PRIMARY RESEARCH

Attackdet: Combining web data parsing and real-time analysis with machine learning

Zeydin Pala [1*], Musa Şana [2]
[1] Department of Computer Engineering, Muş Alparslan University, Muş, Turkey
[2] Department of Pentest of ADEO Security, Ankara, Turkey

Keywords
Machine learning
Real-time processing
Asynchronous programming
HTTP protocol
Websocket

Received: 8 September 2019
Accepted: 9 November 2019
Published: 12 February 2020

Abstract
In this study, the web traffic was analyzed via machine learning (ML) support, and incoming traffic was visualized after real-time classification, prioritizing stability and performance, which are indispensable for real-time applications. Websocket technology was used for instantaneous and fast data transfer. Processes may be blocked due to asynchronous operating structure when Hyper-Text Transfer Protocol (HTTP) traffic is intensive. Synchronous operation of the system was causing both delays and negatively affecting the efficiency of the application. To overcome this bottleneck, the developed application used asynchronous libraries instead of synchronous ones. The essential features of the study were the analysis of HTTP packets captured in real-time, classifying the packets according to whether they are safe or suspicious using ML algorithms, and real-time display of the acquired results. In this way, incoming traffic was classified smartly without getting lost in thousands of log files. A success rate of 96.49% was attained using the logistic regression model, which is very successful in classification.

I. INTRODUCTION

Since the advent of computers in the middle of the last century, our lives have become more and more computerized, socialized and digitalized. Since computer systems are used in different business types for production, more data is continuously produced and collected, especially in retail and finance. Whatever the source (commercial, scientific or personal) smart people are finding new ways to use this data and turn it into a useful product or service. Machine learning (ML) plays a critical role in this transformation. ML is now the driving force of artificial intelligence. After the disappointment of logic-based programmed expert systems in the 1980s, it revitalized the field and achieved essential results [1]. ML can be used to classify web traffic efficiently [2] and intelligently [3]. In other words, web security can be provided more effectively with the help of ML [4, 5]. In this area, many ML technical models are presented by many researchers to prevent malicious traffic flows in the Internet of Things (IoT) network [6]. ML is also used extensively in time series prediction processes [7, 8, 9, 10, 10, 11].

Today, there are many applications which display web traffic in real-time [12, 13]. Unfortunately, the use of such a display which may be beneficial up to a certain degree, falls short of meeting the desired demands. It is expected to check in real-time and without any compromise on performance whether the incoming or outgoing web traffic is suspicious or not real-time. However, using ML for this purpose takes us one step farther. Designing a system that classifies web traffic smartly by considering the shortcomings in current applications and displays the results graphically in real-time is the primary objective of this study.

Incoming HTTP traffic was captured in this study by way of the application layer which was afterward displayed in real-time. Packets of captured HTTP traffic are sent to a model trained via ML and the inferences made by the machine are evaluated. In other words, the packets in the traffic are classified via pre-trained ML model, after which they are displayed in real-time using a web application. It is thus easier

[*]Corresponding author: Zeydin Pala
[†]email: z.pala@alparslan.edu.tr

and more effective to visually detect suspicious activities or harmful interventions. The web-based application can be used easily in all operating systems regardless of the platform and it can be configured rapidly. Besides, analyses are more comfortable and easier thanks to graphic support.
Parsing http packets in web traffic and real-time classification of the results acquired in combination with support from ML is considered an innovation of the study. In this regard, the study shall make significant contributions to the literature.

II. BACKGROUND

As shown in Figure 1, Attackdet is comprised of three basic modules as an application that is the subject of the study. The first module sniffs and parses the network traffic in the background. The second module sends the parsed data to the ML server. Whereas the third module displays in a web application the data for the parsed network packets along with the results classified through ML. In general, two servers run in the background which are the web server and the ML server.

