Received 2 June 2024; accepted 19 June 2024. Date of publication 26 June 2024; date of current version 17 July 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3419570

Latent Semantic Analysis and Graph Theory for Alert Correlation: A Proposed Approach for IoT Botnet Detection

MOEMEDI LEFOANE [1] (Member, IEEE), IBRAHIM GHAFIR[1], SOHAG KABIR [1], IRFAN-ULLAH AWAN[1], KHALIL EL HINDI [2], AND ANAND MAHENDRAN[3]

[1]Faculty of Engineering and Digital Technologies, University of Bradford, BD7 1DP Bradford, U.K.

[2]College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

[3]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore 632014, India

CORRESPONDING AUTHOR: M. LEFOANE (e-mail: m.lefoane@bradford.ac.uk)

This work was supported by the Researchers Supporting Project, King Saud University, Riyadh, Saudi Arabia, under Grant RSPD2024R953.

ABSTRACT In recent times, the proliferation of Internet of Things (IoT) technology has brought a significant shift in the digital transformation of various industries. The enabling technologies have accelerated this adoption. The possibilities unlocked by IoT have been unprecedented, leading to the emergence of smart applications that have been integrated into national infrastructure. However, the popularity of IoT technology has also attracted the attention of adversaries, who have leveraged the inherent limitations of IoT devices to launch sophisticated attacks, including Multi-Stage attacks (MSAs) such as IoT botnet attacks. These attacks have caused significant losses in revenue across industries, amounting to billions of dollars. To address this challenge, this paper proposes a system for IoT botnet detection that comprises two phases. The first phase aims to identify IoT botnet traffic, the input to this phase is the IoT traffic, which is subjected to feature selection and classification model training to distinguish malicious traffic from normal traffic. The second phase analyses the malicious traffic from stage one to identify different botnet attack campaigns. The second stage employs an alert correlation approach that combines the Latent Semantic Analysis (LSA) unsupervised learning and graph theory based techniques. The proposed system was evaluated using a publicly available real IoT traffic dataset and yielded promising results, with a True Positive Rate (TPR) of over 99% and a False Positive Rate (FPR) of 0%.

INDEX TERMS Botnet attack, Internet of Things, graph-based analysis, intrusion detection system, machine learning, latent semantic analysis.

I. INTRODUCTION

IN RECENT years, there has been a growing concern regarding botnet attacks, particularly IoT botnet attacks. A recent security intelligence report has highlighted an upward trend in the number of command and control (C&C) servers during the last quarter of 2023 [1]. The report indicates that the number of active C&C servers has increased by over 16%, with some of the most prominent networks experiencing a surge in C&C server activities. This increase has resulted in a corresponding rise in botnet-related malware and IoT botnet activities. According to Nokia's 2023 Threat Intelligence Report [2], the number of IoT botnet activities has significantly increased, leading to a surge in Distributed Denial of Service (DDoS) attacks originating from IoT devices. The report indicates that the number of devices involved in these attacks has increased from 200,000 to 1 million. This trend has continued in subsequent years, with a 56% increase in the number of C&C servers observed in the previous year (2022 Q4), according to the Spamhouse Project [3].

In response to these threats, considerable research efforts have been made to combat botnet attacks [4], [5], [6], [7], [8], [9], [10]. Machine Learning (ML) has been a popular approach, with many proposing anomaly-based detection methods. Similarly, graph-based botnet detection approaches have been proposed, with a primary focus on the feature selection stage of the ML detection process. Over the years, numerous ML and graph-based approaches have been developed for botnet detection. For example, [11], [12] utilised graph-based features within ML detection approaches. While these methods have shown promising performance in terms of accuracy of detection, their primary limitation lies in an inability to differentiate between detected botnet attack stages as part of a botnet campaign or isolated attacks. Recognising the correlation between botnet attack stages is crucial for robust botnet campaign detection, highlighting the imperative to develop an approach with enhanced capability to identify these attack stages.

To enhance the capability of existing Botnet detection approaches, this work proposes a novel approach to botnet detection based on graph theory and LSA. The proposed approach utilises LSA to identify inherent patterns of communication (candidate clusters/categories) in network traffic alerts. The identified candidate clusters are then mapped to botnet attack stages, which are validated later. Alert correlation is then performed using graph theory concepts to determine correlated stages related to the same botnet campaign. The proposed approach enhances accuracy and efficacy in botnet detection by focusing on alert correlation analysis which is part of ML, thus addressing the limitations of the existing detection approaches. The main contributions of this work are summarised as follows:

- The proposed approach for detecting IoT botnets facilitates the robust development of an ensemble alert correlation approach. This approach effectively identifies botnet stages, resulting in a high TPR and a low FPR. The alert correlation approach is unsupervised and therefore capable of detecting emerging patterns relating to IoT botnet attacks.
- The use of Term-Frequency-Inverse-Document-Frequency (TF-IDF) is crucial for transforming IoT traffic into an input matrix processed by LSA. This approach effectively measures feature frequency and reduces the impact of noisy features. Moreover, automatic threshold selection of an optimal number of clusters is employed to enhance LSA. LSA reveals latent similarities, enabling efficient detection of candidate Botnet stages.
- The use of graph theory for alert correlation analysis of malicious activities which correlates different categories leading to the effective detection of coordinated botnet attack stages with reduced false alarms.

The rest of this paper is organised as follows, Section II outlines related work, Section III covers relevant theoretical background, Section IV explains the proposed methodology, Section V discusses the results and finally, Section VI concludes the paper.

II. RELATED WORKS ON BOTNET AND MALWARE DETECTION

In their study, Ghafir et al. [13] proposed a novel approach for detecting botnet C&C. This approach comprises two stages. The first stage involves four distinct modules that operate in tandem to identify malicious IP addresses, detect domain flux, tor connections, and malicious SSL certificates, respectively. These modules utilise intelligence feeds and rely on blacklisting IP addresses, SSL certificates, and tor server lists for the detection of malicious IP addresses, malicious SSL certificates, and tor connections, respectively. Conversely, the fourth module focuses on domain flux detection. The second stage of the approach is a framework for alert correlation, which aims to minimise false alarms from the output generated by the first stage. The proposed approach has undergone an evaluation and has demonstrated promising results, with a TPR of 82.3% and a FPR of 13.6%.

Alharbi and Alsubhi [11] proposed a ML-based approach that employs graph features for training. Conventional methods utilise flow-based features that are often characterised by high computational overheads and fail to capture the communication of network traffic. The proposed approach mitigates these shortcomings using feature selection based on graph theory concepts. The researchers extract graph features and evaluate them using Pearson correlation to identify the most suitable features that maximise accuracy. The effectiveness of their approach was evaluated on two datasets, namely CTU-13 and IoT23. The results show that their approach outperforms the state-of-the-art in terms of precision and accuracy.

The study conducted by Daya et al. [12] proposed a Botnet detection system that utilises flow-based features. However, these features have certain limitations such as the inability to capture communication patterns from network traffic and also tend to suffer from computational overheads. To address these limitations, the proposed system employs graph-based features for feature selection. These features are then used to train ML models for detection purposes. The system is deployed in two phases, where the first phase prunes the benign traffic from network traffic, and the second phase detects botnet attacks. The proposed system is reported to perform well in an online setting.

Yang et al. [14] have proposed a system for detecting Advanced Persistent Threats (APTs) utilising a causal correlation technique aided by semantic analysis. This system relies on existing alerts, which are input into the correlation technique for mining causal relationships between them. The proposed correlation technique utilises Latent Dirichlet Allocation (LDA) to model alert chains and analyse alert correlations. The utilisation of LDA allows for the capturing of hidden attack intents, which aids in reconstructing APTs scenarios, and the result is the revelation of semantic contexts

for these alert chains. The proposed system is capable of detecting multi-stage threats that span a long time by mining causality between anomalous events. LDA has also been proposed as an unsupervised learning approach for the detection of Multi-Stage attacks based on semantic similarities [15].

Husák et al. [16], in their research, proposed a sequential pattern mining methodology to extract patterns from cyber security alerts. Rule mining was also employed in their analysis. Furthermore, the study surveyed the applicability of alert correlation and attack prediction in cyber security. The authors then evaluated the effectiveness of both Rule mining and sequential pattern Mining methods in detecting cyber attacks. Other works include Sequential Pattern Mining been proposed for cyber attack mitigation in IDSs [17].

Arshad et al. [7] proposed a system for the detection of IoT botnet attacks. The system employs an ensemble method that utilises ML algorithms to analyse network traffic. The three traditional ML algorithms employed in the system are K-Nearest Neighbors (KNN), Decision Tree, and Random Forest. After analysing the network traffic, the system identifies suspicious behaviour that indicates the presence of a botnet attack. The authors evaluated their system's performance using the publicly available CTU-13 dataset and reported an accuracy of 99.7% in 12.99 seconds. Other studies have proposed ML-based approaches that focus on feature selection to improve detection effectiveness [18].

In a recent study, Wang et al. [19] employed honeypots to model IoT botnet attacks on medical devices. The study reveals that botnet attacks exhibit characteristic patterns, with a high correlation between COVID-19 and botnet attacks. The study further shows that botnet attacks are often associated with bots within the same network segment and that adversaries frequently employ repetitive attempts to ensure a successful hit. To uncover these patterns, the study proposed a botnet inference model using unsupervised learning approaches. The model was then evaluated through a series of simulation experiments. The use of unsupervised learning allows for the effective uncovering of previously unknown patterns of botnet attacks.

In their research, Ghafir et al. [20] proposed a system aimed at detecting APTs. The system comprises two phases; the first phase involves scenario reconstruction of APTs from traffic, while the second phase is responsible for attack decoding. For the latter, the Hidden Markov Model (HMM) is employed to determine the likely sequence of APT attack stages from the correlated alerts. The evaluation of the proposed system yielded promising results, with an accuracy of 91.80% for predicting APT attack stages. Furthermore, the prediction of the next step of the attack demonstrated an accuracy of 66.50%, 92.70%, and 100% for two, three, and four correlated alerts, respectively.

