Table of Contents

1. Abstract
2. Introduction to Edge Computing Security
    - Attacks on Different Paradigms -IoT, cloud and edge
    - Types of IDS: SIDS and AIDS
3. State of Art
4. Introduction to passban
    - How does Passban Work
    - Modes of Passban
    - Architecture of Passban
    - Advantages of Passban
    - Limitations of Passban
    - My Research on Passban
5. Testing passban on provided knowledge
    - Experimental setup
    - Known attacks on Passban
6. Exploring passban beyond the basics
    - Attempting new attacks
    - Passban's detection of False Positives
    - Power Consumption during attacks
7. Can Passban be accurately Retrained?
8. Conclusion

Abstract

Edge computing is a distributed computing paradigm that enables real-time data processing by performing computations at the edge of the network and storing data locally. This can be considered as a major advancement in the technology as we are able to connect more to the real-time world via sensors and IoT devices. This helps in reducing latency, and bandwidth and being quicker in response time which were major concerns due to excessive usage of the cloud computing paradigm.

Although it provides us with so many benefits, unfortunately, it adds up to our security concerns as IoT devices are vulnerable to many cyber threats due to their core nature. Thus, in order to be proactive and defensive against these attacks, it's essential to deploy Intrusion detection and prevention systems as a countermeasure. There are various Network Intrusion Detection systems present in the market but only a handful of them are good enough to be deployed in an edge paradigm due to its resource-constrained nature. One can't afford to use up all the resources such as power, bandwidth, and storage in the deployment of an IDS making it unable to perform the tasks which it was originally built for. Among the few IDSes suitable for Edge, Passban is an IDS specially created for its deployment on an Edge device such as Raspberry Pi without consuming many resources and being able to detect a few commonly known attacks.

In this dissertation, I aim to carefully analyze and provide in-depth documentation on Passban-an anomaly-based IDS through extensive experimentation across various network scans and attacks relevant to IoT networks such as brute force and flooding.

Even though Passban has showcased its abilities in detecting some of the common attacks, it has some shortcomings in detecting stealthy attacks and gives false positives during benign network traffic, which isn't a good sign when it comes to the effectiveness of an IDS.

Additional experiments focused on retraining Passban indicated flaws in its training process, with attacks during training still being detected post-training. Power consumption tests confirm Passban's suitability for edge devices. The findings pinpoint significant opportunities for enhancing Passban's attack detection, training methodology, and accuracy. This research aims to expand practical knowledge of anomaly-based intrusion detection specifically created for the domain of edge computing security.

Introduction to Edge Computing Security

Edge computing is a computing paradigm that involves processing data at or near its source. This is usually done with the usage of various machine learning(ML) algorithms on edge devices to enable real-time analysis of data. With the help of this technique, data can be processed at a point that is very close to its production point, that is at the edge of the network [1].

To have an exhaustive understanding of this paradigm, one must start from the beginning i.e. its origin and its boom with the passing years. Edge computing has become extremely crucial nowadays, so let's take a deeper dive to understand why this happened.

It all began in the mid-2010s when everybody and everything such as wearables, vehicles, laptops, tablets, drones, VR, AR and gaming technologies, etc. started being connected to the internet. It consequently created a big bang in the field of wireless networking when lots and lots of things with sensors generated ginormous amounts of data which ultimately coined the term "Internet of Things" popularly known as IoT. Recent years have seen an exponential surge in wireless communication technology, leading to a significant rise in the number of Internet of Things (IoT) devices. It was estimated that by 2020, the Internet would be home to over 25 billion connected devices. Furthermore, the potential yearly economic influence of IoT was predicted to reach between $3.9 trillion and $11.1 trillion by 2025. IoT devices and sensors consistently produce significant volumes of data, crucial for numerous contemporary tech applications[4]. According to a prediction by IDC, the volume of global data is expected to surge from 33 zettabytes in 2018 to 175 zettabytes by 2025. It's also anticipated that nearly half of this data will be stored in the public cloud which is a paradigm that provides services and resources over the internet to anybody who needs it [5].

Although it is an advancement in technology and progress in the field of wireless networking, which is very beneficial as it represents an evolutionary step for our species, as it provides multiple benefits in various sectors including but not limited to technology, healthcare, transportation, agriculture, and everyday lifestyle, sadly, it's not all sunshine and rainbows. With more and more physical devices being connected to the internet, massive amounts of data are generated every day causing a lot of problems such as storage, slow processing, and the requirement of high computational power [4].

The huge quantities of data generated in our growing digital world require processing and comprehension, and it's not easy to do it manually or even with the use of simple machines . We require advanced technologies such as machine learning (ML) systems,

known for their ability to learn on their own without explicit programming, are key tools in this process. Numerous contemporary applications are supported by machine learning (ML), a dynamic subset of artificial intelligence(AI) that has revolutionized our ability to interpret enormous amounts of data, identify hidden patterns, and anticipate future trends [4].

AI and IoT have combined to create a variety of applications, including image categorization, the development of smart cities, and healthcare advancements. Voice assistants and adaptive emotion recognition have both benefited from this fusion. The data processing and storage requirements for these innovations, however, far exceed those of conventional communication systems. This data can be analyzed using ML algorithms, but IoT devices lack the processing power to run such complex algorithms due to their resource limitations. As a result, cloud computing emerged as a remedy, becoming the foundation for IoT applications powered by AI and promoting their widespread adoption [1].

The term "cloud computing" refers to the 100% availability of resources, such as storage, virtual machine management, multi-tenancy, virtual servers, and operating systems, across a network and from any location[7]. Although the Internet of Things (IoT) and cloud computing coexist, it wouldn't be inaccurate to state that the IoT's expansion inspired the development and use of the cloud. To overcome these limitations, jobs like processing and storage are offloaded to the cloud due to their on-demand and scalable nature. Machine learning algorithms can learn from huge datasets on the cloud thanks to the expansive processing and storage capabilities. However, cloud computing facilities, often located far from users, result in significant latency. Additionally, the growing data exchange burdens the network links to these distant data centers [2].

To mitigate these challenges, the concept of edge computing has emerged. This strategy eases the data transmission burden to the cloud by placing computing power near, or even within, end devices at the network's periphery, reducing communication delays, and improving bandwidth, accelerating decision-making and privacy preservation [2].

Edge computing, an increasingly important concept in IoT systems, allows for more calculations to be done on the devices where data is collected, instead of just relying on the cloud[4]. Advances in computational technologies like graphics and tensor processing units make it possible to shift some processing tasks to powerful edge servers. For real-time services like traffic monitoring, facial recognition, or control system applications, aspects like latency, service quality, and user experience become increasingly vital[3]. This approach was illustrated in a study by Floyer who looked at a wind farm filled with various sensors and data-collecting devices. He found that a system combining edge computing and cloud computing was not only 36% less expensive but

also required 96% less data to be sent over the network compared to a system using only cloud computing [4]. By integrating machine learning with edge computing, we can equip IoT devices with the ability to learn from the data they generate, enabling immediate insights and swift decision-making processes at the network's edge.

