Security: Artificial Intelligence Crushes Intrusions

Wenxuan Zhang *Electrical and Computer Engineering*
University of Toronto
Toronto, Canada
wenxuanzhang.zhang@mail.utoronto.ca

***Abstract*—Intrusion detection has always been one of the important research areas of network security. Recently, researchers have become interested in using emerging artificial intelligence technologies to redesign intrusion detection systems. In this article, I survey some of the prominent research achievements on the use of artificial intelligence techniques for intrusion detection, compare their advantages and disadvantages, and discuss possible future research directions.**

***Index Terms*—Intrusion Detection, Machine Learning, Deep Learning**

I. Introduction

The Internet plays a vital role in modern society. According to statistics, the number of Internet users accounts for 66% of the global population [1]. Such a huge user base has also become a breeding ground for increasingly rampant cyber attacks. According to a report by the Anti-Phishing Working Group, the number of phishing attacks increased by 30% in just one quarter [2]. Every attack that slips through the cracks in intrusion detection can cause significant damage to the user [3]. Therefore, it is very necessary to propose an effective intrusion detection system.

Intrusion detection is one of the most common research problems in Internet network security. This problem involves designing a reliable technical framework that can reduce false alarm rates and resist unknown attacks while maintaining high detection accuracy [13]. The common working method of an intrusion detection system [14] is to scan the network and hosts and evaluate the current network activity. If there is an abnormality, it will issue an alarm and respond accordingly. However, this system still faces challenges in fulfilling its primary purpose of intrusion detection [13]. Many current intrusion detection system products still have high false positive rates, which places a huge additional burden on network security staff in actual working environments. Meanwhile, the hustle and bustle of cybersecurity staff can cause real, well-disguised attacks to go unnoticed. In addition, these intrusion detection systems rely heavily on preconfigured configuration files, so they tend to be slow and ineffective in the face of unknown attacks.

In order to solve this series of problems, many researchers have turned their attention to the already relatively mature technology machine learning, and combined machine learning technology with intrusion detection technology. Machine learning technology can learn useful information from massive security data and classify threat data. And if machine learning technology is deployed on the system for training in real time, it is possible to adapt to emerging attack methods and resist unknown attacks in a subtle way.

In this article, I present a survey of some recent research results involving the application of artificial intelligence techniques to intrusion detection with excellent results. The investigated intrusion detection systems based on machine learning technology are mainly used in the field of detecting phishing attacks and detecting DoS & DDoS attacks. I will outline their ideas and solutions, comparing the pros and cons. In this article, I will also discuss potential directions for future research.

II. Phishing Detection With Machine Learning

Phishing attacks are one of the most common intrusion on the Internet and one of the most harmful social engineering techniques. And because users often have insufficient understanding of phishing attacks, users often only realize they have been attacked by phishing attacks after they have already suffered losses [6]. Therefore, it is very important to detect phishing attacks and block them from the user terminal in advance. However, existing phishing attack detection technology uses static detection rules, which is no longer effective enough for highly dynamic phishing attacks [4]. As a new detection solution, artificial intelligence technology is designed to improve detection accuracy as much as possible and intercept all threats.

A. Phishlimiter

CHIN et al. [4] proposed an algorithm called Phishlimiter, which is based on deep packet inspection tech-

nology and SDN (Software-Defined Networking) technology, aimed at identifying and mitigating harmful traffic. The PhishLimiter framework consists of two network traffic channels: the STORE AND FORWARD (SLOW LANE) and FORWARD AND INSPECT (FAST LANE). In the STORE AND FORWARD (SF) method, when packets enter the switch, their information is retained, checked, and classified for analysis. If the packets are deemed malicious, they are directly discarded at the switch. In the FORWARD AND INSPECT (FI) method, the traffic is immediately forwarded to the destination, and its information copy is temporarily stored for inspection.

The Phishlimiter framework uses the PhishLimiter Score (PLS) to classify traffic. When a packet is received, Phishlimiter checks its PLS. If the PLS is greater than the threshold, the packet will be sent to the SF lane; otherwise, it will be sent to the FI lane. Phishlimiter uses an internal database to record received packets, including the source IP address, destination IP address, and PLS value. After the packets are classified into the aforementioned channels, they undergo detection, which is carried out using feature extraction technology and ANN (Artificial Neural Network) techniques.

Feature extractor of PhishLimiter extracts URL, HTML, and domain features of the communication flow, outputting them as a vector of 30 features. This process utilizes a caching module to store historical data, thus eliminating the need for feature extraction for repetitive URLs. Finally, the feature vector output by the feature extractor is input into a five-layer ANN for classification, and the ANN uses dropout techniques to further improve accuracy. Additionally, a key feature of PhishLimiter is its ability to dynamically change the SDN flow channels (SF or FI) to further ensure the detection of flows.

Compared to traditional static detection methods, PhishLimiter demonstrates outstanding classification accuracy. However, due to the use of the SF lane, inspection of PhishLimiter imposes additional overhead on the network, resulting in slight delays and performance degradation in network communication. Nevertheless, it provides an excellent benchmark for comparison with other phishing attack defense systems designed using machine learning.

B. FS-SDN

Wazirali et al. [5] used a similar approach to Phishlimiter, employing embedded SDN technology and a CNN algorithm based on deep learning networks for phishing detection. However, the implementation of FS-SDN in technical detail is more sophisticated, demonstrating significant and substantial differences compared to Phishlimiter.