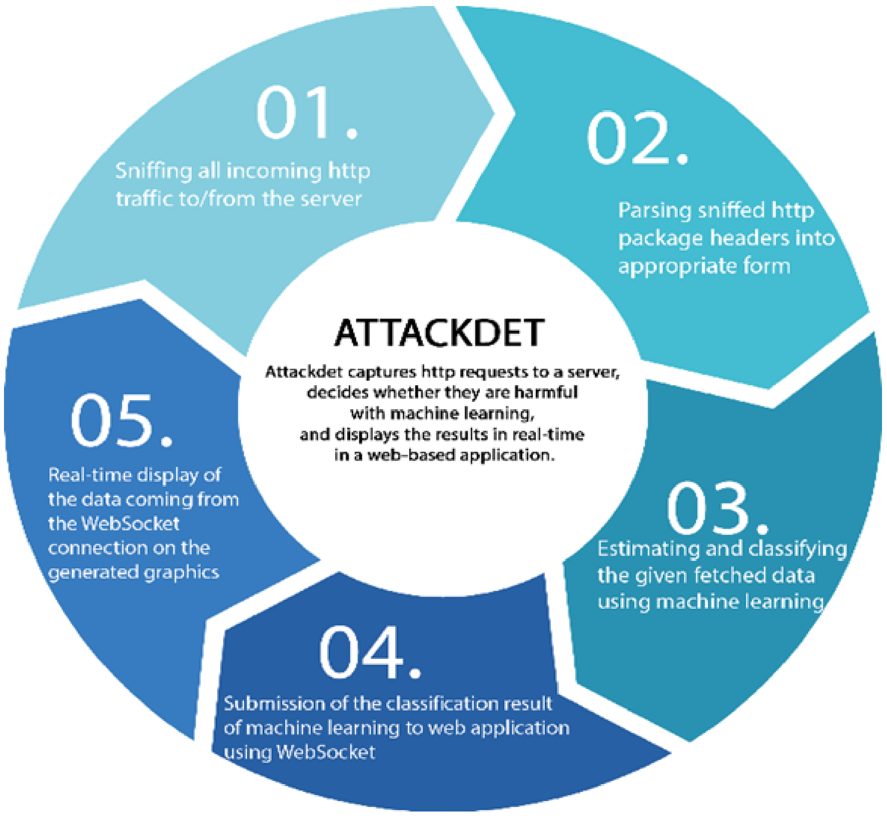
01.
Sniffing all incoming http traffic to/from the server
02.
Parsing sniffed http package headers into appropriate form
03.
Estimating and classifying the given fetched data using machine learning
04.
Submission of the classification result of machine learning to web application using WebSocket
05.
Real-time display of the data coming from the WebSocket connection on the generated graphics
ATTACKDET
Attackdet captures http requests to a server, decides whether they are harmful with machine learning, and displays the results in real-time in a web-based application.

Fig. 1. General system architecture [14]

A. Libraries and Technologies used in the Developed Application

1) Python programming language and the Tornado library developed in this language have been used in th is application's back-end. Processes may be carried out without being blocked since this library works asynchronously.
2) Another essentialvital library used in this study was scapy. It is a very robust library that enables us to carry out all network based processes such as generating the desired packets in all network layers [15]. All incoming to and outgoing traffic to or from the server were thus sniffed, after which they were subject to parsing. The data acquired after parsing were submitted to the web application via WebSockets.
3) Scikit-learn library was used in the ML side. Thanks to the Term Frequency–Inverse Document Frequency (TF-IDF) class in this library, the text expressions were digitalized in accordance with the TF-IDF model [16]. Therefore, it was transformed into a matrix form that can be used by classification algorithms.
4) Pandas library was used to carry out easier operations on the dataset.
5) Pickle library was used for recording the model to ensure that the model is not re-trained several times after the first training after which this model was used for the required operations.
6) Javascript was used intensively in the web application side of the project. On the Python side, JavaScript was used to fetch the data submitted over the websocket thanks to the tornado library. Required graphs were generated to vi-

sualize the incoming data thanks to the chart.js, one of the Javascript libraries.
7) In addition, streaming plug-in was used for the chart.js library since incoming data were stream data. Thus, streaming data transfer was also displayed graphically in real-time.

