Offline RL+CKG: A hybrid AI model for cybersecurity tasks

Aritran Piplai[1], Anupam Joshi[1] and Tim Finin[1]

[1]*University of Maryland, Baltimore County, 1000 Hilltop Cir, Baltimore, MD 21250, USA*

Abstract

AI models for cybersecurity have to detect and defend against constantly evolving cyber threats. Much effort is spent building defenses for **zero days** and unseen variants of known cyber-attacks. Current AI models for cybersecurity struggle with these yet unseen threats due to the constantly evolving nature of threat vectors, vulnerabilities, and exploits. This paper shows that cybersecurity AI models will be improved and more general if we include semi-structured representations of background knowledge. This could include information about the software and systems, as well as information obtained from observing the behavior of malware samples captured and detonated in honeypots. We describe how we can transfer this knowledge into forms that the RL models can directly use for decision-making purposes.

Keywords

Offline reinforcement learning, Cybersecurity knowledge graphs, Conservative Q-Learning, Hybrid AI

1. Introduction

AI is playing an increasingly important role in cybersecurity – that of automating the detection and mitigation of cyber threats at scale. Traditional rule-based systems capturing explicit knowledge elicited from analysts do not work well against modern cyber threats, especially over time [1]. This is because humans do not observe enough variants of cyber threats to create generalizable rules. Moreover, cyber threats are constantly evolving as adversaries develop new tactics, techniques, and procedures (TTPs) [2]. The observed data used for creating these rules may not be as useful over time because attackers have created variants of their attacks that are quite different in their behavioral manifestation than what was used to create these rules. Even approaches using inductive machine learning approaches that try to capture tacit knowledge are limited and not generalizable since the dataset used in training may contain variants that are quite different from the variants of cyber-threats observed during test time or real-time evaluations [3].

Reinforcement learning (RL) can prove to be useful for constantly changing domains like cybersecurity, since it internally creates a simulation map of different variants that may spin off from the data presented for training. This helps deal with the problem of new attacks used by adversaries. However, a significant critique of online RL models is that they need to observe a

In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE 2023), Hyatt Regency, San Francisco Airport, California, USA, March 27-29, 2023.

*Corresponding author.

apiplai1@umbc.edu (A. Piplai); joshi@umbc.edu (A. Joshi); finin@umbc.edu (T. Finin)

vast number of state-action pairs to generate a generalizable policy, making it impractical for many domains [4], including cybersecurity. However, offline RL techniques [5] have claimed to be just as good with a limited and fixed number of observations of state-action pairs [6]. One of the contributions in this paper is we observe how useful offline RL is for cyber-threat detection. Although offline RL models may produce a policy for cyber-threat detection, it is still limited by partially observed possibilities of cyber threats. For example, if we look at data that describes the behavior of a particular malware in a specific environment, the behavior is limited by the particulars of that environment. The processes created by the malware or the values of system parameters may be different in a different environment or at a different time. It is impossible to simulate the possibilities of malware behavior in all environments. This is where explicit external knowledge comes into play. The external knowledge about a domain that an expert has can provide the RL algorithms with key information during the learning process. Our other contribution in this paper is to show how to create knowledge-guided Offline RL algorithms that can leverage the existing body of work that captures explicit domain knowledge about cybersecurity as knowledge graphs [7, 8, 9].

2. Related Work

Machine learning and reinforcement learning have been used in cybersecurity increasingly in recent years. Effective applications of ML have included intrusion detection [10] and attack detection in cyber-physical systems [11, 12]. To tackle the shortcomings of supervised ML in cybersecurity, RL is also being used [13, 14, 15]. There is also a body of research focused on developing cybersecurity knowledge graphs (CKGs) [7, 8, 9] that represent semi-structured knowledge about cybersecurity data supported by logic axioms. The combination of logic and reinforcement learning has been gaining popularity [16, 17], and researchers have begun to experiment with using CKGs for RL for simulated cyber warfare scenarios [18].

