International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 6, December 2022, pp. 6768~6776
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i6.pp6768-6776

Optimizing cybersecurity incident response decisions using deep reinforcement learning

Hilala Alturkistani[1], Mohammed A. El-Affendi[2]
[1]College of Computer and Information Sciences, Prince Sultan University, Riyadh, Kingdom of Saudi Arabia
[2]Department of Computer Science, College of Computer and Information Sciences, Prince Sultan University, Riyadh, Kingdom of Saudi Arabia

Article Info

Article history:

Received Sep 19, 2021
Revised Jun 22, 2022
Accepted Jul 18, 2022

Keywords:

Cybersecurity management
Deep Q-learning
Incident response
Machine learning
Optimization
Reinforcement learning

ABSTRACT

The main purpose of this paper is to explore and investigate the role of deep reinforcement learning (DRL) in optimizing the post-alert incident response process in security incident and event management (SIEM) systems. Although machine learning is used at multiple levels of SIEM systems, the last mile decision process is often ignored. Few papers reported efforts regarding the use of DRL to improve the post-alert decision and incident response processes. All the reported efforts applied only shallow (traditional) machine learning approaches to solve the problem. This paper explores the possibility of solving the problem using DRL approaches. The main attraction of DRL models is their ability to make accurate decisions based on live streams of data without the need for prior training, and they proved to be very successful in other fields of applications. Using standard datasets, a number of experiments have been conducted using different DRL configurations The results showed that DRL models can provide highly accurate decisions without the need for prior training.

Corresponding Author:

Mohammed A. El-Affendi
Department of Computer Science, College of Computer and Information Sciences (CCIS), Prince Sultan University
Riyadh, Kingdom of Saudi Arabia
Email: affendi@psu.edu.sa

1. INTRODUCTION

The main purpose of this paper is to explore the potential of reinforcement deep learning in improving the performance and accuracy of security incident and event management (SIEM) systems. SIEM systems are basically log analysis systems that continuously monitor and manage the security of an organization. They analyze the huge amounts of logs accumulated by different components of the organization information system. A wide range of models are used to detect all types of security intrusions and malicious behaviors and generate alerts to be handled by security operators. There are three levels of security operators:

– Tier 1 security operators (the triage team): perform filtering and screening of alerts before passing them to tier 2 operators,
– Tier 2 security operators: handle the alters and take the appropriate actions based on the type of attack,
– Tier 3 security operators (the team of experts): are responsible for handling advanced types of threats and threat hunting.

Many organizations and enterprises build SIEM systems to protect against cybersecurity threats and satisfy international and local compliance requirements [1]–[5]. SIEM systems are centralized multi-tier platforms that aggregate, transform, and analyze log data collected from all components of organizational information technology (IT) systems to produce cybersecurity insights and handle all types of threats [1], [4], [5]. SIEM solutions usually operate in the context of a security operation center (SOC) in coordination with other cybersecurity solutions [6], [7]. The SOC is a centralized operation unit that handles cybersecurity issues at the technical and non-technical levels through multiple services and specialized security teams.

The main problem with current SIEM solutions is the high rate of generated security alerts, and the high percent of false positives. This slows down the incident response process to the extent that many parts of the alert stream pass without being investigated. This is clearly a serious threat to the organization security. The main bottleneck is that, in most current SOC and SIEM systems the incident response process is manually handled by human SOC analysts who receive the alerts and analyze them to decide what type of reaction to take [1], [3].

Over the past few years, a lot of efforts has been made to apply machine learning to solve problems in many areas of cybersecurity. These efforts have been motivated by the reported great successes of machine learning in other fields [8]–[11]. Areas of applications in cybersecurity include: intrusion detection, botnet finger printing, penetration testing, and anti-jamming. Since the focus of this study is on improving the incident response process, the paragraphs below cover the state-of-the-art in machine learning efforts to improve the incident response process.

Sopan *et al.* [12] demonstrated how machine learning can be applied to automate alert processing in SIEM solutions. They noted that SOC analysts spend a lot of time to take decisions based on the parameters and content of the alert message. They interviewed five senior SOC analysts on the process of analyzing SIEM alerts. Based on the results of the interview they derived feature vectors to characterize alerts. These feature vectors have then been used to train a machine learning model on the decisions corresponding to these vectors. A random forest classifier with more than 200 decision trees has been applied to perform the classification. The underlying dataset was extracted directly from the SIEM environments. The model has then been integrated with a real life SIEM system where analysts can provide feedback regarding the predictions to validate and update the results.