B. Hypertext Transfer Protocol (HTTP)

Significant work has been done in recent years on the characterization of http traffic [17]. HTTP is a stateless request/response protocol. This structure of the http protocol is not suited for web applications that in a significant interaction. A new connection to the server would be generated for each request before Http 1.1. Thanks to Http 1.1, it is now possible to use a single TCP connection for multiple request/response pairs. This new structure is known as http-keep alive or http persistent connection. Http is also a half-duplex, which is a single directional protocol [18].
When considered from this perspective, the http protocol is based on straightforward elementary straightforward rules (but includes small complex systems for the execution of these rules). Fundamentally, http is based on client and server architecture. There is one response for each request in this architecture.
The majority of the protocols that have been developed since the time that the internet started spreading globally do not fully meet our needs in many aspects. Therefore, large technology companies the academic community is trying by way of innovations, arrangements and additions to take the current protocol up to a level that may meet the demands.
In the beginning, data transfers via HTTP (for instance, surfing a web page) resulted in a lot of undesired traffic in large systems. Because it was known that more than half of the systems traffic was due to the re-request of the unchanging data from the server. For instance, the whole page had to be refreshed to see a minor change in only a small part of the web page. Meaning that the whole page was requested from the server once again. Even though the change on the web page resulted in traffic that is less than a kilobyte, refreshing the whole page resulted in traffic that reached up to megabytes. The methods developed for solving this issue have been explained below from past to present.
At first, a method known as polling was used to solve this issue. In this method, the client sent a request to the server at certain intervals. For instance, a request was sent to the server every 30 seconds, and thus it became possible to learn with 30 second delays whether there are any changes in the data or not. However, this structure had its disadvantages. Increasing the number of requests to be made at specific periods in order to decrease the delay time also meant increasing traffic. Sending a request every 3 seconds generated undesired traffic even when there was no change in the system.
Thousands of unnecessary requests were sent in the polling method even when data did not change for hours. This caused significant issues especially for larger systems. Long-polling structure was developed in order to improve this system. In this system, when a request is made from the client to the server for checking whether there is any change in the data or not, the server holds the request and waits for the information to be available if there is no change in any data instead of sending an empty response as in data and once the information becomes available, a complete response is sent to the client. This was relatively better in comparison with the polling method; however, the load would increase significantly on the server side. Even this method was still based on the request and response architecture and the connection operated as half-duplex.

C. Websockets

Efforts are made continuously to develop better systems since the methods above for developing real-time applications are problematic. This process will repeat continuously as the demands change or improve. Because they may provide ideal solutions to our needs in many areas. Instant messages can be used everywhere from stock exchange to wherever there is a need for instantaneous data transfer. Websockets have significant characteristics that set them apart from the others [19].
First of all, the server cannot notify the client when there is a change in the server; however we are notified of this change only when the client sends a request to the server. Websockets transform the half-duplex communication between the server and the client to full-duplex. Thus, it could be possible to transfer data from the client to the server and vice versa. The server may send data to the client even when there is no request from the client side. Because the half-duplex structure of WebSockets has not been taken as basis. Such a structure has been able to provide the most practical solution to many issues. The client does not send unnecessary and continuous requests to the server, and methods that increase the server's workload are not used. The related methods are triggered in case of a change in data, and this information is sent to the client. Websockets result in a traffic expressed in bytes that can provide real-time data transfer with meager traffic even in the largest systems.

D. Synchronous and Asynchronous Programming

The codes of the software are executed in order in synchronous programming. The next step cannot start before the previous step is completed. Synchronously developed software that carries out disk, network input/output operations end their tasks very late. This is because the following code blocks cannot be executed before operations such as reading data from the disk or recording data to the disk are completed. Thus, the waiting time results in significant issues after some time. For instance, all other operations at that time are blocked when asynchronous web server is going to respond to a request meaning that no other operations are allowed. It is very fast to send a request from the client to the server. However, the main issue here is that a delay occurs after the request is sent before the response is received. This is the underlying reason why such software is late in responding.

There is a high probability that the system will be out of service during intensive operations in large systems if there is any synchronous software involved. Even though it changes from system to system, there will be requests sent by several hundred users at the same time intervals. The server is blocked for several milliseconds if it is responding to an individual request. It takes on the next request after responding to the first. It continues to operate in this manner. This blockage issue will not cause any problems in systems with a small number of users. However, the delays will add up to seconds and even minutes in large systems when thousands of requests are made simultaneously thereby making the system unable to operate. Asynchronous programming model was developed to solve this issue [20]. With this model, reading/writing data or receiving/sending packet may take place more rapidly without being blocked.