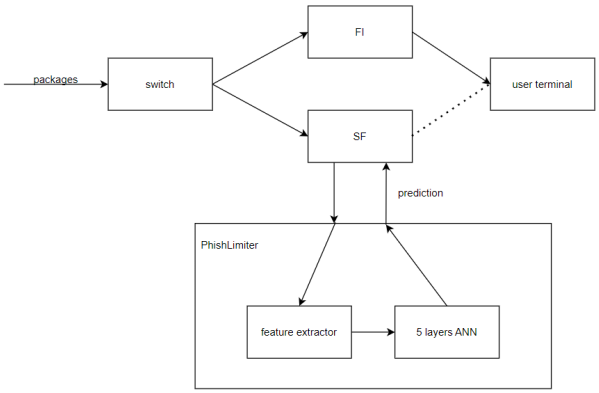

Fig. 1. Simple architecture of PhishLimiter

The FS-SDN framework consists of two main components: SD-Controller and SD-Switches. The SD-Controller is responsible for the CNN functionality, and the authors introduced a technique called FS-CNN for attack detection. Similar to Phishlimiter, FS-CNN also requires feature extraction of data before CNN training, but the execution steps differ significantly.

First, FS-CNN preprocesses the data for URLs, eliminating unnecessary ambiguous characters such as “http” , “https” and “www” . Additionally, all uppercase characters are converted to lowercase to ensure that uppercase characters represent the same meaning as their lowercase counterparts, effectively constructing an efficient feature matrix. After preprocessing, feature extraction is performed to output a feature matrix based on the number of characters in the URL. These feature matrices are normalized to further improve their quality. The final step of feature extraction involves feature selection, where the authors combined SVM and REF techniques to design the REF-SVM algorithm for the purpose of reducing the dimensionality of the feature matrix. The preprocessed feature matrix is then used as input for the CNN of FS-CNN, comprising convolutional layers, max-pooling layers, flattening layers, and dropout. The output of FS-CNN is sent to the SD-Switches component of FS-SDN.

The SD-Switches component serves as the classification layer, using the sigmoid function to categorize the feature maps. Data packets with malicious URLs are discarded, while those with normal URLs are forwarded to the user terminal.

Compared to Phishlimiter, the more sophisticated technical implementation of FS-SDN results in a significant performance improvement in attack detection. Considering the terrifying frequency of phishing attacks, the performance enhancement of FS-SDN in real-world

deployment could be particularly extraordinary. However, FS-SDN faces the same issues as Phishlimiter. It uses SDN to shift the detection process from the hardware layer of user to the controller layer. As a result, detection and training are carried out on the switch, which imposes additional overhead on network communication, leading to extra latency. Moreover, in FS-SDN, this additional latency also notably increases with the length of the URL.

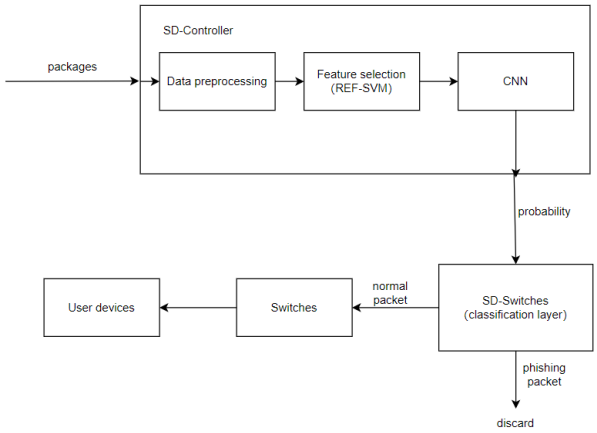

Fig. 2. Simple architecture of FS-SDN

C. PhishNot

Alani et al. [6] believe that most existing research on phishing attack protection focuses solely on accuracy, neglecting the practicality, ease of implementation, and accessibility of security measures. For instance, while Phishlimiter relies on the SDN terminal user environment, FS-SDN conducts online training on switches. Therefore, the authors propose a cloud-based detection system, PhishNot, which maintains a high accuracy rate and is designed with an API for easy access. Thus, PhishNot can be embedded as a plugin in common network communication environments.

PhishNot is primarily divided into two stages: the pre-deployment stage and the deployment stage. All training is completed in the pre-deployment stage, and after testing, PhishNot is stored in a cloud environment for deployment. Clients only need to send the URL to the cloud server to obtain predictions for data packets. The data preprocessing stage of PhishNot is more concise compared to FS-SDN. PhishNot eliminates invalid features (those that do not comply with domain name rules) and redundant features (those that can be deduced from other features). However, PhishNot uses the RFE method for feature selection, reducing the number of features to just 14, thus maintaining the ease of implementation of PhishNot. Finally, the reduced feature vector is used for the random forest classifier, which outputs and returns predictions for the URL, allowing the client to decide whether to block or allow access to that URL accordingly.

In terms of accuracy, PhishNot is lower than Phishlimiter. However, PhishNot is deployed on a cloud server, and the latest model is centrally deployed to the cloud infrastructure, allowing any user to access the latest service through the API. Therefore, PhishNot is indeed easy to implement and accessible. At the same time, PhishNot completes training in the pre-deployment stage, avoiding network communication delays similar to those caused by FS-SDN. A potential drawback of PhishNot is that during the real deployment stage, the client only sends the URL and receives predictions. This process is for prediction only and does not involve training, so PhishNot may feel inadequate when facing rapidly dynamic change phishing attacks without periodic and timely training of the ML model.

