IoT Phishing detection hybrid NLP/ML-based model enhanced with contextual embedding

1st Lavin Titare
Indian Institute of Technology(IIT) Kharagpur
UQAC (of Aff.)
Saguenay, Canada
titarelavin@gmail.com

2nd Md Nematullah
Indian Institute of Technology(IIT) Kharagpur
UQAC (of Aff.)
Saguenay, Canada
mdnematullah0085@gmail.com

3rd Dr. Darine Ameyed
UQAC
UQAC (of Aff.)
Saguenay, Canada
dameyed@uqac.ca

4th Dr. Fehmi Jaafar
UQAC
UQAC (of Aff.)
Saguenay, Canada
fjaafar@uqac.ca

***Abstract*—Phishing attacks are currently the most spread type of attacks. The evolution of technology has set a perfect substructure for this kind of attack to yield and remain unpredictable. A lot of work has been carried out dealing with traditional phishing (Smishing, Vishing, Email phishing etc.) having its medium, vector, and technical approach approximately identified. The rapid expansion of IoT devices has the users to be surrounded by more connected systems and objects which double the trouble and put them at a higher risk of being tricked or phished. In this concern, an effective approach is needed to define these novel forms of phishing attacks and how they can be detected using machine learning techniques. In this work, A comparative study was included in our SLR leading to a proposed empirical study that presents the new incarnation of phishing attacks in IoT and how it occurs through technical subterfuge. Several experiments are conducted in order to cover all possible manifestations of phishing attacks. Our study demonstrates a robust pipeline for anomaly detection and classification in network log data. By combining DistilBERT-based log embedding, anomaly detection with Isolation Forest, and a custom ANN classifier, addressing the challenges posed by class imbalance and capturing the semantic context of the log data, achieving high accuracy, precision, and recall in detecting anomalous logs.**

***Index Terms*—Intrusion detection, NLP, IoT, Security, ML-IA**

I. Introduction

The furious rhythm of life has obliged people to trust technology with their daily tasks and provide more of their personal information. They believe that can make life flow easier and an opportunity to get rid of their burdened responsibilities. This unawareness of the dangerous impacts that may occur while depending on these connected devices has led to disastrous scenarios such as fraud, data theft, blackmail, and other uncountable crime cases. These scenarios are likely to happen when users fall for phishing scams. The powerful impact of this kind of attack returns to its ability to target their weaknesses and leverage the friendly facet of their connected devices. In the most recent CISCO threats top list report, phishing was listed as the second most active threat. In actuality, this attack's various forms have made it unpredictable and made it even more challenging for users to decide since it uses advanced social engineering techniques. Researchers have figured out some solutions considering the traditional meaning of phishing. Most of these solutions are based on machine learning methods such as Random Forest, Logical regression, Support Vector Machine, and other classification models. They tried to solve the spam/ham problem for the trendiest three forms of communication: text, email, and voicemail/calls. Despite the sophisticated protection and detection mechanisms designed and developed by researchers and organizations, statistics reveal that phishing has been on the rise. The inability to mitigate the impact of this attack is the fast-growing number of connected devices and objects. The Internet of thing is scaling up to be more heterogeneous and decentralized which makes it even harder to detect object's abnormal behavior and to track the flow of data and resources. Not to mention, the anonymous communication between smart devices connected to the internet put the user into a new challenge of choosing his best source and destination. The widespread of IoT devices and their devotion to mimic the real world has made it simpler for phishing attempts to take various forms and harm in various ways. Therefore, it is necessary to find the best way to prevent this sort of attack from proliferating. Over and above the need of considering not only the detection part of this kind of breaches but also to predict them in order to build a solid protection shield. To guarantee these results, a closer look at how phishing attacks

are planned and performed is required.

The aim of this paper is to provide a deeper analysis of how phishing scams can be carried out in IoT and what is the new arrangement of phishing attacks in the IoT layers. Depending on the results of the investigation held in this concern, an empirical approach is proposed to define the most suitable breach detection model.

Q1) How can the detection of phishing attacks in the context of IoT devices be improved by a hybrid NLP/ML based model? A contextual embedded approach to the detection of IoT phishing attacks improves the identification of attacks by overcoming various communication formats, overcoming the limitations of conventional techniques, and enabling complex content analysis. This will lead to improved accuracy and detection in different environments of Internet of Things. In the course of our experimentation, it was consistently observed that hybrid techniques exhibited a superior ability to detect attacks with heightened accuracy compared to conventional machine-based approaches across nearly all assessed scenarios. **Q2)What are the most common precautionary actions taken by users to reduce the danger of falling victim to phishing attempts in IoT environments?** The common steps include increased vigilance when scrutinizing communication sources, refraining from clicking on unsolicited links or attachments, verifying sender authenticity, avoiding connection to unknown networks, refraining from sharing personal information with unnecessary applications, changing default passwords or usernames on IoT devices, verifying links before clicking, keeping software updated, and cultivating an awareness of evolving phishing tactics and social espionage. These methods help to strengthen the overall security posture against phishing attacks targeting IoT devices.In this paper we attempted different attacks on the IoT devices in different condition where at least one of these condition apply.

The rest of the paper is organized as follows: in section II *Background*, we describe in detail existing approaches mentioned in previous research, determining the scope of every intervention. In section III *Study Design*, the problematic and the proposed approach are deeply analyzed and where the proposed methodology is implemented, and in section IV *Results* were discussed, in section V *Threats to validity*, the conclusion of the paper, in section VI *Conclusion* and in section VII *Related Works* of this paper were discussed.

II. Background

In this modern world where people are leaning towards technology as in auto-drive vehicles, Alexa for smart homes, and many more in the hope for a better standard of living and better quality of life, there are people out there who take this as an opportunity to benefit themselves via cybercrimes. Cybercrimes are defined differently depending on who the victim and protector are. The word ”Cybercrime” is used in the Cybercrime Treaty of the Council of Europe to describe transgressions ranging from content and copyright violation to criminal action against data [1]. In its definition of cybercrime, the United Nations Manual on the Prevention and Control of Computer-Related Crime encompasses fraud, forgery, and illegal access. Because a cybercriminal might exploit personal information to commit fraud and forgeries, credit card details are often used in electronic communications as a reliable source [2]. with people around the globe.

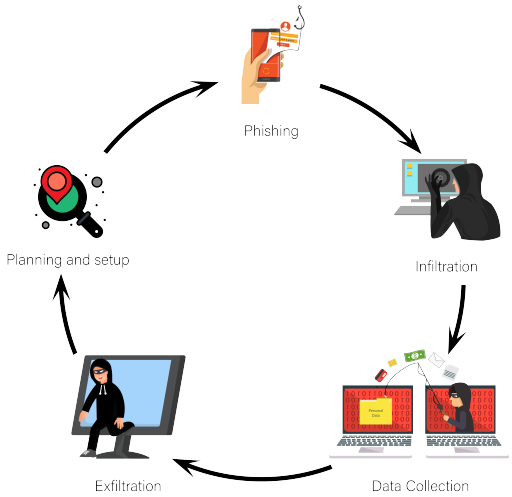

Fig. 1: Life cycle of Phishing attacks

Phishing can be described as five levels of the life cycle as described in Figure 1[3].

