Artificial Intelligence based firewall

1st Tudor Alinei-Poiană
dept. Automation and Computer Science
Tecnical University of Cluj-Napoca
Cluj-Napoca, Romania
Tudor.Alinei@aut.utcluj.ro

***Abstract*—In contemporary digital ecosystems, where security breaches can have far-reaching consequences, the fusion of robust cybersecurity measures and cutting-edge technology is crucial. With the exponential growth of AI and the proliferation of powerful computing systems, the importance of fortifying cyber defenses has never been more critical. This proposition asserts that integrating Artificial Intelligence (AI) into firewall systems is essential to keep sensitive data safe from malicious attacks. By harnessing AI's advanced capabilities in real-time threat detection and adaptive response, AI-powered firewalls offer a proactive defense mechanism against sophisticated cyber-attacks. Thus, the convergence of AI and cyber-security not only addresses the pressing need for enhanced protection but also aligns with the rapid advancements in technology, ensuring resilience in the face of emerging threats.**

***Index Terms*—cybersecurity, artificial intelligence, real-time detection, proactive defence**

I. Introduction

Securing and protecting an application, regardless of the program's specificity plays a key role in gaining a user's long term trust, but also discourages malicious attacks and ensures the safety of sensitive data that could be exploited by the attackers. Irrespective of the kind of data that an application gathers from the user (name, credit card series, address, accounts passwords etc.) an application must ensure a certain level of protection against attempted data thefts. Classic data protection mechanisms of web applications such as: usage of HTTPS (Hypertext Transfer Protocol Secure) to encrypt communication between client and server, input validation (used against SQL injection, cross-site scripting, cross-site request forgery), usage of multi-factor authentication, secure session management using session timeouts or session rotation, keeping all software components up to date, data encryption or even educating the users and developers about common security tasks provide efficient and powerful tools in countering attackers. However, nowadays web applications lack such features or are poorly implemented due to the increase demand and the need for rapid implementation and deployment of a software.

With the increase in computational speed and development of powerful and efficient machine learning (ML) algorithms such as clustering, neural networks (NN), multiclass-classification, regression, Gaussian-Mixture Model etc., the two fields of security and AI merged, contributing to the development of AI based security mechanisms which are currently under intensive research.

Artificial neural networks (ANN) can be used in predicting and protecting a users personal data from the access of an unauthorised person. By using an ANN, combined with different models of malicious attacks [1] describes a protection mechanism of disabling the current access point and alerting the user by predicting the requests intent using the users biometrics and password authentication. The above mentioned method also uses honeypots which are decoy systems or resources intentionally deployed to attract attackers, allowing the network to analyze their behavior and predict potential future threats. In [2], [3] the capabilities of auto-associative neural networks are exploited in order to build a mechanism for efficient and secure data encryption/decryption, while [4] compares two types of NN architectures trained to check and validate documents which may contain sensitive data that needs to be processed, the proposed models reaching an accurracy in sensitive data highlighting between 88.14% and 96.67%.

Also, [5], [6] offer a much complex mechanism of quantifying the similarities between different types of sensitive data by using Shannon information entropy and NNs at the cost of computational speed.

Despite being accurate, the main drawback of the above mentioned methods consists of the reduced real-time response capacity which in web application is crucial in order to prevent and protect a user or server from an attacker.

This paper focuses on building and comparing different types of machine learning models and test their accuracy in predicting whether a http request is malicious or not. The models included in this study are supervised learning type like regression, ridge regression or random forest regression which are used in binary classification problems and unsupervised learning models such as KMeans clustering algorithm. The models are trained to recognise SQL injection attacks, but they can easily be extended to other types of attacks by adapting the training dataset.

This paper is structured as follows: Section II consists of a brief description of the tools used in gathering and parsing the data alongside the concept structure of a real usage of the models, Section III presents the methodology, including conceptualization of data gathering and the description of the main ML algorithms, followed by IV which describes the main results and performances of the algorithms and a brief conclusion presented in Section V.

II. Infrastructure

In this section will be presented the software components that were used and their role gathering and preprocessing data and implementing the machine learning models. The resources used in this paper were:

- Acunetix Web Vulnerability Scanner (WVS) which is a penetration testing software capable of generating automatic http requests (good http requests that test a web application functionality via crawling functionality or bad requests such as cross-site scripting or SQL injection);
- Burp suite which was used as a proxy that intercepts the requests and responses to and from a website and also logs all the necessary information;
- For testing the AI based firewall, Mozilla Firefox was configured to send all the requests to a manually configured proxy;
- The above mentioned proxy was implemented using python;
- For data parsing, a python script that parses Burp Suite logs was implemented;
- For clustering, KMeans algorithm was considered and implemented in python using numpy;
- The rest of the models were implemented using the PYCaret machine learning framework and trained using a Jupiter Notebook configured in a Conda environment.