Haas and Fischer [21] proposed a Graph-based Alert Correlation (GAC) algorithm, which identifies multi-step attack scenarios and performs a Multi-Stage attack reconstruction on the identified alert set. The algorithm's robustness against false alarms and its scalability with increasing alerts have been reported. Furthermore, it is capable of detecting distributed attacks.

Alshamrani et al. [22] surveyed APTs detection techniques. The study emphasises the sophisticated nature of these attacks and their tendency to occur over an extended period. The researchers found that APTs have become increasingly prevalent and prominent in recent years, with the malware constantly evolving. In their analysis of recent works on APT detection, the researchers found that most detection methods focus only on specific stages of the APT attack lifecycle. However, limited research exists on detecting APTs across all stages. The study suggests that methods used for detecting APTs should be applicable across different stages of the attack lifecycle. The researchers also recommend the use of complex correlation approaches and behavioural analysis of users and systems across the network. Overall, the study underscores the need for continued research and development of effective APT detection methods particularly utilising complex correlation approaches to mitigate the risks associated with these sophisticated and persistent cyberattacks.

The study conducted by Kidmose et al. [23] delves into the issue of bot infections and the challenges associated with the alerts generated by IDSs, which are often raised in large numbers. The authors note that although the alerts generated by IDSs for botnet attacks tend to be highly correlated, calling for manual processing to determine which ones are relevant. As such, the study proposes a neural network approach for performing alert correlation on the generated alerts, thereby reducing the need for manual processing, which can be time-consuming. One noteworthy advantage of the proposed approach is that it does not require feature extraction or domain-specific knowledge. The authors evaluated the method using labelled IDS alert data and demonstrated its efficacy in reducing the number of alerts that require manual processing.

Rahal et al. [24] proposed an architecture for the detection of botnet attacks and prediction of DDoS attacks. The study highlights that the conventional approach of mitigating such attacks involves overprovisioning and a sinkhole of malicious traffic. However, the authors propose a technique that predicts such attacks at an early stage and aims to address the challenges involved in their detection. The proposed architecture employs unsupervised learning to cluster network devices based on causality relationships between them. The authors evaluated their proposed architecture using the CTU-13 botnet dataset.

Maestre Vidal et al. [25] have reported that IDSs based on anomaly identification techniques generate a large volume of reports on malicious activities in a monitored network. This large number of reports poses significant challenges in terms of analysis and management. In response to this challenge, the authors proposed an alert correlation system that focuses on payload analysis of network traffic, to detect malware attacks. The proposed framework is designed to analyse

the monitored network traffic's payload, paying attention to the characteristics of the models built during the training process. The framework has been shown to perform attack reconstruction of potential threats, enabling the detection of malware attacks. Overall, the authors' findings suggest that their proposed alert correlation system is an effective tool for detecting malware attacks in a monitored network.

Khosravi and Ladani [26] conducted a study that highlights the stealthy nature of APTs, which are slow, stepwise, and typically long-term planned. These attacks may also exploit zero-day vulnerabilities, moreover, they are continually evolving in their tactics. The complex nature of APTs makes detecting them a challenging task. As such, most current approaches fail to effectively detect APTs. To address the challenges of APTs attacks, the study proposes a real-time detection method that utilises correlation and causal analysis on alerts. The proposed method can compute an infection score by modeling and determining causal relationships between steps of APTs attacks. Furthermore, the study introduces dynamic programming as part of their detection strategy. To evaluate their method, the authors utilised a semi-real-world dataset and simulation. The proposed method demonstrates promising results in effectively detecting APTs.

Numerous effective techniques have been successfully employed within the Information Retrieval (IR) domain. Typically unsupervised, these techniques are utilised for clustering and leverage their core capabilities to capture semantic similarities from latent patterns in the data. They are particularly suitable for analysing unlabeled data, thereby addressing limitations across various domains. These results underscore the potential of unsupervised semantic mining techniques in diverse domains. Notably, these techniques have demonstrated effectiveness in text mining and image retrieval, drawing from the field of computer vision [27], [28], [29]. They excel in extracting representative features that can be effectively employed for classification in ML approaches.

The existing body of literature has predominantly focused on the application of machine ML for botnet detection, primarily as a multi-class classification problem. Likewise, within the domain of graph-based methodologies, researchers predominantly leverage graph theory concepts to select features for training multi-stage classification models aimed at various botnet stages. These approaches yield promising results evidenced by evaluation metrics such as accuracy [11], [12], [30]. However, a major limitation of these approaches is their inability to differentiate between detected stages as part of a botnet campaign or isolated attacks. Therefore these approaches hinder the ability to identify botnet campaigns effectively. Recognising the correlation between botnet stages is crucial for robust botnet campaign detection [22]. Early-stage detection can play a pivotal role in preventing more severe stages, such as Distributed Denial of Service (DDoS) attacks, and can offer a more comprehensive understanding of botnet lifecycles and their underlying objectives. Therefore, the development of a comprehensive approach to botnet detection that effectively identifies individual attack stages is imperative.

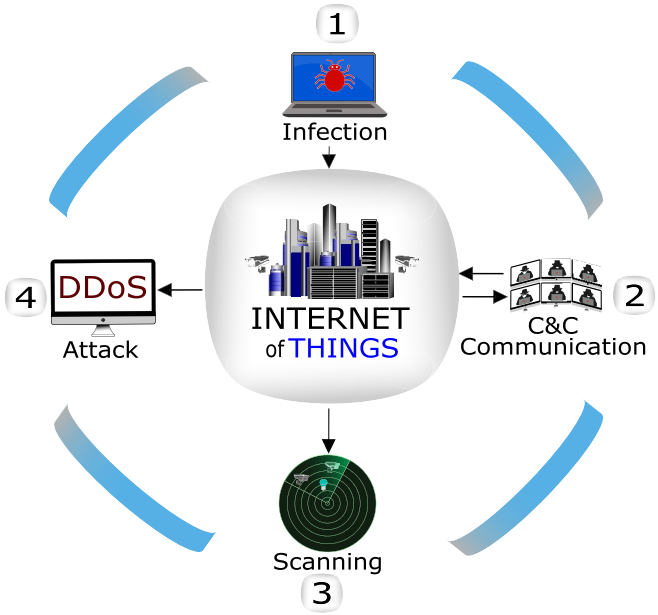

FIGURE 1. Typical life-cycle of IoT Botnet Attack (Derived from [31]).

III. PRELIMINARIES

This section provides an overview and theoretical background in Botnet attacks, LSA, Graph-Based approaches and Alert Correlation.

A. IOT BOTNET ATTACKS

Botnet attacks are a form of Multi-Stage attacks (MSAs) that are executed through various stages, such as the infection stage, C&C stage, scanning stage, and an attack stage, such as a DDoS attack [31]. The typical stages of a botnet attack, including IoT, are illustrated in Fig. 1. In this figure, the scanning stage corresponds to Reconnaissance, the infection stage to Malware injection, while C&C connection represents the C&C connections and subsequent communication to the C&C server. Finally, the Command Execution stage depicts the final stage of the botnet attack, such as a DDoS attack.

During the infection stage, the targeted IoT device is initially infected with bot malware. Subsequently, the infected device establishes connections with the C&C server and periodically receives instructions from the server. Additionally, the infected device communicates with the server to indicate that it is still operational. It is also common for the infected device to download malicious executables that match the device's processor architecture to increase the likelihood of success of a botnet attack. The downloaded malicious executable that matches the device's architecture is executed, and the primary goal is to infect more vulnerable devices to create a sufficient army of infected devices, which is the next stage in the attack lifecycle, the scanning stage. This army of infected devices can then be used to effectively launch

an attack, such as a DDoS attack, which is the final stage of the botnet attack lifecycle.

The capabilities of Botnet attacks lie heavily on the C&C communication mechanisms, as such the evolution of Botnet attacks can be classified according to C&C Server communication architecture [32]. Essentially, the first generation of Botnet attacks was in the 90's Botnet attacks relied on Internet Relay Chat(IRC) communication. These Botnets are they are easy to detect and the centralised architecture makes it easy to take them down, including DDoS-targeted attacks. The Botnet attacks evolved into the second generation which relied heavily on HTTP-Based communication protocol, rendering the C&C communication indistinguishable from benign traffic, and thereby making it challenging to detect, however since it was still centralised, it also made it easy to take down. Finally, the third-generation Botnet attacks employed Peer-to-Peer (P2P) networks for C&C communication, thus eliminating centralised servers and as such making it difficult to take them down, however, they introduce communication overheads that impact their performance leading to increased latency and high resource consumption.

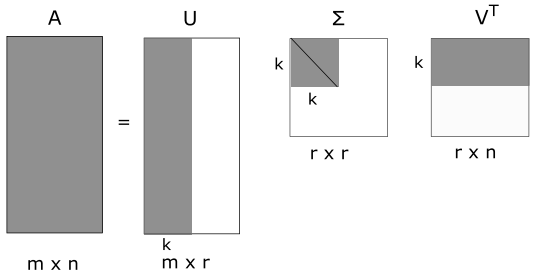

FIGURE 2. Singular Value Decomposition Pictorial Representation [36].

```
1   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
2   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
3   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
4   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
5   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
6   Dst IP:192.186.25.45,     Dst Port:52869,   history:s
7   Dst IP:96.244.2.114,      Dst Port:27015,   history:s
8   Dst IP:96.244.2.114,      Dst Port:27015,   history:s
9   Dst IP:96.244.2.114,      Dst Port:27015,   history:d
10  Dst IP:96.244.2.114,      Dst Port:27015,   history:s
11  Dst IP:142.11.219.83,     Dst Port:45,      history:d
12  Dst IP:142.11.219.83,     Dst Port:45,      history:d
```

FIGURE 3. Sample Network Traffic.

