Mantik Journal, 7 (4) (2024), ISSN 2685-4236 (Online)

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik

Utilizing neural networks with CICIDS2018 dataset for detecting brute force attack anomalies in intrusion detection systems

Ahmad Heryanto*[1], Adi Hermansyah[2], Triwanda Septian[3], Ali Bardadi[4]
[1,2,3]Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya, Indonesia
[4]Jurusan Sistem Informasi, Fakultas Ilmu Komputer, Universitas Sriwijaya, Indonesia

A R T I C L E I N F O

Article history:

Received Feb 20, 2024
Revised Feb 23, 2024
Accepted Feb 26, 2024

Keywords:

Brute force;
Detection;
IDS;
Intrusion;
Neural Network.

ABSTRACT

In this study, the effectiveness of neural networks in Intrusion Detection Systems (IDS) has been tested using the CICIDS2018 dataset to achieve accurate intrusion detection results. The research findings reveal that several neural network parameters will reach optimal results with a learning rate of 0.1, a training and testing data proportion of 80:20, and an optimal number of nodes in the hidden layer of 4. Other parameters such as a minimum error of 0.0001 and 2500 iterations also play a crucial role in improving IDS capability. Based on the research, it is shown that neural network models can provide optimal results in detecting intrusion patterns. This study can assist in the development of reliable and efficient neural network-based IDS to address the challenges of intrusion detection.

Corresponding Author:

Ahmad Heryanto,
Sistem Komputer,
Universitas Sriwijaya,
Jl. Lintas timur km.32 Ogan Ilir, Sumatera Selatan, Indonesia.
Email: hery@unsri.ac.id

1. INTRODUCTION

Brute force attacks are a type of cyber attack aimed at gaining unauthorized access to protected systems or information. Attackers try all possible combinations of passwords, encryption keys, tokens, or identification numbers, hoping to find the correct combination(Amijoyo et al., 2020). Brute force attacks are considered simple yet highly effective, especially in the absence of effective protective measures. Despite their straightforward approach, they remain a serious threat to security systems. To address this threat, Intrusion Detection Systems (IDS) play a crucial role in detecting and issuing early warnings about suspicious activities, including brute force attacks(Sadasivam et al., 2016). IDS monitor network traffic and analyze behavioral patterns to identify signs of suspicious activity, such as repeated login attempts from the same location or IP address in a short period. By collaborating with other security systems, IDS can implement temporary access restrictions and provide appropriate warning or response actions. By detecting brute force attacks, IDS can help maintain the integrity of security systems and prevent unauthorized access. It is important to monitor network traffic and analyze behavioral patterns to detect signs of suspicious activity and take necessary actions to address these threats(Otoum & Nayak, 2021).

Intrusion Detection Systems (IDS) play a crucial role in detecting brute force attacks through two main approaches: signature-based detection and anomaly-based detection. Signature-based detection uses a database of patterns or signatures previously associated with brute force attacks(Guri et al., 2019; Liao et al., 2013; Sung et al., 2020). This approach involves comparing previously identified patterns or signatures with ongoing activities, such as multiple login attempts or a series of attempts with the same combination. On the other hand, anomaly-based detection focuses on identifying unusual or suspicious behavior using machine learning to understand common patterns of network traffic or system activities(Bhatia et al., 2020; Imrana et al., 2021). The integration of these two methods allows IDS to detect brute force attacks more effectively and adaptively. In this way, IDS is responsive to known attacks and can identify new threats that have not been previously detected, thereby creating a stronger and more adaptive layer of protection against evolving attack dynamics. As part of security strategy, IDS integration provides proactive defense against brute force attacks. With its ability to monitor anomalous activities and collaborate with other security systems, IDS helps protect systems and data from potential risks of brute force attacks. Success in handling these attacks includes rapid response actions, accurate pattern analysis, and implementation of adequate protection measures(Ananin et al., 2017; Bul'ajoul et al., 2013; Najafabadi et al., 2016).

