Journal of Network and Computer Applications 163 (2020) 102662

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Detecting Internet of Things attacks using distributed deep learning

Gonzalo De La Torre Parra [a], Paul Rad [b,a], Kim-Kwang Raymond Choo [b,*], Nicole Beebe [b]

[a] Secure AI & Autonomous Laboratory, University of Texas at San Antonio, San Antonio, TX, 78249, USA
[b] Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX, 78249, USA

ARTICLE INFO

Keywords:
Cyber security
Cloud computing
Machine learning
Deep learning
Recurrent neural network

ABSTRACT

The reliability of Internet of Things (IoT) connected devices is heavily dependent on the security model employed to protect user data and prevent devices from engaging in malicious activity. Existing approaches for detecting phishing, distributed denial of service (DDoS), and Botnet attacks often focus on either the device or the back-end. In this paper, we propose a cloud-based distributed deep learning framework for phishing and Botnet attack detection and mitigation. The model comprises two key security mechanisms working cooperatively, namely: (1) a Distributed Convolutional Neural Network (DCNN) model embedded as an IoT device micro-security add-on for detecting phishing and application layer DDoS attacks; and (2) a cloud-based temporal Long-Short Term Memory (LSTM) network model hosted on the back-end for detecting Botnet attacks, and ingest CNN embeddings to detect distributed phishing attacks across multiple IoT devices. The distributed CNN model, embedded into a ML engine in the client's IoT device, allows us to detect and defend the IoT device from phishing attacks at the point of origin. We create a dataset consisting of both phishing and non-phishing URLs to train the proposed CNN add-on security model, and select the N_BaIoT dataset for training the back-end LSTM model. The joint training method minimizes communication and resource requirements for attack detection, and maximizes the usefulness of extracted features. In addition, an aggregation of schemes allows the automatic fusion of multiple requests to improve the overall performance of the system. Our experiments show that the IoT micro-security add-on running the proposed CNN model is capable of detecting phishing attacks with an accuracy of 94.3% and a F-1 score of 93.58%. Using the back-end LSTM model, the model detects Botnet attacks with an accuracy of 94.80% using all malicious data points in the used dataset. Thus, the findings demonstrate that the proposed approach is capable of detecting attacks, both at device and at the back-end level, in a distributed fashion.

1. Introduction

Internet of Things (IoT) devices underpin many technological trends and infrastructures, such as smart homes and smart cities. These internet-connected devices generate, process, and exchange significant volumes of data during their operations. They also establish communications between devices and systems, as well as with their users, through various types of networks. IoT devices have been used in applications such as smart vehicles, healthcare, environmental monitoring, and personal wearable devices, which contribute to the increase in the volume, variety, velocity, and veracity of data (e.g. sensitive data such as personal information) managed by these devices and the connected systems. Hence, there has been increased attention from both industry and academia to develop security solutions for IoT devices, although security requirements may not be placed at the same priority level as product innovation.

The interest in developing security solutions for IoT devices is also partly due to these devices having weaker security and their increasing popularity in our society (Pour et al., 2019). For example, according to Symantec's 2019 Internet Security Threat Report (O'Gorman et al., 2019), routers and cameras were the most infected devices, respectively accounting for 75% and 15% of IoT attacks. Examples of popular IoT malware include the Mirai distributed denial of service (DDoS) worm and VPN Filter, where the latter is reportedly armed with potent payloads, including ones targeted for data exfiltration, credential theft, etc. Similarly, Cisco's 2018 report (Cisco, 2018) revealed that network-based ransomware (unlike previously used malware spreading methods such as drive-by-download, email, or USB drives) do not require human interaction to infect devices. Examples of such ransomware include WannaCry and Nyetya, which exploited "Eternal Blue" (a Microsoft

* Corresponding author.
E-mail address: raymond.choo@fulbrightmail.org (K.-K.R. Choo).

https://doi.org/10.1016/j.jnca.2020.102662
Received 2 October 2019; Received in revised form 20 February 2020; Accepted 1 April 2020
Available online 11 April 2020

Windows security vulnerability). The U.S. government and other security entities also reported that WannaCry utilized the ransom component as a smokescreen to disguise its true purpose of wiping data belonging to infected users. Similarly, Nyetya was disguised as a ransomware in order to wipe out data from its victims. The Nyeta attack was also facilitated by exploiting the "Eternal Blue" vulnerability, a remote code execution vulnerability known as "Eternal Romance," as well as vectors involving credential harvesting.

Attempting to detect attacks based on their network behaviour is an ongoing effort, which is evident by the number of anomaly based intrusion detection systems proposed in the literature. For example, the proposed models of [Shone etal., (2018), Diro and Chilamkurti, (2018) and Moustafa et al., (2019)] use deep learning approaches to detect network traffic anomalies and identify zero-day attacks by learning patterns from normal-traffic and abnormal-traffic during the training phase. There have also been attempts to use artificial intelligence (AI) based tools to complement threat prevention, detection, and remediation, particularly for detecting malicious encrypted web traffic use (Cisco, 2018; Casino et al., 2019; O'Gorman et al., 2019).

Existing proposals focus on detecting attacks at the device level on the client's side or at the back-end hosted on the cloud. Such an approach limits the ability to detect attacks taking place across distributed IoT devices, while simultaneously offering on-device security. Therefore, we propose a model that enables the detection of attacks both at the client's side and the back-end simultaneously. The client's side hosts a micro-security add-on running CNN model for detecting DDoS and phishing attacks. In addition, embeddings from the CNN model are passed to the back-end long-short term memory based model for detecting distributed attacks as well as botnets. This is the first, if not one of few, solutions that detect IoT attacks both at the device level and at the back-end in a distributed fashion. The contributions of this paper are as follow:

- We design a novel distributed Long Short-Term Memory (LSTM) neural network framework and present its implementation, which maps sections of a Convolutional Neural Network (CNN) into a distributed computing hierarchy running on client devices to detect and defend at the point of attack origin.
- We design a joint training method that minimizes communication and resource usage for attack detection in IoT devices, and maximizes usefulness of extracted features that are utilized at the back-end server. This facilitates malicious URL classification via the client's machine learning (ML) engine hosted on the IoT devices (to minimize service disruption).
- We also design aggregation schemes that allow automated fusion of multiple URL requests to improve the overall performance (accuracy and fault tolerance) of the system.

The rest of this paper is organized as follows. Section 2 presents an overview of the attacks of interest and related work. In Section 3, we present our proposed model. In Section 4, we briefly review publicly available datasets that have been used in the literature. In Section 5, we present our experimental setup, the utilized performance metrics, the data pre-processing, the feature generation process, our experimental results, and the comparison of our work with other competing approaches. Finally, in Section 6, we present our conclusion and future work.

2. Attack and mitigation strategies

The emergence of IoT devices has not gone unnoticed to attackers looking for opportunities to infect and control new computational resources in order to steal credentials or generate attacks from infected devices. Although IoT devices typically have limited low processing, power, and memory capabilities, these devices are designed to function 24/7 and connect to the internet, as well as other IoT devices using wireless interfaces. Not surprisingly, IoT devices are an increasingly popular attack target.

HTTP requests made by a client are composed by a request line consisting of a Method that changes depending on the requested action (HEAD, GET, POST, PUT, OPTIONS, DELETE, and CONNECT) as well as the Path to reach. The Header section contains the URL to be reached (in addition to the content language, length, and type); finally, the Body contains desired data to be sent to the web server. On the other hand, the web server's response sends a HTTP status code, a Header, and a Body similar to the request. In Fig. 2, we present a visual example of the HTTP protocol structure.

Attacks fabricating or changing URLs can be easily analyzed within an IoT device due to the low processing requirements to perform this task. On the other hand, attacks that use the Body of the HTTP request, as presented in Fig. 3, require a deeper analysis due to the length of the content as well as the relationship between the data contained in multiple requests which may be linked to a particular attack. As represented in Fig. 1, an attacker can direct its arsenal of attacks towards browser enabled devices such as cellphones, laptops, and IoT devices in order to collect sensitive information or control multiple devices to facilitate DDoS attacks.

2.1. URL attacks

Distributed Denial of Service (DDoS) attacks are designed to prevent legitimate users from accessing a legitimate website or service. The targeted website or service is bombarded with requests coming from multiple sources making it unable to respond to legitimate users. The attack can be produced in a variety of ways including pingback, as presented in Fig. 4, and fake traffic. Such attacks are effective when the server hosting the target website is unable to handle the received traffic. Due to the capability current servers have for handling large amounts of traffic, an attack perpetrated from a single source is not possible. In order to overwhelm the resources of such servers, attackers can use multiple sources to generate the attack. A DDoS attack consists of many computers, servers, or legitimate websites under the control of a botnet to overload a targeted website. Once traffic is directed to the target, the target's system resources (bandwidth, CPU, RAM, etc) are consumed by the incoming traffic, preventing users from accessing it.

DDoS attacks remain a major threat due to their growth and evolution throughout the years. These attacks were initially generated using malformed packets or flooding a specific device with network layer packets. As defense mechanisms evolved and became more robust to DDoS attacks, attackers started targeting the application layer. DDoS attacks at the application layer are more difficult to detect as the traffic generated by these attacks resemble to normal user traffic. These attacks are now generated by using legitimate user requests, ruling out packet inspection approaches to determine whether they are malicious or not. These attacks can also be executed using multiple protocols at the application layer, increasing the danger they pose (Praseed and Thilagam, 2018).

Phishing attacks consist of luring users, for example via email messages, to click links that redirect the user to a particular site. The site's payload can include malicious code designed to steal personal or financial information – see Fig. 5. There are also other variants of targeted phishing, such as spear phishing.

