National College of Ireland

Detecting Brute-Force Attack in IoT Device using Network Flow Data

MSc Internship
Cyber Security

Tobechukwu Treasure Osueke
x17132070

School of Computing
National College of Ireland

Supervisor: Rohan Singla

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Tobechukwu Treasure Osueke
Student ID:	x17132070
Programme:	Cyber Security
Year:	2018
Module:	MSc Internship
Lecturer:	Rohan Singla
Submission Due Date:	11/12/2017
Project Title:	Detecting Brute-Force Attack in IoT Device using Network Flow Data
Word Count:	7098

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	16th September 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detecting Brute-Force Attack in IoT Device using Network Flow Data

Tobechukwu Treasure Osueke
x17132070
MSc Internship in Cyber Security

16th September 2018

Abstract

The concept of Internet of Things has materialize the vision of a society where users, computing systems and everyday objects possessing sensing and actuating capabilities interact in a manner that is convenient and economical using the Internet. Advancement in IoT have led to the adaptation of current Internet architecture and IP-based communication protocols. Such communication protocols have been deployed with components having low-power, low-storage and constraints resources. With the massive increase in number of IoT devices accross the glob and the rapid development of enabling communication protocols, security should be considered.

Brute-force attack is one of the most prevalent attacks that targets Internet enabled devices. IoT objects have suffered from this attack type, leading to disruption of services, data confidentiality and integrity. The goal of this work is to improve the security of IoT devices by building and evaluating five models to detect brute force attack on telnet and secure shell (SSH) communication protocols respectively. The models were implemented using five different machine learning classification algorithm and then evaluated to determine which of the algorithms is more efficient and adequate. Our approach uses network flow data and puts into consideration the fact that IoT devices are composed of constraints resources.

1 Introduction

key Terms: *IoT, Brute-Force, Intrusion Detection Model, Machine Learning, telnet, SSH*

1.1 Background and Motivation

Internet of Things allows objects to have the resources needed to communicate over the internet with agreed protocols Sherasiya et al. (2016). IoT devices are built with components such as wireless sensor, actuators, radio frequency identification, WIFI, bluetooth, micro-controllers and micro processors, that supports Internet connectivity Granjal et al. (2015). This makes it possible for such objects to be accessed by anyone or by other objects at any time and from anywhere. Research have made it possible for IoT objects to use IP-based communication protocols to locate and exchange resources Sherasiya

et al. (2016). Smart objects like the Amazon echo, Smart locks and CCTV cameras are now being used in homes of consumers to automate several tasks. The transportation, manufacturing and health-care industry have also benefited from the possibilities that Internet enabled objects brings. Our society have been greatly impacted and transformed by IoT. According to research, it has been estimated that IoT devices will consist of almost 212 billion objects with significant home and business applications in 2020 Al-Fuqaha et al. (2015). In todays computing environments, any device on the internet is under constant threat of cyber-attacks, and IoT objects are not exempted Sicari et al. (2015). With the rapid growth of the use of IoT devices in today's world, hackers have taken advantage of this to perform malicious actions. This has raised security concerns relating to data confidentiality, integrity and service availability among researchers, manufacturers and consumers Sicari et al. (2015). A question like 'how can we detect an attempt to compromise the confidentiality, availability and integrity of IoT devices'? have been asked Raza et al. (2013); Kasinathan et al. (2013).

The Internet Communication Infrastructure of IoT have evolved to encompass low-energy, low-computing and low-storage resources. Although most IoT devices still use conventional network protocol such as Telnet, IPv4, IPv6 and SSH, they are peculiar in their behaviors due to constraint resources that makes up these devices. IoT network nodes are made up of sensors/components that operate using constraints resources. This has motivated research in designing light weight network protocol to support internet based communication in constraints resources . Due to the novelty of technologies and components that empowers the Internet of Things, traditional security mechanisms seems inadequate for IoT devices. Hence, the need to research on how the security of these technologies can be improved. Granjal et al. (2015); Al-Fuqaha et al. (2015)

Detecting attacks on an Internet enabled device is said to be an effective approach in improving the security of such device. Intrusion detection is the process of detecting any attempt to compromise the availability, integrity and confidentiality of an internet resource. The use of heterogeneous components, constraints network nodes and low power/energy network protocols makes the use of conventional security Intrusion Detection System ill-suited for today's IoT devices Sicari et al. (2015) Raza et al. (2013). IDS is a mechanism that aids the identification of an intrusion on specified network/device. In designing an IDS for IoT network, the following should be considered: The network protocol/service, computational and storage power of the device, and the type of attack to be detected Granjal et al. (2015).

Brute-force attack is one of the most prevalent kind of attacks targeting IoT devices Sha et al. (2018). In a brute-force attack, the attacker tries to gain access to a device by trying different password or passwords on open TCP ports. The attacker's strategy is to try different possible combinations of user-names and passwords until the right combination is found. This type of attack is done using automated tools that tries different combinations of passwords in an attempt to gain access into the target device. A major reason for the success of this kind of attack is because lots of deployed IoT devices still uses default login credentials on services such as telnet and SSH Sicari et al. (2015). This makes it easy for an attacker to brute-force IoT device by just trying few well-known factory setting credentials. In August 2016, the Mirai malware targeted IoT devices running Linux and succeeded in gaining admin access by brute forcing these devices using a table of 63 common default login credentials. Antonakakis et al. (2017).

