DAHID: Domain Adaptive Host-based Intrusion Detection

Oluwagbemiga Ajayi
University of Maryland Baltimore County
Baltimore, Maryland 21250
Email: dk60890@umbc.edu

Aryya Gangopadhyay
University of Maryland Baltimore County
Baltimore, Maryland 21250
Email: gangopad@umbc.edu

***Abstract*—Cybersecurity is becoming increasingly important with the explosion of attack surfaces as more cyber-physical systems are being deployed. It is impractical to create models with acceptable performance for every single computing infrastructure and the various attack scenarios due to the cost of collecting labeled data and training models. Hence it is important to be able to develop models that can take advantage of knowledge available in an attack source domain to improve performance in a target domain with little domain specific data.**

In this work we proposed Domain Adaptive Host-based Intrusion Detection DAHID; an approach for detecting attacks in multiple domains for cybersecurity. Specifically, we implemented a deep learning model which utilizes a substantially smaller amount of target domain data for host-based intrusion detection.

In our experiments, we used two datasets from Australian Defense Force Academy; ADFA-WD as the source domain and ADFA-WD:SAA as the target domain datasets. We recorded a significant improvement in Area Under Curve AUC from 83% to 91%, when we fine-tuned a deep learning model trained on ADFA-WD with as little as 20% of ADFA-WD:SAA. Our result shows transfer learning can help to alleviate the need of huge domain specific dataset in building host-based intrusion detection models.

I. INTRODUCTION

According to a research report prepared by Cybersecurity Ventures [19], the annual cost of cybercrime damages as of 2015 was $3 trillion. This report also predicted that the annual cost would double by 2021. In [6], statistics of losses due to cyber attack incidents within the last decade was reported where incidents claiming $1 Million or more in loss increased from 21 in 2009 to 105 in 2019. This shows that not only has rate of attacks increased but they also cost more. Some of the recent attacks affecting many people include the Yahoo hack of 2013 which has been recalculated to have impacted 3 billion user accounts, and Equifax breach of 2017 that affected over 145 million customers [19]. Very recently, the SolarWind Hack was reported [3]. It was an unusual hack that affected US Government departments such as Homeland Security and Treasury and Commerce. Evaluation of the full impact of this attack is still on going.

There have been some improvements over the years in the area of intrusion prevention and detection, but the cybersecurity problems have become even more important as newly introduced network devices such as Internet of Things, come with increased attack surfaces [4]. This situation brings rise to zero day attacks that existing system may find hard to detect.

978-1-7281-5684-2/20/$31.00 ©2021 IEEE

Attacks can be either insider or outsider attacks. Firewall has been instrumental in preventing outsider attacks but has no effect on insider attacks [18]. To mitigate insider attacks, both Intrusion Prevention Systems (IPS) and Intrusion Detection Systems (IDS) are useful. According to [17], "An IDS is a type of security tool that monitors network traffic and scans the system for suspicious activities and alerts the system or network administrator". IPS on the other hand is described as a system that can autonomously block attacks before they occur [24]. IDS are mainly of two types: Host-based Intrusion Detection Systems (HIDS) which monitors individual hosts and alerts the user in the event of suspicious activities and Network Intrusion Detection Systems (NIDS) which stands at network points to identify intrusions in the network traffic [17], [13]. [8], [25] are early works that tried to define the types of HIDS. [25] identified Audit data OS-level HIDS and Audit data Application-level HIDS. Audit data OS-level HIDS relates to system calls, file system modifications and user logons. Recent works including [1], [16], [11] are system calls based.

Traditional Machine Learning approaches have been used extensively in automating the intrusive behavior and attack detection process including Artificial Neural Network ANN, Decision Tree DT, and Support Vector Machine SVM [17], [13], [15], but these algorithms require good feature engineering in order to achieve good performance and they don't benefit much when data increases beyond a certain point. Deep Learning techniques are able to address the problems with traditional machine learning algorithms as they are well known for automatic feature extraction and capability to cope and in fact, benefit immensely from large scale data in terms of performance [14], [2].