Several studies have utilized Intrusion Detection Systems (IDS) by integrating artificial intelligence techniques such as Neural Networks. Neural Networks are machine learning algorithms inspired by the structure and function of the human nervous system. The use of Neural Networks has brought significant innovation in detecting and addressing security threats, especially when integrated with IDS(Hosseini & Zade, 2020). Neural Networks can learn from training data to understand complex patterns and infer information that may not be visible to traditional methods. In the case of IDS, Neural Networks can be utilized for intrusion detection in a smarter and more adaptive way. The main advantage of Neural Networks is their ability to recognize complex behavioral patterns and anomalies that may be difficult to identify with traditional detection methods. Thus, Neural Networks can improve accuracy in detecting attacks while reducing the number of false positives and false negatives. IDS enhanced with Neural Networks can automatically update their models based on emerging activity patterns, making them more responsive to evolving attacks.

Neural Networks effectively handle security attacks that have never occurred before or continue to evolve because they can learn from experience. This creates a smarter and more adaptive layer of defense in networks, allowing organizations to identify and address security threats more efficiently(Gamage & Samarabandu, 2020; Pawlicki et al., 2022). Neural Networks can recognize complex patterns and learn from data. The ability to understand complex patterns helps detect brute force attacks by identifying patterns of suspicious or unusual activity. Regarding information security, Neural Networks are highly beneficial because they provide quick and accurate responses to attacks. The research gap addressed in this study pertains to developing a brute-force attack detection system that optimizes neural network algorithms. Presently, the conventional approach in intrusion detection involves utilizing signature and anomaly techniques. The signature technique relies on a database of known attack patterns to identify attacks that match existing patterns. In contrast, the anomaly technique monitors system behavior and searches for unusual or non-conforming patterns to detect attacks. However, both approaches have limitations in detecting previously unseen or evolving attacks, such as brute force attacks with patterns differing from recognized attacks or unusual behavior that could evade anomaly detection. This is where the relevance of neural networks emerges. By leveraging neural network technology, intrusion detection systems can learn from experience and recognize complex and unexpected patterns, including brute force attacks that have not been previously

detected. Neural networks can adapt to changes in attack patterns without necessitating manual updates like signature techniques, and they can also identify abnormal behavior that escapes detection by traditional anomaly approaches. Therefore, this research aims to bridge the gap in existing research by developing a brute force attack detection system that capitalizes on the advantages of neural network technology, thereby enhancing intrusion detection capabilities in countering evolving and previously undetected attacks.Therefore, this research aims to develop a brute force attack detection system that utilizes artificial neural network algorithms to enhance effectiveness in identifying and addressing potential security threats that may arise in computer networks.

2. RESEARCH METHOD

2.1 Brute force attack

Brute force is a cyber attack in which the attacker attempts to gain access to a system or data by systematically trying all possible combinations of passwords or important encryptions until they find the correct combination(Hossain et al., 2020; Wang et al., 2023). This method assumes that the attacker will succeed if they try all possible combinations. Although this method is simple, it can be highly effective, especially if the target system requires better login attempt management. Here are some activities related to brute force attacks: a) Trying all possibilities: Attackers try all possible combinations of passwords or important encryptions until they find the correct combination. This includes testing all possible combinations of letters, numbers, and special characters; b) Time required: The main drawback of brute force attacks is the time required to try all combinations. However, increasing computational speed and using distributed technology can speed up this process, c) Automated usage: Brute force attacks are often implemented using scripts or automated tools that can quickly and continuously try combinations without human intervention, d) Primary targets: These attacks are typically aimed at passwords, encryption keys, or access codes that protect systems, applications, or data; e) Detection and prevention: Security systems generally come equipped with brute force detection and prevention mechanisms, such as limiting the number of login attempts within a certain time frame or requiring additional verification after a certain number of failed attempts; f) Importance of strong security: Using strong and complex passwords or encryption keys to combat brute force attacks and implementing effective security policies is crucial.

Brute force attacks highlight the importance of having strong security layers, including good password policies, the use of two-factor authentication, and monitoring suspicious login activities on systems(Najafabadi et al., 2016).

2.2 Intrusion Detection System (IDS)

Intrusion Detection System (IDS) is a critical component of information security designed to monitor and identify suspicious or potentially harmful activities within a network or computer system(Najafian et al., 2015; Shah & Issac, 2018). IDS functions as a device that monitors network traffic or system activities and then provides warnings or takes necessary actions if security threats are detected(Bharati & Tamane, 2017). Here are some key points regarding IDS: a) Monitoring Network or System Activities: IDS continuously monitors network traffic and system activities. This includes log analysis, packet data monitoring, and user behavior observation; b) Anomaly Detection and Signature Pattern: IDS uses two main detection methods. First, anomaly detection method involves monitoring routine activities and detecting significant changes that may indicate an attack. Second, signature pattern detection method utilizes a signature database containing known attack patterns; c) Prompt Warning and Response: IDS provides immediate warnings upon detecting suspicious activities or security threats.