In the realm of the Internet of Things (IoT), the fusion of machine learning with both cloud and edge computing architectures has opened up new avenues for data processing and real-time decision-making. This combination of machine learning with cloud and edge computing offers a complementary approach that combines the strengths of each - the extensive processing power of the cloud and the low-latency capabilities of the edge [6].

Heavyweight machine learning algorithms like deep neural networks (CNNs, RNNs), random forests, and k-means clustering are well-suited for implementation in the resource-rich cloud computing environment. However, these algorithms are typically too computationally intensive for resource-constrained edge devices. To enable machine learning on edge devices, lighter-weight algorithms are often used such as logistic regression, decision trees, model distillation techniques, and transfer learning. These more efficient algorithms allow for near real-time analysis on edge devices despite their limitations.

The interconnected landscape of edge computing, cloud systems, and the Internet of Things (IoT) presents a wide range of security challenges. Each domain, while having unique vulnerabilities, also shares common threats with the others. Let’s categorize these cyber threats based on their applicability to these domains:

Attacks Pertinent to Edge, Cloud, and IoT:

- Denial of Service (DoS): It’s an attack in which the hackers try to attack the system by sending an overwhelming amount of traffic towards the host such that it becomes unable to provide any services to the users [7].
- Data Injection: It’s a technique to add corrupted or malicious data into the system to dwindle the integrity or functionality of the host system [7].
- Eavesdropping: Eavesdropping is an attacker in which the attacker tries to listen to the conversation between two hosts on the network to retrieve unauthorized private information [7].
- Man-in-the-Middle: In this track the attackers deceitfully relay communications between two parties who believe they are speaking directly to each other and may even change them in order to ruin the credibility of the data shared and steal sensitive information [7].
- Virus: It’s a malicious code that can be sent to a host system by being attached to harmless files and replicates itself throughout the computer system [7].

- Ransomware: It's a type of malware that locks the files present in a system by adding some sort of encryption that can only be decrypted by the attacker if the user agrees to pay a hefty ransom to gain access back [7].
- Trojan Horse: The name is derived from ancient Greek mythology where the soldiers were sent inside the trojan horse in order to go unnoticed and pretend to be legitimate and that's exactly how a trojan behaves in the computer system. It appears legitimate but performs covert, harmful actions when executed [7].
- Brute force Attacks: The method of sending an infinite amount of username and password combinations in order to find the correct pair to log into a service running on the host machine to gain unauthorized access to it [7].

Attacks Shared by IoT and Edge:

- Jamming: One of the major dangers to wireless sensor networks is the jamming attack. By interfering with packet transmission, the attack's jammers have the potential to severely harm the network's performance [7].
- Malicious Node Injection: The attackers add a rogue device into the network in order to intercept the network traffic or alter the communication [7].
- Sinkhole: A malicious node acts as the shortest path to send all network traffic and lures all the nodes to send data through it making it the ultimate point of all information. This node is called a sinkhole. The data on sinkhole can be easily stolen and modified which in turn compromises the confidentiality, integrity, and availability of data [7].
- Sybil: Sybil's attack occupies its name from a woman named Sybil who has multiple personality disorder. A similar kind of thing is done when multiple fake accounts are created to overwhelm a system such that these fake nodes outnumber the original nodes and control the traffic on the network [7].

Attacks Common to IoT and Cloud:

- Flooding: Flooding is a simple attack technique where a number of packets are sent at a very high speed to the host causing a flood in the network in order to prevent the normal functioning of the host and ultimately causing a DoS attack [7].
- Malicious Insider: One can assume this as an attack caused by the entity or a member of the organization that is already a trusted part of the network but turns out to be malicious and attempt to tamper or steal valuable data of the organization and jeopardise the confidentiality, integrity and availability of the information. It can lead to financial loss, data breach and reputation damage for the organization [7].
- Phishing: Phishing is a deceptive attack where the attacker seems to look exactly like the legitimate source in order to deceive the user into believing it and adding sensitive information. There are multiple types of phishing attacks such as phone phishing, clone phishing, and spear phishing to name a few [7].

- SQL Injection: If a service is using SQL as a database to store information, an attacker can send malicious SQL commands as input to retrieve data from the database [7].

There are various examples of well-known attacks against IoT devices that can't be forgotten easily. The Mirai Botnet attack stands out as one of the most significant attacks in history. This attack transformed numerous smart devices into 'bots' or 'zombies' by exploiting a basic vulnerability: the use of default usernames and passwords and then using them to launch DDoS attacks. Another infamous attack called "Stuxnet", was uncovered in 2010. It is a special worm created to infect and disrupt certain SCADA systems. It was able to manipulate functionalities of these systems causing it to malfunction. What's particularly cunning about Stuxnet is its ability to send deceptive signals, making operators believe that the systems are running as intended, even when they're not [7].

All these attacks are commonly known depicting the insecure nature of these paradigms and devices. These attacks happen because IoT and IIoT devices have various unpatched software and firmware vulnerabilities such as incorrect access control, usage of default usernames and passwords, outdated software, a combination of various kinds of devices giving a large attack surface, lack of proper encryption, application-level vulnerabilities, and improper security training for users [7].

Understanding these threats is crucial in devising robust security protocols and systems for each domain and for the broader interconnected landscape. CIA(confidentiality, integrity, and availability) triad is used in combination with other goals such as non-repudiation, authentication, authorization, and access control to measure the security of an online ecosystem [7].

While completely eliminating all vulnerabilities might be a challenging endeavor, it's essential to vigilantly monitor network activities as a defensive measure against potential threats. This naturally raises the question: Is there a way to actively anticipate and counter cyber attacks? This is where systems like IDS (Intrusion Detection System) and IPS (Intrusion Prevention System) play a crucial role. An IDS's primary objective is to protect the system from unauthorized access, which can compromise the confidentiality, integrity, or availability of data. The IDS operates by scrutinizing network traffic or resource utilization and thereby sounding an alarm upon spotting malicious activities. While Host-based systems (HIDS) focus on individual computers, Network-based systems (NIDS) oversee entire networks, particularly emphasized in this study in the context of IoT deployments.

Depending on their intrusion detection strategy, NIDSs are classified into two main types:

Signature-based

IDSs that use a database of "signatures" from known attacks are known as signature-based IDSs (SIDSs). In order to compare the extracted signatures of the current activities to those in the database, matching techniques are used. An alarm is set off if a match is discovered. They can work in offline mode, where logs of the system activities are examined, as well as in online mode, where they directly monitor the host's traffic and instantly raise alarms. Depending on which and how many traffic "features" are taken into account, the extraction of traffic signatures may be a difficult and time-consuming task to complete. However, the limitations of such IDSes are that they can't detect zero-day attacks, they require lots of storage which also increases the burn on the system to check each attack, which is a time-consuming process, and they require human supervision [9].