Feng *et al.* [13], developed a user-centric machine learning (ML) framework for SOC's SIEM system to reduce false positive alert rates. The main purpose of this framework is to score risky user behavior feature vectors. The model was tested on a real industry environment. The dataset used was extracted from the raw data logs stored in Symantec internal systems. About 100 features were created to encode and predict user behavior. The ML models used include multi-layer neural network (MLP) with two hidden layers, random forest (RF) with 100 Gini split trees, support vector machine (SVM) with radial basis function kernel and logistic regression (LR). The MLP and RF models provided the best performance.

Nila *et al.* [14] presented a quick way for incident response using efficient ML approach. For the input dataset, the authors analyzed 2.5 million actual log events, and generated one thousand feature vectors, where each vector includes nine manually extracted features. These features have then been used to triage incidents into malicious or safe. The collected dataset was then used to evaluate the efficiency of the ML algorithms showing high accuracies for intrusion detection and prevention systems. Various supervised and unsupervised ML models have been used in the experimentation such as: ZeroR, OneR, naïve Bayes, SVM, J48, RF algorithms (deployed in Weka 3.8). The best accuracy of 95.54% has been achieved by the RF algorithm. In addition, a hybrid classifier technique with linear regression and multi-layer perceptron models were also used to reach maximum accuracy with small number of samples. Security incidents response team may use a proposed approach to quickly differentiate between safe or suspicious events using combination of ML classifiers.

Maeda and Mimura [15] considered a novel solution that automates and emulates red team post-exploitation attacks using reinforcement learning (RL) where agent actions are generated in the context of power shell empire. The model was tested in a real environment to see how efficient are RL models in performing post exploitation attacks. The study deployed 3 RL models: A2C, Q-learning and SARSA, where advantage actor critic (A2C) model showed maximum stability of gaining the rewards. The behavior of the agent was evaluated in Windows test domain network, where the A2C model was able to take admin privileges optimal time.

Lee *et al.* [16] noted that the practice of using machining learning to score and classify events in SIEM solutions is growing, but in many cases the ML models are trained using only public datasets. This is mainly due to the difficulty of preparing real life training datasets. Accordingly, they developed an advanced event profiling system to prepare real life data sets. The authors then demonstrated that deep learning models outperform conventional machine learning models on the collected datasets.

Sethi *et al.* [17] noted that most of the currently existing intrusion detection system (IDS) are ineffective and are uncapable of providing high threat detection accuracies. Accordingly, they proposed a context aware distributed graph IDS detection model comprising multiple deep reinforcement learning (DRL) agents. The DRL agents are installed on inner routers within different contexts in the underlying network. In their experimentation they used three standard public datasets NSL-KDD, UNSW-NB15 and AWID, and compared their results with other models applied to the same datasets. The main contribution of the study is to distribute multiple RL agents across the network to explore the environment and gain rewards. The proposed system is also capable of classifying attacks and events with good accuracies.

Oprea *et al*. [18] developed malicious activity detection in enterprises (MADE) which is a system that could be deployed on a wide range of geographically distributed enterprises. The underlying approach is based on ML supervised learning, where features extracted from web proxy network logs are used to detect malware embedded in hypertext transfer protocol (HTTP), hypertext transfer protocol secure (HTTPS) (beaconing) traffic. The models used include logistic regression, decision trees (DT), random forest (RF), and SVM. The maximum accuracy achieved is 97%. MADE is capable of detecting a wide range of well-known malicious activities, which reduces SOC workload.

Farooq and Otaibi [19] proposed an efficient ML based cyber threat detection approach that can minimize false positive alerts. Their approach is mainly based on selecting the appropriate machine learning model for each of the taxonomies outlined in MITRE. The study applied some existing ML models to some of the MITRE attack taxonomies to demonstrate how these models can reduce false positives.

Chatterjee and Namin [20] applied a deep reinforcement learning based framework to automate the detection of malicious websites used in phishing attacks. The authors used a balanced dataset consisting of legitimate and illegitimate phishing uniform resource locators (URLs) to train and test the model. As expected, the model produced high accuracy, but no comparisons with other models were provided.