1.2 Purpose

The purpose of this work is to find out with what level of accuracy a host-based model built with Network Flow Data can identify if an incoming network frame/data is a brute-force attack or not. Large number of IoT devices are being manufactured with remote access capability over services like Telnet and SSH. A model to detect brute-force attack is of importance. Security mechanism for IoT should be designed to suite the current network and communication protocols being employed Sherasiya et al. (2016). In this work, we focus on detecting brute force attack on Telnet and SSH protocols because these services poses serious danger if compromised. It has been reported that over 40,000 unique IP addresses launches brute force attack against telnet port on a daily basis on IoT devices, which demonstrate the fact that most IoT objects still have telnet service open Hellemons et al. (2012). In conducting this research, we performed deep-packet investigation of network flow data to identify what features can help make our detection model efficient and accurate to detect brute-force attack Granjal et al. (2015). In building our classification model we implemented five algorithms and then performed a test to evaluate which of the models gave better results. Our models are built with data from three different scenarios and light weight machine learning algorithms, as you will see in section 3.

1.3 Research Question

- Question 1: What level of accuracy can network flow data be used to detect brute-force attack with less false negatives? Network flow data contains significant data that can give insights about communicating devices. Informations such as source and destination IP, frame size or protocols can help understand the behaviour of communication between IoT devices. An understanding of how IoT devices behaves over telnet and ssh protocols can provide useful insights that will help improve their security. This study, investigates the behaviour of IoT device using the network flow data. It aims at engaging the use of relevant data gotten from network flow to build a detection model that can help detect a brute-force attack. This is useful because IoT devices do not have large storage capacity to log failed login attempts to detect brute-force attack just like conventional systems. Therefore engaging the use of network flow data to accurately detect brute-force attack will eliminate the need to log failed login attempts and consequently support constraint devices.

- Question 2: Which classification algorithm being used gives the best accuracy for our model? There are a number of machine learning classification algorithms. Each algorithm perform differently when provided with the same data. There performance varies with the kind of data. In tackling certain problems, the performance of one algorithm is different from others. It's therefore imperative to investigate which of the classification algorithms performs more accurately with less false negatives in our situation.

The rest of this paper is structured as follows. Section 2 gives a brief background of relevant terms and reviews of related works. In reviewing previous works, emphasizes was placed on the following: network protocol supported, type of attack detected, IDS type, detection method/approach and their support for constraint resources. In section 3 we explain our solution and methodology. Section 4, 5 and 6 details implementation, evaluation and conclusion/future works respectively.

2 Related Work

2.1 Internet of Things: Enabling Technologies

The Internet of Things is made up of softwares, sensors, actuators, embedded electronics and connectivity that enables physical objects to exchange data in real time Gubbi et al. (2013). This development has led to the extension of Internet connectivity beyond conventional devices, like laptop and desktop computers, to involve everyday physical objects such as Vehicles, and security cameras. IoT objects mainly composed of four components:

- Sensors/hardware/perception
- Connectivity/network
- Data Processing/middleware
- User Interface

The hardware or sensor collects data from the physical environment Al-Fuqaha et al. (2015). An example of this is a security camera that captions images from it environment. These sensors can be actuators, smart sensors or wearable sensing devices. Single Board Computers such as Raspberry PI can be integrated with sensors and built-in TCP/IP functionalities to make an IoT product Al-Fuqaha et al. (2015). The connectivity components is responsible for internet connection to send captured data from one end-point to another. Examples of network components are WIFI, Bluetooth, low-power wide-area network (LPWAN),via Ethernet, IEEE 802.15.4, Radio-Frequency Identification (RFID), and Z-waze. The captured data is then processed by the data processing component and presented for visualization to the user through a presentation interface Farooq et al. (2015).

The identification of IoT devices is crucial to it's functionality. Before now, RFID tags and electronic product code was used for unique identification, however, this has evolved into IP-based address. IPv4 and IPv6 addressing methods ensures that each device has a unique address and can be access globally using the IP address Al-Fuqaha et al. (2015). So, when transmitting information over the internet each device are uniquely identified by it's IP address. This growth support the scalability of IoT devices as an increasing amount of IoT objects are developed and deployed yearly. This also makes IoT devices to be easily targeted by attackers due to the fact that by just scanning for available IP addresses, hackers can identify these devices and check for attack vectors or openings.

When compared with standard computing devices like desktop computers the components that enables IoT are of constraint resources Al-Fuqaha et al. (2015); Granjal et al. (2015). Standardization body like IEEE and IETF have design protocols such as 6LowPan to support the low-power, low-computing and low storage IoT components Gendreau and Moorman (2016). This is a major reason the authors in Zarpelão et al. (2017) emphasized the need to investigate the network behaviour of IoT network components in-order to design and build a suitable security mechanism for IoT devices.

2.2 Brute Force Attack

The rate at which brute-force attack occur in IoT devices have been investigated in previous work. Antonakakis et al. (2017) studied the number of IoT objects that have

been attacked using brute-force in a space of 2016-2017. The authors concluded that over 300,000 IoT devices experienced brute-force attack during the specified period. Kolias et al. (2017) carried out an investigation of brute-force attack on smart home devices and concluded that every internet enabled object is likely to experience brute-force attack.

