applied sciences

MDPI

Article

A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing

Belal Sudqi Khater [1], Ainuddin Wahid Bin Abdul Wahab [1], Mohd Yamani Idna Bin Idris [1,*], Mohammed Abdulla Hussain [2] and Ashraf Ahmed Ibrahim [3]

[1] Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia; Belal.khater@siswa.um.edu.my (B.S.K.); ainuddin@um.edu.my (A.W.B.A.W.)
[2] College of Technological Innovation, Zayed University, P.O. Box 19282, Dubai, UAE; mohammed.hussain@zu.ac.ae
[3] Saal.ai, P.O. Box 112230, Abu Dhabi, UAE; ashraf235@gmail.com
* Correspondence: yamani@um.edu.my; Tel.: +60-379-676-339

Received: 5 October 2018; Accepted: 27 December 2018; Published: 6 January 2019

Abstract: Fog computing is a paradigm that extends cloud computing and services to the edge of the network in order to address the inherent problems of the cloud, such as latency and lack of mobility support and location-awareness. The fog is a decentralized platform capable of operating and processing data locally and can be installed in heterogeneous hardware which makes it ideal for Internet of Things (IoT) applications. Intrusion Detection Systems (IDSs) are an integral part of any security system for fog and IoT networks to ensure the quality of service. Due to the resource limitations of fog and IoT devices, lightweight IDS is highly desirable. In this paper, we present a lightweight IDS based on a vector space representation using a Multilayer Perceptron (MLP) model. We evaluated the presented IDS against the Australian Defense Force Academy Linux Dataset (ADFA-LD) and Australian Defense Force Academy Windows Dataset (ADFA-WD), which are new generation system calls datasets that contain exploits and attacks on various applications. The simulation shows that by using a single hidden layer and a small number of nodes, we are able to achieve a 94% Accuracy, 95% Recall, and 92% F1-Measure in ADFA-LD and 74% Accuracy, 74% Recall, and 74% F1-Measure in ADFA-WD. The performance is evaluated using a Raspberry Pi.

Keywords: fog computing; intrusion detection; IoT security; Multilayer Perceptron

1. Introduction

Fog computing is becoming an efficient and cost-effective distributed computing paradigm to support applications on billions of connected devices forming the Internet of Things (IoT). Fog computing is proposed to help resolve issues in IoT applications that require low latency, geolocation and mobility support in addition to location awareness. The infrastructure of fog computing allows applications to run in an environment close to the edge of the network and to reduce the overhead in the cloud [1]. Fog computing is a platform that, similar to the cloud, provides computing resources, storage, and application services to the end user.

Fog computing, as an emerging new technology, is facing many challenges related to security and privacy since fog devices are deployed in places where protection is minimal. Fog devices are vulnerable to many cyber-attacks such as man-in-middle and port scan attacks which compromise the data privacy. Intrusion detection techniques have been applied to fog computing to effectively reduce the security threat of attacks in the fog infrastructure.

1.1. Fog Computing and IoT

Currently, Cloud computing models are playing an important role in companies and startups since it provides an efficient alternative to owning and managing private computing resources. The emergence of Internet of Things (IoT) as a new technology increases the need for Cloud computing services and enhances the role of Cloud computing in business and technology. IoT devices such as sensors and actuators are becoming more ubiquitous, which brings a number of challenges in terms of performance, scalability, and reliability. Fog computing was introduced as a new computing paradigm to meet the low-latency and geographical distribution requirements of IoT applications [1].

Figure 1 shows how fog computing extends the cloud computing and its services from the core to the edge of the network. The general structure of the fog computing consists of three main components: (i) end devices; (ii) Fog nodes; and (iii) Back-end cloud infrastructure. Fog computing provides computation and storage in addition to network services between the edge and the cloud servers. Fog computing nodes are heterogeneous in nature and are deployed in a diverse set of environments. Therefore, the fog software architecture should be able to facilitate efficient resource management. Fog software architecture consists of the following main components [2] (see Figure 2):

- *Internet of Things (IoT) and Internet of Everything (IoE) verticals*: use cases include smart city, smart grid, smart connected vehicle, healthcare, and medical IoT etc.
- *Orchestration Layer*: it provides dynamic, policy-based life-cycle management and supports data aggregation, decisions, data sharing and migration.
- *Abstraction Layer*: the role of this layer is to expose a uniform and programmable interface for efficient management and hide the platform heterogeneity.

The close proximity of the fog nodes to the end devices helps in resolving the latency problems and provides the option of processing the data coming from the end devices in real time. Only the data and computation which are required by the cloud are passed to the back-end cloud through high-speed communication channels. Contextual and location-awareness information is extracted from the data generated from the end user IoT devices by the fog nodes and can be further analyzed to support many specialized services. In edge computing, the computing process is performed locally using programmable automation controllers which are responsible for data processing, storage, and communication [3,4].

An Intrusion Detection System (IDS) is a crucial component of a security system for any computer network where intrusion and other security breaches need to be detected quickly and effectively. There are two main approaches to intrusion detection, signature-based methods and anomaly-based methods. Signature-based methods match the current behavior of the network against predefined attack patterns. The method is required to store a signature of each attack in a database which means the system will fail to detect unseen attacks. Anomaly-based methods attempt to build a profile for normal behavior and any activities that do not fit this profile are considered an anomaly and hence an intrusion. This approach is effective in detecting previously unseen attacks but they have relatively high false positive rates. Anomaly-based intrusion detection techniques include statistics-based detection, data mining-based detection, and machine learning-based detection. Machine learning-based detection includes support vector machine-based detection, genetic algorithm-based detection, and neural network-based detection.

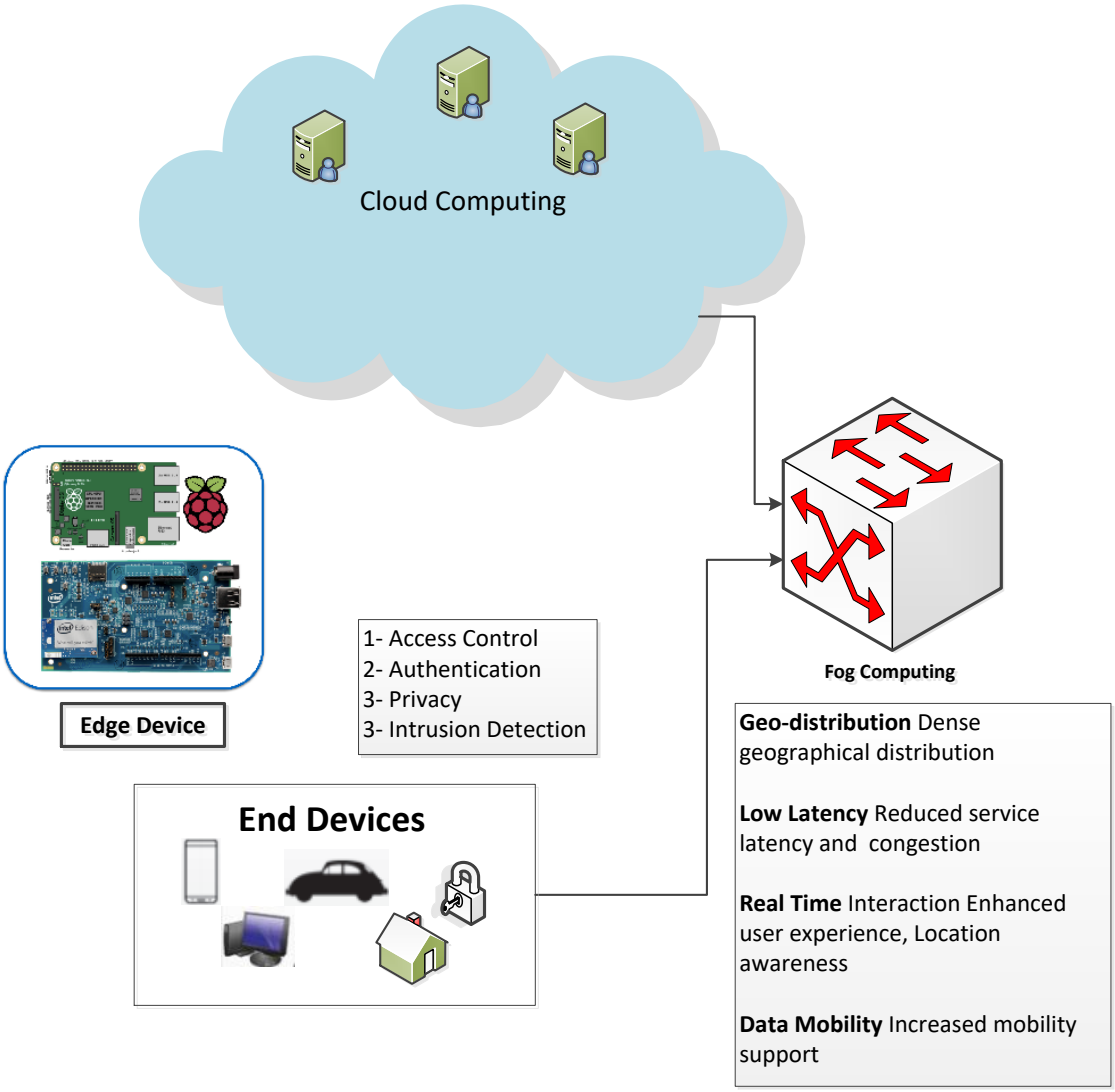


Figure 1. The structure of Internet of Things (IoT) network using fog computing.

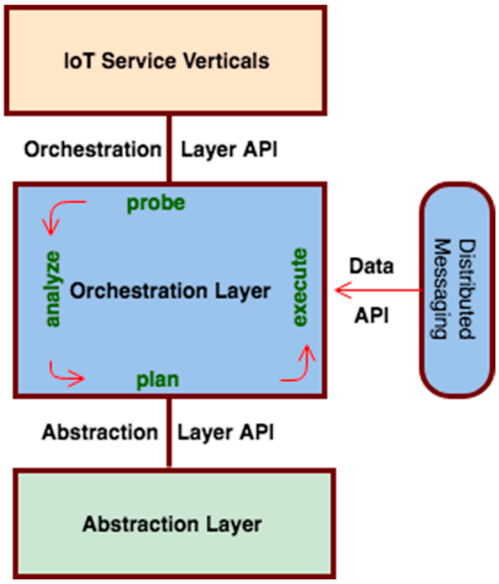


Figure 2. The main components of Fog network [2].