Further actions may include issuing warnings to administrators, blocking access to the system, or taking other preventive measures; d) Security Layer Monitoring: IDS can be implemented across multiple security layers, including network, host, and application layers. This enables early detection and more effective response to various types of attacks; f) IDS Categories: NIDS and HIDS: There are two main categories of IDS - Network-based Intrusion Detection System (NIDS), which focuses on monitoring network traffic, and Host-based Intrusion Detection System (HIDS), which focuses on system or host-level activities; g) Integration with Other Security Systems: IDS is often integrated with other security systems, such as firewalls and other security systems, to create a more comprehensive defense; h) Importance of Logging and Auditing: IDS creates logs that can be used for forensic analysis and security auditing. These logs can help administrators understand the source and nature of attacks and take appropriate preventive actions.

Intrusion Detection Systems (IDS) have two main types, namely Anomaly-based IDS and Signature-based IDS(Bostani & Sheikhan, 2017; Laghrissi et al., 2021; Pawlicki et al., 2022). Anomaly-based IDS functions by detecting unusual activities or deviations from normal patterns in network traffic or system behavior, by comparing them to a baseline of normal behavior(Alrawashdeh & Purdy, 2017). On the other hand, Signature-based IDS examines network traffic or system activity logs against a database of known attack signatures(Otoum & Nayak, 2021). While Anomaly-based IDS is capable of detecting previously unseen attacks, it is vulnerable to false positives. Conversely, Signature-based IDS is more effective at detecting known attacks but is ineffective against novel attacks or significantly modified ones. Combining both types of IDS can enhance detection rates by addressing their respective weaknesses. By leveraging IDS, organizations can enhance their ability to detect, identify, and respond to security threats more quickly and effectively, thus maintaining the security and integrity of their information systems(Chaudhary, 2015).

2.3 Neural network

Neural Network is a mathematical model inspired by the structure and function of the human brain(Kowsher et al., 2021). Like the brain, neural networks consist of interconnected neurons that form complex layers. This allows them to model patterns and learn abstract data representations. Neural networks are used in artificial intelligence applications such as pattern recognition, classification, and prediction. Neural networks can learn from data by adjusting the weights between neurons during training. This learning process typically uses algorithms such as backpropagation. This ability enables neural networks to recognize complex patterns in data and make predictions or decisions based on the learning they have acquired. Neural networks support various types of architectures, such as feedforward neural networks, recurrent neural networks (RNNs), and convolutional neural networks (CNNs), each optimized for specific tasks(Aziz Sharfuddin et al., 2018; Kiranyaz et al., 2020). Their use is not limited to the field of artificial intelligence but encompasses various industries such as healthcare, finance, and technology. Due to their increasing capabilities and ability to solve complex problems, neural networks continue to be the focus of research and innovation. They play a crucial role in advancing artificial intelligence technology and solving programmatically challenging problems. The architecture of neural networks can be illustrated in the following figure.

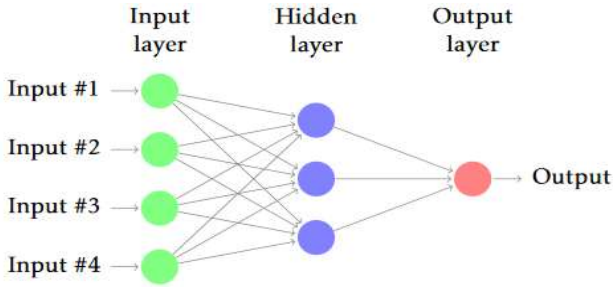

Figure 1. Neural Network Structure

The process of a neural network in recognizing a pattern of attack from input to output involves several stages. Input data is fed forward through the network, with each neuron performing a weighted sum of inputs and applying an activation function to produce an output. Connections between neurons have associated weights, and each neuron has a bias term to capture non-linear relationships in the data. The output layer of the network generates a prediction or classification based on the input data and learned parameters. A loss function quantifies the network's performance compared to the desired output. Backpropagation adjusts the weights and biases to minimize error, improving the network's ability to recognize attack patterns. Training involves iteratively adjusting parameters until satisfactory performance is achieved on labeled data. The network is then evaluated on unseen data to ensure accurate recognition of attack patterns. Finally, the trained network can be deployed for real-time intrusion detection in production environments(Gunjan, 2015).

