A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets

Article · May 2021

CITATIONS
66

READS
5,335

1 author:

Nour Moustafa
UNSW Canberra
247 PUBLICATIONS **13,970** CITATIONS

SEE PROFILE

A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets

Nour Moustafa

School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2612, Australia. E-mail: nour.moustafa@unsw.edu.au

Abstract

While there has been a significant interest in understanding the cyber threat landscape of Internet of Things (IoT) networks, and the design of Artificial Intelligence (AI)-based security approaches, there is a lack of distributed architecture led to generating heterogeneous datasets that contain the actual behaviors of real-world IoT networks and complex cyber threat scenarios to evaluate the credibility of the new systems. This paper presents a novel testbed architecture of IoT network which can be used to evaluate Artificial Intelligence (AI)-based security applications. The platform NSX vCloud NFV was employed to facilitate the execution of Software-Defined Network (SDN), Network Function Virtualization (NFV) and Service Orchestration (SO) to offer dynamic testbed networks, which allow the interaction of edge, fog and cloud tiers. While deploying the architecture, real-world normal and attack scenarios are executed to collect labeled datasets. The generated datasets are named 'TON_IoT', as they comprise heterogeneous data sources collected from telemetry datasets of IoT services, Windows and Linux-based datasets, and datasets of network traffic. The TON_IoT network dataset is validated using four machine learning-based intrusion detection algorithms of Gradient Boosting Machine, Random Forest, Naive Bayes, and Deep Neural Networks, revealing a high performance of detection accuracy using the set of training and testing. A comparative summary of the TON_IoT network dataset and other competing network datasets demonstrates its diverse legitimate and anomalous patterns that can be used to better validate new AI-based security solutions. The architecture and datasets can be publicly accessed from [1].

Keywords: Smart cities, Network datasets, Cybersecurity applications, Machine Learning, Edge, Software-Defined Network (SDN), Network Function

1. Introduction

Smart cities aim to integrate Information and Communication Technology (ICT), edge computing, Internet of things (IoT) devices and blockchain technologies to enhance the efficiency of city operations and services and offer automated services to end-users and organisations. One of the key challenges of smart cities is the design of a secure and distributed architectures that effectively embrace all of these technologies. The development of sustainable smart cities uses smart ICTs and IoT systems to offer automated services to end-users and organisations [2]. IoT systems are becoming a norm in our homes (e.g. smart homes), offices (e.g. smart offices, smart buildings, smart campuses, and smart facilities), and cities (e.g. smart cities) as well as other applications such as smart healthcare systems and industry 4.0 applications (e.g. industrial IoT – IIoT) [3]. Industry 4.0 denotes the fourth phase of integrating communication and manufacturing processes to form smart IIoT and Artificial Intelligence (AI)-enabled systems (broadly defined to include both machine and deep learning), such as smart cities, smart airports, smart factories and smart power systems, managed via their connection to the Internet. Industry 4.0 integrates Operational Technology (OT) and Information Technology (IT) [4], where the former refers to manufacturing and control processes while IT refers to telecommunication and computing processing.

AI algorithms have the potential of maximizing the benefits from deploying IoT systems, including IIoT, to improve their flexibility, productivity and performance. Since IoT systems consist of a wide range of physical and communication devices and services involving electronics, sensors, actuators and software, they are remotely controlled and managed via connection to the Internet [4, 5]. The significance of IoT is the use of heterogeneous sensors and actuators, computing appliances, network elements and internet connectivity and their smart machines, to offer increased manufacturing speed, faster recalibration and greater configuration. This, in turn, enables the flexible designs of new business models that meet user requirements [6].

Security and privacy are still two key challenges associated with IoT system designs and deployments, due to their heterogeneity and complex implementations [7, 8]. Networks of these systems, such as smart cities, have security issues related to the sensors and protocols through which they were developed or configured for use, and even vulnerabilities in cybersecurity applications based on

learning algorithms. There are also privacy issues related to sharing private data of the systems and users to third parties [9, 10]. Sufficiently motivated attackers, such as Advanced Persistent Threat (APT) actors, can and will attempt to exploit these networks to cause disruptions or gain leverage, some instances of which have previously occurred, both individually and together with other traditional forms of warfare [10, 11].

The motivation – Existing cybersecurity solutions, including intrusion detection, malware detection, threat intelligence, threat hunting, privacy preservation and digital forensics, are available to address cybersecurity challenges [4, 12]. Currently, most of the new cybersecurity solutions rely on using AI models that require online or offline big datasets to determine the fidelity of the models. However, with the heterogeneity, complexity and non-standardization of IoT systems, there is a lack of testbeds that generate authoritative and representative heterogeneous datasets [13, 14]. The testbed architecture should embrace IoT services and their network communications at the edge, fog and cloud layers [15]. The testbed should involve realistic normal and threat scenarios that simulate production networks. This will better allow cybersecurity researchers to concurrently collect various data sources from different systems, such as datasets of IoT/IIoT sensors, datasets of network traffic, and datasets of audit traces of operating systems. The logging of concurrent datasets, containing normal and attack events, will enhance the credibility of evaluating cybersecurity solutions-based learning models. It will also assist in the development of new AI models that interpret the correlation of hacking campaigns that exploit heterogeneous systems [16]. For example, if a distributed denial of service (DDoS) attack hacks temperature sensors, network vulnerabilities, and operating systems, how could AI establish security patterns from those diverse systems?

The contribution – This paper presents the design of an orchestrated testbed architecture that comprises IoT and IIoT systems and devices of edge, fog and cloud layers. In order to do this, the NSX vCloud NFV platform of Software-Defined Network (SDN), Network Function Virtualization (NFV) and Service Orchestration (SO) were utilized at the IoT lab of University of New South Wales (UNSW) at Canberra to offer a dynamic testbed network and enable the communications of the three layers. The aim of the architecture is to generate heterogeneous datasets from various distributed data sources. While installing the blueprint of the architecture, real normal and attack scenarios are implemented to gather labeled datasets, hereafter referred to as 'TON_IoT'. These datasets contain both raw and processed data sources collected from **T**elemetry datasets of **IoT** services, **O**perating systems datasets of Windows and Linux, as well as datasets of

Network traffic. The main focus of the study is to only discuss the network architecture and the network dataset, with statistical and machine learning evaluations compared with the state-of-the-art datasets.
The Paper Outline – The remainder of this paper is structured as follows. Section 2 revisits existing datasets and discusses their limitations. The new orchestrated testbed architecture underpinning the TON_IoT architecture and datasets is explained in Section 3, and this is followed by a description of the new network dataset in Section 4. Section 5 presents the newly generated features of the TON_IoT network dataset. The data analytics and machine learning models are discussed in Section 6. Lastly, the paper is summarized in Section 7.

2. Other Existing Datasets

Several datasets are available in the literature to evaluate specific security systems such as intrusion detection. However, there are main limitations of the existing datasets [12, 17]. First, they do not contain current and complex behaviors of cyber-attacks without trustable ground truth of security events. Second, they do not have heterogeneous data sources of IoT, audit traces of operating systems and network traffic, as they were often designed using static architectures with limit legitimate and malicious operations. Third, they cannot be used to evaluate various AI-based cybersecurity solutions (e.g., threat intelligence and hunting, privacy preservation, and forensics models) because every dataset were created to evaluate a particular security solution. In more detail, most of the existing datasets were created to validate Intrusion Detection Systems (IDSs) in network traffic, but they can not be employed to validate IDSs in other environments such as hosts or IoT systems. The existing datasets and their drawbacks are individually discussed in the following:

- *The KDD99 and NSL-KDD datasets* - the IST group at the Lincoln Laboratories in the MIT University designed a simulation testbed that includes normal and attack scenarios traffic in the military network of the U.S. Air Force LAN system to create the DARPA 98 dataset in nine weeks of raw packets that were later named the KDD99 by creating some data features from the packets [18]. The NSL-KDD dataset is an improved version of the KDD99 dataset for addressing some limitations of the original dataset, such as removing redundant data and balancing samples of the training and testing sets. However, both versions are outdated and cannot reflect current network traffic as well as they do not have data of IoT sensors to determine the dynamics of current network systems.

- *The CAIDA datasets* [19] are a set of diverse data sources for validating IDSs with a limited number of attack vectors such as DDoS. The popular CAIDA dataset is the CAIDA 2007 which involves an hour of anonymized network traffic for DDoS attacks without payloads. The datasets do not have a ground truth of security events, do not contain audit traces of operating systems, and telemetry data of IoT sensors.

- *The DEFCON dataset* [20] was collected in the hacking competition of 'capture-the-flag'. This dataset is only effective for evaluating alert correlation methods. It includes only attack events without normal ones, which cannot be used to validate new AI-based security solutions.

- *The UNIBS dataset* [21] was collected from the network router of the University of Brescia- Italy, in three days. Its network traffic was gathered from 20 workstations executing the GT client daemon. The raw packets were extracted and stored on a hard disk of a client system connected to the router using an ATA controller. However, the dataset has the same issues as the CAIDA datasets.

- *The LBNL dataset* [22] was created by the Lawrence Berkeley National Laboratory (LBNL). It involves header data flows without payload, and it was anonymized. Its packets were logged from two routers in the LBNL network that includes about a thousand host systems for nearly a hundred hours. But, the dataset cannot reflect recent network events that include IoT services.

- *The Kyoto dataset* was created at Kyoto University, which includes network traffic gathered from a honeypot environment. It was created using the Zeek tool (previously named BRO) to elicit 24 features from the KDD99 dataset [23]. The main issue of this dataset is that it does not have the testbed configuration with network elements and IoT systems.