There has also been a body of research focused on knowledge graph construction and modeling. Model-based construction has led to the generation of enterprise knowledge graphs [19, 20]. Cybersecurity concepts and data can be derived from datasets in STIX, an industry standard for exchanging threat intelligence [21]. These concepts have led to CKG models such as UCO [7]. In our prior work, we have slightly modified the ontology in other CKGs [8, 22]. Data to populate CKGs can also be extracted by natural language understanding systems applied to cybersecurity-related reports. In this paper, we use the modified UCO ontology that we have previously published.

Offline RL has become more popular because it is able to achieve good performance with static data. Conservative Q-Learning (CQL) is the state-of-the-art for offline RL algorithms, but researchers are just beginning to explore improving CQL with external knowledge. Extracting knowledge from the current data for CQL was recently demonstrated in [23], but this does not serve our purpose for cybersecurity. In our paper, we describe how external knowledge sources can be used in offline RL algorithms for cybersecurity tasks.

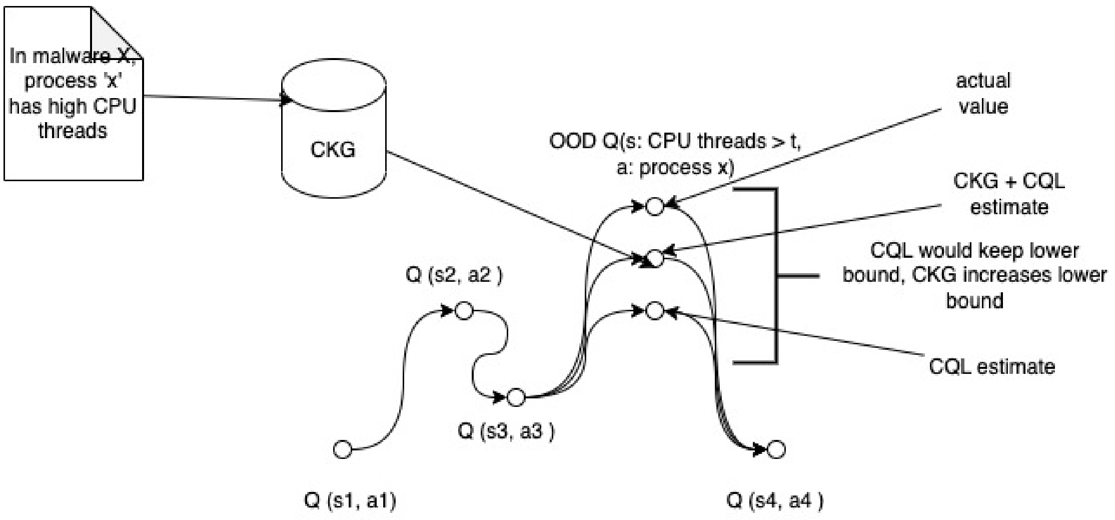

In malware X, process 'x' has high CPU threads
CKG
actual value
OOD Q(s: CPU threads > t, a: process x)
CKG + CQL estimate
CQL would keep lower bound, CKG increases lower bound
Q (s2, a2)
CQL estimate
Q (s3, a3)
Q (s1, a1)
Q (s4, a4)

Figure 1: Figure describing the overview of our approach. The input to the CKG is a piece of intelligence describing some characteristics of a malware 'X'. These characteristics are translated to state-action pairs. If these state-action pairs correspond to out-of-distribution (OOD) state-action pairs for the CQL algorithm, we can use it to re-estimate the Q-value for the OOD state-action pair.

3. Method

Our research approach uses RL to detect malware. We use a malware behavior dataset containing information on how different system parameters change values when a malware instance from a known malware family is detonated. We also incorporate MOTIF [24], a curated dataset of malware samples and their family names. Our RL algorithm creates a policy to detect a specific malware in a system for which we have data. We can then use an external knowledge source that provides information about other malware in the same family, or possibly a similar malware family. The RL algorithm can then use this knowledge to make changes to the policy.

