, *Ag* may increase the time it spends collecting information before selecting the vulnerability to exploit. Hence, *Ag* has more accurate information available, and this reduces both useless actions and failures. We can update *Ag*'s success probability after observing several intrusions, but *Ag* will not repeat a successful intrusion to allow for better probability estimates. Further shifts are due to the discovery of a vulnerability, the deployment of an application, or a new network connection because they may enable new attacks and new intrusions. These intrusions cannot be predicted using historical data because new vulnerabilities increase in a non-linear way the number of intrusions. Lack of information on intrusions may result in a black swan or a perfect storm disaster [3].

An alternative is to adopt synthetic data produced by adversary simulations using security twins and actor twins. In more detail, we advocate the adoption of Int(Ag, S) to produce data to support risk assessment and management. Int(Ag, S) is built by applying a Monte Carlo method that runs multiple independent simulations using St(S) and T(Ag). By timely updating both twins, we can generate relevant and up-to-date synthetic data as

soon as either S of Ag changes. This reduces the impact of data shifts. The data the analyses of Int(Ag, S) return can be used to train robust AI models without relying on historical data. This improves both the prediction of intrusions and the accuracy of their detection by leveraging advancements in intrusion detection systems and the effective use of datasets to model evolving threats [10, 16, 19, 17].

5. Intrusion Graph Analysis and AI

An analysis of Int(Ag, S) can identify and mitigate critical vulnerabilities before Ag exploits them. We discuss factors influencing the accuracy of Int(Ag,S) which, in turn, influence the one of alternative analyses.

In terms of Int(Ag, S), an intrusion is a success path from the initial state to a final one. Any success path that is not an intrusion in the target CPS is a *false positive*. An intrusion that has no success path in Int(Ag, S) is *false negative*. We can formally prove that no false positive exists in Int(Ag, S) if St(S) accurately describes the vulnerabilities of S. The probability of a false negative in Int(Ag, S) decreases with the number of independent simulations we run to build it. Independence ensures that an action in a simulation does not affect those of other simulations and hence does not affect the probability an intrusion is implemented. Each path in Int(Ag, S) has a multiplicity that depends on the number of simulations mapped into it.

We can estimate the probability of an event of interest in intrusions of Ag through Int(Ag, S). As an example, the success probability of an intrusion is the ratio of the number of simulations mapped into the corresponding success path and the overall number of simulations. More in general, the probability of an event is the ratio between the multiplicity of the path where it occurs over the total number of simulations. [15] and [7] describe alternative approaches. Using Int(S,Ag) when can compute the probabilities of events such as:

(1) Ag reaches its goal,
(2) Ag attacks a module,
(3) Ag follows a success path,
(4) an intrusion takes less than a specified time.

Also conditional probability can be estimated e.g., Ag reaches its goal after attacking a given component. Low-probability, high-impact events require many simulations for robust estimation of path multiplicity. They can be executed in parallel (due to the independence assumption) and without disturbing the target CPS in its operations.

5.1. *AI for Intrusion Analysis*

To select countermeasures, intrusions are analyzed using AI and Data Mining techniques to identify the most dangerous intrusion patterns and then deploy the most effective countermeasures.

A twin approach supports the proactive exploration of alternative strategies Ag may apply, allowing defender teams to anticipate evolution in these strategies.

Classic AI planning [23, 22] is the foundation for defining actor strategies. Recent research has explored the effectiveness of adopting large language models to plan and implement intrusions. For example, [5] provides a first example of how an intrusion can be automatically planned, showcasing the potential of simulations to produce data to train an AI-driven planner. The resulting strategies enable defenders to anticipate and manage high-risk scenarios and to proactively deploy proper countermeasures, even against the most advanced attackers.

6. Model Fidelity Issues

This section details three major sources of uncertainty in the security twin that affect its fidelity [8], i.e. its ability to accurately describe the features of interest. These sources are related to attacks against, e.g., Denial-of-Service attacks, user vulnerabilities in phishing attacks, and attacker strategies. Each presents unique challenges that must be addressed to improve the accuracy of simulation outputs and the resulting effectiveness of risk management.

6.1. *Denial of Service Attacks*

The primary sources of uncertainty in modeling these attacks include:

- Time-Varying Conditions,

- Adaptive Attack Patterns,
- Non-Stationary Attack Characteristics.