Two major protocols that have been said and seen to be targeted by attackers are Telnet and SSH Pa et al. (2015); Kolias et al. (2017). Telnet protocol is a service that allows users to communicate with a remote device Pa et al. (2015). It provides a virtual terminal used to remotely access and control IP-based IoT devices through TCP port 23. Due to the weakness of Telnet protocol, in that it sends data in clear text, Secure Shell (SSH) protocol was designed Hellemons et al. (2012). Just like Telnet, SSH provides remote access to devices through TCP port 22. Though researchers has spoken against the use of Telnet protocol, lots of already deployed IoT devices still have Telnet protocol running on them Antonakakis et al. (2017). A successful brute-force attack on Telnet and SSH protocol gives access to all data stored on the device and an attacker can infect the device with a malware which can make the device to be used for other form of attacks such as Denial of Service, Botnet or Distributed Denial of Service Kolias et al. (2017).

A vulnerability that exist in IoT objects due to low-storage and computing power is their inability to log failed login attempts. Hackers has taken advantage of this vulnerability to gain control of IoT devices by brute-forcing the login credentials. Conventional brute force detection mechanism works by logging failed login attempts and checking it for multiple failed login attempts Najafabadi et al. (2014). This approach is based on the fact that hackers engage the use of automated software in a brute force attack. The software will try testing more wrong passwords than a legitimate user who has forgotten their password in a specified time interval. Sicari et al. (2015) argues that this method can be effective in IoT devices, but considering the fact that IoT devices are made with components that has constraint memory space this method appears to be inadequate.

2.3 Intrusion Detection System (IDS)

Intrusion detection systems have been used by information technology experts to provide security for IT infrastructure and resources Zarpelão et al. (2017). IDS aids the detection of cyber attacks on a network or on a system by analyzing the activity on the network or on the system Raza et al. (2013). An IDS involves three kind component: The sensor engine, which is responsible for listening and capturing activities on the network or system. Secondly, the analysis engine that analyze the network data. Thirdly, the reporting system that displays information to the user Zarpelão et al. (2017).

An IDS can either be set up at the network layer or within the IoT device itself(also known as host-based) Sherasiya et al. (2016). These IDS can function using the signature of previously known attacks or can be anomaly-based (that is, checking for anomaly network activities on the network or host machine) Jonsdottir et al. (2017).

In-respective of the significant growth in IDS technology for conventional networks, these IDS solutions are considered ill-suited for IoT devices, because of IoT devices components have constraint resources Gendreau and Moorman (2016). The network nodes that host traditional IDS agent are expected to have very high computing power but IoT network layer components usually composed of nodes with constraints resources, thereby making it a challenge to have the required support needed to work well with traditional IDS Raza et al. (2013). Keeping in mind this novel requirement of IoT, in this paper, we focus on building and evaluating an host-based brute-force detection model that checks

for anomalies in network behaviour using machine learning classification algorithms.

Najafabadi et al. (2014) designed and evaluated brute-force detection models built using four different classifiers algorithms. According to the authors, the dataset used in building the model was collected over a 24 hours period from a production network comprising of approximately 300 users, four different subnets and multiple servers. After critical evaluation of their approach, it was seen to lack support for devices built with constraint resources in that the experiment was performed using conventional systems Gendreau and Moorman (2016). Therefore, its requires high computational, power and storage network components and as such inadequate for Iot devices.

In order to design and build an IDS suited for IoT devices, the authors in Raza et al. (2013) presented a solution named SVELTE. SVELTE is both a signature-based and anomaly-based IDS. According to the researchers SVELTE was designed to be placed at 6LoWPAN border router[1]. Based on the authors validation, SVELTE was efficient in detecting touting attacks like spoofing and sinkhole. Though the authors Raza et al. (2013) argued that SVELTE provides support for ipv6 constraint networks and works well to optimize memory, Surendar and Umamakeswari (2016) believes that SVELTE requires high storage and computing power due to the use of large volume of data in the detection/analysis engine. Another key point was that SVELTE proposed a distributed placement strategy which will ultimately led to large used of the resources Surendar and Umamakeswari (2016).

After completing a survey of IDS for IoT systems between 2009-2016, the author Zarpelão et al. (2017), discovered that previous solutions focused on detecting sink hole, denial of service(DoS), and man-in-the-middle spoofing attacks. Due to recent development these range of attack detection does not seem to satisfy the security needs of IoT devices as more attacks types have been discovered. This has led to discussion around the need for more work to be done in respect to designing detection mechanism for IoT network.

The architecture of IoT devices can be improved to incorporate security mechanism Kasinathan et al. (2013). Work has been done to propose an architecture that incorporates an open source signature based-IDS name suricata with the network manager framework of ebbits[2]. This IDS mechanism was designed to detect denial of service attacks by using a pron to sense and transmit network data to the analysis engine. Although this system was proven to support low-computing capacity nodes, it's deployment model requires large memory and it's use is restricted to ebbits framework Zarpelão et al. (2017).

Thanigaivelan et al. (2016) described a distributed internal detection model for IoT. The idea was to listen and classify frame packets in one-hop neighbor nodes of the IoT network. The authors did not provide details about the process used to build the normal behavior profile and how the detection technique works on constraint components was not clearly specified.