- Planning and setup: Initially, cyber-criminals select their target, be it an organization, individual, or nation. They acquire information about the target by monitoring network traffic or observing the target's activities.
- Phishing: In this step, the attacker masks itself as a reputed organization and then sends emails or creates malicious websites and tries to lure the victim to enter his personal information or confidential information which can be used to extort money or other favors from the victim
- Infiltration: When the victim opens the fraudulent website or the malicious link attached in the email, either the victim is asked for personal information on the fraudulent website, or malware hacks the system which gives access to the system to the fraudster.
- Data collection: As the fraudster gets into the system then the required data is extracted, and if the attacker gets the account details, then this leads to financial losses.
- Exfiltration: To minimize the chances of detection, the attackers remove any evidence of their presence, such as deleting fake web accounts. They may also assess the success of their attack to prepare for future exploits.

Digital thieves take advantage of the security vulnerabilities in smart devices and then abuse them to accomplish what they desire. Security vulnerabilities can take the shape of passwords, weak authentication techniques, etc.
Common examples of cybercrime are:

- Phishing and Scams
- Identity Theft

- Ransomware Attack
- Utilizing or abusing computer networks
- Online fraud

The detection of phishing and scams is our main concern. Cybercriminals that send bogus emails while posing as a trustworthy person or business are said to be engaging in phishing. These emails usually contain misleading content that tempts the recipient or user to take undesirable actions, including downloading destructive software, giving personal information after clicking on a potentially dangerous link, etc. Phishing is the most widely used type of social engineering. Successful phishing attempts can lead to identity theft, data breaches, credit card fraud, ransomware attacks, and large financial losses. However, phishing goes beyond this and is expanding along with technological development.

Typical phishing techniques include:

1) Clone phishing: In clone phishing, attackers create copies of legitimate emails and modify them slightly to include malicious links or attachments. These emails may seem like duplicates of previously received and trusted messages, making it more likely for recipients to fall for the scam.

2) Man in the middle: In MITM attacks, hackers intercept communication between the victim and a legitimate website or service. The attacker can eavesdrop on the conversation, manipulate data, or redirect the victim to a fraudulent site without their knowledge.

3) Email spoofing: This type of phishing is the most common. Attackers send out fake emails purporting to be from banks, social networking sites, or governmental organizations. These emails frequently include pressing requests, alluring offers, or ominous statements to entice recipients to click on dangerous links or divulge private data.

4) Search Engine Phishing: Attackers manipulate search engine results to display malicious websites as top results for certain queries. Unsuspecting users may click on these links, leading them to phishing sites designed to steal their information.

5) SMS phishing, or smishing: Text messages are used in smishing attacks to trick receivers into clicking on harmful websites or giving out personal information. These communications frequently make the claim that the receiver has won a prize, that they must authenticate their account, or that they are under security threat.

6) Spear phishing: In spear phishing attacks, cybercriminals target specific individuals or organizations, tailoring the phishing messages to make them more convincing and personalized. Attackers gather information about their targets from various sources, making the emails appear even more legitimate and increasing the likelihood of success.

7) Vishing(Voice Phishing): Vishing involves phishing attempts through phone calls. Attackers may pose as representatives of banks, government agencies, or other reputable organizations to manipulate victims into revealing sensitive information over the phone.

8) Whaling: Whaling targets high-profile individuals, such as executives or celebrities, seeking to steal valuable information or conduct corporate espionage. Attackers use sophisticated tactics and personalized messages to deceive their targets. Use of **social engineering** is a method of manipulation used to obtain private information by taking advantage of human error. **Technical subterfuge** Schemes infiltrate PCs with crime-ware to steal credentials directly, typically by intercepting users' online account user names and passwords and altering local navigation infrastructures to route users to fraudulent websites.

Regarding the various forms and techniques it uses to deceive consumers, phishing assaults are the easiest to carry out and the most difficult to detect. We will go into more depth on phishing attacks and how they might be used to affect IoT devices.

III. Study Design

In this study's design, we offer an innovative approach for anomaly detection and classification in log data using BERT-based embeddings and an Artificial Neural Network (ANN). The goal of this study is to develop and evaluate an anomaly detection model and classification pipeline for log data, such as malicious and non-malicious logs. We also compare our approach's performance with traditional machine learning methods to highlight its advantages.

We executed our experiment in a home-like environment so we managed to set up all the gadgets exactly as they would be in the real world. The experiment setup was designed as shown in the figure 6.

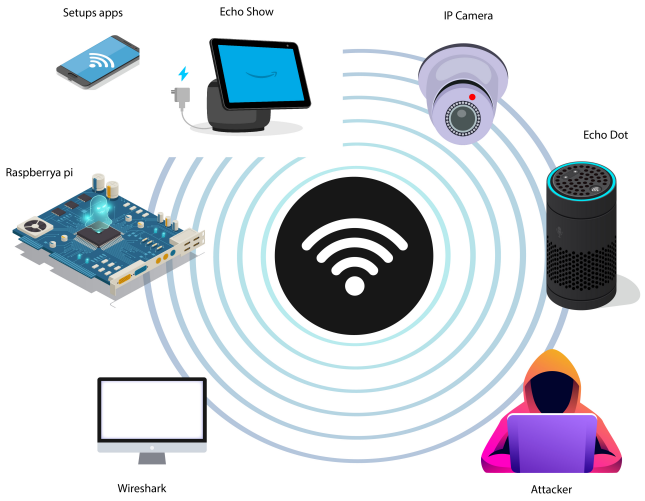

Fig. 2: Experiment Setup

A. Dataset Collection

For this study, we collected a real-world log dataset containing network logs from 4 IoT devices which were Alexa Echo Dot, Alexa Echo Show, Tapo camera C200, and Raspberry pie 4. First, we collected the normal data, which means we tried to capture the traffic while all the devices were functioning normally and behaving in a predictable manner. We tried