2.4 IDS and neural network

Research method is a section of a research manuscript that explains the steps taken in the research, the rationale for sample selection, validation processes, and measurements conducted. This section involves preparing the initial configuration of the Neural Network. It initializes the weights and biases for each neuron in the network with random values. Proper initialization is crucial to enable the network to effectively learn during the training process.

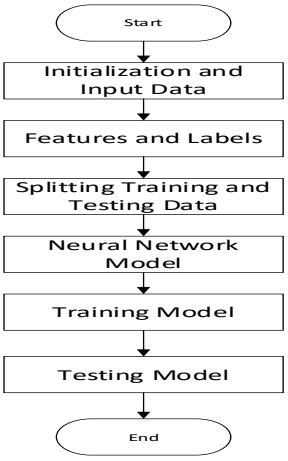

Figure 2. Neural Network Flowchart

```
Initializing the Neural Network
function initialize_neural_network():
    randomly_initialize_weights_and_biases_for_each_neuron()
```

Data preprocessing is an essential step where input features are prepared for the Neural Network. It involves normalizing or standardizing input features on both training and testing data to ensure consistent and effective learning.

```
Preprocessing
function preprocess_data(training_data, testing_data):
    normalize_or_standardize_input_features(training_data)
    normalize_or_standardize_input_features(testing_data)
```

Figure 3. Data training and testing split

This section involves training the Neural Network. It iterates through a number of epochs (training iterations), and for each epoch, it goes through the training dataset. The forward pass calculates the predicted output of the Neural Network, and the backward pass (backpropagation) updates the weights and biases using the gradient descent algorithm to minimize errors. The training loop may stop early if the average error drops below a specified threshold, applying early stopping.

```
Training Network
function train_neural_network(training_data, labels):
    for epoch in range(number_of_epochs):
        for each input_data, target_label in zip(training_data, labels):
            # Forward Pass
            calculate_net_input_for_each_neuron()
            apply_activation_function_to_get_output()
            calculate_error_between_predicted_output_and_target_label()
            # Backward Pass (Backpropagation)
            update_weights_and_biases_using_gradient_descent_algorithm()
        # Calculate average error for this epoch
        if average_error < threshold:
            exit_training_loop()  # Early stopping
```

This section involves using the trained Neural Network to make predictions on the test dataset. It iterates through each input data in the test dataset, performs a forward pass to obtain the predicted output, and saves it for evaluation.

```
Prediction Network
function predict_neural_network(testing_data):
    for each input_data in testing_data:
        # Forward Pass
        calculate_net_input_for_each_neuron()
        apply_activation_function_to_get_output()
        save_predicted_output_for_evaluation()
```

This section involves evaluating the performance of the Neural Network based on the predictions made on the test dataset. It calculates various metrics such as accuracy, precision, recall, and F1 score to assess how well the Neural Network performs on the given task.

```
Evaluating
function evaluate_performance(predictions, true_labels):
    calculate_metrics_such_as_accuracy_precision_recall_and_f1_score()
    print_or_return_evaluation_results()
```

This is the main program that coordinates the entire process. It loads the dataset, performs data preprocessing, initializes the Neural Network, trains it on the training dataset, makes predictions on the test dataset, evaluates its performance, and prints or returns the evaluation results.

```
Main Program
training_data, testing_data, labels = load_data()  # Load dataset
preprocess_data(training_data, testing_data)

initialize_neural_network()
train_neural_network(training_data, labels)

predictions = predict_neural_network(testing_data)
evaluation_results = evaluate_performance(predictions, true_labels)
print(evaluation_results)
```

3. RESULT AND DISCUSSION

In this study, five tests were conducted: iteration number testing, learning rate testing, training and testing data size testing, hidden layer node unit testing, and minimum error testing against accuracy level. For the initial testing, a learning rate of 0.5, 4 hidden layer nodes, and CICIDS2018 dataset for training data were used. The iteration number testing aimed to obtain the optimal number of iterations for accuracy level. The iterations used were 100, 500, 1000, 1500, 2000, 2500, and 3000. The results of the iteration testing are shown in Table 1.