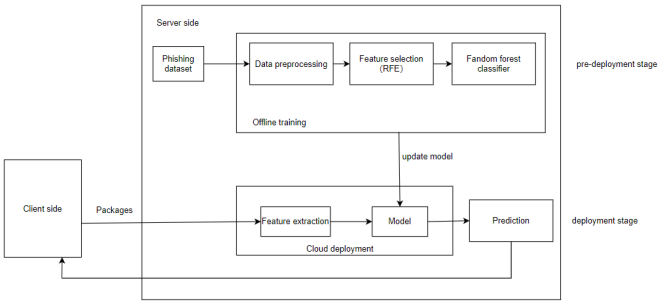

Fig. 3. Simple architecture of PhishNot

D. ICSA

Yoo et al. [7] observed that most defenses against phishing attacks on social networks detect attacks only after they have occurred. Therefore, the authors proposed a new approach to defend against phishing attacks, using NLP (Natural Language Processing) technology and AI chatbots to detect attacks while they are happening and provide victims with expert pre-defined defense advice. ICSA operates with Text-CNN and a chatbot collaborating in real-time. Text-CNN extracts individual chat messages for analysis, while the chatbot, based on the analysis results from Text-CNN, categorizes the stage of the ongoing attack and provides users with defensive suggestions. The hierarchical structure of the CNN model used in Text-CNN is similar to that used in FS-SDN, but the difference lies in the fact that Text-CNN is based on the NLP technology Word2Vec model,

which learns language and outputs probabilities, whereas FS-CNN learns the features of URLs and outputs labels.

The chatbot primarily categorizes attacks into four stages, with the alerted risks increasing with each stage, where Stage 0 represents the silent stage, and Stage 3 signifies the highest risk stage. The chatbot monitors real-time conversations of users and provides chat information to Text-CNN to obtain the current chat risk level in real-time, which decreases only when the chat is with a real acquaintance. During practical operation, when the analysis result indicates a medium to low risk level, the chatbot issues an alert and provides defense options. In the case of the highest risk level, it automatically reports to the local police station and saves the chat logs as evidence.

The experimental accuracy of the ICSA model is remarkably high. However, considering the lack of officially disclosed datasets of SNS phishing attack cases by law enforcement agencies, and the experimental dataset being collected by the authors themselves on SNS, the testing dataset of the ICSA model have a vulnerability, and the actual accuracy of the ICSA model still requires further confirmation. Another significant advantage of ICSA is its simple embedding and deployment in social software such as Telegram, enabling it to respond in real-time to dynamic attacks, making it easy to implement in real-world usage environments. The most apparent drawback of ICSA is that its Text-CNN analyzes individual chat sentences rather than a whole string of chat records, while real-world social network phishing attacks can be more complex. Attackers may manipulate victims through a long string of conversation, and analyzing individual chat sentences separately may not differentiate them from normal conversational language. Another potential flaw is that the ICSA chatbot reduces the risk level when users believe they are chatting with acquaintances. This decision is left to the user rather than the model, allowing phishing attackers to potentially bypass the ICSA model by impersonating identities through identity theft, and in real-life scenarios, attacks by acquaintances are also possible, rendering the ICSA model ineffective.

E. NLP & ML

Gualberto et al. [8] adopted a similar approach to ICSA. The authors aimed to determine whether emails contain malicious content through the analysis of their text content. They sought to address common issues encountered when using NLP technology to counter phishing attacks, such as the curse of dimensionality and sparsity, by using as few features as possible and

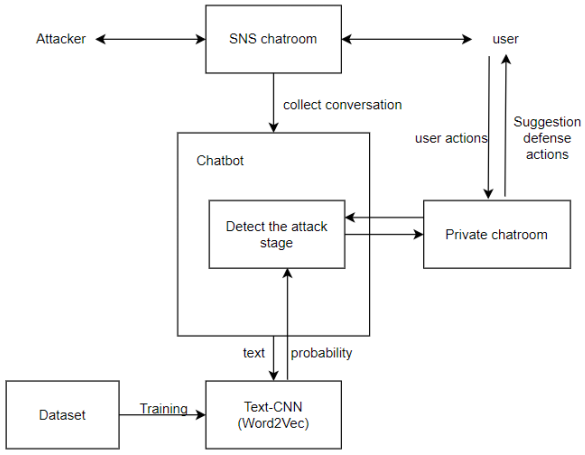

Fig. 4. Simple architecture of ICSA

minimizing computational costs for the identification task. Therefore, the authors proposed two combined techniques based on NLP and ML.

Both methods undergo a common initial step, which involves preprocessing the data using NLP techniques. During preprocessing, all characters are first converted to lowercase and punctuation is removed, followed by tokenization. Processed data undergoes lemmatization using the NLP tool Stanza, ensuring that all words are transformed into their dictionary form. The preprocessed data is then used to construct a Document-Term Matrix (DTM) through a bag-of-words model, where each term in the DTM is unique in the corpus and is accompanied by its frequency in the text. The DTM will be submitted to TF-IDF to convert the term frequency into importance weights. After this series of processes, the data undergoes feature extraction or selection, representing the primary distinction between the two combined methods proposed by the authors.

The two methods proposed by the authors are feature selection and feature extraction. Method 1, feature selection encompasses two technical perspectives: Chi-Squared or Mutual Information. Chi-Squared calculates the dependence of features on email categories, while Mutual Information determines how much information each feature can provide for classification. Method 2, two technical perspectives are PCA and LSA. PCA extracts important components, while LSA determines the topics in the document through relationships between terms in the text. When using Method 1, the feature dimension can be reduced to as low as 2 to 100, while Method 2 extracts 25 singular values as features. Finally, these matrices, after feature extraction, are input into an ML model for classification to determine whether the email contains malicious content. The authors tested a total of 8

ML algorithms: Support Vector Machine (SVM), Naive Bayes Classifier, Logistic Regression for classification, k-Nearest Neighbor, Decision Trees, Random Forest, XGBoost, and MLP.