Machine Learning and IDS

Based on research, machine learning algorithms are productive and efficient in designing IDS Jonsdottir et al. (2017). Machine learning propagates the idea that computing devices can learn from experience by training the system using data associated with past occurrences. These experiences are inputted as data into a machine learning algorithm

[1] `https://www.micrium.com/iot/devices/`

[2] `http://www.ebbits-project.eu/`

for training. The more data inputted into the system, the better the machine learns and will be able to make accurate predictions Cañedo and Skjellum (2016).

The building process of an IDS requires a high volume of data and an efficient machine learning algorithm Nguyen and Armitage (2008). To organize and classify these data to create an intelligent system, we need an algorithm. Cañedo and Skjellum (2016) believes adopting machine learning principles in building intrusion detection system for IoT devices will provide adequate support for IoT technologies as it provides light weight algorithms that can be used to train large network flow data.

Cañedo and Skjellum (2016), used a machine learning algorithm known as Artificial Neural Network (ANN) to design an IDS for detecting routing attacks. Nguyen and Armitage (2008) made used of a unsupervised machine learning algorithm to model an IDS for Internet of Things network to detect Denial of Service (DoS) attack. There is need to be careful when choosing a machine learning approach even though machine learning algorithms have been said to be effective for designing IDS. The reason for this is some algorithms requires high computing power and hence will not work well in IoT devices Midi et al. (2017).

This paper focus on building a host-based detection model to identify brute-force attack on Telnet and SSH protocol. We based our hypothesis on network flow data having some features that indicates a brute-force attack. We therefore engaged in the investigation of network frames and packets data on a constraint device and applied light weight machine learning algorithms to train a model that detects incoming network frames as brute-force or not a brute-force attack. This approach works best for IoT objects because the size of the packet data are relatively small and are computed with less power and requires low memory for storage Cañedo and Skjellum (2016). In the next session, details of the applied methodology used in this paper is provided.

3 Methodology

3.1 Brute-Force Detection Models

Recall that prior to the development of IP-based (IPv4 and IPv6) communication protocol, RFID tags and electronic product code was used for unique identification Al-Fuqaha et al. (2015). IP-based communication protocol makes IoT devices to be globally accessible and thus if security measures are not put in place, malicious users can gain remote access and therefore take control of the device. Also, considering the fact that when compared with conventional systems, components that makes-up IoT are of constraint resources (low energy, computing and storage power).

In this session, a description of the main steps taken to design, build and evaluate the proposed brute-force detection models is presented. Our experiment was done using IPv4 and this is due to the discovery that large amount of IoT devices being deployed globally uses IPv4 address, even though recent development of 6LoWPAN (IPv6) introduces more scalable address system. In conducting our empirical studies, we first set-up an experiment lab which compose of an IoT device and needed software tools. Secondly, we captured network packet by simulating three different scenarios on our IoT device. Thirdly, we perform deep packet inspection and comparison of data from the different scenarios. Fourthly, network flow data is extracted from the captured packet using Wireshark traffic analysis tool. After extracting the flow data we labeled each frame accordingly before building the detection models. Five machine learning classifiers

are used to build the classification models on the labeled network flow data. Towards testing and evaluating our models we used 20% of our dataset to see the rate of accurate predictions (known as Repeated Random subsampling testing technique). We checked for accuracy rate, false positive and false negative using confusion matrix in order to determine the algorithm that gives the best performance. Below, further details of these steps are provided.

3.2 Experiment Lab Set-Up

To build the detection model, network flow data from an IoT device relating to telnet and ssh protocol network activities and brute-force attack is required. The first consideration was to access already captured network flow data available publicly on the internet. A very good example which is used in most research paper is KDD-Cup data. KDD-Cup data is a well known intrusion detection data-set use to build predictive model capable of detecting wide range of attacks such as buffer overflow, nmap probe, DoS, ftp write and portsweep probe. The KDD-Cup dataset was collected from a military network environment in which a wide variety of intrusion was simulated. Other stated examples of intrusion detection dataset are: ADFA dataset [3] Creech and Hu (2014), NSA dataset [4] and publicly available pcap files at NetReSec [5]. The key challenge with the above-specified datasets is they don't meet our project specification. As earlier specified the goal of this project is to build a brute-force detection model for IoT device, we therefore, require dataset comprising of simulated brute-force attack and network activities on ssh and telnet port from constraint IoT device. Hence the need to build an experiment lab comprising of an IoT device and tools as you will see shortly.

The following tools made up the experiment lab:

- Raspberry Pi 3.
- Wireshark, Advance IP Scanner, PuTTy and MobaXterm running on Laptop Computer.
- Ethernet Cable, External Mouse, HDMI Cable and Visual Display Monitor.
- Kali Linux Running on Oracle VM VirtualBox.

1. Raspberry Pi 3:

   Raspberry Pi is a credit card size single board computer designed to teach young people how to program [6]. The first version of Raspberry Pi was released in February 2012 and since then it has become more popular than expected. It has been reported that over 14 million Raspberry Pi has been sold. A major reason for this growth is due to its wide variety of application areas. All models of this device comes with an integrated ARM central processing unit, random access memory (RAM), space for storage micro SD card and on-chip graphics processing unit (GPU). Latest models comes with 40 general-purpose input/output pins that can be integrated with other objects. Due to it's design, the Raspberry Pi board can be integrated with

[3] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-IDS-Datasets/
[4] https://www.westpoint.edu/crc/SitePages/DataSets.aspx
[5] https://www.netresec.com/index.ashx?page=PcapFiles
[6] https://www.raspberrypi.org/help/videos/

sensors to realize a functional IoT product Al-Fuqaha et al. (2015). In the past, IoT products such as home automation system and temperature monitoring system has been implemented using Raspberry Pi. Organizations, individuals and researchers take advantage of this device due to its size, cost, portability, programmability, networking capabilities and wide range of possible projects.

