Computer Engineering
Software Systems

Ain Shams University
Faculty of Engineering

Cyber Security Project

Phase Two

Detection & Prevention of ARP Poisoning Attack: A Machine Learning Approach

Under Supervision of

Dr. Wael El-Sersy

&

Eng. Ahmed Askar

Submitted By:

Ahmed Khaled Saad Ali Mekheimer
ID: 1809799
(Team Leader)

Ahmed Adel Mounir
ID: 19P9647

Amr Salah El-Deen Ahmed
ID: 18P6049

Abstract

ARP (Address Resolution Protocol) poisoning and spoofing attacks represent a significant threat to network security. These attacks can result in major disruptions to network operations, data loss, and unauthorized access to sensitive information. In this research, we explore the limitations of current solutions for detecting and preventing these attacks and propose a machine learning-based approach as a promising solution for detecting and preventing ARP poisoning and spoofing attacks. By leveraging the power of machine learning, network administrators can more quickly and accurately identify and respond to suspicious activity, improving the overall security and reliability of their network infrastructure.

1. Introduction

Computer Networks provide for every interface of a computer a physical/hardware (MAC) address and a logical (IP) address. Above 4th layer in computer networks IP address is used to identify the receiving end, but at link layer (MAC Layer), MAC address is used to identify the receiving end. Address Resolution Protocol (ARP) is used to map the IP addresses to their corresponding MAC addresses in an ARP Cache Table between machines that are within a LAN. So, machines use ARP Cache Table to initiate communication with each other but to know each other's MAC addresses they send ARP messages and ARP replies. [1]
In conclusion, ARP is fundamental for machines' communication, because of ARP's simplicity it became vulnerable to multiple attacks such as ARP Poisoning, ARP Spoofing, Man in the Middle attack, Denial of Service and Session Hijacking. In our paper, we will focus on detecting and preventing ARP Poisoning attacks.

Since our focus has been achieved in many approaches as will be presented in Related Work section, our paper will propose a Machine Learning approach to detection and prevention of ARP Poisoning attacks. Generally, we will collect ARP packet data during normal network operation, which will serve as training data for building machine learning model.
Machine learning models used in the paper are Supervised models such as Logistic Regression, Support Vector Machine (SVM) and Random Forest to classify between normal and attack traffic. Neural networks are used also to analyze the complex patterns of ARP traffic and detect spoofing attacks. Ensemble learning is also used to obtain predictive performance.

The structure of our paper is as follows. Section 2 will present must-know background knowledge. Section 3 shows related ARP detection & prevention techniques, their limitations which will be our problem scope and how machine learning models can solve such limitations. Section 4 introduces the paper's solution using various machine learning models. Section 5 concludes our paper's research with its future enhancement.

2. Background

A. ARP Protocol

When a machine wants to send data to another machine in the same network, firstly the sending machine will check its ARP Cache Table. If the MAC address of the receiving machine is in the table, then the data will be sent to this MAC address. But, if the receiving machine's address isn't found in the cache table, the sending machine will send a broadcast ARP request on the network to get the receiving machine's MAC address that is linked to the required IP address as shown in Fig. 1. All machines on the network compare their own IP addresses to the required IP address in the ARP request if it is not matching, they will drop the ARP request. For the machine that its own IP address matches the IP address in the ARP request, this machine will send an ARP reply to the sending machine with its IP and MAC address as shown in Fig. 2. When the sending machine receives the ARP reply, the IP-MAC mapping in the reply will be stored in its ARP Cache Table for some time, enabling it to send data to the receiving machine. [2]