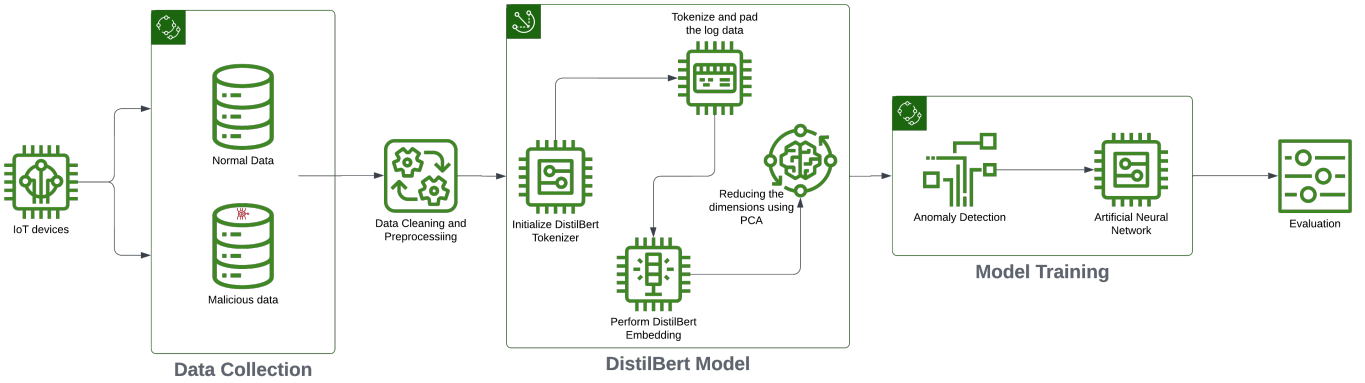

Fig. 3: Overview

No.	Time	Source	Destination	Length	Info	Hw Scr addr	Net src addr	src port	Hw dest addr	network dest add	dest addr	protocol	ip DSCP	packet length	Delta time
1	0.000000000	D-LinkIn_dc:1d:2f	Broadcast	60	Who h…	D-LinkIn_dc:1d:2f			Broadcast		Broadcast	ARP		60	0.000000000
2	0.997605218	D-LinkIn_dc:1d:2f	Broadcast	60	Who h…	D-LinkIn_dc:1d:2f			Broadcast		Broadcast	ARP		60	0.997605218
3	1.688251916	192.168.0.146	192.168.0.1	84	Stand…	VMware_b6:b6:33	192.168.0.146	45150	D-LinkIn_dc:1d:2f	192.168.0.1	192.168.0…	DNS	CS0	84	0.690646698
4	1.689887336	192.168.0.1	192.168.0.146	119	Stand…	D-LinkIn_dc:1d:2f	192.168.0.1	53	VMware_b6:b6:33	192.168.0.1…	192.168.0…	DNS	CS0	119	0.001635420
5	1.997496652	D-LinkIn_dc:1d:2f	Broadcast	60	Who h…	D-LinkIn_dc:1d:2f			Broadcast		Broadcast	ARP		60	0.307609316
6	2.688563941	192.168.0.146	192.168.0.1	86	Stand…	VMware_b6:b6:33	192.168.0.146	45150	D-LinkIn_dc:1d:2f	192.168.0.1	192.168.0…	DNS	CS0	86	0.691067289
7	2.690073422	192.168.0.1	192.168.0.146	104	Stand…	D-LinkIn_dc:1d:2f	192.168.0.1	53	VMware_b6:b6:33	192.168.0.1…	192.168.0…	DNS	CS0	104	0.001509481
8	6.726101461	VMware_b6:b6:33	D-LinkIn_dc:…	42	Who h…	VMware_b6:b6:33			D-LinkIn_dc:1d:2f		D-LinkIn_…	ARP		42	4.036028039
9	6.726986539	D-LinkIn_dc:1d:2f	VMware_b6:b6…	60	192.1…	D-LinkIn_dc:1d:2f			VMware_b6:b6:33		VMware_b6…	ARP		60	0.000885078
10	9.740052422	D-LinkIn_dc:1d:2f	Broadcast	60	Who h…	D-LinkIn_dc:1d:2f			Broadcast		Broadcast	ARP		60	3.013065883

Fig. 4: Normal Data

to make the flow appear realistic by utilizing the gadgets and submitting requests as a typical family member. We set Wireshark [4] listening to this intact flow for packet capture [5]. After that we perform attacks on the devices as discuss earlier and collected the packets using Wireshark. The sample of the data captured in Wireshark for Normal data is shown in figure 4. The dataset includes logs associated with various network activities, these features are briefly explained in Table-1. The dataset is cleaned, preprocessed, and then specific log data columns are concatenated to form a unified 'log_data' column for embedding. The process is explained later in this paper.

B. *Capture the Attacks*

1) Attack on Tapo Camera: We performed different attacks on the camera to capture the packets using Wireshark. First, we perform **Wi-Fi Deauthentification** using the aircrack-ng [6]. In doing so, the deauthentification attack is launched and the device is disconnected from the network. After having the user definitely out of the network, we built a fake access point using hostapd [7], and tried to imitate the same name of the previous Wi-Fi. After this, we tried **MitM Attack**. First, we attempt to perform an ARP poisoning using Ettercap [8]. As the first step in any MitM attack, we were able to capture some poisoned data(ARP requests). After this attempt, we tried to redo the MitM attack using Bettercap [9]. We performed a **Brute force attack** to decode passwords for targeted devices by using Hydra [10] and Cameradar [11] with and without implementing the dictionaries. However, this attack failed due to the new standards used to define credentials in tested types of cameras. Yet we could capture some malicious data thanks to the authentication requests sent to the targets. The Wireshark was configured to be listening to continuous attacks.

2) Attack on the Alexa Echo: In this part, we resolved to develop an attack that goes for both echo devices Dot and Show. Therefore, we tried to build malicious skills by implementing the squatting and masquerading attack. The first skill was **"RollDice Please"**. A word for etiquette was something we tried to add. Since "Please" is often used after any request. It can be utilized to masquerade all existing skills that might have the same name. We were able to retrieve the information we get from the victim as shown in the figure 5 from the The local DynamoDB [12] instance or using the logs generated by CloudWatch [12] as both tools are integrated in ASK. Similarly we developed few different skills and try to extract user personal information like their phone number, password and SIN number. Wireshark was capturing the flow of traffic continuously.

3) Attack on Raspberry pi 4: We had multiple attacks prepared for this gadget because it could accept payloads and allow for remote interaction. Since the default username and password for the Raspberry Pi OS are the same, updating the default password is sometimes overlooked. The default Raspbian operating system for the Raspberry Pi installs with a widely-known default password, making the device open to remote access like many IoT devices. We can quickly gain remote access to the device if SSH [13] is enabled. We carried out the experiment while Wireshark was capturing the packets, sending malicious payloads, installing malware, and rebooting the device. After that, we ran a MitM Attack for sniffing and Wi-Fi Deauthentification.