CKGs contain information about recent developments in cyber threats. In our prior work, we demonstrated how to generate CKGs from textual threat intelligence and other sources, and used them in downstream tasks [18]. This is also a part of ongoing research that focuses on enriching these knowledge graphs with additional information from Wikidata [25, 26]. In our problem, we use offline RL because observing how system parameters change in real-time is cumbersome and impractical. Offline RL algorithms typically have the problem of overestimating some unseen state-action pairs. To address this, researchers have come up with approaches such as IQL [27] and CQL [4]. CQL downgrades updates on unseen state-action pairs. However, CQL has been argued to be too conservative in updating unseen data. In our domain, we need to update unseen data because the unseen variants of malware data will be out-of-distribution (OOD).

For example, the system parameter observations are made at specific time intervals and not all changes are caught when a malware is active in the system. When a process, or action, is created not all state change observations are available to us. As seen in Figure 1, a large number of CPU threads may be created when a particular process is active. This observation, if available to us, can be used for modeling in both online and offline RL algorithms. However, if it is not available to us, offline RL algorithms such as CQL will relatively downgrade this OOD state-action pair. If we have an external knowledge source that creates its own state-action pair distribution K, this can be used to relatively upgrade the OOD state-action pair (high number of CPU threads and process 'x').

Thus, we can create less conservative updates on unseen data if we have evidence of it in the CKG's. Thus we create a β term in the equation if the state-action evidence can be retrieved from the CKG. In the following equation, D is the data distribution, and K is the distribution from the knowledge source.

$$\hat{Q}^{\pi} = \arg\min_{Q} \max_{\mu} \alpha\beta E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})] - \beta E_{(\mathbf{s},\mathbf{a})\sim K}[Q(\mathbf{s},\mathbf{a})]$$

$$+E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + E_{\pi}\left[Q\left(\mathbf{s}',\mathbf{a}'\right)\right]\right)\right)^{2}\right]$$

The equation above minimizes the Bellman error values for all state-action pairs. α is the hyperparameter that tackles the increase of the Q-value estimates of data that is not OOD. β is another hyperparameter that increases the Q-value estimates of the OOD data if the CKG's information supports them.

We use malware behavior data that is achieved after detonating a malware sample in a controlled environment. The state space is defined by features of the systems' parameters. In our experiments, we calculate the temporal difference between system parameters and normalize them with min-max normalization. The actions are individual processes that cause these state changes. We propose to find out malicious processes with the help of RL. We have labeled malicious process names for malware samples from different families, and we are going to use information from our CKG for these malware families to help RL algorithms. For example, in Figure 1, we see an OSINT source with some information about CPU threads of a process 'x' for a specific malware or a malware belonging to a particular family. As the RL algorithm goes over the state-action pairs in our system parameter data while constructing Q-tables, it may come across a state-action pair that was not encountered in the actual system parameter data. This may be possible due to two reasons. The first reason is the malware sample in this family may not be adhering to the information provided by the CKG about the same family. Secondly, because of the periodic nature of data collection of the system parameter data, this particular state-action pair may not be present in the collected data. This is where the information from CKG comes in, as it lifts the Q-values for those state-action pairs.

4. Experiments and Results

We experimented with 25 malware samples belonging to four malware families. Our first experiment used RL algorithms that work on all types of samples and all malware families, and our second applied family-specific RL algorithms. Each malware sample has time steps (or a number of states) that range from 2163 to 3750. In some cases, we trim the number of states to