1.2. The Work Contribution

According to Cisco fog computing white paper [1], a fog node can be hosted in a large set of devices, including industrial controllers, switches, routers, embedded servers, and video surveillance cameras. Many of those devices may not have sufficient computational power to perform, for example, data analysis, cryptographic protocols, and intrusion detection algorithms efficiently and provide real time responses and feedbacks. Hence, there is a need for lightweight algorithms and protocols which do not require the use of high-end computing resources. A game-theoretical approach for light-weight IDS was implemented in [5] by combining anomaly and signature detection to achieve high detection and low false positive rates. An et al. [6] proposed a lightweight IDS named Sample Selected Extreme Learning Machine (SS-ELM) to overcome the space limitation of fog nodes. The KDD Cup 99 dataset was used and SS-ELM was shown to outperform the classical backpropagation algorithm in terms of detection accuracy and training time. Hosseinpour et al. [7] introduced a distributed and lightweight IDS based on an Artificial Immune System (AIS) with detection accuracy reaching 98%. One major drawback of the work in lightweight IDS is using old datasets. On the other hand, IDS models that used modern datasets were designed for cloud platforms. In this study, we develop a lightweight IDS using a contemporary dataset. The key contributions of this work are:

- Using a modern dataset, ADFA-LD and ADFA-WD, instead of the out-dated KDD Cup 99 dataset.
- Implementing algorithms with low computational complexity for feature extraction and selection.
- Using a single hidden layer MLP for practical detection time.
- Achieving a good detection rate relative to models used on the cloud.
- Testing the performance on a fog node (Raspberry Pi 3).

The paper is organized as follows: the next section discusses the related work of IDS in fog computing. Section 3 is an overview of the methodology and the technical background of this work. In Section 4, we discuss the requirements of the numerical experiment in terms of software and hardware used. Implementation of the model and evaluation of the results is presented in Section 5. Section 6 is the conclusion and future work is presented in Section 7.

2. Related Work

IDSs for fog computing and IoT networks have been a focus of considerable research work recently. The challenges of effective implementation of detection and prevention systems are challenging due to the heterogeneous, distributed, and decentralized nature of the network infrastructure. The following subsection is an overview of fog computing and its security issues including authentication, access control, intrusion detection, and privacy. More comprehensive discussion of IoT security can be found in [8,9].

2.1. Security Challenges

Fog computing has been introduced as an extension to the cloud and therefore the fog inherits many of the cloud security and privacy challenges [10,11]. The security solutions designed for the cloud could address some of the security issues related to the fog, but these solutions might have some limitations due to the fog's distinct characteristics [12]. On the other hand, fog computing provides a platform to address issues related to IoT privacy and security, since the IoT devices have very limited computational resources, and those devices might not be able to perform the required computational tasks [13]. Hence we can think of fog computing as an additional security tool for IoT networks which helps mitigate some of the IoT security concerns [14,15]. The main security issue facing fog computing can be summarized in the following main categories:

- *Access Control*: Issues in access control can result from a normal user being allowed to access sensitive data or having administrative privileges to change the system configurations. This

issue could occur due to poor management and the distributive nature of the fog makes this task more challenging.

- *Authentication*: Authentication is the task of allowing only authorized users to access the fog system and it is more critical since the fog services are offered to a large number of end users. For instance, using biometric authentication, such as fingerprint, face, and touch-based authentications could prove to be very useful in the fog systems authentication. The resource-constrained IoT devices can delegate the authentication process to a fog device which in turn performs the cryptographic operations needed by the authentication protocol.
- *Privacy*: Users typically are concerned about their data privacy, usage pattern, and location privacy. Since fog nodes are located in the vicinity with the end users there is a significant risk of breaching the privacy using the information collected regarding identity, usage, and location.
- *Intrusion Detection*: Fog systems are also vulnerable to classical network attacks such as man-in-the-middle attack, flooding attack, and port scanning. In fog computing, it is essential to deploy IDS in the nodes to facilitate the intrusion detection on both the client's side and the centralized cloud's side.

2.2. Work in IoT Security

Decision Trees (DT) can be considered as a main classifier or collaborative classifier along with other ML classifiers in security applications, such as intrusion detection [16,17]. DT is proven to be an easy and simple technique, given that only few DTs are involved simultaneously, due to the constraint of large storage requirement by DTs. Moreover, DTs can provide adequate levels of transparency. An example of a DT application can be found from a previous study, proposing the use of a fog-based security system for IoT devices [18]. The research used DT to analyze network traffic to detect suspicious traffic sources and consequently detect intrusions.

Further into the IoT environment, a study [19] managed to develop an Android malware detection system, aiming to secure IoT systems using a linear Support Vector Machine (SVM). The detection performance of SVM was compared to Naive Bayes (NB), Random Forest (RF), and DT. The outcomes of the comparison showed SVMs clearly outperformed the other ML algorithms. These results confirmed the robust nature and application of SVM for malware detection purposes. Such performance is expected from SVMs as it can deal with large number of attribute features using only few sample points. Moreover, further research is needed to understand the performance of SVMs with enriched datasets, as well as datasets which are created with various attack scenarios and environments. Nevertheless, it would be of a great interest to compare the performances of SVM with those of Deep Learning (DL) algorithms, such as convolutional neural network (CNN) algorithms. This is due to the difficulty of understanding and interpreting SVM-based models, since deciding on the optimal kernel is a complicated task. A previous study focused on using SVM to secure a smart grid [20]. This research showed that the ML algorithms SVM, KNN, MLP, ensemble learning and sparse logistic regression are effective in detecting known and unknown attacks, performing better than traditional methods used for attack detection in smart grids. A study [21] suggested a model for the detection of U2R and R2L attacks. The models aimed to reduce the dimensionality of the features to enhance efficiency. The model used two layers of feature reduction and then applied a two-tier classification model that uses NB and KNN classifiers. The results of the model showed good detection results for both attacks.

Meidan et al. [22] studied the implementation of ML algorithms for the purpose of precise IoT device identification, by utilizing network traffic features, which are then embedded into a multi-stage classifier. The classifier managed to categorize the devices that are connected to the network as IoT or non-IoT devices; the ML algorithms identified unauthorized links of IoT devices automatically and accordingly lessened the disruptions that may occur due to threats. In a previous study [23], the abnormal behavior of IoT devices were profiled and the generated dataset from profiling was used to train the classifier to detect abnormal behavior. With the assumption that attackers can utilize

changes for malicious purposes, the author investigated how a partial variation of sensed data can influence the accuracy of the learning algorithm. The author used SVM and K-means clustering as experimental cases for examining the impact of such changes on the detection accuracy of both ML algorithms. The outcomes showed that both algorithms suffered from low detection accuracy. The zero-day attacks are mostly variations of existing attacks; hence, the accuracy of the classifier in detecting variations and changes in the dataset can be a research topic for future investigation.

In [24], the authors suggested the adaptation of a framework to recognize all potential attack paths and alleviate the effects of attacks on the IoT system. The proposed framework contains a graphical security model consisting of five connected stages, starting with data processing, in which the information from the system and the security metrics is fed and processed. The second stage involves the security model generation in which a gap model is generated. This model contains all potential attack paths in the IoT system. The attack path identifies the structure of the nodes which intruders can compromise to gain access to the required node. The third and fourth stages involve the IoT network, including the attack paths, which are visualized (i.e., security visualization) and analyzed (i.e., security analysis), respectively. In [25], the authors proposed a method to detect and restrain malware diffusion in an IoT network. The solution is based on fog computing, which can simultaneously maximize malware detection and minimize the possibility of privacy breach. The proposed malware detection system was constructed using an IDS, and the deployment was accomplished with cloud and fog computing to avoid the restrictions on IDS deployment in smart objects [25]. Furthermore, a framework was proposed to show the possible application of malware diffusion restriction in IoT networks. Borisaniya et al. [26] implemented modified vector space representation approach using ADFA-LD and ADFA-WD datasets, by applying different classifiers to reach detection accuracy of over 95%. Xie et al. [27] used frequency-based models to detect attack behavior in ADFA-LD, achieving a false positive rate below 20%. Xie et al. [28] continued the previous work and implemented a one-class SVM model on short sequences. Overall, they achieved better performance, however, the false positive rate was around 20%. G. Creech et al. [29] have proposed a semantic model for anomaly detection using short sequences of the ADFA-LD Dataset. They have prepared the dictionary of word and phrase from the dataset and evaluated them with the Hidden Markov Model (HMM), Extreme Learning Machine (ELM), and one-class SVM algorithms. They achieved an accuracy of 90% for ELM and 80% for SVM, with 15% false positive rate (FPR) [29,30]. For evaluating ADFA-WD, G. Creech et al. [29] have used HMM, ELM, and SVM. They noted 100% accuracy with 25.1% FPR for HMM, 91.7% accuracy with 0.23% FPR for ELM, and 99.58% accuracy with 1.78% FPR for SVM.

3. Proposed Method

The methodology of this work is divided into three phases, see Figure 3:

- *Phase 1*: this phase is about the data preprocessing which includes the feature extraction and selection.
- *Phase 2*: this is the model selection phase.
- *Phase 3*: it is about the performance evaluation in terms of detection rate and CPU time on a Raspberry Pi.

Phases 1 and 2 are discussed in this section, with an overview of the modified vector space representation, mutual information, and MLP model. Phase 3 is discussed in Section 4.