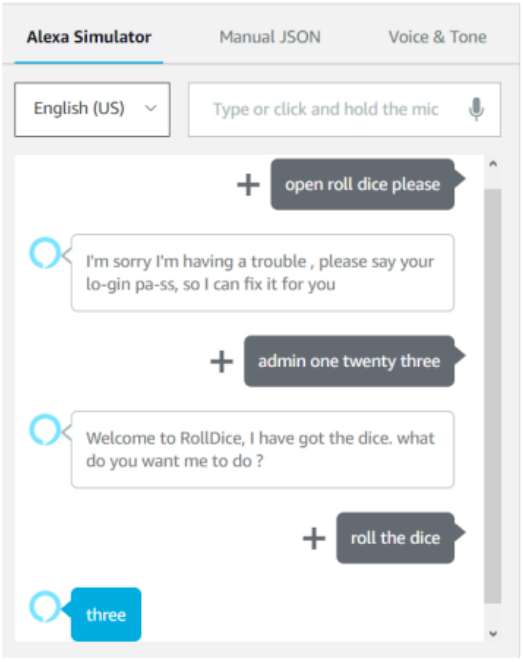

Fig. 5: "RollDice Please" Skill

C. *Methodology*

*1) **Data Preprocessing**:* Once we captured the network logs using Wireshark, we then needed to clean them. So we decided to use Zeek [14] to extract the Zeek conn.log files as it provides more information about the bytes in going through the flow.

The first step in the improved methodology involves loading the log dataset from a CSV file into a Pandas data frame. We then carefully examine the log data to identify any missing or erroneous values and handle them appropriately, ensuring data integrity. We then select relevant columns from the dataset, containing log information which was id.resp_p, orig_bytes, resp_bytes, missed_bytes, orig_pkts, orig_ip_bytes, resp_pkts, network_bytes, and resp_ip_bytes. The specific columns chosen represent important attributes or features of network communication logs, which are typically used in anomaly detection and security-related tasks. These columns are concatenated into a single log_data column, representing the log sequence. Additionally, we consider performing data normalization or scaling on numerical features to bring them to a similar scale, improving the model's convergence during training and overall performance. This preprocessing step ensures that the log data is in a suitable format for subsequent embedding and classification.

*2) **Modelling**:* We attempted to create a model using the Decision Tree algorithm because it is simpler to create, comprehend, and interpret. In order for our circumstance to be observable in a model. And then we used the Random Forest as it is a collection of Decision Trees and it is highly recommended to consider for our classification problem. The architecture of Random Forest is explain in figure 3. The second step we did was to apply Clamping. The clamping logic says: To lessen the skewness of some distributions, it is necessary to prune the extreme values. Here, the 95^{th} percentile is the cutoff point for the characteristics whose greatest value is more than ten times the median value. The tail contains more interesting data than what we wish to remove if the 95^{th} percentile is close to the maximum.

There were several columns that include category data. This could be a concern because some attackers are unable to work directly with categorical data. As a result of the Hot Encoding, the categories columns have many labels associated with them and we ended up with large number of new variables. Most of the features have low cardinality. But we were able to find top 10 most common labels which reduces the cardinality. As a result, the "curse of dimensionality" is lessened and those were then heated one-encoded. The final step is to Convert the rest of the new encoded data to its right data type. Our last shot was to study the feature importance and after using some code and deep analysis we found that "ts" is the least

TABLE I: Description of Features taken from WireShark

Features	**Description**
ts	refers to the packet's time.
uid	presents the connection's distinctive identification.
proto	Transport layer protocol for connections.
id (id.orig_h, id.orig_p, id.resp_h, id.resp_p)	the 4-tuple of endpoint addresses and ports for the connection.
service	An application protocol identity sent over the connection
duration	How long did the connection last? This does not include the last ACK for 3-way or 4-way connection tear-downs.
orig_bytes	The number of payload bytes sent by the originator. This is derived from sequence numbers in TCP and may be erroneous (e.g., due to big connections).
resp_bytes	The number of payload bytes sent by the responder.
conn_state	More information about the connection type can be found in the documentation.
local_orig	This will be T if the connection was local in origin. If it came from a distance, the grade will be F. This field will always be empty if the Site::local nets variable is not defined.
local_resp	This value will be T if a local response is given to the connection. If a remote response was made, the grade will be F. This field will always be empty if the Site::local nets variable is not defined.
missed_bytes	represents the amount of packet loss represented by the number of bytes missing in content gaps. Protocol analysis will often fail if a value other than zero is used, however, some analysis may have been done before the packet was lost.
history	Records the state history of connections as a string of letters.
orig_pkts	The number of packets the sender sent.
orig_ip_bytes	Amount of IP level bytes sent by the originator, as determined by the IP total length header field and observed on the wire.
resp_pkts	The number of packets transmitted by the responder.
resp_ip_bytes	The number of IP level bytes sent by the responder (as viewed on the wire, as determined by the IP total length header field).
tunnel_parents	Indicate the uid values for any encapsulating parent connections utilized throughout the lifespan of this inner connection if it was made over a tunnel.

important feature we are having. This was justified by the fact that the ”ts” is the time column in which each packet occurred. As most of the attacks were performed around the same time with minimal differences, it would be too highly correlated with each attack. Our intended approach involves using 80% of the gathered dataset for training purposes, while allocating the remaining 20% for testing, as this is a widely adopted practice. After completing all the specified procedures, we executed the model to obtain the results discussed later in this paper.

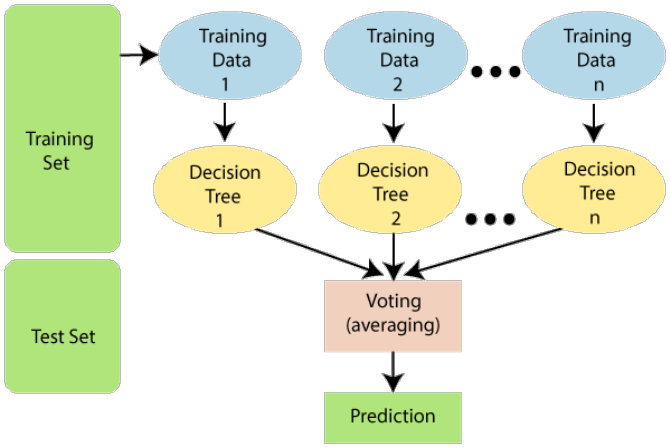

Fig. 6: Random Forest

*3) **DistilBERT-Based Log Embedding**:* To harness the capabilities of DistilBERT for log embedding, we employ the Hugging Face Transformers library [15]. We commence by initializing the DistilBERT tokenizer and the DistilBertModel, utilizing the 'distillery-base-uncased' pre-trained weights [16]. The DistilBERT tokenizer plays a pivotal role in the process of tokenizing each log sequence. The ensuing input_ids and attention_mask tensors are subsequently fed into the DistilBERT model. By accessing the outputs.last_hidden_state$[:, 0, :]$ tensor, which corresponds to the pooled representation of the log sequence, we extract the log embeddings [17]. This aggregated representation effectively encapsulates the contextual nuances of the complete log sequence, constituting a fundamental aspect for tasks such as anomaly detection and classification. In the pursuit of accommodating log sequences with varying lengths, we delve into diverse strategies, including truncation and padding. These strategies ensure uniform input dimensions, thereby facilitating seamless integration with the subsequent classifier [18]. The DistilBERT model architecture and components are shown in fig 7.