Indeed, processes take a very little amount of time thanks to asynchronous libraries that operate with a single thread and a more significant number of processes can be performed in the same amount of time. An asynchronous application performs other processes when it will send a request to a remote server instead of waiting for the response. After receiving the remote server's response, it returns back and continues from where it left off. In the asynchronous model, if data will be recorded on the drive or if a packet will be sent and received in network processes, other processes are performed until the response is received following the I/O command. This indicates that the next code blocks may be performed before finishing off with the previous code block. When the response is received from the related location (disk, network), a notification is sent via the callback method indicating that the process is completed. An interrupt is sent. Upon receiving this interrupt, the software starts processing the received response. Thus, the waiting time in reading/recording processes is put into use in some other processes. Because usually there is no need for the CPU in such processes. The same thread can perform other processes until the response is received while the processes are run via DMA.

III. EXPERIMENTAL RESULTS AND DISCUSSION

There is a need for a trained system in order to apply the ML models [9] on systems that operate in real-time. Because it may be necessary to make instantaneous decisions regarding real-time data.

In this project, term frequency-inverse document frequency (TF-IDF) was used, which is used frequently in text processing and natural language processing. All URL addresses in the dataset were digitalized by way of TF-IDF since the machine learning model will make decisions via Uniform Resource Locator (URL).

After this stage, the model was serialized and recorded as binary. Thus, the issues encountered when running the project more than once were eliminated such as training the system repeatedly and waiting time. The model trained at the start was recorded which was then imported directly in the subsequent runs.

Logistic Regression (LR) model Islam et al. [21] was used in this study which is among the favorite models of machine learning.

Contrary to its name, LR is not a regression model. It is a classification model. It is also a classification model that is frequently used in the industry that is easy to use with quite a good performance on linearly classifiable classes.

The idea behind the LR model which is a probabilistic model, is based on the logit function. The logit function is simply based on the probability ratio [22]:

$$\text{logit}(p) = \log \frac{p}{1-p} \tag{1}$$

Here, p denotes the positive probability. Logit function takes on values that vary between the interval of 0 and 1 and transforms them into real numbers:

$$\text{logit}(p(y=1 \mid x)) = w_0x_0 + w_1x_1 + w_mx_m = \sum_{i=0}^{n} w_mx_m = w^Tx \tag{2}$$

Here, $p(y = 1 \mid x)$, x represents the conditional probability for which the property vector has been given and that belongs to class-1. It is the logistic or the sigmoid function that estimates the probability of a certain sample to belong to a certain class that is the inverse form of the logit function:

$$\Phi(\mathrm{v}) = \frac{1}{1+e^{-v}} \tag{3}$$

Here, v represents the system input as the linear combination of the weights and inputs and can be calculated as such:

$$v = w^T x = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m \tag{4}$$

The URL dataset acquired from the Kaggle website [23] was used in machine learning. The dataset with a single input property vector and a single output vector contained 420.464 records. As shown in Figure 2, the values of newton-cg, lbfgs, liblinear, sag and saga were used respectively in the LR model for the solver parameter. In this case, success rates of 96.462%, 96.496%, 96.462%, 96.462% and 96.462% were acquired respectively. Better results were obtained with the lbfgs value of the solver parameter even if the difference was minor. In addition, as shown in Figure 3, the same results were obtained with a success rate of 96.21% as a result of the training carried out by varying the newton-cg and max_iter parameters between 200 and 1600.