B. LATENT SEMANTIC ANALYSIS

LSA is a text analysis technique that was introduced in the scientific literature [33], [34]. LSA, driven by Singular Value Decomposition (SVD), is a powerful approach for extracting latent features from data. It has been proven successful in various applications, including computer vision and clustering. The proposed system leverages the power of SVD to identify candidate botnet categories and map them to attack stages. The proposed system utilises LSA in the clustering phase to generate clusters of candidate botnet categories. These clusters are then mapped to stages of a botnet attack. SVD takes a matrix of numbers as input and decomposes it into three matrices that capture linear transformation, namely rotation, stretch, and rotation. The SVD formula is given by equation (1) [35].

$$SVD = U\Sigma V^T \tag{1}$$

SVD is a powerful mathematical tool that can be used to obtain a compact representation of an input matrix. The decomposition results in three matrices: a left singular matrix U, a diagonal matrix Σ, and a right singular matrix V^T. The matrix U is of dimensions $m \times r$, where m is the number of rows in the input matrix, and r is a user-defined parameter. Matrix Σ is a diagonal matrix of dimensions $r \times r$, containing the singular values of the input matrix arranged in descending order. Finally, matrix V^T is of dimensions $r \times n$, where n is the number of columns in the input matrix.

The SVD is often used for data compression, as the diagonal matrix Σ contains values that decrease in magnitude as their corresponding singular vectors become less important. By selecting a relatively small value of r, i.e., reducing it to k, we can represent the input matrix as a subset of $U\Sigma V^T$, effectively compressing the data while retaining most of its original information.

In the context of network traffic analysis, the value of r can be used to determine the number of clusters that the input matrix can be divided into. Matrix U captures instances by their similarity to different concepts, while matrix V^T captures features by their similarity to these same concepts. The resulting matrices can be used to identify patterns in the data and to develop more effective algorithms for network intrusion detection and other related tasks.

Fig. 2 provides a visual representation of the resulting matrices after SVD decomposition. Overall, the SVD provides a valuable tool for reducing the dimensionality of large datasets and for identifying hidden patterns and structures within the data.

The illustration provided in Fig. 3 displays a set of sample data related to network traffic, which comprises three distinct features to be subjected to analysis through LSA, specifically using SVD for clustering. This sample data serves as an example to elucidate the process of transforming the input into a matrix of numerical values and ultimately to demonstrate how LSA decomposes the input matrix.

Table 1 serves as an illustration of the transformed network traffic matrix. Prior to being processed by SVD, the network traffic is altered into X features matrix. In this matrix, rows represent instances of communications that take place between two hosts or devices, while columns denote features such as the protocol used, port number, IP addressed, among others. The frequency of each feature value is recorded as a count.

When examining the matrix, it becomes evident that certain feature values occur in all network traffic instances. These values, when considered solely in terms of their

TABLE 1. Sample transformed matrix for network traffic with count scores.

history:D	history:S	Dst IP:142.11.219.83	Dst IP:192.186.25.45	Dst IP:96.244.2.114	Dst Port:27015	Dst Port:45	Dst Port:52869
0	1	0	1	0	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	0	0	0	1
0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	0
1	0	0	0	1	1	0	0
0	1	0	0	1	1	0	0
1	0	1	0	0	0	1	0
1	0	1	0	0	0	1	0

frequency, seemingly possess more information that can distinguish between various categories of network traffic. However, this is not true as the value that is present in all clusters/categories mined does not convey more information that helps distinguish between clusters. The goal, therefore, is to score these types of features slightly less indicating that they are not as informative as feature values that are more predominant in respective traffic categories/clusters, a TF-IDF scoring mechanism is employed.

TF-IDF considers not only the frequency of a particular feature value but also the occurrence of instances of the network traffic. For instance, a value that occurs in all the rows does not have any useful information to distinguish between categories in data. On the other hand, a value that tends to be present only in one cluster, ideally should have a higher score, TF-IDF achieve this goal. The formula for TF-IDF is derived from variations of the formula by Manning et al. [37] and is given by Equation (2). The $tf_{t,d}$ variable captures the document frequency of a term's occurrence in a document, which is typically normalised as needed. In the *idf* part of the formula, $log\frac{N}{df_t}$, N represents the total number of documents in the collection, while *df* denotes the number of documents that contain the term t. The purpose of idf_t is to provide higher scores to terms that occur less frequently across different categories of the analysed dataset.

Utilising TF-IDF enhances the quality of the candidate categories of network traffic in botnet detection. Additionally, it can uncover some of the emerging patterns in the network traffic that relate to an emerging botnet attack.

The manifestation of botnet stages, especially the infection phase, may not produce a substantial quantity of traffic that can lead to the emergence of patterns that are identifiable by the clustering module. In some cases, the initial infection phase may occur only once, which can be viewed as an insignificant activity by the analysis module. Nevertheless, the initial infection phase is typically followed by further infection, in which the infected device connects to the C&C server and downloads additional malicious code that is relevant to the device architecture. If these subsequent activities are detected during the analysis, they can provide additional evidence that the device was previously infected, thereby instilling greater confidence in the initial infection event. Furthermore, if the initial infection goes unnoticed, the detection of further infection can trigger an alarm that provides more insights to the Security Operations Center about this infection.

$$tf.idf_{t,d} = tf_{t,d} \times log\frac{N}{df_t} \tag{2}$$

The use of Bigram in network security analysis is a common practice that strengthens the results by considering features occurring within proximity to one another. This approach is especially useful when considering IP addresses together with port numbers. The discovered features are then utilised by the network security team, with the resulting alerts carrying more informative insights into attack behaviour and the precise goal of the attack.

In experimenting with different numbers of topics (clusters) K, it is observed that as the value of K increases from a smaller value to a larger value, the discovered clusters or categories start with more informative terms that have a major contribution to deciding which clusters are relevant. This is because the diagonal matrix captures these concepts and orders them from the most variance to the least. As the value of K continues to increase, we observe duplicate clusters or candidate categories, and the deciding terms start to increasingly omit device identifiers as part of the deciding features.

It is worth noting that computing the accuracy of the first few concepts encoded by the first few values of the diagonal matrix reveals that most of the concepts are captured by the first few values, and the resulting clusters are still representative of almost all the entire traffic. Therefore, it is essential to consider an optimal value of K that ensures a balance between informative terms and duplicate clusters, while still being representative of the entire network traffic.

Figures 4, 5 and 6 shows the decomposed matrices resulting from the SVD for the input matrix derived from Table 2.

The following analysis pertains to the decomposition of a given input matrix into its corresponding thematic concepts through Singular Value Decomposition (SVD). The semantic concepts are represented by each column of the matrix (shown in Fig. 4), and each row represents an instance of the input data. The resulting scores for each row and column represent the strength of each semantic concept and its association with the input data. The highest scores for each instance and semantic concept are highlighted in red, green, and blue, respectively. The results demonstrate that the input data can be characterized by three significant themes,

3.98394525e-01	-1.47534792e-01	-3.95029487e-02	-8.14517793e-02	-4.52238816e-01	-1.65534025e-02	-5.41233404e-01	-5.59497291e-01
3.98394525e-01	-1.47534792e-01	-3.95029487e-02	-8.14517793e-02	6.74721757e-01	-2.05734500e-01	-4.59231374e-01	2.32894940e-01
3.98394525e-01	-1.47534792e-01	-3.95029487e-02	-8.14517793e-02	1.42157954e-01	1.53394397e-01	6.00440222e-01	-3.84502953e-01
3.98394525e-01	-1.47534792e-01	-3.95029487e-02	-8.14517793e-02	1.36229723e-02	7.38577797e-01	3.89619216e-02	2.84703325e-01
2.84408480e-01	-1.17344826e-01	2.87196808e-01	6.79803512e-01	1.49847469e-01	-2.87439501e-01	1.93840104e-01	-1.67549883e-01
3.98394525e-01	-1.47534792e-01	-3.95029487e-02	-8.14517793e-02	-5.10991426e-01	-4.15084442e-01	1.89368552e-01	5.74809471e-01
1.91884958e-01	4.73413547e-01	-1.13917406e-01	-1.31667935e-01	5.11794019e-02	-9.81730416e-02	6.62047926e-02	-5.72255435e-02
1.91884958e-01	4.73413547e-01	-1.13917406e-01	-1.31667935e-01	5.11794019e-02	-9.81730416e-02	6.62047926e-02	-5.72255435e-02
1.13917372e-01	4.43084184e-01	1.82656753e-01	5.48546687e-01	-1.68557527e-01	3.23329395e-01	-2.18043112e-01	1.88470277e-01
1.91884958e-01	4.73413547e-01	-1.13917406e-01	-1.31667935e-01	5.11794019e-02	-9.81730416e-02	6.62047926e-02	-5.72255435e-02
3.22811932e-02	6.60032783e-02	6.47078266e-01	-2.75483593e-01	6.36264242e-18	2.59519060e-32	8.26223946e-32	1.44444746e-32
3.22811932e-02	6.60032783e-02	6.47078266e-01	-2.75483593e-01	6.36264242e-18	2.59519060e-32	8.26223946e-32	1.44444746e-32

FIGURE 4. Resulting Decomposed U Matrix for sample Network traffic.

2.49904686e+00							
	1.84459608e+00						
		1.43836914e+00					
			5.32282170e-01				
				6.35292485e-17			
					2.65035275e-31		
						1.55458547e-32	
							7.18911667e-34

FIGURE 5. Resulting Decomposed Σ Matrix for sample Network traffic.

1.84655305e-02	5.33272216e-01	2.44117912e-03	5.81914661e-01	1.37874044e-01	1.37874044e-01	2.44117912e-03	5.81914661e-01
2.15175486e-01	1.49802508e-01	6.96234827e-02	-2.24155804e-01	6.40730915e-01	6.40730915e-01	6.96234827e-02	-2.24155804e-01
5.61296591e-01	-7.61299726e-02	5.65939987e-01	5.39260430e-02	-1.27980840e-01	-1.27980840e-01	5.65939987e-01	5.39260430e-02
6.84588666e-01	-4.26721053e-01	-3.76516443e-01	1.77272537e-01	3.78624570e-02	3.78624570e-02	-3.76516443e-01	1.77272537e-01
4.11877444e-01	7.10839871e-01	-1.81917318e-01	-2.77120791e-01	-2.29438404e-01	-2.29438404e-01	-1.81917318e-01	-2.77120791e-01
0.00000000e+00	1.52655666e-16	1.86963095e-01	6.81941169e-01	-1.02146589e-03	1.02146589e-03	-1.86963095e-01	-6.81941169e-01
0.00000000e+00	1.66533454e-16	6.81533730e-01	-1.86814329e-01	2.47422895e-02	-2.47422895e-02	-6.81533730e-01	1.86814329e-01
0.00000000e+00	-3.29597460e-17	-2.35918560e-02	7.52652728e-03	7.06673033e-01	-7.06673033e-01	2.35918560e-02	-7.52652728e-03

FIGURE 6. Resulting Decomposed V^T Matrix for sample Network traffic.