Phishing solutions have encountered many difficulties to detect these attacks as phishers are constantly developing new and creative ways to trick users to fall for illegitimate websites or emails. Phishers are forging websites that appear to be identical to original sources by using logos and graphics that make phishing emails to look more convincing. In addition, new approaches that scrape public personal data from potential victims are used for generating highly convincing attacks. Social phishing and context aware phishing are two methods that use public personal information for increasing the effectiveness of these attacks. A study has shown that victims are 4.5 times more likely to fall for these attacks if the source involves a personal contact or per-

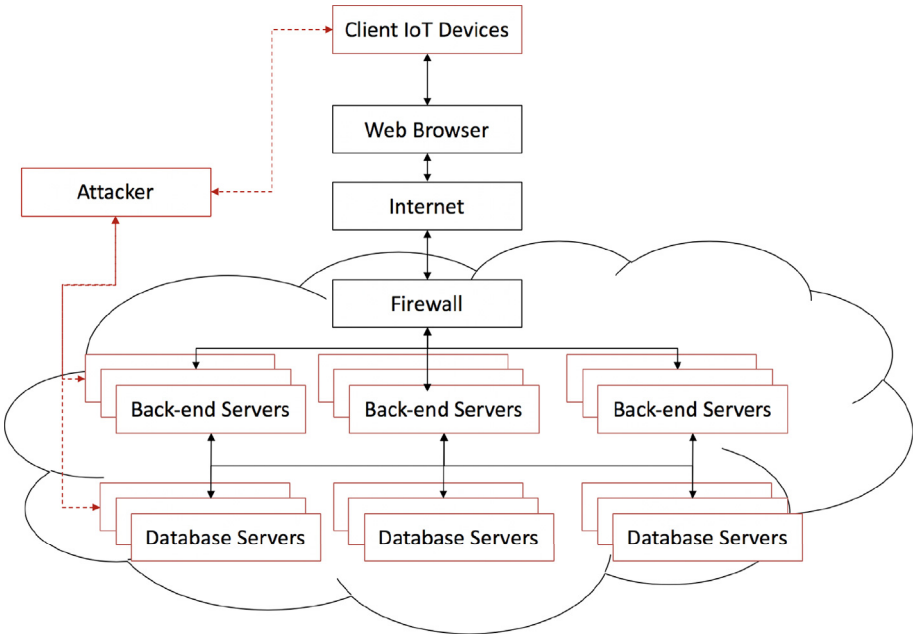

Fig. 1. Infrastructure Vulnerabilities: An attacker is capable of attacking a client's IoT device as well as websites, web servers, and databases located at the cloud back-end server side.

Request Method	GET
Protocol Version	HTTP/1.1
Request URL	http://localhost:8080/tienda1/publico/anadir.jsp?id
Request Headers	User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko) Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 Accept-Encoding: x-gzip, x-deflate, gzip, deflate Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5 Accept-Language: en Host: localhost:8080 Cookie: JSESSIONID=54E25FF4B7F0E4E855B112F882E9EEA5
General Headers	Pragma: no-cache Cache-control: no-cache Connection: close
Message	id=1&nombre=Jam%F3n+Ib%E9rico&precio=39&cantidad=41&B1=A%F1adir+al+carrito

Fig. 2. Structure of a HTTP Request: It comprises (1) a request line containing a Method, a URL, and protocol version; (2) the Headers section containing the HOST of interest; and (3) the Body carrying the message containing data sent by the client or the web server.

Request Method	GET
Protocol Version	HTTP/1.1
Request URL	http://localhost:8080/tienda1/publico/autenticar.jsp?
Request Headers	User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like Gecko) Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 Accept-Encoding: x-gzip, x-deflate, gzip, deflate Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5 Accept-Language: en Host: localhost:8080 Cookie: JSESSIONID=FBD7B67801CAF901F3F378668C5FF56F
General Headers	Pragma: no-cache Cache-control: no-cache Connection: close
Message	modo=entrar&login=mcnair&pwd=81cob6r&remember=off&B1=%27%3B+DROP+TABLE+usuarios%3B+SELECT+*+FROM+datos+WHERE+nombre+LIKE+%27%25

Fig. 3. HTTP Malicious request containing the attack in the message. As it can be observed the message contains a malicious SQL request attempting to acquire data.

sonally relates to the victim (Vayansky and Kumar, 2018). Such methods fall within the classification of spear-phishing with attacks targeting specific victims. Other phishing attacks include malware and trojans to directly compromise a victim's computer which is used for generating new attacks.

2.2. Examples of SQL injection and phishing attack detection

Abdelhamid et al. (2017) present a machine learning-based analysis of a variety of phishing detection methods. Using the WEKA machine learning tool, they evaluate the performance of eight machine learning techniques on phishing datasets. Their findings suggest that the C4.5 decision tree algorithm and the eDRI outperform the other six algorithms, but it is also observed that the C4.5 algorithm generates 297 rules for phishing classification while the eDRI only generates 25 rules for phishing classification; thus, eDRI appears to be ideal for computationally constrained devices. The authors fail to present how their approach can be extended beyond the implementation of the proposed classifiers within a web browser.

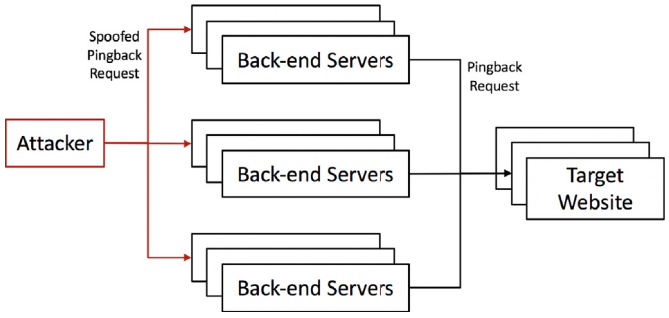

Fig. 4. WordPress Pingback Attack: A DDoS attack exploiting pingback vulnerabilities found in WordPress. The attacker distributes spoofed pingback requests to legitimate WordPress websites redirecting HTTP requests to a target's website.

RE: Your Winning Batch #P8997611.
Felipe Alfredo Rossler Hargous
<felipe.rossler@mineduc.cl>
Reply
Wed 4/18, 2:53 PM
Felipe Alfredo Rossler Hargous
(felipe.rossler@mineduc.cl)

From: Felipe Alfredo Rossler Hargous
Sent: 19 April 2018 01:18
Subject: Your Winning Batch #P8997611.

Attention,

You are among the selected candidates to claim the publishers clearing house (P.C.H) sweepstakes prize worth $1 Million US Dollars. Make sure to send your winning batch Number:#P8997611. To the claims department email via (pch.affairs15@hotmail.com) to process the claims of your prize.

P.C.H.

Fig. 5. Phishing attacks: The attackers use social engineering to lure a victim to share critical information.

2.3. Examples of DDoS attack detection

DDoS attacks can be broadly categorized into network/transport level DDoS-based flooding attacks and application layer DDoS flooding attacks (Yusof et al., 2019). Network/transport level DDoS-based flooding attacks use ICMP, DNS protocol packets, TCP, and UDP to disrupt the connection from legitimate users. On the other hand, application layer DDoS flooding attacks are designed to consume the target's server resources, such as memory, disk/database and I/O bandwidth and sockets. A number of solutions have been proposed in the literature to detect both categories of DDoS attacks. For example, Jing et al. (2019) use the Chinese remainder theorem based on reversible sketch, which compresses and fuses big-volume network traffic. In addition, their approach includes the discovery of anomalous keys. The authors propose a modified multi-chart cumulative sum algorithm that supports self-adaptive and protocol independent detection to detect DDoS flooding attacks. Cheng et al. (2018) present an abnormal network flow feature sequence prediction approach. Doshi et al. (2018) present an IoT-based low-cost machine learning algorithms that uses feeds from traffic data that is flow-based and protocol-agnostic. A major disadvantage of the presented approach is that it requires a middlebox for packet capturing (PCAP), processing network traffic, and generating features. Nevertheless, the tested K-nearest neighbors algorithm, support vector machine with linear kernel, decision tree, random forest, and DNN (Deep Neural Network) present an accuracy over 0.99. A different approach based on complex event processing is presented by Cardoso et al. (da Silva Cardoso et al., 2018). The proposed model consists of a packet analyzer and an attack detection module.

2.4. Botnets

A botnet is a collection of internet-connected computing devices under the control of an outside party without the owner's knowledge. With the control of a massive amount of distributed devices, attackers can perpetrate an attack, such as DDoS, against a specific target. Such Botnets are created by compromising devices though the exploitation of existing vulnerabilities. An IoT botnet is created by controlling multiple IoT devices, which have become an attractive target due to their lack of security and stripped down operating systems.

Two examples of such botnets are Bashlite (also known as Gafgyt) and Mirai. Bashlite uses scanners to identify vulnerable devices. Once identified, loaders establish a connection to load and run malware. Once infected, the infected bot connects to a command and control unit to await for further instructions. According to Marzano et al. (2018), 583 command names were found in Bashlite. Bashlite targets IoT devices to build botnets and conduct DDoS attacks. On October 12, 2016 Mirai, for which the source code is based on Bashlite's (Marzano et al., 2018), exploited hundreds of thousands of infected IoT devices to perpetrate a massive DDoS attack. The attack reached an offensive capability of 1.2 Terabits per second that left the U.S. East Coast without access to the internet. Mirai was able to infect such a large scale of devices by using a dictionary attack with about 60 entries, taking advantage of default credentials in IoT devices (De Donno et al., 2018). For the purpose of studying Botnets, Mirsky et al. (2018) and Meidan et al. (2018) captured the statistics of IoT devices' network traffic packets over sliding windows, generating a dataset for studying Mirai and Bashlite botnets. Both works propose a solution for detecting Bashlite and Mirai attacks based on Autoencoders. Radford et al. (2018) propose the use of a simple frequency-based method for detecting outliers as well as the use of a LSTM model for detecting attacks. Chawathe (2018) presents a methodology for reducing the number of features in the dataset presented in the works of Mirsky et al. (2018) and Meidan et al. (2018). In addition, the authors tested the trimmed datasets using different simple classifiers.