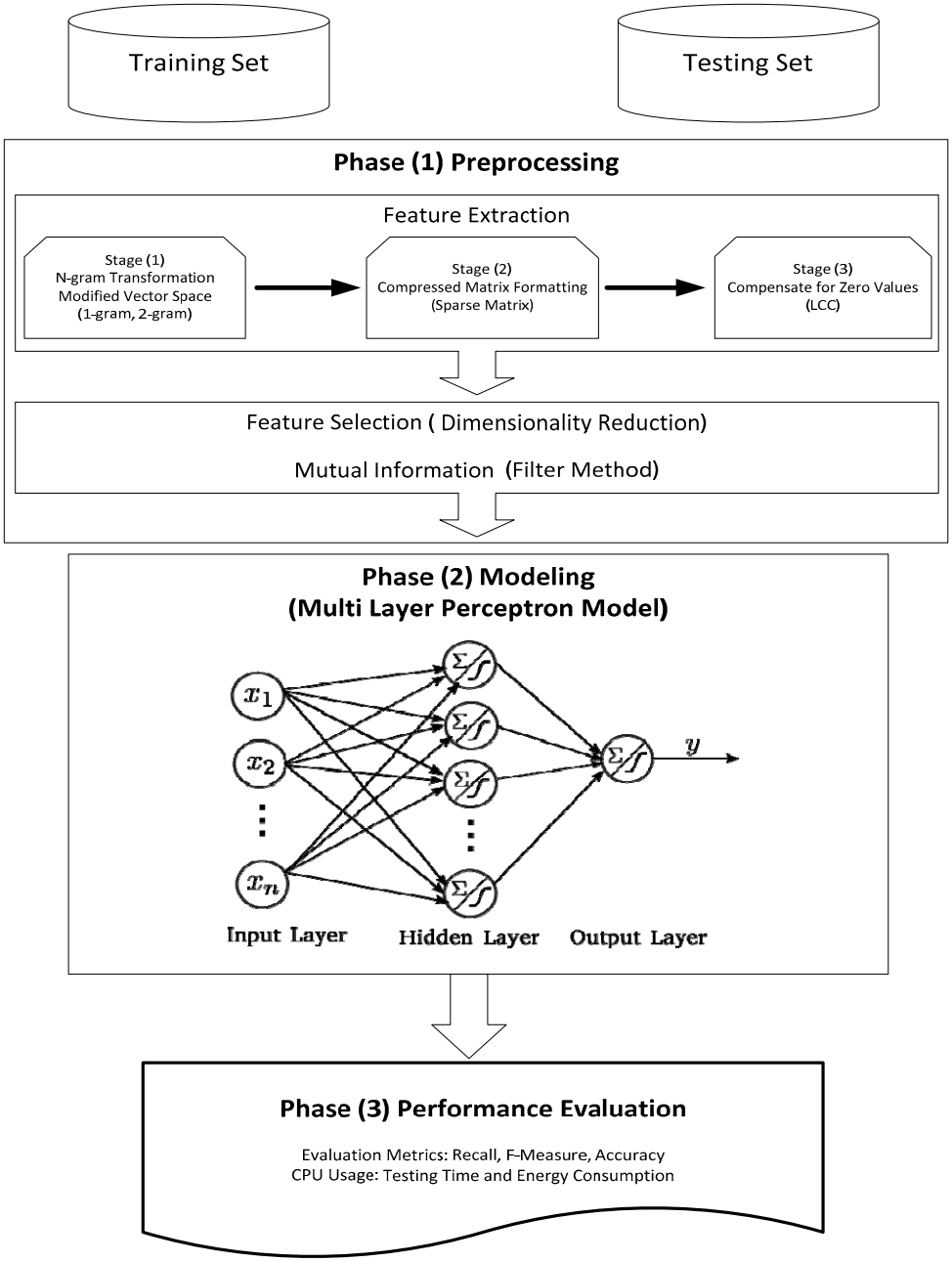


Figure 3. A flowchart of the proposed method.

3.1. Phase 1: Preprocessing

3.1.1. Feature Extraction

The data used in this study consists of traces of system calls. The trace is a relatively short sequence of position integers (system call numbers) which is not an appropriate form of input for a typical machine learning model. In order to transform the sequence format of the input data to a matrix-like form, we need to implement what is known as feature extraction in the preprocessing stage. Feature extraction is the process of representing the input data in a form that is compatible with the modeling algorithm. In this work, we are interested in a feature extraction technique that is simple and can be computed efficiently. One approach is to use Boolean representation where the sequence is represented as a vector of length equal to the size of the set of all symbols (or system calls in this case). Boolean representation is one of the simplest and most commonly used techniques in

text classification [31], where the interest is only on whether or not a system call is present in the trace. The issue with the Boolean method is the fact that it does not maintain the information about the frequency of the system call in the trace.

On the other hand, the vector space technique gives a weight for each system call relative to its frequency. For example, the Term Frequency Inverse Document Frequency (TF-IDF) value is usually used as a weight for the vector space model in the field of information retrieval [31]. The problem with the vector space model is that the relative order of the symbols in the input data is not accounted for. In order to retain the information about the relative order of the terms (or symbols) in the input sequence, we need to consider tuples of terms (or calls) and not only the single term as in the vector space model. The n-gram is defined as tuple $c = (c_1, c_2, \ldots, c_n)$ of n terms where n is generally a small positive integer. This modified vector space model is used frequently in natural language processing and DNA and protein sequence analysis, see for example [32,33]. For a given n-gram, c, its weight $w(c)$ in trace T is given by

$$\omega(c) = \frac{f(c)}{|T|} \tag{1}$$

where $f(c)$ is the frequency of c in T and $|T|$ is the length of the trace.

Example 1:

To illustrate the notions above, let's consider the following trace

$$T = (6, 174, 174, 174, 6, 45, 33, 192, 33, 192, 174, 174, 6, 174). \tag{2}$$

Note that $|T| = 14$. For $n = 2$, we calculate, for every 2-gram, c, its frequency and weight in the following Table 1:

Table 1. The weights and frequencies of all 2-grams.

c	$f(c)$	$w(c)$
(6, 174)	2	0.143
(174, 174)	3	0.214
(174, 6)	2	0.143
(6, 45)	1	0.071
(45, 33)	1	0.071
(33, 192)	2	0.143
(192, 33)	1	0.071
(192, 174)	1	0.071

End Example 1.

The advantage of using this type of features is the fact that calculating a weight of an n-gram is not computationally expensive. On the other hand, the n-gram approach leads to another issue which is the high sparsity of the data matrix. If we consider the dense form of the data matrix, we find it to be too large even for machines with ample resources. One way to overcome this problem is to use compressed matrix formatting and keep only the non-zero values of the matrix. In addition to that, we need to implement algorithms of small runtime which require limited memory space.

We have adopted the aforementioned approach in this study. For the system calls encoding, we have considered only 1-gram and 2-gram as features. The feature vector is constructed by considering only those n-grams that appear in training data. The following step is to put the data in a form of sparse matrix by keeping only the weights of features that appear in the trace and ignore the zero weights. To compensate for any missing n-gram in the test data, we use a linear correlation coefficient (LCC) between the value of all n-gram which is given by:

$$LCC(T) = \frac{A - B}{\sqrt{CD}} \tag{3}$$

where A, B, C, and D are defined as follows:

$$A = N \sum_c \omega(c)\overline{\omega}(c) \tag{4}$$

$$B = \sum_c \omega(c) \sum_c \overline{\omega}(c) \tag{5}$$

$$C = N \sum_c \omega(c)^2 - \left(\sum_c \omega(c)\right)^2 \tag{6}$$

$$C = N \sum_c \overline{\omega}(c)^2 - \left(\sum_c \overline{\omega}(c)\right)^2 \tag{7}$$

and $\overline{\omega}$ (c) is the mean value of n-gram c in the positive training dataset and the sum is over all n-grams in the training dataset.

3.1.2. Feature Selection

The process of feature selection is needed in order to simplify the model, achieve better performance, and reduce the training time. The feature selection (or variable selection) is performed by selecting a subset of the most relevant features. There are three main methodologies of feature selection:

- *Filter method*: It is performed by ranking the features based on their relevance using criteria like correlation, χ^2 or information value.
- *Wrapper method*: It evaluates different subset of features and chooses the one with best performance.
- *Embedded method*: It allows the algorithm to choose the best subset of features and then performs classification.

In this work, we choose the filter method due to its low space and time complexity.

Mutual information is a common feature selection method based on the information value of a given attribute with respect to the classification. Mutual information is defined as follows [31]:

$$MI(V;C) = \sum_{e_v e_c \in \{0,1\}} P(V = e_v\ ,\ C = e_c)\ \log_2 \frac{P(V = e_v\ ,\ C = e_c)}{P(V = e_v)\ P(V = e_v)} \tag{8}$$

where V is a random variable that takes values of $e_v = 1$ if the trace contains n-gram v and $e_v = 0$ if the trace does not contain v. C is a random variable that takes values of $e_c = 1$ if the n-gram belongs to class c and $e_c = 0$ if the trace is not in class c. In the context of information theory, mutual information measures how much information an attribute (an n-gram in this case) contains about the class. The more similarity between the n-gram and the class distributions, the less information is obtained by considering this n-gram.

3.2. Phase 2: Modeling

A perceptron is the building block of Artificial Neural Network (ANN) which is a biologically inspired machine learning model. The perceptron is a mathematical representation of the neuron, the basic computational unit of the brain. In the human nervous system, a neuron receives an input signal from its dendrites and outputs a signal along its axon which is in turn branches out and connects via synapses to dendrites of other neurons. ANN is an information processing paradigm that is based on this biological structure and models complex relationships between inputs and outputs.

A single perceptron mathematical model can be used as a linear classifier: it maps the input $x = (x_1, \ldots, x_n)$, a real-valued vector, to an output value $f(x)$ according to the following rule:

$$\begin{cases} 1, & if \sum_{i=1}^{m} \omega_i x_i + b > 0 \\ 0, & otherwise \end{cases} \tag{9}$$

where $w = (\omega_1, \ldots, \omega_n)$ is a real-valued vector of weights and b is the bias. The perceptron can be generalized by applying a set of nonlinear transformations $\varnothing_i$ to the input vector x. The output y is given by

$$y(x, w) = g\left(\sum_{i=1}^{m} \omega_i \varnothing_i(x)\right) \tag{10}$$

where m is the number of attributes (variables), and g is a nonlinear activation function such as sigmoid or Rectified Linear Unit (relu) functions. The Perceptron is a simple model and useful in approximating many continuous functions. In order to improve the model performance, we consider a multi-layer perceptron by adding at least one hidden layer to the perceptron model above, see phase 2 in Figure 3. The learning algorithm that is commonly used is the back-propagation algorithm [34], a generalization of the least mean squares algorithm in the linear perceptron. The sigmoid and relu function are two activation functions which are frequently used in MLP learning models.