However, both traditional machine learning and deep learning algorithms rely heavily on an assumption that training data (source domain) and test data (target domain) not only belong to the same feature space, but also share the same distribution [10]. This assumption hardly holds for many real-life problems. It is also important to have sufficient training data in order to achieve good performance, especially with deep learning models. These constraints very often lead to poor performance of machine learning algorithms in solving such problems. Hence there is a need to create some sort of agreement between source domain and target domain data such that learning in source can be useful for training in target. Transfer learning can help in this situation.

Transfer learning is an approach that attempts to improve the performance of a target domain model by transferring the knowledge gained from a different but related source domain model in such a way as to reduce the dependence on a large quantity of target domain data in building target domain model. Transfer learning can be grouped into two main categories: homogeneous and heterogeneous transfer learning [7]. Homogeneous transfer learning handles situations where the domains are of the same feature space but only differ in marginal distribution. Heterogeneous transfer learning on the other hand handles situations where the domains have different feature spaces. Apart from having different set of features making up the features spaces of domains, the number of features could also be different.

Our goal in this work is to explore the possibilities of building better performing models for detecting attacks in a target domain (usually a low resource domain) using help from a source domain (usually a high resource domain). Our focus here is on Host-based Intrusion Detection Systems HIDS. The rest of the paper is organized as follows: in Section 2 we discuss the related work in this area. Section 3 describes our methodology. In Section 4 we present the experimental results and lastly in section 5 we draw our conclusions.

II. RELATED WORK

This section presents the existing work in the area of intrusion detection using transfer learning approach. More work has been done for Network Intrusion Detection systems so we will discuss some of these works in NIDS scenarios of homogeneous features (section II-A) and heterogeneous features (section II-B). As there are presently no work for domain adaptive HIDS, we would also discuss current achievements in HIDS from machine learning standpoint in II-C

A. Domain Adaptive NIDS with Homogeneous Features

Some of the works found in this area have taken the approach of converting NIDS dataset into image and treating the problem as an image classification one using Convolutional Neural Networks CNN. Xu et al. [28] in their work used a source domain KDD Cup 99 dataset and a target domain ”corrected” KDD Cup 99 dataset (which has 17 intrusion types not found in the source dataset) for their experiment. As the two datasets used in their experiment have the same set of features and only differ in the types of intrusion, this is clearly an homogeneous features scenario with source task different than target task: ($\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$). In their method, they started with preprocessing where the goal was to convert the 119 features into a 11 X 11 pixels grayscale image. For the training using the source dataset, 150,000 samples was set aside while each of validation and test got 10,000 samples each. This was fed into CNN model as they evaluated different values of hyperparameters to get the optimal. Their result recorded 97.9% accuracy outperforming other methods like SVM, DT, K-nearest Neighbor KNN and Long Short-term Memory LSTM.

Adaptation to the target domain was done by fine-tuning the source model using data from the target domain. Their result shows that test performance on target domain data improved after fine-tuning especially when larger potion of the test domain data was used in evaluation. The limitation of this work however lies in their choice of benchmark dataset - KDD 99 which has been criticized by [9] among others, for it’s age, highly skewed target, pattern redundancy, and irrelevant features among other issues. It would be interesting to see how this method performs on a dataset like NSL-KDD which has been created specifically to address problems with KDD Cup 99 dataset.

Another work by Gangopadhyay et al. [10] took a similar approach. In their work, image-based representation of the feature set was done on CICIDS2017 dataset from Canadian Institute for Cybersecurity [21]; mimicking 50 X 5 X 3 RGB image. Portscan attack type was taken as the source domain data while each of the other four attack represents a target domain data. Source data was used to train a model by feeding it into a four-layered CNN architecture using A batch size of 32, Stochastic Gradient Descent SGD optimizer and 100 epochs. It was then tested with 50% validation split achieving over 95% accuracy and a loss around 0.3. To transfer the learning from source domain to target domain, the model was re-used on other types of attack with minimal additional learning by adding a dense layer with 32 units. This resulted in validation accuracies of 100% for both DDoS and Infiltration attacks and about 95% for Botnet and Web attacks.