- *The DARPA 2009 dataset* [24] was synthetically created to simulate the traffic between 16 sub-networks and the internet for 10 days. It involves synthetic HTTP, SMTP and DNS data traffic, and contains attack types such as DoS and DDoS. It comprises 7000 raw files with over 6.5 TB, with each file including approximately a one- or two-minute timing window. This dataset is not labeled, and does not include any operations of IoT systems.

- *The CDX dataset* [25] was synthetically generated by the Cyber Research Centre at the US Military Academy. It was captured while running a network warfare competition, and it includes ASNM features created by the tcpdump tool. It has security events of DoS attacks and does not include various normal and attack events that mimic real networks.

- *The CTU-13 dataset* [26] was created by the CTU University, and it involves a set of 13 botnet scenarios and normal events involving 13 captures of different botnet scenarios. In every scenario, specific malware was executed to collect raw network packets. Although the dataset has a reasonable hacking scenario, it does not include new hacking events against IoT and operating systems.

- *The TUIDS dataset* [27] was gathered from the Network Security Lab at the University of Tezpur, India using various hacking types. Its raw packets were extracted using the nfdump and gulp tools for capturing data features. Their features were classified as basic, content, time, window and connectionless, plus the class label. The dataset lacks the new hacking events against IoT systems and audit traces of operating systems.

- *The ISCX dataset* [28] was designed using the concept of profiles which contains descriptions of attacks and distribution models for network architecture. Its records were captured from a real-time simulation conducted over seven days of normal network traffic and synthetic attack simulators. Several multi-stage attack scenarios were included to help in evaluating Network Intrusion Detection Systems (NIDSs). However, the dataset did not provide the ground truth of attack events and the proofing concept demands high computational resources to be applied in real-time. Further, the dataset does not have new security events against IoT sensors installed in a network system.

- *The CICIDS2017 dataset* [29] was generated at the Canadian Institute for Cybersecurity. It involves multiple normal and hack scenarios using the concept of data profiling that is like the ISCX dataset. Its network traffic was inspected by the CICFlowMeter with labeled data flows using the time stamp and the flow identifiers, but it has the same limitations of the ISCX dataset.

- *The N-BaIoT dataset* [30] was collected from a simulated IoT environment to capture several normal and botnet events. There were some IoT devices

linked with WiFi and several access points, and a wire connected to a switch and a router. Network traffic was collected from a small-scale network using the Wireshark that drops many packets in high-bandwidth networks. The main limitation of this dataset is that it does not include telemetry data of IoT sensors to determine the efficiency of new federated security solutions and does not include data traces of operating systems.

- *The UNSW-NB15 dataset* [31] was created at the University of New South Wales Canberra for evaluating intrusion detection. It has several normal and nine attack vectors. It is network traffic was simulated using the IXIA traffic generator to roundly create 100 Gigabytes, with 2,540,044 observations logged in four CSV files. Each vector includes 47 features and the class label. However, the dataset does not have security events against IoT and operating systems.

- *The Bot-IoT dataset* [32] was created at the University of New South Wales Canberra for evaluating intrusion detection and network forensics systems. It has various botnet and malware events and large-scale raw packets collected from different virtual machines. It has several IoT systems for normal operations with various data features. Nonetheless, the dataset does not have hacking vectors against IoT systems and does not include audit traces of operating systems.

The new datasets, TON_IoT, have addressed the main limitations of existing datasets. They were designed using a novel orchestrated architecture that demonstrates the interconnections of edge, fog and cloud layers. These interactions were dynamically deployed using the technologies of SDN, NVF and service orchestration. The datasets have four heterogeneous data sources, along with concurrent collections of legitimate and attack events happened at the operating systems, IoT/IIoT services and network systems.

3. Proposed Orchestrated Testbed Architecture

This section discusses the proposed orchestrated testbed architecture for generating new TON_IoT datasets, as an example of IoT and edge networks of smart cities, depicted in Figure 1. The testbed was designed based on interacting network and IoT/IIoT systems with the three layers of edge, fog and cloud described below to simulate the realistic implementation of recent real-world IoT/IIoT networks. Edge computing and Fog computing are similar in offering on-premise ser-

vices, like cloud services, including Software-as-a-Services (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). The services are offered near to organisations to manage IoT systems and their generated data, allowing analytics and intelligence close to end-users rather than sending a huge amount of data sources to the cloud, which has limitations related to the network bandwidth, security, and latency [33].

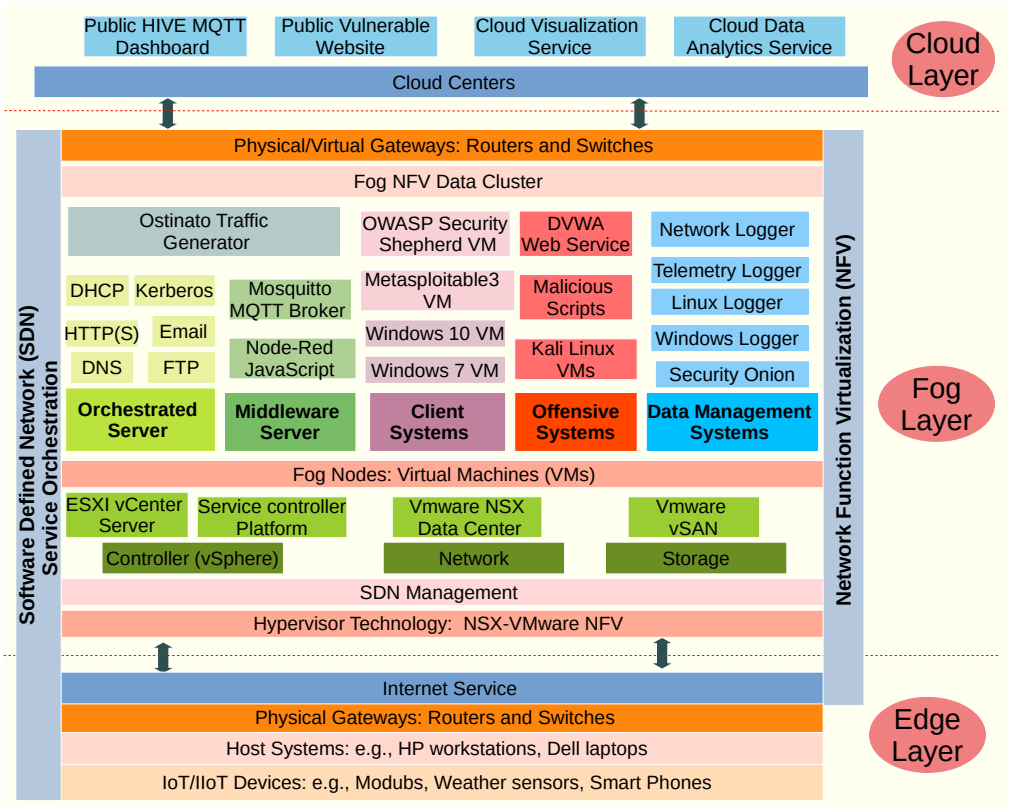


Figure 1: Proposed dynamic architecture for evaluating security applications in IoT and edge networks of smart cities

The key difference between edge computing and fog computing is that the two lies in where the location of intelligence, analytics, and computing is placed. A fog layer places intelligence, analytics, and computing at Local Area Networks (LANs). In this layer, data sources are transmitted from endpoints to a gateway, where they are then transferred to sources for processing and return transmission. An edge layer places intelligence, analytics, and processing power in devices such as embedded automation controllers, and lightweight IoT devices. In our proposed

architecture, the edge layer is used to configure and manage IoT/IIoT devices and gateways while the fog layer is employed to offer the computing, analytics, and intelligence using the virtualisation technology via LAN connections [10]. The key functions of the three layers used in our architecture are explained below.

- *Edge Layer* includes the physical devices and their operating systems employed as the infrastructure of deploying the virtualization and cloud services at the layers of fog and cloud, respectively [34]. It includes multiple IoT/IIoT devices, including weather and light bulb sensors, smartphones and smart TVs, as well as host systems, such as workstations and servers, used to connect IoT/IIoT devices, hypervisors and physical gateways (i.e., routers and switches) to the Internet service. The hypervisor technology of NSX-VMware was deployed on a host server at the edge layer to manage the Virtual Machines (VMs) created at the fog layer.

- *Fog Layer* involves the virtualization technology that programs and controls the VMs and their services using the NSX-VMware platform to offers the framework of executing SDN, NVF and SO in the proposed testbed. This layer offers these capabilities via the components: hypervisor technology, SDN management, fog nodes of VMs and fog NFV data cluster.

- *Cloud layer* includes the cloud services configured online in the testbed [35]. The fog and edge services connected with the public HIVE MQTT dashboard, public PHP vulnerable website, cloud virtualization services (e.g., Microsoft Azure or AWS), and cloud data analytics services. The public HIVE MQTT dashboard enabled us to publish and subscribe to the telemetry data of IoT/IIoT services via the configuration of the node-red tool. The public PHP vulnerable website was used to simulate injection hacking events. The other cloud services were configured either in Microsoft Azure or AWS to transmit sensory data to the cloud and visualize their patterns.

The dynamism between the three layers, including physical and simulated systems, was flexibly managed using the three technologies of Software Defined Networks (SDN), Service Orchestration (SO) and Network Function Virtualization (NFV). The elements of the architectural testbed are explained below to illustrate the functions of the three layers and technologies involved. The technologies of SDN, NVF and SO and their execution in the testbed of the TON_IoT are described below.