TABLE 2. Sample transformed matrix for network traffic with TF-IDF scores.

history:D	history:S	Dst IP:142.11.219.83	Dst IP:192.186.25.45	Dst IP:96.244.2.114	Dst Port:27015	Dst Port:45	Dst Port:52869
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.48281331	0.00000000	0.61922989	0.00000000	0.00000000	0.00000000	0.61922989
0.00000000	0.4152511	0.00000000	0.00000000	0.64325987	0.64325987	0.00000000	0.00000000
0.00000000	0.4152511	0.00000000	0.00000000	0.64325987	0.64325987	0.00000000	0.00000000
0.61883151	0.00000000	0.00000000	0.00000000	0.55544917	0.55544917	0.00000000	0.00000000
0.00000000	0.4152511	0.00000000	0.00000000	0.64325987	0.64325987	0.00000000	0.00000000
0.52977168	0.00000000	0.59972576	0.00000000	0.00000000	0.00000000	0.59972576	0.00000000
0.52977168	0.00000000	0.59972576	0.00000000	0.00000000	0.00000000	0.59972576	0.00000000

which are well-captured by the SVD-generated matrix. The remaining rows can be discarded while still representing the thematic concepts well, using only three columns.

Illustrated in Fig. 5, the diagonal values of the matrix representing the strength of the semantic concepts are in descending order and illustrate the semantic strength of each

cluster. The highlighted blue values indicate that the input data is predominantly comprised of three significant themes, consistent with the input data. In unsupervised learning, where the number of clusters must be provided, these values can be useful in determining how many coherent clusters the input data can be split into. The highlighted red values, on the other hand, are fractions significantly smaller than the blue values, indicating that they contribute less to the semantic strength of the overall matrix.

Finally, in the matrix V^T, each row represents a semantic concept, and each column represents the corresponding feature score for each feature value. For instance, the highest scores for concept 1 are for the second, fourth, and eighth features, which are history:s, Dst IP:192.168.25.45, and Dst Port:52869. Interestingly, even though the feature value history:s occurs more frequently, its score is not the highest for concept one features, due to the use of TF-IDF. The importance of the feature history:S and history:D for concept two is highlighted, indicating their significance for that concept. Overall, it is evident that the entire input matrix can be approximated by the first few values of this matrix, demonstrating its usefulness for data compression, such as in image compression.

C. GRAPH-BASED NETWORK ANALYSIS

Fig 7 illustrates a typical scenario of a network under botnet attack. In the illustration a graph capturing malicious activity has been generated, it shows that two infected devices performing scans on devices illustrated in blue and green. The infected devices are shown in red and finally, the C&C server is shown in magenta colour.

Graph theory concepts including In-Degree, Out-Degree of different nodes, and degree of centrality are used to capture the relationship between different stages of botnet attack, and in the process effectively identify infected devices, C&C server communication and DDoS attack stage.

Equation (3) to Equation (6) [12] illustrate different measures of interests from the graph-based theory approach from In-Degree through to the computation of betweenness centrality.

Equation (3) is a piecewise function that assigns 1 or 0 depending on whether the edge $e_{i,j}$ is from i to j, therefore 0 is assigned, otherwise if $e_{i,j}$ then 1 is assigned if $e_{j,i}$, This function is utilised in Equation (4) and Equation (5) to differentiate between incoming edges and outgoing edges of a respective node for which the measures are computed.

$$\mathcal{F}(e_{i,j}) = \begin{cases} 0, \text{ if } e_{i,j} \in E \\ 1, \text{ otherwise} \end{cases} \tag{3}$$

Equation (4) computes the In-Degree, it is the number of edges from other nodes that are incoming to a particular node. This also applies to weighted edges for a graph which is computed by adding up all the weight of the incoming edges to a particular node.. For network security if a device was under a DDoS attack, its In-Degree score will be significantly higher than normal and the rest of the devices in the network.

$$f_{i,0} = \sum_{v_j \in V, v_i \neq v_j} \mathcal{F}(e_{j,i}) \quad \forall v_i \in V \tag{4}$$

Equation (5) computes Out-Degree, which is the number of edges from other nodes that are outgoing from a particular node. Also, this applies to the weighted edges in which the weight of the edges capturing the importance of an edge would be summed together with respect to the originating node.

$$f_{i,1} = \sum_{v_j \in V, v_i \neq v_j} \mathcal{F}(e_{i,j}) \quad \forall v_i \in V \tag{5}$$

Equation (6) computes Betweenness Centrality score, where $\sigma_{v_j v_k}(v_i)$ is the number of shortest paths for all pairs on nodes passing through node v_i, $\sigma_{v_j v_k}$ is the total number of shortest paths between all possible pairs of nodes and V is a set of all nodes in the graph. This captures the influence of a node to other nodes in the network. This information is important in the network traffic as it captures the devices in the network that are central and most likely infected devices, if the devices have more outgoing links to other devices significantly more than the rest that would be linked to scanning other devices for vulnerabilities.

$$f_{i,4} = \sum_{v_j, v_k \in V, v_i \neq v_j \neq v_k} \frac{\sigma_{v_j v_k}(v_i)}{\sigma_{v_j v_k}} \quad \forall v_i \in V \tag{6}$$

TABLE 3. Computed centrality scores for graph in Fig 7.

C&C	ID 1	ID 2	D 1	...	D 10	D 01	...	D 012
0.043	0.036	0.043	0	...	0	0	...	0
0.167	0.500	0.583	0.042	...	0.042	0.042	...	0.042
2	1	1	1	...	1	1	...	1
2	11	13	0	...	0	0	...	0

Table 3 shows the different centrality scores for the network graph illustrated in Fig 7. In the table, C&C refers to C&C Server, ID 1 refers to Infected Device 1, ID 2 refers to Infected Device 2, D 1$\cdots$ D 10 refers to Device 1 through to Device 10 as shown in the network graph, and finally, D 01$\cdots$ D 012 refers to Device 01 to Device 012 as shown the graph. In the table, the first row captures the Degree of Betweenness centrality scores for the devices, the second row shows the Degree of centrality scores, the third row shows the In-Degree scores and finally, the fourth row shows Out-Degree centrality.

The Betweenness centrality scores are highest for the two infected devices and the C&C Server. This score captures or reveals the most influential nodes in the graph. In the context of network traffic, a graph generated from malicious traffic would highlight the hosts or devices that should be looked at when addressing the alerts as top priority when generating and sending alerts to the network security team, thus cutting the malicious activities on the hosts with the highest scores

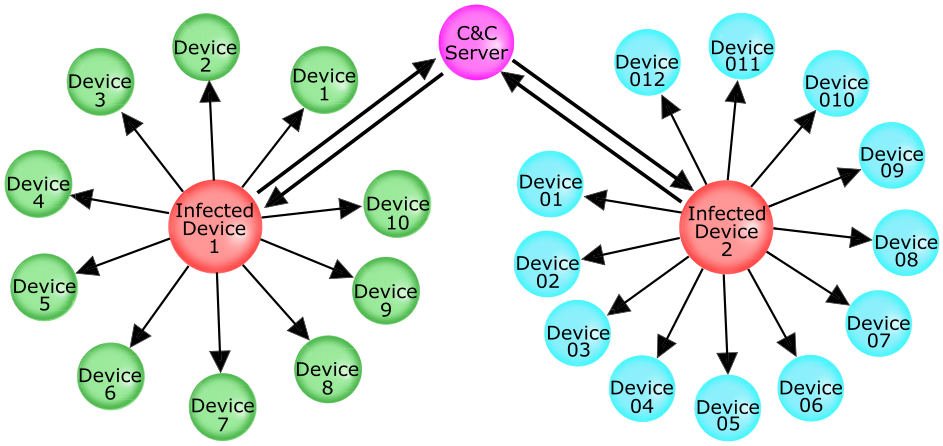

FIGURE 7. Network Graph Diagram illustrating Infected devices, scanned devices and C&C Server connections.

would reduce the computational cost that would otherwise be spent on malicious activities that may not be very efficient in eradicating an attack. This score would be suitable for identification of malicious devices that are responsible for expanding the network in this case the infected devices and C&C Server.

The degree of centrality which is shown in the second row is closely related to Betweenness centrality in that it captures how influential a particular node is taking into account the connected nodes' influence, however, it differs from the Betweenness centrality because it considers only immediate connections or edges connecting to that particular node while in the Betweenness centrality, the importance of the neighbouring node is also considered. This is well captured as the score indicates that for this measure while C&C is important and at the center of the graph, it is not as important as the infected devices which have influence with respect to scanning other devices for potential vulnerabilities to expand the botnet. This score would therefore be suitable for the identification of malicious activities such as scanning activities, The C&C Server would still have the high score but coming second after infected devices.

The In-Degree and Out-Degree scores when used together they will be able to effectively identify good candidate devices or hosts likely to be C&C Server communications at it will typically tend to be two way communication. Additionally, Out-Degree can help validate scanning activities captured by other scores as it easily identifies the devices that has a lot of activities in terms of interacting with many devices which most likely would not be a normal behaviour.