In this study, we use Raspberry Pi 3 model B+ as an IoT device. There are a couple of reasons why we made this choice, they are: (1) With a linux operating system running on this device we could set up telnet and ssh service on the device. (2) The visual display capabilities of Raspberry Pi also contributed to the choice. (3) Raspbery Pi 3 model B+ comes with ethernet and an on-board WIFI which makes it possible to be connected to a network and can be accessed remotely using it's IP address. (4) The low-cost of Raspberry Pi is also a motivation. (5) As we seek to perform our experiment on devices with constraint resources, we find Raspberry Pi to be a suitable choice due to it low-power (can be powered on using phone charger), low-storage and low-computing capacity. When compared with standard desktop and laptop computers, Raspberry Pi components are of constraint resources. We feel that the Raspberry Pi is a solid choice for our empirical study as it provides us with the required specification.

A Linux operating system known as Raspbian was installed on the Raspberry Pi. We also setup telnet and ssh service/server on the device. Lastly we installed a command line network packet analyzer called tcpdump.

2. Wireshark, Advance IP Scanner, PuTTy and MobaXterm running on Laptop Computer:

   Wireshark is a free and open source tool that is widely used to analyze network protocol. It gives users insight into what is happening on the network at an interactive and detailed level. This tool can run on Unis, Linux and Microsoft Windows operating systems. Wireshark[7] provides a graphical user interface and information filtering features, which makes it different from tcpdump. Though Tcpdump is an efficient packet analyzer it works on command line interface. In this paper, Wireshark is used to perform deep packet analysis of the captured network traffic

   Advance IP Scanner was installed on a laptop computer to simulate the situation in which an attacker scans for available devices on the network. This tool is a reliable and free network scanner that shows all network devices [8]. It is particular popular in the scientific community because it is free and easy to use.

   PuTTy and MobaXterm are used as ssh and telnet client to establish remote connection to the IoT device. PuTTy is the most used software for remote connection. These tools are standard and freely available software.

3. Kali Linux Running on Oracle VM:

   Kali [9] is a debian-based linux distribution designed for advanced security auditing and penetration testing. Kali contains more than 600 penetration testing tools and it is widely accepted as a standard tool in the information security community Muniz (2013). In comparison with other Linux distribution we prefer Kali because

[7] `https://www.techopedia.com/definition/25325/wireshark`
[8] `https://www.advanced-ip-scanner.com/`
[9] `https://docs.kali.org/introduction/what-is-kali-linux`

it provides tools that are required for our study. Tools that are of particular interest to this project are hydra and metasploit. Hydra [10] is said to be the tool of choice when it comes to brute-force attacks Stiawan et al. (2016). Metasploit [11] is also a globally recognized tool that is used by security experts to check for vulnerabilities on system by simulating various attacks. Our choice for these tools is because these tools are used by both cyber criminals and security experts to perform brute-force attacks globally Rodas and To (2015).

How all these devices/tools connect to each other

3.3 Data Collection

The target dataset is network flow data. Network flow data represent what is known as network sessions Najafabadi et al. (2014). These sessions are composed of an aggregated number of packets that possess certain properties within a specified time interval. These properties are known as packets features and some common examples are protocol type, source port, destination port, source and destination IP address Najafabadi et al. (2014).

A key question that was asked during project development was why network flow data? In reviewing literature, we discovered that by analyzing and monitoring relevant network data, the authors Kasinathan et al. (2013) built a DoS detection model. Also, By analyzing the network packet data, Cañedo and Skjellum (2016) detected anomalies in IoT network using artificial neural network (ANN) algorithm. The network packet contains information such as source and destination IP, sequence code and payload size that when accurately analyzed will aid intrusion detection Zarpelão et al. (2017).

We built and evaluated two category of models using five different machine learning algorithms: SSH and Telnet brute-force detection models. For both ssh and telnet detection models, network packets were captured using tcpdump from three different simulated scenarios on the Raspberry Pi:

- Scenario One: Normal (non brute-force) remote network login activities on telnet and ssh port (23 and 22 respectively) using correct credentials.
- Scenario Two: A situation in which a user input wrong password either because of mistake or the password is forgotten.
- Scenario Three: Simulated brute-force attack using a list containing 64 pairs of well-known/default user-name and password. This list contains the username and password pairs that were used in the 2016 Mirai attack and also some well know default credentials for Raspberry Pi.

These different scenarios were done repeatedly over a 48 hours period and the captured data saved as a packet capture (pcap) file. Also, each of these scenarios were separately done for both telnet port 23 and ssh port 22. The reason for this simulations is to ensure that our models are built using data representing or near real world situations and this is also why tools like PuTTy, hydra and metasploit were engaged. Step by step guide to show how each of these scenarios were simulated is provided in the configuration manual. In the following sub-subsection we state what tools were used in each scenarios.