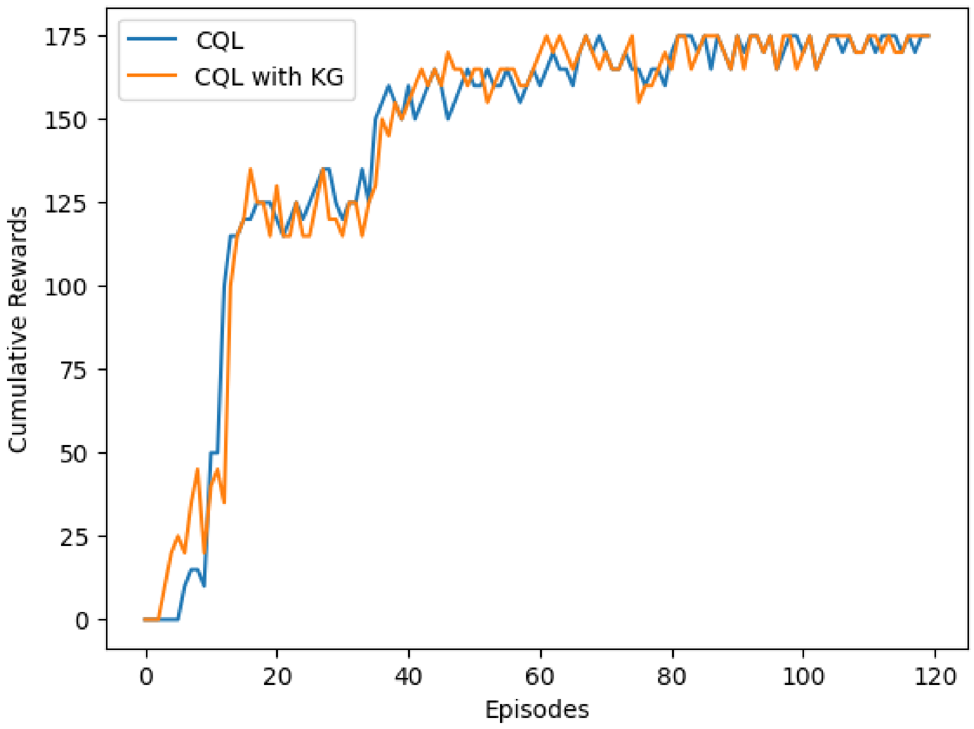


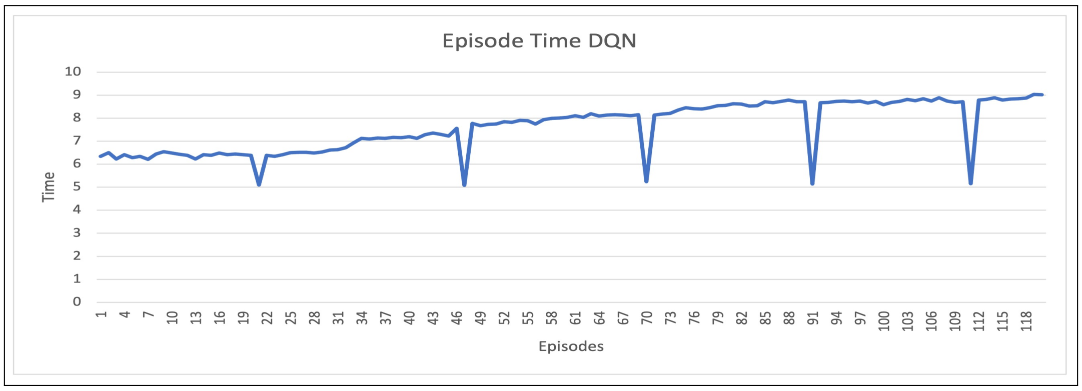
Figure 2: Comparison of cumulative rewards between CQL and CQL with KG guidance

Table 1
Hits@10 calculated for Q-values of malicious processes

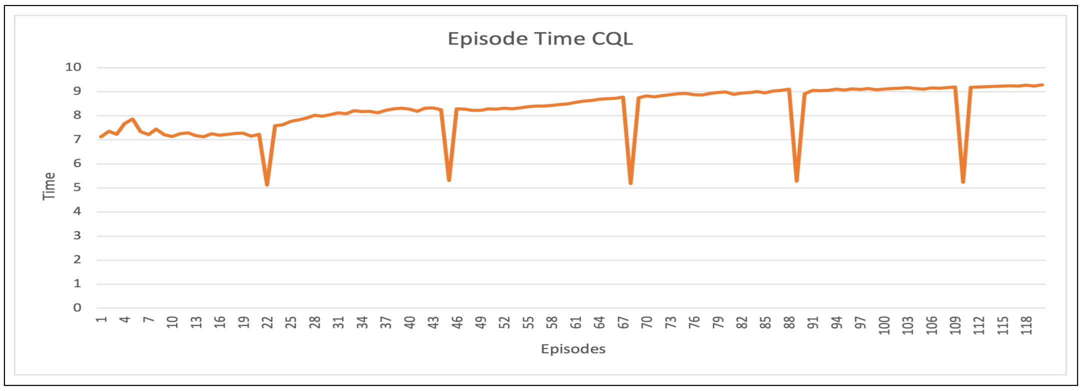
Malware Family	CQL	CQL+CKG (β=0.05)	CQL+CKG (β=0.1)
All	0.74	0.72	0.72
Acidbox	0.68	0.73	0.71
RogueRobin	0.87	0.89	0.91
vhd	0.93	0.91	0.91
hiddenwasp	0.86	0.93	0.93