- *Software-Defined Network (SDN)* is a solution that facilitates dynamic and programmable network configurations to enhance their performances. The NSX-VMware data center platform [36] is used to provide an SDN solution of the proposed testbed of the TON_IoT datasets. This technology permits the creation of overlay networks with the same capabilities of physical networks. VMware NSX was deployed with VMware vSphere hypervisor NFV to allow the creation and management of various virtual machines that concurrently operate to offer the IoT/IIoT and network services.

- *Network Function Virtualization (NVF)* is an emerging network architecture that allows the replacement of expensive hardware devices with software-based network devices that implement as VMs on hypervisor servers [37]. In VMware NSX, the vCloud NFV platform was employed to provide a modular design with abstractions that enable multi-domain, hybrid physical and VM deployments [38]. The NSX vCloud NFV platform enables the design of a dynamic testbed IoT/IIoT network of the TON_IoT with creating and controlling several VMs for hacking and normal operations, allowing the communications between the edge, fog and cloud layers.

- *Service Orchestration (SO)* is the implementation of the operational and functional procedures contained in establishing and delivering end-to-end services [39]. To offer the SO in the proposed testbed, the NSX vCloud NFV platform allowed the execution of SDN and NVF, where NFV was to reduce the dependency on using single-purpose devices by using multiple VMs, and SDN was to allow programming network devices as software and improve the IoT/IIoT network performances. The platform operated with external functions for service orchestration and management, and a fully incorporated suite for operational intelligence and monitoring. This suite was used to improve the runtime systems with workflows for dynamic workload optimization and proactive issue avoidance.

The fog layer contains the fundamentally virtualized components of the proposed testbed architecture. The components of hypervisor technology, SDN management, fog nodes of VMs and fog NFV data cluster, are executed by the NSX vCloud NFV platform, as explained in the following:

- *Hypervisor technology and SDN management* were implemented and configured using the controller, network and storage platforms [36].

First, the *controller (vSphere)* includes the ESXI vCenter server and the service controller platform. The ESXI vCenter server provides a bare-metal virtualization server that combines applications on less hardware, where it offers a centralized platform for managing vSphere systems that involve vSphere replication and vSphere data protection for security purposes. The service controller platform involves multiple infrastructure services, which comprises single sign-on, licensing, and a certificate authority that secures accessing the controller.
Second, the *network* involves the VMware NSX and vCloud NFV platform that provides virtualization of the testbed network, as well as service orchestration and management (i.e., SND, NFV and SO) to dynamically communicate the layers of the edge, fog and cloud. It enables granular overlay networking and security at the hypervisor level, offering distributed network services for NFVs including granular network isolation using NSX micro-segmentation and simplified operations.
Third, the *storage* is software-defined storage embedded in the ESXi hypervisor. It contains the VMware vSAN that provides simple hypervisor converged storage embedded in the hypervisor that can be co-located with the VNF workloads to reduce jitter and latency. It enabled to configure the suitable capacity of the VMs fog nodes used in the testbed [40].

- *Fog NFV data cluster* is a service in the controller, where ESXi hosts are the basic compute building blocks of vCloud NFV and NSX [36]. ESXi host resources were grouped together to provide an aggregate set of resources in the created fog nodes of VMs, called a cluster. Clusters are used to logically separate between management and NFV components.

- *Fog nodes of VMs* include the VMs employed to simulate the testbed IoT/IIoT network of generating the TON_IoT datasets. There are five main systems included in the fog nodes that contain 17 VMs of executing normal and attack scenarios and managing IoT/IIoT services between the edge and cloud layers, as depicted in Figure 2. The functions of the five main systems are discussed below and their network features are shown in Figure 1.

  1) *Orchestrated server* is one of the main virtualized servers configured in the testbed using the Ubuntu 14.04 LTS with the IP address *192.168.1.190*.

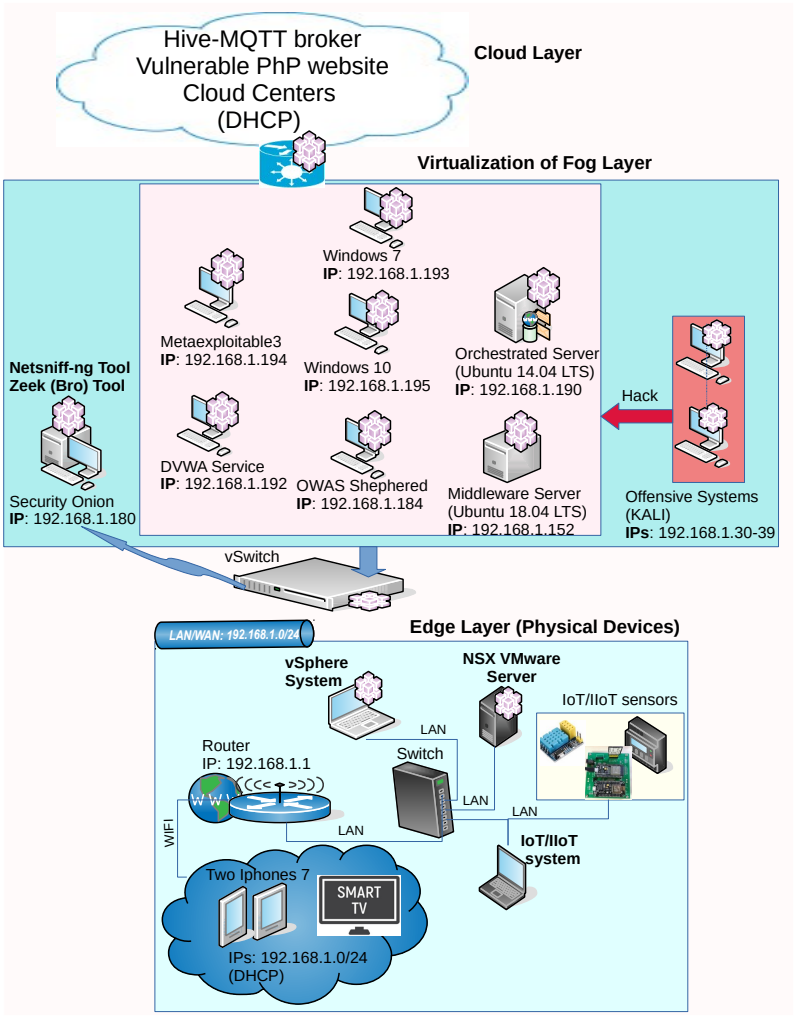


Figure 2: Configured Testbed of TON_IoT datasets for collecting network data

This server offered many orchestrated services to simulate real production networks and generate more simulated network traffic that transmits with other VMs in the testbed [41]. It included network services, including DNS (i.e., mydns.com), HTTP(s), DHCP, email server (i.e., Zimbra), Kerberos and FTP. In addition, the Ostinato traffic generator, which generates network traffic based on well-defined protocols and services of the TCP/IP model, and transmits it to other VMs in the testbed using their IP addresses. This allows us to automatically generate various legitimate packets that would

happen in real production networks. While collecting datasets, the orchestrated server has a workflow engine that automates running the services and the traffic generator all the period of simulating normal and attack scenarios.

2) *Middleware server* is the IoT/IIoT virtualized server deployed in the testbed using the Ubuntu 18.04 with the IP address *192.168.1.152*. This server included the scripts that implemented IoT/IIoT services linked them with the cloud layer to subscribe and publish telemetry data of IoT/IIoT sensors. The node-red and Mosquitee MQTT broker tools were installed in this server to offer seven IoT/IIoT sensors: weather, smart garage door, smart fridge, smart TCP/IP Modbus, GBS tracker, motion-enabled light, and smart thermostat. The details of these sensors and their features are published in [1]. The node-red is a programming tool for linking together IoT/IIoT devices, APIs and online services such as cloud ones, where the configured IoT/IIoT services were developed using Java Script linked to simulated and physical IoT/IIoT devices. Mosquito is an open-source message broker that implements MQTT protocols for publishing and subscribing to telemetry data of IoT/IIoT data. The node-red tool was configured to transmit the data to the public HIVE MQTT dashboard in the cloud layer. The telemetry data were logged using the telemetry logger in CSV files while executing normal and attack scenarios.

3) *Client Systems* are a set of one smart TV, two smartphones and virtualized client systems, configured in the testbed and linked with the edge, fog and cloud layers. The smart TV (Samsung TV) and two smartphones (iPhone 7) were configured by dynamic IP addresses withing the subnet range using the DHCP server installed on the orchestrated server. The client systems include Windows 7 VM (IP address: *192.168.1.193*), Windows 10 VM (192.168.1.195), DVWA web service (IP address: *192.168.1.192*), OWASP security Shepherd VM (*192.168.1.184*), Metaspoitable3 (IP address: *192.168.1.194*). The two Windows OSs were used as the remote web interface of the node-red IP (*192.168.1.152*) and their network traffic and audit traces were logged. The DVWA (Damn Vulnerable Web App) was used to make security vulnerabilities through the web applications hacked using the virtualized offensive systems. The OWASP security Sphered VM is an open-source platform that has many security vulnerabilities against mobile and web applications exploited using the offensive systems. In ad-

dition, the Metasploitable3 VM was deployed in the testbed to increase vulnerable fog nodes and hack them using various attacking techniques by the offensive systems.