Table 1. Results of Iteration Number Testing

Iterasi	Accuracy i-th Experiment					Average
	1	2	3	4	5	
100	80.15	80.88	81.73	82.15	82.36	81.45
500	88.32	89.13	90.01	90.25	90.80	89.70
1000	89.25	90.07	90.77	91.25	91.62	90.59
1500	89.67	90.64	91.62	92.14	92.41	91.30
2000	89.86	90.60	91.54	91.88	92.80	91.33
2500	90.22	91.11	91.53	92.17	92.23	91.45
3000	89.54	90.07	90.47	90.98	91.71	90.55

The optimal iteration is shown at iteration 2500 because the highest average accuracy value is obtained, which is 91.45%. After obtaining the optimal number of iterations, the next test will use 2500 iterations. The purpose of the learning rate test is to obtain the optimal learning rate for accuracy level. Learning rate values are set from 0.1 to 0.9 with increments of 0.1. The results of the learning rate test are shown in Table 2.

Table 2. Results of Learning Rate Testing

Learning Rate	Accuracy i-th Experiment					Average
	1	2	3	4	5	
0.1	91.70	92.32	92.81	93.56	93.87	92.85
0.2	89.30	90.25	91.01	91.90	92.31	90.95
0.3	90.15	90.73	91.50	91.97	92.21	91.31
0.4	90.20	90.27	90.84	91.74	92.24	91.06
0.5	90.22	91.11	91.53	92.17	92.23	91.45
0.6	90.20	91.12	91.77	92.19	92.59	91.57
0.7	89.80	90.17	90.71	91.57	92.03	90.86
0.8	89.87	89.92	90.24	90.81	92.65	90.69

From the results of the learning rate test, the optimal value is shown at a learning rate of 0.1, with the highest accuracy value being 92.85%. After obtaining the optimal learning rate, the next test will use a learning rate of 0.1. The purpose of the training and testing data test is to obtain the optimal comparison of training and testing data for accuracy level. The combinations of data used are 20% training data and 80% testing data, 50% training data and 50% testing data, and 80% training data and 20% testing data. The results of the training and testing data test are shown in Table 3.

Table 3. Results of Training and Testing Data Comparison

Training data	Testing data	Accuracy i-th Experiment					Average
		1	2	3	4	5	
20	80	88.81	88.91	89.84	90.20	91.00	89.75
50	50	91.70	92.32	92.81	93.56	92.87	92.85
80	20	91.93	92.28	93.02	93.91	94.74	93.18

From the test results, it is known that the more training data, the better the accuracy. This is because more diverse data variations allow the model to recognize various data patterns during testing. The highest accuracy is obtained when 80% of the data is used for training and 20% for testing, with a percentage of 93.18%. After obtaining the optimal number of training and testing data, the next test will use 80% of the data for training and 20% for testing. The purpose of testing the number of nodes in the hidden layer is to obtain the optimal number of nodes for accuracy level. The number of nodes in the hidden layer set at 2, 3, 4, and 5.

Table 4. Hidden Layer Results

Hidden Layer	Accuracy i-th Experiment					Average
	1	2	3	4	5	
2	90.22	90.89	91.36	92.05	92.66	91.44
3	91.47	91.61	91.68	92.29	92.71	91.95
4	91.93	92.28	93.02	93.91	94.74	93.18
5	91.56	91.64	91.70	92.62	93.30	92.16

The results of the node count testing are shown in Table 4. From the test results, it is found that the optimal number of nodes is 4, with an average accuracy of 93.18%. After obtaining the optimal number of hidden units, the subsequent testing will use 4 hidden units. The purpose of the minimum error testing is to determine the optimal minimum error for accuracy. The set minimum errors are 0.1, 0.01, 0.001, 0.0001, and 0.00001. From Table 5, it is evident that the optimal minimum error is 0.00001, with an accuracy of 93.79%.

