A Long-lasting Reinforcement Learning Intrusion Detection Model

Roger Robson dos Santos, Eduardo Kugler Viegas, Altair Santin and Vinicius Vielmo Cogo

Abstract Several works have proposed highly accurate network-based intrusion detection schemes through machine learning techniques. However, they are unable to address changes in network traffic behavior over time. Authors often assume periodic model updates, but without taking into account the challenges they entail. This paper proposes a long-lasting reinforcement learning model for intrusion detection that withstands long periods without model updates. Our proposal builds machine learning models through reinforcement learning to keep their accuracy for longer periods. Then, we cope it with a verification technique to ensure that only reliable classifications are accepted over time. Experiments performed using a dataset spanning a year of real network traffic, composed of 10TB of data, show that the technique we propose remains reliable for ten months without model updates. Additionally, our proposal increases its accuracy when coped with the verification technique.

1 Introduction

The number of cyberattacks has significantly increased in recent years [1]. Not surprisingly, a Kaspersky security [2] report has revealed that 44% of organizations faced some incident of vulnerability security exploit in the last year [2]. A cyberattack can disrupt, render a service unavailable to its legitimate users, or even cause

Roger Robson dos Santos
Pontifícia Universidade Católica do Paraná, PUC-PR, e-mail: robson.roger@ppgia.pucpr.br

Eduardo Kugler Viegas
Pontifícia Universidade Católica do Paraná, PUC-PR e-mail: eduardo.viegas@ppgia.pucpr.br

Altair Santin
Pontifícia Universidade Católica do Paraná, PUC-PR e-mail: santin@ppgia.pucpr.br

Vinicius Vielmo Cogo
LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal e-mail: vvcogo@fc.ul.pt

monetary losses. For instance, the exploit of a zero-day privilege escalation discovered in 2019 affected several versions of Microsoft Windows [3]. Therefore, administrators need to have access to security solutions that enable the classification of new malicious activities [4].

To achieve such a goal, administrators usually resort to intrusion detection systems (IDS), which employ techniques either *signature-based* or *behavior-based* [5]. Signature-based detection uses a knowledge database of previously known attack behaviors. However, it is unable to detect new or unforeseen attacks [6]. In contrast, behavior-based detection relies upon a behavior database to build a behavioral model of the system. Consequently, this technique is more likely to detect new attacks if they behave similarly to known malicious activities.

Several works have been proposing highly-accurate *behavior-based* IDS schemes, which usually rely upon pattern recognition techniques [7]. In such an approach, a machine learning (ML) model is built according to a behavioral database containing both legitimate and attack activities [8]. The built model is used in production over time, assuming that it will detect new kinds of attacks. However, pattern recognition detects similar items rather than new diverse ones [7]. Consequently, if a new attack does not behave similarly to known ones, the ML model will only be able to classify it correctly if a model update takes place. Alternatively, the ML model update task is not easily feasible [9]. This is because the training task requires the building of an updated training database, the proper labeling of events (often only achieved through human assistance), and the execution of a computationally expensive model update process. Therefore, in order to enable the reliable deployment of *behavior-based* IDS schemes, it becomes imperative techniques that are able to withstand long periods without human intervention [10]. Otherwise, the designed techniques may become outdated before they are even used in production (real world) [9].

In recent years, another popular approach for *behavior-based* classification resort to reinforcement learning (RL) techniques. Such a technique addresses classification through an agent, which learns the environment behavior through trial-and-error interactions, hence, improving itself over time. As a result, the underlying built model learns the environment behavior over time. RL-based approaches have yielded promising results in several fields, such as actor behavior learning in virtual gaming [11], autonomous driving [12], and even prosthesis control learning [13]. However, their applicability in the intrusion detection field still in its infancy.