In contrast to [28], a more recent dataset with attacks relevant to our time has been used by [10]. The work however revealed little details on their method for the choice of hyperparameters.

B. Domain Adaptive NIDS with Heterogeneous Features

The problem of transfer learning becomes more complicated with the heterogeneous features scenario as there is a need to unify the feature space of the source and target. Wu et al. [27] in their work, applied CNN to heterogeneous feature scenario of network intrusion detection by concatenating CNN source (base) model and CNN target model which then adds a fully-connected layer as the ouput. Their experiment is based on source or base dataset, UNSW-NB15 of Australian Centre for Cyber Security (ACCS) [20] and NSL-KDD dataset of Canadian Institute for Cybersecurity [23]. These two datasets have different set of features so clearly this is an heterogeneous feature scenario. The set of attacks are also not the same between source and target domain data. A dedicated NSL-KDD test dataset (KDDTest+ and KDDTest-21) were used in evaluation which has 17 attacks not present in the train dataset. To unite the feature space of source and target domain, this work did a preprocessing that resulted in each instance in the two datasets becoming a vector of 113 values. They claimed to have combined the data fields of the UNSW-NB15 and NSL-KDD datasets.

Their result shows that the method recorded an accuracy of 87.30% on the KDDTest+ dataset versus 81.94% on the KDDTest-21 dataset. This is an improvement over the ordinary ConvNet without transfer learning where they recorded an accuracy of 84% on KDDTest+ dataset versus 59.92% on KDDTest-21 dataset.

C. HIDS Literature

Based on literature search, we could not find any work that has attempted domain adaptation in HIDS, neither in homogeneous nor heterogeneous scenario. It is however important to note the recent improvements recorded in HIDS with non-transfer learning approaches. [26] in their experiment on ADFA-WD dataset compared models of deep neural network DNN with SVM. Their result showed that 5 layered DNN with keras embedding recorded an accuracy of 83%, outperforming SVM with TFIDF which recorded an accuracy of 80.1%. One clear drawback of [26] is with their choice of accuracy as performance metric for ADFA-WD which is an imbalanced dataset.

[29] in their work compared performance of SVM and random forest in building a host-based intrusion detection system. For experiment they used ADFA-WD and ADFA-WD:SAA datasets. Their result shows that for both ADFA-WD and ADFA-WD:SAA, Random Forest recorded 82% detection rate outperforming SVM Sigmoid kernel at 71% and Radial Basis Function RBF kernel at 68%. Performance of these models can be improved with a domain adaptive approach.

In this work we investigated the possibility of building a better performing model in one domain of Host Based Intrusion Detection System HIDS by leveraging on the learning from a different but similar domain. We made the following contributions:

- Development of a new domain adaptive approach for Host Based Intrusion Detection.
- Mitigating need of large domain-specific data in building intrusion detection models with good performance for a low resource host domain.
- Alleviating the pains of zero-day attacks against host infrastructures.

III. METHODOLOGY

In this paper, we have assumed that our type of transfer learning problem is homogeneous (both source domain $\mathcal{D}_S$ and target domain $\mathcal{D}_T$ of HIDS have the same set of features). We have also assumed that both domains are labeled. The machine learning task in this work is that of classification of Host sequential process trace as either benign or attacks. Figure 1 presents the general process flow of DAHID.

System calls-based HIDS datasets being sequence of token have been approached in other works in a Natural Language Processing NLP fashion [26]. We have also followed the NLP approach but in a transfer learning way stepping through the process as laid out in Figure 1. This process presents 4 activities: Pre-processing, Network architecture, Evaluation and Fine tuning. We have in this work used the process to address the homogeneous features scenario of domain adaptive HIDS but this process can also work for the heterogeneous features scenario.