Undoubtedly, the performance of this combined algorithm proposed by the authors is excellent. Notably, the combination of Method 2 LSA technique and the XGBoost algorithm achieves 100% accuracy and F1 score with only 25 features. Equally astonishing is that even when using only 2 features, the combination of Method 2 two technical perspectives and the ML algorithm can achieve 99.53% accuracy. Moreover, there is no concern about data vulnerability similar to that of ICSA in the results presented here, as the authors used public corpora datasets similar to or the same as those used by Phishlimiter, FS-SDN, and PhishNot. These results undoubtedly open up new prospects for researchers, suggesting that instead of engaging in a battle of wits with attackers over URLs, it might be more fruitful to directly explore suspicious elements within email texts.However, the authors only provide a discernment strategy tailored to phishing attacks, without offering a deployment architecture in the real world as provided in several frameworks mentioned earlier. Thus, the actual effectiveness of the proposed methods and the cost implications of their implementation require further research.

F. HSOHDL-PDC

Shaiba et al. [9] propose a model based on Hunger Search Optimization with hybrid deep learning capabilities to identify and classify phishing attacks. Since the probing object is URL, HSOHDL-PDC uses character-level embedding instead of word-level. In the data processing stage, the data is embedded into a fixed length using one-hot at the character level and sorted. The classification model used by HSOHDL-PDC is Hybrid CNN-LSTM Based Classification Model, which is characterized by its ability to long-term memory data, which can effectively avoid gradient explosion or gradient disappearance. During the calculation process, HCNN-LSTM will extract 1D-CNN features at the primary method layer and then forward them to other LSTM method layers for classification learning. Finally, the authors used the Hunger Search Optimization algorithm to optimize the hyperparameters of the HCNN-LSTM model. The accuracy of the HSOHDL-PDC model is similar to that of Phishlimiter. Compared with Phishlimiter and other models that use CNN algorithms, its main difference is the long-term data memory capability brought by LSTM. However, the method provided by the author does not have feature extraction and dimensionality reduction steps, which may cause the model to have too many parameters and require a lot of computational costs to train.

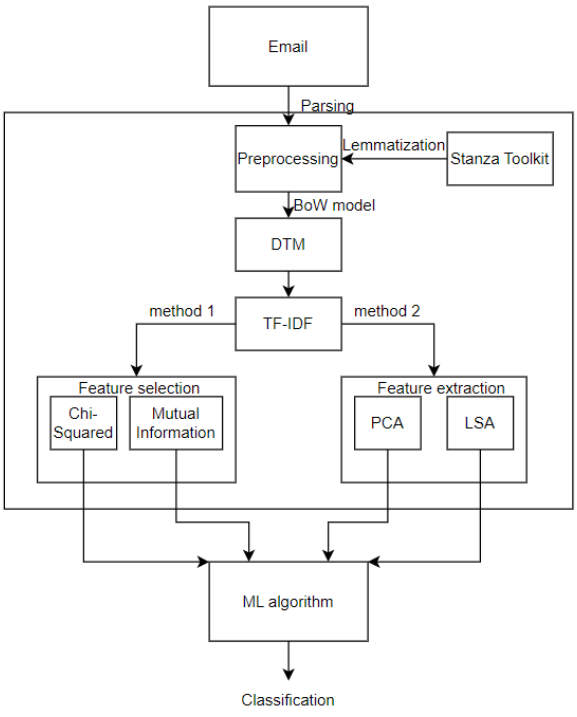

Fig. 5. Simple architecture of combined techniques

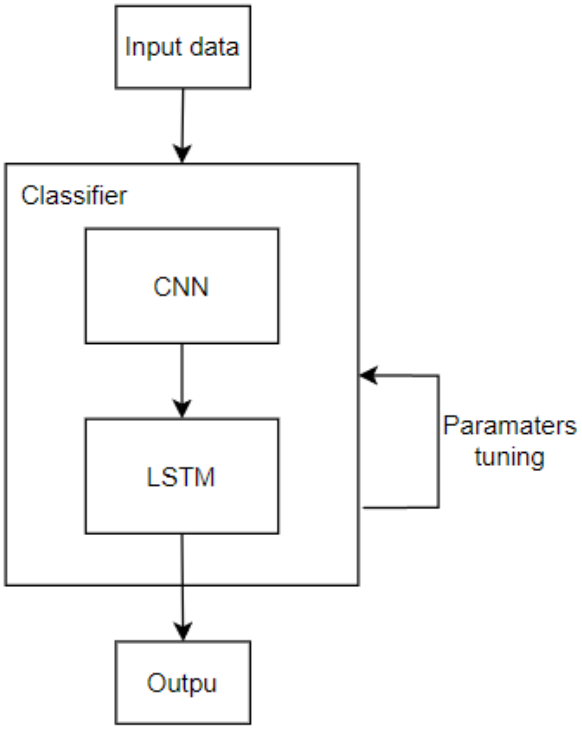

Fig. 6. Simple architecture of HSOHDL-PDC

G. SDN & EL

Miao et al. [10] proposes a flexible phishing website detection method based on SDN. Its main feature is that the detection module is separated and uses integrated learning methods and superposition learning methods. In this architecture, when a data packet arrives at the switch, the switch will check whether there is a matching incoming entry. If it matches, it will be forwarded directly. Otherwise, it will be submitted to the SDN controller of the architecture. The SDN controller will compare the URL of the data packet with the black and white list. If it is in the white list, it will be forwarded directly. If it is in the black list, the data packet will be directly discarded. When the URL is not in the black and white list, the data packet will be submitted to the detection module. The detection module consists of two Tiers. TIER-1 has three classification models that classify based on the URL characters of data packets, web page visual features and Web characters. Its output will be used as the input of TIER-2 for logistic regression training. And finally return the detection results and update the black and white lists.