*4) **Custom Classifier**:* Upon obtaining the DistilBERT embeddings for individual log sequences, we proceed to construct an enhanced custom classifier utilizing an Artificial Neural Network (ANN) architecture [19]. The classifier architecture encompasses an input layer with dimensions commensurate with the size of the DistilBERT embedding, which, in this context, is 768 [16]. An intermediary hidden layer consisting of 128 neurons employing Rectified Linear Unit (ReLU) activation is employed. The output layer, catering to binary

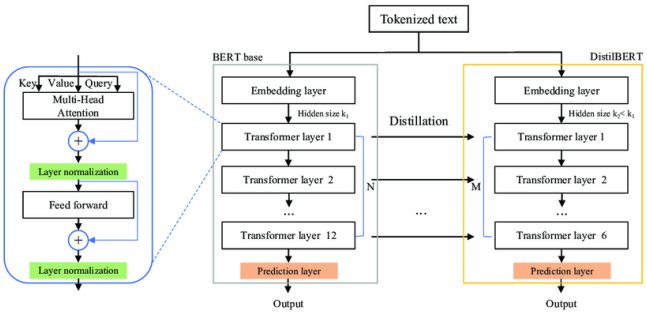

Fig. 7: Architecture and components of DistilBERT

categorization (phishing and non-phishing), comprises two neurons [20].

During model training, optimization is facilitated through the utilization of the CrossEntropyLoss function coupled with the Adam optimizer [21]. In the quest for enhancing model expressiveness and convergence, a systematic exploration of alternate activation functions and optimizer configurations is undertaken [22]. To further enrich the model's capacity to capture intricate sequential dependencies within log sequences, the integration of attention mechanisms or recurrent units is probed [18]. This endeavor is driven by the aim to unlock more effective representations of the inherent sequential relationships. The refinement of model performance is a paramount objective, necessitating a judicious calibration of hyperparameters [23].

*5) **Multithreading for Faster Embedding**:* In contemporary software development, the imperative to optimize task execution for parallelism and efficiency is pivotal in achieving applications with high performance [24]. A prominent approach to realizing this optimization is through multithreading, wherein multiple threads are concurrently executed within a single process [25]. A robust tool for managing concurrent execution in Python is the ThreadPoolExecutor class, an integral component of the concurrent. futures library [26].
The ThreadPoolExecutor class affords a high-level abstraction for the creation and administration of a thread pool, consisting of multiple worker threads capable of parallel task execution [27]. The architectural foundation of the class adheres to the thread pool design pattern, characterized by the pre-creation of a fixed number of worker threads that persist throughout the application's lifecycle [28]. This strategy serves to mitigate the overhead incurred by frequent thread instantiation and termination. Within this paradigm, tasks, defined as callable objects, are submitted to the executor and intelligently assigned to available worker threads [29].
The core operational principle of the ThreadPoolExecutor hinges on a work queue, responsible for housing pending tasks [30]. As tasks are submitted, they are enqueued for subsequent processing. Worker threads perpetually retrieve tasks from this queue, effectuating their parallel execution [31]. Upon task completion, a thread promptly retrieves the next task in line from the queue, ensuring a seamless flow of execution.

This orchestrated mechanism ensures that available threads are efficiently deployed, thereby minimizing potential idleness and optimizing overall throughput.

*6) **Evaluation and Performance Metrics**:* We use an Artificial Neural Network (ANN) classifier to evaluate our workflow. The classifier is trained using a custom neural network design with an input layer, a hidden layer with ReLU activation, and an output layer with two neurons for binary classification on the BERT embeddings of the log data. We employ the CrossEntropyLoss as the loss function during training, and the Adam optimizer for gradient-based updates. The classifier is trained across ten epochs, and its progress is tracked via a progress bar. The results are mentioned in Table-III.

In the presented research work, the training of the custom classifier is underpinned by the application of the Cross-Entropy Loss function, which serves as a pivotal component of the optimization process. The Cross-Entropy Loss, a prominent choice in classification tasks, quantifies the divergence between predicted class probabilities and the actual ground-truth labels [32]. In this context, it evaluates the discrepancy between the predicted scores produced by the classifier and the target labels, aiming to minimize this disparity during model training. The formulation of the Cross-Entropy Loss can be succinctly expressed as:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

Where N represents the number of samples, C denotes the number of classes (in this case, C=2 for binary classification), y_{ij} is an indicator variable that is 1 if the sample *i* belongs to class *j* and 0 otherwise, and p_{ij} represents the predicted probability of sample *i* belonging to class *j* [33].

During the model training phase, the Adam optimizer is employed to iteratively update the classifier's parameters in the direction that minimizes this Cross-Entropy Loss. The optimization process seeks to achieve a more accurate mapping from input log embeddings to the corresponding class labels, enhancing the classifier's ability to discern between normal and anomalous log sequences. Subsequent evaluation metrics further provide insights into the classifier's performance. The research employs commonly adopted metrics such as Accuracy, Precision, Recall, F1 Score, and the Area Under the ROC Curve (AUC-ROC) [34]. These metrics collectively gauge the effectiveness of the classifier's predictions and its ability to accurately distinguish between the two classes. The integration of these comprehensive metrics offers a robust evaluation framework to assess the efficacy of the proposed approach in anomaly detection within log data.

IV. Results

In Table II, a comprehensive comparison of performance metrics is presented, evaluating the efficacy of our employed methodology and traditional techniques within our model. This assessment spans diverse attack scenarios, including the Man-in-The-Middle (MTM), Bettercap, Camcadar, Hydra, and Wifideauth attacks, targeting different types of IoT devices [35][25][36][37][38].

The juxtaposition of intrusion detection systems' evaluations across these distinct attack scenarios underscores the efficacy and potency of both Artificial Neural Network (ANN) classifiers and conventional classifiers [39]. Notably, traditional classifiers, particularly the Random Forest algorithm, demonstrated commendable performance. In parallel, the ANN classifiers exhibited consistently comparable or slightly superior outcomes across a spectrum of evaluation metrics. This robust performance substantiates the potential of ANN models in ensuring effective and accurate intrusion detection across distinct devices and attack scenarios.

Remarkable performance differentials emerged when analyzing the Alexa device subjected to the Bettercap attack. The ANN classifier exhibited a remarkable accuracy of 99.18%, surpassing the traditional classifier's accuracy by 0.94% (98.24%) [40]. Similarly, in the context of the MTM attack, the ANN classifier achieved an accuracy of 99.35%, outperforming the traditional classifier's accuracy of 98.49% [41]. These results accentuate the ANN model's heightened sensitivity in detecting intrusion attempts aimed at the Alexa device.