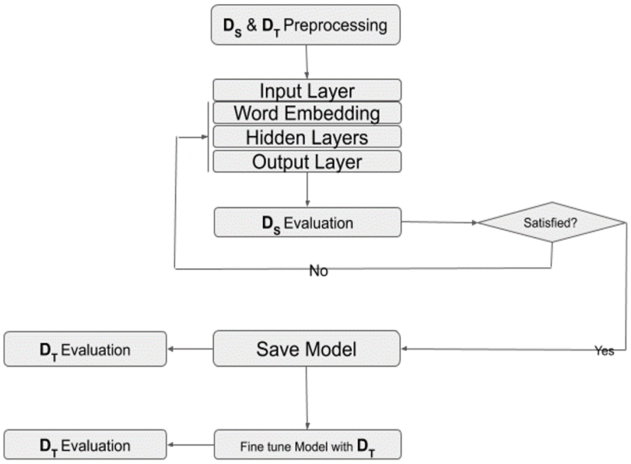


Fig. 1. General Flow of Domain Adaptation Method

A. Pre-processing

Preprocessing involved converting system calls into tokens, feature extraction and sequence padding. We considered Host-based Intrusion Detection using ADFA-WD and ADFA-WD:SAA datasets generated at the next generation cyber range infrastructure of the Australian Centre of Cyber Security (ACCS) in the University of New South Wale (UNSW) at Australian Defense Force Academy(ADFA), Canberra [12].

Both ADFA-WD and ADFA-WD:SAA contain traces of syscalls from Windows XP SP2 system with ADFA-WD acknowledging "Windows-based vulnerability-oriented zero-day attacks" while ADFA-WD:SAA is designed to test the effectiveness of HIDS against "Windows-based stealth attacks". These datasets contain nine core DLL calls (ntdll.dll, kernel32.dll, user32.dll, comctl32.dll, ws2_32.dll, mswsock.dll, msvcrt.dll, msvcpp.dll, and ntoskrnl.dll) [12].

Algorithm 1: Tokenize

```
Input: list of syscall's trace files $Trace\_files1$
Input: dictionary of syscall-integer pair $DLL\_dic$
Result: Tokenized trace files $Trace\_files2$
for file $\in Trace\_files1$ do
    for line $\in file$ do
        for syscall, integer $\in DLL\_dic.items()$ do
            Replace syscall with integer in line;
            Write to Trace_files2;
        end
    end
end
```

To tokenize these, a dictionary that maps each of the DLL to integers was created. Algorithm 1 presents steps for conversion of system calls into token. In order to represent the data in a form acceptable to the modeling algorithm, feature extraction will be needed.To keep things simple we are going to use n-gram setting n = 1.

There are 4 dataset files folder, see the breakdown in Table I

TABLE I
ADFA-WD AND ADFA-WD:SAA BREAKDOWN

	Trace File Count	Max Length	Median of Length
ADFA-WD			
Training	355	1670685	4208
Validation	1827	3212884	6834.0
Attack	5542	225973	1840.0
ADFA-WD:SAA			
Attack	862	200664	2032.5

With the median length of both attack datasets at about 2000, we're going to zero pad trace files with length less than 2000 and cut down to 2000 trace files with greater length. This is to ensure our trace files are of the same length. Training and Validation are combined and taken to be the negative (benign cases) while Attacks are positive.

From Table I we can see that there are 5542 attack instances in ADFA-WD while ADFA-WD-SAA has only 862 attack instances. Clearly, ADFA-WD is our high resource domain and so we are going to make it our source domain and ADFA-WD:SAA our target domain.

B. Network Architecture

In Figure 1, Input layer accepts $\mathcal{D}_S$ while choice of hidden layers is based on result of hyperparameter tuning. Choice of word embedding is based on whether we are solving a homogeneous or heterogeneous transfer learning problem: monolingual word embedding if homogeneous and bilingual word embedding if heterogeneous. Choice of output layer is based on whether the problem is binary or multi-class: sigmoid activation if binary and softmax if multi-class.