The network load and traffic patterns fluctuate rapidly and affect both detection thresholds. This increases the complexity of anticipating the exact conditions that determine the success or failure of the attack. Modern DoS attacks are rarely static because attackers tune their techniques to evade detection and overcome defensive measures. This creates a moving target for the simulation models. Statistical properties of DOS attacks, such as packet rates, burstiness, and duration, vary over time. Hence, models based on historical data may not capture sudden shifts or emergent behaviors.

Monte Carlo simulations can address these uncertainties by incorporating randomness into the simulation process. However, the very nature of stochastic simulations implies that the accuracy of the model and its output should be carefully quantified and managed.

6.2. *Phishing Attacks*

Uncertainties in modeling these attacks arise from variability in user security awareness and technical expertise. This results in a vulnerability range due to differing training levels and behavioral factors. Furthermore, recent phishing campaigns, improved training, or high-profile breaches can quickly alter the range.

Adaptive techniques that integrate real-time threat intelligence and behavior analytics are essential to refine these parameters.

6.3. *Strategy Modeling*

Modeling attacker strategies is uncertain due to first of all the large number of strategies attackers have available. Furthermore, the choice of an action is influenced by system state and historical interactions are modeled probabilistically. This may not fully capture adaptive adversary behavior. The interplay between evolving offensive strategies and defensive measures adds further complexity.

To address these uncertainties, a range of parameter values is considered, and Monte Carlo simulations are run across the extreme cases, thereby exploring alternative scenarios and reducing overall uncertainty.

7. First Experimental Results

We describe a first prototype of a twin-based simulation platform and show an example of an analysis using its outputs.

7.1. *Security Twin*

The current security twin includes a database that describes vulnerabilities in system components and related attacks with their preconditions, post-conditions, execution times, and noise levels. It is built using data from sources such as the NVD and CVE databases [21, 12]. This simplifies simulations and reduces replications because a single storage for vulnerability information.

7.2. *Actor Database*

This database stores the twins of actors that implement the intrusions to be simulated. Data to build the twins are gathered from threat intelligence reports and security analysis.

7.3. *Implementation*

The platform returns an intrusion graph to be analyzed to discover dynamic interactions between actors and the target system. As previously described, this graph is the output of multiple independent twin-based simulations, the vulnerability database, and a profile from the actor database.

Strategies and Intrusion Planning

The current prototype supports four strategies:

- **Random**: selects the next action randomly.
- **Privilege First**: prefers the acquisition of access rights.
- **Success First**: prefers actions with the highest success probability
- **Noise First**: selects actions with the lowest noise.

Future versions will implement strategies based on AI planning methods.

Data Shift Analysis

Data shifts can occur for any change in the CPS or actor behavior. An important reason for an expected and positive change is the patching of

vulnerabilities chosen through some simulations. The impacts of all these shifts can be estimated through new simulations.

Simulation Outputs

The outputs of multiple simulations support an estimation of the probabilities of the events of interest as well as the evaluation of centrality measures for the CPS nodes weighted w.r.t. successful intrusion probabilities. The latter is important for patch scheduling.

The output of simulations can be used as the input for AI and Data Mining analyses to discover:

- **Attack Paths and Sequences**: Detailed sequences of actions in intrusions,
- **Success Rates**: Estimated intrusion success probabilities, True Positive Rate (Recall), True Negative Rate (Specificity),
- **Time Metrics**: Time to reach goals.
- **Impact Assessment**: Evaluations of attack impacts, based on a cost model related to the business processes using the CPS.

7.4. *A First Example*

We experiment with a simplified CPS where components include servers, workstations, network devices, and users. The overall system architecture consists of 60 nodes and 15 users, featuring a diverse array of components including servers, workstations, a router, a firewall, and a database server. Fig. 7.4 illustrates a subset such a system.

Node	Type	Connections
Node A	Server	B, C
Node B	Workstation	A, D, E, User 1
Node C	Router	A, D, E
Node D	Workstation	B, C, F, User 2
Node E	DB Server	B, C, F
Node F	Firewall	D, E
User 1	User	B
User 2	User	D

We assume the system nodes are affected by a total of 50 vulnerabilities. Despite efforts to maintain security through regular patching, around 15 vulnerabilities remain unaddressed.

Our simulation software is implemented in Python, using NumPy for numerical computations and NetworkX[a] for network analysis.

We have run simulations to analyze alternative scenarios to investigate the baseline attack paths, the selection of countermeasures, and the existence of new actors. A first experiment with 100 simulations shows that the common attack path is from node A to B to E and the attacker success probability is 65% if a phishing attack targeting User 1 is successful. The average time to compromise is 2 hours.