Table 5. Minimum Error Testing Results

Minimum error	Accuracy i-th Experiment					Average
	1	2	3	4	5	
0.1	90.22	91.10	91.86	91.90	92.37	91.49
0.01	91.47	91.51	92.01	92.78	93.32	92.22
0.001	91.93	92.28	93.02	93.91	94.74	93.18
0.0001	92.76	92.96	92.84	94.54	94.83	93.59
0.00001	90.17	90.92	91.60	92.39	92.72	91.56

From the training results, the optimal parameter values for achieving the highest accuracy are as follows: learning rate of 0.1, 80% training data and 20% testing data, 4 nodes in the hidden layer, a minimum error of 0.0001, and 2500 iterations.

4. CONCLUSION

This research conducted testing of neural networks in an Intrusion Detection System (IDS) through five main experiments. The training results demonstrate that selecting the appropriate parameters for the neural network can yield optimal accuracy for intrusion detection. The use of a learning rate of 0.1 has shown success in the Neural Network model. The utilization of 80% training data and 20% testing data is identified as an optimal combination, striking a good balance between training and testing. Furthermore, employing 4 nodes in the hidden layer is deemed the optimal number of nodes, indicating that a neural network architecture with this hidden layer complexity can yield good results. Additionally, adjusting the minimum error to 0.0001 and conducting 2500 iterations suggests that the neural network model achieves good convergence and enhances its ability to detect intrusion patterns. Overall, this research highlights the importance of parameter tuning to enhance the performance of neural network-based IDS in facing intrusion detection challenges.

This research has several limitations. First, the CICIDS2018 dataset may contain noise or have class imbalance that can affect the performance of the classification model. Second, this dataset may not encompass all types of attacks that could occur in real-world environments. New variations or attacks that are not represented may reduce the sensitivity of the model. Third, the use of Neural Network classification algorithms requires proper tuning of network architecture and training parameters for optimal results.

Future research could enhance neural network-based IDS by integrating advanced techniques like CNNs and LSTM networks. Leveraging CNNs for feature extraction and LSTM for modeling temporal dependencies can improve intrusion detection. Exploring hybrid architectures combining CNNs and LSTM may outperform traditional models. Investigating robustness against adversarial attacks is crucial for real-world reliability. Integrating CNNs and LSTM networks holds promise for addressing evolving challenges in intrusion detection.

REFERENCES

Alrawashdeh, K., & Purdy, C. (2017). Toward an online anomaly intrusion detection system based on deep learning. *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016*, 195–200. https://doi.org/10.1109/ICMLA.2016.167

Amijoyo, T., Umar, R., & Yudhana, A. (2020). Bruteforce In The Hydra Process And Telnet Service Using The Naïve Bayes Method. *Jurnal Mantik*, *4*(1).

Ananin, E. V., Nikishova, A. V., & Kozhevnikova, I. S. (2017). Port scanning detection based on anomalies. *11th International IEEE Scientific and Technical Conference &Amp;Quot;Dynamics of Systems, Mechanisms and Machines&Amp;Quot;, Dynamics 2017 - Proceedings*, *2017-Novem*, 1–5. https://doi.org/10.1109/Dynamics.2017.8239427

Aziz Sharfuddin, A., Nafis Tihami, M., & Saiful Islam, M. (2018). A Deep Recurrent Neural Network with BiLSTM model for Sentiment Classification. *2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018*, 1–4. https://doi.org/10.1109/ICBSLP.2018.8554396

Bharati, M., & Tamane, S. (2017). Intrusion detection systems (IDS) & future challenges in cloud based environment. *Proceedings - 1st International Conference on Intelligent Systems and Information Management, ICISIM 2017*, *2017-January*, 240–250. https://doi.org/10.1109/ICISIM.2017.8122180

Bhatia, V., Choudhary, S., & Ramkumar, K. R. (2020). A Comparative Study on Various Intrusion Detection Techniques Using Machine Learning and Neural Network. *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 232–236. https://doi.org/10.1109/ICRITO48877.2020.9198008

Bostani, H., & Sheikhan, M. (2017). Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach. *Computer Communications*, *98*, 52–71. https://doi.org/10.1016/j.comcom.2016.12.001

Bul'ajoul, W., James, A., & Pannu, M. (2013). Network intrusion detection systems in high-speed traffic in computer networks. *Proceedings - 2013 IEEE 10th International Conference on e-Business Engineering, ICEBE 2013*, 168–175. https://doi.org/10.1109/ICEBE.2013.26

Chaudhary, A. (2015). Development of a new intrusion detection system for mobile and ADHOC networks using soft computing techniques. *University*. http://hdl.handle.net/10603/184962