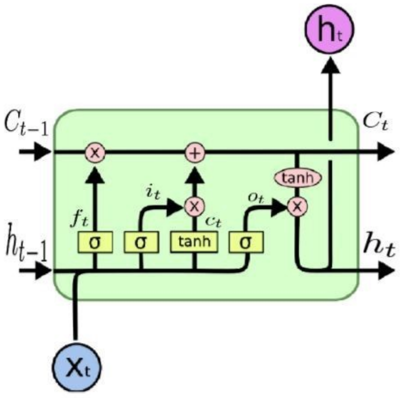


Fig. 2. LSTM Architecture Cell Visual Representation [22]

The appropriate deep learning architecture for sequence data is Recurrent Neural Network (RNN) but there are different variants of RNN. LSTMs are type of Recurrent Neural Network (RNN) architecture of deep learning with capabilities of learning long-term dependencies. Generally, all RNN models are designed to handle time or sequence dependence such as language, stock prices etc. While models such as CNN are feed-forward neural networks, LSTM has a feedback connection. In place of neural network layers, LSTM network presents LSTM cell blocks comprising of input gate, forge gate and output gate as components. Figure 2 is the LSTM Architecture Cell Visual Representation as presented in [22].

With our deep learning network set up we can build and evaluate a $\mathcal{D}_S$ model.

C. Evaluation

We then evaluate the model on $\mathcal{D}_T$ (this is expected to suffer performance degradation as evaluation is on a foreign domain). Lastly we use increasing portions of $\mathcal{D}_T$ to fine tune the $\mathcal{D}_S$ model as we evaluate using $\mathcal{D}_T$.

The most common performance metric is the Accuracy which estimates the ratio of correctly classified instances to the entire sample size.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP = True Positive , TN = True Negative , FP = False Positive and FN = False Negative.

However, accuracy only serves as a good metric for samples that contains balanced classes. One of the metrics that can help when we have class imbalance problem is Area Under Curve AUC of Receiver Operating Characteristics (ROC) curve [5].

$$AUC = \int_0^1 \frac{TP}{TP + FN} \, d\frac{FP}{TN + FP}$$

.

D. Fine-tuning

Fine-tuning could be done by either freezing some hidden layers or not. Freezing helps in reducing the training time and allows the model to concentrate on layers that capture the differences between the two domains. The only advantage of not freezing any layer is that no assumption is being made hence every layer that could benefit from fine-tuning no matter how small will be available.

TABLE II
ADFA-WD AND ADFA-WD:SAA TRAIN/TEST SAMPLE SIZES

Source (ADFA-WD)		
Sample	**Positive (5542)**	**Negative (2182)**
Train	37% (2051)	92% (2008)
Test	62% (3491)	8% (174)

Target (ADFA-WD:SAA)		
Sample	**Positive (862)**	**Negative (2182)**
Train	84% (725)	33% (721)
Test	16% (137)	67% (1461)

IV. EXPERIMENTAL RESULT

A. *Data*

As we have disproportionate number of records of negative and positive examples, to guard against the problem of class imbalance in our training samples we used resampling to create Training and Test samples as seen in Table II

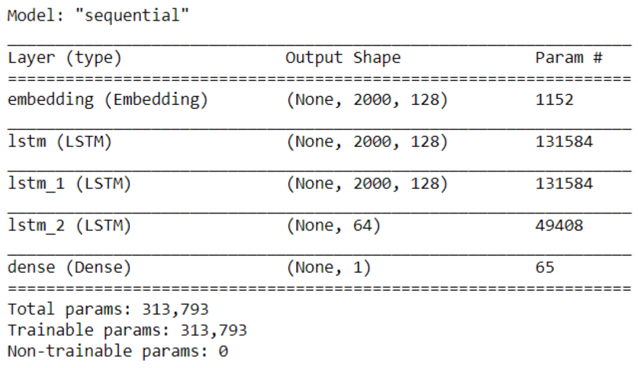
B. *Deep Learning Architecture*

The network as seen in figure 2 accepts 3 input items: h_{t-1} (output from the previous LSTM cell), X_t (input at the present time) and C_{t-1} (memory from the previous LSTM cell). While it returns 2 output items: C_t (memory of the present LSTM cell) and h_t (output from the present LSTM cell).