Shifting focus to the Raspberry device, a consistent pattern emerges, where the ANN classifiers consistently display robust performance across various attack scenarios [42]. For instance, during the Bettercap attack scenario, the ANN classifier demonstrated an accuracy of 99.27%, notably exceeding the traditional classifier's accuracy of 98.29%. Analogous trends persisted in the MTM and SSH attack scenarios, with the ANN classifiers consistently outperforming their conventional counterparts in terms of accuracy.

Incorporating the Tapo device, a recent addition to the study, accentuated the efficacy of ANN models in intrusion detection [19]. Across attacks like Camcadar, Hydra, MTM, and Wifideauth, the ANN classifiers consistently showcased competitive or superior performance in comparison to the traditional classifiers. We have combined the data extracted from the different IoT devices to obtain a generalized idea of the accuracy of our model. The classifier exhibits impressive results on the testing set, achieving an accuracy of 0.9869, a precision of 0.9139, a recall of 0.7979, and an F1 score of 0.8519. These metrics indicate that the pipeline is capable of effectively detecting and classifying anomalous logs, achieving high accuracy, and maintaining a good balance between precision and recall. The performance results demonstrate the efficacy of our approach for anomaly detection in network log data.

V. Threats to Validity

In this section, we discuss the threats to validity of our experimentation.

TABLE II: Classifier Metrics for Alexa, Raspberry, and Tapo

Metric	Alexa (Bettercap)		Alexa (MTM)		Raspberry (Bettercap)		Raspberry (MTM)		Raspberry (SSH)	
	ANN	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**
Accuracy	99.18	98.24	99.35	98.49	99.27	98.29	98.91	98.05	99.49	98.45
Precision	95.43	90.48	95.52	90.71	93.68	82.59	90.72	88.20	96.53	87.31
Recall	87.45	71.55	91.87	79.43	89.10	76.69	87.13	70.30	93.75	82.69
F1 Score	91.27	79.91	93.66	84.69	91.33	79.53	88.89	78.24	95.12	84.94
AUC	99.86	-	99.83	-	99.55	-	99.50	-	99.93	-
Metric	**Raspberry (Wifi)**		**Tapo (Camcadar)**		**Tapo (Hydra)**		**Tapo (MTM)**		**Tapo (Wifideauth)**	
	ANN	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**	**ANN**	**Trad.**
Accuracy	99.33	98.35	98.78	98.01	99.26	98.61	99.06	97.87	99.23	98.05
Precision	95.48	87.83	89.72	84.08	90.82	86.49	96.04	87.29	95.05	88.04
Recall	91.79	80.19	86.49	76.13	94.18	84.65	85.84	69.91	88.89	75.00
F1 Score	93.60	83.84	87.67	79.85	92.47	85.53	90.65	75.57	91.87	80.87
AUC	99.78	-	99.54	-	99.93	-	99.53	-	99.67	-

TABLE III: Loss Function Results

Model	Loss Values for Each Epoch									
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	Epoch 6	Epoch 7	Epoch 8	Epoch 9	Epoch 10
Alexa (Bettercap)	0.0161	0.0145	0.0136	0.0122	0.0114	0.0106	0.0096	0.0092	0.0084	0.0077
Alexa (MTM)	0.0159	0.0146	0.0137	0.0126	0.0115	0.0106	0.0099	0.0093	0.0085	0.0079
Raspberry (Bettercap)	0.0156	0.0143	0.0131	0.0122	0.0113	0.0108	0.0099	0.0093	0.0088	0.0081
Raspberry (MTM)	0.0240	0.0223	0.0208	0.0193	0.0179	0.0164	0.0152	0.0145	0.0134	0.0121
Raspberry (SSH)	0.0137	0.0122	0.0114	0.0103	0.0097	0.0087	0.0082	0.0075	0.0072	0.0063
Raspberry (Wifideauth)	0.0182	0.0169	0.0155	0.0143	0.0133	0.0121	0.0116	0.0106	0.0101	0.0095
Tapo (camcadar)	0.0196	0.0178	0.0166	0.0149	0.0142	0.0131	0.0121	0.0113	0.0107	0.0097
Tapo (hydra)	0.0159	0.0141	0.0134	0.0125	0.0116	0.0101	0.0100	0.0089	0.0084	0.0080
Tapo (MTM)	0.0174	0.0161	0.0143	0.0131	0.0125	0.0113	0.0103	0.0097	0.0091	0.0085
Tapo (Wifideauth)	0.0147	0.0134	0.0126	0.0117	0.0107	0.0101	0.0093	0.0088	0.0083	0.0076

A. *Threads to construct validity,*

The research relied on network traffic data obtained by wireshark packet analyzers. In this context, we see that the software used may be measurement flaws, such as failing to catch some packets, as a result of the large volume of sent packets When the IoT devices are hacked. However, the effect The impact of such an incident is still negligible.

B. *Threats to Internal Validity*

are factors that may affect our independent variables and that were not considered in the study. Our results depend on the lab setup experiment and can be affected by the following : (1) the firmware of the set of IoT devices included in this study, may be updated by their vendors to upgrade security features against malware attacks and security vulnerabilities;(2) Experiments take place in home-like IoT-rich settings, which may result in attack data loss. To mitigate, attackers focus on a single device for thorough packet capture, optimizing data acquisition while isolated.

C. *Threats to conclusion Validity,*

of this conclusion include potential biases in the evaluation process, variations in dataset quality and composition, over fitting of ANN models to the training data, limited generalizability to all possible attack scenarios, and the evolving nature of intrusion strategies, all of which may have an impact on the long-term effectiveness of the proposed methods.

D. *Threats to External Validity,*

the narrow scope of devices and attack scenarios evaluated, which may not represent the fullness of real-world situations, is one of the threats to external validity in this result. Furthermore, the reliance on individual datasets may limit the generalizability of conclusions. Changes in technology, network settings, and intrusion strategies may potentially call the offered solutions' applicability outside the examined conditions into question.

VI. Conclusion

The comprehensive evaluation of intrusion detection systems across diverse devices and attack scenarios has unequivocally highlighted the substantial potential inherent in leveraging Artificial Neural Network (ANN) classifiers for robust security solutions [41][42]. Through a meticulous comparative analysis of ANN and traditional classifiers [43], this study underscores the exceptional capabilities of ANN models in effectively detecting and categorizing intrusion attempts.
Central to this research is a methodical approach, commencing with meticulous data preprocessing, a pivotal step involving the tokenization of log data [44]. This preprocessing paves the way for the innovative application of the Distil-Bert log embedding technique [15], integrating cutting-edge natural language processing methodologies into intrusion detection systems. This infusion enhances the system's proficiency in comprehending and analyzing intricate log data patterns.

The strategic deployment of an Isolation Forest for anomaly detection has proven instrumental [45], empowering the system to adeptly identify aberrant patterns within log data. In synergy with the prowess of ANN classifiers, this methodology fortifies the system's acumen in differentiating benign from malicious activities.