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, *169*, 102767. https://doi.org/10.1016/J.JNCA.2020.102767

Gunjan, V. K. (2015). Advancement of artificial neural network algorithms for intrusion detection in computer network. *INFLIBNET*. http://hdl.handle.net/10603/148064

Guri, M., Puzis, R., Choo, K. K. R., Rubinshtein, S., Kedma, G., & Elovici, Y. (2019). Using malware for the greater good: Mitigating data leakage. *Journal of Network and Computer Applications*, *145*, 102405. https://doi.org/10.1016/J.JNCA.2019.07.006

Hossain, M. D., Ochiai, H., Doudou, F., & Kadobayashi, Y. (2020). SSH and FTP brute-force attacks detection in computer networks: Lstm and machine learning approaches. *2020 5th International Conference on Computer and Communication Systems, ICCCS 2020*, 491–497. https://doi.org/10.1109/ICCCS49078.2020.9118459

Hosseini, S., & Zade, B. M. H. (2020). New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN. *Computer Networks*, *173*, 107168. https://doi.org/10.1016/J.COMNET.2020.107168

Imrana, Y., Xiang, Y., Ali, L., & Abdul-Rauf, Z. (2021). A bidirectional LSTM deep learning approach for intrusion detection. *Expert Systems with Applications*, *185*(July), 115524. https://doi.org/10.1016/j.eswa.2021.115524

Kiranyaz, S., Ince, T., Iosifidis, A., & Gabbouj, M. (2020). Operational neural networks. *Neural Computing and Applications*, *32*(11), 6645–6668. https://doi.org/10.1007/S00521-020-04780-3

Kowsher, M., Tahabilder, A., Islam Sanjid, M. Z., Prottasha, N. J., Uddin, M. S., Hossain, M. A., & Kader Jilani, M. A. (2021). LSTM-ANN & BiLSTM-ANN: Hybrid deep learning models for enhanced classification accuracy. *Procedia Computer Science*, *193*, 131–140. https://doi.org/10.1016/j.procs.2021.10.013

Laghrissi, F., Douzi, S., Douzi, K., & Hssina, B. (2021). Intrusion detection systems using long short-term memory (LSTM). *Journal of Big Data*, *8*(1), 65. https://doi.org/10.1186/s40537-021-00448-4

Liao, H. J., Richard Lin, C. H., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004

Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., & Kemp, C. (2016). Detection of SSH brute force attacks using aggregated netflow data. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 283–288. https://doi.org/10.1109/ICMLA.2015.20

Najafian, Z., Aghazarian, V., & Hedayati, A. (2015). Signature-Based Method and Stream Data Mining Technique Performance Evaluation for Security and Intrusion Detection in Advanced Metering Infrastructures (AMI). *International Journal of Computer and Electrical Engineering*, *7*(2), 128–139. https://doi.org/10.17706/ijcee.2015.v7.879

Otoum, Y., & Nayak, A. (2021). AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *Journal of Network and Systems Management*, *29*(3). https://doi.org/10.1007/S10922-021-09589-6

Pawlicki, M., Kozik, R., & Choraś, M. (2022). A survey on neural networks for (cyber-) security and (cyber-) security of neural networks. *Neurocomputing*, *500*, 1075–1087. https://doi.org/10.1016/J.NEUCOM.2022.06.002/A_SURVEY_ON_NEURAL_NETWORKS_FOR_CYBER_SECURITY_AND_CYBER_SECURITY_OF_NEURAL_NETWORKS.PDF

Sadasivam, G. K., Hota, C., & Anand, B. (2016). Classification of SSH attacks using machine learning algorithms. *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*. https://doi.org/10.1109/ICITCS.2016.7740316

Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems*, *80*, 157–170. https://doi.org/10.1016/j.future.2017.10.016

Sung, Y., Jang, S., Jeong, Y. S., & Park, J. H. (James J.). (2020). Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations. *Computer Communications*, *153*(January), 342–348. https://doi.org/10.1016/j.comcom.2020.02.005

Wang, Y. C., Houng, Y. C., Chen, H. X., & Tseng, S. M. (2023). Network Anomaly Intrusion Detection Based on Deep Learning Approach. *Sensors 2023, Vol. 23, Page 2171*, *23*(4), 2171. https://doi.org/10.3390/S23042171