With choice of deep learning architecture made, using manual tuning method we have selected the following hyper-paramater values:

- Embedding dimension: 128
- Sequence Maximum Length: 2000
- number of LSTM layers: 3
- number of units: First 2 hidden layers (128 units), last one (64 units)
- dropout: 0.2
- batch size: 10
- Number of Epochs: 100

Figure 3 presents the network architecture.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 2000, 128)         1152
_________________________________________________________________
lstm (LSTM)                  (None, 2000, 128)         131584
_________________________________________________________________
lstm_1 (LSTM)                (None, 2000, 128)         131584
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 1)                 65
=================================================================
Total params: 313,793
Trainable params: 313,793
Non-trainable params: 0
```

Fig. 3. Network Architecture

In the first step of our experiment, a model was trained using the train sample of source domain ADFA-WD dataset, with a validation split of 0.2. See the plots of accuracy and loss of the model as epoch increased in Figure 4.

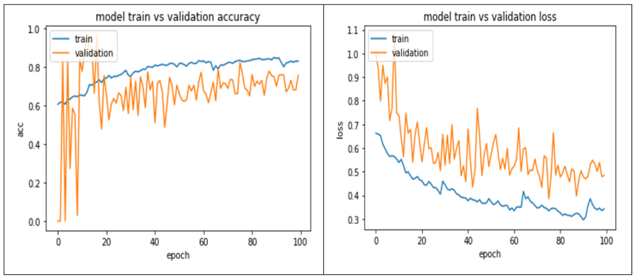


Fig. 4. Accuracy/Loss Plot

The model was evaluated using the test sample of source domain ADFA-WD dataset. To see the impact of performance degradation across domain, evaluation using the test sample of target domain ADFA-WD:SAA dataset was carried out. See below the performance of the source domain model in Table III

TABLE III
SOURCE DOMAIN MODEL PERFORMANCE

	Loss	Accuracy	AUC
Source Train	0.3419	0.8317	0.9188
Source Test	0.477	0.759	0.912
Target Test	0.313	0.851	0.831

Looking at the target test performance in III, we can see the drop in AUC from 91.8% to 83.1% (because of the class imbalance, accuracy cannot be a reliable performance metric).

In the second step of our experiment, we attempted to fine-tune the model (without freezing any layer) created using ADFA-WD with some portion of ADFA-WD:SAA. We would like to see if performance of target test improves. See below the performance as the amount of target train sample used in fine-tuning increased.

TABLE IV
FINE-TUNING USING TARGET DOMAIN MODEL PERFORMANCE

Portion of ADFA-WD:SAA	Loss	Accuracy	AUC
10%	0.7388	0.7837	0.8886
20%	0.6133	0.7781	0.9151
30%	0.4751	0.7444	0.9184
40%	0.4951	0.8006	0.9126
50%	0.3973	0.7931	0.9205
60%	0.4269	0.8118	0.9222
70%	0.4765	0.8306	0.9275
80%	0.3667	0.8125	0.9213
90%	0.3667	0.8125	0.9272

It can be observed in table IV that fine-tuning with just 10% portion of ADFA:SAA train sample resulted in 88.86% of AUC. This improvement continued as more portion of ADFA:SAA is used in fine-tuning. Figure 5 presents a plot of Performance (AUC) against percentage of ADFA:SAA train sample used in fine-tuning.

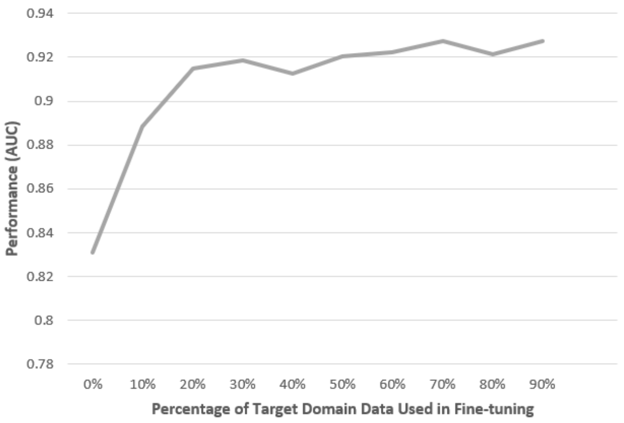


Fig. 5. Amount of Data Used in Fine-tuning Vs Performance Plot

V. CONCLUSION

In this paper we have demonstrated a new approach for domain adaptation for deep learning in cybersecurity specifically, homogeneous feature scenario of Host-based IDS. While transfer learning has been applied to many applications including a few in cybersecurity domain the need remains critical because of lack of availability of data, let alone data with labels.