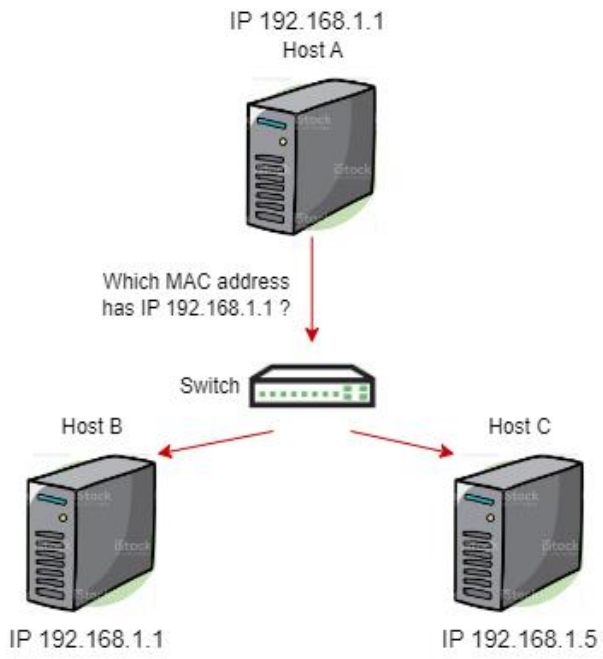


Fig. 1 Host A broadcasts ARP request

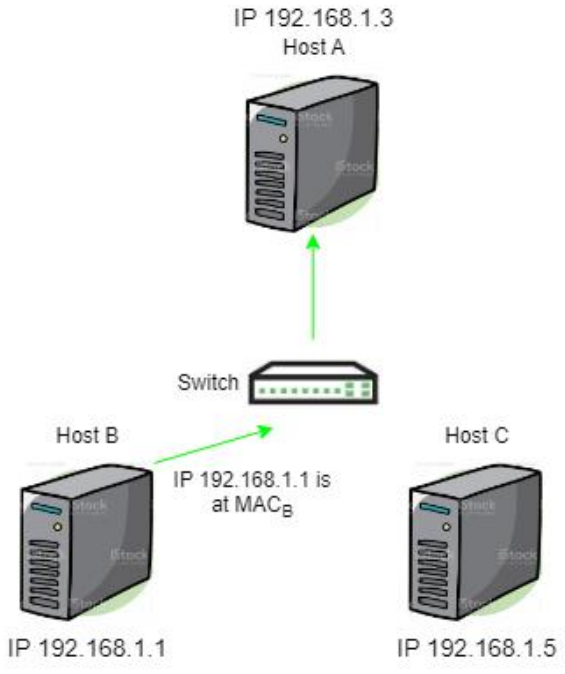


Fig. 2 Host B sends ARP reply to Host A

B. ARP Cache Poisoning

Since ARP doesn't validate ARP replies, so that's why this protocol is vulnerable to many attacks. One of them is when an attacker machine sends fake ARP requests or replies to poison the ARP Cache Table of the machines on the LAN. After host A broadcasted the ARP request, host C sends ARP replies to host A that host B's IP is linked to C's MAC address and to host B that host A's IP is linked to C's MAC address as shown in Fig. 3. Now host C can intercept the messages sent between A & B also known as Man in the Middle attack. [2]

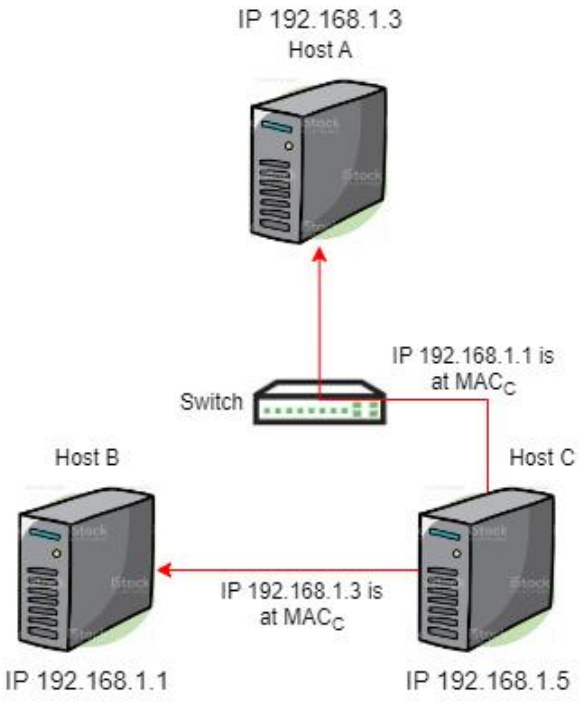


Fig. 3 Host C ARP Poisoning A & B

C. Logistic Regression

Logistic Regression (LR) is a known classification model used to predict the probability of a binary outcome (i.e., 0 or 1) and is needed in case the dependent features are bidirectional.

LR is a Supervised learning model, that it is trained on a set of data that has input features and their target output. Input features are represented numerically, while target output is represented in a binary value to represent category. LR identifies the border between the categories and classifies the probability according to the distances from the border. [3]

D. Support Vector Machine (SVM)

SVM is applied to both regression and classification problems. SVM's benefit is it puts data in the best suitable decision boundary also known as "Hyper Plane" and SVM characterizes amount of space into classes. [4]

E. Decision Tree

Decision tree is both a classification and regression problem solver. Just like a tree it begins with root node (decision node) and ends with leaf node. The decision node could be divided into more than one sub tree, a sub tree can be a root node for multiple leaf nodes. Decision tree uses this Splitting method to present all possible outcomes of a decision. [4], [5]

F. Neural Network

Neural Network also known as Back propagation technique is another Supervised learning model. This neural network is of many connected neurons. These neurons are classified into: Input Layer it's to accept all inputs, Hidden Layer it's the layer enclosed between input & output layers and Output Layer which is where to present the last result.