4) *Offensive systems* include the kali Linx VMs and scripts of hacking scenarios that hacked vulnerable systems in the network testbed . Ten static IP addresses (i.e.,*192.168.1.30-39*) were employed in the testbed to launch attacking scenarios and breach vulnerable systems either IoT/IIoT services (client and public MQTT brokers and node-red IP), operating systems (i.e., Windows 7 and 10, and Ubuntu 14.04 LTS and 18.04 LTS), and network systems (i.e., IP addresses and open protocols of the VMs). In the TON_IoT datasets, nine recent attack events: scanning, backdoor, ransomware, DoS, DDoS, Man-in-The-Middle (MITM), data injection, cross-site scripting (XSS) and password violations, were executed, as discussed in Section 4 .

5) *Data management systems* involve the logger tools that capture and store raw and filtered data from operating systems, IoT/IIoT services and network systems [42]. Since the TON_IoT datasets have four heterogeneous data sources concurrently collected, several logging tools were applied to log data of different systems. First, For logging the Windows 7 and 10 data, the Windows logger, the so-called Microsoft Data Collector Sets, was used to store data of processes, CPUs, memories and hard disks. Second, the Linux logger, the so-called atop, was employed to capture the data of processes, CPUs, memories and hard disks from the orchestrated and middleware servers. Third, a telemetry logger was configured to store the sensory data of IoT/IIoT services that passed through the MQTT brokers installed in the middleware VM. Fourth, the focus of this study, for logging and inspecting network traffic in the TON_IoT datasets, the Security Onion VM (IP address: *192.168.1.180*) was used to log network data from all the active systems in the testbed using a virtual mirror switch that forwards the entire network traffic to this VM without dropping any traffic. As shown in Figure 2, the netsniff-ng tool was used to capture the entire network packets from the entire systems in pcap formats without packet drops. The Zeek Network Security Monitor tool (previously named Bro) was used to generate data features from the pcap files, as discussed below.

4. Description TON_IoT Network Datasets

The TON_IoT datasets are new generations of IoT/IIoT, network traffic and operating systems datasets for evaluating the fidelity and efficiency of various AI-enabled cybersecurity applications. The datasets have been called 'TON_IoT' as they include heterogeneous data sources collected from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets of Windows 7 and 10 as well as Ubuntu 14.04 and 18.04 TLS and Network traffic. The datasets were collected from a realistic and large-scale testbed network, and it was designed at the IoT Lab of the UNSW Canberra Cyber, the School of Engineering and Information technology (SEIT), UNSW Canberra. The datasets were simulated based on a realistic testbed that includes the properties of SDN, NVF and SO to allow the communications between the layers of edge, fog and cloud. The datasets were gathered in parallel processing to collect labelled data of normal and cyber-attack events from the proposed testbed above.

In this study, the TON_IoT network dataset and its new characteristics are explained. As published in [43], the directories of the datasets contain the following:

- **"*Raw_datasets/Raw_Network_dataset*"**- it involves two sub-directories, namely 'network_dataset_pcap' and 'network_dataset_Bro'. The first involves the entire raw packets of normal and attack events collected from the testbed using the netsniff-ng tool. The second includes the log and CSV files extracted from the pcap files using the Zeek tool. The CSV files include the data features extracted from the log files.

- **"*Processed_datasets/Processed_Network_dataset*"**- This directory includes 23 CSV files that extracted using the Zeek tool. Every file includes about million records except the last file contains 329,021 records, where each file has 44 attributes and two attributes of the class label and types either normal or attack category.

- **"*SecurityEvents_GroundTruth_datasets/Security Events_Network_datasets*"**- this directory contains the ground truth tables used for labeling the TON_IoT dataset for network traffic. To create these CSV tables, while launching each attack, the IP addresses of the offensive systems and their timestamps were logged to accurately label records. This stage is essential to ensure the fidelity of the datasets and their credibility in evaluating cybersecurity solutions using AI algorithms. There is also a directory, named "Hacking_techniques", which includes the scripts and links of hacking scenarios used in the datasets.

- **"*Train_Test_datasets/Train_Test_Network_dataset*"**- this directory contains a CSV file that includes a subset of the entire attack types and normal records. This file is suggested to be used for evaluating new AI-based cybersecurity solutions and make fair comparisons between the new security solutions.

- **"*Description_stats_datasets/Description_stats _Network_dataset*"**- this directory involves two sub-directories: data feature descriptions and number of recorded included in the proposed directory and the training and testing file.

4.1. Benign and Attack Scenarios and Agents

Generating normal/benign and attack scenarios is the main task of creating authentic and realistic datasets [10]. Agent systems were developed to create realistic normal and hacking scenarios in the testbed for collecting heterogeneous data sources. On the one hand, normal scenarios were used to generate a broad number of normal records. These happened through the legitimate operations of the orchestrated and middleware servers with the host clients without launching any attacking events. For example, publishing and subscribing to telemetry data between edge, fog and cloud layers, sending network traffic using the ostinato traffic generator between the VMs, using services such as FTP, DNS, HTTP installed on the orchestrated server. On the other hand, hacking scenarios were employed to launch nine attack categories against vulnerable elements of IoT/IIoT applications, operating systems, network systems. The scripts and links of these attacking categories are published in [4]. The nine attack families utilized in the datasets are explained as follows:

1. **Scanning attack**- is also called reconnaissance or probing, is the first stage of the cyber kill chain model or penetration testing. The aim of this attack is to collect information about victim systems such as finding active IP addresses and open ports in the testbed network. The Nessus and Nmap tools were used from the offensive systems with IP addresses *192.168.1.20-38* against the target subnet *192.168.1.0/24* and all other public vulnerable systems such as the Public MQTT broker and vulnerable PHP website. For example, *nmap 192.168.1.40-254*, and the scans of the Nessus tool for the same range of IP addresses.
2. **Denial of Service (DoS) attack** – is any fake flooding that attempts to corrupt resources of IoT/IIoT services and network systems. DoS attack scenarios on the offensive systems with IP addresses *192.168.1.{30,31,39}*

were applied to hack vulnerable elements in the IoT testbed network. Python scripts using the Scapy package were developed to launch the DoS attacks, for example, *send(IP(src=srcip, dst="broker.hivemq.com", ttl=rnd2)/ TCP(sport=port,dport=port, ack=rnd, window=rnd), count=100)*.

3. **Distributed Denial of Service (DDoS) attack**- launches multiple DDoS attacks to corrupt targeted systems. Systems, such as network and IoT devices, were infected with malware, turning each one into a bot or zombie. The attacker then had remote control over the group of bots named a botnet. DDoS attacks in the offensive systems with IP addresses *192.168.1.{30,31,34, 35,36,37,38}* were executed to breach several weaknesses in the IoT testbed network. Python scripts using the Scapy package were developed to launch the DDoS attacks. Further, automated bash scripts were developed to launch DDoS against vulnerable nodes of the testbed using the ufonet toolkit. For example, *python2 ./ufonet -a "http://www.mqtt-dashboard.com"-r "100" --loic "1000" --loris "1000" --ufosyn "1000" --spray "1000" --smurf "1000" --xmas "1000" --nuke "1000" --tachyon "1000" --threads "100"*.
4. **Ransomware attack**- is a complex type of malware that prevents normal users to access systems or services by encrypting them until they pay a ransom. The Kali Linux with IP addresses *192.168.1.{33, 37}* was used to execute this malware against windows operating systems and their webpages of monitoring IoT services. This attack was executed by the Metasploit framework that hacked the SMB vulnerability of the systems, named eternalblue.
5. **Backdoor attack** - is any malicious technique by which attackers can get around normal security measures and gain high-level user access on a victim system. Once hackers gain access to targeted systems using attack scenarios, for example, ransomware, DoS or DDoS, hackers keep persistence to the hacked systems using the backdoor technique to steal personal and financial data, install additional malware, and hijack devices. The offensive systems with IP addresses *192.168.1.{33,37}* were used to keep the hacking persistence using the Metasploit framework by executing a bash script of the command *"run persistence -h"*.
6. **Injection attack** – is an attacking technique that injects or inserts any fake input data from clients to applications, such as SQL injection attacks for exploiting PHP and ASP applications. various injection scenarios were implemented at the offensive systems with IP addresses *192.168.1.{30, 31, 33, 35}* to inject data inputs against web applications of DVWA and Security Shepherd VMs and webpages of IoT services through other VMs, including SQL injection, client-side injection, broken authentication and data

management, and unintended data leakage. For example, using a bash script of the IoT testbed systems such as *sqlmap -u http://192.168.1.195/dvwa/vulnerabilities/sqli/?id=53*.

7. **Cross-site Scripting (XSS) attack** – is a malicious technique that injects normal and trusted web applications such as webpages of IoT services. The XSS attack happens when an attacker employs a web application to transmit malicious code, generally in the form of a browser side script, to different end-users. The offensive systems with IP addresses *192.168.1.{32,35,36,39}* were used to illegally inject the web applications of DVWA and Security Shepherd VMs and webpages of IoT services through other VMs. In these systems, malicious bash scripts of python codes were developed to hack the web applications of the testbed network using the Cross-Site Scripter toolkit (named XSSer). For instance, *python2 xsser -u "https://192.168.1.184/index.jsp" --auto --Cem "Hex,Str,Hex" --user-agent "hacking" --timeout "20" --threads "5"*.
8. **Password cracking attack**- is any password hacking technique, such as brute-force and dictionary attacks, that depends on guessing possible combinations of a targeted password until the correct password is discovered. It was used to breach passwords of operating systems, IoT services and web applications deployed in the testbed. The offensive systems with IP addresses *192.168.1.{30, 31, 32, 35, 38}* were used to execute this attack. In these systems, the hydra and cewl toolkits were configured using automated bash scripts to concurrently launch password hacking scenarios against vulnerable nodes in the testbed. For example, *hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt -t 5 ftp://192.168.1.152*, and *cewl http://192.168.1.195/dvwa --auth_type Digest --auth_user admin --auth_pass password -v*.
9. **Man-In-The-Middle (MITM) attack**- happens when attackers place themselves between users and application to snoop or masquerade as one of the parties, establishing the deceptive appearance as if an ordinary exchange of information is afoot. Such hacking scenarios could steal information about web applications, network and IoT services. The offensive systems with IP addresses *192.168.1.{31,34}* was employed to launch various MITM scenarios in the testbed network. In the systems, the Ettercap tool was used to execute ARP spoofing, ICMP redirection, port stealing and DHCP spoofing.