In light of this, this paper proposes a new reliable intrusion detection model through RL techniques aiming to withstand long periods reliably without human assistance. Our proposal is twofold. First, it addresses the challenge of performing intrusion detection through an RL technique by aiming the preservation of the obtained accuracy over time, even without periodic model updates. Second, we propose a verifier technique coped with the proposed RL approach to assess the classification output. As a result, our proposal maintains or even improves the classification accuracy over time, even in the absence of model updates, neither human assistance. In summary, this paper presents the following contributions:

- A novel RL-based intrusion detection scheme to provide classification models that withstand long periods without human intervention. Our proposed technique

outperforms the approaches from the state-of-the-art network traffic classification concerning their accuracy over time;
- A verifying technique, coped with the proposed RL-based approach, to assess the classification reliability over time. The proposed technique maintains or even improves the classification accuracy over time according to the administrator needs.

2 Preliminaries

2.1 Network-based Intrusion Detection

Over the last years, several highly accurate ML-based Network-based Intrusion Detection Systems (NIDS) were proposed in the literature [14, 15, 17, 18]. In general, the proposed approaches are composed of four sequential modules: *Data Acquisition*, *Feature Extraction*, *Classification* and *Alert* modules. First, the *Data Acquisition* module gathers network data from the monitored environment, e.g., network packets. Second, the *Feature Extraction* module extracts behavioral features from the network data composing a feature vector. In NIDS, behavioral features are often represented through a network flow, with the network behavior from a host/service in a given time window, e.g., the ratio of exchanged packets/bytes in a 15 seconds interval. Third, the *Classification* module executes the underlying classification algorithm based on the extracted feature vector and classifies it as normal or attack. Finally, if a malicious event is found, the *Alert* module reports it.

In NIDS, the classification task is usually performed through pattern recognition techniques [19]. In pattern recognition, a ML model is built through a computationally expensive process named training. The training phase produces a model through a training dataset, which contains both normal and malicious activities (e.g., network flows) in a given period. As a result, the ML model detects events according to the behavior extracted from the training dataset [21]. Therefore, when the environment behavior changes, either due to the discovery of new attacks or due to the offering of new services, a new ML model must be built. However, the ML model update process is time-consuming and often demands human assistance.

2.2 Reinforcement Learning

Different from traditional ML-based techniques, reinforcement learning (RL) approaches aim at finding an optimal policy strategy. To achieve such a goal, RL-based approaches rely upon an *agent*. The *agent* is connected to the environment through the *perception* and *action*. The *perception* enables the *agent* to gather the environment *state*, whereas the *action* enables the *agent* to act over the environment. At the training phase, the RL algorithm is executed iteratively over the environment, in

which at each iteration, the *agent* receives the current environment state as input, as measured through the feature vector. Then, the *agent* performs an *action* over the environment, through an underlying ML model, which changes its state, and generates a *reward* as output. As a result, the training phase aims at finding an agent able to choose actions that increase its obtained rewards.

RL-based approaches are significantly different from traditional ML-based ones. Traditional ML relies on an input and output pair. An ML model receives a given event feature vector as input and outputs its estimated class value. In contrast, RL-based techniques are not given the event class value. Instead, after an *action* is performed, the *agent* only receives the *reward* for its *action* and the subsequent environment *state*. Therefore, the *agent* learns the optimal decision threshold for the long-term, given that it optimizes its rewards over time.

3 Related Works

Several highly accurate *behavior-based* NIDS have been proposed in the literature over the last years. In general, proposed approaches aim at improving their obtained accuracy in a test dataset, without taking into account the evolving behavior of network traffic over time.

For instance, Farnaaz *et al.* [8] applied a Random Forest classifier for non-linear classification to improve their obtained accuracy in the NSL-KDD dataset. Although the authors were able to improve detection in their system, the dataset they used is almost 20-years old and present several flaws, tampering the evaluation of their system reliability over time. Similarly, Nanda *et al.* [14] cope a rule-based classifier with the Random Forest classifier to perform intrusion detection. In their work, using the original KDD99 version, the authors were able to improve the system accuracy, without taking into account the detection of new behaviors. In contrast, a clustering-based approach coped with a genetic algorithm was proposed by Sukumar *et al.* [15]. Their approach performs unsupervised anomaly detection, i.e., without the prior knowledge of events' label. However, the dataset they used also is unrealistic, while their obtained accuracy is significantly lower than other proposals.

