Modeling attacker behavior in Cyber-Physical-Systems

Juan Betancourt Osorio
Dept. Of Industrial Engineering
Universidad de los Andes
Bogotá, Colombia
jm.betancourt@uniandes.edu.co

Germán Pardo González
Dept. Of Industrial Engineering
Universidad de los Andes
Bogotá, Colombia
gr.pardo@unianeds.edu.co

Samuel Rodriguez Gonzalez
School of Industrial and Systems Egineering
University of Oklahoma
Norman, OK, US
s.rodriguez@ou.edu

Daniel Cuellar
Dept. Of Industrial Engineering
Universidad de los Andes
Bogotá, Colombia
dh.cuellar@uniandes.edu.co

Camilo Gomez
Dept. Of Industrial Engineering
Universidad de los Andes
Bogotá, Colombia
gomez.ch@uniandes.edu.co

Francesco Mariotti
Dip. Sistemi e Informatica
University of Florence
Florence, Italy
francesco.mariotti@unifi.it

Leonardo Montecchi
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
leonardo.montecchi@ntnu.no

Paolo Lollini
Dip. Sistemi e Informatica
University of Florence
Florence, Italy
paolo.lollini@unifi.it

***Abstract*— Societies are increasingly dependent on Cyber Physical Systems (CPSs), which are exposed to natural and human-made attacks. Attacks on CPSs can result in security breaches and behaviors that may impose harm on their environments. Understanding attack mechanisms is crucial to preventing losses or damage to people, assets or information. We develop a computational environment based on the ADVISE formalism to model attack paths on CPSs, using a generalized stochastic optimization framework that allows to implement attacker agents based on different techniques, including approximate dynamic programming, reinforcement learning, or stochastic programming among others. We test the proposed environment by simulating attacks on a SCADA system previously addressed in the literature, demonstrating satisfactory convergence for a Q-learning algorithm, which allows to identify the attack steps that most frequently lead to successful attacks. The proposed approach allows to conceive models with interacting attacker and defender agents, which is left as the main goal of future work.**

Keywords— attacker-defender, cyber-physical systems, reinforcement learning, stochastic optimization.

I. Introduction

Cyber-Physical-Systems (CPSs) are an instance of complex engineering systems involving interrelated physical assets, technological intricacy (particularly regarding their cyber nature), and humans-in-the-loop. Typically, CPSs are involved in providing societal services such as the automation of processes in transportation or infrastructure systems, among others. Moreover, CPSs are exposed to harm due to natural or man-made (intentional) hazards, but also can impose harm on their embedding environments (including people) as a result of inherent or induced errors. This motivates the need to detect and prevent disruptions in their operation, which may affect the services they provide, the security and privacy of sensible information, and the integrity of people and the environment.

Modeling the complex dynamics of technological systems and human decisions is a challenge for which no single technique or approach is sufficient; physical, logical, and organizational aspects of the problem demand specific approaches and modeling techniques. The Stochastic Optimization framework proposed by Powell [1] offers a general scheme (based on the logic of Markov Decision Processes -MDPs) to articulate sequential decision problems in which the dynamics of a system evolves because of the actions made on the system and the occurrence of external phenomena over time. Different forms of simulation and optimization can be integrated, along with probabilistic and statistical models, to describe the dynamics and decisions of complex systems. This general modeling approach has proven useful to solve complex problems in robotics, autonomous driving, and games, thanks to the adoption of prescriptive techniques such as Approximate Dynamic Programming (ADP) and Reinforcement Learning (RL).

We explore the use of Powell's framework to model an attacker decision process based on their access to a CPSs, taking advantage of the MDP-like nature of the problem, as currently implemented in the ADVISE framework [2]. Our objective is to implement a computational environment to model an attacker's decisions on a CPSs, which can be compared against ADVISE, to later be extended to an attacker-defender problem.