[10] `http://sectools.org/tool/hydra/`
[11] `https://www.metasploit.com/`

PuTTy:

PuTTy was used for both scenario one and two. We remotely connected to Raspberry Pi using both telnet and ssh port. Advance IP scanner was used to scan the network in-order to identify the IP of the device.

Kali Linux Terminal:

Kali terminal was also used for scenario one and two. The reason for this was to investigate the differences in network packets representing traffic coming from both Kali and PuTTy. Also we wanted to ensure that our models are built with data from multiple source, so as to ensure better performance.

Hydra and Metasploit:

As already explained metasploit is a penetration testing tool used to simulate several kind of attacks. Hydra is one of the most used tools for performing brute-force attack. Both Hydra and Metasploit were used for scenario three.

3.4 Data Inspection and Feature Identification

The captured pcap files was copied to a laptop computer and opened using Wireshark for visual/manual comparison. Each of the pcap files gotten from the scenarios were viewed in Wireshark and a manual comparison was done to look out for differences in the TCP/IP stack. By looking at the files using Wireshark graphical user interface we were able to identify features/fields that were variables. This technique seem appropriate for our situation because we aimed at figuring out the differences between network flow data from non brute-force attack traffic and network flow data from brute-force attack traffic. Through this technique we discovered that there were some inconsistencies in fields like Frame-Length, Time, Time delta from previous captured frame, TCP Segment Len, Window size value, Calculated window size, and Maximum segment size. An interesting discovery was that the size of frames in the brute-force attack packets was larger than that of non brute-force attack flow data.

These inspections gave us insights into the behaviour of network flow data and we were able to identify features that we used to build our models. It is important to use the right features/variables when building a detection model as it can contribute to increasing the performance of the model Ni et al. (2017).

Wireshark has an inbuilt functionality that let us specify fields and extract them into a comma separated values (csv) file format. So, we engaged the use of wireshark to export network flow data for each pcap files into csv file format because we built our models to expect such file type. Since we have identified the variables or features that will be used for our model, in the future work we recommend using a script to specify the features and extract them from the pcap file into a csv file. More details about this in section 6.

3.5 Data Labeling

The goal of our model is to classify a network frame as a brute-force attack or not a brute-force attack. Hence, the need to accurately label the dataset into different classes.

The concept of labeling in machine learning allows us to tag each instance of the dataset with one or more labels Nguyen and Armitage (2008). Labeling simply means taking a set of unlabeled data and assigning a label to it. In our case we created an extra field called marked and assigned 1 to non brute-force attack instances and 0 to brute-force attack data.

3.6 Machine Learning Methods

Machine learning is broadly categorized into supervised, unsupervised and reinforcement learning. Each of these three classes have different sub categories Cañedo and Skjellum (2016). In this study we focus on the supervised learning technique. We choose five supervised classification algorithms: Logistic Regression, Decision Tree, Random Forrest, k-Nearest Neighbor, and Support Vector Machine. These classifiers/learners were chosen because of the following:

1. Due to the time constraint, we require algorithms that works with little data preparation.
2. These algorithms are simple to use and understand.
3. Not all machine learning algorithms can handle both numeric and categorical data, the selected algorithms works well with both scientific data types.

3.7 Programming Language and Development Environment

Python 3.6 was used to build the models. Research community has embraced python because of it provides libraries and modules that can be used to quickly automate task Pedregosa et al. (2011). These modules makes use of large state-of-the-art machine learning algorithms for building supervised and unsupervised models. The development IDE used is Spyder. Spyder is an Integrated development environment used to write and run python code. It provides good display of data and ease of navigating files stored in the working directory.

4 Implementation

This section describe the development of brute-force detection models for both telnet and ssh protocol. The concepts and steps described below was applied for both telnet and ssh models. So we will not discuss them separately as they are the same for both protocols, but in the evaluation section we will discuss the evaluation of each models for both protocols (telnet and ssh) separately.

4.1 Data Preprocessing

Data preprocessing is an important aspect of any machine learning process to improve the accuracy of the result Najafabadi et al. (2014). It involves transforming data into a format that our classifier can read and process. This is needed because the dataset can be incomplete, meaning some rows might have missing values or there might be some form of imbalance in the dataset. Data preprocessing comprises of cleaning, transformation, standardization, and feature selection. Kotsiantis et al. (2006).

After capturing data, identifying relevant features, labeling individual data instances and exporting the data to csv file format, the next step is to start preprocessing our data in python to make it fit into our classifier for best performance. The total number or rows for the telnet dataset is 914. The total number of rows for the ssh dataset is 3121. Below are the actions taken to prepare our dataset for machine learning classifiers.

Importing the dataset:

It is important to make the data available for processing. We used the Pandas library in Python to import the dataset.

Feature selection:

In every scientific experiment like this, there are two main variables: independent and dependent variable [12] . The independent variable is a variable that when it is changed it affects the dependent variable. That is, a change in the independent variable will or have an impact on the dependent variable.

In building our models, we had to specify independent variables (also called matrix of features) and dependent variables (known as feature vector). Out of the 19 features/fields of the dataset 18 were selected to be part of the matrix of features, while 1, which is the labeled field was specified as the dependent variable. The reason for this selection is so our classification algorithms will learn the correlation between the independent variables and the dependent variable.

Taking care of missing data:

Missing values can affect the performance of our model negatively. We discovered that some rows had fields with no value. The from Imputer method in sklearn.preprocessing class was used to fit and transform fields with missing values using the mean replacement strategy.