Some authors have also proposed MLP-based approaches for intrusion detection. For instance, Darkaie *et al.* [17] propose an intrusion detection based on MLP coupled with a feature reduction technique. In their work, also using an unrealistic dataset, the authors were able to improve their accuracy when increasing the MLP architecture layout. Similarly, Congyuan *et al.* [18] cope a recurrent neural network with a MLP to improve their system accuracy. Authors have shown that MLP-based techniques present higher accuracy rates. However, the dataset they used is unrealistic, and the accuracy behavior is not evaluated over time.

Due to its promising reported results, RL-based techniques have drawn attention in the research community. However, its applicability in the intrusion detection field still is in its infancy. Caminero *et al.* [10] propose a RL-based detection technique in which the environment is simulated using the training dataset. Authors were able

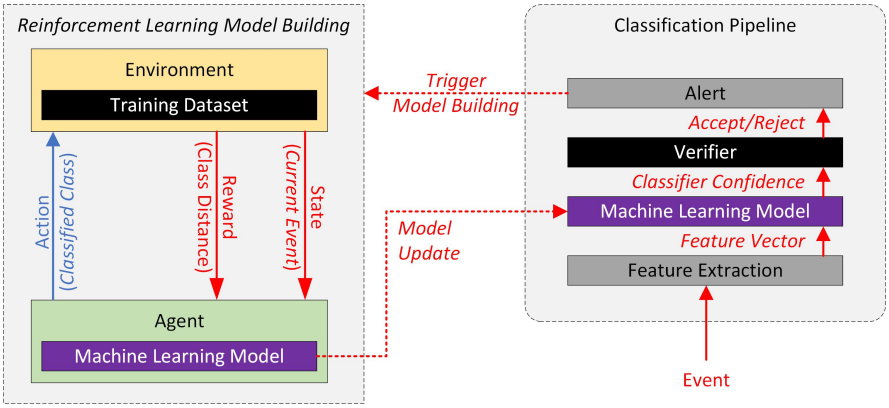

Fig. 1: Long-lasting reinforcement learning intrusion detection model architecture.

to improve their accuracy, when compared to traditional approaches, by using a MLP architecture and through the evaluation using two datasets. In Lopez-Martinz *et al.* [20], the authors evaluate several reinforcement learning algorithms for the intrusion detection task. In their work, the evaluated models presented good results with highly accurate rates.

To the best of our knowledge, this paper presents the first work to evaluate and address the challenge of network traffic behavior changes over time through an RL-based technique. To achieve such a goal, our work presents a novel dataset containing a year of real network traffic anomalies, enabling the proper evaluation of machine learning techniques regarding their reliability over time.

4 A Long-lasting Intrusion Detection Model

In this section, we propose a novel long-lasting intrusion detection model to address the challenges of classifying evolving network traffic behavior over time. It is composed of two main components (*Reinforcement Learning Model Building* and *Verification*, as shown in Figure 1) and focuses on maintaining the accuracy obtained at the training time for longer periods without human assistance or model updates.

Our proposal considers a custom-tailored model built with reinforcement learning. The classification starts with a to-be-classified network event, which is forwarded to the *Feature Extraction* module to extract the feature vector. Then, the ML model classifies the given feature vector, producing a classifier confidence value. The classifier confidence measures the underlying ML model certainty on its classification. The classification confidence is classifier-dependent. For instance, the Random Forest classifier computes its confidence according to the ratios of trees that classified a given instance to a class. Finally, the *Verifier* module establishes whether the classified event can be reliably accepted or not according to the ob-

tained confidence value. As a consequence, only highly-confident classifications are accepted, while rejected instances are assumed to be unknown event behavior and can be used as a measure to trigger the model update task, for instance.