3750 due to a high number of redundant states present in the data collected. The action space for the experiment with all malware families is 493, and for individual malware families ranged from 102 to 193. We used user-defined reward functions for these exercises formed by our own understanding of the malware samples.

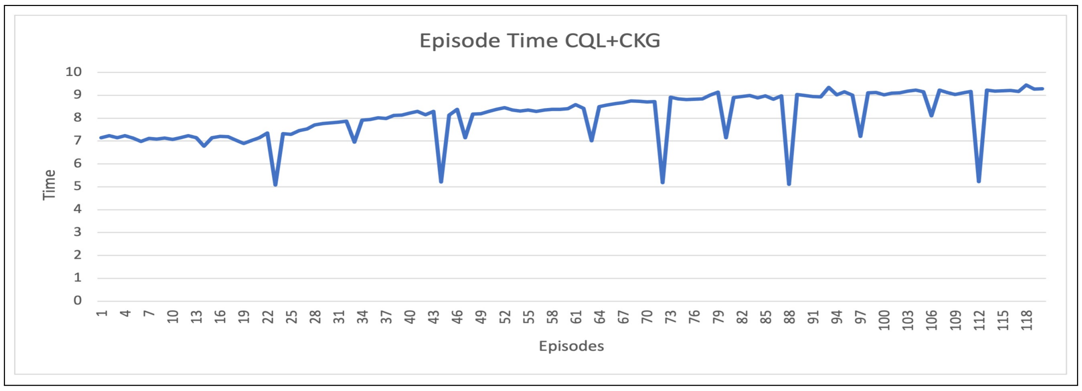
In order to check how CKG influences the performance of CQL, we take two approaches. We plot the cumulative rewards of CQL and CQL with CKG guidance as shown in Figure 2. However, it is difficult to quantifiably argue that the difference in cumulative rewards actually



(a) Episode Time for DQN with Q-learning as baseline



(b) Episode Time for CQL



(c) Episode Time for CQL with knowledge guidance

Figure 3: Comparison of time required to complete each episode for two experiments

results in better detection capabilities since the reward functions are hand-crafted based on our own knowledge of the malware. One algorithm having a higher cumulative reward may be due to the fact that the reward functions are crafted more favorably. One way to check that the Q-values are representative of the actual quality of the state-action pairs is to see how the associated Q-values are for state-action pairs involving malicious processes. As we mentioned before, we labeled some of the malicious processes in the data collected. We check the top 10% of the Q-values for all state-action pairs in our dataset and observe what percentage of them involve malicious processes.

Ideally, the malicious processes should have the highest Q-values because our RL training is designed in such a way. In Table 1, we observe that the Q-values for RL models are trained for all families and RL models trained for individual families. The CQL model trained for all models together does not benefit from CKG information. This may be because of the generic nature of the CKG's information for all models. A particular process name with a higher Q-value may be relevant for a particular family, but it may mistakenly raise the Q-values for a state-action pair for a different family. However, if we look at specific malware families, we see three of the four malware families observe some lift in scores. We also note how the variable β affects the scores. This variable controls by what factor we raise the scores of the Q-values for OOD state-action pairs that are suggested by the CKG. Raising the value of β may sometimes decrease the score because, in some cases, it may artificially inflate the values of some state-action pairs. Further experimentation is needed to calculate the appropriate values of this hyper-parameter.