Anomaly-based

The limitations of a signature-based IDS were addressed by the development of anomaly-based IDS (AIDS). A model of the nominal behavior of the system is typically built during the training phase of AIDS. IDS deployment involves monitoring computer hosts and comparing their actions to the expected ones. The IDS may issue an alert when there is a notable discrepancy between the behavior of the hosts and the model. Since this approach makes no attempt to match the current host behavior with attack signatures stored in a database, it may enable an anomaly-based IDS to detect zero-day attacks. An additional benefit of an anomaly-based IDS is that it is challenging for an attacker to determine a target host's typical behavior without carrying out interactions with it, as doing so would probably trigger an alert from the IDS. Additionally, anomaly-based IDS could be used as a system analysis tool in addition to security purposes. If the IDS detects anomalies, it means that something is operating differently from the expected behavior. This could be a sign of both an intrusion and a bug in the logic of the device [9].

The deployment of the IDS depends on the necessary requirements of the system or the people and it can vary according to the needs. There are many NIDS in the market today but it's definitely hard to find one that can be deployed over the edge. Edge devices, often characterized by limited processing power and memory, can struggle to support the resource-intensive operations of traditional NIDS. Additionally, edge environments demand real-time or near-real-time processing, and introducing a NIDS might unintentionally add latency, affecting prompt data processing. Differentiating between "normal" and "suspicious" traffic patterns is difficult due to the extreme diversity of traffic in edge networks, which results from a wide range of device types and protocols.

There are multiple IDS created especially for edge environments keeping these issues in mind. One such NIDS is “Passban” and it’s an anomaly-based NIDS created in 2020 to protect IoT devices by analyzing the network traffic and raising an alert if something is suspicious with the help of machine learning algorithms[3]. In this dissertation, I aim to understand the strengths and weaknesses of Passban in the context of modern network attacks. My contributions include employing tools like `nmap`, ‘hydra’ and `hping3`, to simulate various attack vectors as mentioned in the paper to gauge how well Passban can detect and respond to them. I also tried a few new attacks and analyzed if passban can detect those attacks. After that, I trained the model again and reanalyzed its performance as confirmed by the authors.

State of Art

There used to be a lot of dependency on cloud storage platforms and their computation capabilities in the olden times but it caused issues such as delays in the response. Fortunately, transitioning Information Technology (IT) resources from conventional cloud data centers to user-side locations reduces the substantial gap between users and these resources. Due to this shift, the users were consequently provided with solutions including minimal latency and superior stability as it conserves network traffic and achieves decreased latency in data interactions.Due to the capabilities that edge possesses, it has seamlessly become an integral part of various applications that impact various facets of our daily lives, including autonomous driving, smart city infrastructure, industrial sectors, and commercial applications [8].

Prevalent Edge computing architecture

The existing general architecture of edge computing is characterized as a three-tier heterogeneous network comprising the thing layer, edge layer, and cloud layer [8] explained as follows:

Thing Layer:

As the name implies, the thing layer consists of various things or smart devices also known as IoT devices such as smart watches, smart health sensors, smart locks, cameras, Internet of Vehicles (IOVs), augmented reality devices, etc [8].

Edge Layer:

The edge layer lies in the center of this three-tier architecture and is situated at the network’s edge. It is comprised of numerous edge nodes including routers, gateways, switches, access points, base stations, and specific edge servers. The key functions of this layer consist of data forwarding, network connection, network capability invocation through API interfaces, platform middleware, control, and infrastructure planning [8].

Cloud Layer:

Lastly, the "cloud layer" provides copious amounts of data processing and storage capabilities. This layer works by collaborating with the edge layer and provides a cloud platform to perform various functions such as network-wide scheduling, arithmetic distributions, and other cloud services. This is the topmost layer and performs a major role as most of the data is processed and stored here for long-term purposes [8].

Edge computing can be combined with decision-making- technologies also known as machine learning(ML) to further widen its scope of various applications. Notably, studies focusing on edge computing's amalgamation with ML, DL, auction methods, and game theory were found to be quite pervasive. It's a great idea to explore ML techniques in edge-cloud environments for reliable resource provisioning [8]. Unfortunately, there are several challenges to implementing machine learning in an Edge AI environment such as:

- Edge devices do not have a lot of resources such as processing power, battery, storage, or memory [8].
- Edge devices generate low-quality data which can deteriorate the performance of ML algorithms [8].
- ML algorithms should be small to fit in the storage capacity of these devices [8].
- Deployment is difficult [8].

In order to overcome these challenges, there are various ML algorithms specifically designed for edge computing that can be trained with memory requirements considerably lower than other modern ML algorithms. Moreover, Edge Intelligence focuses on model compression and shared cognition to efficiently deploy DL models at the edge, given the limited computing resources and energy consumption [8].

Although edge computing is growing at an amazing pace, is it really secure?

Edge computing encompasses various layers including many IoT devices, and edge and cloud servers that are not that secure. There are various common vulnerabilities known to man in IoT and other devices. The detailed investigation and analysis of the security threats present in cloud computing, edge computing, and the Internet of Things (IoT) provide critical insight into the nature and diversity of potential vulnerabilities within these rapidly evolving technological paradigms. It wouldn't be wrong to conclude that these vulnerabilities can be exploited and it could lead to several security issues such as privacy, data breaches, and many damaging cyber attacks [7].

Just identifying and categorizing the threats is not enough, the next focus has to be finding solutions and countermeasures. A range of proactive and reactive measures were proposed to mitigate these security challenges. Given that several attacks exploit human error, the importance of screening employees and controlling access to sensitive areas of the system was highlighted. Encryption was cited as another essential tool for securing data, alongside rigorous monitoring mechanisms to scan for malicious activities. Furthermore, traditional firewalls and antivirus software were noted as essential elements of any robust security system [7].

Building on these rudimentary measures, the introduction and enhancement of the Intrusion Detection System (IDS) became the highlighting security measure that must be deployed. The IDS emerged as a prominent solution, introduced by various authors for its role in detecting and promptly responding to threats. The discussion around IDS becomes crucial as it represents a dynamic line of defense, adapting to evolving threats and offering real-time protection in complex digital environments [7].

In today's cybersecurity landscape, there's no shortage of Intrusion Detection Systems (IDSes) available to safeguard our digital networks. These systems come with a range of features, tailored to different environments. However, when we shift our focus to edge computing, the suitability of these IDSes comes into question. Can the majority of these IDSes be seamlessly deployed in an edge environment? The edge computing paradigm, with its unique demands for real-time processing and localized decision-making, entails a fresh perspective on intrusion detection. There have been various NIDSes that have been created.

To set forth the evolution and trajectory of Intrusion Detection Systems, it is vital to categorize IDSes developed pre and post-Passban era based on their typology — be it Signature-based Intrusion Detection Systems (SIDS) or Anomaly-based Intrusion Detection Systems (AIDS).