5. Generated Features

The Zeek tool [44] was employed to extract 44 flow features/attributes from the network packets (i.e., pcap files). The attributes were classified into four groups based on the principle of service profiles which include information about the connection, statistics, user attributes (i.e., DNS, HTTP and SSL activities), and violation attributes. The first group, connection attributes, contains information about flow identifiers (i.e., source and destination IPs and ports, as well as protocol types), and their time and size, as listed in Table 1. The second group, statistical attributes, comprises information about the numbers of flow identifiers and their statistics, as demonstrated in Table 2. The third group, user-centric group, includes information about high-level services and their interactions with end-users, including DNS attributes listed in Table 3, HTTP attributes shown in Table 4 and SSL attributes exhibited in Table 5. The fourth group, violation attributes, includes any abnormal information happened while transmitting packets, as listed in Table 6. The four groups include sufficient flow attributes that can be used to apply AI-based security applications, such as intrusion detection, malware classification, intrusion detection and forensics analysis.

It is worth mentioning here an authentic labeling process was applied to tag each record, either normal or attack, involving its type. In order to ensure the correctness of the labeling process, the flow identifiers and timestamp generated from the attributes were matched with flow identifiers and timestamp of every attack executed, as described above. Then, two new attributes, namely *label* and *type*, were added to tag the entire vectors in the datasets, as displayed in Table 7. The *label* attribute can be used when a binary classification model is used to classify normal and attack data. The *type* attributes can be applied when a multi-classification model is used to classify normal and attack types.

6. Applications and Results of Data Analytics and Machine Learning

This section discusses the experimental results of the datasets, including data analytics, use of data features, use of machine learning, and comparisons with other datasets.

6.1. Data Analytics and Important Features

It is worth mentioning here that The data preprocessing, features selection (i.e., variable importance), and machine/deep learning techniques were

Table 1: Connection features

ID	Feature	Type	Description
1	ts	Time	Timestamp of connection between flow identifiers
2	src_ip	String	Source IP addresses which originate endpoints' IP addresses
3	src_port	Number	Source ports which Originate endpoint's TCP/UDP ports
4	dst_ip	String	Destination IP addresses which respond to endpoint's IP addresses
5	dst_port	Number	Destination ports which respond to endpoint's TCP/UDP ports
6	proto	String	Transport layer protocols of flow connections
7	service	String	Dynamically detected protocols, such as DNS, HTTP and SSL
8	duration	Number	The time of the packet connections, which is estimated by subtracting 'time of the last packet seen' and 'time of the first packet seen'
9	src_bytes	Number	Source bytes which are originated from payload bytes of TCP sequence numbers
10	dst_bytes	Number	Destination bytes which are responded payload bytes from TCP sequence numbers
11	conn_state	String	Various connection states, such as S0 (connection without replay), S1 (connection established), and REJ (connection attempt rejected)
12	missed_bytes	Number	Number of missing bytes in content gaps

Table 2: Statistical features

ID	Feature	Type	Description
13	src_pkts	Number	Number of original packets which is estimated from source systems
14	src_ip_bytes	Number	Number of original IP bytes which is the total length of IP header field of source systems
15	dst_pkts	Number	Number of destination packets which is estimated from destination systems
16	dst_ip_bytes	Number	Number of destination IP bytes which is the total length of IP header field of destination systems

Table 3: DNS features

ID	Feature	Type	Description
17	dns_query	string	Domain name subjects of the DNS queries
18	dns_qclass	Number	Values which specifies the DNS query classes
19	dns_qtype	Number	Value which specifies the DNS query types
20	dns_rcode	Number	Response code values in the DNS responses
21	dns_AA	Boolean	Authoritative answers of DNS, where T denotes server is authoritative for query
22	dns_RD	Boolean	Recursion desired of DNS, where T denotes request recursive lookup of query
23	dns_RA	Boolean	Recursion available of DNS, where T denotes server supports recursive queries
24	dns_rejected	Boolean	DNS rejection, where the DNS queries are rejected by the server

Table 4: SSL features

ID	Feature	Type	Description
25	ssl_version	String	SSL version which is offered by the server
26	ssl_cipher	String	SSL cipher suite which the server chose
27	ssl_resumed	Boolean	SSL flag indicates the session that can be used to initiate new connections, where T refers to the SSL connection is initiated
28	ssl_established	Boolean	SSL flag indicates establishing connections between two parties, where T refers to establishing the connection
29	ssl_subject	String	Subject of the X.509 cert offered by the server
30	ssl_issuer	String	Trusted owner/originator of SLL and digital certificate (certificate authority)

Table 5: HTTP features

ID	Feature	Type	Description
31	http_trans_depth	Number	Pipelined depth into the HTTP connection
32	http_method	String	HTTP request methods such as GET, POST and HEAD
33	http_uri	String	URIs used in the HTTP request
34	http_referrer	String	Values of the 'referer' header in the HTTP protocol
35	http_version	String	The HTTP versions utilised such as V1.1
36	http_request_body _len	Number	Actual uncompressed content sizes of the data transferred from the HTTP client
37	http_response_body _len	Number	Actual uncompressed content sizes of the data transferred from the HTTP server
38	http_status_code	Number	Status codes returned by the HTTP server
39	http_user_agent	Number	Values of the User-Agent header in the HTTP protocol
40	http_orig_mime _types	String	Ordered vectors of mime types from source system in the HTTP protocol
41	http_resp_mime _types	String	Ordered vectors of mime types from destination system in the HTTP protocol

Table 6: Violation features

ID	Feature	Type	Description
42	weird_name	String	Names of anomalies/violations related to protocols that happened
43	weird_addl	String	Additional information is associated to protocol anomalies/violations
44	weird_notice	Boolean	It indicates if the violation/anomaly was turned into a notice

Table 7: Labelling attributes

ID	Feature	Type	Description
45	label	Number	Tag normal and attack records, where 0 indicates normal and 1 indicates attacks
46	type	String	Tag attack categories, such as normal, DoS, DDoS and backdoor attacks, and normal records

applied using the training_testing_set data and the *H2O.ai* package of the R-programming language [45]. The R code of applying these techniques is publicly available in this Github's link [46]. We used only a small part of the entire dataset, i.e., *Train_Test_Network_dataset*, and the IP addresses and ports were used when the four machine learning models were built to obtain the high detection accuracy and low false alarm rate. These results can be used as benchmark outputs for comparison purposes, and we recommend that the researchers should remove the source and destination IP addresses and ports when they develop new machine learning algorithms. Removing the IP addresses and ports from the data features and using the entire records of the dataset will demonstrate the complexity of security events involved in this dataset and will reduce the performances of machine learning algorithms.

The TON_IoT network dataset has 223,390,21 records of normal and attacks data, as listed in Table 8. The dataset has 461,043 records collected from the entire network dataset to include all the attacks and normal events. These records can be used to apply different machine learning models and handle the challenge of imbalanced normal and attack records that are usually challenging. This challenge refers to that the number of normal records is too much greater than abnormal ones. Since the TON_IoT network dataset has a large number of attributes and different attribute types, such as categorical and numeric ones, it demands filter-

ing and processing the attributes to improve the performances of machine learning techniques. It is important to apply data preprocessing and feature selection techniques to improve the performances of machine learning models, as discussed below.

Table 8: Number of records and their data types in the entire network dataset of TON_IoT and its training and testing set

	backdoor	**ddos**	**dos**	**injection**	**mitm**	**password**	**ransomware**	**scanning**	**xss**	**normal**	**Total**
Total data records	508116	6165008	3375328	452659	1052	1718568	72805	7140161	2108944	796380	**22339021**
Train_Test records	20000	20000	20000	20000	1043	20000	20000	20000	20000	300000	**461043**

- *A label encoder technique* was applied to transform categorical attribute's values into numeric ones [47]. It converts every categorical value in an attribute into the integer of its index. For instance, the protocol's attribute with a categorical value (tcp, udp, icmp) is converted into (1,2,3) to improve the performance of machine learning.

- *A Wrapper Feature selection technique-based Random Forest (RF)* was employed by estimating the relative influence of each attribute. The technique was developed based on the Mean Square Error (MSE) and the decision tree algorithm to determine to what extent MSE improves the results. The feature's attributed reduction of MSE is the difference in an squared error between the node of a tree and its children nodes. The squared error for each individual node is the reduction in the variance of the response value within that node. As shown in Figure 3, the most important features of the TON_IoT network dataset are scaled in a range of [0,1], where 1 refers to the highest correlated attributed (e.g., scr_IP) and 0 denotes the lowest correlated attributes (e.g., duration).