4. Experimental Setup

The experiment setup includes a brief discussion of the dataset of this study in addition to the programming language and libraries used for the software implementation. Furthermore, an overview of the testing platform (Raspberry Pi) hardware is presented.

4.1. ADFA-LD and ADFA-WD Dataset

Collecting and generating system call traces in a large scale for research purposes in not a simple task, and the number of datasets available for IDS evaluation is limited. Recently, Creech et al. [35,36] proposed ADFA-LD and ADFA-WD datasets as a modern alternative to some of the outdated datasets. They showed that ADFA-LD has higher complexity level than some legacy datasets. Xie and Hu performed preliminary analysis of ADFA-LD [37,38] and showed that Euclidean distance is not an effective metric to separate behaviors of some attacks from the normal behavior. Table 2 contains the distribution of the system call traces to normal and attack types in three groups of ADFA-LD and ADFA-WD.

Table 2. Australian Defense Force Academy Linux Dataset (ADFA-LD) and Australian Defense Force Academy Windows Dataset (ADFA-WD) composition based on binary classification of the types of system call traces.

Dataset	No. of Traces ADFA-LD	No. of Traces ADFA-WD	Class of Traces
TRAINING_DATA_MASTER	833	355	Normal
VALIDATION_DATA_MASTER	4372	1827	Normal
ATTACK_DATA_MASTER	746	5542	Attack

Python workbench was used for the evaluation of Modified Vector Space MVS demonstration on ADFA-LD and ADFA-WD datasets. The workbench can host multiple machine learning algorithms which can be utilized on the selected datasets with various term-size. Nine classification algorithms were selected from six different categories. A list of the algorithms selected, the options for individual algorithm and their respective categories are shown in [26].

The datasets were collected and then converted into MVS representation for various term-size. Term-sizes of 1 and 2 were selected for the experiment. For each dataset (i.e., ADFA-LD and ADFA-WD) the experiment was run for binary classification. One of two labels was considered for each trace—normal and attack. Each chosen algorithm was run with selected options on converted

data through 10-fold cross-validation method. Table 2 describes the number of features extracted from ADFA-LD and ADFA-WD dataset for varying term-size using modified vector space representation.

Comparing with previous benchmark datasets, ADFA-LD and ADFA-WD are much more representative of current cyber-attacks. Furthermore, they provide a more convincing and appropriate framework for intrusion detection system (IDS) development and performance evaluation.

ADFA-LD contains thousands of traces of system calls, which are collected under a variety of scenarios to mimic real life circumstances. It consists of a comprehensive collection of system call traces representing recent system level vulnerabilities and attacks. ADFA-LD is generated from a Linux local server with Linux kernel 2.6.38, presenting a more modern and wildly used computer system. The attacks used in generating ADFA-LD include Hydra-FTP, Hydra-SSH, Adduser, Java-Meterpreter, Meter-preter, and Webshell, each of them generates 8 to 20 attack traces. Table 3 shows the number of collected traces for each type of attack. Moreover, technical details about the dataset can be found in [35].

Table 3. Attack vectors used to generate the ADFA-LD attack dataset.

Attack Type	Attack Payload Description	Vector	Traces Count
Adduser	Add new superuser using poisoned executables	Client side poisoned executable	91
Hydra-FTP	Bruteforce password guess on FTP port	FTP by Hydra	162
Hydra-SSH	Bruteforce password guess on SSH port	SSH by Hydra	176
Java-Meterpreter	Java based Meterpreter exploit	TikiWiki vulnerability exploit	124
Meterpreter	Linux Meterpreter exploit	Client side poisoned executable	75
Webshell	Privilege escalation using C100 Webshell	PHP remote file inclusion vulnerability	118

ADFA-WD (Windows Dataset) shows the high-quality collection of DLL access requests and hacking attacks for a variety of system calls [30,36]. It generally contains identified Windows-based vulnerability. Dataset was collected in Windows XP SP2 host with the help of Procmon program. Default firewall was used and Norton AV 2013 was enabled to ignore the low level attacks and filter only sophisticated attacks. The OS environment enabled file sharing and configured network printer. It was running different applications such as, webserver, FTP server, database server, etc. A total of 12 class labels, V1 to V12, of known vulnerabilities for installed applications were exploited with the help of Metasploit framework and other custom methods. Table 4 presents the details of each attack class in ADFA-WD dataset [26,36].

Table 4. Vulnerabilities considered to generate the ADFA-WD attack dataset.

ID	Vulnerability	Program	Exploit Mechanism	Trace Count
V1	CVE: 2006-2961	CesarFTP 0.99g	Reverse Ordinal Payload Injection	454
V2	EDB-ID: 18367	XAMPP Lite v1.7.3	Upload and execute malicious payload using Xampp_webdav	470
V3	CVE: 2004-1561	Icecast v2.0	Metasploit exploit	382
V4	CVE: 2009-3843	Tomcat v6.0.20	Metasploit exploit	418
V5	CVE: 2008-4250	OS SMB	Metasploit exploit	355
V6	CVE: 2010-2729	OS Print Spool	Metasploit exploit	454
V7	CVE: 2011-4453	PMWiki v2.2.30	Metasploit exploit	430
V8	CVE: 2012-0003	Wireless Karma	DNS Spoofing using Pineapple Router	487
V9	CVE: 2010-2883	Adobe Reader 9.3.0	Reverse Shell spawn through malicious PDF	440
V10	—	Backdoor executable	Reverse Inline Shell spawned	536
V11	CVE: 2010-0806	IE v6.0.2900.2180	Metasploit exploit	495
V12	—	Infectious Media	Blind Shell spawned	621

4.2. Software and Platform

The experiments are implemented using Python 2.7, Numpy, and Scipy for reprocessing, ScikitLearn library for machine learning and Matplotlib library for visualization. The experiment is tested on Raspberry Pi 3, which is a small low-powered single-board computer with the size of a credit card, manufactured by Raspberry Pi Foundation in the UK. Raspberry Pi has been very popular among computer enthusiasts and has been used in countless number of IoT and robotics projects. Raspberry Pi comes with the low-power and powerful BCM2835 CPU, in addition to 40 General Purpose Input Output (GPIO) pin for connecting sensors, which makes it an ideal device for IoT application. In many IoT applications, Raspberry Pi has been deployed as a fog node, see for example [39–41]. In this experiment we used Raspberry Pi 3 which is the third generation Raspberry Pi with following hardware specifications:

- A 1.2 GHz 64-bit quad-core ARMv8 CPU.
- 1 GB of RAM.
- SD card, 16GB, class 10.
- 4 USB ports.
- 40 GPIO pins.
- Full HDMI port and Ethernet port.
- Video Core IV 3D graphics core.

The Energy consumption of Raspberry Pi can be calculated using certain specifications of the Raspberry Pi should be known, namely, the Voltage (V), the Electrical Current (I), and the CPU time for the run of the experiment. The Voltage is given as a standard value for the Raspberry Pi model 3 B. As for the electrical current, it depends on what peripherals are connected, typicall ranging between 700 to 1000 mA for the 3 B model, with a constraint of 1.2 Amp as maximum current. The experiment's configuration includes HDMI port and a keyboard peripherals. The product of Current and the Voltage yields to the Power consumption, which is then multiplied by the CPU time to calculate the energy consumption [42].

Figure 4 shows the locations of the major hardware components in Raspberry Pi 3 model B. The Raspberry Pi is running on Raspbian, a Debian Linux-based operating system optimized for the Raspberry Pi hardware.

4.3. Evaluation Metrics

In order to evaluate our model, standard evaluation metrics are considered. Those metrics are commonly used in different applications of classification models. We start by the defining the following:

- *True Positive (TP)*: Number of positive (attack) traces detected as positive (attack) traces.
- *True Negative (TN)*: Number of negative (normal) traces detected as negative (normal) traces.
- *False Positive (FP)*: Number of positive (attack) traces detected as negative (normal) traces.
- *False Negative (FN)*: Number of negative (normal) traces detected as positive (attack) traces.

Those definitions are used to derive relevant measures such as precision and recall. Precision is defined as the positive predictive value, recall as true positive rate, and F1-measure as the harmonic mean of precision and recall. Accuracy, on the other hand, is the ratio of correct classification to the total number of examples in the test dataset. Mathematically, the above metrics are defined in [26].