Botnets are still an open challenge in the research community. Approaches such as domain blacklisting have been used for attenuating botnet attacks; nevertheless, these have shown a limited success as botnets adopt domain generation algorithms. In addition, the increasing number of IoT devices connecting to the internet has increased the number of resources that are used as bots due to their low or lack of security. Also, botnets are continually evolving to set their level of stealthiness, which has gone so high that these remain undetected for years. In addition, botnets are using social network sites, such as Twitter and Instagram, and development platforms, such as github, as their command and control server, increasing the number of challenges as these sites cannot be blocked (Singh et al., 2019).

Based on the covered material, we have identified deep learning as a potential approach detecting these attacks, as these can unravel a significantly greater amount of hidden patterns in the data that standard deterministic or probabilistic techniques fail to model. Current methods for detecting the discussed attacks have proved ineffective in prevent-

ing them. This occurs largely due to attack mutations that render attack signatures obsolete. In addition, deep learning approaches offer a significant improvement in accuracy of classification and predictions in complex tasks. Another main benefit of deep learning approaches is the absence of manual feature engineering, unsupervised pre-training, and compression capabilities enabling its application in resource constraint networks. This results in higher accuracy and faster processing that can be used for novel distributed attack detection in IoT systems (McDermott et al., 2018). For our proposed approach, we classify attacks into two categories: Botnet attacks and URL Based attacks. From our perspective, Botnet attacks require a deep analysis from features generated from the statistics of network traffic. Such deep analysis requires greater computational processing in order to detect attacks in real time. For such cases, we determine that the best approach is to perform the detection of these attack types on the back-end, past a firewall. On the other hand, the detection of Phishing and DDoS attacks is highly dependent on the URLs. For such cases, minimum processing capabilities are required in order to analyze URLs and detect Phishing sites. Therefore, we can leverage the computational power of IoT devices to detect these attacks on the client's side.

2.5. Comparative analysis

In contrast with existing proposals, our work focuses on the detection of various attacks on large pools of IoT devices, as well as back-end servers using a distributed architecture with aggregation capabilities. Other proposals presented in the literature simply focus on detecting attacks either on the client's side or the back-end's side. In addition, our proposal leverages the high volume of low computational power in client IoT devices, as well as high computational power of specialized servers on the back-end for efficiently detecting a variety of attacks depending on their complexity. In Table 15 we provide a performance comparison with anomaly detection models trained and tested using the CSIC 2010 dataset.

Works following a client-based application approach include: Phishing-Alarm (Mao et al., 2017), which relies on the fundamental features of the page's visual layout; Off-the-Hook (Eng) (Marchal et al., 2017), which collects features from URL, webpage content, RDN usage, term consistency usage, and usage of mlds to build their classification model; and proposals presented in PhishShield (Rao and Ali, 2015) and Kausar et al. (2014), who follow a heuristic-based approach and present high misclassification rates. We approach the detection of Phishing attacks by extracting features from URLs and generating a model that learns the syntactical structure of these. On the other hand, approaches following a back-end based application include Ma et al. (2009a), who based the proposed model on features related to web page hosting. Chen et al. (2014) follow a phishing-target site visual similarity approach. Thomas et al. (2011) present a model that crawls URLs and determine if they direct to spam. Their approach achieves a very high accuracy, but a high computational cost, which is not a resource not available in most IoT devices. Habeeb et al. (2019) propose a framework for anomaly detection evaluated using K-means HDBSCAN, isolation forest, spectral clustering, and agglomerative clustering. Their results show a 96.51% accuracy for detecting anomalies using network data. Their approach does not present a solution for detecting anomalies at the client's side. Kim et al. (2016) propose a model based on LSTM for anomaly detection. While their results show an accuracy of 97.87% for detecting DoS attacks, their model is evaluated using the KDD Cup 1999 dataset which does not reflect the network behaviour present in novel attacks.

While we have observed various works presenting innovative solutions to detect web attacks, most proposals present frameworks providing standalone solutions in one side of the spectrum (Client or Back-end) and leave the other side vulnerable to a wide variety of attacks, opening doors for attackers to take advantage of existing vulnerabilities. A comparative summary is presented in Table 13.

3. Proposed distributed deep learning

Our proposed model aims to protect IoT devices owned by multiple clients, as well as the back-end servers, through a distributed attack detection framework, powered by deep learning models, based on a special kind of recurrent neural network known as long-short term memory networks (LSTM). Such networks have become relevant due to their capability of learning long-term dependencies.

Due to the increasing number of IoT devices deployed in home and commercial settings, it is essential for corporate and academic organizations to focus on the development of defense mechanisms, such as those based on deep learning models, for defending against such devices. These models have the capability to analyze URL requests on devices with low computational resources, as well as network information on server-grade systems. IoT devices, laptops, tablets, mobile phones, and many others use the HTTP protocol for sending/receiving requests to/from web servers. Based on current frameworks and network topologies, we identify two essential groups requiring distributive security mechanisms for analyzing traffic at runtime in order to counter the ingenious approaches that attackers have been adopting in past years. In Fig. 6, we present an overview of our distributed CNN and LSTM architecture. Our proposal can be decomposed into two components: 1) an IoT micro-security add-on based on CNN hosted a client's IoT devices, and 2) the LSTM model hosted in back-end servers. For simplicity, our distributed solution can be viewed as a standard CNN running in IoT devices and a cloud-based LSTM distributed across multiple back-end servers.

For enforcing security in user devices such as smart phones, smart watches, and other IoT devices, we propose the use of an attack detection IoT micro-security add-on. The micro-security add-on is a lightweight module capable of running on devices with low computational and power resources, a common characteristic of IoT devices. Such an approach opens the opportunity for developing a distributed attack detection model without requiring users to install additional software. Since this model analyzes data in the IoT device, it honors a client's privacy by maintaining the data on the device. At its core, the add-on is hosted in a machine learning (ML) engine capable of training a ML model such as a CNN and inferring whether a request is valid or is attempting to perpetrate an attack to the IoT device. This mechanism is used for training, validating, and testing deep learning models that analyze incoming Internet traffic in real-time that could potentially carry harmful attacks. Furthermore, such ML engine allows the device to import existing deep learning models, develop new models, or retrain imported models with new Internet traffic data.

The IoT micro-security add-on at the device hosts the CNN model. This facilitates the real-time detection of attacks. For example, phishing and application-based DDoS attacks are detected by feeding features generated from parsed URLs. The CNN model is trained with labeled normal and abnormal URLs that allows the model to learn from these features. During the training phase, the model adjusts its weights for distinguishing normal traffic from malicious traffic such as the one generated by phishing and DDoS attacks.

Furthermore, we map the embedding in the last hidden layer of the CNN model to back-end servers, which is then aggregated through LSTM models with the embeddings of other IoT devices for data fusion. Data fusion takes place at the back-end servers hosted in the cloud, allowing us to detect distributed attacks, such as DDoS, including those that are not detected by the CNN model hosted in the IoT device. Since the back-end servers are hosted in a cloud environment, it allows the LSTM inference (ingesting multiple CNN embeddings) to utilize a large pool of distributed computing resources to perform this task.

The second component of our attack detection model consists of a model based on LSTM cells, a variant of recurrent neural networks. The LSTM model is hosted on back-end servers with greater computa-

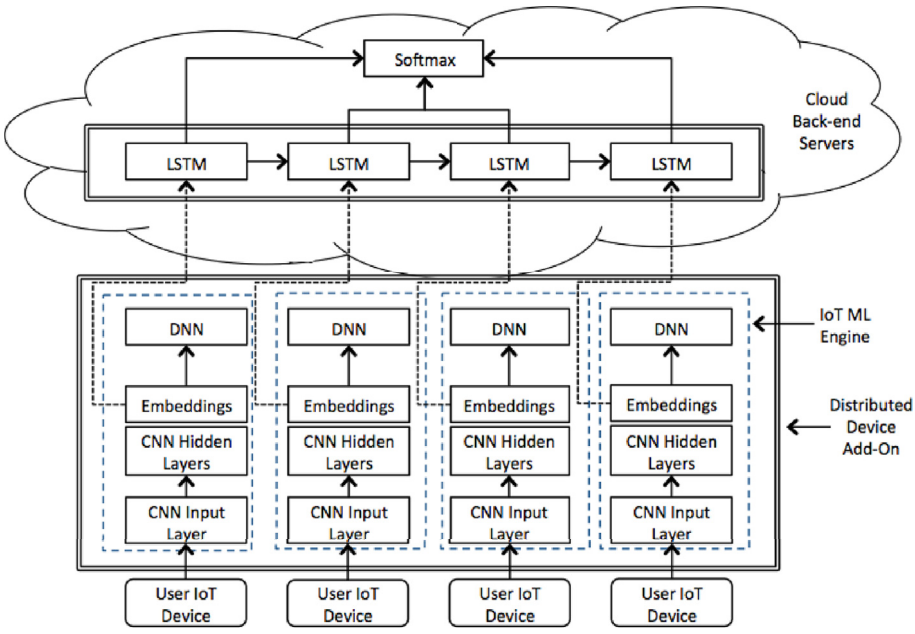

Fig. 6. Distributed Deep Learning Framework: The IoT micro-security add-on is hosted in a ML engine and distributed to a swarm of client IoT devices enabling them to detect anomalous URLs. The micro-security add-on is based on a Convolutional Neural Network (CNN) model composed of convolutional layers and a fully-connected deep neural network (DNN) layer. The LSTM detection model hosted in cloud-based back-end servers is capable of detecting attacks requiring greater computational demands for analysis.