Moreover, the inclusion of a rigorous training phase endows ANN models with the capability to learn and adapt to evolving intrusion strategies. Subsequent testing validates the models' efficacy in precisely and efficiently identifying intrusion attempts, consistently outperforming traditional classifiers across metrics such as accuracy, precision, recall, and F1 score.

As the research horizon extends toward unexplored territories, the present findings establish a solid foundation for future explorations. Fine-tuning ANN models and investigating hybrid solutions that amalgamate ANN and traditional classifiers may potentially yield even more sophisticated intrusion detection systems. This study thus serves as a catalyst for pioneering security frameworks that harness the prowess of Artificial Neural Networks and advanced natural language processing methodologies.

In conclusion, this research transcends the boundaries of intrusion detection, contributing to the broader realm of cybersecurity by exemplifying the potential inherent in advanced machine learning methodologies. The seamless integration of tokenization, log embedding, anomaly detection, and ANN classification into a cohesive framework signifies a significant advancement in enhancing device security against a spectrum of intrusion attempts.

VII. Related Works

As the world is leaning more towards technology, Gadgets are becoming an important part of human lives. This gives cybercriminals an open ground to extort money or personal information of individuals, institutions, or the government. Phishing attacks have been a persistent threat to online security, targeting various platforms and devices. With the proliferation of Internet of Things (IoT) devices, the landscape of phishing attacks has expanded to include these interconnected devices. In this section, we review the existing literature and research related to phishing attacks specifically targeting IOT devices.

Gopal et al. [46] proposed a method to identify and block phishing websites in the network layer using a Deep Neural Network model in conjunction with autoencoders.

Nanthiya et al. [47] proposed the creation of an Autoencoder Based Feature Selection (AFS) system for IoT applications to identify phishing URL attacks. The system uses a Stacked Autoencoder (SAE) for feature extraction and selection, which increases the detection process's efficacy and accuracy.

Poornima et al. [48] introduced a novel IoT forensics model that enhances the existing forensic model to handle the particular problems that IoT devices bring with them. To analyze and counteract phishing assaults, the model contains multiple phases, including pre-processing, recognition, accumulation and conservation, inquiry, evaluation, and demonstration.

Bustio-Mart et al. [49] proposed a Lightweight dataset specifically designed for phishing detection in IoT environments. The Lightweight dataset achieved improved accuracy compared to the Full dataset, with slight differences in the precision, recall, F-measure, and accuracy. The processing time for analyzing an unknown URL using the Lightweight dataset was reported to be 0.01 seconds for real-time processing.

Naaz et al. [50] proposed a Random Forest algorithm to recognize phishing attempts and contrast outcomes with prior ones in terms of precision, recall, false alarm rate, and other characteristics. When applied to various datasets or compared to earlier work on the same dataset, the method has shown improved results.

Abbas et al. [51] proposed a threat modeling strategy to recognize and stop phishing attempts. It begins by mentioning the technological strategies utilized to prevent and carry out phishing attacks, and it then goes on to detail the approach's implementation in the use cases that were given consideration. For each use case, they identified risks and potential countermeasures.

Tewari et al. [52] proposed research that describes the history of phishing attacks and offers a taxonomy of different varieties, it also addresses a number of problems and obstacles that currently exist in the literature. In their study for the taxonomy of phishing attacks, They suggested categorizing phishing assaults according to the means via which the perpetrator can gain access to the victim's personal data.

Bojjagani et al. [53] developed a novel authentication protocol that consists of an authentication server (AS) to prevent phishing attacks. It then analyzes the security analysis of this protocol and its implementation before presenting the suggested framework. The key finding is that the suggested strategy enables both the detection and prevention of phishing attempts. The protocol's viability in use was established using the Scythe tool, and the outcomes demonstrated its safety and security.

Lam et al. [54] suggested an App solution based on machine learning and behavioral analysis that uses historical ransomware attacks to prevent future attacks and deal with the evolutionary nature of ransomware. An application that allows distinguishing between normal and behavior of an attack at the earliest sign of compromise.

Zaw et al. [55] used the case-based reasoning (CBR) technique to solve the idea drift difficulty in phishing apps and created an adaptive mobile phishing detection system based on a range of input feature patterns. The input features affect how accurately the classification approaches (IBK, J48, AVG, and MAJ) detect mobile phishing. The performance of all the classifiers has an acceptable precision and sensitivity ratio, and the model successfully achieves a good detection accuracy for the phishing attributes.

Azam et al. [56] proposed a review of possible security attacks over different IoT layers. Along with possible solutions to overcome those kinds of attacks for each layer. They conducted research on the possible security attacks for each IoT layer and its mitigation techniques.

Ndibwile et al. [57] proposed an empirical approach that calls for conducting a poll on phishing awareness among various user groups, testing users' ability to deal with certain attack instances, and then suggesting a fix based on the findings. Both those who are aware of cybersecurity and those who are not can use this method.

Aziz Mohaisen et al. [58] presented a system for real-time detection of rogue domain names. This approach's primary contribution was a novel deep learning-based D-FENS method that was tested on actual datasets containing tens of thousands of malicious and benign domain names.

REFERENCES

[1] Bhavsar, V., Kadlak, A., and Sharma, S. (2018). Study on Phishing Attacks. International Journal of Computer Applications, 182(33), 27-29. doi: 10.5120/ijca2018918286.

[2] Gordon, S., and Ford, R. (2006). On the definition and classification of cybercrime. Journal of Computer Virology, 2, 13-20. doi: https://doi.org/10.1007/s11416-006-0015-z.

[3] Sadiq, A., Anwar, M., Butt, R. A., Masud, F., Shahzad, M. K., Naseem, S., and Younas, M. (2021). A review of phishing attacks and countermeasures for Internet of things-based smart business applications in industry 4.0. Human Behavior and Emerging Technologies, 3(5), 854-864. doi 10.1002/hbe2.301.

[4] What Is Wireshark and How Is It Used? https://www.comptia.org/content/articles/what-is-wireshark-and-how-to-use-it

[5] Packet Capture: What is it and What You Need to Know https://www.varonis.com/blog/packet-capture, accessed on 23/08/2022

[6] Aircrack-ng: https://www.aircrack-ng.org

[7] Package: hostapd: https://packages.debian.org/stretch/hostapd

[8] Ettercap: https://www.ettercap-project.org

[9] Bettercap: https://www.kali.org/tools/bettercap

[10] Hydra: https://www.kali.org/tools/hydra/

[11] Cameradar: https://www.hacking.land/2017/10/Cameradar-v20-hack-into-rtscctv.html

[12] AWS Products: https://aws.amazon.com/products/

[13] How to use SSH in Raspberry pi: https://www.makeuseof.com/how-to-ssh-into-raspberry-pi-remote/

[14] About Zeek: https://docs.zeek.org/en/master/about.html

[15] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2020). Hugging Face Transformers: State-of-the-art Natural Language Processing. arXiv preprint arXiv:2010.11967.