6.2. Attack and Normal Classification using Machine Learning

After applying label encoder and feature selection techniques, four popular machine learning algorithms were used to determine their efficiency in classifying normal and attack events. The algorithms and their parameters and results are discussed as follows:

- *Gradient Boosting Machine(GBM)* is a forward learning ensemble technique for classifying data. It sequentially establishes trees on the important features of the dataset in a fully distributed manner [48]. Using the H20.ai

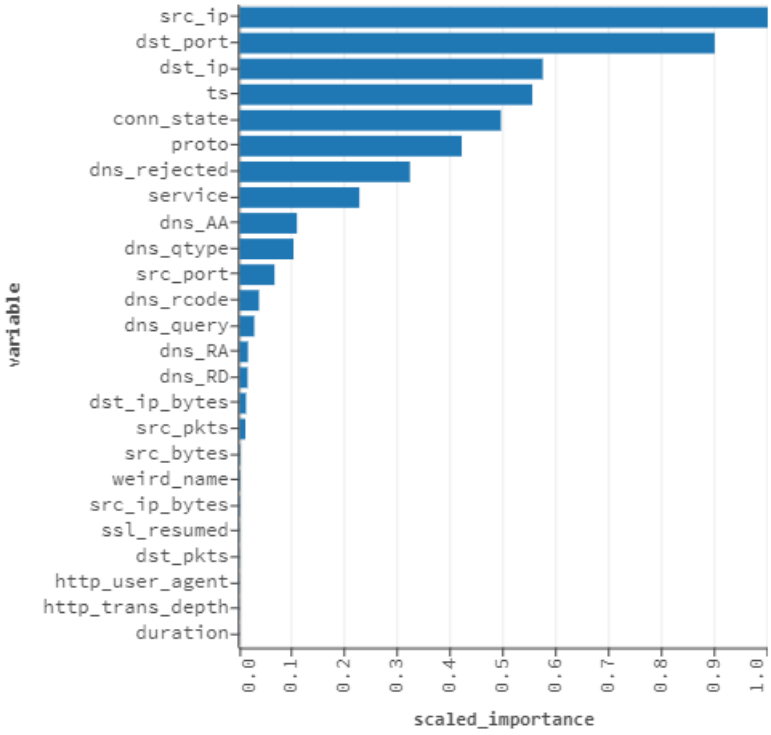


Figure 3: Important variables/features selected from the TON_IoT network dataset

package, the GBM's parameters were adapted by these values (*ntrees = 15, max_depth = 5, min_rows =2, learn_rate = 0.01, distribution= "multinomial", nfolds=10*). The evaluation criteria and confusion matrix of the GBM model using the Training_testing set of the TON_IoT network dataset are depicted in Figure 4. The Area Under the Curve (AUC)-Receiver Operating Characteristics (ROC) plot represents the performance measurement of classifying data at various threshold settings (i.e., 0.6899 is the best outcome). The ROC-AUS shows the relation between True Positive Rate (tpr) and False Positive Rate (fpr), which reflects that this model achieves AUC=0.982945. This model also accomplishes high precision, recall, specificity, sensitivity and f1, and f2 measures (i.e., > 0.99).

- *Random Forest (RF)* is a robust classification algorithm that classifies data based on trees. Every tree is considered a weak classifier established on a subset of records and features. The prediction of the trees was used to make a final prediction that classify normal and attack data [49]. Using the

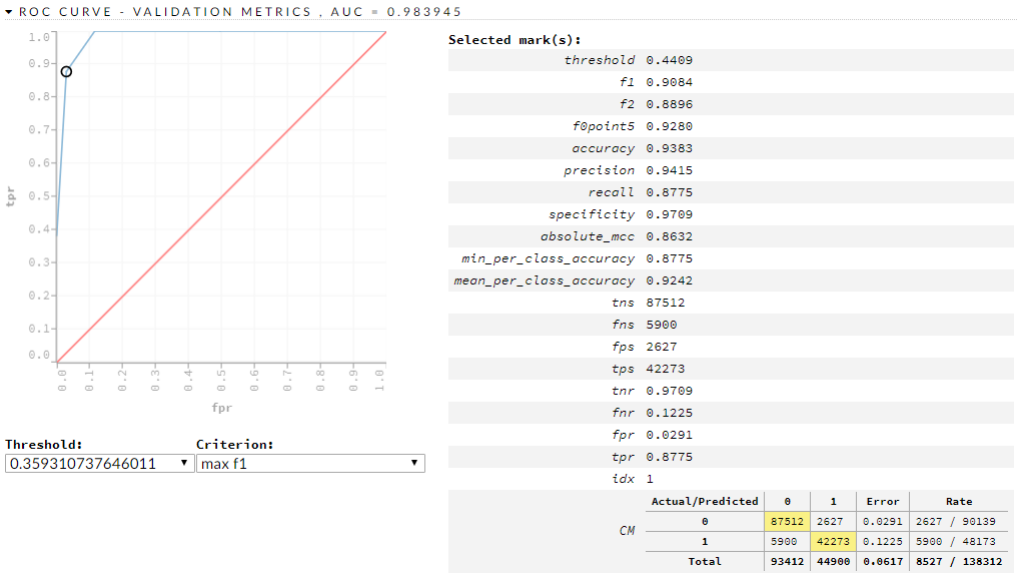


Figure 4: GBM's evaluation criteria, ROC curve, and confusion matrix on the Training_testing set of the TON_IoT network dataset

H2O.ai package, the RF's parameters were set by these values (*ntrees = 150, stopping_rounds = 2, seed = 10000, nfolds=10*). The evaluation criteria and confusion matrix of the RF model using the Training_testing set of the TON_IoT network dataset are shown in Figure 5, with a low improvement of AUC compared with the GBM model.

- *Naive Bayes (NB)* is a classification technique, which depends on solid assumptions of the independency of covariates in executing the Bayes theorem. It assumes the independence between the predictor attributes constrained on the response feature and a Gaussian distribution of numeric predictors with mean and standard deviation computed from the training data to classify normal and attack data. The Laplace smoothing function was used to predict the maximum likelihood of normal and attack classes [50]. The NB's parameters were adapted by these values (*nfolds=10, laplace = 3*). The evaluation criteria and confusion matrix of the NB model using the Training_testing set of the TON_IoT network dataset are shown in Figure 6, with lower performance (i.e., AUC=0.912804) than the GBM and RF models.

- *Deep Neural Network (DNN)* is a multi-layer feedforward artificial neu-

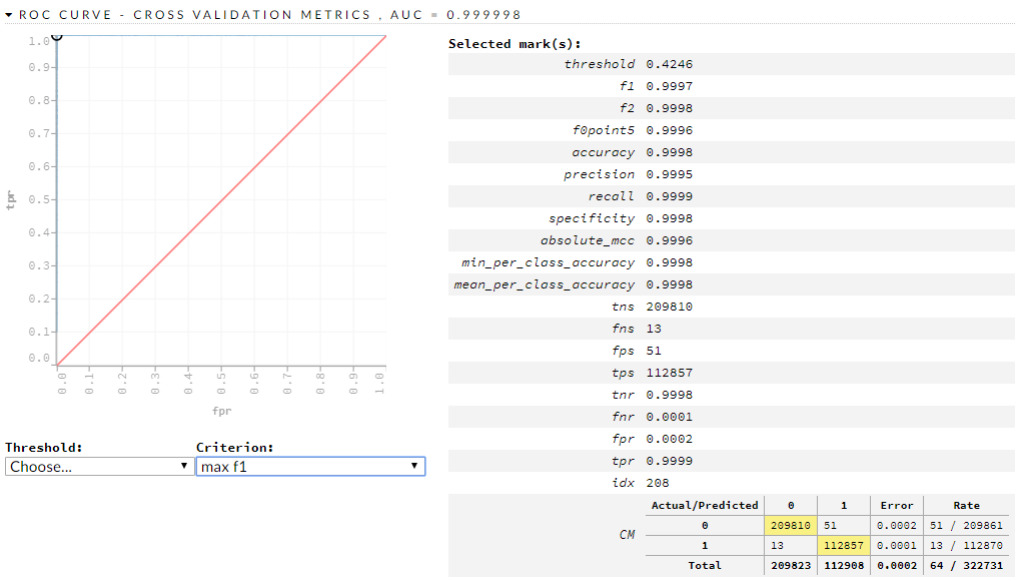
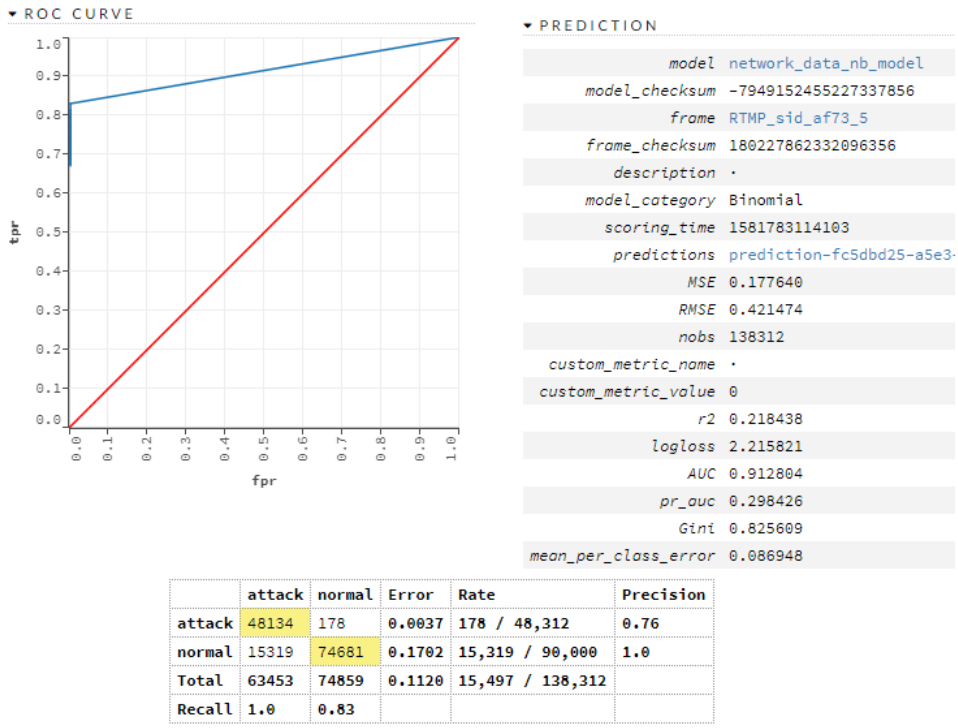


Figure 5: RF's evaluation criteria, ROC curve, and confusion matrix on the Training_testing set of the TON_IoT network dataset

ral network which is trained by stochastic gradient descent using back-propagation [51]. The network was adapted to include 15 hidden layers consisting of 15 neurons with a tanh activation function, 10 cross-validation and epochs, and a softmax activation in the output layers function to classify normal and attack data. The evaluation criteria and confusion matrix of the DNN model using the Training_testing set of the TON_IoT network dataset are represented in Figure 7, with lower performance than RF and higher performance than the other two techniques.