We use the SCADA system analyzed in [2] as an instantiation of the proposed approach. We model the SCADA system as an

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

MDP based on the HTML description of the ADVISE model. The MDP modeling is developed in the form of a computational environment compatible with the OpenAI-Gym standard [3], thus, allowing the use of different solution techniques (e.g., rule-of-thumb policies, RL, ADP). We implement a value iteration algorithm that provides optimal attack paths over the MDP, allowing us to evaluate the convergence of other policies. We implement and evaluate the performance of a Q-learning algorithm as a promising alternative approach to model attacker agents. While further testing is required to suit realistic applications, the implemented algorithms prove the capability to replicate ADVISE's main features, with possibility of further insight based on the access to the complete MDP and setting the basis to develop attacker-defender simulation for CPSs.

II. Related work

The Supervisory Control and Data Acquisition (SCADA) systems are widely used in sectors such as power transmission, telecommunications and manufacturing control systems. The reliable and continuous operation of SCADA systems is vital because data acquisition and control are critically important. Due to its wide use, a number of standards and directives dealing with the cyber security of SCADA systems have emerged [4]. There have been different cyber-attacks on SCADA systems over time such as the targeted Stuxnet attack in 2010, which was attributed to the US and Israel and pretended to exploit the Siemens Programmable Logic Controllers in SCADA networks with the intention of destroying centrifuges used to process nuclear material [5].

There have been different approaches to model cyber security attacks over the years such as Model-Based Security Metrics using ADversary VIew Security Evaluation (ADVISE), in which they create an executable state-based security model of a system and an adversary. This method consists of three main phases: characterization of the system and its adversaries and the specification of the desired security metrics; generation of a dynamic model; and the execution of the model [2] [6]. It is also possible to use the ADVISE model to cover the attack patterns in the CAPEC database to enrich the combination of adversaries' profiles and attack paths [7].

One of the main approaches related to our work is the MDP model of competition for control of the power substations from [8]. They propose three different cyber-attack models to study the attackers' actions and its consequences: the adversary action model, the intrusion activity model and the strategy competitions between adversary and defender. They use interesting defensive tools like an anomaly detection system. In the end, they obtained the optimal policies of the attacker's and defender's worst scenario based on the MDP model.

M. Rasouli et al. [9] divided cybersecurity in two main categories: static and dynamic. In the first category, the agents (attacker and defender) receive no new information during the time horizon; these problems can be classified as resource allocation, where the defender and the attacker make a single decision regarding where to allocate their resources. Dynamic security problems permit the system and the agents to evolve over time, that is, the defender is continuously taking actions while observing the evolution of the system. [9] modeled the problem from a defender's point of view and with imperfect information and determined an optimal policy using dynamic programming.

Another interesting approach is in [10] where they used a deep reinforcement learning algorithm to maximize the robustness of autonomous vehicles' dynamics control to cyber-physical attacks. They study the autonomous vehicles' reaction to the attacker's actions, such as the injection of faulty data to the sensor readings or the manipulation of the inter-vehicle safe spacing in order to increase the risk of accidents on these types of vehicles. On the other hand, they also modeled autonomous vehicles acting as a defender where its goal is to ensure robustness to the attacker's action.

A. Nandi et al. [11] presented a bi-level mixed integer linear program model to find an optimal subset of arcs (interdiction plan) based on an attack graph representing the cyber-physical system. The attack graph of a system contains all the possible paths that an attacker can use to penetrate the system in diverse ways. The final goal is to minimize the loss due to security breaches given by the attack graph and the exact algorithm proposed can solve relatively large instances of the problem.

III. Methodology

A. *Problem rationale*

In order to model attacker behaviors in CPS under Powell's MDP-based framework for stochastic optimization, we rely on the ADVISE formalism to define the rules and dynamics of the "game" to be played by the attacker (and, at a later stage, the defender). The formalism is based on sets of: attack steps the attacker may execute at a cost, with certain probability of success and certain probability of detection; knowledges and skills with which an attacker must comply to achieve different attack steps; accesses that situate the attacker in a position within the system where certain attack steps are reachable; and goals that the attacker may obtain, with an associated reward based on the completion of specific attack steps.