There has been a great development of different kinds of IDS over the years, as summarized in this table before Passban which was developed in 2020. Here’s a short summary of each IDS to get a better insight into their functioning:

SNORT: Snort is a signature-based IDS and IPS developed to analyze the traffic on the network and make decisions such as creating a log, alerting the user, or letting it pass. It is frequently used to actively block or passively detect a variety of attacks such as buffer overflows, stealth port scans, web application attacks, operating system fingerprinting attempts, etc. Snort has to manage a big rule set and compare the signatures to the attacks which seems like a resource-constraining task and thus fails as an appropriate IDS in an Edge computing environment [17].

SURICATA: Suricata can be used for both real-time and offline Intrusion detection using Pcaps. It's a free and Open source IDS which is signature-based as well. Suricata can benefit edge devices that have multi-core processors, enabling faster and more efficient traffic analysis. However, that being said, it still uses a list of signatures to detect attacks which might be difficult to maintain in the case of Edge devices or IoT devices being heterogeneous in nature [17].

ZEEK: ZEEK formerly known as Bro monitors and parses each packet in the network traffic and notifies if it identifies a suspicious activity in the network.It's an open-source IDS and works on multiple platforms including Linux and Mac OS. Due to the fact that it's customizable, it can be good for an edge environment, however, it's very resource-intensive in nature [17].

IDS pattern matching engine: The authors optimized the pattern matching technique using auxiliary shifting and Early decision to make it more appropriate for IoT devices by reducing the load and limiting memory usage [18].

IDS based on traffic signature detection: It's an IDS created for wireless sensor networks with three modules for traffic monitoring, and matches against policy rules to detect any attacks and ultimately take an action based on it i.e. an alert to notify the network administrator about the the possible attack [19].

IDS based on Energy consumption: The developed IDS uses a mobile agent that randomly moves from node to node while containing the battery level of each node. A linear regression model is used to calculate the anticipated power consumption based on prior battery status observations. It makes sense that this metric would be able to identify different resource-consumption-focused denial-of-service attacks as the battery status of different nodes is drastically altered by flooding attacks [20]. This method is however not always reliable as IoT devices use different protocols and are heterogeneous in nature [3].

SVELTE: It's an IDS specifically created for IoT devices in order to protect them from network or routing attacks such as spoofing and sinkhole attacks.It consists of a server and a client as its two main modules. The server module constructs the IPv6 network topology, maps it with IDS parameters, and defends the sensor network from Internet-based attacks. The client module, on the other hand, offers an IPv6 network topology mapper and calculates lost packets. It was exhaustively examined in the Contiki Operating System [21].

Neural Network-based IDS: This IDS was created to protect critical infrastructures from various attacks using neural network-based techniques namely the Error-Back

Propagation and Levenberg Marquardt, for normal behavior modeling of the system by analyzing data in real-time from actual infrastructures [22].

IDS based on ANN used to detect DOS attacks: This IDS was developed to find out DoS attacks in the network by analyzing data from various IoT devices. They used ANN to distinguish between normal traffic and DoS attacks by using the supervised ML technique [23].

Heimdal: It's a lightweight, open-source host-based IDS that uses unsupervised ML technique and is created for a network of IoT devices and can be deployed on routers. It uses a whitelisting strategy where it checks profiles of each IoT device on the network including details like legitimate destinations, types, and the number of packets, and monitors both incoming and outgoing traffic in order to figure out what is normal. The rest is categorized as malicious and is blocked for good [24].

There has been a certain escalation in the growth and usage of hybrid IDS as a typical AIDS or SIDS had a certain foible causing them to be not fully perfect to be deployed. A signature-based IDS can't detect zero-day or unknown attacks and anomaly-based has a high tendency to generate false alerts.

EHIDF: Edge-based Hybrid Intrusion Detection Framework is able to detect both known and unknown attacks using ML by combining two states- signature-based, and anomaly-based, and creating a hybrid module to overcome the known weaknesses of both [25].

Hybrid IDS for Edge-Based IIoT Security: In order to improve the data preparation process, the IDS first implements a feature engineering solution based on the PCA algorithm and then validates a K-NN classifier model to produce an accurate intrusion detection method. To detect an attack, the authors integrated the Snort intrusion detection system and to detect anomalies, they used PCA feature engineering and K-NN classifier [26].

PCCNN-Based NIDS: This IDS works on a novel method for effective and precise intrusion detection in IoT environments that combines the advantages of PCA for feature collection and CNN classifiers based on deep learning. It works efficiently on an edge-computing paradigm and can easily detect DoS attacks [27].

DF-IDS: It's also a hybrid IDS that uses a neural network to distinguish malicious traffic from benign one. This is done with the help of feature selection techniques using SpiderMonkey, Principle Component Analysis, Information Gain, and Correlation

Attribute Evaluation which increases the overall efficiency of the IDS and reduces the training and testing time [28].

There has been a great development of IDS over the years and they have become more accurate and more specific to different paradigms with the development of technology over the years.

	Signature-based IDS		**Anomaly-based IDS**	
	Name	**Year**	**Name**	**Year**
Before Passban (2020)	Snort	1998	Neural Network-based IDS	2009
	Bro	1999	IDS based on ANN used to detect DOS attacks	2016
	Suricata	2010	IDS based on Energy consumption	2013
	IDS based on traffic signature detection	2014	Svelte	2013
	IDS pattern matching engine	2014	Heimdall	2017
After Passban	**Hybrid IDS**			
	EHIDF	2022		
	Hybrid IDS for Edge-Based IIoT Security	2022		
	PCCNN-Based NIDS	2021		
	DF-IDS	2022		

Table I: An overview of the research works discussed

Passban: Analyzing Anomaly Detection IDS

Passban is an anomaly-based network intrusion detection system that is capable of assessing data produced by various IoT devices. It was designed to be anomaly-based to overcome the aforementioned limitations of a typical Signature-based IDS. It was built to be lightweight and platform-independent such that it can be used on simple edge devices to monitor the traffic close to the edge of the network that is close to the IoT devices. Thus it's easy to execute and deploy it on resource-constrained gateways such as Raspberry Pi. Being an edge-based IDS, it lessens the overall network's processing load while also providing better security, as an edge device is more sensitive to privacy concerns due to its proximity to a data source [3].

How does PassBan work exactly?

It's easy to run Passban on any system as the authors provided a docker image that can be easily pulled and make the passban platform independent and very easy to deploy, replicate and validate. One can easily pull up the docker image make the necessary configurations as mentioned on the docker page and start using it. Once up and running Passban starts to function as per the mode it's been selected to work. One can easily choose the mode from the web interface it provides an intuitive web interface for selecting modes and interacting with the IDS which is very convenient for the user [3].