The *Verifier* module is responsible for ensuring that only known, and most likely correct classifications, are accepted by our model. As a consequence, the proposed architecture maintains or even improves (despite a higher rejection rate) the obtained accuracy over time. Therefore, through the proposed verification technique, an administrator is now able to ensure that the desired level of accuracy is kept, even in the absence of model updates.

The model update procedure (*Reinforcement Learning Model Building*, Figure 1) aims at providing underlying ML models with a higher model lifespan. ML models become capable of maintaining their accuracy (i.e., the one obtained during the training) for longer periods. To achieve such goal, our proposal leverages techniques of reinforcement learning to build ML models that can cope with the network traffic behavior changes over time with less impact on accuracy.

The next subsections detail this novel technique to build models using reinforcement learning, focusing on obtaining higher model lifespan and the proposed verification technique.

4.1 Reinforcement Learning Model Building

Our proposal leverages the RL technique to build underlying ML models with longer lifespans. ML models become capable of maintaining, for longer periods, the accuracy they obtained during the training period. Hence, they become able to be reliably deployed in production environments without demanding periodic model updates.

To achieve such a goal, we propose a custom-tailored RL algorithm for NIDS, through the well-known *QLearning* algorithm. The proposed algorithm characterizes the environment as a Markov Decision Process (MDP) through a modified version of Q-Learning [25] algorithm, a well-known RL algorithm. The adopted RL algorithm was used because it is agnostic since it can use any underlying ML classifier.

The novel algorithm receives as input a set of $States$, which comprises the set of instances from the training dataset. The *agent*, in turn, may perform two $actions$, i.e., classify the read instance as either *normal* or *attack*. The $action$ represents the *agent* classification for the current environment $state$. Consequently, the *agent* receives the *reward* for its $action$ (*classification*) according to Eq. 1, where $confidence^{inst_i}$ denotes the confidence of the classifier's decision for the current environment $state$. Therefore, the *reward* is computed according to the distance of the classifier confidence to the proper instance label. Hence, correct classifications (true-positives and true-negatives) receive higher rewards, while misclassifications receive lower ones.

$$Reward = \begin{cases} \text{confidence}^{inst_i} & \text{if } truepositive \text{ or } truenegative \\ 1-\text{confidence}^{inst_i} & \text{otherwise} \end{cases} \quad (1)$$

As a result, the proposed algorithm iteratively aims at increasing its rewards. By doing so, the *agent* attempts to improve the underlying ML model confidence values to approximate to the proper instance label, instead of only increasing its overall accuracy. Therefore, the proposed RL algorithm can build underlying ML models with a higher model lifespan, assuming they are also optimized to increase their rewards (i.e., improve their confidence values), instead of only improving their accuracy, as made by traditional techniques.

4.2 Verification of Classifications Over Time

Through the mentioned RL model building technique, our proposal can build ML models with a longer lifespan. However, the network traffic behavior changes over time, and, as a consequence, any ML model will perform unreliable classifications and demand model updates. However, the model update task is not easily feasible in production.

Therefore, the goal of the *Verifier* module (*Verifier*, Figure 1) is to leverage the underlying ML model, built aiming a longer model lifespan to ensure the classification reliability over time. To achieve such a goal, the *Verifier* module asses the individual classification confidence values output by the underlying ML model, as shown in Figure 1. The insight is that instances that were classified with highly-confident values can be reliably accepted, while low-confident ones are most likely misclassifications, and should be rejected by our model. As a consequence, our proposal ensures that only highly-confident instances are accepted over time, maintaining the system reliability even in the absence of model updates.

Rejected instances can be used as a measure of aged ML models. A high rejection rate indicates that the ML model needs an update, taking into account that the current model cannot reliably classify new events without impact on its accuracy. Therefore, it can be used to trigger the model update process (*Trigger Model Building*, Figure 1).

4.3 Discussion

The proposed model aims at addressing the challenge of evolving network traffic behavior without human intervention nor model updates. To achieve such a goal, our proposal is twofold. First, we build underlying ML models aiming a longer model lifespan through a RL technique. Our proposal aims at building ML models with better classification confidence values instead of only aiming higher accuracy. As a consequence, the built model provides reliable classifications over time. Second, we cope our built ML model with a verification technique. The verification approach ensures that only highly-confident classifications are accepted, maintaining the system reliability over time, even in the absence of model updates.