Given an initial set of access(es), knowledge(s) and skill(s) available to the attacker, we aim at determining the sequence of valid attack steps that maximizes the attackers objective function, which combines the cost of executing attack steps, the probability of being detected, and the reward obtained when attaining the goal.

B. *Modeling attacks in CPSs under the unified stochastic optimization framework*

The unified stochastic optimization framework consists of five elements: states (S), which include any known or belief related information about the current situation of the system of interest; actions (X), which contain the available decisions at the current state; information (W), which represents exogenous events (often uncertain, for which only distributions may be known) that are revealed after an action is executed; transition function, which determines the problem's logic or dynamics, i.e., how the system or process evolves from being in one state to another as a result of the execution of a certain action and the realization of exogenous events at that particular original state. In the following, we define the five elements of the framework for the case of modeling attacks in CPSs following the ADVISE formalism.

State: The state variable, all the relevant information for any algorithm/agent, can be decomposed on three main components. For this application we found that two of the three categories are enough:

- Physical State: The agent has available information of which of the access, knowledge, skills and goals that it has achieved. This was encoded on a binary vector which for each component, indicates if the attacker has this component available. In an example with three accesses and two knowledges, skills and goals, the state would be:

r1	r2	r3	k1	k2	s1	s2	g1	g2
1	0	0	1	0	1	0	1	0

- Other deterministic information: The agent also has available which goal they are pursuing, and which are the available actions on a given time-step.

Action: The actions on this MDP are the attacks to perform on the time-step. The agent can only perform one action per time-step and cannot repeat actions it has already successfully executed. As an example, with the action space $X_t = \{a_2, a_3, a_6\}$, the action on a given time-step could be:

$$x_t = a_6$$

Exogenous information: Professor Powell proposes an additional variable which contains the realizations of the stochastic phenomena of the system. This information is computed after the actions are performed. This means that the agent has no knowledge of the value and cannot act once the variable is known. For this problem, the stochasticity is given on the success or failure of the attack performed. That is, each attack has an intrinsic probability of success. For the given state and the chosen action, the exogenous information could be

$$w_t = Success$$

Transition Function: The transition function models the dynamics of the system on the decision process. When the agent is in a given state, takes a given action and the exogenous information is computed, there is a transition to a new state. This function is.

$$S_M^X(S_t, A_t, W_{t+1}) \rightarrow S_{t+1}$$

Cost Function: As the transition function, the cost function takes a given state, an action and stochastic realizations and returns the cost/reward product of the transition. For this problem, the reward is composed of the following:

- Action cost: Each action has a defined cost; it is charged every time the action is performed (even in unsuccessful attempts).
- Goal payoff: When the active goal is reached and the episode terminates a defined reward is given to the agent.
- Termination penalization: When an episode is terminated because the maximum number of time-steps is reached, a large cost is given as a penalization to the agent.

$$C_{t+1}(S_t, A_t, W_{t+1}) \rightarrow c_t = Goal\ payoff - Action\ cost - Te$$

Where Te stands for the termination penalization

C. *Implementation of the computational environment*

We modeled the system under the adopted framework: Let S_t be the state variable of the system, x_t are all the decision variables, $X_t(S_t)$ is the set of the available actions given the state S_t, the exogenous information (information available just after we make a decision) is W_{t+1}, the transition function is $S^M(S_t, x_t, W_{t+1})$ and finally the objective function would be:

$$\max_{\pi} \mathbb{E}\left[\sum C\left(S_t, X^{\pi}(S_t)\right) | s_0\right]$$

where $C\left(S_t, X^{\pi}(S_t)\right)$ is the cost or reward function given by the state S_t and the policy $X(S_t)$.