The main purpose of this paper is to investigate how DRL [21] can be used to improve the post-alert incident response process. Although machine learning has been applied at different levels of cybersecurity systems such as intrusion detection [16], [17], [22], network traffic reduction and botnet detection [23], defense in software-defined networks [24], detect malicious URLs and phishing websites [25], network anti-jamming development [26], network penetration-testing [27], very little has been done regarding the improvement of the last mile incident response process [14].

RL is the third approach in the main taxonomy of machine learning [28]. The main attraction of this approach is that models learn from experience using a trial-and-error process without the need for prior training. This makes it possible to place these models in positions where they can learn from continuous streams of log events passing out of SIEM systems. Over the past few years, a lot has been done and written about reinforcement learning and deep reinforcement learning [21], [28]–[31]. An RL model consists of five components:

a. Agent: a dynamic entity that takes "Actions" and makes decisions based on the context;
b. Environment: the agent makes decisions based on feedback form a stochastic "environment", which represents the context in which the agent operates. It is assumed that the inners of the environment are unknown to the agent;
c. Actions: in order to get feedback, the agent performs actions on the environment;
d. Sate: taking an action moves the environment from one state to another; and
e. Reward: Feedback from the environment comes in the shape of a reward, which is a numeric indicator showing how good or bad is the action.

A good example is the driverless autonomous car. Where the virtual controller of the car (the agent), drives (takes actions) the car through roads (the environment), and responding properly to events and signals (feedback) from the environment. Figure 1 shows a how agents interact with the environment in RL models.

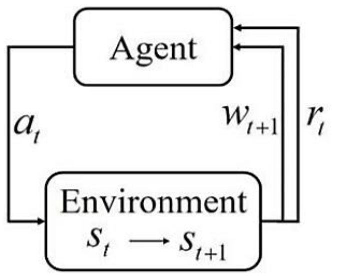

Figure 1. Structure of reinforcement learning models

Q-learning is a recent simple reformulation of the initial RL algorithm that triggered the current popularity of RL. The algorithm is based on the Q version of the temporal difference equation [28].

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma * \underset{a_s \in \mathcal{A}(s)}{argmax} Q(s', a_s) - Q(s,a) \right)] \tag{1}$$

Here s is the set of states, a is the set of actions, *Q(s,a)* is the action value function whose value is based on the current state and the action taken. The computation of *Q(s,a)* requires the use of the of a two dimensional table that contains computed values of Q based on s and a. Figure 2 describes the implementation of tabular Q-learning algorithm [28].

Note that the main purpose of this algorithm is to compute the optimal action a, given the current state and the value of *Q(s,a)*. The value of *Q(s,a)* is computed iteratively as shown in Figure 3. One main problem with the tabular implementation of the Q-learning algorithm as shown in Figure 3 is that the table may grow dramatically for large values of s and a. To solve this problem, researchers developed a special version of the algorithm where the Q table is replaced by an approximation neural deep learning function. In this type of model, a deep learning model is used to predict the value of *Q(s,a)* [28]. The term "deep Q-learning (DQN)" has been coined to refer to this version of the Q-learning algorithm. DQN is the main model used in our experimentation below. DQN models apply an experience replay approach as shown in Figure 4 to improve the prediction power of the model.

```
1:  **input**:apolicy that uses the action-value function, $\pi(a|s,Q(s,a))$
2:  Initialize $Q(s,a) \leftarrow 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
3:  **loop**:for each episode
4:      Initialize environment to provide $s$
5:      **do**:
6:          Choose $a$ from $s$ using $\pi$, breaking ties randomly
7:          Take action, $a$, and observe $r, s'$
8:          $Q(s,a) \leftarrow Q(s,a) + \alpha \left[ r + \gamma * \underset{a_s \in \mathcal{A}(s)}{argmax} Q(s', a_s) - Q(s,a) \right]$
9:          $s \leftarrow s'$
10:     **while** $s$ is not terminal
```

Figure 2. The Q-learning algorithm

```
Repeat
    1- Take action a (either selected randomly, or based on Q(s,a))
    2- Update the value of Q(s,a) based on the second component of (1). Note that the second component contains a weighted
       sum of the reward r and the max delta value [Q(s',a)-Q(s,a)], where Q(s',a) is the value of Q when the system moves to
       one of the future states s'.
    3- s=s'
until the optimal action is reached
```