The four machine learning techniques achieved high performances using the train-test set, which is a small portion of the entire dataset. The reason for accomplishing these outcomes is that the generated features in this dataset have high variations between normal and attack events. The results of the models are very high, in terms of detection accuracy and false alarm rates, as we employed a small subset for training and testing the models, and we used the IP addresses and ports that assist in classifying normal and attack observations. We recommend excluding IP addresses and port when we evaluate new machine learning-based cybersecurity models and use the entire records of the dataset to ensure the complexity of the attack and legitimate behaviours involved in the dataset.



	attack	normal	Error	Rate	Precision
attack	48134	178	0.0037	178 / 48,312	0.76
normal	15319	74681	0.1702	15,319 / 90,000	1.0
Total	63453	74859	0.1120	15,497 / 138,312	
Recall	1.0	0.83			

Figure 6: NB's evaluation criteria, ROC curve, and confusion matrix on the Training_testing set of the TON_IoT network dataset

6.3. Comparisons and Future Security Applications using TON_IoT Datasets

In order to provide a fair comparison between the TON_IoT datasets and popular ones, Table 9 illustrates the significant characteristics between the datasets [12]. As demonstrated in the table, the KDD99 and NSLKDD datasets are outdated and do not reflect current network traffic. The CAIDA and UNIBS dataset have realistic network traffic and configuration, but they do not involve heterogeneous data sources and are not labeled with a few numbers of attack events. The DEFCON and LBNL datasets were collected from a simulation environment for collecting various attack activities, and they do not contain heterogeneous data collections. The TUIDS ISCX and CICDS2017 datasets were collected from realistic environments and have full packet capture with a variety of security events, but they do not contain heterogeneous data sources, as they have only network traffic. The DARPA-2009 dataset was gathered from a realistic environment, but it is not labeled with a large volume of network packets and its ground truth is not authentic.

Table 9: Comparisons of popular datasets with the new TON_IoT datasets

Datasets	**Realistic network configuration**	**Realistic network traffic**	**Labelled observations**	**Heterogeneous data sources**	**Total interaction capture**	**Full packet capture**	**Many malicious scenarios**
KDD99 and NSL-KDD	$True$	$False$	$True$	$False$	$True$	$True$	$True^{8}$
CAIDA	$True^{1}$	$True$	$False$	$False$	$False^{5}$	$False^{4}$	$False^{2}$
DEFCON	$False$	$False^{5}$	$False$	$False$	$True$	$True$	$True^{8}$
LBNL	$False$	$True^{1}$	$False$	$False$	$False$	$False^{4}$	$True$
UNIBS	$True$	$True^{2}$	$True$	$False$	$True^{4}$	$True$	$False^{2}$
TUIDS	$True$	$True^{9}$	$True$	$False$	$True^{4}$	$True$	$True^{8}$
ISCX and CICDS2017	$True^{2}$	$True^{6}$	$True$	$False$	$True$	$True$	$False$
DARPA-2009	$True^{1}$	$True^{5}$	$False$	$False$	$False^{5}$	$True$	$True$
UNSW-NB15	$True$	$True^{9}$	$True$	$False$	$True$	$True$	$True^{10}$
N-BaIoT	$False$	$True^{1}$	$False$	$False$	$False$	$False^{4}$	$True$
BoT-IoT	$True$	$True$	$True$	$False$	$True^{9}$	$False$	$True^{10}$
New TON_IoT	$True$	$True^{9}$	$True^{6}$	$True$	$True$	$True$	$True^{10}$

1. Network configuration information not available
2. Basic captured network traces
3. No payload available; most simply reduced/summarised trace information
4. No payload available; in some packets, protocol, destination and flags deleted
5. Comprises no packet contents and no host or protocol information
6. Designed to include profiles of network information
7. Only malicious traffic
8. Does not reflect current trends
9. Contains a large number of protocols and services
10. Has modern security events and malware scenarios

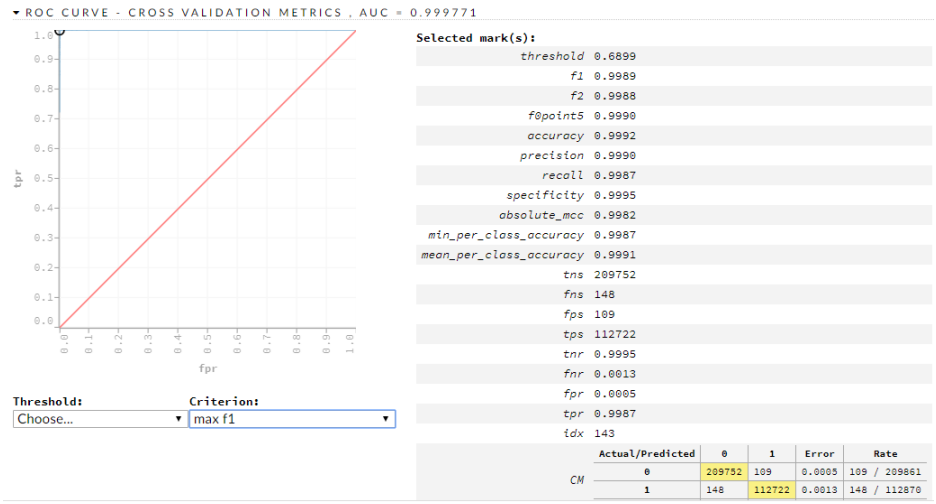


Figure 7: DNN's evaluation criteria, ROC curve, and confusion matrix on the Training_testing set of the TON_IoT network dataset

The UNSW-NB15 was designed in a realistic network that includes a variety of attack events but it includes only network traffic. The N-BaIoT and BoT-IoT datasets were designed to collect network traffic of IoT systems, with the focus of botnet events. The main drawback of both datasets is that they do not include telemetry data of IoT services that could assist to establish IoT security systems. Finally, the TON_IoT datasets were designed using a new realistic environment that includes the interaction between edge, fog, cloud layers. They also have new normal and attacking scenarios with an authentic ground truth table that verifies attack events occurred. They have complete capture of network traffic, audit traces of windows and Linux operating systems as well as IoT/IIoT data. This is a unique characteristic of the TON_IoT datasets compared with the other datasets, which is collecting four heterogeneous data sources of Windows and Linux operating systems, network traffic and telemetry data of IoT services. The four data sources were labeled using the same timestamps of attack events that exploited vulnerable nodes. The datasets have a wide variety of protocols and services such as TCP, UDP, HTTP, MQTT, SSL and ICMP. They have also the interconnections of WIFI (such as the connections of smartphones, smart TV and IoT devices) and LAN (such as the connections of VMs and their services).

The new features and characteristics of the TON_IoT datasets enable the development of federated security systems that need to learn from various datasets. Since the new datasets include heterogeneous data sources (i.e., network traffic,

telemetry data of IoT devices and audit traces of operating systems), the datasets will be employed to develop federated learning-enabled security. For example, the development of a federated learning model that can learn network traffic, audit traces of operating systems, and sensing data, to discover anomalous observations from network communications, operating systems, and IoT devices, respectively. The datasets have also enriched features in the four datasets to develop new machine learning-enabled security systems. Although the existing datasets have been mainly used to evaluate intrusion detection systems. The TON_IoT datasets have new features in the four datasets to assess the performances of multiple machine learning-based security solutions, including intrusion detection, privacy preservation, threat intelligence, threat hunting and digital forensics. The datasets can be used to assess and validate the future security solutions that would be deployed at host systems (i.e., windows and Linux operating systems), IoT/IIoT systems, network systems, fog, cloud, edge and software-defined networks. This is because of the potential design of the new testbed architecture that involved these systems and executing attacking and normal scenarios.

7. Conclusion

A new testbed architecture has been explained to illustrate the dynamic interaction of edge, fog and cloud tiers in IoT networks. The architecture was designed using the platform NSX vCloud NFV to offer SDN, NFV and SO for allowing the flexibility between linking the layers of edge, fog and cloud. The architecture was used to collect concurrent four heterogeneous data sources network traffic, windows and Linux operating systems, as well as IoT/IIoT services. New generated features and nine security events were contained in the datasets to evaluate machine learning-based cybersecurity applications. Statistics of data records and their attack types were described, with applying four machine learning algorithms on the training-testing set of the network dataset. The results revealed that applying data preprocessing and feature selection improved the performances of the machine learning algorithms. Although the outputs of machine learning are too high, this reflects the performance of a small set of the dataset in a binary classification problem of classifying normal and attack classes. The dataset has a large number of current normal and attack vectors, with the authentic ground truth of security events that can be used in the future to assess multiple AI-based cybersecurity applications, such as intrusion detection, privacy preservation, threat intelligence and hunting, as well as digital forensics.