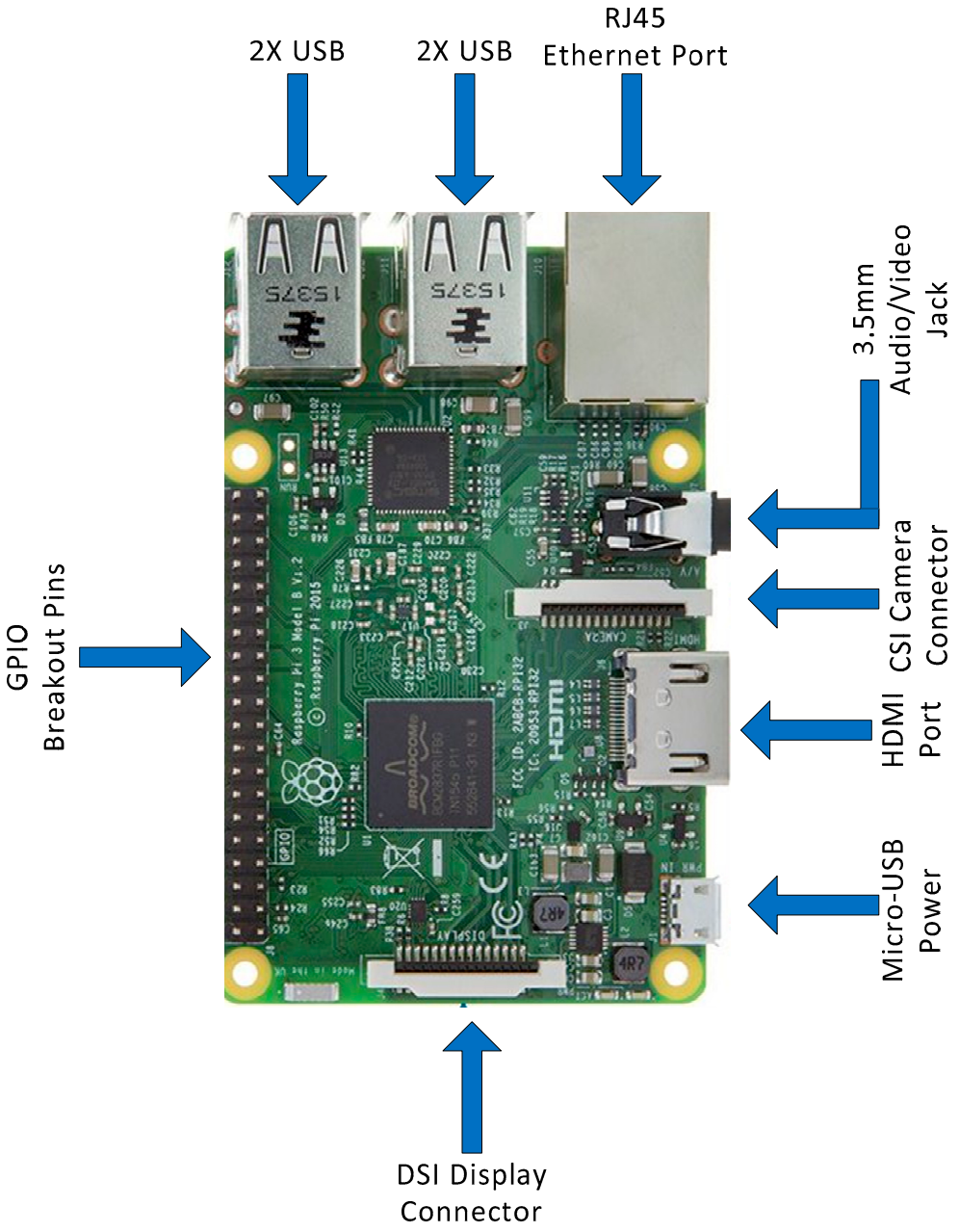


Figure 4. The hardware components of Raspberry Pi 3 model B.

5. Results

The first step in the experiment is to compute 1-gram and 2-gram for the system call traces using sparse matrix formatting in Python to reduce the memory size used. This method of formatting greatly reduces the memory size of the data matrix since 98.2% of the 2-gram matrix entries are zero values. The compressed form of the matrix is around 4% of memory space of the original dense matrix in ADFA-LD. In addition, online algorithms are used to compute the linear correlation coefficient for each trace in order to reduce the time and space of such computations.

In the following stage, the top 60 and 80 features for 1-gram and top 80 and 120 features for 2-gram are selected based on their information value using mutual information, and are combined for training the model. The numbers 60, 80, and 120 are chosen to reduce the data size and to have good representation of the data. Figure 5 shows the distribution of the top 10 2-gram features based on mutual information. We notice clear disparities in the frequencies in the two classes (Attack and

Normal). The above preprocessing phase was performed on Ubuntu 17.10 runnig on Core i7 with 16 GB RAM.

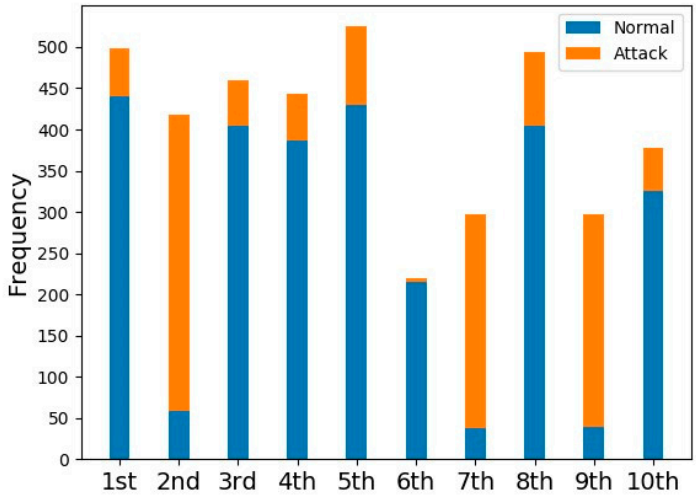
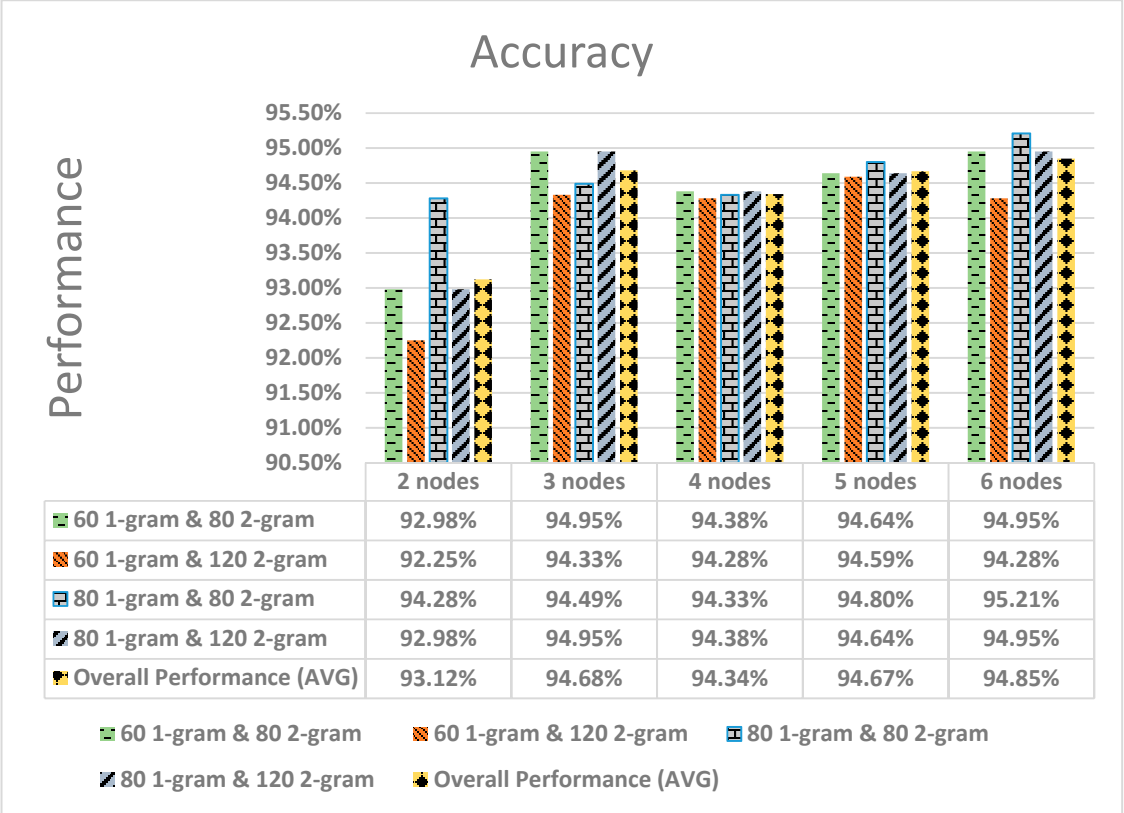


Figure 5. The distribution of top 10 2-grams based on the mutual information in the two classes.

After normalizing the processed data for better performance, a single layer perceptron is implemented for a different number of hidden nodes. The flow graph, Figure 3, shows the steps taken to produce the results in this study. We have used all normal training data and part of the attack data, chosen randomly for training and selected random subset, selected randomly, of the validation data, and the rest of the attack data for testing. Considering higher orders of *n*-grams does not improve the performance significantly.

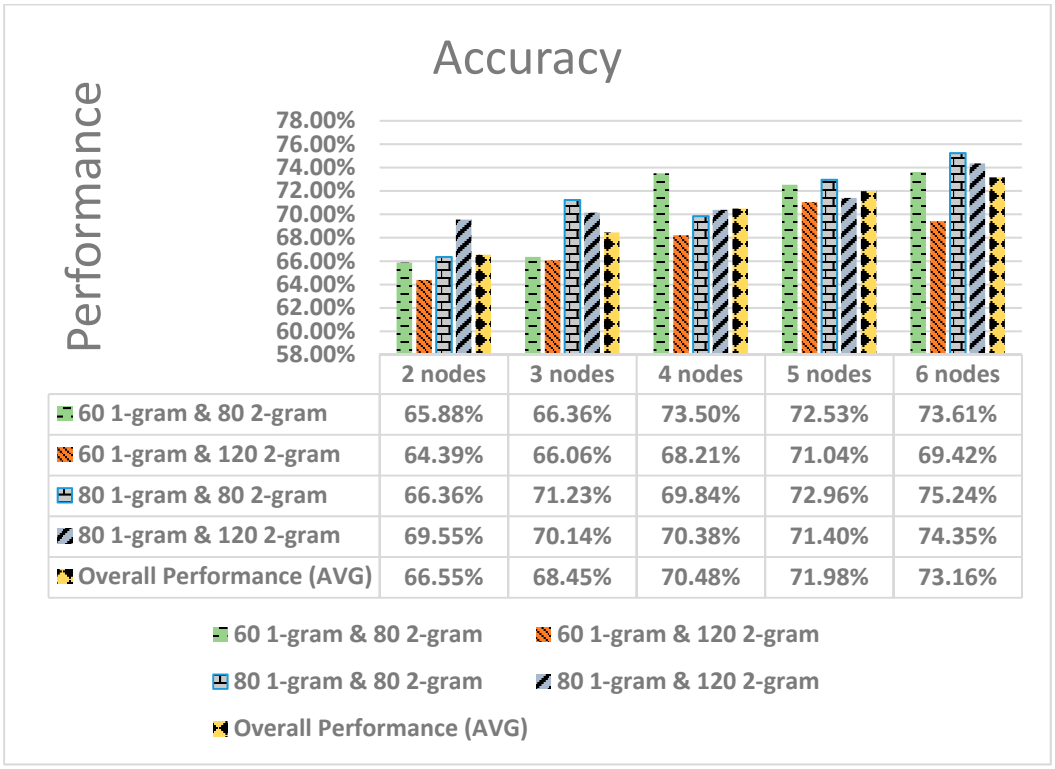
The experiment is performed using combinations of top 60 and 80 1-gram and top 80 and 120 2-gram with different number of nodes in the hidden layer and the results are then evaluated. Figures 6 and 7 show the performance accuracy of the models using four combinations of top features with their respective *n*-grams. We notice that the best accuracy value is achieved with 3 nodes in the hidden layer using 80 1-grams and 120 2-grams in ADFA-LD compared to 6 nodes in ADFA-WD. For the recall values, Figures 8 and 9 show the best value (i.e., 95%) is reached in 3 nodes model using 80 1-gram and 120 2-gram for ADFA-LD, and 74% is reached in 6 nodes for ADFA-WD. Figures 10 and 11 show the F1-measure values for all combinations of 1-gram and 2-gram and the best F1-measure value (95%) is achieved on the 3 nodes model using with 80 1-gram and 120 2-gram for ADFA-LD, while 74% using 6 nodes in ADFA-WD.