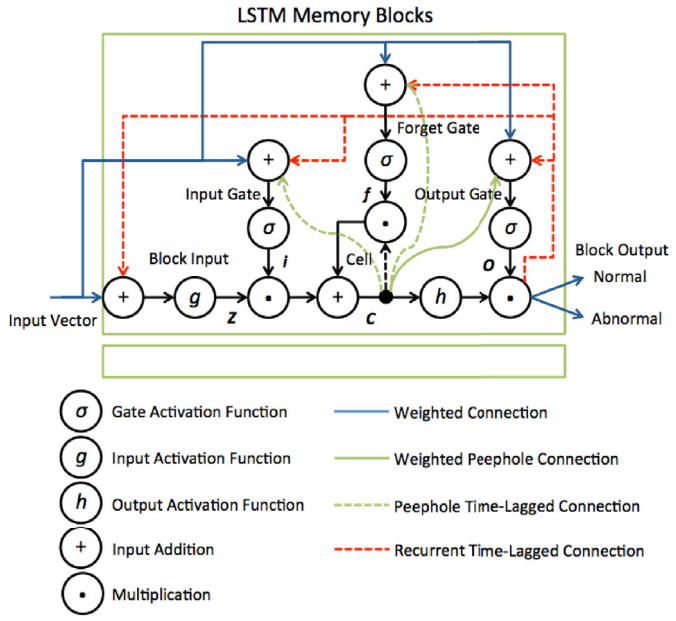

Fig. 7. RNN-LSTM architecture: The model consists on two LSTM blocks with features as input and a classification result as an output.

tional capabilities. Due to these capabilities, the LSTM model focuses on detecting attacks that require complex analysis of the network traffic. The LSTM block is presented in Fig. 7 featuring an input gate, a forget gate, an output gate, the block input, peephole connections, the output activation function, and a single cell. The block's output is recurrently connected to all gates and the block's input. In order to further understand how the information of our input vector is being computed within the LSTM block, we shall explore the calculations for the forward pass, back propagation through time, and the gradients for the weights.

For the forward pass, we define x^t as the input vector at time t where N is the number of LSTM blocks and M is the number of inputs. Furthermore, Z^t denotes the block input, i^t denotes the input gate, f^t denotes the forget gate, c^t denotes the cell, o^t denotes the output gate, and y^t denotes the block's output. In addition, we define the weights as follows:

- Input Weights: $W_z, W_s, W_f, W_o \in \Re^{N\times M}$
- Recurrent Weights: $R_z, R_s, R_f, R_o \in \Re^{N\times M}$
- Peephole Weights: $p_s, p_f, p_o \in \Re^{N}$
- Bias Weights: $b_z, b_s, b_o \in \Re^{N}$

The calculations corresponding to the forward pass are represented in the following equations:

$$Z^{-t} = W_z x^t + R_z y^{t-1} + b_z \tag{1}$$

Block Input

$$Z^t = g(Z^{-t}) \tag{2}$$

$$i^{-t} = W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i \tag{3}$$

Input Gate

$$i^t = \sigma(i^{-t}) \tag{4}$$

$$f^{-t} = W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f \tag{5}$$

Forget Gate

$$f^t = \sigma(f^{-t}) \tag{6}$$

Cell

$$c^t = Z^t \odot i^t + c^{t-1} \odot f^t \tag{7}$$

$$o^{-t} = W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_0 \tag{8}$$

Output Gate

$$o^t = \sigma(o^{-t}) \tag{9}$$

Block Output

$$y^t = h(c^t) \odot o^t \tag{10}$$

Δ^t represents the vector of deltas passed from the previous layer and the point-wise nonlinear activation functions are denoted by σ, g, and h. Furthermore, in our model we used the logistic sigmoid as our gate activation function and the hyperbolic tangent activation function for our block input and output.

Logistic Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{11}$$

Table 1
Relevant datasets used for evaluating intrusion detection systems.

Attribute	Description
H	Statistics summarizing the latest traffic from the packet's host (IP)
MI	Statistics summarizing the latest traffic from the packet's host (IP + MAC)
HH	Statistics summarizing the latest traffic from the packet's host (IP) to the packet's destination host
HH_jit	Statistics summarizing the jitter of the traffic from the packet's host (IP) to the packet's destination host
HpHp	Statistics summarizing the latest traffic from the packet's host + port (IP + PORT) to the packet's destination host + port (IP + PORT)
Time-Frame	How much recent history of stream is captured in the statistics. Represented as L5, L3, L1, L0.2, L0.01
Packet stream statistics	Weight, Mean, STD, Radius, Magnitude, Approximated Covariance, Approximated Correlation

Hyperbolic Tangent

$$g(x) = h(x) = tanh(x) \tag{12}$$

The Deltas inside the LSTM are calculated as per the following equations:

$$\delta y^t = \Delta^t + R_z^T \delta Z^{t+1} + R_i^T \delta i^{t+1} + R_f^T \delta f^{t+1} + R_o^T \delta o^{t+1} \tag{13}$$

$$\delta o^t = \delta y^t \odot h(c^t) \odot \sigma'(o^{-t}) \tag{14}$$

$$\begin{aligned} \delta c^t =& \delta y^t \odot o^t \odot h'(c^t) + p_o \odot \delta o^t + p_i \odot \delta i^{t+1} \\ &+ p_f \odot \delta f^{t+1} + \delta c^{t+1} \odot f^{t+1} \end{aligned} \tag{15}$$

$$\delta f^t = \delta c^t \odot c^{t-1} \odot \sigma'(\overline{f}^t) \tag{16}$$

$$\delta i^t = \delta c^t \odot Z^t \odot \sigma'(\overline{i}^t) \tag{17}$$

$$\delta Z^t = \delta c^t \odot i^t \odot g'(\overline{Z}^t) \tag{18}$$

The deltas from the inputs are passed to the second layer and are computed with the following equations:

$$\delta x^t = W_z^t \delta Z^t + W_i^t \delta i^t + W_f^t \delta f^t + W_o^t \delta o^t \tag{19}$$

Finally, the gradients from the weights are calculated with the equations presented below where ϱ can be any of $\overline{z}, \overline{i}, \overline{f}, \overline{o}$.

$$\delta W_\varrho = \Sigma_{t=0}^{T} \langle \delta \varrho^t, x^t \rangle \quad \delta p_i = \Sigma_{t=0}^{T-1} c^t \odot \delta i^{t+1} \tag{20}$$

$$\delta R_\varrho = \Sigma_{t=0}^{T-1} \langle \delta \varrho^{t+!}, y^t \rangle \quad \delta p_f = \Sigma_{t=0}^{T-1} c^t \odot \delta f^{t+1} \tag{21}$$

$$\delta b_\varrho = \Sigma_{t=0}^{T} \delta \varrho^t \quad \delta p_o = \Sigma_{t=0}^{T} c^t \odot \delta o^t \tag{22}$$

4. Available public datasets

There are a variety of datasets used for training and testing neural network models focused on detecting anomalies. In Table 14, we present some of the frequently used datasets in the literature. For the training, validating, and testing our LSTM back-end model, we selected the "Detection of IoT botnet attacks N_BaIoT" dataset presented in the works of Mirsky et al. (2018) and Meidan et al. (2018). The dataset was generated by collecting network data from 9 IoT devices – namely, Danmini Doorbell, Ecobee Thermostat, Ennino Doorbell, Philips B120N10 Baby Monitor, Provision PT737E Security Camera, Provision 838 Security Camera, Samsung SNH1011 N Webcam, SimpleHome XCS71002 Security Camera, and SimpleHome XCS71003 Security Camera. The network data is summarized in 115 features presenting the information in Table 1. The data is divided into benign traffic and 10 attacks carried by two common IoT botnets – namelly, Mirai and BASHLITE.

Additionally, for training, testing and validating the CNN add-on security model, we created a dataset consisting of phishing and non-phishing URLs. The phishing URLs were obtained from two sources: PhishTank (OpenDNS, 2016) and OpenPhish (OpenPhish). PhishTank offers a community-based phish verification system where users submit suspected phishing sites and other users vote whether a submitted site is a phishing site or not. Data was obtained using the site's API. Additional phishing URLs were collected from OpenPhish. This source maintains an updated list of live phishing URLs collected through a global partner network. These phishing URLs are labeled as abnormal, while non-phishing URLs collected from Curlie (2018) and labeled as normal. Curlie, a successor of directory Mozilla (DMOZ) dmoztools.net (AOL, 2017), contains a categorized directory of non-phishing URLs. In total 4,898 Phishing URLs and 6,157 Non-Phishing URLS were collected for training, validating, and testing our CNN add-on security model.

5. Experimental setup, results, and security analysis

In this section, we present the experimental setup as well as the performance metrics used to measure the accuracy of each of our models. In addition, we present the feature generation process utilized after pre-processing the used datasets. Furthermore, we present in detail our experiments and finally compare the achieved results with the ones presented in other proposals.

5.1. Experimental setup

All the experiments were performed on a Dell PowerEdge R630 running 2 Xeon e5-2670 V3 CPU @ 2.30 GHz with an Ubuntu 16.04 Operating System. The server has a storage capacity 250 GB Seagate (ST9250610NS) Hard Drive and a 10 Ghz connection to an academic network. The developed neural networks were developed using Python 3.5.2, Tensorflow 1.8.0, Scikit-learn 0.19.1, Numpy 1.14.3, Pandas 0.22.0, and Scipy 1.1.0. The data utilized for training, validating, and testing our models was obtained from PhishTank, OpenPhish, Curlie, and the "Detection of IoT botnet attacks N_BaIoT" dataset presented in the works of Mirsky et al. (2018) and Meidan et al. (2018).

5.2. Performance metrics

For evaluating the performance of the IoT micro-security add-on as well as the RNN-LSTM back-end model, we calculated metrics using our model's predicted values and actual values of our data. In Table 2 we present the confusion matrix showing the classification of a model results. These values are calculated by comparing a model's prediction against their true labeled value. As an example, if we have 100 phishing attacks (Actual Positive) and 30 of them where predicted as Phishing, these 30 will be classified as True Positive and the remaining 70 will be classified as False Negative.

In Table 3 we present the metrics used to compare and contrast the performance of different models. With the exception of the F1-Score, calculated using the Precision and Recall values, all metrics presented in Table 2 are calculated using the resulting values of a model. These metrics allow us to compare each of our model's individual performance against other models proposed in the literature. Nevertheless, the aggregation of CNN embeddings to LSTM networks is an approach that differs our work to other proposals.