Figure 1 shows the whole pipeline which forms the base of this study. The methodology behind this pipeline will be thoroughly presented in the next subsection.

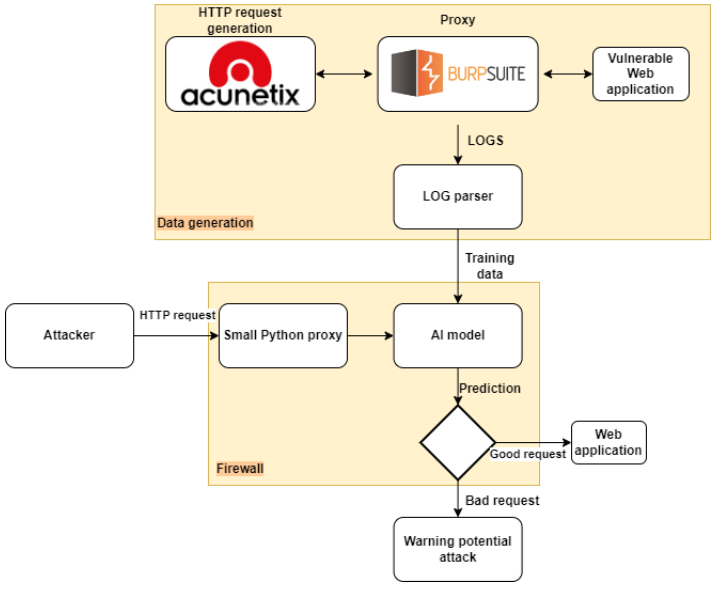

Fig. 1. AI-based firewall pipeline

III. Methodology

A. Data generation

In order to generate the required data to train the models a Web Vulnerability Scanner (WVS) was used to automatically generate good (via a crawler) and malicious requests (in this case the malicious requests were SQL injection-type requests, but it can be generalised to other type of attacks). In order to make no harm to other web applications, the requests were sent to dedicated pen testing websites such as: https://demo.testfire.net/, http://testphp.vulnweb.com/ ,http://rest.vulnweb.com/ etc. Because the SQL injection requests were much more numerous than the good requests, the data set could have been unbalanced so in order to get more good requests, the crawler was configured to send normal type of requests to much bigger web applications such as Reddit.

The WVS was configured to send requests through a proxy. The scanner routed its requests through the specified proxy server, which then forwards them to the target web application. The proxy essentially acts as an intermediary, relaying the scanner's requests to the target and intercepting responses. This setup allowed logging, monitoring, and modifying requests and responses for analysis and data generation.

The proxy was configured using Burp Suite, another WVS, to listen to all IPs and ports in order to log all the requests sent by Acunetix. Those logs were saved and parsed using a python script that essentially divided the content of the XML log file and decoding it.

After this operation, the newly obtained data contains the headers, method, body and path of the http request.

The most important and difficult part in training a machine learning model is how you define and quantify the key features from which the model will be able to learn relevant patterns. For this study, the proposed method to quantify and make use of the above mentioned data is to count the number of occurrences of words such as: sleep, drop, delay, system, admin etc. and certain combinations of those words along with the occurrences of certain symbols (which normally don't belong to a harmless request) such as: \, ', –, (, @,).

Those counted words, symbols or combinations are saved in a .csv file alongside the class from which the http request belongs to (0 for malicious requests or 1 for good requests). For the unsupervised learning model, the type of request will not be used to train the model, because those models learn patterns from unlabeled data, but the associated labels will be useful in order to train the supervised learning models. In order to compare the influence of the dataset to the performance of the firewall, for each model were considered two datasets: one with a limited number of counted features and one with an extended number of features considered.

In the next subsection the main principles behind the unsupervised learning model KMeans clustering will be presented.