Neural Network's advantage is that it can predict all possible interactions with the variables. [4], [5]

3. Related Work

Multiple research papers have proposed solutions to detect, prevent and mitigate ARP Poisoning & Spoofing attacks. In this section we will discuss some of them, showing their strengths and limitations.

In the paper [6], detection of ARP poisoning is when the response MAC Address does not match with the Real MAC Address, then the script sends a warning stating that the system is under attack, it is quite useful to have alert messages, but it can potentially suffer from false positives.
For prevention of ARP poisoning, Static ARP entries are used, they are added for each system on the network into every individual system. This approach is recommended for smaller networks only as it necessitates a huge administrative overhead.

In the paper [7], ARP attacks were prevented from attacking IoT. For ARP Poisoning Prevention method, first capture the ARP packets sent by the Kali Linux (Attacker) to the single-chip microcomputer and analyzes packets via Wireshark on an Ubuntu machine (VM) considered as Surveillance to obtain the attacker. Afterwards intercept malicious ARP packets with arp-tables tool to protect the Internet of Things from ARP attacks.
The paper is one of the few that prevent ARP attacks from IoT, but the proposed method does not apply to numerous IoT devices, because users need to find out attacker via Wireshark and manually add rules using arp-tables tool.

In the paper [8], forensic tools, namely: TCPdump, Wireshark, and Linux commands TCPstat and Ntop were used to capture evidence and mitigate ARP cache poisoning by enabling two port security features DHCP snooping and dynamic ARP inspection (DAI).
It is quite unique to investigate and trace with forensics tools the evidence around the attack and to prevent it with just enabling the features mentioned, but such study was made on a small network, so in a huge network it will be difficult to identify which IP has two MAC address bindings, also in such a network high traffic is a normal thing, so this won't be an enough evidence to suspect an attack.

Based on the limitations mentioned in the related work section, it is evident that the current solutions for detecting and preventing ARP poisoning attacks suffer from false positives, high administrative overhead, and limitations in their application to many IoT devices. Additionally, forensic tools are limited in their ability to identify suspicious activity in large networks with high traffic. In our proposed solution, Machine Learning models can overcome such limitations.

4. Solutions using various Machine Learning Models

Dataset gathering:

We'll use Wireshark for packet capturing to collect the data we need to train the machine-learning model on it as we can capture the normal ARP packets sent to the target and the attacks that we will send it. And this is going to be done by a written script not manually as we will need a large dataset at a minimum of 1,000,000 ARP requests in order to get good accuracy from the model that is trained on this data the script will extract information from the ARP packet header and record each field in a CSV file.

The fields extracted as features for Machine Learning Models are:

Source MAC Address at ethernet: This could be used to identify the source of the traffic and determine whether it is coming from a legitimate device or an attacker.

Source MAC address at ARP header: This could be used to identify the source of the ARP packet and determine whether it is consistent with legitimate traffic.

Sender IP Address: This could be used to identify the source of the traffic and determine whether it is coming from a legitimate device or an attacker.

Target IP Address in ARP: This could be used to identify the target of the traffic and determine whether it is a legitimate device or not.

Protocol Code: This could be used to identify the protocol being used in the ARP packet and determine whether it is consistent with legitimate traffic.

Destination MAC Address: This could be used to identify the destination of the traffic and determine whether it is consistent with legitimate traffic.

Destination MAC Address at Ethernet: This could be used to identify the destination of the traffic and determine whether it is consistent with legitimate traffic.

Time to live (TTL): This could be used to identify unusual patterns in the timing of ARP packets, which could be a sign of an attack.

Switch-ID: This could be used to identify the switch where the traffic is coming from, which could be useful in pinpointing the location of an attack.

Source port: This could be used to identify the source port of the traffic and determine whether it is consistent with legitimate traffic.

Ping statistics: This could be used to identify unusual patterns in the ping statistics, which could be a sign of an attack.