After patching vulnerabilities and improving firewall rules, we run 100 simulations and according to this experiment there is a new scenario with a new attack path from Node A to C to E, the success probability is reduced to 30%. and the average time to compromise increases to 4 hours.

The deployment of countermeasures resulted the choice of a distinct path due to the patching of some vulnerabilities. The ability to predict the reaction of an actor to a countermeasure is fundamental for risk management.

The simulation of an advanced strategy aimed at minimizing noise results in an attack path from A to B to C, with a success probability of 75% and an average time to compromise of 1.51 hours. This shows that a sophisticated threat agent can adapt to existing security controls, modifying its strategy to optimize success probability and minimize time to compromise.

To assess the impact of emerging vulnerabilities, we have run a what-if analysis that introduces a new vulnerability on node C. This results in a significant change because the attack path is from A to C to E, its success probability is 90% and the average time to compromise 1.04 hour. This emphasizes how even minor changes, the introduction of one vulnerability, heavily influence intrusions and affects both success probability and attack paths.

8. Conclusion

CPSs face significant threats because of vulnerabilities in software, hardware, and human ele-

[a]https://networkx.org/

ments, often using physical or phishing attacks. Although countermeasure selection needs intrusion data, rapidly evolving risk scenarios reduce the accuracy of prediction using historical data. Merging digital twins with adversary simulation can generate synthetic data to counter data shifts. AI integration further improves the emulation of intelligent threat actors.

Future work will refine statistical analysis of the intrusion graph, expand applications across CPSs, and leverage AI to strengthen defenses against evolving cyber threats.

References

1. D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.
2. A. Applebaum, J. Baker, D. Beck, and M. Haase. Attack flows — beyond atomic behaviors. *MITRE Engenuity*, 2022.
3. T. Aven. On the meaning of a black swan in a risk context. *Safety science*, 57:44–51, 2013.
4. M. Bada and J. R. Nurse. The social and psychological impact of cyberattacks. In V. Benson and J. Mcalaney, editors, *Emerging Cyber Threats and Cognitive Vulnerabilities*, pages 73–92. USA, 2020.
5. G. D. P. et al. ChainReactor: Automated privilege escalation chain discovery via AI planning. In *33rd USENIX Security Symposium*, pages 5913–5929, Philadelphia, PA, Aug. 2024. USENIX Association.
6. S. Furnell, H. Heyburn, A. Whitehead, and J. N. Shah. Understanding the full cost of cyber security breaches. *Computer fraud & security*, (12):6–12, 2020.
7. A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. 2006.
8. J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller. Synthetic data - what, why and how? *CoRR*, abs/2205.03257, 2022.
9. C. McLean, Y. T. Lee, S. Jain, C. Hutchings, and C. Hurchings. Modeling and simulation of critical infrastructure systems for homeland security applications. *NIST, Gaithersburg, MD, USA, Tech. Rep. NISTIR*, 7785, 2011.
10. Y. Mirsky et al. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv*, 2018.
11. J. Moreno-Torres, T. Raeder, R. Alaiz-Rodrguez, N. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012.
12. NIST. National vulnerability database, 2024.
13. J. Quinonero, M. Sugiyama, N. Lawrence, and A. Schwaighofer. *Dataset Shift in Machine Learning*. MIT Press, USA, 2009.
14. M. Ryan. *Ransomware Revolution: The Rise of a Prodigious Cyber Threat*. Springer Cham, Berlin, 2021.
15. D. W. Scott. *Multivariate density estimation: Theory, practice, and visualization*. 2015.
16. F. Shahid et al. Cyber threat intelligence using deep learning: An overview. *IEEE Access*, 8:212345–212356, 2020.
17. A. Storkey. When training and test sets are different: Characterizing learning transfer, dataset shift. In *Machine Learning*, pages 3–28, 2009.
18. B. Strom et al. Mitre att&ck™: Design and philosophy, 2020.
19. A. Subbaswamy and S. Saria. From development to deployment: Dataset shift, causality, and shift-stable models in health ai. *Biostatistics*, 21(2):345–352, 2020.
20. K. Thakur, M. L. Ali, N. Jiang, and M. Qiu. Impact of cyber-attacks on critical infrastructure. In *BigDataSecurity/HPSC/IDS*, pages 183–186. IEEE, 2016.
21. The MITRE Corporation. Cve, 2024.
22. Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan. Embodied task planning with large language models. *ArXiv*, abs/2307.01848, 2023.
23. Z. Zhao, W. Lee, and D. Hsu. Llms as commonsense knowledge for large-scale task planning. *ArXiv*, abs/2305.14078, 2023.