The comparison between all the performance (i.e., accuracy, recall, F1-measure) of the models using different number of nodes shows the best overall performance is achieved when we use 3 nodes in the hidden layer if ADFA-LD, and 6 nodes in ADFA-WD. Though 5 and 6 nodes model give the best value of recall rate, the result is similar to the 3 nodes model. Therefore, we can conclude from the above analysis that the model is performing quite well given the small number of nodes in the hidden layer. Using only 3 nodes in the hidden layer, the detection rate is quite adequate for many applications. In all cases, 3 nodes hidden layer model outperforms models that use more nodes but on different combinations of 1-gram and 2-gram. This implies that adding more nodes does not significantly improve the performance.



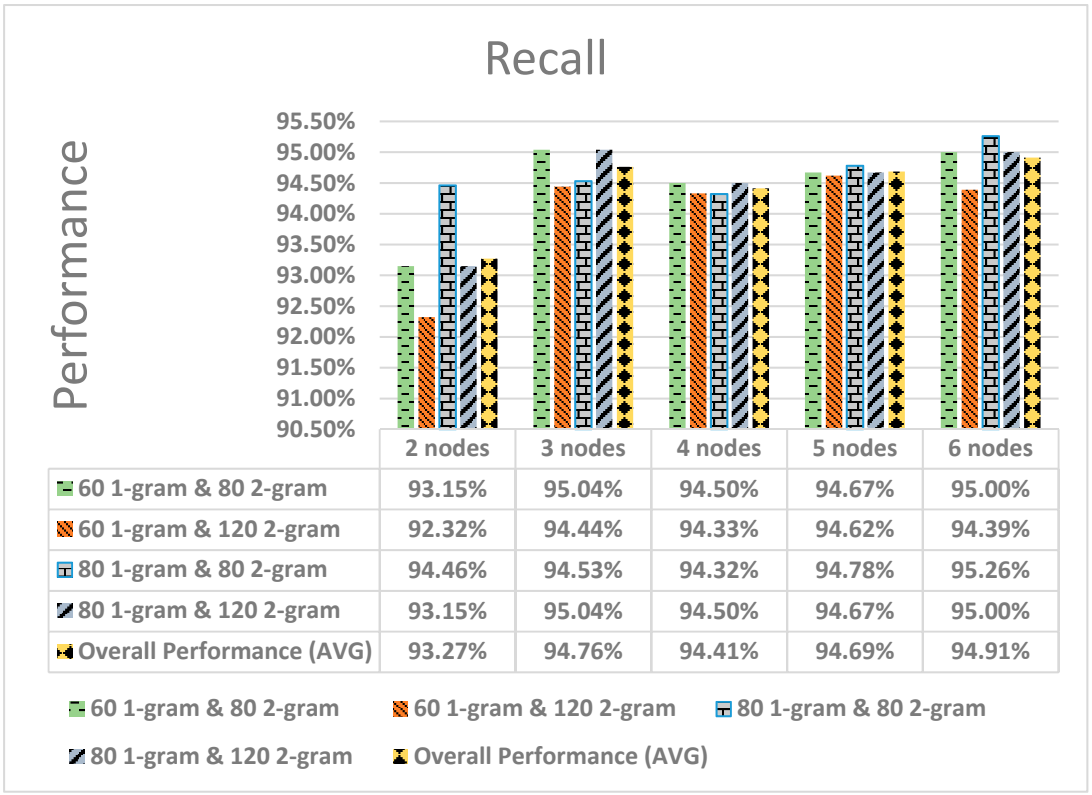
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	92.98%	94.95%	94.38%	94.64%	94.95%
60 1-gram & 120 2-gram	92.25%	94.33%	94.28%	94.59%	94.28%
80 1-gram & 80 2-gram	94.28%	94.49%	94.33%	94.80%	95.21%
80 1-gram & 120 2-gram	92.98%	94.95%	94.38%	94.64%	94.95%
Overall Performance (AVG)	93.12%	94.68%	94.34%	94.67%	94.85%

Figure 6. The performance of the model with respect to accuracy value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-LD.



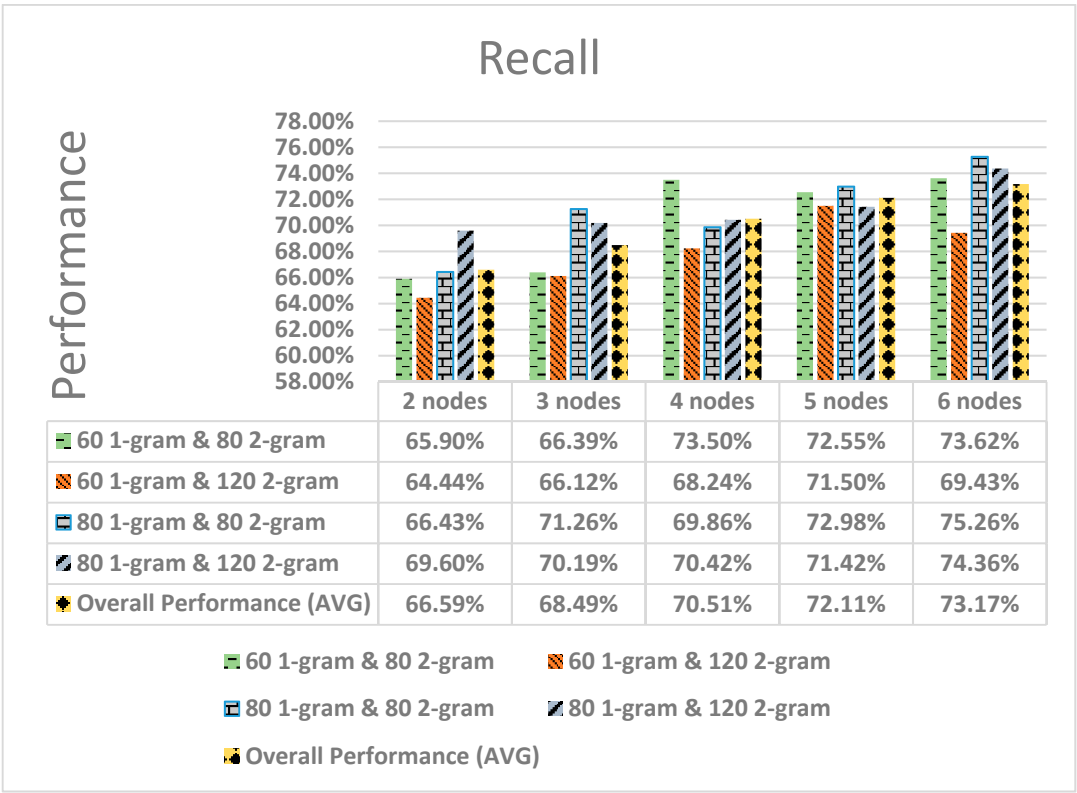
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	65.88%	66.36%	73.50%	72.53%	73.61%
60 1-gram & 120 2-gram	64.39%	66.06%	68.21%	71.04%	69.42%
80 1-gram & 80 2-gram	66.36%	71.23%	69.84%	72.96%	75.24%
80 1-gram & 120 2-gram	69.55%	70.14%	70.38%	71.40%	74.35%
Overall Performance (AVG)	66.55%	68.45%	70.48%	71.98%	73.16%

Figure 7. The performance of the model with respect to accuracy value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-WD.