Figure 3. Computing the value of *Q(s,a)*

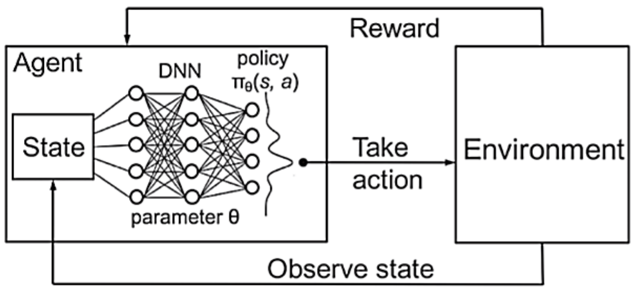

Figure 4. Experience replays in DQN models

2. METHODS

2.1. Identify and acquire the public datasets

The first step is to look for appropriate public datasets that can be used to evaluate the model. The main criterion for selecting the datasets is the relevance of the context in which they have been used, namely, being used in the context of attempts to automate decision making in SIEM solutions. This makes it possible to benchmark our results. Since the main purpose of our study is to explore the possibilities and provide a proof of concept, the choice has been limited to the following datasets:

- The SNL-KDD “intrusion detection” dataset, originally prepared by DARPA by simulating operations of US Air Force local area network (LAN) network with multiple attack types categorized into four classes [32], [33]. Lately the dataset has been taken care of and further developed by the Canadian Institute of Cybersecurity [34]. The training component of the dataset contains 125,973 items, while the testing component consists of 22,54. As an intrusion dataset, the SNL-KDD dataset provides records about suspected attacks on networks systems. Each row in the dataset represents an attempted attack with relevant attributes that describe the the context and behavior of the attack. The last column in the dataset is a label indicating the type (class) of attack.
- Ebub 2017 URL web phishing URL dataset [35] prepared by a group of researchers [25]. The dataset initially prepared by Sahingoz *et al.* [25], and later used by Chatterjee and Namin [20] to test their DQN model. The dataset contains 73,575 URLs-36,000 legitimate and 37,175 phishing websites.
- We started experimentation with the malware API dataset, which has been prepared by a Turkish cybersecurity Institution [33], but unfortunately, we realized that the dataset is unbalanced due to the lack of enough benign samples in that.

2.2. Model configuration

The second step in the methodology is to configure the DQN model described in section 1 above. The main purpose of the configuration is to determine the optimal parameters for (1). These include the discount factor, the learning rate alpha, and the greedy policy factor. Table 1 shows the optimal values tuned through experimentation. The other set of parameters in Table 1 include the type of deep learning model used in the DQN, the actions and the rewards. These are the fundamental parameters for the operation of the model and vary from one implementation to another based on the type of problem addressed. The deep learning model selected is the well-known MLP, which led to the excellent results we obtained. Regarding the actions, we have only two actions “fix” or “escalate” to the upper tier in the hierarchy of SOC operators. Remember that our model is addressing the post-alert stage in SIEM systems. It attempts to automate the work of the two lower tiers of SOC operators. Since we have only two actions, there are two values for the rewards, 0 for “fix”, and 1 for escalate. The values are used to update the reward matrix *R*, which is crucial to the computation of *Q(s,a)* in the (1). The symbol “s” is used to represent the current state. With each action, the system moves from one state to another. In our case the number of states is equal to the length of the dataset, since the dataset represents “the environment”.

Table 1. DQN model configuration

Parameter	Description/Value
Discount factor (gamma)	0.73
Learning rate (alpha)	0.5
Epsilon greedy policy factor	0.5
Experience replay	A three-layer MLP deep learning model
Actions	Two actions, ‘fix’ or ‘escalate’
Rewards	Reward for ‘fix’ =0, reward for ‘escalate.’ =1
States	The states are represented by the input stream vectors. The stream moves from one state to another based on the value of passing vectors

2.3. A multiagent DQN framework for deploying the model in production environments

To deploy the DQN model in production environments, we developed a framework to show how multiple DQN agents may be used to address multiple streams of alerts in parallel. This will of course improve performance and lead to more accurate results. Our multiagent framework is similar to the frameworks developed by Sethi *et al.* [17] and Maeda and Mimura [15]. The main difference between our framework and the frameworks reported in the above papers is that the DQN agents in our case are positioned on top of the alert layer in SIEM systems. This means that the main function of the underlying DQN models is to predict the correct decisions. Figure 5 shows the structure of the framework and the functionality of each DQN agent.