We ran experiments on source ADFA-WD and target ADFA-WD:SAA and the results show that it is possible to develop transfer learning models in detecting different types of cyber-attacks with little fine-tuning on the target domain.

In the future we will explore the possibilities of success in heterogeneous feature scenario of Host-based IDS. Lastly, we would be investigating how useful our findings would be for attack detection in IoT security.

ACKNOWLEDGMENT

This research is partially supported by NSF grant number 1923982, “HDR DSC: Collaborative Research: Creating and Integrating Data Science Corps to Improve the Quality of Life in Urban Areas”.

REFERENCES

[1] E. Aghaei, ”Machine Learning for Host-based Misuse and Anomaly Detection in UNIX Environment”, University of Toledo, 2017.

[2] E. Aminanto and K. Kim, ”Deep learning in intrusion detection system: An overview”, in 2016 International Research Conference on Engineering and Technology (2016 IRCET), 2016.

[3] ”Audacity of SolarWinds hack will harden Western policy — Emerald Insight”, Emerald.com, 2021. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/OXAN-ES258311/full/html. [Accessed: 17- Mar- 2021].

[4] E. Benkhelifa, T. Welsh and W. Hamouda, ”A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems”, IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3496-3509, 2018. Available: 10.1109/comst.2018.2844742.

[5] A. Bradley, ”The use of the area under the ROC curve in the evaluation of machine learning algorithms”, Pattern Recognition, vol. 30, no. 7, pp. 1145-1159, 1997. Available: 10.1016/s0031-3203(96)00142-2.

[6] C. Crane, ”42 Cyber Attack Statistics by Year: A Look at the Last Decade — InfoSec Insights”, InfoSec Insights, 2020. [Online]. Available: https://sectigostore.com/blog/42-cyber-attack-statistics-by-year-a-look-at-the-last-decade/. [Accessed: 17- Mar- 2021].

[7] O. Day and T. Khoshgoftaar, ”A survey on heterogeneous transfer learning”, Journal of Big Data, vol. 4, no. 1, p. 29, 2017. Available: 10.1186/s40537-017-0089-0.

[8] P. De Boer and M. Pels, ”Host-based intrusion detection systems”, 2005.

[9] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole, ”Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives”, in 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), IEEE, 2018, pp. 1–8.

[10] A. Gangopadhyay, I. Odebode and Y. Yesha, ”A Domain Adaptation Technique for Deep Learning in Cybersecurity”, in OTM Confederated International Conferences” On the Move to Meaningful Internet Systems, 2019, pp. 221–228.

[11] M. Grimmer, M. Röhling, D. Kreusel and S. Ganz, ”A modern and sophisticated host based intrusion detection data set”, IT-Sicherheit als Voraussetzung für eine erfolgreiche Digitalisierung, pp. 135–145, 2019. [Accessed 17 March 2021].

[12] W. Haider, G. Creech, Y. Xie and J. Hu, ”Windows Based Data Sets for Evaluation of Robustness of Host Based Intrusion Detection Systems (IDS) to Zero-Day and Stealth Attacks”, Future Internet, vol. 8, no. 4, p. 29, 2016. Available: 10.3390/fi8030029.

[13] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, ”Survey of intrusion detection systems: techniques, datasets and challenges”, Cybersecurity, vol. 2, no. 1, p. 20, 2019. Available: 10.1186/s42400-019-0038-7.

[14] K. Kim and M. Aminanto, ”Deep learning in intrusion detection perspective: Overview and further challenges”, in 2017 International Workshop on Big Data and Information Security (IWBIS), IEEE, 2017, pp. 5–10.