Passban has two modes namely:

Training Mode

The IDS undergoes its training phase under normal network conditions, meaning it presumes that the system is not under any attack. It computes network flow metrics and constructs a feature set to train the machine learning model. Since Passban is deployed on edge-a resource-constrained platform, it's optimal to deploy lightweight ML algorithms that do not consume too many resources. In this context, a beneficial machine learning approach is a one-class classification. This method falls under unsupervised machine learning because it doesn't rely on labeled data to construct its classification model. Various one-class classification methods are grounded in profiling or isolation techniques. The profiling method focuses on learning the typical behavior of the system using key features, which are then compared with the behavior of the system when it's in action mode. When something different happens that doesn't typically happen, it's marked as an intrusion which can be a little misleading sometimes as it might show an intrusion even when there's a slight difference from the normal pattern which might not necessarily be an attack raising the chances of giving false positives. It can also be computationally demanding, while isolation is often more efficient, especially with diverse IoT device attributes. It focuses on partitioning the datasets and looking for something

that’s not ordinary as they work on the concept that anomalies are easier to detect because of the way they look inconsistent. The authors evaluated the isolation forest as an isolation-based method and the local outlier factor (LOF) as a profiling-based method for anomaly detection on edge devices [3].

Isolation Forest uses unique data attributes to detect anomalies, while LOF, being distance-based, might be less suitable for IoT due to its higher computational needs. Once the training concludes, the resultant model is saved to the gateway's internal memory, enabling it to identify potential threats in incoming traffic. The authors affirm that within Passban, initiating a new training is achievable by simply activating a software button via the web-based management interface, eliminating the need for in-depth technical expertise.If no training is done then initially, the algorithm checks for a pre-existing trained model in the memory; if found, it is loaded and used to detect anomalies or intrusions in the current traffic [3].

Prediction Mode

After deployment on the network gateway, it continuously monitors network traffic by collecting raw packets and deriving network flows. It incorporates a tool named 'NetMate' for Network Measurement and Accounting. This tool actively listens to network traffic, transforming raw packets into network flows, and subsequently calculates metrics for these flows. For traffic capture, NetMate leverages the LibPCAP library. Passban identifies patterns not recognized as standard during its learning phase. Given its computational efficiency during inference, this approach is especially valuable for real-time network traffic monitoring where immediate anomaly detection is essential. If it’s out of that box, it flags it as an attack which goes to the action manager component that undertakes essential actions based on the determinations produced by the trained model for identifying attacks. Some such actions include recording attack occurrences, restricting incoming or outgoing traffic to a specific device, and alerting a user about the attack. Everything is displayed on the web interface created by the authors [3].

The architecture of Passban

In the general architecture of this paradigm, there are 4 layers starting from the IoT layer including all the devices, then the Edge gateway layer, then the router layer, and finally the cloud layer. Each layer sums up the data generated in the previous layer. Thus, the best place to deploy an IDS would be at the base IoT layer as it would only consist of traffic generated by IoT devices giving a better idea of what may or may not be an attack. However, as discussed before IoT devices don’t have that kind of computational power and resources to sustain an IDS, so the second best place to deploy the IDS is the Edge gateway layer and that’s exactly where passban is positioned to keep track of the incoming and outgoing traffic [3].

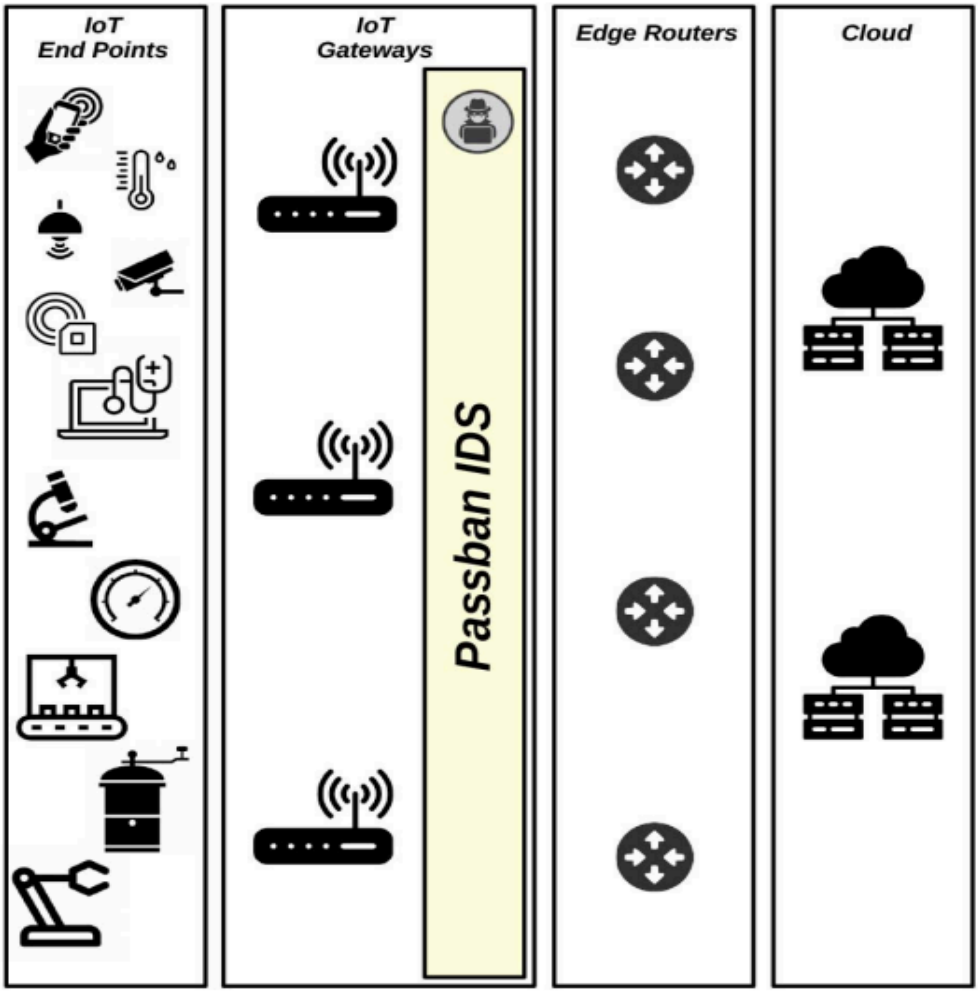

Figure 1: Architecture of Passban [3]

Advantages of Passban as specified in the paper [3]

- Works in an edge resource-constrained environment
- Platform independent in nature
- Protection at the gateway level

Limitations of passban as specified in the paper

- Passban operates under the assumption that the IoT network is not compromised during the IDS training phase [3].
- There are potential instances where Passban could flag an irregularity even when the network traffic deviates from the norm but isn't necessarily malicious, leading to false positives [3].
- Notably, during an SYN Flood attack, when the gateway's resources are heavily utilized, the efficiency of Passban in detecting threats might decline [3].

My Research on Passban-IDS

In this research task, the primary objective is to delve into Passban and critically assess the assertions made by its original authors. Undertaking an in-depth examination of research prototypes such as Passban sheds light on the intricacies and challenges of translating academic security frameworks into pragmatic applications. Through rigorous replication of the IDS and meticulous analysis of the original work, this thesis offers insightful observations, thereby aiming to augment the comprehension and practical applicability of the tool.