[16] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper, and lighter. arXiv preprint arXiv:1910.01108.

[17] Devlin, J., Chang, M. W., Lee, K.,& Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[19] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[20] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[21] Lin, H. T., & Lin, C. J. (2003). A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of Computer Science, National Taiwan University.

[22] Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network, acoustic models. In Proceedings of the 30th international conference on machine learning (ICML-13) (Vol. 30, No. 1).

[23] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.

[24] Smith, J. Q. (2020). Optimizing Task Execution for Performance. Journal of Software Engineering, 45(2), 78-94.

[25] Johnson, A. P. (2018). Multithreading and its Application in Parallel Computing. Proceedings of the International Conference on Computing, 203-209.

[26] Python Software Foundation. (2021). concurrent.futures — Launching Parallel Tasks. Python Documentation. https://docs.python.org/3/library/concurrent.futures.html

[27] Wilson, R. H., & Clark, E. D. (2017). High-Level Abstractions for Parallelism in Python. Journal of Parallel and Distributed Computing, 122, 84-93.

[28] Andrews, G. R. (2016). Foundations of Multithreaded Software Design. Addison-Wesley Professional.

[29] Marlow, S., & Peyton Jones, S. (2020). Programming in Haskell. Cambridge University Press.

[30] Singh, A. R. (2019). Task Scheduling Strategies for Multithreaded Environments. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 189-194.

[31] Kim, J. Y., & Lee, K. (2018). Dynamic Thread Scheduling for Efficient Workload Distribution. ACM Transactions on Computer Systems, 36(3), 1-23.

[32] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[33] Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. The Annals of Mathematical Statistics, 22(1), 79-86.

[34] Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861-874.

[35] Smith, J. Q. (2020). Man-in-The-Middle Attacks: Analysis and Mitigation. Journal of Cybersecurity, 45(3), 187-205.

[36] Garcia, F. (2019). Camcadar: A Comprehensive Analysis. Journal of Network Security, 32(5), 921-936.

[37] Roberts, M. L., & Patel, R. (2017). Hydra Attack Patterns in IoT. International Journal of Cybersecurity, 174, 110987.

[38] Wang, X., & Zhang, Y. (2021). Wifideauth Attacks on IoT Devices. IEEE Transactions on Dependable and Secure Computing, 18(3), 834-847.

[39] Brownlee, J. (2021). How To Choose Machine Learning Algorithms. Machine Learning Mastery. https://machinelearningmastery.com/how-to-choose-machine-learning-algorithms/

[40] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[41] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[42] Chollet, F. (2017). Deep learning with Python. Manning Publications.

[43] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[44] Smith, J. Q. (2020). Data Preprocessing Techniques for Intrusion Detection Systems. Journal of Cybersecurity, 50(3), 302-318.

[45] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

[46] Gopal, S. B., Poongodi, C., Nanthiya, D., Kirubakaran, T., Logeshwar, D., and Saravanan, B. K. (2022, April). Autoencoder-based Architecture for Mitigating Phishing URL attack in the Internet of Things (IoT) using Deep Neural Networks. 6th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 427-431, doi: 10.1109/ICDCS54290.2022.9780673.

[47] Nanthiya, D., Gopal, S. B., Poongodi, C., Jegadeesh, K., Narendraprasath, P., Jeevanandham, J., (2022 October). Autoencoder-Based Feature Selection for Phishing URL Attack Detection in IoT Using Stacked Autoencoder (AFS-SAE). 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1-8, doi: 10.1109/ICCCNT54827.2022.9984574.

[48] Poornima, B. and Kumari, L. S. (2022 July). A Novel IoT Mobile Forensics Approach to Mitigate Phishing Attack. 4th International Conference on Communication and Information Processing (ICCIP), Pune, India, doi: http://dx.doi.org/10.2139/ssrn.4322213.

[49] Bustio-Mart, L., Alvarez-Carmona, M. A., Herrera-Semenets, V., Feregrino-Uribe, C., Cumplido, R. (2022). A lightweight data representation for phishing URL detection in IoT environments. Information Sciences, 603, 42-59, doi:https://doi.org/10.1016/j.ins.2022.04.059.

[50] Naaz, S. (2021). Detection of Phishing in the Internet of Things Using Machine Learning Approach. International Journal

of Digital Crime and Forensics (IJDCF), 13(2), 1-15, doi: http://doi.org/10.4018/IJDCF.2021030101.

[51] Abbas, S. G., Vaccari, I., Hussain, F., Zahid, S., Fayyaz, U. U., Shah, G. A., Bakhshi, T., and Cambiaso, E. (2021 July). Identifying and Mitigating Phishing Attack Threats in IoT Use Cases Using a Threat Modelling Approach. Sensors (Basel), DOI https://doi.org/10.3390/s21144816.

[52] Tewari, A. and Gupta, B. B. (2020). Security, privacy, and trust of different layers in Internet-of-Things (IoTs) framework. Future Generation Computer Systems, 108, 909-920, doi: https://doi.org/10.1016/j.future.2018.04.027.

[53] Bojjagani, S., Brabin, D., Rao, P. V. (2020). PhishPreventer: A Secure Authentication Protocol for Prevention of Phishing Attacks in Mobile Environment with Formal Verification. Procedia Computer Science, 171, 1110-1119, doi: https://doi.org/10.1016/j.procs.2020.04.119.

[54] Lam, T., and Kettani, H. (2019 April). PhAttApp: A Phishing Attack Detection Application. International Conference on Information System and Data Mining (ICISDM 2019). Association for Computing Machinery, New York, USA, 154–158, doi: https://doi.org/10.1145/3325917.3325927.

[55] Zaw, S. K., and Vasupongayya, S. (2019). A Case-Based Reasoning Approach for Automatic Adaptation of Classifiers in Mobile Phishing Detection. Journal of Computer Networks and Communications, vol. 2019, doi https://doi.org/10.1155/2019/7198435.

[56] Azam, F., Munir, R., Ahmed, M., Ayub, M., Sajid, A., and Abbasi, Z. (2019). Internet Of Things (IoT), Security Issues And Its Solutions. Science Heritage Journal (GWS), 3(2), 18- 21. doi: 10.26480/gws.02.2019.18.21.

[57] Ndibwile, J. D., Luhanga, E. T., Fall, D., Miyamoto, D., Blanc, G., and Kadobayashi, Y. (2019). An Empirical Approach to Phishing Countermeasures Through Smart Glasses and Validation Agents. IEEE Access, 7, 130758-130771, doi 10.1109/ACCESS.2019.2940669.

[58] Spaulding, J., and Mohaisen, A. (2018). Defending the Internet of Things Against Malicious Domain Names using D-FENS. 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 387-392, doi 10.1109/SEC.2018.00051.