[15] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai and K. Dai, ”An efficient intrusion detection system based on support vector machines and gradually feature removal method”, Expert Systems with Applications, vol. 39, no. 1, pp. 424-430, 2012. Available: 10.1016/j.eswa.2011.07.032.

[16] M. Liu, Z. Xue, X. Xu, C. Zhong and J. Chen, ”Host-Based Intrusion Detection System with System Calls”, ACM Computing Surveys, vol. 51, no. 5, pp. 1-36, 2018. Available: 10.1145/3214304.

[17] P. Mishra, V. Varadharajan, U. Tupakula and E. Pilli, ”A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection”, IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 686-728, 2019. Available: 10.1109/comst.2018.2847722.

[18] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel and M. Rajarajan, ”A survey of intrusion detection techniques in Cloud”, Journal of Network and Computer Applications, vol. 36, no. 1, pp. 42-57, 2013. Available: 10.1016/j.jnca.2012.05.003 [Accessed 17 March 2021].

[19] S. Morgan, ”Cybercrime Damages $6 Trillion by 2021”, Cybercrime Magazine, 2017. [Online]. Available: https://cybersecurityventures.com/annual-cybercrime-report-2017/. [Accessed: 17- Mar- 2021].

[20] N. Moustafa and J. Slay, ”UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”, 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6, 2015. Available: 10.1109/milcis.2015.7348942 [Accessed 17 March 2021].

[21] I. Sharafaldin, A. Habibi Lashkari and A. Ghorbani, ”Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, Proceedings of the 4th International Conference on Information Systems Security and Privacy, pp. 108–116, 2018. Available: 10.5220/0006639801080116 [Accessed 17 March 2021].

[22] M. Soni, ”Understanding architecture of LSTM cell from scratch with code. — Hacker Noon”, Hackernoon.com, 2018. [Online]. Available: https://hackernoon.com/understanding-architecture-of-lstm-cell-from-scratch-with-code-8da40f0b71f4. [Accessed: 17- Mar- 2021].

[23] M. Tavallaee, E. Bagheri, W. Lu and A. Ghorbani, ”A detailed analysis of the KDD CUP 99 data set”, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6, 2009. Available: 10.1109/cisda.2009.5356528 [Accessed 17 March 2021].

[24] L. Tidjon, M. Frappier and A. Mammar, ”Intrusion Detection Systems: A Cross-Domain Overview”, IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3639-3681, 2019. Available: 10.1109/comst.2019.2922584.

[25] G. Vigna and C. Kruegel, Host-based intrusion detection. na, 2006.

[26] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, ”Deep Learning Approach for Intelligent Intrusion Detection System”, IEEE Access, vol. 7, pp. 41525-41550, 2019. Available: 10.1109/access.2019.2895334 [Accessed 17 March 2021].

[27] P. Wu, H. Guo and R. Buckland, ”A Transfer Learning Approach for Network Intrusion Detection”, 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA), pp. 281–285, 2019. Available: 10.1109/icbda.2019.8713213 [Accessed 17 March 2021].

[28] Y. Xu et al., ”Intrusion Detection Based on Fusing Deep Neural Networks and Transfer Learning”, Communications in Computer and Information Science, pp. 212-223, 2020. Available: 10.1007/978-981-15-3341-9_18 [Accessed 17 March 2021].

[29] C. Simon and I. Sochenkov, ”EVALUATING HOST-BASED INTRUSION DETECTION ON THE ADFA-WD AND ADFA-WD:SAA DATASETS”, Semanticscholar.org, 2021. [Online]. Available: https://www.semanticscholar.org/paper/EVALUATING-HOST-BASED-INTRUSION-DETECTION-ON-THE-Simon-Sochenkov/840891cbe180bd860b16d6d1d8af7a15c5a80db7. [Accessed: 07- Jun- 2021].