We also observe the time required to complete each episode. In Figure 3, we see the episode time for DQN as baseline, CQL, and CQL+CKG with $\beta = 0.5$. The periodic dips that we see are due to exploration-exploitation. As we see, the average time to complete each episode is higher for CQL than for DQN. This is mostly due to higher computation costs for each iteration. When we compare CQL with CQL with CKG, we observe that there are more dips in the episode times. We stop each episode when the cumulative rewards reach a threshold, indicating that the malicious process has been found or when all the states are covered. In some cases, CKG might have helped to raise the cumulative rewards and made it reach the threshold faster than vanilla CQL. The average episode time, even though the computation is more complex in CQL+CKG, is lower because of these dips. However, in some cases, CKG also contributes to the distributional shift problem of RL training leading to higher episode times.

5. Conclusion and Future Work

In this paper, we study the effects of knowledge guidance on CQL for cybersecurity tasks. CQL, like other offline RL algorithms, was introduced to tackle the problem of distributional shift that we observe in online RL training. However, since CQL is too conservative, we see how knowledge guidance can make the training faster or make the model better at the specific task. We see that the training time is lower owing to faster episode end times with helpful knowledge guidance. While looking at the performance of the models, we see that in specific malware families, knowledge about that type of malware may have helped. However, generalizing this knowledge at CQL training time across all families causes a small dip in performance. In malware families, where malware samples do follow a certain trend, the Q-values for the associated malicious processes have indeed increased, resulting in a more effective model.

Our preliminary results show that for some malware families, such as *hiddenwasp*, we observe a lift of 7% in detection capabilities. The ability of KG guidance in increasing the quality of Q-learning depends on how representative the knowledge is and how easy it is to transfer the knowledge to RL parameters. Variance between samples of malware families also play an important role. Some of the problems that we faced in this can be mitigated by using a wider range of knowledge. We are currently enriching our cyber knowledge bases with the help of Wikidata [25, 26]. It will be interesting to see how more knowledge about malware families affects the generalizability of offline RL models trained with knowledge guidance. In this paper, we talked about specific cybersecurity tasks, such as malicious process detection. We would like to extend this to other cybersecurity tasks like cyber attacks and defenses.

Acknowledgments

This work was supported by the National Security Agency and National Science Foundation award 2114892. We thank researchers from the University of Texas at San Antonio for their data collection infrastructure and for sharing collected malware behavior data.

References

[1] Cybersecurity News, Why signature-based detection struggles to keep up with the new attack landscape?, http://cybersecuritynews.com/signature-based-detection/, 2022.

[2] C. Johnson, L. Badger, D. Waltermire, J. Snyder, C. Skorupka, et al., Guide to cyber threat information sharing, NIST special publication 800 (2016).

[3] Forbes, Comparing legacy rules-based cyber-security platforms and AI-based platforms, http://forbes.com/sites/forbestechcouncil/2022/02/14/comparing-legacy-rules-based-cybersecurity-platforms-and-ai-based-platforms, 2022.

[4] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative q-learning for offline reinforcement learning, Advances in Neural Information Processing Systems 33 (2020) 1179–1191.

[5] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: Tutorial, review, and perspectives on open problems, arXiv preprint arXiv:2005.01643 (2020).

[6] A. Nair, A. Gupta, M. Dalal, S. Levine, Awac: Accelerating online reinforcement learning with offline datasets, arXiv preprint arXiv:2006.09359 (2020).

[7] Z. Syed, A. Padia, T. Finin, L. Mathews, A. Joshi, UCO: A unified cybersecurity ontology, in: Artificial Intelligence for Cyber Security: Report WS-16-03, AAAI, 2016, pp. 195–202.

[8] A. Piplai, S. Mittal, A. Joshi, T. Finin, J. Holt, R. Zak, Creating cybersecurity knowledge graphs from malware after action reports, IEEE Access 8 (2020) 211691–211703.