Table 2
Confusion Matrix table used to describe the performance of a classification model on a set of data.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

5.3. Data collection, pre-processing, and feature generation

The CNN model was trained, validated, and tested with data collected from PhishTank, OpenPhish, and Curlie. In total, 4,898 phishing sites were collected from PhishTank and OpenPhish while 6,157 non-phishing (Normal) URLs were collected from Curlie. The dataset is fairly balanced consisting of 44.3% phishing URLs and 55.7% non-phishing URLs. Each URL is parsed and split into multiple strings. Since the longest URL in our dataset contained 36 strings, this number was set as the limit for each URL sample. Due to the different number of strings per URL, shorter URLs are zero-padded in order to maintain a fixed size length for each phishing and non-phishing sample. In total, 36 features were extracted from each URL sample. Furthermore, each string was encoded into number ranging between 0 and 1 representing a string index. During the training phase, each epoch consisted of batches of 100 URL samples. Each URL sample was resized to 6×6 for the purposes of feeding it into the CNN model. Table 4 presents a summary of the data used in the training, validation, and testing phases of our CNN model.

The LSTM model was trained, validated, and tested using statistics on network data presented in the "Detection of IoT botnet attacks N_BaIoT" dataset. In total, we found 6,506,674 data points divided into 10 malicious subgroups and 555,932 data points presenting benign network traffic statistics were extracted from the dataset as summarized in Table 5. Out of the 115 features, 40 were eliminated based on the work presented by Chawathe (2018). Specifically, we used correlation based feature selection (CFS), which evaluates the contribution of a subset of attributes by considering the individual ability of each feature along with the degree of redundancy between them. The coefficients are used for estimating the correlation between a subset of attributes an a class, as well as inter-correlations between features (Karegowda et al., 2010). Equation of CFS is given as follows:

$$r_{zc} = \frac{k\overline{r_{zi}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}} \tag{23}$$

Where r_{zc} is the correlation between the summed feature subsets and the class variable, k is the number of subset features, r_{zi} is the average of the correlations between the subset features and the class variable, and r_{ii} is the average inter-correlation between subset features (Karegowda et al., 2010).

Table 4
Collected Dataset for training, validating, and testing the CNN model for our web-add on. The dataset is split into 60% (Train), 20% (Validation), and 20% (Test).

Traffic Type	Training	Validation	Testing	Total
Phishing	2,938	979	981	4,898
Normal	3,695	1,232	1,230	6,157
Total	6,633	2,211	2,211	11,055
Batches	67	22	22	111

Out of the 115 features, the top 75 features with the highest merit score were selected. The data comprising malicious attacks is divided in the original dataset in the form of "target device" directories, each of them containing 11 individual files, 10 corresponding to the different attacks carried out by 2 botnets –namely, Mirai and Bashlite, and 1 file containing benign network traffic data statistics. We aggregated the benign traffic data from each of the devices into a single file. The same process was done for each Mirai and Bashlite attack. In total, 11 files were generated which comprise the complete dataset. As for pre-processing, the benign network traffic statistics and a single file containing a specific attack type were aggregated. Afterwards, the data is normalized to values between 0 and 1 and vectorized into a rank 3 matrix [N,1,75] where N is the number of data points in this subset and 75 is the number of selected features. As presented in Table 5, we split the data into 60% training, 20% validation, and 20% testing.

5.4. Experiment results

The developed CNN model consists of 1 input layer, 3 hidden layers, and one fully connected layer. The input layer is fed with a batch size of 100 URL samples. Each URL sample is resized to a 6×6 dimension before being fed to the CNN. The first hidden layer consists of 100 3×3 kernels, the second hidden layer consists of 20 3×3 kernels, and the third hidden layer consists of 30 1×1 kernels. Finally, the final fully connected layer provides 2 outputs used for classifying the URL into Phishing or Normal. All hidden layers use relu as an activation function and the output layer uses softmax. In addition, we used a learning rate of 0.01, a gradient descent optimizer, and a keep rate of 0.60. In Table 8, we present the number of trainable parameters per layer as well as the output shape of each layer in our CNN model. It can be observed that before passing the embedding of the third hidden layer, the parameters were flattened to a 1×120 dimension before being transferred to the fully connected output layer.

From the confusion matrix in Table 6, we can observe the balance of true positive and true negative values as well as the low concentration of false positives and false negatives based on the CNN

Table 3
Metrics calculated for evaluating CNN and LSTM performance: True Positive (TP) corresponds to the actual positive values correctly predicted as positive. True Negative corresponds to the actual negative values correctly predicted as negative. False Positive corresponds to the actual negative values incorrectly predicted as positive. False Negative corresponds to the actual positive values incorrectly predicted as negative. Based on the values in the confusion matrix as presented in Table 2.

Metric	Formula	Description
Precision (PR)	$\frac{TP}{TP+FP}$	Ratio of correctly predicted positive values to the total predicted positive values.
Recall (RC)	$\frac{TP}{TP+FN}$	Recall (Sensitivity): Ratio of correctly predicted positive values to the total actual positive values.
F1-Score	$2 * \frac{PR*RC}{PR+RC}$	Weighted average of Precision and Recall.
True Positive Rate (Sensitivity)	$\frac{TP}{TP+FN}$	Ratio of correctly predicted positive values to the total actual positive values.
True Negative Rate (Specificity)	$\frac{TN}{TN+FP}$	Ratio of correctly predicted negative values to the total actual negative values.
False Positive Rate	$\frac{FP}{FP+TN}$	Ratio of incorrectly predicted positive values to the total actual negative values.
False Negative Rate	$\frac{FN}{FN+TP}$	Ratio of incorrectly predicted negative values to the total actual positive values.
Accuracy	$\frac{TP+TN}{Total}$	Ratio of correctly predicted values to the total values.

Table 5
Dataset usage for LSTM back-end.

Subset	Benign	Malicious	Training	Validation	Testing
gafgyt combo	555,932	515,156	771,182	85,688	214,218
gafgyt junk	555,932	261,789	588,758	65,418	163,545
gafgyt scan	555,932	255,111	583,950	64,884	162,209
gafgyt tcp	555,932	859,850	1,019,261	113,264	283,157
gafgyt udp	555,932	946,366	1,081,653	120,185	300,460
mirai_ack	555,932	643,821	863,821	95,981	239,951
mirai_scan	555,932	537,979	787,615	87,513	218,783
mirai_syn	555,932	733,299	928,245	103,139	257,847
mirai_udp	555,932	1,229,999	1,285,868	142,876	357,187
mirai_udp plain	555,932	523,304	777,048	86,340	215,848
Sum	555,932	6,506,674	5,085,074	565,010	1,412,522

Table 6
CNN add-on security model confusion matrix. Based on the values presented in the confusion matrix we calculate the Precision: 0.9348, Recall: 0.9367, and Accuracy: 0.9430.

	Predicted Positive	Predicted Negative
Actual Positive	919	62
Actual Negative	64	1166

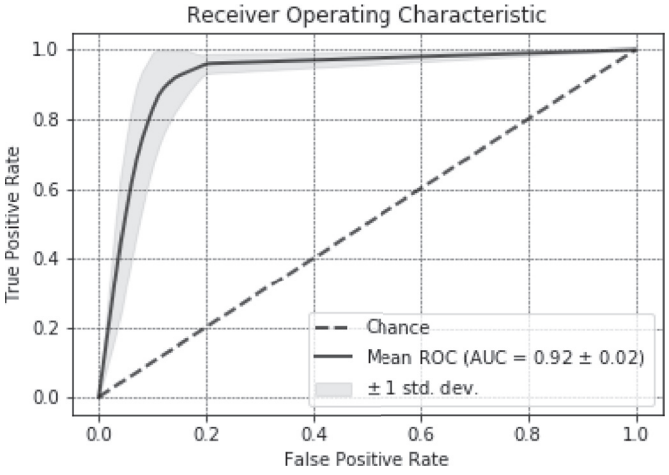

Fig. 8. CNN add-on security model ROC: Based on the calculated TPR,FPR, and AUC a value of 0.919 was computed.

model predictions during the testing phase. The accuracy we calculated during the training phase was 98% and 94.30% in the testing phase. The F1-Score calculated based on the precision and recall computed to 0.9358. In addition, our model scored a true positive rate of 0.9367 and a false positive rate of 0.0520. The Receiver Operating Characteristic in Fig. 8 shows the recall and precision relationship as the threshold is varied for identifying a positive value in our model.

The model was trained, evaluated, and tested in the Dell PowerEdge R630 server. Afterwards, we transferred the model for inference to an Nvidia Jetson TX2 for assessing the real-time inference performance in an IoT device. This was achieved by freezing the TensorFlow graph, optimizing the model for ARM, loading the model, and running inference in the Jetson TX2. To evaluate the performance, we used the test dataset as an input for the inference model. We tested the model using different sizes of inputs to observe the number of inferred frames per second. Table 7 shows the performance of the inference process of our model in the Nvidia Jetson TX2.

The LSTM model was trained, validated, and tested using the data presented in Table 5. The model consists of 2 hidden layers. We used

Table 7
Embedded-CNN performance in Nvidia Jetson TX2.

Input Size	Average Frames Per Second	Average FPS FP16 Quantized
6 × 6	13.1	32.4
28 × 28	12.8	31.5
32 × 32	12.7	31.4
256 × 256	8.8	22.3
608 × 608	2.8	8.1

Table 8
Number of computed outputs and parameters per layer in the CNN model.

Layer Type	Output Shape	Number of Parameters
Input Layer	(6,6,1)	0
Conv. 2D Hidden Layer 1	(4,4,100)	100
Conv. 2D Hidden Layer 2	(2,2,20)	1820
Conv. 2D Hidden Layer 3	(2,2,30)	630
Flatten	120	0
Dense Layer	2	242
Total Trainable		2792

Table 9
RNN-LSTM accuracy by dataset subset in mirai and gafgyt botnets.