5 Evaluation

The present evaluation focuses on answering three research questions: ($Q1$) *Does the proposed RL technique aids at building ML models with longer lifespans?* ($Q2$) *What is the maximum period our proposal maintains its reliability without model updates?* ($Q3$) *Does the proposed verification technique provide reliable classifications over time, even in the absence of model updates?*

The next subsections describes our used dataset, how do we build the model and how does it perform in the used dataset.

5.1 A Year-long Intrusion Dataset

In general, proposed detection approaches for NIDS do not evaluate accuracy over time of their proposed techniques. This choice is justified by the fact that proposed techniques often rely on outdated datasets, with unrealistic network traffic behavior, which also does not take into account the time of occurrence of its events. As a consequence, authors often assume that periodic model updates are performed.

In light of this, to build a proper intrusion dataset, we built a dataset through the Samplepoint-F from the MAWI archive, which consists of actual network traffic collected daily for an interval of fifteen minutes from a transit link between Japan and the USA. The network traffic of 2016 was used to evaluate the model lifespan and accuracy degradation of ML-based NIDS. The built dataset comprises more than 10TB of data, from around 300 billion network packets. We employed an unsupervised ML technique from MAWILab [22] to automatically label the input records, i.e., to label events either as normal or attack. MAWILab employs several unsupervised machine learning algorithms to find anomalies in MAWI data without individual event labeling. The found anomalies are tagged as attacks, while the remaining data are assumed to be normal events. The feature extraction algorithm grouped events in intervals of fifteen seconds while extracting the 20 flow-based features from Nigel [23] work.

5.2 Model Building

The proposed RL-based technique to build the model is evaluated using a multilayer perceptron (MLP) with 250 neurons. The chosen MLP layout enables the proper comparison between our proposal and the traditional approach. The RL technique relies on a MLP, trained with 500 epochs, a learning rate of 0.3, a momentum rate for the backpropagation algorithm of 0.2, as implemented through the TensorFlow API [24]. The traditional approach also relies on the same MLP configuration. Each classifier was evaluated for their true-negative (TN) and true-positive (TP) rates. The TN rate denotes the ratio of normal events correctly classified as normal.

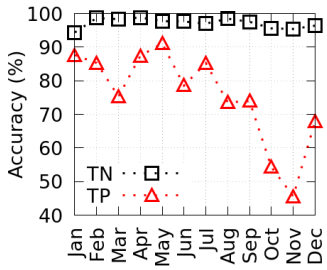

Fig. 2: Traditional approach performance through the year, without updates (MLP (250 neurons)).

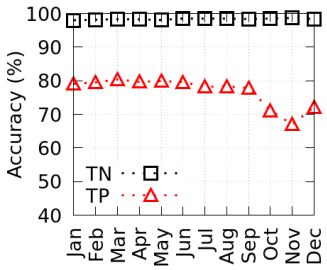

Fig. 3: Long-lasting model performance through the year, without updates (MLP (250 neurons)).

In contrast, the TP denotes the ratio of attack events correctly deemed as an attack. The proposed modified Q-Learning algorithm was implemented on top of OpenAI Gym API [26]. The algorithm executes 10 thousand iterations to build the model, in which each iteration performs 100 turns. Each turn computes the Q-Learning policy gradients according to the obtained rewards (see Eq. 1) from the classification of 1000 training instances. These parameters were identified empirically, and varying them result in similar classification results.

To build the ML models, due to the imbalanced nature of the dataset (only $\sim 2\%$ of instances are classified as attacks), a random undersampling without replacement was performed in the training data. Hence, the data distribution used for training purposes was equally distributed between the classes. To properly evaluate the network traffic behavior change impact on ML models, the classifier was trained through the first dataset month (January 2016), while being evaluated throughout the remaining time without model updates.