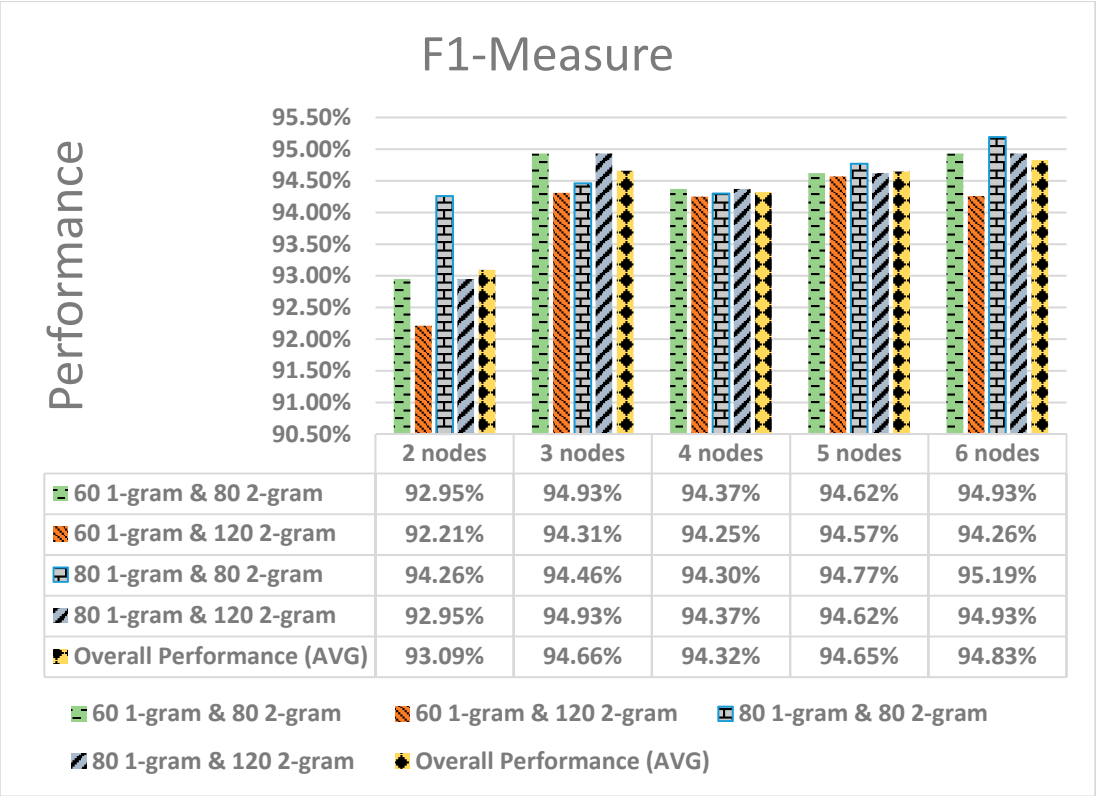
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	93.15%	95.04%	94.50%	94.67%	95.00%
60 1-gram & 120 2-gram	92.32%	94.44%	94.33%	94.62%	94.39%
80 1-gram & 80 2-gram	94.46%	94.53%	94.32%	94.78%	95.26%
80 1-gram & 120 2-gram	93.15%	95.04%	94.50%	94.67%	95.00%
Overall Performance (AVG)	93.27%	94.76%	94.41%	94.69%	94.91%

Figure 8. The performance of the model with respect to recall value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-LD.



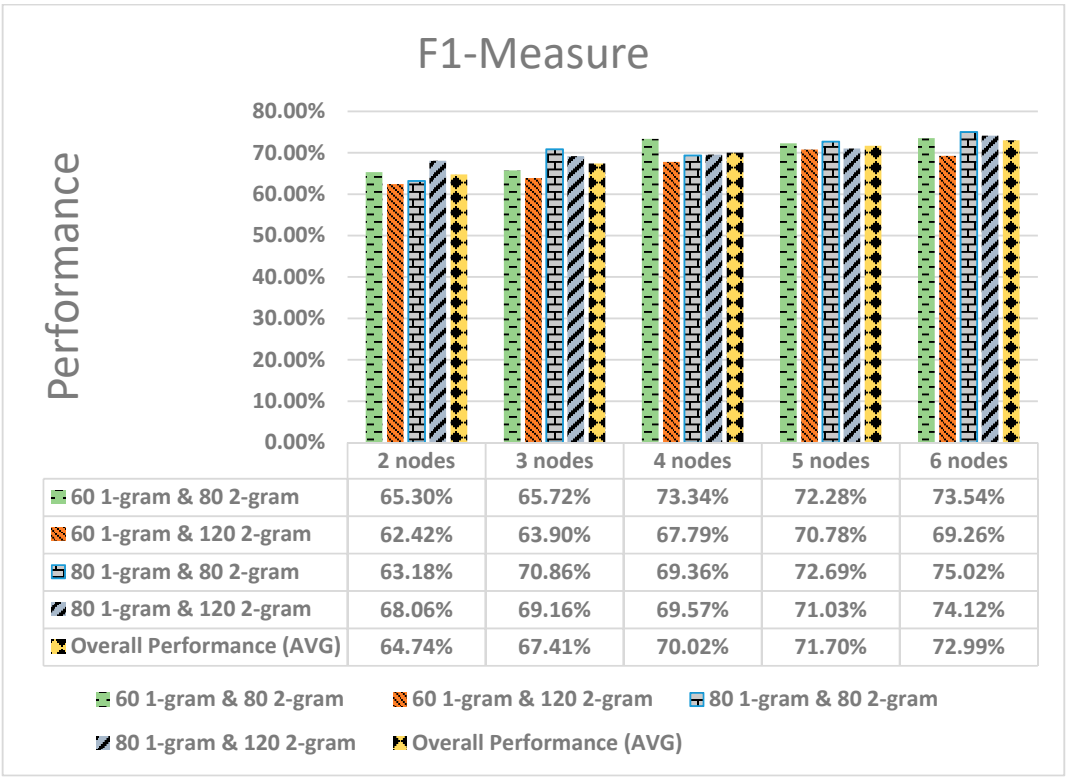
	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	65.90%	66.39%	73.50%	72.55%	73.62%
60 1-gram & 120 2-gram	64.44%	66.12%	68.24%	71.50%	69.43%
80 1-gram & 80 2-gram	66.43%	71.26%	69.86%	72.98%	75.26%
80 1-gram & 120 2-gram	69.60%	70.19%	70.42%	71.42%	74.36%
Overall Performance (AVG)	66.59%	68.49%	70.51%	72.11%	73.17%

Figure 9. The performance of the model with respect to recall value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-WD.



	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	92.95%	94.93%	94.37%	94.62%	94.93%
60 1-gram & 120 2-gram	92.21%	94.31%	94.25%	94.57%	94.26%
80 1-gram & 80 2-gram	94.26%	94.46%	94.30%	94.77%	95.19%
80 1-gram & 120 2-gram	92.95%	94.93%	94.37%	94.62%	94.93%
Overall Performance (AVG)	93.09%	94.66%	94.32%	94.65%	94.83%

Figure 10. The performance of the model with respect to F1-measure value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-LD.



	2 nodes	3 nodes	4 nodes	5 nodes	6 nodes
60 1-gram & 80 2-gram	65.30%	65.72%	73.34%	72.28%	73.54%
60 1-gram & 120 2-gram	62.42%	63.90%	67.79%	70.78%	69.26%
80 1-gram & 80 2-gram	63.18%	70.86%	69.36%	72.69%	75.02%
80 1-gram & 120 2-gram	68.06%	69.16%	69.57%	71.03%	74.12%
Overall Performance (AVG)	64.74%	67.41%	70.02%	71.70%	72.99%

Figure 11. The performance of the model with respect to F1-measure value using 2, 3, 4, 5, and 6 nodes and all 1-gram and 2-gram combinations in ADFA-WD.

Performance on the Raspberry Pi

Raspberry Pi is significantly slower than regular computers and its limited computational resources makes a suitable platform for evaluating the model performance. Table 5 shows the CPU time and energy of the model in a Raspberry Pi 3 model B for both datasets. The CPU time and energy could be optimized by utilizing of the distributed nature of fog computing. We observe that the average testing time is around 750×10^{-6} seconds or 750 microseconds which is a reasonable delay for many IoT applications.

Table 5. CPU Time and Energy of testing.

No. of Nodes	ADFA-LD		ADFA-WD	
	CPU-Time (Testing)	Energy Joule	CPU-Time (Testing)	Energy Joule
2 hidden nodes	751 µsec.	0.001502037	741 µsec.	0.00148201
3 hidden nodes	756 µsec.	0.001512051	746 µsec.	0.001492023
4 hidden nodes	753 µsec.	0.001505852	769 µsec.	0.001538277
5 hidden nodes	704 µsec.	0.0014081	684 µsec.	0.001368046
6 hidden nodes	817 µsec.	0.001634121	779 µsec.	0.001558304

The Energy consumption of Raspberry Pi is calculated using a conventional method similar to a regular computer. The Voltage value for Raspberry Pi model 3 B is +5.1 V. As for the current requirement, an HDMI port and a keyboard are connected to the Raspberry Pi, leading to an electrical current requirement of 400 mA. At this point, the product of the Current (I) and the Voltage yields (V) the Power consumption, which is then multiplied by the CPU time (t) to calculate the energy consumption shown in Table 5 [42].

$$P\ (Power) = I * V \tag{11}$$

$$Energy\ Consumption = P * t \tag{12}$$

6. Conclusions

In this paper, we present a lightweight intrusion detection model based on single hidden layer Multilayer Perceptron (MLP) model for Fog computing. To be lightweight, our proposed feature extraction technique utilizes modified vector space representation via n-gram transformation. Sparse matrix is also applied to compress the matrix formatting. Other than that, the linear correlation coefficient (LCC) is used to compensate the zero values. We also use mutual information feature selection to reduce the number of features. The proposed method is then tested using the ADFA-LD dataset using a Raspberry Pi, which acts as the Fog device. The test is executed with 2 nodes, 3 nodes, 4 nodes, 5 nodes, and 6 nodes of the MLP single hidden layer. From the experiment, we found out that 3 nodes of the MLP single hidden layer with 80 1-gram & 120 2-gram are able to achieve 94% accuracy, 95% recall rate, and 94% F1 measure. This is comparable with the higher number of nodes (i.e., 3 nodes to 6 nodes), which gives similar results. Since the 3-node set-up has lower complexity, and there is no significant advantage when a higher number of nodes is implemented, thus the 3-node implementation is suggested for the fog device. The values from ADFA-LD, i.e., 94% accuracy, was benchmarked, while 74% was achieved using 6 nodes in ADFA-WD. ADFA-LD showed better performance than ADFA-WD.

The power consumption of the Raspberry Pi in the experiment's configuration was calculated using the electrical current requirement and the voltage. Along with the CPU time of the experiment, the energy consumption was then calculated, which is around 0.0014 Joules.