Subset	Mean Accuracy	Standard Deviation
gafgyt_combo	97.17%	(+- 2.17%)
gafgyt_junk	94.70%	(+- 4.22%)
gafgyt_scan	92.70%	(+- 3.60%)
gafgyt_tcp	73.21%	(+- 3.95%)
gafgyt_udp	74.70%	(+- 5.20%)
mirai_ack	97.16%	(+- 1.08%)
mirai_scan	86.80%	(+- 3.88%)
mirai_syn	90.32%	(+- 2.73%)
mirai_udp	97.84%	(+- 1.21%)
mirai_udp plain	95.98%	(+- 1.00%)
all malicious	94.80%	(+- 0.79%)

tanh as our activation function, a learning rate of 0.01, adam optimizer, binary cross entropy as our loss function in each hidden layer and softmax in our output layer, a keep rate of 0.50, 30 epochs for training, and a batch size of 100,000. We trained, validated, and tested the model 10 times. One for each attack where we merged the benign traffic along with the malicious traffic representing the attack of interest. In Table 9 we present the calculated accuracy and the standard deviation on the validation training set. In addition, we present in Table 12 the performance metrics calculated on the testing set along with the confusion matrix values for calculating these metrics. The metrics are based on the formulas presented in Table 2. In Table 11, we present

Table 10
Accuracy (Testing set) Metrics Summary: Obtained from the CNN add-on security model and the RNN-LSTM back-end security model.

Parameter	CNN	LSTM Mirai UDP
Precision	0.9348	0.9781
Recall	0.9367	0.9500
F1-Score	0.9358	0.9625
FPR	0.0520	0.0001
TPR	0.9367	0.9999
AUC	0.919	0.9999
Accuracy	0.9430	0.9784

Table 11
Number of computed outputs and parameters per layer in the LSTM model.

Layer Type	Output Shape	Number of Parameters
LSTM 0	Multiple	11,648
LSTM 1	Multiple	6384
Dense 0	Multiple	812
Dense 1	Multiple	29

detailed information about the number of parameters calculated per layer.

From the results presented in Table 12, we can observe the balance of the malicious (abnormal) and the normal requests as well as the low concentration of false positives and false negatives predicted by the LSTM model in the different experiments. Due to the number of experiments performed on each attack, we are unable to present all of the ROC graphs presenting the recall and precision relationship as the threshold is varied for identifying a positive value in our LSTM model. Nevertheless, we present the resulting ROC on one experiment using benign network traffic statistics and mirai udp malicious traffic. The model's performance metrics for this specific experiment are presented in Table 10. In Fig. 9, we present the recall and precision relationship, as well as the AUC.

In summary, for the CNN model hosted in the IoT micro-security add-on, we collected data presented in Table 4 to train, evaluate, and test the model. To test the effectiveness of our model, 5-fold cross-validation was used to split the URL based dataset. The choice of 5-fold cross-validation is due to the hyper-parameters selected for this model; specifically, the use of 3 layers for our CNN model. Based on the performance presented in Table 6, we analyze our model's phishing detection capabilities of the ML engines. In addition, we evaluated the inference performance of the model in the Nvidia Jetson TX2 as presented in Table 7.

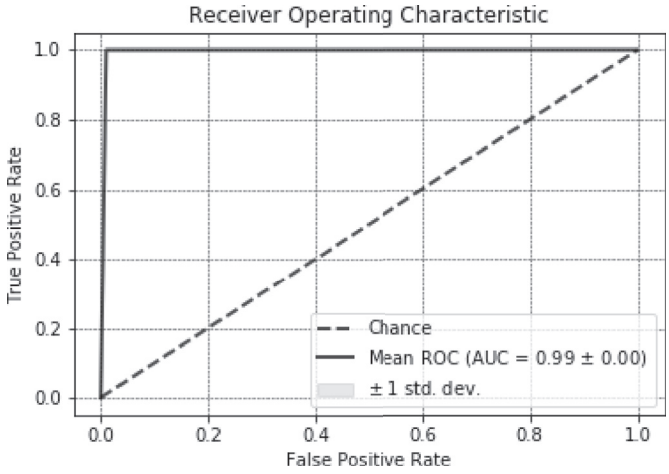

Fig. 9. LSTM security model ROC: Based on the calculated TPR, FPR, and AUC a value of 0.99 was computed.

In addition we generated a LSTM model running in the back-end for detecting attacks generated by the Mirai and Bashlate botnets. The analysis of these attacks require higher memory and computational resources for its analysis. For generating this model, we collected the data presented in Table 5 from the "Detection of IoT botnet attacks N_BaIoT" dataset presented in the works of Mirsky et al. (2018) and Meidan et al. (2018) to train, evaluate, and test the model. The "Detection of IoT botnet attacks N_BaIoT" dataset was split based on the attack types. We used 10-fold cross-validation for each subset generated by aggregating benign network traffic statistics with malicious network traffic statistics from a specific attack. The reduced number of features enabled the model to learn from the selected parameters in the dataset. As a result of this selection, the use of 10-fold cross validation allowed us to evaluate our model's effectiveness with greater precision. The performance metrics in Table 12 show the model's capability to detect botnet attacks using statistics generated from the network traffic IoT devices.

6. Conclusion and future work

In this paper, we presented an IoT micro-security add-on hosted in the device using a CNN model for detecting URL based attacks directed to a client's IoT devices. The add-on works in conjunction with a RNN-LSTM model hosted at the back-end servers for detecting botnet attacks in IoT devices. Furthermore, the embedding on the last hidden layer of the CNN model is transferred to the back-end servers, which is aggregated through LSTM models with the embeddings of other clients' mobile devices and IoT devices for data fusion in order to detect distributed attacks. In addition, we briefly reviewed the publicly available datasets that have been used in the literature and presented the ones

Table 12
This table presents the confusion matrix values along with the performance metrics results for each tested attack. For each result, the model was trained, validated, and tested using benign data along with the malicious network traffic data of the corresponding attack.

Attack	Precision	Recall	F1-Score	TPR	TNR	FPR	FNR	Accuracy
gafgyt combo	0.9560	0.9991	0.9770	0.9991	0.9573	0.0427	0.0009	0.9774
gafgyt_junk	0.9999	0.9984	0.9992	0.9984	1.0000	0.0000	0.0016	0.9995
gafgyt_scan	0.7722	0.9992	0.8711	0.9992	0.8654	0.1346	0.0008	0.9073
gafgyt_tcp	0.9461	0.9989	0.9718	0.9989	0.9123	0.0877	0.0011	0.9649
gafgyt_udp	0.9462	0.9991	0.9719	0.9991	0.9033	0.0967	0.0009	0.9636
mirai_ack	0.9471	1.0000	0.9728	1.0000	0.9352	0.0648	0.0000	0.9700
mirai_scan	0.9459	0.9999	0.9722	0.9999	0.9443	0.0557	0.0001	0.9717
mirai_syn	0.9399	1.0000	0.9690	1.0000	0.9155	0.0845	0.0000	0.9636
mirai_udp	0.9704	0.9999	0.9849	0.9999	0.9329	0.0671	0.0001	0.9790
mirai_udp plain	0.9931	0.9999	0.9965	0.9999	0.9935	0.0065	0.0001	0.9966

Table 13
Comparison of proposals presenting solutions for detecting web attacks. The same metrics are considered as a means of comparison between different proposals. The proposals presented were not evaluated on the same dataset.

Model	Classification/Evaluation	Side	Precision	Recall	F1 Score	Accuracy	FPR	Year
Phishing-Alarm (Mao et al., 2017)	Page Component Similarity Algorithm	Client	1.0	0.979	0.990	–	0.020	2017
Off-the-Hook (Eng) (Marchal et al., 2017)	Gradient Boosting	Client	0.980	0.952	0.965	0.999	0.0002	2017
Off-the-Hook (Marchal et al., 2017)	Gradient Boosting	Client	0.975	0.951	0.962	0.999	0.0008	2017
Ma et al. (Ma et al., 2009a)	Support Vector Machine Bayesian	Back-end	0.998	0.924	0.959	0.955	0.001	2017
Marchal et al. (Marchal et al., 2016)	Gradient Boosting	Client	0.956	0.958	0.957	–	0.0005	2016
Jain et al. (Jain and Gupta, 2016)	Heuristics-Based	Client	0.993	0.861	0.922	0.894	0.015	2016
Varshney et al. (Varshney et al., 2016)	Search Engine Based Antiphishing Technique	Client	0.723	0.995	0.837	0.936	0.076	2016
PhishShield (Rao and Ali, 2015)	Heuristics-Based	Client	0.999	0.971	0.984	0.966	0.0004	2015
Ramesh et al. (Ramesh et al., 2014)	Time Frequency - Inverse Document Frequency (TF-IDF) Support Vector Machine Random Forest C4.5	Client	0.998	0.996	0.996	0.966	0.005	2014
PhishStorm (Marchal et al., 2014)	Random Tree Logistic Model Tree Jrip	Client	0.984	0.913	0.947	0.949	0.014	2014
Chen et al. (Chen et al., 2014)	Logistic Regression	Back-end	0.992	1.0	0.995	0.994	0.007	2014
Kausar et al. (Kausar et al., 2014)	Heuristics-Based Naïve Bayes	Client	0.888	0.887	0.887	0.875	0.112	2014
Thomas et al. (Thomas et al., 2011)	Logistic Regression Support Vector Machine Logistic Regression TF-IDF	Back-end	0.961	0.734	0.832	0.866	0.003	2011
Cantina+ (Xiang et al., 2011)	Bayesian-based J48 Decision Tree AdaBoost	Client	0.964	0.955	0.959	0.969	0.013	2011
Whittaker et al. (Whittaker et al., 2010)	Logistic Regression	Back-end	0.989	0.915	0.950	0.999	0.0001	2010
PhishNet (Prakash et al., 2010)	Heuristics-Based	Client	–	–	–	–	–	2010
Ma et al. (Ma et al., 2009b)	Confidence Weighted	Back-end	–	–	–	0.99	–	2009
Xiang et al. (Xiang and Hong, 2009)	Time Frequency - Inverse Document Frequency (TF-IDF)	Client	0.957	0.9	0.927	0.955	0.019	2009
Cantina (Zhang et al., 2007)	Time Frequency - Inverse Document Frequency (TF-IDF)	Client	0.212	0.89	0.342	0.969	0.03	2007
Garera et al. (Garera et al., 2007)	Logistic Regression	Client	–	–	–	0.973	0.012	2007
Fu et al. (Fu et al., 2006)	Earth Mover's Distance	Back-end	0.998	0.888	0.94	–	–	2006

Table 14
Relevant datasets used for evaluating intrusion detection systems.