For our system consider the representation of a CPS presented on Fig 1:

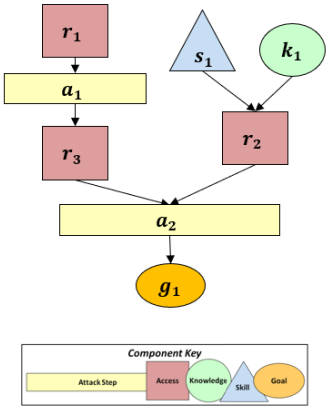

Fig 1. Example of CPS system

In this example the state variable is defined as:

- $S_t = \{(r_1, r_2, r_3), (k_1), (s_1), (g_1)\}$ where every variable is binary and takes the value of 1 if the access, knowledge, skill, or goal is available. The r represents the accesses, k stands for knowledge, s for skills and g for the goals achieved.
- $X_t = \{a_1, a_2\}$ are the available actions depending on the current system state S_t.
- W_t is the outcome of taking action X_t, that is, whether the action is successful or not.
- $S_{t+1} = S_X^M(S_t, X_t, W_t)$ represents the transition function. Where, depending on the outcome of the action, the system remains in the same state or transitions to a new state.
- $R_t(S_t, W_t, S_{t+1})$ is the cost or reward function that considers the cost of performing action X_t and if the attacker reached a goal or not.

The challenges in this problem are the uncertainty modelling and the optimal policy design.

D. Solving the decision problem for the CPS attacker

First, we describe the value iteration algorithm, which provides convergence guarantees (at the expense of potentially long computational times). Then, we describe the Q-Learning algorithm as a first attempt to train a reinforcement learning based attacker agent within our computational environment.

1) Value iteration

The value iteration algorithm is one of the most famous algorithms in dynamic programming and it is used to solve a broad variety of sequential decision problems. It resembles backwards induction but instead of using the epochs from T to 0 it uses an iteration counter n. Therefore, it is used for infinite horizon problems and the convergence is guaranteed under a specified criterion. One of the issues of value iteration is that its applicability can be limited because of the so-called curse of dimensionality, that is, the computational time needed to solve problems with a large space of states can become ridiculously long. Nevertheless, in our example case the computational time is not a problem. Value iteration is based on the computation of the following expression:

$$v^n(s) = \max_a \left\{ C(s,a) + \gamma \sum_{s \in s'} P(s'|s,a)\, v^{n-1}(s') \right\}$$

Where $C(s,a)$ is the cost of taking action a given that the system is in the state s, γ is the discount factor that reflects how valuable is the future reward and $P(s'|s,a)$ represents the probability of transitioning to state s' given that the system is in the state s and took the action a.

The convergence of the algorithm occurs when:

$$|v^n - v^{n-1}| < \delta$$

Where δ is the criterion predetermined for convergence. When convergence is reached, the optimal policy is obtained by computing:

$$\pi(s) = \operatorname*{argmax}_a \left\{ C(s,a) + \gamma \sum_{s \in s'} P(s'|s,a)\, v^{n-1}(s') \right\}$$

Where $\pi(s)$ represents the optimal policy that maps an action given a state. [1] [12]

2) Q-learning implementation to experiment with the agent construct

Q-Learning is an off-policy Temporal Difference Reinforcement Learning algorithm developed in the early 90's in order to solve Markov Decision Processes (MDPs) and to find their optimal or near optimal action-policies. The idea behind Q-Learning is to estimate an optimal action-value function for each state, known as the Q-function or Q-table, which consists of the expected returns or quality (hence the "Q" in Q-Learning) of taking a determined action at a particular state. The Q value for every state-action pair converges asymptotically towards their optimal expected value under the assumption that all of these state-action pairs are continued to be explored given a sufficiently large amount of time. The exploration policy is commonly known as an epsilon-greedy policy: with probability ε, the action selected is going to be greedy with respect to the highest Q-value for the current state, and the action selected is randomly sampled with equal probability for all actions in the actions pool with probability $1 - \varepsilon$.

$$Q^{n+1}(s,a) \leftarrow (1-\alpha) Q^n(s,a) + \alpha [r(s,a) + \gamma \max_{a' \in A} Q^n(s',a')]$$

The above Q-Learning update rule represents the value of taking action a in state s at each step. The reward $r(s,a)$ could be a random variable following a specific probability distribution but, without loss of generality, the Q-Learning update rule can be interpreted as the expected value of $r(s,a)$. The transition dynamics from state s to s' are determined by a state transition function. Parameter $0 < \alpha < 1$ represents a learning rate or step-size, which is basically a weighted average of past rewards and the initial estimate for the Q-value. Additionally, $0 < \gamma < 1$ can be interpreted in two different ways: i) When γ is close to 1, a greater importance is given to future rewards, whereas a value of close to 0 gives more importance to immediate rewards. ii) It can be seen as a mathematical convenience to guarantee the convergence to the optimal Q-values when the underlying MDP consists of only 1 episode with infinite steps.