Nour Moustafa is a leader of Intelligent Security at SEIT, University of New South Wales (UNSW)'s UNSW Canberra Australia. He was a Post-doctoral Fellow at UNSW Canberra from June 2017 till December 2018. He received his PhD degree in the field of Cyber Security from UNSW Canberra in 2017. He obtained his Bachelor and Master degree of Computer Science in 2009 and 2014, respectively, from the Faculty of Computer and Information, Helwan University, Egypt. His areas of interest include Cyber Security, in particular, Network Security, IoT security, intrusion detection systems, statistics, Deep learning and machine learning techniques. He has several research grants with totalling over AUD 1.2 Million. He has been awarded the 2020 prestigious Australian Spitfire Memorial Defence Fellowship award. He is also a Senior IEEE Member, ACM Distinguished Speaker, as well as CSCRC and Spitfire Fellow. He has served his academic community, as the guest associate editor of IEEE transactions journals, including IEEE Transactions on Industrial Informatics, IEEE IoT Journal, as well as the journals of IEEE Access, Future Internet and Information Security Journal: A Global Perspective. He has also served over seven conferences in leadership roles, involving vice-chair, session chair, Technical Program Committee (TPC) member and proceedings chair, including 2020-21 IEEE TrustCom and 2020 33rd Australasian Joint Conference on Artificial Intelligence.

References

[1] Ton_iot datasets (February 2020). URL: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-ton-iot-Datasets/.

[2] M. A. Ahad, S. Paiva, G. Tripathi, N. Feroz, Enabling technologies and sustainable smart cities, Sustainable cities and society 61 (2020) 102301.

[3] B. N. Silva, M. Khan, K. Han, Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities, Sustainable Cities and Society 38 (2018) 697–713.

[4] V. Alcácer, V. Cruz-Machado, Scanning the industry 4.0: A literature review on technologies for manufacturing systems, Engineering Science and Technology, an International Journal (2019).

[5] N. Moustafa, E. Adi, B. Turnbull, J. Hu, A new threat intelligence scheme for safeguarding industry 4.0 systems, IEEE Access 6 (2018) 32910–32924.

[6] P. O'donovan, C. Gallagher, K. Bruton, D. T. O'Sullivan, A fog computing

industrial cyber-physical system for embedded low-latency machine learning industry 4.0 applications, Manufacturing Letters 15 (2018) 139–142.

[7] J. Lee, M. Azamfar, J. Singh, A blockchain enabled cyber-physical system architecture for industry 4.0 manufacturing systems, Manufacturing Letters 20 (2019) 34–39.

[8] N. Moustafa, M. Ahmed, S. Ahmed, Data analytics-enabled intrusion detection: Evaluations of ton_iot linux datasets, arXiv preprint arXiv:2010.08521 (2020).

[9] J. Dizdarević, F. Carpio, A. Jukan, X. Masip-Bruin, A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration, ACM Computing Surveys (CSUR) 51 (2019) 1–29.

[10] N. Moustafa, B. Turnbull, K.-K. R. Choo, Towards automation of vulnerability and exploitation identification in iiot networks, in: 2018 IEEE International Conference on Industrial Internet (ICII), IEEE, 2018, pp. 139–145.

[11] T. Haq, J. Zhai, V. K. Pidathala, Advanced persistent threat (apt) detection center, 2017. US Patent 9,628,507.

[12] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, Journal of Network and Computer Applications 128 (2019) 33–55.

[13] D. Raca, Y. Sani, C. J. Sreenan, J. J. Quinlan, Dashbed: a testbed framework for large scale empirical evaluation of real-time dash in wireless scenarios, in: Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 285–290.

[14] I. Sharafaldin, A. H. Lashkari, S. Hakak, A. A. Ghorbani, Developing realistic distributed denial of service (ddos) attack dataset and taxonomy, in: 2019 International Carnahan Conference on Security Technology (ICCST), IEEE, 2019, pp. 1–8.

[15] M. A. Rahman, A. T. Asyhari, L. Leong, G. Satrya, M. H. Tao, M. Zolkipli, Scalable machine learning-based intrusion detection system for iot-enabled smart cities, Sustainable Cities and Society 61 (2020) 102324.

[16] F. Yang, S. Zhang, S. Song, R. Li, Z. Zhao, H. Zhang, A testbed for intelligent software defined security framework, in: Proceedings of the ACM Turing Celebration Conference-China, 2019, pp. 1–2.

[17] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of network-based intrusion detection data sets, Computers & Security (2019).

[18] The darpa98 and kddcup99 datasets (February 2020). URL: `http://www.ll.mit.edu/ideval/data/1998data.html`.

[19] The caida datasets (February 2020). URL: https://www.caida.org/data/.
[20] The defcon dataset (February 2020). URL: http://www.netresec.com/?page=PcapFiles.
[21] The unibs dataset (February 2020). URL: http://netweb.ing.unibs.it/ ntw/tools/traces/.
[22] The lbnl dataset (February 2020). URL: http://powerdata.lbl.gov/download.html.
[23] M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita, Towards generating real-life datasets for network intrusion detection., IJ Network Security 17 (2015) 683–701.
[24] The darpa-2009 dataset (February 2020). URL: http://www.darpa2009.netsec.colostate.edu/.
[25] The cdx datasets (February 2020). URL: https://www.usma.edu/crc/SitePages/DataSets.aspx.
[26] The ctu-13 dataset (February 2020). URL: https://www.stratosphereips.org/datasets-ctu13.
[27] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, J. K. Kalita, Packet and flow based network intrusion dataset, in: International Conference on Contemporary Computing, Springer, 2012, pp. 322–334.
[28] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, computers & security 31 (2012) 357–374.
[29] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., in: ICISSP, 2018, pp. 108–116.
[30] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot—network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Computing 17 (2018) 12–22.
[31] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems conference (MilCIS), IEEE, 2015, pp. 1–6.
[32] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future Generation Computer Systems 100 (2019) 779–796.

[33] N. Moustafa, M. Keshk, E. Debie, H. Janicke, Federated ton_iot windows datasets for evaluating ai-based security applications, arXiv preprint arXiv:2010.08522 (2020).

[34] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms, ACM Computing Surveys (CSUR) 52 (2019) 1–37.

[35] R. Piyare, S. Park, S. Y. Maeng, S. H. Park, S. C. Oh, S. G. Choi, H. S. Choi, S. R. Lee, Integrating wireless sensor network into cloud services for real-time data collection, in: 2013 International Conference on ICT Convergence (ICTC), IEEE, 2013, pp. 752–756.

[36] Vmware sdn (February 2020). URL: https://lenovopress.com/lp0661.pdf.

[37] Y. Park, N. V. Kengalahalli, S.-Y. Chang, Distributed security network functions against botnet attacks in software-defined networks, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE, 2018, pp. 1–7.

[38] Vmware vcloud nfv (February 2020). URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/nfv/vmware-vcloud-nfv-vcloud-director-edition-datasheet.pdf.

[39] P. Meyer, T. Hackel, F. Langer, L. Stahlbock, J. Decker, S. A. Eckhardt, F. Korf, T. C. Schmidt, F. Schüppel, A security infrastructure for vehicular information using sdn, intrusion detection, and a defense center in the cloud, in: 2020 IEEE Vehicular Networking Conference (VNC), IEEE, 2020, pp. 1–2.

[40] C. De Cusatis, R. Cannista, L. Hazard, Managing multi-tenant services for software defined cloud data center networks, in: 2014 IEEE 6th International Conference on Adaptive Science & Technology (ICAST), IEEE, 2014, pp. 1–5.

[41] P. Shabisha, A. Braeken, P. Kumar, K. Steenhaut, Fog-orchestrated and server-controlled anonymous group authentication and key agreement, IEEE Access 7 (2019) 150247–150261.

[42] D. Li, L. Deng, M. Lee, H. Wang, Iot data feature extraction and intrusion detection system for smart cities based on deep migration learning, International journal of information management 49 (2019) 533–545.

[43] Directories of ton_iot (February 2020). URL: https://cloudstor.aarnet.edu.au/plus/s/ds5zW91vdgjEj9i.

[44] L. N. Tidjon, M. Frappier, A. Mammar, Intrusion detection using astds, in:

International Conference on Advanced Information Networking and Applications, Springer, 2020, pp. 1397–1411.

[45] P. Hall, N. Gill, M. Kurka, W. Phan, Machine learning interpretability with h2o driverless ai, H2O. ai (2017).

[46] Code of ml techniques (February 2020). URL: https://github.com/Nour-Moustafa/TON_IoT-Network-dataset.

[47] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, J. W. Choi, Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 1672–1678.

[48] P. Verma, S. Anwar, S. Khan, S. B. Mane, Network intrusion detection using clustering and gradient boosting, in: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, 2018, pp. 1–7.

[49] N. Farnaaz, M. Jabbar, Random forest modeling for network intrusion detection system, Procedia Computer Science 89 (2016) 213–217.

[50] S. Mukherjee, N. Sharma, Intrusion detection using naive bayes classifier with feature reduction, Procedia Technology 4 (2012) 119–128.

[51] C. Xu, J. Shen, X. Du, F. Zhang, An intrusion detection system using a deep neural network with gated recurrent units, IEEE Access 6 (2018) 48697–48707.