The framework proposed by the authors is similar to Phishlimiter's idea. The main difference from Phishlimiter's SDN is that the framework proposed by the authors will manage traffic more intensively. In this framework, SDN will perform attack detection on all incoming data packets. In contrast, Phishlimiter uses PLS to analyze incoming data packets and forward them to the fast and slow lanes respectively according to the threshold. Therefore, it is obvious that this framework will bring The network overhead will be more than Phishlimiter. But the advantage that comes with it is that the framework proposed by the authors can ensure that all threats are checked, while in Phishlimiter PLS may mistakenly classify threat packets into the fast lane and avoid detection. Another advantage of this framework is flexibility. Since its detection module adopts an integrated learning method, it can flexibly expand or change the classification model used according to needs.

Table 1 summarizes the features and performance differneces across Phishlimiter, FS-SDN, PhishNot, ICSA and NLP & ML so on. And Table 2 shows weaknesses in the technical framework investigated. In addition, Figures 1 to 7 provide a simplified schematic diagram of the architecture of these detection frameworks. From the perspective of experimental results, the combined use of multiple artificial intelligence technologies is a method with low computational cost and high detection accuracy. But they may also bring about potential problems that need to be solved. Meanwhile, based on the above research, we can find that effective data preprocessing and appropriate feature extraction are crucial to accuracy and computational cost.

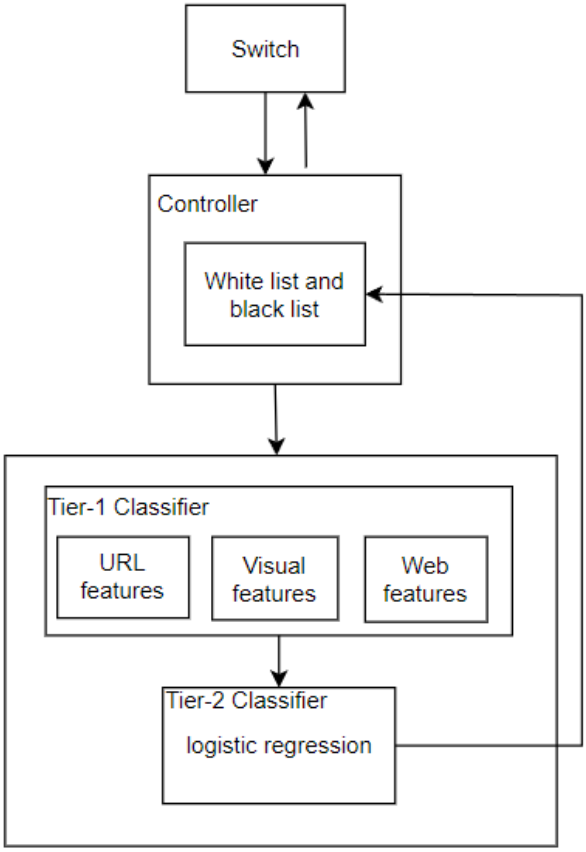

Fig. 7. Simple architecture of SDN & EL

	Techniques	Accuracy	Deployment
Phishlimiter	SDN+ANN	98.39%	SDN
FS-SDN	SDN+CNN	99.5%	SDN
PhishNot	Random forest	97.5%	Cloud based
ICSA	NLP+CNN	100%	SNS
NLP & ML	NLP+ML	100%	-
HSOHDL-PDC	CNN+LSTM	98.07%	-
SDN & EL	Ensemble Learning	98.61%	SDN

Table 1: Comparison different phishing intrusion detection framework.

	Defect
Phishlimiter	Additional network overhead
FS-SDN	Increased network overhead as URL length increases
PhishNot	Pre-trained models may not adapt to unknown attacks
ICSA	The real chat environment is too complex to fully analyze,
NLP & ML	Lack of actual deployment testing
HSOHDL-PDC	The cost of training time is huge
SDN & EL	Additional network overhead

Table 2: Comparison defect of different phishing intrusion detection framework.

III. DoS & DDOS Detection With Machine Learning

Through the above research literature, we understand that intrusion detection systems using machine learning technology have great potential in targeting phishing attacks. However, these intrusion detection systems all need to detect and classify email data packets or communication data packets. The inherent flaw of such solutions is that they often bring additional burden to the network.

Phishlimiter, FS-SDN, and PhishNot intrusion inspection architectures all require packet disassembly and analysis of their characteristics. ICSA and NLP & ML architecture requires NLP processing of text to analyze threats. These processes will occupy a certain amount of network resources. Moreover, they can only analyze and look for malicious information in these packets. If meaningless spam information such as frequent login requests is sent to the system, their policies will not be able to distinguish these attacks. They will pass all attacks and forward them to the endpoint after spending a certain amount of time and network resources analyzing the data of these attacks.

Therefore, when encountering huge batches of spam information attacks such as DoS and DDoS, such solutions will be ineffective and will instead accelerate the exhaustion of network resources, causing system crashes and business failure.

Fortunately, in recent research literature, a technical framework for applying machine learning technology to detect DoS and DDoS attacks has been proposed. These technical frameworks do not require the analysis of each data packet, which alleviates as much as possible the problem of additional network occupation by intrusion detection systems in the face of large-scale attacks.

A. SNORT & ZEEK

AbdulRaheem et al. [11] found that due to the centralized structure characteristics of SDN, it has become the main target of current DDoS attacks, especially the controller of SDN. Therefore, the authors proposed a technology based on machine learning technology to classify normal traffic and DDoS attack traffic to achieve the purpose of improving SDN security.

The technology proposed by the authors is a combination of ML-based SNORT software and ZEEK software. In this combined technology, Snort exists as an active blocker. Snort can perform traffic analysis and packet recording on IP networks in real time. It performs method analysis and content search and matching in conjunction with Extreme Gradient Boosting (XGBoost) technology for gradient boosting decision tree models to discover a wide variety of attacks and flag and block them. ZEEK can match signature-based tools to resolve security issues when a security alert is triggered. Therefore, ZEEK will be used in the background to combine with the Ryu controller and run as a network monitor and traffic analyzer. It can disperse the traffic to the Locke IDS awareness system and notify users of potentially harmful code that needs to be processed.