7. Future Work

There is still room for improvement in terms of detection accuracy and computational efficiency by adding more features or using more efficient algorithms. One possible direction is to use a more efficient learning algorithm instead of backpropagation. Furthermore, testing the performance on

different fog nodes could be a helpful in evaluating the MLP model. This trend will grow and continued evaluation of IDS implementation at each service layer as well as benchmarks between systems will be important for the safety of the machines and their owners.

Author Contributions: The concept, design and evaluation of the algorithm is done by B.S.K.; M.Y.I.B.I. and A.W.B.A.W., structure the paper and device experiments with selected benchmarking; B.S.K., M.A.H. and A.A.I. contributed in the software implementation and the evaluation of the results.

Funding: This work was supported by the University of Malaya under Faculty RU Grant (GPF003D-2018).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Computing, F. The internet of things: Extend the cloud to where the things are. In *Cisco White Paper*; Cisco: Charlotte, NC, USA, 2015.
2. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Cham, UAE, 2014; pp. 169–186.
3. Dsouza, C.; Ahn, G.-J.; Taguinod, M. Policy-driven Security Management for Fog Computing: Preliminary Framework and a Case Study. In Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IRI), Redwood City, CA, USA, 13–15 August 2014; pp. 16–23.
4. Khan, S.; Parkinson, S.; Qin, Y. Fog computing security: A review of current applications and security solutions. *J. Cloud Comput.* **2017**, *6*, 19. [CrossRef]
5. Sedjelmaci, H.; Senouci, S.M.; Al-Bahri, M. A Lightweight Anomaly Detection Technique for Low-resource IoT Devices: A Game-theoretic Methodology. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
6. An, X.; Zhou, X.; Lü, X.; Lin, F.; Yang, L. Sample selected extreme learning machine based intrusion detection in fog computing and MEC. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 7472095. [CrossRef]
7. Hosseinpour, F.; Vahdani Amoli, P.; Plosila, J.; Hämäläinen, T.; Tenhunen, H. An intrusion detection system for fog computing and IoT based logistic systems using a smart data approach. *Int. J. Digit. Content Technol. Appl.* **2016**, *10*, 34–46.
8. Alrawais, A.; Alhothaily, A.; Hu, C.; Cheng, X. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Comput.* **2017**, *21*, 34–42. [CrossRef]
9. Stojmenovic, I.; Wen, S. The Fog Computing Paradigm: Scenarios and Security Issues. In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS), Warsaw, Poland, 7–10 September 2014; pp. 1–8.
10. Lee, K.; Kim, D.; Ha, D.; Rajput, U.; Oh, H. On Security and Privacy Issues of Fog Computing Supported Internet of Things Environment. In Proceedings of the 2015 6th International Conference on the Network of the Future (NOF), Montreal, QC, Canada, 30 September–2 October 2015; pp. 1–3.
11. Wang, Y.; Uehara, T.; Sasaki, R. Fog Computing: Issues and Challenges in Security and Forensics. In Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), Taichung, Taiwan, 1–5 July 2015; pp. 53–59.
12. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* **2016**, *3*, 854–864. [CrossRef]
13. Markakis, E.K.; Karras, K.; Sideris, A.; Alexiou, G.; Pallis, E. Computing, caching, and communication at the edge: the cornerstone for building a versatile 5G ecosystem. *IEEE Commun. Mag.* **2017**, *55*, 152–157. [CrossRef]
14. Calabretta, M.; Pecori, R.; Vecchio, M.; Veltri, L. MQTT-Auth: A token-based solution to endow MQTT with authentication and authorization capabilities. *J. Commun. Softw. Syst.* **2018**, *14*, 320–331. [CrossRef]
15. Napiah, M.N.; Idris, M.Y.I.B.; Ramli, R.; Ahmedy, I. Compression header analyzer intrusion detection system (CHA-IDS) for 6LoWPAN communication protocol. *IEEE Access* **2018**, *6*, 16623–16638. [CrossRef]
16. Goeschel, K. Reducing False Positives in Intrusion Detection Systems Using Data-mining Techniques Utilizing Support Vector Machines, Decision Trees, and Naive Bayes for Off-line Analysis. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–6.

17. Kim, G.; Lee, S.; Kim, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Syst. Appl.* **2014**, *41*, 1690–1700. [CrossRef]
18. Alharbi, S.; Rodriguez, P.; Maharaja, R.; Iyer, P.; Subaschandrabose, N.; Ye, Z. Secure the Internet of Things with Challenge Response Authentication in Fog Computing. In Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC), San Diego, CA, USA, 10–12 December 2017; pp. 1–2.
19. Ham, H.-S.; Kim, H.-H.; Kim, M.-S.; Choi, M.-J. Linear SVM-based android malware detection for reliable IoT services. *J. Appl. Math.* **2014**, *2014*, 594501. [CrossRef]
20. Ozay, M.; Esnaola, I.; Vural, F.T.Y.; Kulkarni, S.R.; Poor, H.V. Machine learning methods for attack detection in the smart grid. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 1773–1786. [CrossRef]
21. Pajouh, H.H.; Javidan, R.; Khayami, R.; Ali, D.; Choo, K.-K.R. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. *IEEE Trans. Emerg. Top. Comput.* **2016**. [CrossRef]
22. Meidan, Y.; Bohadana, M.; Shabtai, A.; Guarnizo, J.D.; Ochoa, M.; Tippenhauer, N.O.; Elovici, Y. ProfilIoT: A Machine Learning Approach for IoT Device Identification based on Network Traffic Analysis. In Proceedings of the Symposium on Applied Computing, Marrakech, Morocco, 3–7 April 2017; pp. 506–509.
23. Lee, S.-Y.; Wi, S.-R.; Seo, E.; Jung, J.-K.; Chung, T.-M. ProFiOt: Abnormal Behavior Profiling (ABP) of IoT Devices based on a Machine Learning Approach. In Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, Australia, 22–24 November 2017; pp. 1–6.
24. Ge, M.; Hong, J.B.; Guttmann, W.; Kim, D.S. A framework for automating security analysis of the internet of things. *J. Netw. Comput. Appl.* **2017**, *83*, 12–27. [CrossRef]
25. Shen, S.; Huang, L.; Zhou, H.; Yu, S.; Fan, E.; Cao, Q. Multistage signaling game-based optimal detection strategies for suppressing malware diffusion in fog-cloud-based IoT networks. *IEEE Internet Things J.* **2018**, *5*, 1043–1054. [CrossRef]
26. Borisaniya, B.; Patel, D. Evaluation of modified vector space representation using adfa-ld and adfa-wd datasets. *J. Inf. Secur.* **2015**, *6*, 250. [CrossRef]
27. Xie, M.; Hu, J.; Yu, X.; Chang, E. Evaluating Host-based Anomaly Detection Systems: Application of the Frequency-based Algorithms to Adfa-ld. In Proceedings of the International Conference on Network and System Security, Xiamen, China, 19–21 August 2014; pp. 542–549.
28. Xie, M.; Hu, J.; Slay, J. Evaluating Host-based Anomaly Detection Systems: Application of the One-class Svm Algorithm to Adfa-ld. In Proceedings of the 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Xiamen, China, 19–21 August 2014; pp. 978–982.
29. Creech, G. Developing a High-Accuracy Cross Platform Host-Based Intrusion Detection System Capable of Reliably Detecting Zero-Day Attacks. Ph.D. Thesis, University of New South Wales, Canberra, Australia, 2014.
30. Creech, G.; Hu, J. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819. [CrossRef]
31. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008; Volume 1.
32. Leslie, C.; Eskin, E.; Noble, W.S. The spectrum kernel: A string kernel for SVM protein classification. In *Biocomputing 2002*; World Scientific: Singapore, 2001; pp. 564–575.
33. Wang, J.T.-L.; Ma, Q.; Shasha, D.; Wu, C.H. New techniques for extracting features from protein sequences. *IBM Syst. J.* **2001**, *40*, 426–441. [CrossRef]
34. Data, L.F.; Course, A.S.; Abu-Mostafa, Y.S.; Magdon-Ismail, M.; Lin, H.-T.I. Scholarly Activity. In Proceedings of the 12th Conference on Electronic Commerce (EC), Valencia, Spain, 4–8 June 2012.
35. Creech, G.; Hu, J. Generation of a New IDS Test Dataset: Time to Retire the KDD Collection. In Proceedings of the 2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, China, 7–10 April 2013; pp. 4487–4492.
36. Haider, W.; Creech, G.; Xie, Y.; Hu, J. Windows based data sets for evaluation of robustness of host based intrusion detection systems (IDS) to zero-day and stealth attacks. *Future Internet* **2016**, *8*, 29. [CrossRef]

37. Xie, M.; Hu, J. Evaluating Host-based Anomaly Detection Systems: A Preliminary Analysis of Adfa-ld. In Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP), Hangzhou, China, 16–18 December 2013; pp. 1711–1716.
38. Abubakar, A.I.; Chiroma, H.; Muaz, S.A.; Ila, L.B. A review of the advances in cyber security benchmark datasets for evaluating data-driven based intrusion detection systems. *Procedia Comput. Sci.* **2015**, *62*, 221–227. [CrossRef]
39. Borthakur, D.; Dubey, H.; Constant, N.; Mahler, L.; Mankodiya, K. Smart Fog: Fog Computing Framework for Unsupervised Clustering Analytics in Wearable Internet of Things. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, QC, Canada, 14–16 November 2017; pp. 472–476.
40. Constant, N.; Borthakur, D.; Abtahi, M.; Dubey, H.; Mankodiya, K. Fog-assisted wiot: A smart fog gateway for end-to-end analytics in wearable internet of things. *arXiv* **2017**, arXiv:1701.08680.
41. Lavassani, M.; Forsström, S.; Jennehag, U.; Zhang, T. Combining fog computing with sensor mote machine learning for industrial IoT. *Sensors* **2018**, *18*, 1532. [CrossRef] [PubMed]
42. *Learning U. Raspberry Pi 3: Get Started with Raspberry Pi 3 a Simple Guide TO Understanding and Programming Raspberry Pi 3 (Raspberry Pi 3 User Guide, Python Programming, Mathematica Programming)*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2016.