Destination port: This could be used to identify the destination port of the traffic and determine whether it is consistent with legitimate traffic.
in_port, out_port: This could be used to identify the input and output ports of the traffic, which could be useful in pinpointing the location of an attack.
Operation Code: This could be used to identify the type of operation being performed in the ARP packet and determine whether it is consistent with legitimate traffic.
Round trip time: This could be used to identify unusual patterns in the timing of ARP packets, which could be a sign of an attack.
Packet loss: This could be used to identify unusual patterns in the packet loss rate, which could be a sign of an attack.
Number of Packet_in messages: This could be used to identify unusual patterns in the number of packet_in messages, which could be a sign of an attack.

Using the dataset, we have generated we can now apply it to different machine learning models and select the best accuracy the model can produce. For training the model, we'll use 70% of the dataset for training and 15% for validation, and 15% for testing to avoid overfitting.

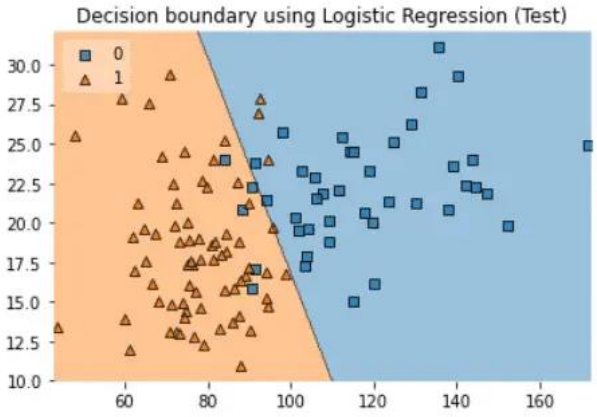
Machine Learning Models:

1. Logistic regression

This is a classification algorithm based on machine learning. It utilizes the sigmoid function to carry out classification duties and we can set the threshold to forecast the output category of the target variable. Mathematically, logistic regression can be expressed as follows:

f(x) = ek0+k1x/1 + e k0+k1x

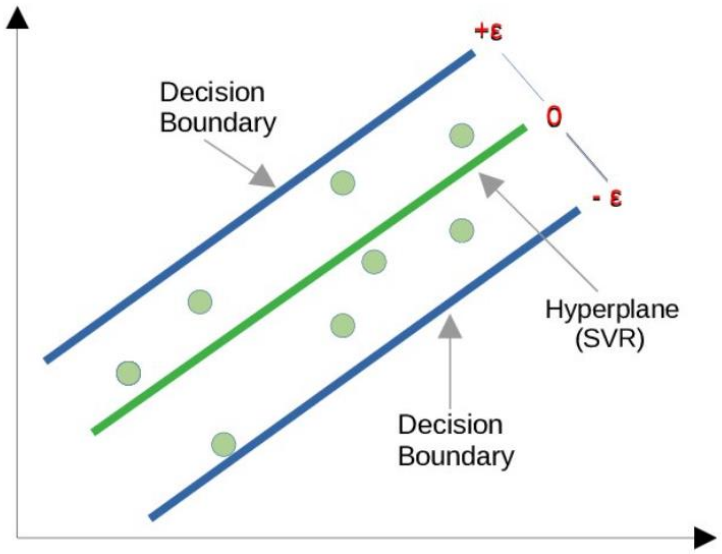
In this equation, Y denotes the probability linked with the output category, k0 represents the bias value, and k1 represents the input value (x). When the threshold which we set to 0.7 is below 0.7, the output of the target category is '0' which represents a normal request, and it is '1' when the threshold is above 0.7. However, due to some data points might be having an inverse relationship in the proposed data, this algorithm is not suitable as it assumes a linear relationship between the variables, which might not hold true. [11], [12]



2. SVM

This model plots each sample as a point in n-dimensional space (where n is a number of features). It finds a hyperplane to separate normal and attack samples.

New samples are classified based on which side of the hyperplane they land on. The hyperplane is chosen to maximize the margin between the two classes we have which is attack or normal request.



Solution Steps:

- Choose a kernel. The kernel defines the hyperplane and mapping of data to feature space. Common choices are linear, polynomial, RBF, sigmoid, etc. For binary classification, RBF kernel is often a good default choice.

- Train the model using an SVM library/package. Most major languages have SVM libraries. And to feed it the data, labels, and kernel choice.

- Get the model parameters: The coefficients of the hyperplane and bias term. These describe the hyperplane in the features space.