Compared with traditional technology and existing models, the performance of this combined technology is quite good. The correct performance in classifying normal traffic and attack traffic reaches 97.2%, and its performance varies depending on the specific type of attack traffic. When it is UDP Its performance can be maximized during food attack. But the potential problem is that the use of Snort and ZEEK will have a check time overhead, so this technology will cause a slight delay, which is mainly from ZEEK. And too intensive checking activity may cause computer resources to be exhausted, and large amount of RAM usage of ZEEK may lead to other potential problems. Therefore, based on the above characteristics and defects, this technical framework is only suitable for flood DDoS attacks, but cannot identify non-volume attacks, such as low-rate DDoS attacks, and only considers attacks targeting the control layer.

B. KNNPL-AOA

Liu et al. [12] found that currently wireless sensor networks (WSN), which are popular in the market, are very vulnerable to network attacks, especially DoS, due to their lightweight and lack of sufficient hardware resources. Therefore, solving their security issues is an important research direction.

Therefore, the author proposed an algorithm called KNNPL-AOA, which consists of two main modules: the

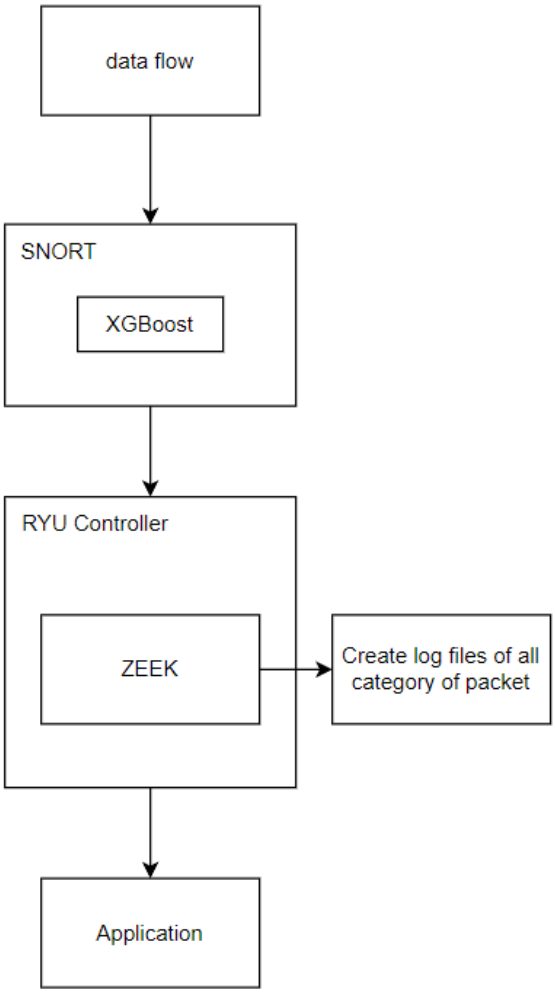

Fig. 8. Simple architecture of SNORT & ZEEK

PL-AOA algorithm for finding the optimal hyperparameters and the fine-tuned KNN. PL-AOA is a combination of levy flight algorithm and AOA algorithm. When looking for the optimal, it first divides the initialized relevant parameters into four groups and evaluates them within the group to find and update the historical optimal and global optimal. Every interval of a certain number of iterations, the algorithm will communicate between these groups and then select the best parameters to put into the levy flight algorithm and record the global optimal particle and its fitness value. The fine-tuned KNN with these optimal hyperparameters will then start training and classify which traffic is a network attack. The main feature of fine-tuned KNN is that it weights each coordinate subscalar, so coordinates with larger fluctuations have smaller weight coefficients than coordinates with smaller fluctuations. Another feature of the KNNPL-AOA technical framework is that due to the lack of sufficient hardware resources in WSN, the intrusion detection system will be deployed at edge nodes outside the WSN. WSN uses KNNPL-AOA to promote WSN intrusion detection by communicating with edge nodes.

The performance of KNNPL-AOA is quite excellent, and its accuracy can reach 99%. This is due to two reasons. The first is that the PL-AOA algorithm can effectively converge when looking for the best hyperparameters and avoid falling into a local optimum prematurely, so PL-AOA can effectively find the optimal solution. Secondly, the fine-tuned KNN weights the coordinates, so it can effectively avoid noise and false alarms. However, the KNNPL-AOA architecture also has obvious potential problems. Since WSN needs to use wireless communication to communicate with edge nodes to use KNNPL-AOA, and in actual deployment situations, the wireless communication environment may be quite complex and susceptible to interference, so KNNPL-AOA technology The performance of the architecture in actual deployment environments may be greatly affected. Compared with the way in which the AbdulRaheem [11] technology framework directly deploys intrusion detection software on hardware, it is unrealistic to perform intrusion detection in the same way only relying on the hardware conditions of WSN. Therefore, the intrusion detection deployment idea proposed by the authors may be a very effective alternative, but in order to make it truly meaningful for practical deployment, the solution of potential problems needs to be considered in future research.

C. mIDS

Ioannou et al. [15] proposed another idea of an online DoS intrusion detection technology architecture called mIDS. mIDS will be deployed at the network layer to perform statistical analysis on network topology nodes through machine learning technology, and classify these topology nodes into benign and malignant. mIDS consists of two main modules, namely the runtime monitoring tool (RMT) and the BLR detection module. RMT can monitor local sensor activity from the MAC and network layers at predefined intervals. When mIDS is in the training phase, it can collect local sensor activities in benign and malicious scenarios as training data for mIDS. When mIDS is deployed, RMT can provide mIDS with local sensor activity for detection. The BLR detection module consists of a binary logistic regression model that classifies these topological nodes into benign and malignant nodes based on the activity data of sensors in the topology. And in this process, the binary logistic regression model evaluates the importance of each variable in completing the identification task.