IV. Computational Experiments

As a proof of concept, two scenarios were tested on both the value iteration and Q-Learning algorithms using the developed computational environment. Both scenarios represent different initial states of the SCADA system under study (i.e., available accesses, knowledges, etc.). Algorithms were tested and averaged over a defined set of 50 episodes. The experiments were performed on Python 3.10.4 and on a computer with the following specifications: MacBook Pro 2018, CPU Quad-Core Intel Core i5 of 2.3 GHz and 16 GB RAM. The main hyper-parameters of the algorithms are:

Value iteration:

- Threshold δ: 1e-15

Q-Learning:

- Number of training episodes: 15000
- α: 0.05
- γ: 0.99
- Initial ε: 0.7

a) Experiment 1

The initial components available were: Access 1, Knowledge 1 and Skill 2. The episode terminates when the agent reaches Goal 5. The algorithms were given a maximum of 6 time-steps to reach the goal. Figure 2 shows the initial state for Experiment 1, whereas Figures 3-6 provide a summary of results, including, in order: the obtained attack path; the frequency that each attack step was part of a successful attack path; the evolution of the reward during training; and the number of successful attacks per episode during training. The overall stats are as follow:

Value iteration:

- Computation time: 13.11 s
- Average reward of the trained agent: -1.47
- Average success rate of the trained agent: 0.62

Q-Learning

- Training time: 8.05 s
- Average reward of the trained agent: -1.47
- Average success rate of the trained agent: 0.62