As apparent from the diagram in Figure 5, multiple DQN models can be executed in parallel to predict the appropriate decisions based on the incoming stream of alerts. It is assumed that the underlying SIEM solution provides multiple streams of alerts based on some event profiling approaches like the one reported by Lee *et al.* [16]. Although, the Figure 5 shows six possible types of DQN agents, the number of agents is dynamic and can be more or less than six.

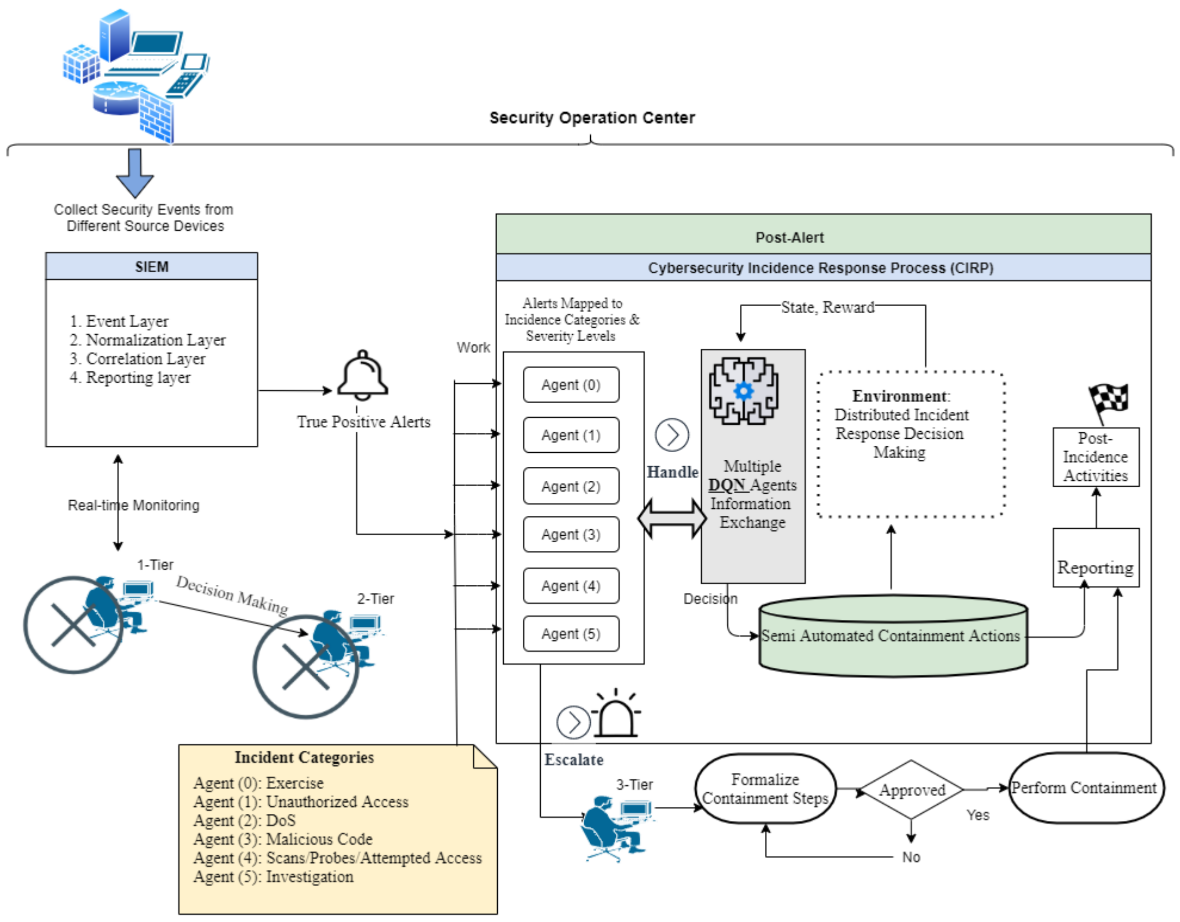
Security Operation Center
Collect Security Events from Different Source Devices
SIEM
1. Event Layer
2. Normalization Layer
3. Correlation Layer
4. Reporting layer
True Positive Alerts
Real-time Monitoring
1-Tier
Decision Making
2-Tier
Work
Post-Alert
Cybersecurity Incidence Response Process (CIRP)
Alerts Mapped to Incidence Categories & Severity Levels
Agent (0)
Agent (1)
Agent (2)
Agent (3)
Agent (4)
Agent (5)
Handle
State, Reward
Multiple DQN Agents Information Exchange
Environment: Distributed Incident Response Decision Making
Post-Incidence Activities
Reporting
Decision
Semi Automated Containment Actions
Escalate
3-Tier
Formalize Containment Steps
Approved
Yes
No
Perform Containment
Incident Categories
Agent (0): Exercise
Agent (1): Unauthorized Access
Agent (2): DoS
Agent (3): Malicious Code
Agent (4): Scans/Probes/Attempted Access
Agent (5): Investigation