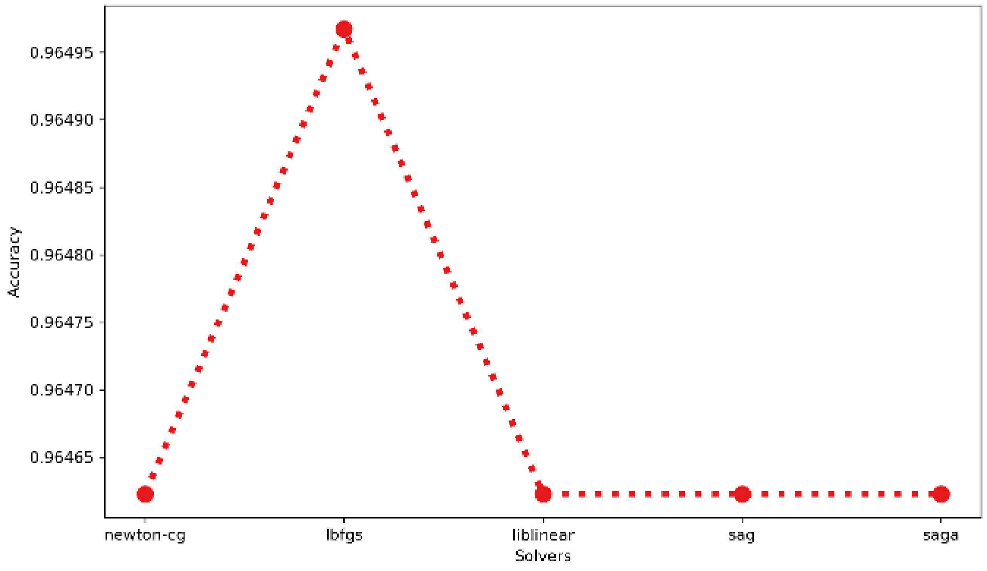

Fig. 2. Solver parameter values for system performance [14]

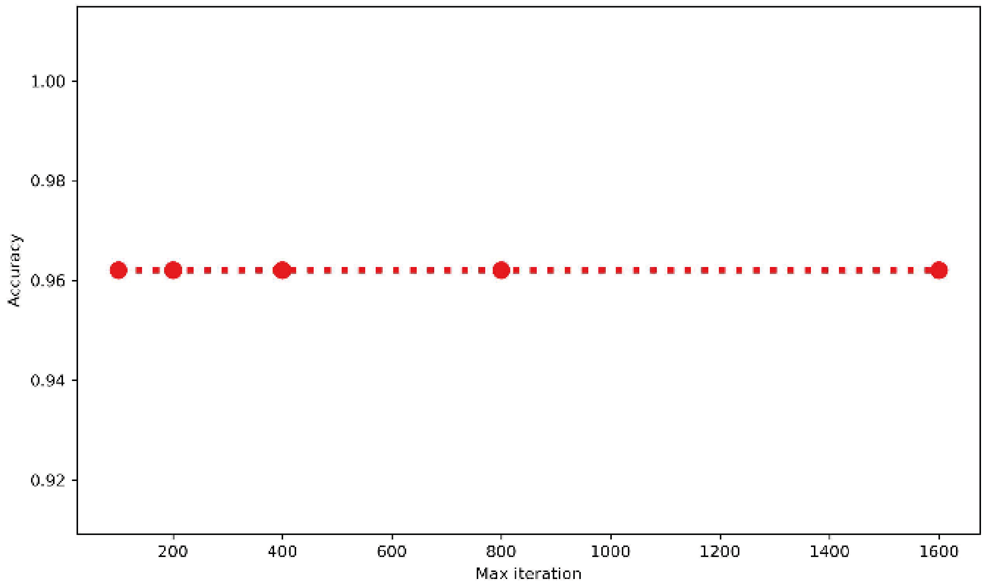

Fig. 3. Max iter parameter values for system performance [14]

ISSN: 2414-4592
DOI: 10.20474/jater-6.1.4

A better success rate of 96.92% was attained by using the Decision Tree model on the same dataset which is quite slower than the LR.

As shown in Table 1, the classification performances obtained in our study are better than the results obtained from previous studies.

TABLE 1
COMPARISON OF OVERALL ACCURACY FOR CLASSIFICATION

Reference Paper	Dataset	Methodology and Classification Average
Kurt et all. [24]	Coronary artery disease	LR (91.8%)
Abu-Nimeh et all. [25]	Spam email database	LR (95.11%)
Our study	The URL dataset LR (96.21%),	Decision Tree (96.92%)

Figure 4 shows all of the real-time http traffic that was received while the system was operating, while Figure 5 shows the suspicious http traffic analyzed in real-time via machine learning support.