B. KMeans clustering

K-means clustering is the most widely used partitional clustering algorithm. It starts by choosing K representative points as the initial centroids. Each point is then assigned to the closest centroid based on a particular proximity measure chosen. Once the clusters are formed, the centroids for each cluster are updated. The algorithm then iteratively repeats these two steps until the centroids do not change or any

other alternative relaxed convergence criterion is met. K-means clustering is a greedy algorithm which is guaranteed to converge to a local minimum but the minimization of its score function is known to be NP-Hard. Typically, the convergence condition is relaxed and a weaker condition may be used. In practice, it follows the rule that the iterative procedure must be continued until 1% of the points change their cluster memberships or until the distance between the old and the updated centroids is less than a specific treshold ϵ. A detailed proof of the mathematical convergence of K-means can be found in [7].

The mathematical way to describe the purpose of this algorithm is: given a set of n data points $\{x_1, x_2, ..., x_n\}$, and a specified number of clusters k, let $C = \{c_1, c_2, ..., c_k\}$ be the set of cluster centroids. The objective function of K-means clustering is defined as:

$$\min J = \sum_{i=1}^{k} \sum_{x \in S_i} ||x - c_i||^2 \tag{1}$$

where S_i is the set of data points assigned to cluster i, and $||x - c_i||^2$ represents the squared Euclidean distance between data point x and centroid c_i.

The algorithm proceeds with the following steps:

1) Initialize cluster centroids randomly.
2) Assign each data point to the nearest centroid.
3) Update each centroid by computing the mean of the data points assigned to it.
4) Repeat steps 2 and 3 until convergence, i.e., until the centroids no longer change significantly or a maximum number of iterations is reached.

The algorithm converges to a solution that minimizes the sum of variances of each cluster.

The most important factors that have relevant impact on the performance of the KMeans algorithm are:

1) Choosing the initial centroids;
2) Estimating the number of clusters.

Estimating the number of clusters in this case is not a concern, because depending on how many kind of attacks or how we choose to label our types of requests (e.g. good and bad or good, SQLi and Cross-ref attacks etc) we will previously know how many clusters will be. The only open problem in this case is choosing an efficient method to initialize our centroids.

A poor initialization will lead to very poor performance of our clustering algorithm, as shown in Figure 2, while a good initialization will lead to better results like in Figure 3.

The initialization method chosen for this model is the KMeans++ which follows a simple probability-based approach where initially the first centroid is selected at random. The next centroid selected is the one which is farthest from the currently selected centroid. This selection is decided based on a weighted probability score which seeks maximizing the distance between all the selected K centroids. The selection is continued until we have K centroids and then K-means clustering is done using these centroids.

Fig. 2. Bad centroids initialization

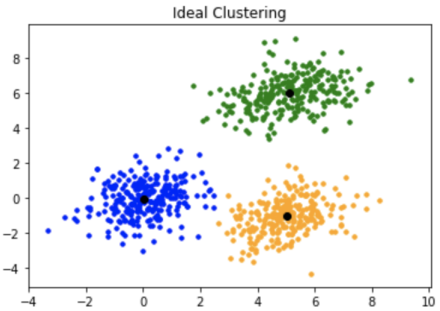

Fig. 3. Good centroids initialization

The biggest advantage of this algorithm is it's simplicity and versatility, because, unlike the classification type algorithms, once the KMeans clustering was implemented, the number of clusters (classes) can be varied, without having to change the actual implementation and also doesn't make any assumptions on the distribution of data.

Though it is a very powerful tool, it's main disadvantage is its sensitivity to initialization which can lead to poor clustering performance.

In the next subsection, another type of machine learning model will be detailed alongside its possible advantages and drawbacks.

C. Logistic regression

In order to compare the supervised and unsupervised types of models used for a firewall, the data must be divided in the same number of classes/clusters. By using only two clusters to distinguish between the good and malicious requests, for a meaningful comparison, a binary classification algorithm will be used for the supervised learning model.

There are various classification models such as: logistic regression, K neighbors classifier, random forest classifier, linear discriminant analysis etc. In this subsection, we will briefly describe the simplest and most known algorithm, logistic regression.

Logistic regression is a fundamental statistical technique used for binary classification tasks. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability of an event occurring, typically denoting one of two outcomes. Despite its name, logistic regression is a

classification algorithm rather than a regression one. It models the relationship between the dependent variable and one or more independent variables by estimating probabilities using the logistic function, also known as the sigmoid function.