Figure 5. A multiagent DQN framework for incident response optimization

3. RESULTS

Based on the above framework, and the available reliable public datasets we experimented with two parallel DQN agents, and two emulated steams of data. Each stream is emulated with one of the two datasets described below. The experiments have been performed using a Surface Laptop 3 with Intel Core i7 processor and 16 GB RAM. Tables 2 and 3 show the results for the two datasets used in the experimentation: the SNL-KDD and the phishing URLs respectively. In both cases, the experience replay model of the DQN has been used to produce the results. For experience replay, we used a standard three-layer MLP deep learning model. As apparent from the tables, the algorithm started to converge after episode 10. Note that according to experience replay, the MLP model is trained again using batches of the predictions produced by the model during the first phase where the model sees the data stream for the first time. Accordingly, experience replay is a sort of testing and training at the same time.

Table 2. DQN results for the SNL-KDD dataset

Eps	acc	precision	recall	F1
0	0.5001	0.4663	0.5013	0.4832
1	0.7524	0.7257	0.7523	0.7388
2	0.8137	0.7919	0.8134	0.8025
3	0.8523	0.8337	0.8525	0.8430
4	0.8919	0.8778	0.8919	0.8848
5	0.9212	0.9100	0.9219	0.9159
6	0.9431	0.9347	0.9438	0.9392
7	0.9598	0.9534	0.9608	0.9571
8	0.9698	0.96475	0.9706	0.9677
9	0.9761	0.9720	0.9767	0.9743
10	0.9792	0.9761	0.9792	0.9776
11	0.9819	0.9792	0.9820	0.9806
12	0.9826	0.9804	0.9825	0.9814

Table 3. Results for the phishing URL dataset

Eps	acc	precision	recall	F!
0	0.5006	0.8589	0.5015	0.6333
1	0.7485	0.9489	0.7477	0.8364
2	0.8111	0.9639	0.8107	0.8807
3	0.8510	0.9712	0.8519	0.9077
4	0.8893	0.9783	0.8911	0.9327
5	0.9163	0.9847	0.9169	0.9496
6	0.9416	0.9886	0.9429	0.9652
7	0.9569	0.9916	0.9579	0.9745
8	0.9664	0.9936	0.9671	0.9802
9	0.9723	0.9949	0.9728	0.9837
10	0.9751	0.9956	0.97538	0.9855
11	0.9776	0.9968	0.9771	0.9868
12	0.9786	0.9975	0.9775	0.9874

3.1. Results for the SNL-KDD dataset

It is interesting to note that there is an exponential growth in accuracy at the beginning, the model started to converge at about episode 10. This is expected, since the knowledge gained from the early parts of the data stream is not enough to take firm decisions. But as knowledge accumulates the decision become clearer and clearer.

3.2. Results for the phishing URL dataset

The behavior here again is similar to the behavior of the model with the previous dataset. Again, there is some exponential growth in accuracy at the beginning, with convergence at the same episode level. However, the big differences between the values of precision, recall and F1 metrics in the early episodes indicate that there is imbalance in the data. Unlike deep learning models, the effects of imbalance on the accuracy of the model completely disappeared at convergence time

3.3. Comparisons

We compared our results for the SNL-KDD with results obtained by Sethi *et al.* [17], who applied a DQN to the same dataset as part of their context-aware framework for intrusion detection framework. The accuracy obtained by Sethi *et al.* [17] is 81.8%, compared to 98.26% obtained through our model. As for the phishing URL Dataset, we compared our results with the results obtained by Chatterjee and Namin [20], whose DQN model achieved 0.901% accuracy compared to 97.85% achieved by our model. Table 4 compares the accuracy of our model with state-of-the-art models:

Table 4. Comparisons with state-of-the-art results

Dataset	Our Model	Sethi *et al.* [17]	Chatterjee and Namin [20]
SNL-KDD IDS Dataset [34]	98.26%	81.8%	---
Phishing URLs [35]	97.86%	---	90.01%

4. DISCUSSION

It is clear from the table that the accuracy of our model is higher than those produced by the other two models, although both used a variation of a DQN model. The main reason is that our model has been built from scratch, and a lot of time has been spent on configuring, tuning, and testing it. Another reason is the way rewards are allocated. We used a simple binary scheme, 0 for 'fix' action, and 1 for 'escalate' action. The other two models used a similar two-level reward system, but the actual values of the rewards are different. The major advantage of reinforcement learning is that no prior training is required. The model learns from experience, while consuming the streams of data. This implies that there is no danger of overfitting (like what happens in the case of deep learning).

5. CONCLUSION

In this study we explored the potentials of DRL in optimizing the decision process in SIEM incident response systems. A DQN model has been used to investigate the possibilities. A couple of standard public datasets have been used to emulate data streams in real life SIEM systems. The results are very encouraging, and the accuracies are even higher than those produced by trained supervised models. This is an indication that DRL models can be used in the post-alert and post-exploitation decision phases, where model training is not feasible.

ACKNOWLEDGEMENTS

The authors would like to than PSU EIAS Data Science and Cyber Analytics Lab for the great help and resources provided by the lab.