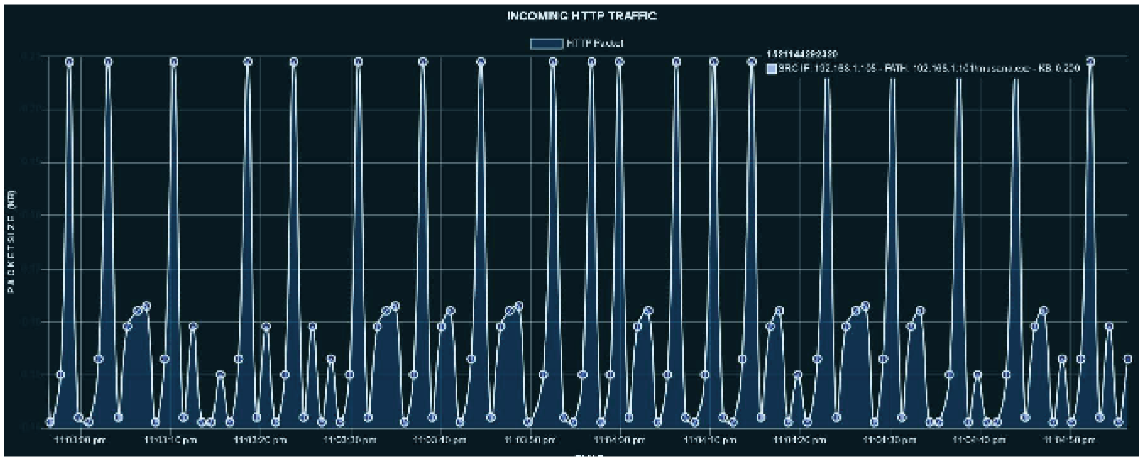

Fig. 4. The real-time http traffic [14]

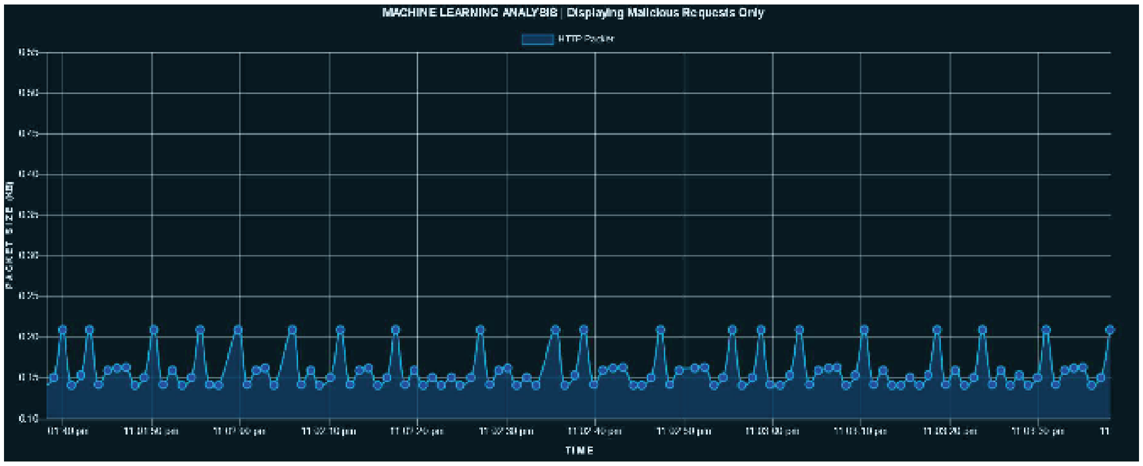

Fig. 5. The suspicious http traffic analyzed in real-time via machine learning support [14]

IV. FUNDAMENTAL DIFFICULTIES IN THE STUDY

There is no doubt that the most fundamental problem in real-time developed applications [26, 27, 28] is to present the properly processed data via user-friendly interfaces. However, things may get quite complicated at this stage if architecture is not known well. We faced several main issues at this point, which gave us a hard time [29].

The first difficulty was sending the parsed packets captured by way of sniffing the network instantaneously [30, 31] to the web application and displaying them as graphs. Because no other process can be performed when the network is sniffed. Even though it seems as if this problem can be solved by way of multiprocessing or multithreading, an unexpected situation developed. The probability of facing deadlock and race conditions was quite high since the threads operate in the same process space and many processes will take place in the same memory space when the number of sniffed packets increases.

The reason for not choosing Multiprocessing was that it would complicate things further, resulting in performance problems. Even though it seemed quite logical and high-performance to work with two processes, it could not solve our issues properly since it would not be possible to synchronize the processes when a single process was to be performed. So there was only one other option left. The project would have to be developed asynchronously. Thus, better results would be acquired thanks to an asynchronously operating structure over a single thread. Therefore, the project was developed asynchronously.

The second problem was waiting for the web application to process the data while sending the packets captured when the network is sniffed to the web application. In other words, the issue was that a second network packet was not processed before the response is received after the processing of the first received network packet.

Such processes operate synchronously as a matter of architecture. Synchronous processes operate with one request, one response method. The second request cannot be sent until the response is received for the first request sent. Here, the second main issue was also resolved by using asynchronous programming.