Following is the inventory of my work:

- The authors claim that they tried 4 attacks on the network namely Port scan, HTTP brute force, SSH brute force, and SYN flood, and passban was able to detect them successfully [3]. I created a setup to replicate those attacks and analyzed the performance of passban.
- The authors mentioned they performed port scans but didn't specify which scans were detected [3] , so I tried a wide array of possible port scans using Nmap and noted Passban's response.
- After that, I tried 4 more novel attacks that weren't mentioned in the paper namely VNC brute force, UDP flood, ICMP flood, and ARP spoofing, and monitored Passban's performance. These attacks provide a good representation of the common attacks in Edge computing environment faced by IoT devices so it's essential to detect Passban's sensitivity against such attacks.
- The authors claim that during training passban considers that the traffic is in attack-free mode [3] , so I tried to run attacks during the training phase and check if Passban considered those attacks as normal traffic or still flags them as attacks.
- I used a multimeter to check the power consumption during an attack to confirm if it's actually good for the edge environment.

These scans and attacks provide a robust test of Passban's anomaly detection approach. Passban was an early pioneer of anomaly detection on resource-constrained gateways, making it a worthwhile research prototype to explore. The outcomes seek to both validate Passban's contributions and illuminate opportunities to advance the state of the art. Through rigorous inquiry and experimentation, this work hopes to expand the frontiers of knowledge surrounding intrusion detection for the proliferating domain of the Internet of Things [3].

Testing Passban on Provided Knowledge

The idea behind this section is to reproduce the Passban attacks mentioned in the paper and derive a comprehensive conclusion by monitoring and analyzing Passban's response to the performed attacks [3].

Experimental setup

To implement Passban within an edge computing context, it's essential to establish a network integrated with edge devices. For this setup, I've utilized a Raspberry Pi 4 Model B Rev 1.2, which is a single-board computer that acts as an edge device as it processes data locally interfaced with a monitor, keyboard, and mouse which acts as a system on which Passban-IDS is running. Additionally, this Raspberry Pi is linked to a laptop, which serves as the platform from which I instigate attacks and concurrently assess Passban's performance on the monitor screen when the target - Raspberry(berry) is under attack. In this scenario, it's Raspberry that immediately processes data locally implying the use of edge paradigm in this context, and acts as a gateway simultaneously. Once the setup is configured, the next ideal step is to download the necessary files and start running them on the Raspberry's terminal.

After that, the next optimal step is to pull the docker image make the necessary changes as recommended by the authors, and create a docker container. I used the command "**docker run -v /home/berry/passban/:/data -it -p 5200:5200 --privileged --network host mojiz/passban-ids bash /usr/src/passban-ids/app/start.sh**".
Run the docker command and start the web interface.

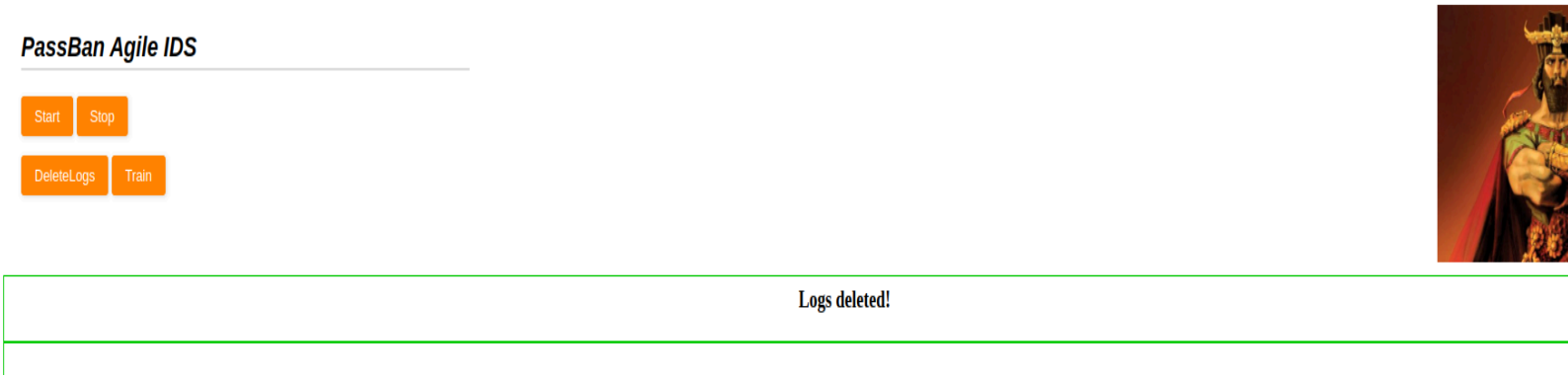

Figure 2: Web Interface of Passban

Once it's all done, the previously trained model is loaded which should ideally detect the aforementioned attacks by the authors and I'm going to test those attacks [3].

The authors tried commonly known attacks that IoT devices easily fall prone to namely:

- **Port Scanning**
  It is a way for attackers to look for open doors or ports in a network. They can identify which services are running on the network and what versions of the software are being used to find a potential vulnerability and attempt an attack. Although it's not an attack in itself, it serves pretty much as the first step in the

chain of a multi-step attack. High volumes of port scanning show that someone is trying to get access to the network, which is very insecure. Overuse of port scans can also hinder the working of the system and might cause a DoS attack as well.
I used a multimeter to figure out the power consumption during each of these attacks.
Tools like Nmap are used to scan various ports and a good IDS must detect it. The authors specify that Passban can potentially detect port scanning [10]. However, they didn't mention what scan they exactly performed [3]. Nmap provides a range of scans that an attacker can attempt [10].

In order to confirm, what kind of scans are detected by passban, I decided to monitor the response of IDS while running a wide array of Nmap scans namely:

- **Ping Scan**
  It's used to find the presence of all active IPs or hosts inside the network. Passban didn't show any response during this attack.

```
msc-lab@msclab-Latitude-5431:~$ nmap -sn 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 13:36 BST
Nmap scan report for 192.168.2.80
Host is up (0.00034s latency).
```