D. ALERT CORRELATION

IDSs are commonly deployed as an additional layer of defence for networks. They monitor the network in addition to other cybersecurity technologies such as the firewall. While monitoring the network their task is to detect any malicious incoming traffic and prevent it from entering the network monitored. IDSs are typically deployed towards the perimeter of the network as part of the Security Operations Center (SOC). While monitoring the network, the detection of malicious traffic triggers alerts which are generated and sent to the network security team who in turn perform all the necessary forensics and take steps to secure the affected devices. For IoT the alerts are generated in large numbers as the IoT devices are deployed in large numbers and therefore the alerts are generated in large numbers resulting in large number of alerts to process. The analysis of the alerts is therefore time-consuming if analysed manually, for Botnet attacks there are typically relationships within the network traffic which are extremely challenging to determine manually, this calls for automated analysis for aggregating related alerts and computing relationships between them, which will then lead to successful detection of botnet attacks and alert correlation is the answer.

IV. PROPOSED METHODOLOGY

The overall architecture of an IoT system including deployed Intrusion Detection System (IDS) that enables the detection of malicious activities which includes the lifecycle of an IoT botnet is illustrated in Fig. 8. The system is designed to monitor IoT traffic within an IoT network, such as in a smart city or smart home setup. The IoT traffic is generated by IoT sensors, comprising CCTV cameras and smart bulbs, that receive control signals from the cloud or users with hand-held devices, such as tablets or mobile phones. A subset of the captured traffic is transmitted to the cloud for further processing and backup purposes, which forms the IoT traffic that traverses the network.

To secure the network, network security devices, including firewalls, are deployed typically close to the network perimeter before the uplink router. Furthermore, as an added layer of security, IDSs are often positioned within the network in close proximity to the perimeter as well. To detect the botnet lifecycle, the proposed system is deployed as part of an IDS in the overall architecture setup.

A detailed methodology for the proposed IDS is illustrated in Fig. 9. The proposed IDS operates in two phases.

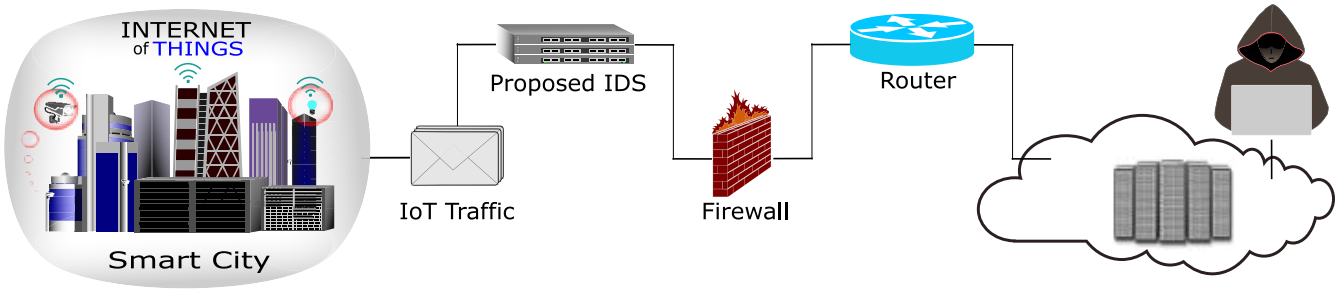

FIGURE 8. Overall Architecture for the Detection of IoT Botnet Lifecycle.

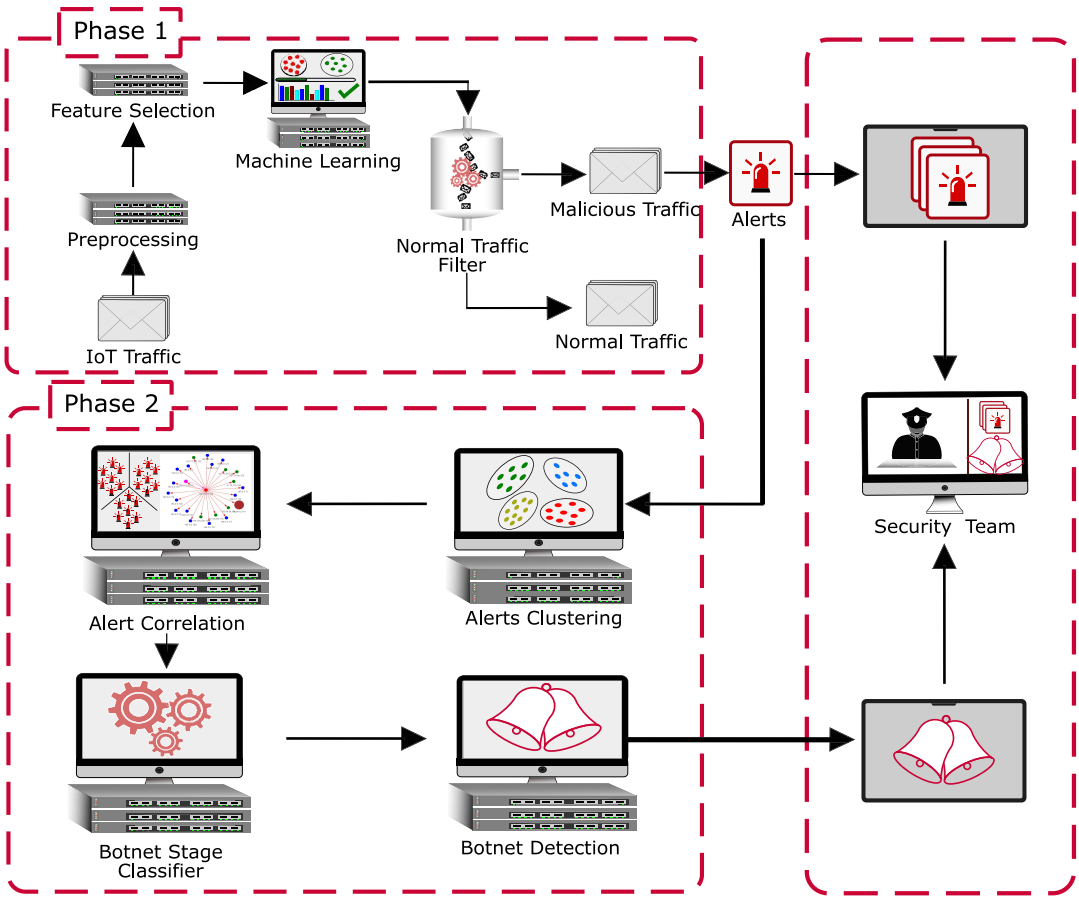

FIGURE 9. Proposed methodology for the detection of Botnet attack lifecycle.

The objective of Phase 1 is to distinguish between normal and malicious IoT traffic. An ML approach is employed for detecting and filtering out benign traffic from malicious traffic, more details about phase 1 can be found in our previous work published in [38]. These are the devices that are typically used in IoT systems such as smart home systems. The selected training features comprised the IP address, port number, transport layer protocol for the connection, connection stage history, number of IP originator bytes, and number of bytes sent by the originator. Any malicious traffic detected by this phase triggers an alert that is forwarded to the security team for further investigation. The team takes necessary actions to recover from the malicious attack and secure affected devices. Additionally, the alerts are sent to Phase 2 for further processing and analysis to detect IoT botnet lifecycles. In the second stage, the primary objective is to conduct alert correlation to identify and understand the connections among the malicious activities occurring within the network. This process involves reconstructing an attack scenario, which further helps in detecting different Botnet campaigns.

In Phase 2, malicious activity alerts are taken as input and processed through a clustering module, which groups similar alerts into their respective clusters (candidate categories). This module utilises LSA to produce a prioritised list of candidate categories that are ranked according to their volume,

Algorithm 1 Implementation Pseudocode for LSA Clustering of Malicious Traffic Alerts Into Candidate Categories

Require: *Mal_Traffic* ▷ Dataset
Require: *SVD* ▷ for SVD Decomposition
1: *feature_list* ← {} ▷ To store optimal number of features for cluster
2: *cluster_list* ← {}
3: *input_Matrix* ← *convertToMatrix*(*Mal_Traffic*) ▷ Create Rectangular Matrix with TF-IDF Scores
4: $U\Sigma V^T$ ← *SVD*(*input_Matrix*)
5: *clusters_cutoff* ← *KMeans*($\Sigma, K = 2$)
6: *feature_names* ← *getFNames*(*Feature_Matrix*)
7: $i = 0$ ▷ number of clusters counter
8: **for** *Row* in V^T **do**
9: *sorted_scores* ← *Sort*(*Row*)
10: **if** $i <$ *clusters_cutoff* **then**
11: *feature_dic* ← *Dict*(*feature_names*, *Row*)
12: *fs_cutoff* ← *KMeans*(*sorted_scores*, $K = 2$)
13: j = 0
14: *feature_list* ← {} ▷ Empty the list
15: **for** *score* in *sorted_scores* **do**
16: **if** $j <$ *fs_cutoff* **then**
17: *feature_list* ← *feature_list* + *score*
18: **end if**
19: $j \leftarrow j + 1$
20: **end for**
21: *cluster_list* ← *cluster_list* + *feature_list*
22: **end if**
23: $i = i + 1$
24: **end for**
25: **Return** *cluster_list*

with predominant categories being given higher priority. The resulting candidate categories are then forwarded to the Alert Correlation module, which aims to identify different botnet stages from the malicious clusters. The botnet stage classifier employs a rule-based approach to detect various stages of botnets buy taking into account the results of the previous modules. Algorithm 1 provides a step-by-step guide for implementing LSA clustering of malicious network traffic into candidate clusters. The main objective of this approach is to map the identified clusters to various stages of botnet attacks during the analysis process carried out by the alert correlation module. The algorithm takes in malicious traffic as input and utilises SVD for input matrix decomposition, which helps to identify and group similar patterns of traffic together.

The process begins with the instantiation of the temporary feature list (*feature_list*) to contain top features for each cluster is executed in line 1, followed by the instantiation of the list (*cluster_list*) representing cluster features for all uncovered clusters in line 2. The *input_Matrix* is prepared, as demonstrated in line 3, by transforming malicious traffic into a rectangular matrix of TF-IDF scores representing malicious traffic feature scores. This input matrix is then passed to SVD and decomposed accordingly, resulting in $U\Sigma V^T$ matrices. From these matrices, the strength of each cluster is captured by the diagonal entries of the Σ matrix in descending order. These entries are therefore used as a basis for establishing the cutoff point for the number of clusters detected with more confidence. This is achieved by employing K-Means clustering with $k = 2$. K-Means is used here as an auto threshold turning for LSA, which results in the automatic selection of the most distinct clusters and top features that best describe each candidate cluster. The steps for computing and accumulating different clusters, along with their corresponding top features are depicted in lines 8 to 24. These steps produce a set of candidate clusters subjected to further analysis by the alert correlation module, which maps the traffic to the botnet attack stages by the next module.