Python is an interpreted language and due to their nature, interpreted languages have to repeat all processes from the start when operated. Time is significant for a real-time application. It was a major issue that the system had to spend some time to re-train itself at each run. The solution to this issue was recording the serialized model in binary format which was generated after the training of the system using the training data. Thus, there was no need for an additional waiting time for re-training the system since it would operate using the recorded model.

V. CONCLUSION

In this study, scapy, sklearn and pickle libraries were used, which are used frequently for ML in Python programming language. Websockets were utilized in order to analyze web traffic faster and Asynchronous programming was used to prevent delays. Two critical metrics of stability and performance were given priority for the real-time operation of the Attackdet application that combines data separation and real-time data classification. Thus, it would be possible to get successful results even with the weakest hardware. For this purpose, an asynchronous server was developed using an asynchronous library. Thus, we tried to prevent blocking even in cases of high HTTP traffic. Logistic regression model from among the machine learning models was preferred due to its speed and popularity and a quite satisfactory classification success rate of 96.49% was obtained with the URL dataset.

The dataset used for the proposed system is not long enough. Therefore, it does not seem possible to analyze all web content with this. For the proposed system to analyze web traffic efficiently in any public institution, a larger-sized training dataset should be preferred. Also, the server that will do this job online must have perfect hardware and software configuration.

REFERENCES

[1] E. Alpaydin, *Machine Learning: The New AI*. New York, NY: MIT Press, 2016.

[2] S. K. Gouda *et al.*, ``Smart traffic management system using iot and machine learning approach,'' 2020. [Online]. Available: https://bit.ly/3hI91eF

[3] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, ``A review on machine learning based approaches for internet traffic classification,'' *Annals of Telecommunications*, vol. 75, no. 11, pp. 673-710, 2020. doi: https://doi.org/10.1007/s12243-020-00770-7

[4] M. Amrollahi, S. Hadayeghparast, H. Karimipour, F. Derakhshan, and G. Srivastava, ``Enhancing network security via machine learning: Opportunities and challenges,'' in *Handbook of Big Data Privacy*. Berlin, Germany: Springer, 2020.

[5] O. A. Osahenvemwen and O. F. Odiase, ``Effective management of handover process in mobile communication network,'' *Journal of Advances in Technology and Engineering Studies*, vol. 2, no. 6, pp. 176-182, 2016. doi: https://doi.org/10.20474/jater-2.6.1

[6] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, ``Corrauc: A malicious bot-iot traffic detection method in iot network using machine learning techniques,'' *IEEE Internet of Things Journal*, vol. 57, no. 5, pp. 56-70, 2020. doi: https://doi.org/10.1109/JIOT.2020.3002255

[7] T. Shelatkar, S. Tondale, S. Yadav, and S. Ahir, ``Web traffic time series forecasting using ARIMA and LSTM RNN,'' in *ITM Web of Conferences,* California, CA, 2020.

[8] Z. PALA and O. Özkan, ``Artificial intelligence helps protect smart homes against thieves,'' *Dicle University Faculty of Engineering Journal of Engineering*, vol. 11, no. 3, pp. 945-952, 2011. doi: https://doi.org/10.24012/dumf.700311

[9] Z. Pala and R. Atici, ``Forecasting sunspot time series using deep learning methods,'' *Solar Physics*, vol. 294, no. 5, pp. 50-34, 2019. doi: https://doi.org/10.1007/s11207-019-1434-6

[10] Z. Pala, ``Using forecasthybrid package to ensemble forecast functions in the r,'' 2019. [Online]. Available: https://bit.ly/3nfPWl2

[11] H. Unluk and Z. Pala, ``Prediction of monthly electricity consumption used in Mus Alparslan university complex by meansof classical and deep learning methods,'' in *Conference on Data Science, Machine Learning and Statistics,* Van, Turkey, 2019.

[12] M. R. Islam, T. K. Koirala, and F. Khatun, ``Network traffic analysis and packet sniffing using UDP,'' in *Advances in Communication, Devices and Networking*. New Jersy, NJ: Springer, 2018.