Figure 3:Ping scan Command

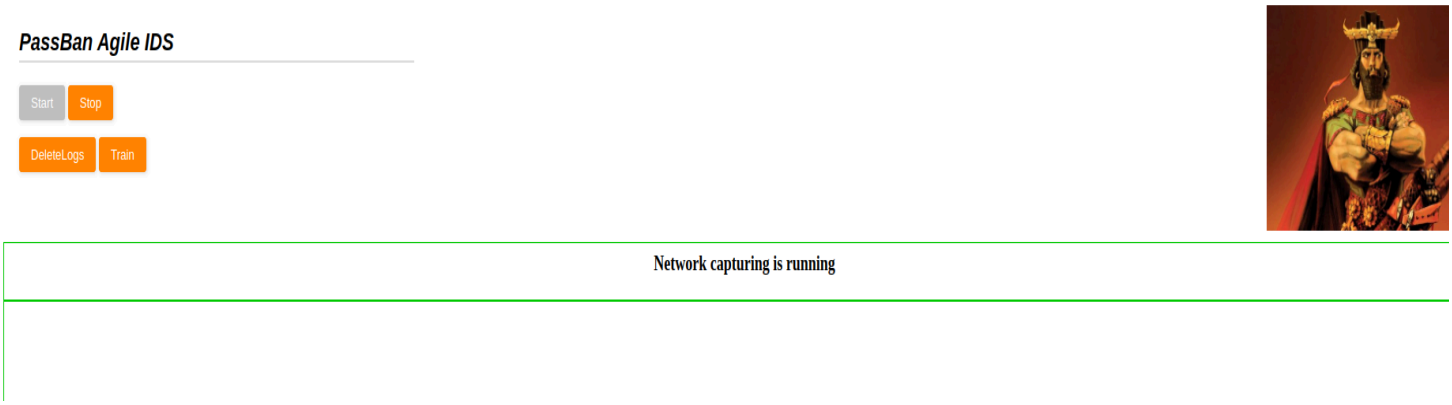

Figure 4:Ping scan output

- **Nmap syn scan**
  It's a stealthy scan as it doesn't complete a TCP connection and it can scan multiple ports very fast [10]. Passban doesn't detect it.

```
msc-lab@msclab-Latitude-5431:~$ nmap 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 13:33 BST
Nmap scan report for 192.168.2.80
Host is up (0.0025s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5200/tcp open  targus-getdata
5900/tcp open  vnc
```

Figure 5: Syn scan Command

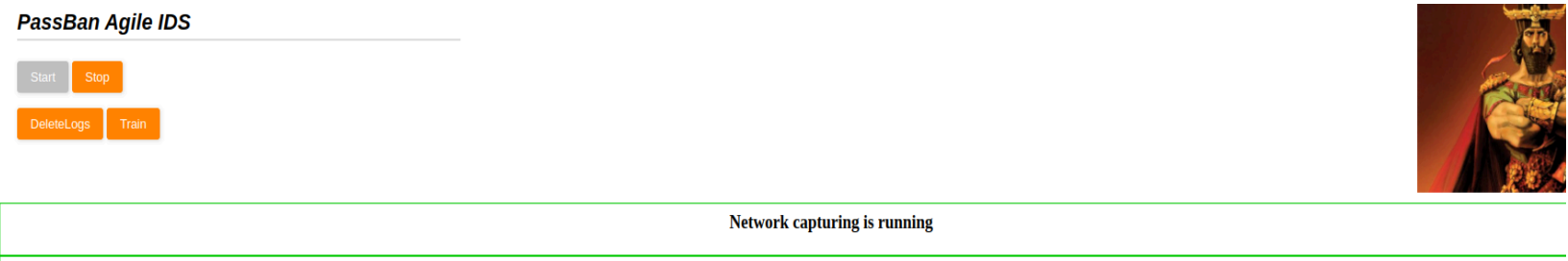

Figure 6: Syn scan output

- **<u>TCP connect scan</u>**
  By concluding the TCP three-way handshake, a TCP connect scan creates a full connection to the target host [10].Passban shows no response to this scan.

```
msc-lab@msclab-Latitude-5431:~$ nmap -sT 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 13:38 BST
Nmap scan report for 192.168.2.80
Host is up (0.0025s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5200/tcp open  targus-getdata
5900/tcp open  vnc
```

Figure 7:TCP connect scan command

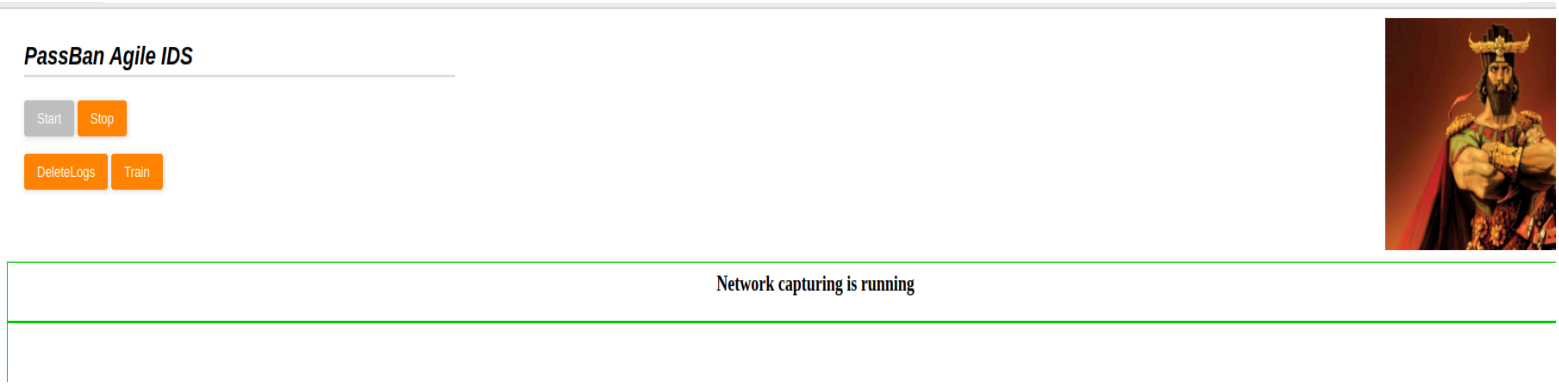

Figure 8:TCP connect scan output

- **<u>UDP scan</u>**
  A target is scanned for any UDP ports in contrast to the standard scan which only examines TCP ports. TCP scans are typically faster and simpler than UDP scans [10].Passban shows a response but it also generates a lot of false and irrelevant traffic.