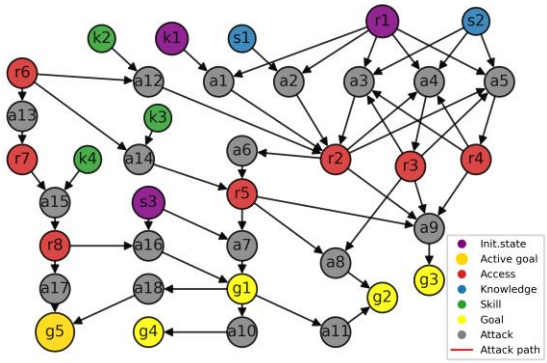

Fig 2. Initial state of the SCADA system for Experiment 1

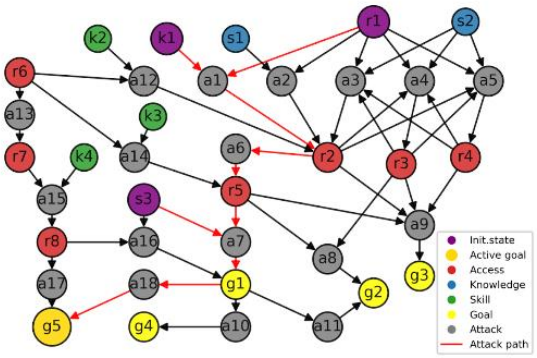

Fig 3. Optimal attack path for Experiment 1

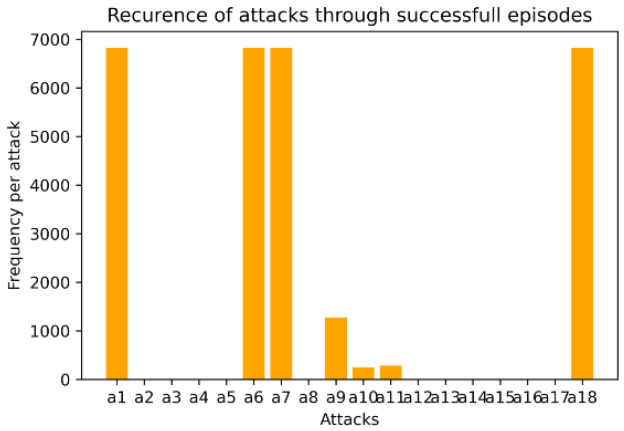

Fig 4. Histogram of the frequency for each attack in Experiment 1

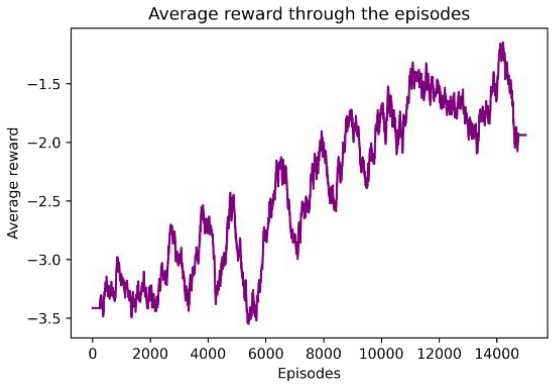

Fig 5. Average reward through the episodes in Experiment 1

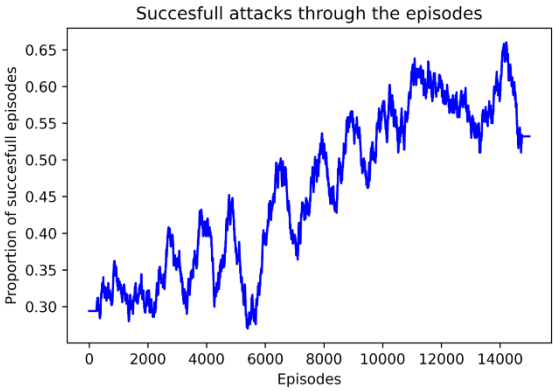

Fig 6. Proportion of successful episodes in Experiment 1

b) Experiment 2

The initial components available were: Access 6, Knowledge 4, and Skill 2 and 3. The episode terminates when the agent reaches Goal 2. The algorithms were given a maximum of 6 time-steps to reach the goal. Figure 7 shows the initial state for Experiment 1, whereas Figures 8-11 provide a summary of results, including, in order: the obtained attack path; the frequency that each attack step was part of a successful attack path; the evolution of the reward during training; and the number of successful attacks per episode during training. The overall stats are as follows:

Value iteration

- Computation time: 11.58 s
- Average reward of the trained agent: 0.38
- Average success rate of the trained agent: 0.88

Q-Learning

- Training time: 8,64 s
- Average reward of the trained agent: 0.38
- Average success rate of the trained agent: 0.88