Taking care of Categorical Data:

Categorical data are data fields that can contain one of a restricted number of possible values. In our case, the protocol field could either take on one value from two possible values. Python provides two methods: LabelEncoder and OneHotEncoder that is used to assign binary values to each possible values. These binary values are not given mathematical meaning.

Data Standardization:

In the dataset, the range of values of feature data varies widely and this will cause the classification algorithm not to work properly. Support Victor Machine for example, works by calculating distance between two values of features by the Euclidean distance. If one of the feature happens to have a wide range of values, the calculated distance will be dominated by the feature containing the value. Kotsiantis et al. (2006)

We use a python method called StandardScaler to standardize our data. This will make the features to be rescaled.

[12]https://www.thoughtco.com/independent-and-dependent-variables-differences-606115

4.2 Applying Classifiers

With Python applying machine learning classifiers is relatively simple. It involves knowing the right class, method and attributes to set. Each of the classifiers are contained in the sklearn library. After importing this library, the class and method are imported. The method object is created with the required attribute. Then, using the object the fit method is called and applied to the training dataset to train the model.

Logistic Regression: The logisticRegression method was used to train the training data with the random_state set to 0.

Support Vector Machine: The SVC method was applied to build the model.

Decision Tree: The criterion attribute of the DecisionTreeClassifier method was set to entropy and fit into the training data to train the model.

Random Forrest: We experimented using different number of trees to determine which one gives better accuracy. 10 number of trees was used with RandomForestClassifier method.

K-Nearest Neighbors: 8 K neighbors were used with the KNeighborClassifier method. This value was chosen after engaging in a try and error technique to see which value of K gives better result.

5 Evaluation

5.1 Testing and Validation Method

Checking for the validity of a model is an important aspect of building a model. After fitting the training data into the classifiers, how can we test or validate that our model will work accurately for real data that it has never seen before? Cross validation is a useful method for assessing the effectiveness of a model Nguyen and Armitage (2008). It is mainly used in prediction problems where a model is built to accurately classify a data instance. Though there are several types of cross validation techniques, in this study Repeated Random sub-sampling is used.

In repeated random sub-sampling (also known as Monte Carlo cross-validation) the dataset is split into training and testing/validation data. The training data is used to build the classification model, while the testing dataset is used to test the validity of the model. In conducting this experiment we used 80% of the dataset as training data, while 20% as testing data.

5.2 Models Comparison

Five machine learning classifiers were implemented for both the ssh and telnet detection models. The question is which of the classifiers gives the best result? A vital criterion used to different between machine learners is the predictive accuracy. That is, how accurately does the model makes prediction when presented with new data. Also, aside the accuracy

of the detection model, it is vital that an attack detection model gives less false negatives and false positive. In clear terms false positive refers to a situation in which there is no attack but the model says there is an attack, while false negative refers to when there is an actual attack but the model says there is no attack. Nguyen and Armitage (2008)

To check for the accuracy of the models, we used Confusion matrix. Confusion matrix (also known as error matrix) is used by researchers because of it simplicity Nguyen and Armitage (2008). It provides visualization of the performance of an algorithm. The column in the matrix represents the actual class, while the row represent instances in the predicted class Cañedo and Skjellum (2016). In the following sections we outline and discussed the result gotten from the confusion matrix of both the telnet and ssh detection models.

Telnet Models Result Discussion:

Table 1 shows the result of the telnet models.

Models	Incorrect Predictions	False Positive	False Negative	Accuracy Rate(%)
Logistic Regression	8	7	1	96
Support Vector Machine	9	5	4	95
Random Forrest	8	6	2	96
Decision Tree	3	3	0	98
K-Nearest Neighbors	11	3	8	94

Table 1: Telnet Detection Models Evaluation Results

By looking at the incorrect prediction and accuracy field in table 1, one will state that the best performing model is the decision tree classifier. But for a detection model these should not be the only criteria used to determine the efficiency of a model. It also appears that when compared with other model, the decision tree has less false positive and no false negatives.

SSH Models Result Discussion

Models	Incorrect Predictions	False Positive	False Negative	Accuracy Rate(%)
Logistic Regression	27	2	25	95
Support Vector Machine	26	5	21	96
Random Forest	10	6	4	98
Decision Tree	9	4	5	96
8-Nearest Neighbors	32	0	32	94

Table 2: SSH Detection Models Evaluation Results

Table 2 presents the results gotten from the ssh detection models. Random Forrest displayed more efficiency with a 96%, 6 false positive and 4 false negative. Though 8-Nearest neighbors has 0 false positive, it has high number of false negative which makes it not suitable in terms of accuracy. After random forest the next well performing model is the decision tree classifier. This classifier is relatively easy and learns using a tree like format.

6 Conclusion and Future Work

In this study we have attempted to show that network flow data can be used to build a brute-force detection model for both telnet and ssh protocols. We engaged in simulating three different scenarios to get our dataset. An interesting discovery of this work is that network flow data from non brute-force attack contains different values from network flow data coming from brute-force attack. These variables were used by the detection classifiers to learn the patterns of both classes of data. We believe this is a step towards improving the security of IoT devices running telnet and ssh services.

In this paper, work was limited to building a detection model, in the future, work can be done to enhance this model into a complete host-based IDS for IoT devices. This will involve capturing the network traffic in real time, selecting the specified features, extracting the features into a csv format and inputting the data into the model. Also, the storage capacity should be considered.