REFERENCES

[1] Exabeam, "The SOC, SIEM, and other essential SOC tools," *exabeam*. https://www.exabeam.com/siem-guide/the-soc-secops-and-siem/ (accessed Aug 27, 2021).
[2] Elastic, "Anomaly detection with machine learning," *elastic*. https://www.elastic.co/guide/en/siem/guide/current/machine-learning.html (accessed Aug 27, 2021).
[3] S. Anson, *Applied incident response*. Wiley, 2019., doi: 10.1002/9781119560302.
[4] E. C. Thompson, *Cybersecurity incident response*. Berkeley, CA: Apress, 2018., doi: 10.1007/978-1-4842-3870-7.
[5] Y. Diogenes, N. DiCola, and J. Trull, "Microsoft azure sentinel: planning and implementing microsoft cloud-native SIEM solution," in *IT Best Practices-Microsoft Press series*, 2020.
[6] J. Muniz, G. McIntyre, and N. AlFardan, *Security operations center: building, operating, and maintaining your SOC*. Pearson Education, 2015.
[7] J. Muniz, A. Lakhani, O. Santos, and M. Frost, *The modern security operations center*. Addison-Wesley Professional, 2021.
[8] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review," *Robotics*, vol. 10, no. 1, Jan. 2021, doi: 10.3390/robotics10010022.
[9] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 4909–4926, Feb. 2020, doi: 10.1109/TITS.2021.3054625.
[10] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Systems with Applications*, vol. 173, Jul. 2021, doi: 10.1016/j.eswa.2021.114632.
[11] A. Buchard *et al.*, "Learning medical triage from clinicians using deep Q-Learning," *arXiv preprint arXiv:2003.12828*, Mar. 2020
[12] A. Sopan, M. Berninger, M. Mulakaluri, and R. Katakam, "Building a machine learning model for the SOC, by the input from the SOC, and analyzing it for the SOC," in *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Oct. 2018, pp. 1–8., doi: 10.1109/VIZSEC.2018.8709231.
[13] C. Feng, S. Wu, and N. Liu, "A user-centric machine learning framework for cyber security operations center," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Jul. 2017, pp. 173–175., doi: 10.1109/ISI.2017.8004902.
[14] C. Nila, I. Apostol, and V. Patriciu, "Machine learning approach to quick incident response," in *2020 13th International Conference on Communications (COMM)*, Jun. 2020, pp. 291–296., doi: 10.1109/COMM48946.2020.9141989.
[15] R. Maeda and M. Mimura, "Automating post-exploitation with deep reinforcement learning," *Computers and Security*, vol. 100, Jan. 2021, doi: 10.1016/j.cose.2020.102108.
[16] J. Lee, J. Kim, I. Kim, and K. Han, "Cyber threat detection based on artificial neural networks using event profiles," *IEEE Access*, vol. 7, pp. 165607–165626, 2019, doi: 10.1109/ACCESS.2019.2953095.
[17] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, and Y. Venu Madhav, "A context-aware robust intrusion detection system: a reinforcement learning-based approach," *International Journal of Information Security*, vol. 19, no. 6, pp. 657–678, Dec. 2020, doi: 10.1007/s10207-019-00482-7.
[18] A. Oprea, Z. Li, R. Norris, and K. Bowers, "MADE," in *Proceedings of the 34th Annual Computer Security Applications Conference*, Dec. 2018, pp. 124–136., doi: 10.1145/3274694.3274710.
[19] H. M. Farooq and N. M. Otaibi, "Optimal machine learning algorithms for cyber threat detection," in *2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim)*, Mar. 2018, pp. 32–37, doi: 10.1109/UKSim.2018.00018.
[20] M. Chatterjee and A.-S. Namin, "Detecting phishing websites through deep reinforcement learning," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, Jul. 2019, pp. 227–232, doi: 10.1109/COMPSAC.2019.10211.
[21] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, 2018, doi: 10.1561/2200000071.
[22] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, Mar. 2020, doi: 10.1016/j.eswa.2019.112963.
[23] M. A. Sankaran, A. K. B. Murat, M. Tharrshinee, and G. Yuvasree, "Botnet detection using machine learning," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 7, pp. 5116–5121, 2020.
[24] Y. Han *et al.*, "Reinforcement learning for autonomous defence in software-defined networking," in *Lecture Notes in Computer Science*, Springer International Publishing, 2018, pp. 145–165, doi: 10.1007/978-3-030-01554-1_9.
[25] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications*, vol. 117, pp. 345–357, Mar. 2019, doi: 10.1016/j.eswa.2018.09.029.
[26] Y. Li *et al.*, "On the performance of deep reinforcement learning-based anti-jamming method confronting intelligent jammer," *Applied Sciences*, vol. 9, no. 7, Mar. 2019, doi: 10.3390/app9071361.
[27] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," *Information*, vol. 11, no. 1, Dec. 2019, doi: 10.3390/info11010006.
[28] P. Winder, *Reinforcement learning*. O'Reilly Media, Inc., 2020.
[29] N. Habib, *Hands-on Q-Learning with Python*. Packt Publishing, 2019.
[30] A. Lonza, *Reinforcement learning algorithms with Python*. Packt Publishing, 2019.
[31] M. Lapan, *Deep reinforcement learning hands-on-second edition*. Packt Publishing, 2020.
[32] D. Protić, "Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets," *Vojnotehnicki glasnik*, vol. 66, no. 3, pp. 580–596, 2018, doi: 10.5937/vojtehg66-16670.
[33] F. O. Catak and A. F. Yazi, "A benchmark API call dataset for windows PE malware classification," *Machine Learning Based Cyber Security*, 2019
[34] UNB, "Canadian Institute for Cybersecurity (CIC) and UNB, NSL-KDD dataset." University of New Brunswick. [Online]. Available: https://www.unb.ca/cic/datasets/index.html
[35] E. Buber, "*Ebbu2017 phishing dataset*," GitHub. [Online]. Available: https://github.com/ebubekirbbr/

BIOGRAPHIES OF AUTHORS

Hilala Alturkistani received the BS in Information Systems from Prince Sultan University, then she obtained her MS in Cybersecurity from the same university. She is currently preparing for her PhD while teaching parttime with the college of computer and information sciences in PSU. She is conducting research. She can be contacted at email: 219420735@psu.edu.sa and hilalahfeda@gmail.com.

Mohammed A. El-Affendi is a Professor of Computer Science in the Department of Computer Science, PSU, KSA and a former dean of the college. Currently, prof ELAffendi is the founder and Director of the Center of Excellence in Cybersecurity and the founder and leader of the Data Science and Cyber Analytics lab. Currently he is conducting research in the áreas of data scinece and applied artificial intelligence (AI) with a focus on applications of machine deep learning and reinforcement learning to real life AI problems. He can be contacted at email: affendi@psu.edu.sa.