The problem can be stated as: given a set of features denoted $x = (x_1, x_2, ..., x_n)$ we want to predict the probability of x of being or not in one of the two considered classes. The key mathematical concept underlying logistic regression is the logistic function (also known as the sigmoid function), which maps any real-valued number to a value between 0 and 1. The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Here, z is the linear combination of the predictor variables and their respective coefficients:

$$z = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n \tag{3}$$

The set of parameters $\theta = (\theta_0, \theta_1, .., \theta_n)$ seek to find the line/hyperplane which best separates two classes, as shown in a 2D case in Figure 4. Also, this separation is performed by training the set of parameters such that they minimize the entropy cost function:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \Big[y^{(i)} \log(P(y^{(i)}|x^{(i)}; \theta)) + (1 - y^{(i)}) \log(1 - P(y^{(i)}|x^{(i)}; \theta)) \tag{4}$$

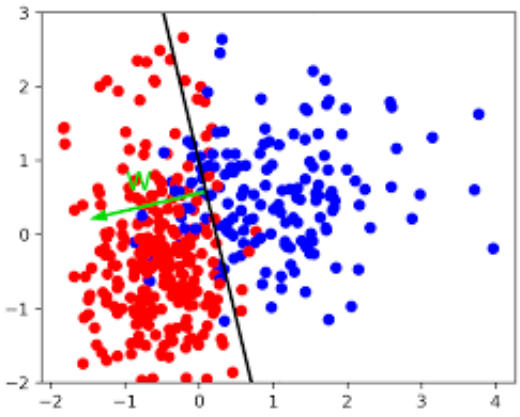

Fig. 4. Class separation

The term $\log(P(y^{(i)}|x^{(i)}; \theta))$ penalizes the model heavily if it predicts a low probability for an actual positive outcome and vice versa. The sum of these penalties over all training examples provides a measure of how well the logistic regression model fits the data.

Minimizing the cost function $J(\theta)$ is typically accomplished using optimization algorithms such as gradient descent, stochastic gradient descent, or more advanced optimization techniques. These algorithms iteratively update the parameters θ in the direction that decreases the cost function until convergence is achieved.

This algorithm is well suited in cases where the classification problem can be splitted in two classes due to its simplicity and efficiency, but for multi-class classification, more advanced techniques are required.

The next subsection focuses on briefly describing the proxy used for request interception and AI firewall protection.

D. AI based firewall

The firewall is built as it follows:

1) **HTTP Proxy Server**:
   - The code creates an HTTP proxy server that listens for incoming HTTP requests on `127.0.0.1` (localhost) at port `8080`.
   - When a client sends a GET request to this proxy server, it intercepts the request and processes it.
2) **Proxy Routes**:
   - It allows defining custom proxy routes using the `set_routes` class method. This enables the proxy server to forward requests to different destinations based on predefined routes.
3) **Request Handling**:
   - Upon receiving a GET request, the server inspects the request path to extract relevant information, such as the requested URL.
   - If the request path contains at least two parts, indicating it's a valid URL, the server proxies the request to the specified destination. Otherwise, it serves files from the local directory (default behavior of `SimpleHTTPRequestHandler`).
4) **Machine Learning Model Integration**:
   - The code loads a pre-trained machine learning model using a function named `load_model`. This model is presumably trained to classify HTTP requests as either malicious or benign.
   - It extracts features from a portion of the request path using a function called `extractFeatures` (the same function used to extract the features in the training process).
   - The extracted features are then used as input to the machine learning model to predict whether the request is malicious or not.
5) **Response Handling**:
   - If the machine learning model predicts that the request is malicious (based on the label 0), the server prints a warning message indicating that it's a potentially dangerous request.
   - If there's an error while processing the request (e.g., HTTP error), the server sends an error response with the appropriate status code.
6) **Starting and Stopping the Server**:
   - Once the server is set up, it begins listening for incoming requests indefinitely using `serve_forever`.

- It catches the `KeyboardInterrupt` exception, allowing the server to be stopped gracefully when the user interrupts the program (e.g., by pressing Ctrl+C).

The firewall combines the functionality of a basic HTTP proxy server with machine learning-based request inspection to enhance security by identifying and potentially blocking malicious HTTP requests.

IV. Results

In this section two cluster based firewalls (a simple cluster firewall, SCF, and, a so called, enhanced cluster firewall, ECF) and binary classification based firewalls (BCF) will be presented alongside some results and performance analysis via relevant graphs and metrics used for machine learning algorithms.

A. Simple clustering based firewall

This subsection, some results for the SCF will be presented, the main difference between this one and the ECF being the features considered in the training process. For SCF, the total number of relevant features considered was six.