Dataset	Year	Reference	Description
KDD-CUP	1999	Tavallaee et al. (2009)	This dataset is built based on the DARPA'98 IDS evaluation program. The KDD training dataset consists of 4,900,000 single connection vectors containing 41 features each and is labeled as normal or attack. The features can be classified into three groups: basic features, traffic features, and content features.
DARPA 1999	1999	Lippmann et al. (2000)	The dataset was generated by the Defense Advanced Research Projects Agency (DARPA) for the off-line intrusion detection evaluation in 1999. It consists of more than 200 instances of 58 attack types launched against Unix and Windows NT users.
Kyoto University Benchmark 2006	2006	Song et al. (2006)	The Kyoto University Benchmark 2006 Dataset was created using honeypots. Due to this, there is no manual process for labeling traffic. Normal traffic is simulated between attacks but only in the form of DNS and mail traffic that lead to 0 false positive rates.
NSL-KDD 2009	2009	Tavallaee et al. (2009)	The NSL-KDD 2009 dataset is the improved version of the KDD-CUP dataset. This dataset filtered duplicate instances that produced a biased classification. The first 41 attributes can be categorized into 4 different classes: basic features, content features, traffic features, and host features.
CSIC 2010	2010	Giménez et al. (2010)	The CSIC 2010 Dataset was created by the Information Security Institute from the Spanish Research National Council. The dataset contains thousands of HTTP requests generated by targeting an e-commerce web application. The labeled dataset contains 36,000 sessions containing normal requests and 25,000 sessions containing malicious requests. Each session comprises one or more requests. The 36,000 sessions of normal behaviour comprises 104,000 requests. On the other hand, the 25,000 sessions of abnormal behaviour consist of 119,585 requests.

used for training, validating, and testing our models. Then, we presented the experimental results and performance of the CNN model and the RNN-LSTM model, hosted respectively at the IoT devices and the back-end servers. The results revealed that the IoT micro-security addon is capable of detecting phishing attacks with an accuracy of 94.3% and a F-1 score of 93.58%. Moreover, the back-end LSTM model for detecting Botnet attacks achieved an accuracy of 94.80% when using all malicious and benign data points in the dataset. Results for detecting

Table 15
Comparison of proposals presenting solutions for detecting web attacks. The proposals presented in this table were evaluated using the CSIC 2010 dataset.

Model	Classification/Evaluation	Precision	Recall	F1 Score	Accuracy	FPR	Year
Vartouni et al. (2018)	Isolation Forest	0.8029	0.8834	0.8412	0.8832	–	2018
Alrawashdeh and Purdy (2017)	DBN	–	–	–	0.951	–	2017
Liang et al. (2017)	GRU	–	–	–	0.9788	–	2017
Liang et al. (2017)	RNN-LSTM	–	–	–	0.9842	–	2017
Kozik et al. (2016)	ELM	–	–	–	0.9500	0.0079	2016
Reddy et al. (2016)	SVM	0.998	0.998	0.998	0.999	–	2016
Choraś and Kozik (2015)	Graph-Based	–	–	–	0.9446	0.0434	2015
Kozik et al. (2014)	Graph-Based & Segmentation	–	–	–	0.9446	0.0434	2014

specific attacks when the model is trained using benign data and malicious data corresponding to a specific attack of our interest is presented Table 12. In this table, we can observe that the model achieves a greater accuracy for detecting a specific attack. This occurs due to a lower variability between malicious request of the same type and a contrasting difference with normal traffic that the model is able to capture. A comparative summary of proposed models, presented by other authors, for detecting attacks is also presented in Tables 13–15. After comparing the works covered in the literature, we observed that the proposed models presented solutions for detecting attacks either at the back-end or at the client's side. A key advantage of our proposed approach is the capability to perform different levels of detection at client devices and at the back-end servers; thus, leveraging the distributed computational capabilities of client IoT devices and computationally robust servers.

Future work includes extending the proposed approach to detect emerging attacks targeting IoT devices and systems, including those using encrypted traffic to obfuscate or evade detection.

CRediT authorship contribution statement

Gonzalo De La Torre Parra: Conceptualization, Data curation, Formal analysis, Methodology, Investigation, Validation, Writing - original draft. **Paul Rad:** Conceptualization, Methodology, Validation, Supervision, Writing - review & editing, Funding acquisition. **Kim-Kwang Raymond Choo:** Conceptualization, Methodology, Supervision, Writing - review & editing, Funding acquisition. **Nicole Beebe:** Methodology, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that there is no conflict of interest associated with this paper.

Acknowledgements

This project and the preparation of this publication were funded in part by monies provided by CPS Energy through an agreement with The University of Texas at San Antonio. The first author also acknowledges the support provided by a ConTex fellowship from the University of Texas System and the Consejo Nacional de Ciencia y Tecnologia de Mexico (CONACyT), and the third author is also supported by the Cloud Technology Endowed Professorship.

References

Abdelhamid, N., Thabtah, F., Abdel-jaber, H., 2017. Phishing detection: a recent intelligent machine learning comparison based on models content and features. In: Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on. IEEE, pp. 72–77.

Alrawashdeh, K., Purdy, C., 2017. Reducing calculation requirements in fpga implementation of deep learning algorithms for online anomaly intrusion detection. In: 2017 IEEE National Aerospace and Electronics Conference (NAECON). IEEE, pp. 57–62.

AOL, 2017. dmztools. https://dmoztools.net/.

Casino, F., Choo, K.R., Patsakis, C., 2019. Hedge: Efficient Traffic Classification of Encrypted and Compressed Packets. IEEE Trans. Inf. Forensic. Sec. 14, 2916–2926.

Chawathe, S.S., 2018. Monitoring iot networks for botnet activity. In: 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). IEEE, pp. 1–8.

Chen, T.-C., Stepan, T., Dick, S., Miller, J., 2014. An anti-phishing system employing diffused information. ACM Trans. Inf. Syst. Secur. 16 (4), 16.

Cheng, R., Xu, R., Tang, X., Sheng, V.S., Cai, C., 2018. An abnormal network flow feature sequence prediction approach for ddos attacks detection in big data environment. Comput. Mater. Continua (CMC) 55 (1), 95.

Chora, M., Kozik, R., 2015. Machine learning techniques applied to detect cyber attacks on web applications. Log. J. IGPL 23 (1), 45–56.

Cisco, 2018. Cisco 2018 Annual Security Report.

Curlie, 2018. Curlie. https://curlie.org.

da Silva Cardoso, A.M., Lopes, R.F., Teles, A.S., Magalhes, F.B.V., 2018. Real-time ddos detection based on complex event processing for iot. In: 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI). IEEE, pp. 273–274.

De Donno, M., Dragoni, N., Giaretta, A., Spognardi, A., 2018. Ddos-capable Iot Malwares: Comparative Analysis and Mirai Investigation. Security and Communication Networks. 2018.

Diro, A.A., Chilamkurti, N., 2018. Distributed attack detection scheme using deep learning approach for internet of things. Future Generat. Comput. Syst. 82, 761–768.

Doshi, R., Apthorpe, N., Feamster, N., 2018. Machine learning ddos detection for consumer internet of things devices. In: 2018 IEEE Security and Privacy Workshops (SPW). IEEE, pp. 29–35.

Fu, A.Y., Wenyin, L., Deng, X., 2006. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). IEEE Trans. Dependable Secure Comput. 3 (4), 301–311.

Garera, S., Provos, N., Chew, M., Rubin, A.D., 2007. A framework for detection and measurement of phishing attacks. In: Proceedings of the 2007 ACM Workshop on Recurring Malcode. ACM, pp. 1–8.

Gimnez, C.T., Villegas, A.P., Maran, G.., 2010. Http Data Set Csic 2010.

Habeeb, R.A., Nasaruddin, F., Gani, A., Amanullah, M.A., Hashem, I.A.T., Ahmed, E., Imran, M., 2019. Clustering-based real-time anomaly detectiona breakthrough in big data technologies. Trans. Emerg. Telecommun. Technol. e3647 vol. 0, no. 0.

Jain, A.K., Gupta, B., 2016. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP J. Inf. Secur. 2016 (1), 9.

Jing, X., Yan, Z., Jiang, X., Pedrycz, W., 2019. Network traffic fusion and analysis against ddos flooding attacks with a novel reversible sketch. Inf. Fusion 51, 100–113.

Karegowda, A.G., Manjunath, A., Jayaram, M., 2010. Comparative study of attribute selection using gain ratio and correlation based feature selection. Int. J. Inf. Technol. Knowl. Manag. 2 (2), 271–277.

Kausar, F., Al-Otaibi, B., Al-Qadi, A., Al-Dossari, N., 2014. Hybrid client side phishing websites detection approach. Int. J. Adv. Comput. Sci. Appl. 5 (7), 132–140.

Kim, J., Kim, J., Thu, H.L.T., Kim, H., 2016. Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). IEEE, pp. 1–5.

Kozik, R., Chora, M., Houbowicz, W., Renk, R., 2016. Extreme learning machines for web layer anomaly detection. In: International Conference on Image Processing and Communications. Springer, pp. 226–233.

Kozik, R., Chora, M., Renk, R., Houbowicz, W., 2014. Modelling http requests with regular expressions for detection of cyber attacks targeted at web applications. In: International Joint Conference SOCO14-CISIS14-ICEUTE14. Springer, pp. 527–535.

Liang, J., Zhao, W., Ye, W., 2017. Anomaly-based web attack detection: a deep learning approach. In: Proceedings of the 2017 VI International Conference on Network, Communication and Computing. ACM, pp. 80–85.

Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K., 2000. The 1999 darpa off-line intrusion detection evaluation. Comput. Network. 34 (4), 579–595.

Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009a. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 1245–1254.

Ma, J., Saul, L.K., Savage, S., Voelker, G.M., 2009b. Identifying suspicious urls: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, pp. 681–688.

Mao, J., Tian, W., Li, P., Wei, T., Liang, Z., 2017. Phishing-alarm: robust and efficient phishing detection via page component similarity. IEEE Access 5, 17020–17030.
Marchal, S., Armano, G., Grndahl, T., Saari, K., Singh, N., Asokan, N., 2017. Off-the-hook: an efficient and useable client-side phishing prevention application. IEEE Trans. Comput. 66 (10), 1717–1733.
Marchal, S., Franois, J., State, R., Engel, T., 2014. Phishstorm: detecting phishing with streaming analytics. IEEE Trans. Netw. Serv. Manag. 11 (4), 458–471.
Marchal, S., Saari, K., Singh, N., Asokan, N., 2016. Know your phish: novel techniques for detecting phishing sites and their targets. In: Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on. IEEE, pp. 323–333.
Marzano, A., Alexander, D., Fonseca, O., Fazzion, E., Hoepers, C., Steding-Jessen, K., Chaves, M.H., Cunha, ., Guedes, D., Meira, W., 2018. The evolution of bashlite and mirai iot botnets. In: 2018 IEEE Symposium on Computers and Communications (ISCC). IEEE, pp. 813–818.
McDermott, C.D., Majdani, F., Petrovski, A.V., 2018. Botnet detection in the internet of things using deep learning approaches. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.
Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y., 2018. N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders. IEEE Pervas. Comput. 17 (3), 12–22.
Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A., 2018. Kitsune: an Ensemble of Autoencoders for Online Network Intrusion Detection. arXiv preprint 1802.09089.
Moustafa, N., Choo, K.-K. R., Radwan, I., Camtepe, S., 2019. Outlier dirichlet mixture mechanism: adversarial statistical learning for anomaly detection in the fog. IEEE Trans. Inf. Forensics Secur. 14 (8), 1975–1987.
O'Gorman, B., Wueest, C., O'Brien, D., Cleary, G., Lau, H., Power, J.-P., Corpin, M., Cox, O., Wood, P., Wallace, S., 2019. Symantec Internet Security Threat Report 2019, vol. XXIV.
OpenDNS, L., 2016. Phishtank: an anti-phishing site. Online: https://www.phishtank.com.
OpenPhish, OpenPhish. https://openphish.com.
Pour, M.S., Bou-Harb, E., Varma, K., Neshenko, N., Pados, D.A., Choo, K.-K. R., 2019. Comprehending the iot cyber threat landscape: a data dimensionality reduction technique to infer and characterize internet-scale iot probing campaigns. Digit. Invest. 28 (Suppl.), S40–S49.
Prakash, P., Kumar, M., Kompella, R.R., Gupta, M., 2010. Phishnet: predictive blacklisting to detect phishing attacks. In: INFOCOM, 2010 Proceedings IEEE. IEEE, pp. 1–5.
Praseed, A., Thilagam, P.S., 2018. Ddos attacks at the application layer: challenges and research perspectives for safeguarding web applications. IEEE Commun. Surv. Tutor. 21 (1), 661–685.
Radford, B.J., Richardson, B.D., Davis, S.E., 2018. Sequence Aggregation Rules for Anomaly Detection in Computer Network Traffic. arXiv preprint 1805.03735.
Ramesh, G., Krishnamurthi, I., Kumar, K.S.S., 2014. An efficacious method for detecting phishing webpages through target domain identification. Decis. Support Syst. 61, 12–22.
Rao, R.S., Ali, S.T., 2015. Phishshield: a desktop application to detect phishing webpages through heuristic approach. Procedia Comput. Sci. 54, 147–156.
Reddy, R.R., Ramadevi, Y., Sunitha, K.N., 2016. Effective discriminant function for intrusion detection using svm. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, pp. 1148–1153.
Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q., 2018. A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. 2 (1), 41–50.
Singh, M., Singh, M., Kaur, S., 2019. Issues and Challenges in Dns Based Botnet Detection: A Survey. Computers & Security.
Song, J., Takakura, H., Okabe, Y., 2006. Description of Kyoto University Benchmark Data. Academic Center for Computing and Media Studies (ACCMS), Kyoto University.
Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set. In: Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on. IEEE, pp. 1–6.
Thomas, K., Grier, C., Ma, J., Paxson, V., Song, D., 2011. Design and evaluation of a real-time url spam filtering service. In: Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, pp. 447–462.
Varshney, G., Misra, M., Atrey, P.K., 2016. A phish detector using lightweight search features. Comput. Secur. 62, 213–228.
Vartouni, A.M., Kashi, S.S., Teshnehlab, M., 2018. An anomaly detection method to detect web attacks using stacked auto-encoder. In: 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). IEEE, pp. 131–134.
Vayansky, Ike, Kumar, Satish, 2018. Phishing–challenges and solutions. Comput. Fraud Secur. 1, 15–20 In press.
Whittaker, C., Ryner, B., Nazif, M., 2010. Large-scale automatic classification of phishing. In: NDSS, vol. 10, p. 2010.
Xiang, G., Hong, J., Rose, C.P., Cranor, L., 2011. Cantina: a feature-rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur. 14 (2), 21.
Xiang, G., Hong, J.I., 2009. A hybrid phish detection approach by identity discovery and keywords retrieval. In: Proceedings of the 18th International Conference on World Wide Web. ACM, pp. 571–580.
Yusof, A.R., Udzir, N.I., Selamat, A., 2019. Systematic literature review and taxonomy for ddos attack detection and prediction. Int. J. Dist Educ. Technol. 1 (3), 292–315.
Zhang, Y., Hong, J.I., Cranor, L.F., 2007. Cantina: a content-based approach to detecting phishing web sites. In: Proceedings of the 16th International Conference on World Wide Web. ACM, pp. 639–648.

Gonzalo De La Torre Parra received the B.S. in Electrical Engineering from Texas A&M University-Kingsville, USA, in 2009 and the M.S. in Electrical Engineering (Telecommunications Concentration) from The University of Texas at San Antonio, USA, in 2015. He is currently pursuing the Ph.D. in Electrical Engineering at The University of Texas at San Antonio. His current research is focused on the development of Cybersecurity solutions based on Deep Learning Algorithms. From 2015 to 2017, he has been a developer of Chameleon Cloud, an NSF sponsored cloud infrastructure, has held the Co-Chair position of OpenStack's Cloud Application Hack Work Group, and manages research projects focused on cloud, networks, and security at the Open Cloud Institute. Mr. De La Torre Parra's awards and honors include the 2nd Place Award on SHPE's Extreme Engineering National STEM Competition (2013), the 2nd Place Award on HENAAC's XIV National STEM Competition (2013), the San Antonio Mexican Foundation for Education (SAMFE) award (2013), CONACYT's Master's Degree Fellowship (2013), NSF-Open Cloud Institute Fellowship (2016), and CONACYT's Ph.D. Degree Fellowship (2017).

Paul Rad received the Ph.D. degree in electrical and computer engineering on cyber analytics from the University of Texas at San Antonio, San Antonio, TX, USA. He is a co-founder and Associate Director of the Open Cloud Institute (OCI), and an Associate Professor with the information systems and cyber security (ISCS) from University of Texas at San Antonio. He has received his first B.S. degree from Sharif University of Technology in Computer Engineering in 1994, his 1st master in artificial intelligence from the Tehran Polytechnic, the 2nd master's in computer science from the University of Texas at San Antonio (Magna Cum Laude) in December 1999, and the Ph.D. in electrical and computer engineering from the University of Texas at San Antonio. He was a recipient of the Most Outstanding Graduate Student in the College of Engineering, 2016, Achieving Rackspace Innovation Mentor Program Award" for establishing Rackspace patent community board structure and mentoring employees, 2012, Achieving Dell Corporation Company Excellence (ACE) Award in Austin for exceptional performance and innovative product research and development contributions, 2007, and Dell Inventor Milestone Award, Top 3 Dell Inventor of the year, 2005. He holds 15 U.S. patents on cyber infrastructure, cloud computing, and big data analytics with over 300 product citations by top fortune 500 leading technology companies such as Amazon, Microsoft, IBM, Cisco, Amazon Technologies, HP, and VMware. He has advised over 200 companies on cloud computing and data analytics with over 50 keynote presentations. He serves on the advisory board for several startups, high performance cloud group chair at the Cloud Advisory Council (CAC), OpenStack Foundation Member, the \#1 open source cloud software, San Antonio Tech Bloc Founding Member, Children's Hospital of San Antonio Foundation board member.

Kim-Kwang Raymond Choo received the Ph.D. in Information Security in 2006 from Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, Outstanding Associate Editor of 2018 for IEEE Access, British Computer Society's 2019 Wilkes Award Runner-up, 2019 EURASIP JWCN Best Paper Award, Korea Information Processing Society's JIPS Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, Inscrypt 2019 Best Student Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, an IEEE Senior Member, and Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.

Nicole Beebe is the Melvin Lachman Distinguished Professor in Entrepreneurship, Professor of Cybersecurity, Department Chair of Information Systems and Cyber Security, and Director of the Cyber Center for Security and Analytics at The University of Texas at San Antonio (UTSA). UTSA is a National Center of Academic Excellence in Information Assurance and Cyber Defense for both education and research. Dr. Beebe received her Ph.D. in Information Technology from UTSA, an M.S. in Criminal Justice from Georgia State University, and a B.S. in Electrical Engineering from Michigan Technological University. She has over twenty years of experience in information security and digital forensics, from both the commercial and government sectors, is a Certified Information Systems Security Professional (CISSP), and has held three professional certifications in digital forensics. She has published several journal articles related to information security and digital forensics in Decision Support Systems (DSS), IEEE Transactions of Information Forensics and Security, Digital Investigation, and many other journals. Her research interests include digital forensics, cybersecurity, and data analytics.