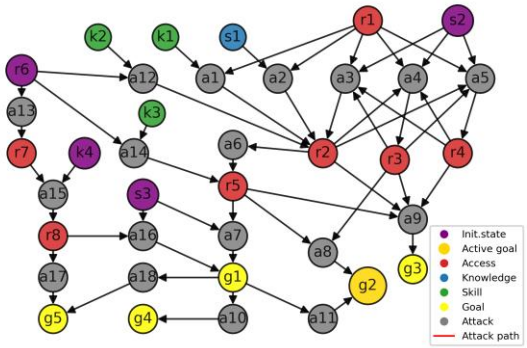

Fig 7. Initial state of the SCADA system for Experiment 2

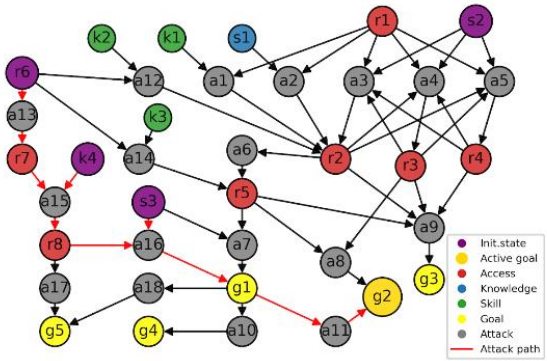

Fig 8. Optimal attack path for Experiment 2

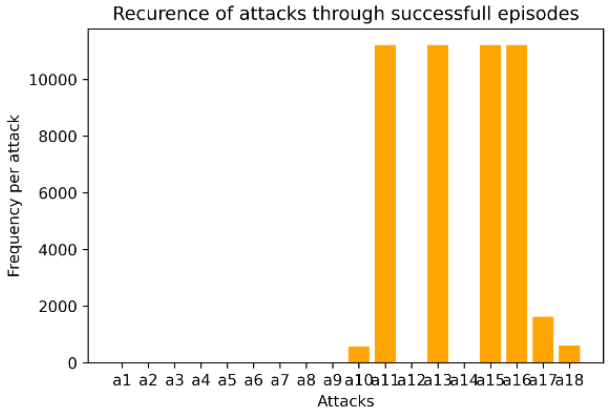

Fig 9. Histogram of the frequency for each attack in Experiment 2

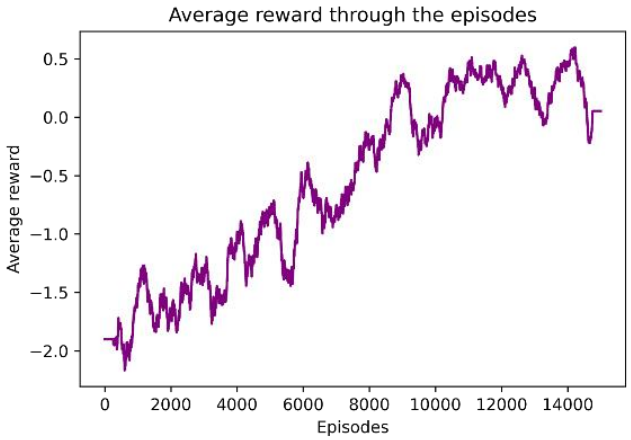

Fig 10. Average reward through the episodes in Experiment 2

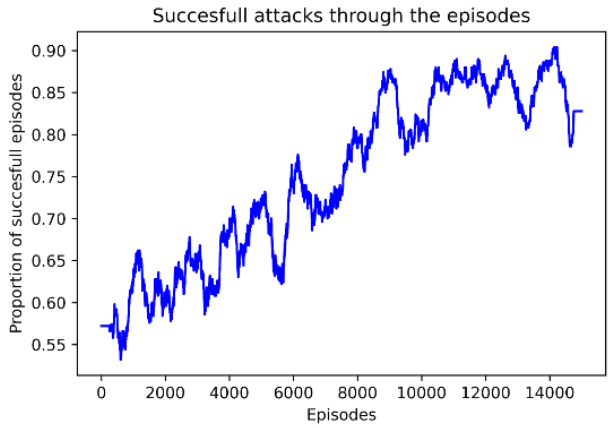

Fig 11. Proportion of successful episodes in Experiment 2

To analyze the average reward and average success rate of the Q-Learning algorithm there are a few important aspects to consider:

- The algorithm explores the state and action space throughout the whole training. The exploration rate is significatively higher at the beginning of the training period (0.7) and declines (down to 0.04) at the end of the training. This means, even with consolidated estimates of the Q-values, that the agent might still take random actions which can mean unsuccessful episodes.
- The number of maximum time-steps implies that even with an optimal policy, there will be unsuccessful episodes due to realizations of the actions' successes.

With this basis, the obtained results are very promising. On both experiments, the average reward and success graphs reveal a visible learning curve in which the agent starts the training having erratic performance. But, as the training advances, the average performance increases and stabilizes at the end of the training. This means, that the estimates the agent has calibrated allows it to take effective actions and reach the goal. Having the optimal policy assured by the value iteration algorithm, we can conclude that the Q-Learning agent learned the optimal policy for both experiments. In terms of computational performance, the Q-Learning training took around 35% less time than the Value Iteration computation. This represents a big contribution when working with more complex systems.