Once LSA categorisation has been completed, the next module focuses on the graph-based correlation analysis of malicious traffic, taking into account candidate clusters from LSA to verify detected correlated botnet stages. Algorithm 2 presents the pseudocode for implementing graph-based correlation analysis, thereby generating a graph of devices involved in botnet activities.

To generate the graph, lists and dictionary data structures are utilised, the steps are indicated by lines 1 to 8. Each device from the malicious traffic is added including corresponding edges. Appropriate dictionaries and lists of IP addresses and ports are generated. These dictionaries are kept accordingly, serving as lookups for later modules. For instance, when adding devices to the graph, if the node and the connection to the same destination already exist, the dictionary used to update the volume of traffic for each connection is updated. This is accomplished to facilitate weighted edges that can detect stages like attack stages, which are determined by a large volume of traffic directed to the same destination. Additionally, a list of each category and the initial timestamps for each candidate category from LSA clustering module are generated to aid in the detection of botnet stages. Other dictionary data structures used include dictionaries to capture useful insights for prediction, including a dictionary of destination port: list of IP address pairs to aid in determining horizontal port scan. This provides additional information to be added to appropriate graph-based scores, such as centrality, In-Degree (ID), Out-Degree (OD).

The procedure for computing the relevant Graph-Based analysis scores is described in lines 9 to 14. This includes the computation of the Betweenness centrality (BC) in line 13, OD in line 14, ID in line 15, and the creation of a dictionary that keeps track of the feature name for each node and a corresponding index for lookup later. All of these graph-based measures are computed and saved in lists with the same length as the number of nodes in a generated graph. Following the computation of the scores, the mean

Algorithm 2 Implementation Pseudocode for Graph-Based Correlation Analysis of Botnet Traffic

Require: *Mal_Traffic* ▷ Dataset
Require: *Networkx* ▷ for Graph generation
Require: *pyvis.netowrk* ▷ for visualisation
Require: *xml.etree.ElementTree* ▷ For parsing of xml
1: **for** *category* in *traffic* **do**
2: **for** *device* in *device_list* **do**
3: **if** *device NOT_IN Traffic_Graph* **then**
4: *Traffic_Graph.Apend*(*device*)
5: *updateDictionaries*()
6: **end if**
7: **end for**
8: **end for**
9: **for** *Node* in *Traffic_Graph* **do**
10: $BC \leftarrow \sum \frac{\sigma_{v_j v_k}(v_i)}{\sigma_{v_j v_k}}$
11: $OD \leftarrow \sum_{v_j \in V, v_i \neq v_j} \mathcal{F}(e_{i,j})$
12: $ID \leftarrow \sum_{v_j \in V, v_i \neq v_j} \mathcal{F}(e_{j,i})$
13: $Data\_Dict \leftarrow feature\_name : feature\_index$
14: **end for**
15: $\mu \leftarrow \sum \frac{X_i}{N}$ ▷ mean - Centrality scores
16:
17: $\sigma \leftarrow \sqrt{\frac{(x_i-\mu)^2}{N}}$ ▷ standard deviation - Centrality scores
18:
19: **for** *Node* in *Traffic_Graph* **do**
20: **if** $DC[i] > (\mu + \sigma)$ **then**
21: $Central\_Node \leftarrow Node[i]$
22: **end if**
23: **if** $(ID[i] = 1)$ *AND* $(OD[i] = 0)$ **then**
24: $Scanned\_Node \leftarrow Node[i]$
25: **end if**
26: **if** $(ID[i] = 1)$ *AND* $(OD[i] = 1)$ **then**
27: **if** (*Central_Node*) *CONNECT* $(OD[i])$ **then**
28: $C\&C \leftarrow Node[i]$
29: **end if**
30: **end if**
31: **end for**
32: $Output.append([Device\_ID, BC, ID, OD])$
33: **Return** *Output*

and standard deviation are computed as demonstrated in lines 15 and 17, respectively.

Once the appropriate scores have been computed, the next step is to detect different activities, such as infection, scanning, and C&C communication, by utilising the computed measures. Firstly, to determine the infected device, the dictionary with centrality scores is utilised. The infected device is expected to be a central node in the graph as it exerts the most influence in the network. When considering the typical goals of a botnet attack and the general steps followed, one of the primary objectives would be to grow the number of infected devices (bots) to a sufficient number, following which a botmaster can orchestrate a plan to use these devices to launch a DDoS attack in a coordinated fashion.

To achieve this goal, the infected device seeks to find more devices and infect them to join the botnet, which is done primarily through scanning, a crucial stage that ensures the success of a botnet attack. Consequently, in a particular network of devices involved in botnet activities, as already stated, the infected device becomes the central node. When examining different types of connections in the graph, the degree of centrality and betweenness centrality are the two measures that can help identify the infected device(s) from the graph. The answer therefore lies in the list of centrality scores for each of the devices. From the list of centrality scores for each device examined, the infected device is expected to have significantly higher scores compared to the devices being scanned, while the devices being scanned are expected to be as many as possible sharing the same port number to maximise the impact of the attack, resulting in only a few nodes in the graph having significantly large centrality scores.

Therefore, the challenge in detecting the central node using the centrality scores becomes how to select the central node. Several approaches can be considered. For instance, a threshold-based approach can be employed to determine the cutoff between significantly high scores and lower scores, essentially selecting infected device from other devices. This approach would be particularly beneficial if domain-specific knowledge is known and may not work effectively if the number of scanned devices changes. For this work, the approach employed is the one that selects the cutoff based on the central tendencies of the centrality scores without setting a particular score value.

Overall, the detection of different categories is shown by lines 19 to 31. The cutoff is determined in lines 20 to 22 by checking if each score is greater than $(\mu + \sigma)$, and only the device with greater scores is returned, translating to infected devices. Lines 23 to 30 show how the other types of attacks are determined, and finally, lines 32 and 33 compile the output and return.

V. EVALUATION OF RESULTS

The proposed methodology employs various Python libraries such as Pandas, NumPy, and the Scikit-learn module for experimentation. For malicious traffic clustering, TruncatedSVD for LSA, TfidfVectorizer, CountVectorizer, and KMeans have been used. NetworkX and Pyvis were used for Graph-Based analysis. The experimentation was carried out using the University of Bradford High-Performance Computing platform, which consists of nodes equipped with 2x Intel Xeon Gold 6138 20C 2.0GHz processors, a Standard RAM with a capacity of 192 GB per node, and a high memory node with 384 GB.

When evaluating IDSs, it is customary to employ a combination of typical measures such as TPR, FPR, and Overall Success Rate (OSR), which is also referred to as Accuracy, Precision, and F-Score [39]. These measures are frequently used together to ensure a comprehensive evaluation of Intrusion Detection Systems (IDSs), addressing

TABLE 4. CTU-IoT-Malware-Capture-43-1 (Mirai) labels distribution.

Label	Flows
Benign	20574934
C & C	3498
C & C - File Download	14
DDoS	65803
File Download	1
Okiru	8765885
PartOfHorizontalPortScan	37911674

the limitations of network traffic. For instance, a metric such as accuracy might yield a high score if the traffic is highly imbalanced, but this could be misleading if it is the only metric reported. As such to evaluate the proposed approach the same metrics are used together. Equations (7) to (11), are presented as described in [39], can be used to calculate TPR, FPR, and F-Score.

TPR is the ratio of correctly predicted malicious traffic to all malicious traffic.

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

FPR on the other hand is the proportion of benign traffic classified malicious among all the normal traffic.

$$FPR = \frac{FP}{TN + FP} \tag{8}$$

OSR is the proportion of the traffic predicted correctly among all the traffic.

$$OSR = \frac{TP + TN}{TP + FP + TN + FN} \tag{9}$$

Precision is the ratio of correctly predicted malicious traffic to all traffic predicted malicious.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Finally, F-Score is a composite metric that takes into account both TPR and Precision.

$$F - Score = \frac{2 \cdot Precision \cdot TPR}{Precision + TPR} \tag{11}$$

The rest of this section provides an evaluation of the proposed system, first Section V-A discusses the dataset followed by the experimental set-up covered in Section V-B.

A. DATASET DESCRIPTION

The IoT23 dataset is a valuable resource derived from IoT traffic [40], and its label distribution is provided in Table 4. This dataset represents labelled connection log files generated by Zeek [41]. It consists of 20 features, including ts, uid, id_orig_p, id_resp_h, id_orig_h, id_resp_p, proto, service, duration, conn_state, local_orig, local_resp, orig_bytes, resp_bytes, missed_bytes, history, orig_pkts, orig_ip_bytes, resp_pkts, and resp_ip_bytes. Table 4 presents a comprehensive list of categories and respective summary statistics of flows for each category included in the dataset.

In the case of ML applications, feature selection is often employed as a pre-processing step before training appropriate models for detection.

B. EXPERIMENTAL SETUP AND RESULTS

The network traffic is preprocessed and transformed into a format compatible with clustering and graph generation tools. This includes converting the traffic into a rectangular matrix and applying appropriate scoring that captures the importance of each of the features. Additionally, it is well known that when training ML models, features such as the IP address and port number values do not carry informative values when it comes to differentiating between different classes. In such cases, these values are typically converted to categorical features, often the values are the meaning of the features. For example, internal_IP or external_IP, in essence, encodes the insight or meaningful values as a result of generalisation [42]. In this current work, the same mechanism is followed for such features. Additionally, as part of preprocessing, the feature selection approach is applied to the network traffic to filter out any features that are not used in terms for differentiating between categories in unsupervised clustering to uncover botnet attack candidate stages [38].