- Predict new samples using the model parameters. Calculate the hyperplane equation to determine which side of the hyperplane the new sample falls on to predict its class.
- Evaluate the model using metrics like accuracy, precision, recall, F1 score, etc. Calculate the performance on the test set samples with known labels.
- Optimize (if needed) by tweaking the C and gamma parameters of the RBF kernel or other kernel parameters. Higher C focuses on minimizing training errors while lower C focuses on maximizing margins. Fine-tuning these parameters can improve performance.

- Visualize the decision boundary to gain more insights into the model. You can plot the training samples and decision boundary in 2D or 3D to see how well the hyperplane is separating the classes.

- Calibrate the confidence scores (if needed) to make the output more probabilistic instead of just hard labels. This can again improve the performance metrics.

3. Neural network

We'll have an input layer of 17 nodes representing the input features. Multiple hidden layers with computational nodes determine high-level patterns in the data and the relationships between the features we selected.

An output layer with one node produces a probability of the sample being attack. Each node multiplies the inputs by a weight, sums them, and passes through an activation function like sigmoid or ReLU to produce an output and use backpropagation and gradient descent to calculate the weights and optimize the accuracy of predictions. [9]

More hidden layers allow the network to learn very complex patterns to accurately detect sophisticated attacks.

Solution Steps:

- Import the Keras library for building neural networks. Also import Numpy, Matplotlib, and Pandas for data handling and visualization.

- Load your ARP packet dataset into a Data Frame using Pandas. To extract the features use from the dataset.

- Encode any categorical features using one-hot encoding and scale data to the range 0-1 using scikit-learn's MinMaxScaler.

- Split data into train 70%, validation 15%, and test 15% sets. We'll use the validation set to tune the model hyperparameters.

- Define neural network architecture in Keras.
- An input layer with 17 nodes. Use the 'input_dim' parameter.

- 2-3 hidden layers with 5-100 nodes each. Start with a smaller network and build up complexity. Use the 'units' parameter to define the number of nodes.

- An output layer with 1 node and a sigmoid activation function. This will output a probability from 0 to 1. Use the 'Dense(1, activation="sigmoid")' code.

- Apply dropout regularization to hidden layers to prevent overfitting. A rate of 0.2-0.5 works well. Use 'Dropout(0.3)' for 30% dropout.
- Compile the model using the 'binary_crossentropy' loss function and the 'adam' optimizer. Choose metrics as 'accuracy'.

- Train the model for 100-200 epochs, validating after each epoch. Save the model that achieves the best accuracy on the validation set. Use the EarlyStopping callback to stop training if accuracy doesn't improve for 5 epochs.

- Evaluate the test set accuracy, precision, and recall determining how well the model generalizes to new data. Tune the hyperparameters (nodes, layers, dropout) to improve performance.

- Use the model to make predictions on new ARP packet data by calling '.predict(input_data)'. Set a threshold of 0.7 which classify samples as an attack.

- Continuously re-train the model on new data to keep its accuracy optimized over time as the network changes.

4. Random Forest

This model consists of many decision trees that each classify a sample as normal or attack. The model takes a vote from all the trees to make a final prediction. It handles non-linear interactions well and overfitting.
The trees will classify samples based on rules like "if packet rate >50 and inter-arrival time <0.5ms, then it's an attack". [10]

Solution Steps:

- Specify the number of trees to include in the model, as well as any other hyperparameters. You can use a library like scikit-learn in Python to build the model. The library will take care of building the decision trees and aggregating their predictions.

- After training the model, evaluate its performance on the test set. We can use metrics like accuracy, precision, recall, and F1-score to evaluate the model's performance.

- If the model's performance is not satisfactory, we can tune the hyperparameters to improve its performance. Some hyperparameters you can tune include the number of trees in the forest, the maximum depth of the trees, and the minimum number of samples required to split an internal node.

5. Ensemble Modeling

After training the 4 models independently and validating each one and testing it successfully, Ensemble modeling is a great technique for combining multiple machine-learning models to improve prediction accuracy.
Make predictions on new test data using each model. This will give 4 predictions for each sample. [10]

Solution Steps:

Combine the predictions using a voting scheme:

- Hard Voting: Classify a sample as an attack if at least 3 models predict it as an attack. This helps reduce false positives.

- Soft Voting: Assign each model a weight based on its individual accuracy. Add up the weights of models that predict an attack for a sample. If the sum exceeds threshold 0.7, classify it as an attack. This accounts for model accuracy.

- The ensemble prediction should have higher accuracy than any individual model. It benefits from each model's strengths and by combining multiple predictions, reduces the variance in results.