[13] N. Ugtakhbayar., B. Usukhbayar, S. H. Sodbileg, and J. Nyamjav, ``Detecting TCP based attacks using data mining algorithms,'' *International Journal of Technology and Engineering Studies*, vol. 2, no. 1, pp. 1-4, 2016. doi: https://doi.org/10.20469/ijtes.2.40001-1

[14] P. Joshi, J. Hearty, B. Sjardin, L. Massaron, and A. Boschetti, *Python: Real World Machine Learning*. Birmingham, UK: Packt Publishing Ltd, 2016.

[15] R. Montante, ``Using scapy in teaching network header formats: Programming network headers for non-programmers,'' in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education,* Scotland, UK, 2018.

[16] A. Dhar, N. S. Dash, and K. Roy, ``Application of TF-IDF feature for categorizing documents of online bangla web text corpus,'' in *Intelligent Engineering Informatics*. New York, NY: Springer, 2018.

[17] P. Jiang, F. Liu, H. Wang, and C. Li, ``Characterizing http traffic of mobile internet services in provincial network,'' in *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics,* Rome, Italy. IEEE, 2014.

[18] T. M. Tukade and R. Banakar, ``Data transfer protocols in IoT—An overview,'' *International Journal of Pure and Applied Mathematics*, vol. 118, no. 16, pp. 121-138, 2018.

[19] J. Newmarch, *Network Programming with Go: Essential Skills for Using and Securing Networks*. California, CA: Apress, 2017.

[20] R. Von Hanxleden, T. Bourke, and A. Girault, ``Real-time ticks for synchronous programming,'' in *Forum on Specification and Design Languages (FDL),* New Dehli, India, 2017.

[21] S. Chatterjee and A. S. Hadi, *Regression Analysis by Example*. Hokoben, NJ: John Wiley & Sons, 2015.

[22] S. Raschka, *Python Machine Learning*. Oxford, UK: Packt Publishing LTD, 2015.

[23] J. N. Van Rijn, V. Umaashankar, S. Fischer, B. Bischl, L. Torgo, B. Gao, P. Winter, B. Wiswedel, M. R. Berthold, and J. Vanschoren, ``A rapidminer extension for open machine learning,'' in *RapidMiner Community Meeting and Conference,* California, CA, 2013.

[24] I. Kurt, M. Ture, and A. T. Kurum, ``Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease,'' *Expert Systems with Applications*, vol. 34, no. 1, pp. 366-374, 2008. doi: https://doi.org/10.1016/j.eswa.2006.09.004

[25] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, ``A comparison of machine learning techniques for phishing detection,'' in *Proceedings of the Anti-Phishing Working Groups 2nd Annual E-Crime Researchers Summit,* Pittsburgh, PA, 2007.

[26] J. Kim, E.-Y. Park, B. Park, W. Choi, K. J. Lee, and C. Kim, ``Towards clinical photoacoustic and ultrasound imaging: Probe improvement and real-time graphical user interface,'' *Experimental Biology and Medicine*, vol. 245, no. 4, pp. 321-329, 2020. doi: https://doi.org/10.1177/1535370219889968

[27] T. Ahmad and H. Chen, ``A review on machine learning forecasting growth trends and their real-time applications in different energy systems,'' *Sustainable Cities and Society*, vol. 54, pp. 102-110, 2020. doi: https://doi.org/10.1016/j.scs.2019.102010

[28] L. Mishra, S. Varma *et al.*, ``Performance evaluation of real-time stream processing systems for internet of things applications,'' *Future Generation Computer Systems*, vol. 113, pp. 207-217, 2020. doi: https://doi.org/10.1016/j.future.2020.07.012

[29] A. Gupta, ``Analysis of real-time big data: its applications and challenges,'' *Journal of Data Mining and Management*, vol. 1, no. 2, pp. 1-10, 2016.

[30] M. Gregorczyk, P. Żórawski, P. Nowakowski, K. Cabaj, and W. Mazurczyk, ``Sniffing detection based on network traffic probing and machine learning,'' *IEEE Access*, vol. 8, no. 5, pp. 149 255-149 269, 2020. doi: https://doi.org/10.1109/ACCESS.2020.3016076

[31] D. Glăvan, C. Răcuciu, R. Moinescu, and S. Eftimie, ``Sniffing attacks on computer networks,'' *Scientific Bulletin" Mircea cel Batran" Naval Academy*, vol. 23, no. 1, pp. 202-207, 2020. doi: https://doi.org/10.21279/1454-864X-20-I1-027