Deep Reinforcement Learning for Automated Cyber-Attack Path Prediction in Communication Networks

Franco Terranova, Abdelkader Lahmadi, Isabelle Chrisment

Deep Reinforcement Learning for Automated Cyber-Attack Path Prediction in Communication Networks

Franco Terranova[1], Abdelkader Lahmadi[1], Isabelle Chrisment[1]

[1]Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France

Background & Motivation

- Traditional methods of manual **attack path discovery** struggle to scale with the dynamic nature of potential threats and time-varying communication networks.
- **Deep Reinforcement Learning** (RL) [1] can be leveraged to realize an automated network security analysis.
- The RL agent will aim to learn an effective **attack policy** to exploit vulnerabilities and compromise the largest amount of nodes within the network. The **cyber-attack paths** learned can be used to assess and increase the situational awareness of the network security.
- This work aims to improve existing studies with a topology-independent neural network (NN) structure and a comprehensive evaluation of generalization, particularly across various topology sizes [2, 3].

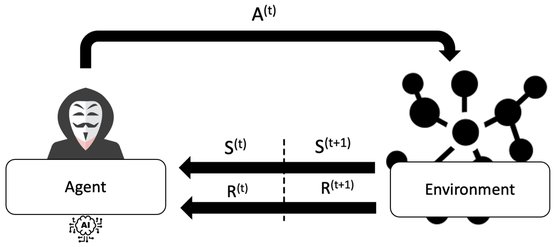


Figure 1. The agent receives a state or observation $S(t)$, takes action $A(t)$, receives reward $R(t)$, and transitions to a new state $S(t+1)$.

Methodology

Our contribution includes initial improvements to the existing attack simulation approaches, providing:

- An enhancement of the **Partially Observable Markov Decision Process (POMDP)** discarding assumptions about prior knowledge of the communication network structure.
- **Local nodes' observation and action spaces**: re-formulating the NN's input and output spaces focusing on pair of nodes, leading to a topology-independent NN's structure.

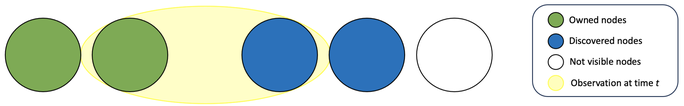


Figure 2. Representation of the POMDP with the distinction of owned, discovered, and not discovered nodes. The observation vector at a given moment t will be the concatenation of the feature vectors of the source and target nodes.

- The observation space O includes the partially visible features for the source and the target nodes of the attack.
- The action space A includes:
  - Local vulnerabilities
  - Remote vulnerabilities
  - Port connections
  - Source node selection
  - Target node selection
- The reward function R will represent the control gained by the agent as a consequence of the exploitation of a vulnerability.

$$R(o,a) = \sum_{i \in \text{owned}} \text{value}(i) + K_{\text{node discover}} \cdot \text{nodes discovered} + K_{\text{credential discover}} \cdot \text{credentials discovered} + K_{\text{success}} - cost_{\text{vulnerability}}$$

Simulation Environment

CyberBattleSim [4] has been used for generating the abstract network environment scenario **CyberBattleChain**.

- Start node as the entry point
- Variable-size chain of alternating Linux and Windows host with fixed vulnerabilities per OS
- Terminate node with a goal flag

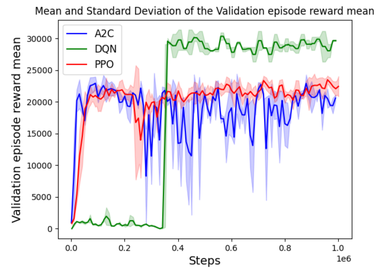
Figure 3. The representation of the CyberBattleChain simulation environment with the Windows-Linux chain.

The **experimental scenarios** has been generated with the following procedure:

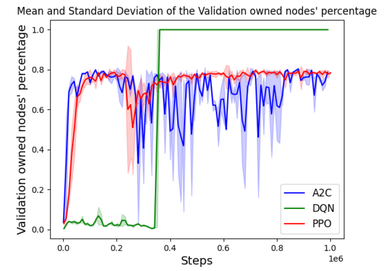
- 200 different chains with a number of nodes in the range [100, 300] divided in training set (60%), validation set (20%), and test set (20%) by size.
- Deep Q-Network (DQN) [5], Proximal Policy Optimization (PPO) [6], and Actor-Critic (A2C) [7] algorithms with default hyper-parameters [8].
- Multiple runs of 1,000,000 steps with episodes' cut-off set to 2 * optimal number of steps:

$$\text{optimal steps' number} = (3 * (\text{chain size} + 1)) \quad (1)$$

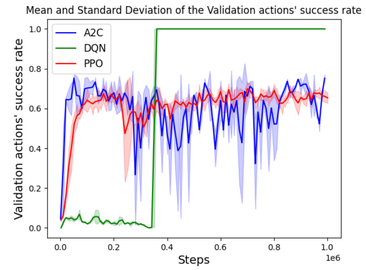
Results



(a) Average reward on the validation set during training.



(b) Owned nodes' percentage on the validation set during training.



(c) Success rate percentage of the actions performed on the validation set.

Algorithm	Average Owned Nodes (%)
Random Agent	0.02 ±0.01
A2C	0.75 ±0.02
PPO	0.78 ±0.07
DQN	1.00 ±0.00

(d) Average owned nodes' percentage on the test set.

Conclusions

- Our experiments of the value-based, policy-based, and actor-critic methods showed convergence results on a chain-based topology environment being able to **generalize among chain sizes**.
- The DQN method has converged to the **optimal policy** required to solve the deterministic environment, also on reserved sets of larger chains.
- Future work will aim to explore new environments and leverage Graph Neural Networks (GNNs) capabilities.

Acknowledgment

This work has been partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22-PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press Cambridge, 2018.

[2] Q. Li, M. Hu, H. Hao, M. Zhang, and Y. Li, "Innes: An intelligent network penetration testing model based on deep reinforcement learning," *Applied Intelligence*, vol. 53, pp. 1–18, Sep. 2023.

[3] X. Guo *et al.*, "Automated penetration testing with fine-grained control through deep reinforcement learning," *J. Commun. Inf. Netw.*, vol. 8, no. 3, pp. 212–220, Sep. 2023.

[4] Microsoft, "Cyberbattlesim: A cyber security research environment," https://github.com/microsoft/CyberBattleSim.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[8] A. Raffin *et al.*, "Stable baselines3: Reinforcement learning in python," https://github.com/DLR-RM/stable-baselines3, 2022.

franco.terranova@inria.fr