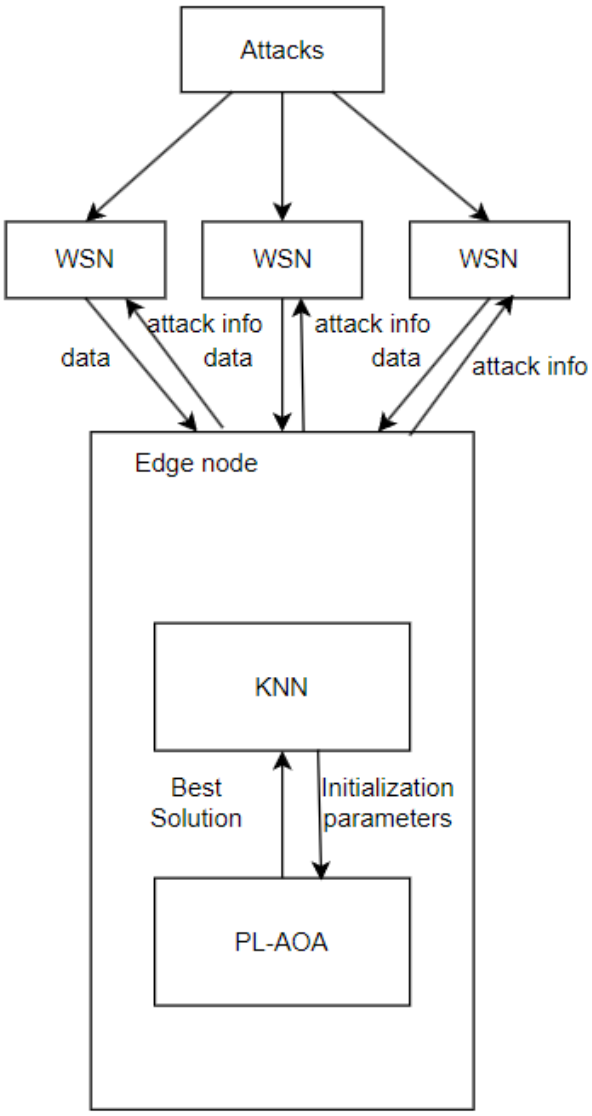

Fig. 9. Simple architecture of SNORT & ZEEK

Although mIDS is a lightweight intrusion detection system with a simple structure, its performance is still extremely excellent. The authors deployed mIDS on the Tmote Sky platform for performance measurement, and the accuracy rate was 96% to 100% depending on different scenarios. Unlike the above two technical frameworks, mIDS does not analyze classified traffic, but classifies network topology, so it does not need to consider the network delay overhead caused by detection activities. But correspondingly, saving time will consume space. mIDS will cause additional overhead to ROM and RAM memory, which is similar to the above-mentioned SNORT & ZEEK technology architecture. This will be a key factor in determining whether it can be deployed on the vast majority of WSNs. In addition, deploying mIDS will also cause slight additional power energy overhead. Although this overhead is not obvious enough in the experiment, according to different equipment and usage environments, if we study the issue of power energy overhead with similar ideas in the future, we must also pay attention to it. Compared with KNNPL-AOA, the main difference between mIDS and it is the deployment environment. KNNPL-AOA is deployed at the edge node, while mIDS is at the network layer. Both of them cleverly avoid the defect of WSN lacking sufficient hardware computer resources and open up an idea for WSN intrusion detection system. It may be an interesting topic for future research to compare the two approaches in actual deployment situations.

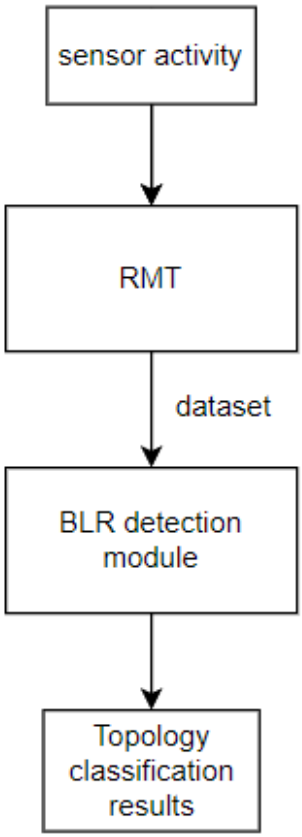

Fig. 10. Simple architecture of mIDS

Table 3 summarizes the applied ML technology and performance differneces across SNORT & ZEEK, KNNPL-AOA and mIDS. In addition, Figures 8 to 10 provide a simplified schematic diagram of the architecture of these detection frameworks. From the experimental results, it can be seen that the three technical frameworks can well resist DoS attacks, although they are developed through different deployment ideas. These

technical frameworks can well avoid the network overhead caused when detecting intrusions, but correspondingly they also pay a cost in space occupation. Table 4 will summarize the main shortcomings of these technical frameworks. At the same time, we can find that machine learning based intrusion detection systems may not need to use very complex machine learning algorithms to defend against DoS attacks, and it may be better to just use simple algorithms such as logistic regression or KNN. This is because complex algorithms may take up more space.

	Techniques	Accuracy	Deployment
SNORT & ZEEK	XGBoost	97.2%	SDN
KNNPL-AOA	KNN	99%	Edge Node & WSN
mIDS	Logistic Regression	96-100%	Network layer

Table 3: Comparison different DoS intrusion detection framework.

	Defect
SNORT & ZEEK	RAM consumption is too high
KNNPL-AOA	Edge nodes may be interfered with in real deployment environments
mIDS	Additional memory and power overhead

Table 4: Comparison defect of different DoS intrusion detection framework.