Acknowledgment

Firstly, I will like to express my heart felt gratitude to God Almighty for giving me the grace, strength and wisdom to see this through. My appreciation also goes to my supervisor Rohan Singla, who provided guidance during the course of this project. I also want to thank my parents and friends for their support during this journey. Finally, I will like to appreciate all Cyber Security faculty.

References

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications, *IEEE Communications Surveys & Tutorials* **17**(4): 2347–2376.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M. et al. (2017). Understanding the mirai botnet, *USENIX Security Symposium*.

Cañedo, J. and Skjellum, A. (2016). Using machine learning to secure iot systems, *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, IEEE, pp. 219–222.

Creech, G. and Hu, J. (2014). A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns, *IEEE Transactions on Computers* **63**(4): 807–819.

Farooq, M. U., Waseem, M., Khairi, A. and Mazhar, S. (2015). A critical analysis on the security concerns of internet of things (iot), *International Journal of Computer Applications* **111**(7).

Gendreau, A. A. and Moorman, M. (2016). Survey of intrusion detection systems towards an end to end secure internet of things, *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*, IEEE, pp. 84–90.

Granjal, J., Monteiro, E. and Silva, J. S. (2015). Security for the internet of things: a survey of existing protocols and open research issues, *IEEE Communications Surveys & Tutorials* **17**(3): 1294–1312.

Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions, *Future generation computer systems* **29**(7): 1645–1660.

Hellemons, L., Hendriks, L., Hofstede, R., Sperotto, A., Sadre, R. and Pras, A. (2012). Sshcure: a flow-based ssh intrusion detection system, *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, pp. 86–97.

Jonsdottir, G., Wood, D. and Doshi, R. (2017). Iot network monitor, *Undergraduate Research Technology Conference (URTC), 2017 IEEE MIT*, IEEE, pp. 1–5.

Kasinathan, P., Pastrone, C., Spirito, M. A. and Vinkovits, M. (2013). Denial-of-service detection in 6lowpan based internet of things, *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, IEEE, pp. 600–607.

Kolias, C., Kambourakis, G., Stavrou, A. and Voas, J. (2017). Ddos in the iot: Mirai and other botnets, *Computer* **50**(7): 80–84.

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006). Data preprocessing for supervised leaning, *International Journal of Computer Science* **1**(2): 111–117.

Midi, D., Rullo, A., Mudgerikar, A. and Bertino, E. (2017). Kalisa system for knowledge-driven adaptable intrusion detection for the internet of things, *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, IEEE, pp. 656–666.

Muniz, J. (2013). *Web Penetration Testing with Kali Linux*, Packt Publishing Ltd.

Najafabadi, M. M., Khoshgoftaar, T. M., Kemp, C., Seliya, N. and Zuech, R. (2014). Machine learning for detecting brute force attacks at the network level, *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on*, IEEE, pp. 379–385.

Nguyen, T. T. and Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning, *IEEE Communications Surveys & Tutorials* **10**(4): 56–76.

Ni, X., Huang, H. and Du, W. (2017). Relevance analysis and short-term prediction of pm2. 5 concentrations in beijing based on multi-source data, *Atmospheric Environment* **150**: 146–161.

Pa, Y. M. P., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T. and Rossow, C. (2015). Iotpot: analysing the rise of iot compromises, *EMU* **9**: 1.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011). Scikit-learn: Machine learning in python, *Journal of machine learning research* **12**(Oct): 2825–2830.

Raza, S., Wallgren, L. and Voigt, T. (2013). Svelte: Real-time intrusion detection in the internet of things, *Ad hoc networks* **11**(8): 2661–2674.

Rodas, O. and To, M. A. (2015). A study on network security monitoring for the hybrid classification-based intrusion prevention systems, *International Journal of Space-Based and Situated Computing* **5**(2): 115–125.

Sha, K., Wei, W., Yang, T. A., Wang, Z. and Shi, W. (2018). On security challenges and open issues in internet of things, *Future Generation Computer Systems* pp. 219–250.

Sherasiya, T., Upadhyay, H. and Patel, H. B. (2016). A survey: intrusion detection system for internet of things, *International Journal of Computer Science and Engineering (IJCSE)* **5**(2).

Sicari, S., Rizzardi, A., Grieco, L. A. and Coen-Porisini, A. (2015). Security, privacy and trust in internet of things: The road ahead, *Computer networks* **76**: 146–164.

Stiawan, D., Idris, M. Y. B., Abdullah, A. H., AlQurashi, M. and Budiarto, R. (2016). Penetration testing and mitigation of vulnerabilities windows server., *IJ Network Security* **18**(3): 501–513.

Surendar, M. and Umamakeswari, A. (2016). Indres: An intrusion detection and response system for internet of things with 6lowpan, *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on*, IEEE, pp. 1903–1908.

Thanigaivelan, N. K., Nigussie, E., Kanth, R. K., Virtanen, S. and Isoaho, J. (2016). Distributed internal anomaly detection system for internet-of-things, *Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual*, IEEE, pp. 319–320.

Zarpelão, B. B., Miani, R. S., Kawakani, C. T. and de Alvarenga, S. C. (2017). A survey of intrusion detection in internet of things, *Journal of Network and Computer Applications* **84**: 25–37.