5.3 Model Evaluation

Consequently, we apply the model built without and with our technique (Section 5.2) throughout the whole year to answer Question $Q1$. Figure 2 shows the accuracy behavior over time of the traditional approach, through the MLP with 250 neurons, while Figure 3 shows the obtained accuracy with our RL-based technique without updates using the same underlying classifier parameters. It is possible to note a significant improvement in its reliability over time. Our proposed approach maintains the accuracy obtained at the training period for longer periods, hence, increasing the model lifespan. Regarding the worst accuracy decrease, occurred in November, the TP rate decreased by only 13%, a significant improvement when compared to the traditional approach (which decreased by 42% in the same period).

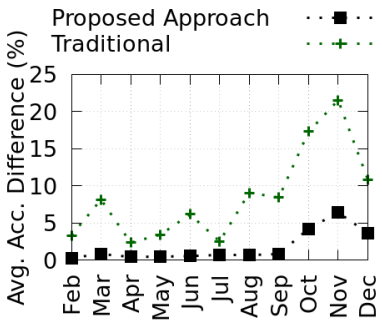

Fig. 4: Average monthly accuracy difference to January throughout time (lower is better), without *Verifier* module.

We consider a scenario in which a 5% of accuracy decrease is tolerated when addressing Question $Q2$. In such a case, our model remains reliable until October, when the TP rate decreases by 8%. In contrast, the traditional model building approach remains reliable only until March, when the TP rate decreases by 12%. Figure 4 details the average accuracy difference, when compared to January, from both model building approaches. In the figure, the proposed RL-based approach significantly improves the reliability in the absence of model updates, showing a small accuracy variation over time.

5.4 Verification

We evaluate the error-reject performance tradeoff when using our proposed *Verification* technique (Section 4.2) to address Question $Q3$. We find the optimal class rejection threshold for the RL-based model by applying the Class-Related-Threshold (CRT) [16] technique. In the first evaluation, we only considered the data from the training period. This choice is justified by the fact that the administrator only has training data to define the rejection thresholds. Finally, we select a rejection operation point at 10% rejection rate, and apply it throughout the year. The operation point is selected according to the administrator's discretion. The goal is to evaluate if the proposed *Verification* module maintains the system reliability over time.

Figure 5 shows the obtained accuracy and rejection rate throughout the year when the *Verifier* module is used. The proposed technique maintains the system reliability over time, despite the rejection rate, even without model updates.

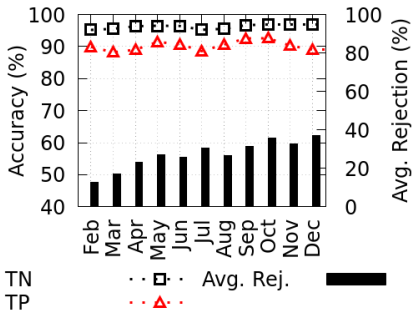

Fig. 5: Long-lasting model performance through the year, without updates, with *Verifier* module.

6 Conclusion

Current approaches for network-based intrusion detection are unable to deal with changes in the network traffic behavior over time. In general, authors from the literature often assume that periodic model updates are performed, ignoring the challenges it introduces. In this work, we have proposed a novel long-lasting intrusion detection model based on reinforcement learning. Our proposed scheme significantly improved the underlying ML model reliability over time, maintaining their accuracy for longer periods. Additionally, through our verification technique, our proposal improved its accuracy, even in the absence of model updates. In the future, we intend to pursue the reduction of the rejection rate using autonomous, periodic model updates coped with our RL-based approach.

Acknowledgments

Authors thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for partial financial support (grant 430972/2018-0 and 315322/2018-7) and the FCT through the LASIGE Research Unit (ref. UIDB/00408/2020).