In Figure 5 and Figure 6 it can be seen the result of the clustering algorithm. Having considered six features in the dataset, a normal graphical representation for the clusters could not have been possible without reducing the dimensionality of the variables represented. This was achieved using principal component analysis (PCA) in order to the feature space to two, respectively three dimensions. The PCA reduces the number of features in order to obtain fewer linear combinations of the original variables that maximally explain the variance of all the variables (this was only performed for visual purposes, the actual training being done with the original feature set).

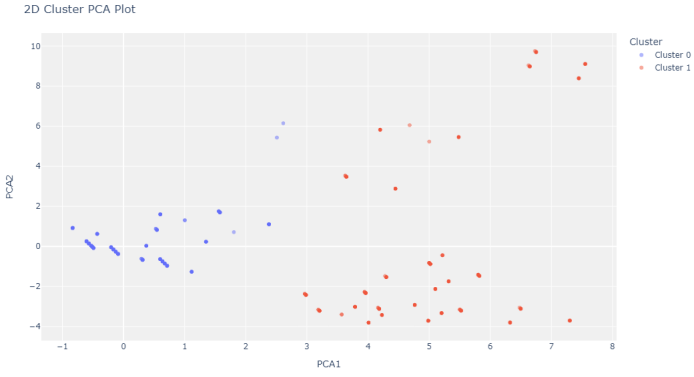

Fig. 5. SCF-2D clustering plot of the requests using PCA

The red dots represent the cluster denoting malicious requests (Cluster 1), while the blue dots represent the cluster of benign requests (Cluster 0). Validating a clustering algorithm can be done using the internal cluster validation index i.e. Calinski-Harabasz (CH) index. This index can be used to evaluate the model when ground truth labels are not known where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. The CH index is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Higher values of the CH index means better separation between clusters and also that clusters are dense. The computed value for the CH index is 4024.05. The percentage of false neagtive results (malicious requests that were passed as good requests from the total of http requests used for testing) is 33.33%, while the success rate in identifying malicious requests is 66.66%. For the true positive http requests the rate was 100%, so no benign request were rejected by the SCF.

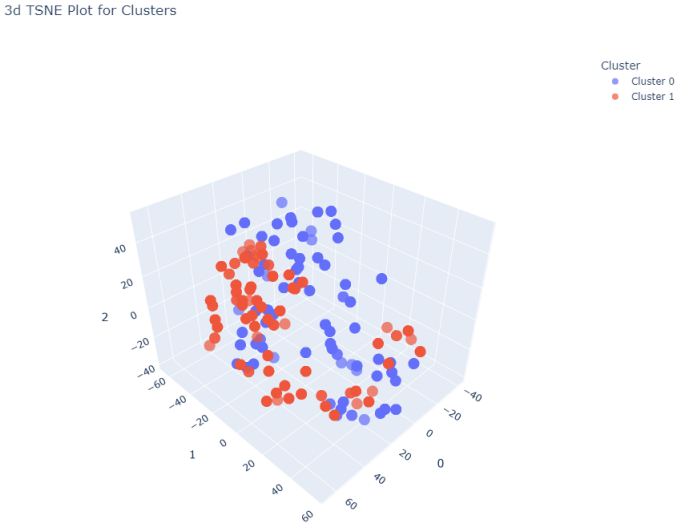

Fig. 6. SCF-3D clustering plot of the requests using PCA

B. Enhanced clustering based firewall

In this subsection the performance of the ECF will be presented. For this model, a total of 23 features were considered during the training process.

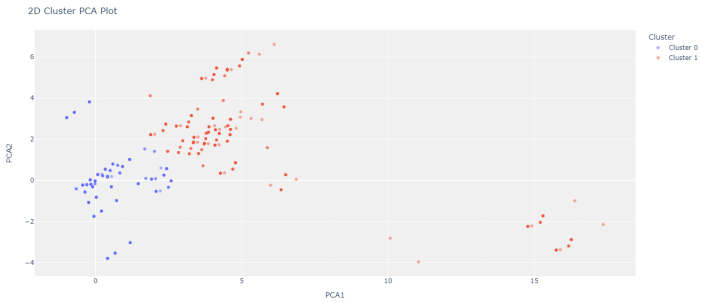

Fig. 7. ECF-2D clustering plot of the requests using PCA

The CH index computed for this clustering model is 1906.4, which is not as high as the SCF CH index, but the model compensates by using a higher order feature space.