Fig. 10 illustrates the outcome of alert correlation utilising graph theory. In the figure, the node with the highest degree of centrality represents the infected device. The infection status is further validated by candidate botnet categories that link to the infected device.

Furthermore, the results of the clustering provide further insights on the type of activities that took place, this includes attack patterns uncovered from the traffic data indicating the severity of the attack by highlighting the volume of the attack by size. A node with out-degree of two linking to the central node (device) suggests communication between C&C server and infected device, thus providing more evidence that the device has been infected.

The confusion matrices for Botnet stage prediction, specifically Scanning, C&C Communication, and DDoS attack stages, are presented in Fig. 11. Subsequently, typical evaluation performance metrics for IDSs are computed based on these matrices, and the results are summarised in Table 5. The evaluation demonstrates that the scanning and DDoS predictions exhibit superior performance, with scanning activities achieving an accuracy rate of over 99.99% and a TPR of the same value. Similarly, the prediction of the DDoS attack stage demonstrates a perfect accuracy and TPR of 100%. These outcomes are not unexpected, as DDoS traffic typically has comparable patterns directed at the same target.

In comparison to the detection of scanning and DDoS botnet traffic, C&C prediction exhibits a lower level of performance. This is because C&C communication patterns are typically diverse, thus detecting the patterns for this stage becomes more challenging resulting in lower accuracy, TPR, and F-Score measures. Unsurprisingly, this is the case, given

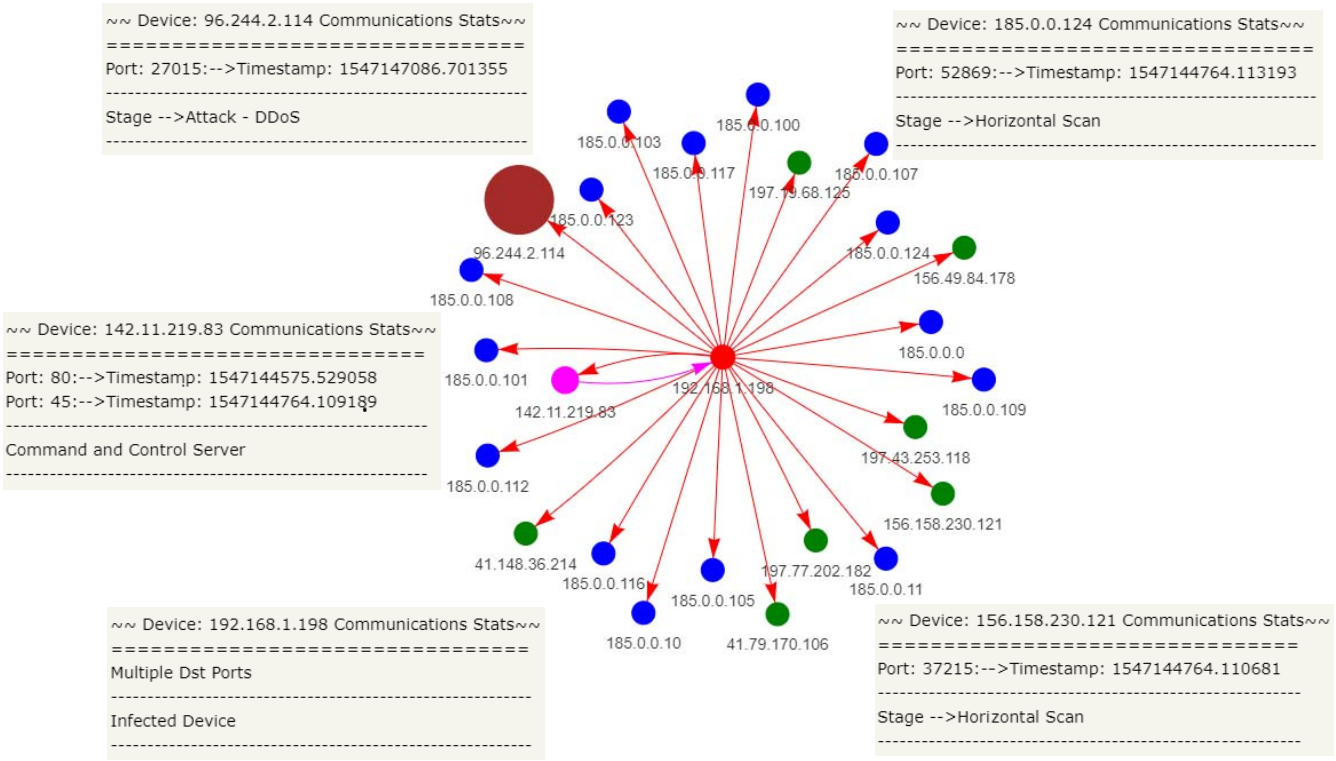

FIGURE 10. Sample of Alert Correlation Graph for Malicious Traffic.

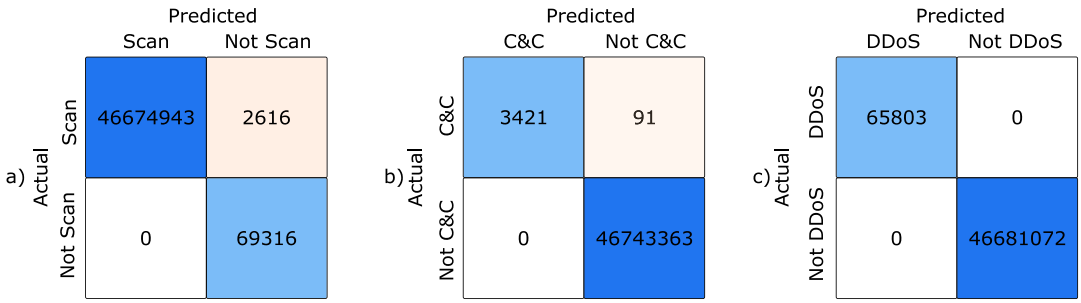

FIGURE 11. Confusion Matrix for Botnet Stages Detected.

TABLE 5. Evaluation of the detection of IoT botnet stages.

Attack Stage	Accuracy	Precision	TPR	FPR	F-Score
Scanning	99.9944%	100%	99.9944%	0%	99.9972%
C&C	99.9998%	100%	97.4089%	0%	98.6874%
DDoS	100%	100%	100%	0%	100%

that communication patterns from infected devices joining a C&C channel differ from those of C&C servers instructing infected devices to perform scanning or DDoS attacks on specific targets.

The obtained FPR of 0% in the present study shown in Table 5 suggests that the utilisation of the auto-threshold tuning model during LSA clustering played an instrumental role in eliminating candidate clusters and cluster features with low strength scores. This was achieved by eliminating noise in the clusters, which resulted in an increased TPR that, in some cases, reached 100%. This was accomplished by eliminating low-strength candidate clusters and cluster features captured by the diagonal matrix entries (Σ) and V^T. The results indicate that the auto-threshold tuning model is an effective approach for improving the efficacy of LSA clustering and reducing the FPR.

In the realm of detecting malicious activities from Botnet traffic, works utilising IoT23 dataset have been compared and evaluated. Table 6 presents a comparison of these works, highlighting the proposed methods under the methodology column followed by performance scores. Evaluation metrics were used to compare the results of each method. The

TABLE 6. A performance comparison of existing systems, utilising IoT23 dataset, for IoT botnet detection.

Paper	Methodology	Accuracy	Precision	TPR	F-Score
[42]	ML FIM Correlation	99%	X	X	99%
[43]	Ensemble ML	99.88%	X	99.7%	X
[44]	ML Multiclass Prediction	99.9%	X	X	X
Proposed	LSA, Alert Correlation	99.9981%	100%	99.1344%	99.56%

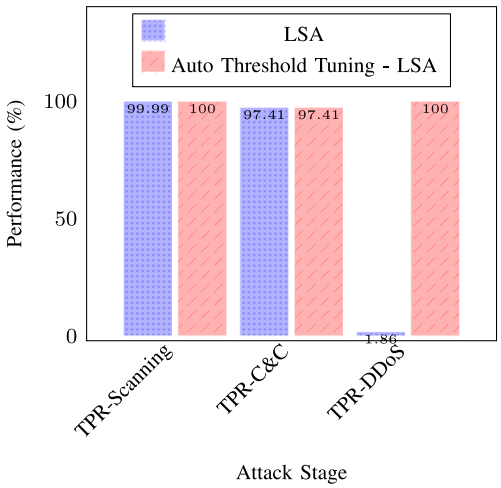

FIGURE 12. A comparison showing the effect of auto threshold turning for LSA.

TABLE 7. Detected stages of botnet attack by timestamp.

Stage	Timestamp
Infection	1547144575.32292
C&C Communication	1547144575.52906
Scanning	1547144764.11068
DDoS	1547147086.70136

findings reveal that the proposed approach consistently outperforms recent works on detection approaches for Botnet traffic. The effect of LSA auto-tuning of the threshold, which reveals the significant benefit of boosting the detection of the DDoS attack that would otherwise be missed, is illustrated in Fig. 12. Additionally, Table 7 presents the detected botnet stages ordered by their timestamp, providing a means of validating the detected stages. This information is useful in determining the temporal order of botnet stages, which is essential in accurately identifying and countering botnet attacks.

IoT devices, despite their high degree of heterogeneity, are commonly interconnected using wireless and wired network technologies within IoT networks. This communication involves a standard level of network traffic, requiring identifying devices through IP addresses and targeting vulnerable services via port numbers. Standard TCP and UDP protocols are commonly used for communication, although IoT-specific protocols such as MQTT may also be utilised, however, they still rely on TCP/UDP for transport layer communication. The unsupervised nature of alert correlation enables the effective detection of coordinated botnet stages previously unseen.