References

1. Varonis. Available online: https://www.varonis.com/blog/cybersecurity-statistics/ *Accessed 10 december 2019*
2. Kaspersky. Available online: https://kaspersky.com/ *Accessed 10 december 2019*
3. Threatpost Windows Flaw. Available online: https://threatpost.com/windows-uac-flaw-privilege-escalation/150463/ *Accessed 10 december 2019*
4. Abreu, V., Santin, A.O., Viegas, E.K., Stihler, M. (2017). A multi-domain role activation model. *IEEE Int. Conf. Commun. (ICC) 3–8.*

5. Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches *Peer-to-Peer Networking and Applications - March 2019, Volume 12, Issue 2, pp. 493–501*
6. Singh, P. B., Chugh, U., Kathuria, M. (2018). A Review on Intrusion Detection System. *International Research Journal of Engineering and Technology (IRJET)*
7. Thaseen, S., Kumar, A. (2013). An Analysis of Supervised Tree Based Classifiers for Intrusion Detection System. *Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME)*
8. Farnaaz, N., Jabbar, M. A. (2016). Random Forest Modeling for Network Intrusion Detection System. *Twelfth International Multi-Conference on Information Processing*
9. E., Viegas, Santin, A. , Bessan, A., Neves, N. (2019). BigFlow: Real-time and Reliable Anomaly-based Intrusion Detection for High-Speed Networks *Future Generation Computer Systems Volume 93, April 2019, Pages 473-485*
10. Caminero, G., Lopez-Martin, M., Carro (2019).Adversarial environment reinforcement learning algorithm for intrusion detection *Computer Networks - Volume 159, 4 August 2019, Pages 96-109*
11. Bom, L., Henken, R., Wiering, M. (2013). Reinforcement learning to train Ms. Pac-Man using higher-order action-relative input. *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*
12. Pan, X., You, Y., Wang, Z., Lu, C. (2017). Virtual to Real Reinforcement Learning for Autonomous Driving. *arXiv, arXiv:1704.03952*
13. Pilarski, P. M., Dawson, M. R., Degris., T., Fahimi, F., Carey J. P., Sutton, R. S. (2011). Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. *IEEE International Conference on Rehabilitation Robotics*
14. Nanda, N. B., Parikh, A. (2019). Hybrid Approach for Network Intrusion Detection System Using Random Forest Classifier and Rough Set Theory for Rules Generation. *International Conference on Advanced Informatics for Computing Research - ICAICR 2019*
15. Sukumar, J. V. A., Pranav, I., Neetish, MM, Narayanan, J. (2018). Network Intrusion Detection Using Improved Genetic k-means Algorithm. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*
16. Viegas, E., Santin, A., Oliveira, L., França, A., Jasinski, R., Pedroni, V. (2019). A reliable and energy-efficient classifier combination scheme for intrusion detection in embedded systems. *Computers Security Volume 78, Pages 16–32*
17. Darkaie, M., Tavoli, R. (2019). Providing a method to reduce the false alarm rate in network intrusion detection systems using the multilayer perceptron technique and backpropagation algorithm. *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*
18. Xu, C., Shen, J., Du, X., Zhang, F. (2019). An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units. *IEEE - Volume: 6, pp. 48697-48707*
19. Viegas, E., Santin, A.O., Franca, A., Jasinski, R., Pedroni, V.A., Oliveira, L.S.. (2017) Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. *IEEE Trans. Comput. Volume 66, Pages 163-177*
20. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A. (2020).Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems With Applications 141 (2020) 112963*
21. Viegas, E., Santin, A.O., Abreu V., Oliveira, L.S.. (2018) *Enabling Anomaly-based Intrusion Detection Through Model Generalization. IEEE Symposium on Computers and Communications (ISCC) Pages 934-939*
22. Mawilab. Available online: http://www.fukuda-lab.org/mawilab *Accessed 10 december 2019*
23. Mawi. Available online: http://mawi.wide.ad.jp/mawi/ *Accessed 10 december 2019*
24. Tensorflow. Available online: https://www.tensorflow.org/ *Accessed 10 december 2019*
25. Dayan, P. (1992). Q-Learning. *Machine Learning, 8, 279-292*
26. OpenAI. Available online: https://gym.openai.com/ *Accessed 10 december 2019*