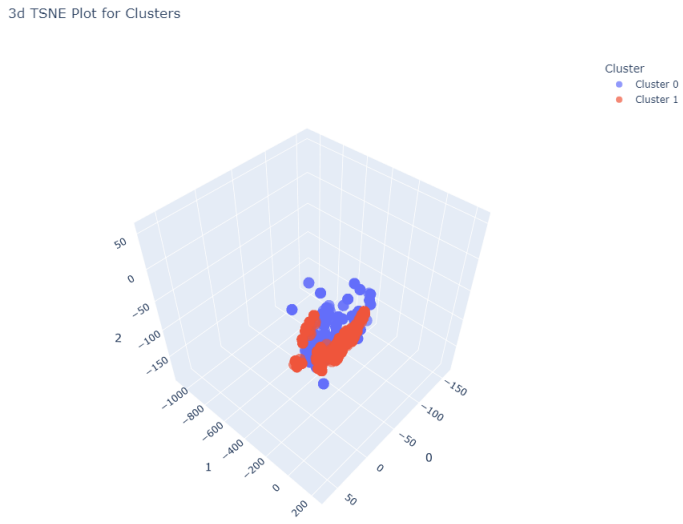

Fig. 8. ECF-3D clustering plot of the requests using PCA

Figures 7, 8 describe the obtained clusters, the representation being done using the same PCA method as in the previous subsection. This model was able to correctly detec a percentage of 76% percent of the malicious requests, while the false negative percentage was only 24%, with 9% better than the SCF case. This thing shows the importance of a relevant and complex dataset in training a machine learning based firewall.

C. Binary classification based firewall

In this last subsection, a binary classification algorithm was used in order to classify the http requests sent to a web application.

Figure9 depicts the performance of multiple binary classification algorithms, the one with the best cost to performance ration being the Decision tree classifier (DTC).

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
dt	Decision Tree Classifier	0.9967	0.5883	0.9833	1.0000	0.9915	0.9895	0.9896	0.0190
rf	Random Forest Classifier	0.9967	0.6727	0.9833	1.0000	0.9915	0.9895	0.9896	0.0960
ada	Ada Boost Classifier	0.9967	0.6715	0.9833	1.0000	0.9915	0.9895	0.9896	0.0600
gbc	Gradient Boosting Classifier	0.9967	0.6712	0.9833	1.0000	0.9915	0.9895	0.9896	0.0770
et	Extra Trees Classifier	0.9967	0.6435	0.9833	1.0000	0.9915	0.9895	0.9896	0.0740
lightgbm	Light Gradient Boosting Machine	0.9967	0.6732	0.9833	1.0000	0.9915	0.9895	0.9896	0.0370
lr	Logistic Regression	0.9951	0.9829	0.9750	1.0000	0.9872	0.9842	0.9844	0.7650
knn	K Neighbors Classifier	0.9951	0.6292	0.9750	1.0000	0.9872	0.9842	0.9844	0.0380
svm	SVM - Linear Kernel	0.9940	0.0000	0.9694	1.0000	0.9843	0.9806	0.9809	0.0150
nb	Naive Bayes	0.9689	0.5000	0.9944	0.8685	0.9268	0.9073	0.9109	0.0240
qda	Quadratic Discriminant Analysis	0.8234	0.5000	0.1000	0.1000	0.1000	0.1000	0.1000	0.0170
ridge	Ridge Classifier	0.8037	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0160
lda	Linear Discriminant Analysis	0.8037	0.9286	0.0000	0.0000	0.0000	0.0000	0.0000	0.0190
dummy	Dummy Classifier	0.8037	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0140

Fig. 9. Different binary classification algorithms performances

The accuracy of the model is about 99.6 percent after the training process. The area under the curve (AUC) index, which is a performance measurement for the classification problems at various threshold settings has a value arounf 0.58. AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. Though, the AUC for other classification algorithms is higher, e.g. Ada boost classifier which is 0.67, the preferred model was DTC due to it speed, the prediction time being around 20 milliseconds, compared to 60 milliseconds (the reaction time for a firewall being crucial in promptly identifying and blocking any potential attack).

In Figure 10 we can clearly see the performance of the chosen classifier with the help of a confusion matrix. In this case, the malicious requests were considered having the label equal to zero, while the good requests have the label one. The confusion matrix tells us the following: there were 632 true positive results (632 true malicious requests predicted), 0 benign requests that were classified as malicious, 2 malicious requests which were classified as good ones and 153 good requests which were correctly classified as benign requests.