VI. CONCLUSION

This paper proposes a novel approach for detecting IoT botnets that consists of two phases: Phase 1 and Phase 2. The first phase utilises ML algorithms to differentiate benign traffic from IoT traffic, isolating malicious traffic that triggers alerts to the network security team for further investigation. Once detected, the malicious traffic is forwarded to the second phase for further analysis. In Phase 2, the approach employs LSA, an unsupervised learning technique, to cluster malicious traffic into candidate botnet stages. These stages are then correlated using graph-based theory to identify botnet stages within the network. By utilising LSA, the approach reduces computational costs and prioritises the most significant malicious activities, which results in faster alert addressing. For stages that generate insufficient traffic to form patterns detected by LSA, the approach employs a rule-based detection technique, such as the initial infection stage. Timestamps are computed for each detected category and used to organise botnet stages based on their order of occurrence. This information provides crucial insights that help the network security team understand the botnet attack and develop effective mitigation steps to recover from the attack, as well as potential future ones. The proposed approach consistently and accurately detects botnet campaigns by correlating different stages of the same botnet attack with a TPR and F1 Scores of up to 100% for some of the stages, and low FPR of 0%. The proposed approach has been evaluated on the IoT23 dataset, based on static data. Therefore, in the future, it is worth trying to experiment with different datasets to evaluate further the effectiveness of the approach in different contexts. Moreover, future research could be directed towards applying this approach to real-time, dynamically changing datasets to further understand its capability, and thereby make adjustments to the approach if required.

REFERENCES

[1] (Spamhaus Project, Andorra la Vella, Andorra). *Botnet Threat Update, Q4 2023*. Jan. 2024. Accessed: Mar. 1, 2024. [Online]. Available: https://www.spamhaus.org/resource-hub/botnet-c-c/botnet-threat-update-q4-2023/

[2] (Nokia Threat Intelligence Report, Espoo, Finland). *Nokia Threat Intelligence Report Finds Malicious IoT Botnet Activity has Sharply Increased*. Jun. 2023. Accessed: Mar. 1, 2024. [Online]. Available: https://www.nokia.com/about-us/news/releases/2023/06/07/nokia-threat-intelligence-report-finds-malicious-iot-botnet-activity-has-sharply-increased/

[3] (Spamhaus Project, Andorra la Vella, Andorra). *Botnet Threat Update, Q4 2022*. Jan. 2022. Accessed: Mar. 1, 2024. [Online]. Available: https://www.spamhaus.org/resource-hub/botnet-c-c/botnet-threat-update-q4-2022/

[4] T. N. Nguyen, Q.-D. Ngo, H.-T. Nguyen, and G. L. Nguyen, "An advanced computing approach for IoT-Botnet detection in Industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8298–8306, Nov. 2022.

[5] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, Aug. 2020.

[6] S. I. Popoola, B. Adebisi, R. Ande, M. Hammoudeh, K. Anoh, and A. A. Atayero, "SMOTE-DRNN: A deep learning algorithm for botnet detection in the Internet-of-Things networks," *Sensors*, vol. 21, no. 9, p. 2985, 2021.

[7] A. Arshad et al., "A novel ensemble method for enhancing Internet of Things device security against botnet attacks," *Decis. Anal. J.*, vol. 8, Sep. 2023, Art. no. 100307.

[8] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[9] T. Trajanovski and N. Zhang, "An automated and comprehensive framework for IoT botnet detection and analysis (IoT-BDA)," *IEEE Access*, vol. 9, pp. 124360–124383, 2021.

[10] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: A malicious Bot-IoT traffic detection method in IoT network using machine-learning techniques," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3242–3254, Mar. 2021.

[11] A. Alharbi and K. Alsubhi, "Botnet detection approach using graph-based machine learning," *IEEE Access*, vol. 9, pp. 99166–99180, 2021.

[12] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "Botchase: Graph-based bot detection using machine learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 15–29, Mar. 2020.

[13] I. Ghafir et al., "Botdet: A system for real time botnet command and control traffic detection," *IEEE Access*, vol. 6, pp. 38947–38958, 2018.

[14] J. Yang, Q. Zhang, X. Jiang, S. Chen, and F. Yang, "POIROT: Causal correlation aided semantic analysis for advanced persistent threat detection," *IEEE Trans. Depend. Secure Comput.*, vol. 19, no. 5, pp. 3546–3563, Oct. 2022.

[15] M. Lefoane, I. Ghafir, S. Kabir, and I.-U. Awan, "Latent Dirichlet allocation for the detection of multi-stage attacks," in *Proc. 24th Int. Arab Conf. Inf. Technol. (ACIT)*, 2023, pp. 1–7.

[16] M. Husák, J. Kašpar, E. Bou-Harb, and P. Čeleda, "On the sequential pattern and rule mining in the analysis of cyber security alerts," in *Proc. 12th Int. Conf. Availabil., Rel. Secur.*, 2017, pp. 1–10.

[17] M. Lefoane, I. Ghafir, S. Kabir, and I. U. Awan, "Sequential pattern mining: A proposed approach for intrusion detection systems," in *Proc. 7th Int. Conf. Future Netw. Distrib. Syst.*, 2024, pp. 599–604.

[18] M. Lefoane, I. Ghafir, S. Kabir, and I.-U. Awan, "Machine learning for botnet detection: An optimized feature selection approach," in *Proc. 5th Int. Conf. Future Netw. Distrib. Syst.*, 2021, pp. 195–200.

[19] H. Wang, H. He, W. Zhang, W. Liu, P. Liu, and A. Javadpour, "Using honeypots to model botnet attacks on the Internet of Medical Things," *Comput. Elect. Eng.*, vol. 102, Sep. 2022, Art. no. 108212.

[20] I. Ghafir et al., "Hidden Markov models and alert correlations for the prediction of advanced persistent threats," *IEEE Access*, vol. 7, pp. 99508–99520, 2019.

[21] S. Haas and M. Fischer, "On the alert correlation process for the detection of multi-step attacks and a graph-based realization," *SIGAPP Appl. Comput. Rev.*, vol. 19, no. 1, pp. 5–19, Apr. 2019.

[22] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.

[23] E. Kidmose, M. Stevanovic, and J. M. Pedersen, "Correlating intrusion detection alerts on bot malware infections using neural network," in *Proc. Int. Conf. Cyber Secur. Prot. Digit. Services (Cyber Security)*, 2016, pp. 1–8.

[24] B. M. Rahal, A. Santos, and M. Nogueira, "A distributed architecture for DDoS prediction and bot detection," *IEEE Access*, vol. 8, pp. 159756–159772, 2020.

[25] J. Maestre Vidal, A. L. Sandoval Orozco, and L. J. García Villalba, "Alert correlation framework for malware detection by anomaly-based packet payload analysis," *J. Netw. Comput. Appl.*, vol. 97, pp. 11–22, Nov. 2017.

[26] M. Khosravi and B. T. Ladani, "Alerts correlation and causal analysis for apt based cyber attack detection," *IEEE Access*, vol. 8, pp. 162642–162656, 2020.

[27] N. T. U. Nhi, T. M. Le, and T. T. Van, "A model of semantic-based image retrieval using c-tree and neighbor graph," *Int. J. Semant. Web Inf. Syst.*, vol. 18, no. 1, pp. 1–23, Feb. 2022.

[28] A. Almomani et al., "Phishing Website detection with semantic features based on machine learning classifiers: A comparative study," *Int. J. Semant. Web Inf. Syst.*, vol. 18, no. 1, pp. 1–24, Feb. 2022.

[29] J. Chu, X. Zhao, D. Song, W. Li, S. Zhang, X. Li, and A.-A. Liu, "Improved semantic representation learning by multiple clustering for image-based 3D model retrieval," *Int. J. Semant. Web Inf. Syst.*, vol. 18, no. 1, pp. 1–20, Aug. 2022.

[30] A. H. Omopintemi, I. Ghafir, S. Eltanani, S. Kabir, and M. Lefoane, "Machine learning for malware detection in network traffic," in *Proc. 7th Int. Conf. Future Netw. Distrib. Syst.*, 2024, pp. 605–610.

[31] M. Lefoane, I. Ghafir, S. Kabir, and I.-U. Awan, "Multi-stage attack detection: Emerging challenges for wireless networks," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, 2022, pp. 01–05.

[32] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Comput. Netw.*, vol. 57, no. 2, pp. 378–403, 2013.

[33] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman, "Using latent semantic analysis to improve access to textual information," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1988, pp. 281–285.

[34] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Am. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[35] T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, *Handbook of Latent Semantic Analysis*. Oxfordshire, U.K.: Taylor & Francis, 2013. [Online]. Available: https://books.google.co.uk/books?id=JbzCzPvzpmQC

[36] T. A. Letsche, "Toward large-scale information retrieval using latent semantic indexing," *Comput. Sci.*, 1996, to be published.

[37] C. D. Manning, P. Raghavan, and H. Schütze, *Scoring, Term Weighting, and the Vector Space Model*. Cambridge, U.K.: Cambridge Univ. Press, 2008, pp. 100–123.

[38] M. Lefoane, I. Ghafir, S. Kabir, and I.-U. Awan, "Unsupervised learning for feature selection: A proposed solution for botnet detection in 5G networks," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 921–929, Jan. 2023.

[39] Y. Zhang, Q. Yang, S. Lambotharan, K. Kyriakopoulos, I. Ghafir, and B. AsSadhan, "Anomaly-based network intrusion detection using SVM," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2019, pp. 1–6.

[40] S. Garcia, A. Parmisano, and M. J. Erquiaga, Jan. 2020, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Stratosphereips. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

[41] "The-Zeek-Project." Zeek. 2021, Accessed: Mar. 20, 2024. [Online]. Available: https://zeek.org/

[42] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "ADEPT: Detection and identification of correlated attack stages in IoT networks," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6591–6607, Apr. 2021.

[43] H. Chunduri, T. G. Kumar, and P. V. S. Charan, "A multi class classification for detection of IoT botnet malware," in *Computing Science, Communication and Security* (Communications in Computer and Information Science 1416), N. Chaubey, S. Parikh, and K. Amin, Eds., Cham, Switzerland: Springer, 2021, pp. 17–29.

[44] M. Hegde, G. Kepnang, M. Al Mazroei, J. S. Chavis, and L. Watkins, "Identification of botnet activity in IoT network traffic using machine learning," in *Proc. Int. Conf. Intell. Data Sci. Technol. Appl. (IDSTA)*, 2020, pp. 21–27.