In each scenario, the frequency of the actions performed on successful episodes was recorded. With this information, it is possible to identify the critical attacks (and components) when an attacker is pursuing a specific goal. This can provide insightful information for any defender decision maker, it allows the profiling of the attacker and taking adequate countermeasures. The attack-path graphs provide a visual tool to recognize the sequence of actions that the attacker might take when exploiting vulnerabilities.

a) Definition of available defender actions with their associated cost and impact on the probability of attack success.
b) Definition of attacker variations (e.g., number of available attempts to attack; initial accesses/knowledges)
c) Execution of all game-settings defined in (a) and (b) under the implemented environment, and policy evaluation reports with visual analytics of players strategies and outcomes.

V. Conclusions

We proposed a modeling approach for attacker behaviors in CPS based on a unified framework for stochastic optimization and following the ADVISE formalism. The modeling approach follows the logic of Markov decision processes to generalize the process of sequential decisions under uncertainty and admits a variety of solution approaches, including approximate dynamic programming and reinforcement learning. This versatility enables the implementation of attacker agents based on different policies (algorithms) and eventually include interacting attacker and defender agents.

A computational environment was developed as a Python class designed to be compatible with OpenAI's gym standards [3]. A proof of concept was implemented for a SCADA system previously addressed in the literature, using two different scenarios of initial states and desired goals. Two algorithms were successfully implemented and evaluated under the policy evaluation scheme: Value Iteration and Q-Learning. After training and calibration, the policies can navigate the ADVISE graph efficiently and achieve the goal in over 70% of episodes. This is remarkable considering that the number of maximum time-steps was designed to be restrictive to few unsuccessful attacks. The most critical attack steps (i.e., those most frequently associated with successful attack paths) for each scenario were identified and serve as a basis for including defender actions.

Future work includes building an instance set with more complex systems to evaluate and improve the performance of the proposed environment and algorithms, as well as including a defender agent to implement preventive and reactive actions to reduce the impact of attacks.

VI. References

[1] W. Powell, Reinforcement Learning and Stochastic Optimization, NJ: John Wiley & Sons Inc., 2022.

[2] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders and C. Muehrcke, "Model-based Security Metrics using ADversary VIew Security Evaluation (ADVISE)," in *Eighth International Conference on Quantitative Evaluation of SysTems*, 2011.

[3] G. Brockman, V. Cheung, L. Petterson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[4] Cherdantseva, B. P. Yulia, A. Blyth, P. Eden, K. Jones, H. Soulsby and K. Stoddart, "A review of cyber security risk assessment," *Computers & Security*, vol. 56, pp. 1-27, 2015.

[5] A. Sood and R. J. Enbody, "Targeted Cyberattacks," Cyberwarfare, 2013.

[6] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe and W. H. Sanders, "Adversary-Driven State-Based System Security Evaluation," in *6th International Workshop on Security Measurements and Metrics*, New York, 2010.

[7] F. T. M. Mariotti, L. Montecchi and P. Lollini, "Extending a security ontology framework to model CAPEC attack paths and TAL adversary profiles," 2022.

[8] Y. Chen and J. L. C. Hong, "Modeling of Intrusion and Defense for Assessment of Cyber Security at Power Substations," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2541-2552, 2018.

[9] M. Rasouli and E. T. D. Miehling, "A Supervisory Control Approach to Dynamic Cyber-Security," in *Decision and Game Theory for Security*, 2014.

[10] A. Ferdowsi, U. Challita, W. Saad and N. B. Mandayam, "Robust Deep Reinforcement Learning for Security and Safety in Autonomous Vehicle Systems," in *21st International Conference on Intelligent Transportation Systems*, 2018.

[11] A. K. Nandi, H. R. Meda and S. Vadlamani, "Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender–attacker model," *Computers & Operations Research*, pp. 118-131, 2016.

[12] R. Sutton and A. Barto, "Reinforcement Learning: An Introdutcion," The MIT Press , 2015.