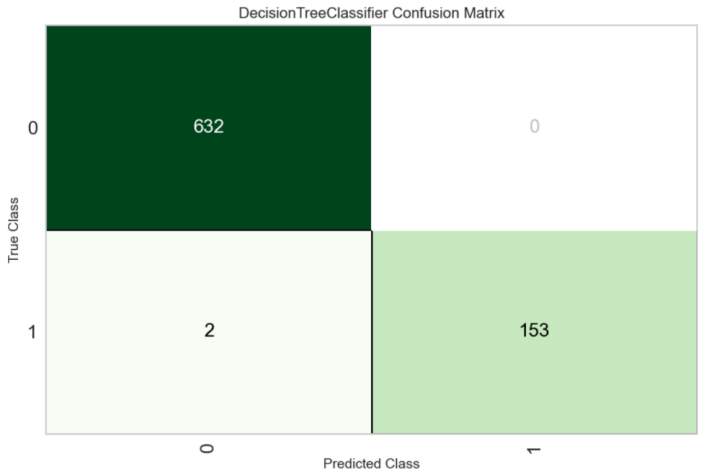

Fig. 10. Different binary classification algorithms performances

The percentage of malicious requests intercepted by the BCF when using testing data was around 53.1%, which is lower than the unsupervised learning firewalls, but the good part for this firewall is that it was able to classify all the good requests as benign and no false negative requests were predicted.

V. Conclusion

In conclusion, both supervised and unsupervised machine learning based firewalls (SCF, ECF and BCF) can provide satisfactory results. The efficacy of AI-powered firewalls heavily relies on the sophistication of their underlying algorithms and the quality of the datasets they are trained on. In this comparison, we observed that ECF employing unsupervised learning with a complex dataset outperformed its counterparts. This

highlights the importance of utilizing advanced techniques and rich data sources for optimal threat detection and mitigation.

While BCF showed some promise, its performance fell short compared to the unsupervised approach. The BCF, despite its structured approach, proved to be the least effective in this context, but with the proper fine tuning and improvements in the dataset, it can surely reach or even surpass the ECF.

Future research directions will include enhancing and diversifying the dataset in order to provide much relevant information in the training process, optimizing algorithmic approaches (e.g. studying different clustering initialization methods or using better optimization techniques for the classification models) and extending the types of attacks which can be detected by the AI based firewall, such as cross-site scripting, cross-site request forgery, XML external entity injection, file upload vulnerabilities etc.

REFERENCES

[1] J. Jagannathan, M.Y.M. Parvees, "Security breach prediction using Artificial Neural Networks", Measurement: Sensors, Volume 24, 2022, 100448, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2022.100448.

[2] I. Tsmots, V. Rabyk, Z. Lyubun, and O. Skorokhoda, "Method and means of symmetric real-time neural network data encryption," Proc. XVth International Scientific and Technical Conference Computer Science and Information Technologies. pp. 47–51, 2020.

[3] I. Tsmots, V. Rabyk, Y. Lukaschuk, V. Teslyuk and Z. Liubun, "Neural Network Technology for Protecting Cryptographic Data", 2021 IEEE 12th International Conference on Electronics and Information Technologies (ELIT), Lviv, Ukraine, 2021, pp. 103-108, doi: 10.1109/ELIT53502.2021.9501094.

[4] C.H. Lin, P.K. Yang and Y.C. Lin, "Detecting Security Breaches in Personal Data Protection with Machine Learning", 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), Taichung, Taiwan, 2020, pp. 1-7, doi: 10.1109/IMCOM48794.2020.9001710.

[5] C. Wang, "Privacy Data Measurement and Classification Model Based on Shannon Information Entropy and BP Neural Network", 2023 International Conference on Data Science and Network Security (ICDSNS), Tiptur, India, 2023, pp. 1-5, doi: 10.1109/ICDSNS58469.2023.10244966.

[6] B. Zhang, "Research on Data Security Protection Algorithm Based on BP Neural Network in Cloud Computing Environment", 2022 2nd International Conference on Networking, Communications and Information Technology (NetCIT), Manchester, United Kingdom, 2022, pp. 334-337, doi: 10.1109/NetCIT57419.2022.00085.

[7] S.Z. Selim and M.A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(1):81–87, 1984.