IV. Discussion

Intrusion detection has always been one of the popular areas in network security research. Recently, many researchers have begun to use intrusion detection systems based on machine learning algorithms to replace traditional intrusion detection systems that use configuration files to deal with different types of attacks such as phishing attacks and DoS attacks. For phishing attacks, researchers prefer to use complex algorithms to try to find malicious features in each data packet for judgment. For DoS attacks, researchers prefer to use simple machine learning algorithms to directly classify traffic and topology nodes.

The existing challenge is that although these intrusion detection systems based on machine learning technology show strong capabilities, there are still some potential problems that need to be solved. The first is that these technical frameworks often have additional overhead in some aspects depending on the deployment environment. Trade-offs are inevitable, but finding a good trade-off is still worth studying. Secondly, although these technical frameworks can solve the problem of high false positives in traditional intrusion detection systems, researchers do not have an accurate answer as to whether they can deal with emerging unknown attacks. This process can require researchers to spend a lot of time making measurements and trial and error. In addition, many machine learning components require good design, which involves algorithm selection, hyperparameter adjustment, data collection, and appropriate loss functions. This can make the design of an intrusion detection system more complex and time-consuming.

As one of the hot research directions in the future, studying the comparison of deployment environments of intrusion detection systems based on machine learning technology may be a very meaningful topic. The value of correct technical ideas may be greater than the value of applying more complex algorithms. Meanwhile, whether the machine learning component of the system is trained in real time or pre-trained is also a research issue worthy of attention. This is related to whether the intrusion detection system can still have a defensive effect when dealing with unknown attacks. With some luck, these may be the key to whether machine learning technology can be applied to real-life intrusion detection systems on a large scale.

References

[1] I. W. Stats. Internet Usage Statistics—The Internet Big Picture: World Internet Users and 2020 Population Stats. Accessed: Jun. 26, 2020. [Online]. Available: https://www.internetworldstats.com/stats.htm.

[2] "PhishTank," Cisco Talos Intelligence Group. https://www.phishtank.com/.

[3] A.K. Jain, B.B. Gupta, A survey of phishing attack techniques, defence mechanisms and open research challenges, Enterp. Inf. Syst. 00 (00) (2021) 1–39, https://doi.org/10.1080/17517575.2021.1896786.

[4] T. Chin, K. Xiong, and C. Hu, "Phishlimiter: A Phishing Detection and Mitigation Approach Using Software-Defined Networking," IEEE Access, vol. 6, pp. 42516–42531, 2018, doi: https://doi.org/10.1109/access.2018.2837889.

[5] R. Wazirali, R. Ahmad, and A. A.-K. Abu-Ein, "Sustaining accurate detection of phishing URLs using SDN and feature selection approaches," Computer Networks, vol. 201, p. 108591, Dec. 2021, doi: https://doi.org/10.1016/j.comnet.2021.108591.

[6] M. M. Alani and H. Tawfik, "PhishNot: A Cloud-Based Machine-Learning Approach to Phishing URL Detection," Computer Networks, vol. 218, p. 109407, Dec. 2022, doi: https://doi.org/10.1016/j.comnet.2022.109407.

[7] J. Yoo and Y. Cho, "ICSA: Intelligent chatbot security assistant using Text-CNN and multi-phase real-time defense against SNS phishing attacks," Expert Systems with Applications, vol. 207, p. 117893, Nov. 2022, doi: https://doi.org/10.1016/j.eswa.2022.117893.

[8] E. S. Gualberto, R. T. De Sousa, T. P. De Brito Vieira, J. P. C. L. Da Costa, and C. G. Duque, "The Answer is in the Text: Multi-Stage Methods for Phishing Detection Based on Feature Engineering," IEEE Access, vol. 8, pp. 223529–223547, 2020, doi: https://doi.org/10.1109/access.2020.3043396.

[9] H. Shaiba, J. S. Alzahrani, M. M. Eltahir, R. Marzouk, H. Mohsen, and M. Ahmed Hamza, "Hunger Search Optimization with Hybrid Deep Learning Enabled Phishing Detection and Classification Model," Computers, Materials & Continua, vol. 73, no. 3, pp. 6425–6441, 2022, doi: https://doi.org/10.32604/cmc.2022.031625.

[10] M. Miao and B. Wu, "A Flexible Phishing Detection Approach Based on Software-Defined Networking Using Ensemble Learning Method," Proceedings of the 2020 4th International Conference on High Performance Compilation, Computing and Communications, Jun. 2020, doi: https://doi.org/10.1145/3407947.3407952.

[11] M. AbdulRaheem et al., "Machine learning assisted snort and zeek in detecting DDoS attacks in software-defined networking," International Journal of Information Technology, Sep. 2023, doi: https://doi.org/10.1007/s41870-023-01469-3.

[12] G. Liu, H. Zhao, F. Fan, G. Liu, Q. Xu, and S. Nazir, "An Enhanced Intrusion Detection Model Based on Improved kNN in WSNs," Sensors, vol. 22, no. 4, p. 1407, Feb. 2022, doi: https://doi.org/10.3390/s22041407.

[13] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," Applied Sciences, vol. 9, no. 20, p. 4396, Oct. 2019, doi: https://doi.org/10.3390/app9204396.

[14] R. Ahmad, R. Wazirali, and T. Abu-Ain, "Machine Learning for Wireless Sensor Networks Security: An Overview of Challenges and Issues," Sensors, vol. 22, no. 13, p. 4730, Jun. 2022, doi: https://doi.org/10.3390/s22134730.

[15] C. Ioannou and V. Vassiliou, "An Intrusion Detection System for Constrained WSN and IoT Nodes Based on Binary Logistic Regression," Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Oct. 2018, doi: https://doi.org/10.1145/3242102.3242145.