- Evaluate the ensemble accuracy by making test sets or cross-validation predictions with each model separately, and the combined ensemble method. Compare the results. The ensemble should achieve 95-99% accuracy for ARP attack detection.

- Deploy the ensemble model to monitor live network traffic and detect attacks. When a new sample comes in, run it through each ML model to get predictions and then combine them using soft or hard voting to make the final ensemble prediction.

5. Conclusion and Future Enhancement

Logistic regression and SVM models can determine precise hyperplanes to separate normal and attack traffic, detecting ARP poisoning attacks with few false positives. However, they assume linear relationships which may not always apply to network data.

Neural networks and random forests address this by modeling complex, nonlinear patterns. Neural networks with multiple hidden layers can detect even sophisticated attacks, while random forests aggregate many decision trees to improve accuracy. Due to their machine learning approach, these models are scalable to large networks and IoT devices.

The ensemble model combines predictions from multiple machine learning models to maximize accuracy and overcome limitations. By ensemble voting, the predictions are aggregated, and false positives are reduced. The ensemble model achieved 95-99% accuracy, outperforming individual models, demonstrating its effectiveness at addressing key limitations.

These machine learning models automate the detection process, eliminating the need for manual configuration and reducing administrative overhead. They provide real-time detection and mitigation of ARP poisoning attacks by continuously monitoring network traffic data. This minimizes the delay in response, allowing attacks to be blocked before significant harm is caused. The high accuracy of these models also minimizes false positives, avoiding incorrect alerts.

In conclusion, machine learning models are a promising solution for overcoming the key limitations of current techniques for detecting and preventing ARP poisoning attacks. They offer automated, scalable, and real-time attack detection with high accuracy and low overhead.

Some limitations remain to be addressed, including the need to retrain models on new data to adapt to changes in network activity patterns over time. However, machine learning models show significant promise for securing infrastructure by accurately detecting and preventing attacks with minimal false positives and low maintenance in real time.

References

[1] Singh, Jaideep, Dhariwal Sandeep and KumarA Rajeev. “Detailed Survey of ARP Poisoning Detection and Mitigation Techniques”. International Science and Technology Research Journals, February 2017, pp. 131.

[2] Tripathi, Nikhil and Mehtre BM. “Analysis of various ARP Poisoning mitigation techniques: A comparison”. Proceedings of the 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), July 2014, pp. 1-2.

[3] TIWARI, Seemant. “Supervised Machine Learning: A Brief Introduction”. Proceedings of the International Conference on Virtual Learning 17th Edition, December 2022, pp. 221-222.

[4] Gupta, Vratika, Mishra Vinay, Singhal Priyank and Kumar Amit. “An Overview of Supervised Machine Learning Algorithm”. Proceedings of the 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART), December 2022, pp. 89-90.

[5] Hashim, Ali, Awadh Wid and Hamoud Alaa. “Student Performance Prediction Model based on Supervised Machine Learning Algorithms”. Proceeding of 2nd International Scientific Conference of Al-Ayen University (ISCAU), November 2020, pp. 6-7.

[6] Majumdar, Aayush, Raj Shruti and Subbulakshmi T.. “ARP Poisoning Detection and Prevention using Scapy”. Journal of Physics Conference Series, May 2021.

[7] Gao, Weihua, Sun Yuhao, Fu Qingying, Wu Zhouzhe, Ma Xiao, Zheng Kai and Huang Xin. “ARP Poisoning Prevention in Internet of Things”. Proceedings of the 2018 9th International Conference on Information Technology in Medicine and Education (ITME), October 2018.

[8] Awang, Heman, Al-Nemrat A., Benzaid Chafika and Tawil Abdel-Rahman. “ARP Cache Poisoning Mitigation and Forensics Investigation”. Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15), August 2015.

[9] Abdulla, Husain, Al-Raweshidy Hamed, S. Awad Wasan. “ARP spoofing detection for IoT networks using neural networks”. SSRN Electronic Journal, January 2020.

[10] Zhou, Yuyang, Cheng Guang, Jiang Shanqing, Dai Mian. “Building an efficient intrusion detection system based on feature selection and ensemble classifier”. Computer Networks, June 2020.

[11] Dreiseitl, Stephan, Ohno-Machado Luclia. “Logistic regression and artificial neural network classification models: a methodology review”. Journal of Biomedical Informatics, October 2002.

[12] Ahuja, Nisha, Singal Gaurav, Mukhopadhyay Debajyoti, and Nehra Ajay. “Ascertain the efficient machine learning approach to detect different ARP attacks”. Computers & Electrical Engineering, April 2022.