[9] N. Rastogi, S. Dutta, M. J. Zaki, A. Gittens, C. Aggarwal, Malont: An ontology for malware threat intelligence, in: International Workshop on Deployable Machine Learning for Security Defense, Springer, 2020, pp. 28–44.

[10] K. A. da Costa, J. P. Papa, C. O. Lisboa, R. Munoz, V. H. C. de Albuquerque, Internet of things: A survey on machine learning-based intrusion detection approaches, Computer Networks 151 (2019) 147–157.

[11] D. Ucci, L. Aniello, R. Baldoni, Survey of machine learning techniques for malware analysis, Computers & Security 81 (2019) 123–147.

[12] D. Ding, Q.-L. Han, Y. Xiang, X. Ge, X.-M. Zhang, A survey on security control and attack detection for industrial cyber-physical systems, Neurocomputing 275 (2018) 1674–1683.

[13] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, M. Guizani, Security in mobile edge caching with reinforcement learning, IEEE Wireless Communications 25 (2018) 116–122.

[14] X. Xu, Y. Luo, A kernel-based reinforcement learning approach to dynamic behavior modeling of intrusion detection, in: International Symposium on Neural Networks, Springer, 2007, pp. 455–464.

[15] T. T. Nguyen, V. J. Reddi, Deep reinforcement learning for cyber security, IEEE Transactions on Neural Networks and Learning Systems (2021).

[16] G. Anderson, A. Verma, I. Dillig, S. Chaudhuri, Neurosymbolic reinforcement learning with formally verified exploration, Advances in neural information processing systems 33 (2020) 6172–6183.

[17] D. Kimura, M. Ono, S. Chaudhury, R. Kohita, A. Wachi, D. J. Agravante, M. Tatsubori, A. Munawar, A. Gray, Neuro-symbolic reinforcement learning with first-order logic, arXiv preprint arXiv:2110.10963 (2021).

[18] A. Piplai, M. Anoruo, K. Fasaye, A. Joshi, T. Finin, A. Ridley, Knowledge guided two-player reinforcement learning for cyber attacks and defenses, in: International Conference on Machine Learning and Applications, 2022.

[19] P.-L. Glaser, S. J. Ali, E. Sallinger, D. Bork, Model-based construction of enterprise architecture knowledge graphs, in: 26th Int. Conf. on Enterprise Design, Operations, and Computing, Springer, 2022, pp. 57–73.

[20] M. Smajevic, D. Bork, From conceptual models to knowledge graphs: a generic model transformation platform, in: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), IEEE, 2021, pp. 610–614.

[21] B. Jordan, R. Piazza, T. Darley, Stix version 2.1, oasis standard, OASIS Committee Specification (2021). Http://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html.

[22] A. Piplai, S. Mittal, M. Abdelsalam, M. Gupta, A. Joshi, T. Finin, Knowledge enrichment by fusing representations for malware threat intelligence and behavior, in: International Conference on Intelligence and Security Informatics (ISI), IEEE, 2020, pp. 1–6.

[23] J. Lyu, X. Ma, X. Li, Z. Lu, Mildly conservative q-learning for offline reinforcement learning, in: Advances in Neural Information Processing Systems, 2022.

[24] R. J. Joyce, D. Amlani, C. Nicholas, E. Raff, Motif: A large malware reference dataset with ground truth family labels, arXiv preprint arXiv:2111.15031 (2021).

[25] C. Hanks, M. Maiden, P. Ranade, T. Finin, A. Joshi, Recognizing and Extracting Cybersecurity Entities from Text, in: Workshop on Machine Learning for Cybersecurity, International Conference on Machine Learning, PLMR, 2022.

[26] V. Mulwad, T. Finin, V. S. Kumar, J. W. Williams, S. Dixit, , A. Joshi, A Practical Entity Linking System for Tables in Scientific Literature, in: 3rd Workshop on Scientific Document Understanding, AAAI, 2023.

[27] I. Kostrikov, A. Nair, S. Levine, Offline reinforcement learning with implicit q-learning, arXiv preprint arXiv:2110.06169 (2021).