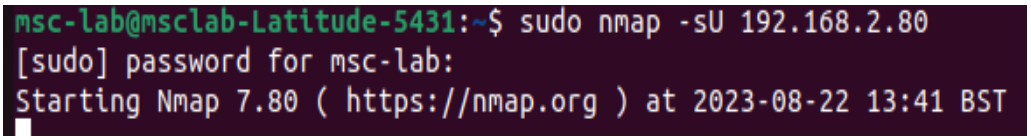

Figure 9:UDP scan command

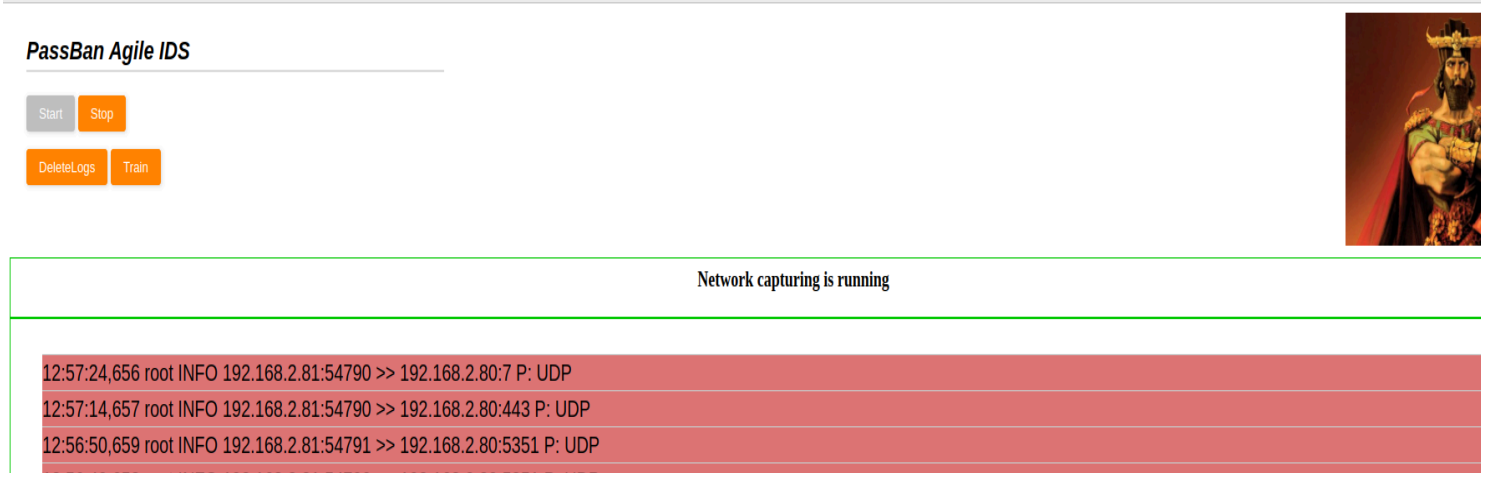

Figure 10:UDP scan output

- **<u>Stealth syn scan</u>**
  The goal of a stealth scan in Nmap is to reduce the chance that firewalls and other security measures on the target host will notice scanning activity [10]. Passban shows no response to this attack.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sS 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 14:01 BST
Nmap scan report for 192.168.2.80
Host is up (0.00075s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5200/tcp open  targus-getdata
5900/tcp open  vnc
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 11:Stealth scan command

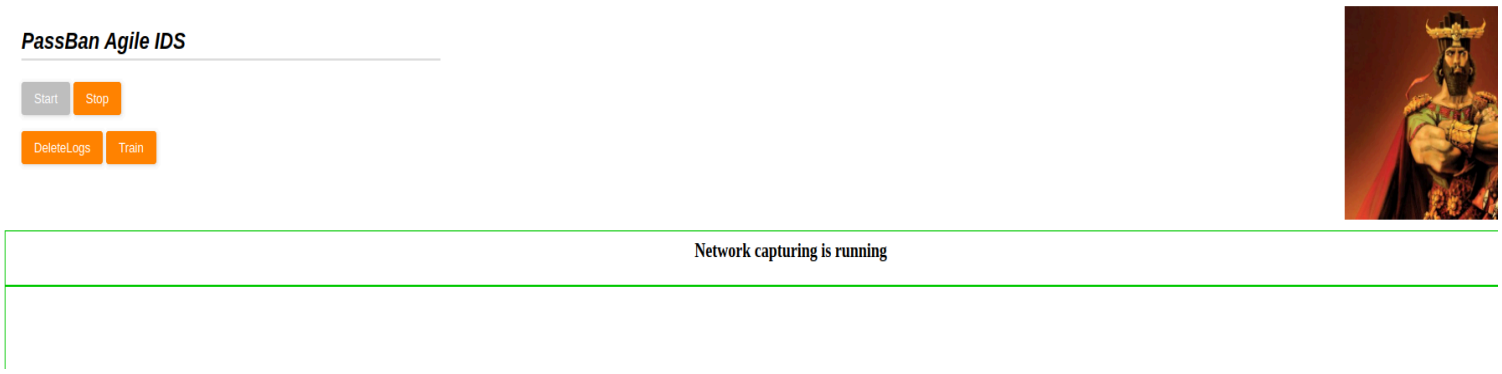

Figure 12:Stealth scan output

- **<u>Null scan</u>**
  A null scan involves the attacker sending the target a packet with no flags set. The target will be baffled and unresponsive once more. This will show that the target's port is open. The port on the device is closed, though, if the target replies with an RST packet [10].Passban shows no response towards these attacks just a lot of false positives.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sN 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 14:03 BST
Nmap scan report for 192.168.2.80
Host is up (0.00082s latency).
Not shown: 997 closed ports
PORT     STATE         SERVICE
22/tcp   open|filtered ssh
5200/tcp open|filtered targus-getdata
5900/tcp open|filtered vnc
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 13:Null scan command

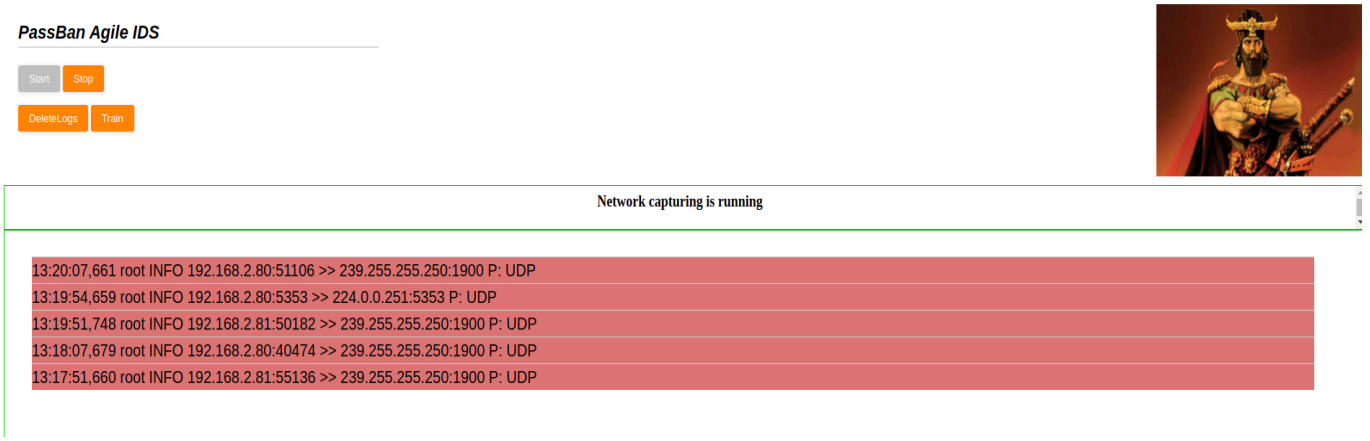

Figure 14:Null scan output

- **FIN scan**

  A TCP packet with only the FIN flag set is transmitted to the remote host during a FIN scan attack. If the host makes no attempt to respond, the port is open [10]. Passban shows no response towards this attack but generates a lot of false positives.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sF 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 14:25 BST
Nmap scan report for 192.168.2.80
Host is up (0.00064s latency).
Not shown: 997 closed ports
PORT     STATE         SERVICE
22/tcp   open|filtered ssh
5200/tcp open|filtered targus-getdata
5900/tcp open|filtered vnc
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 15:FIN scan command

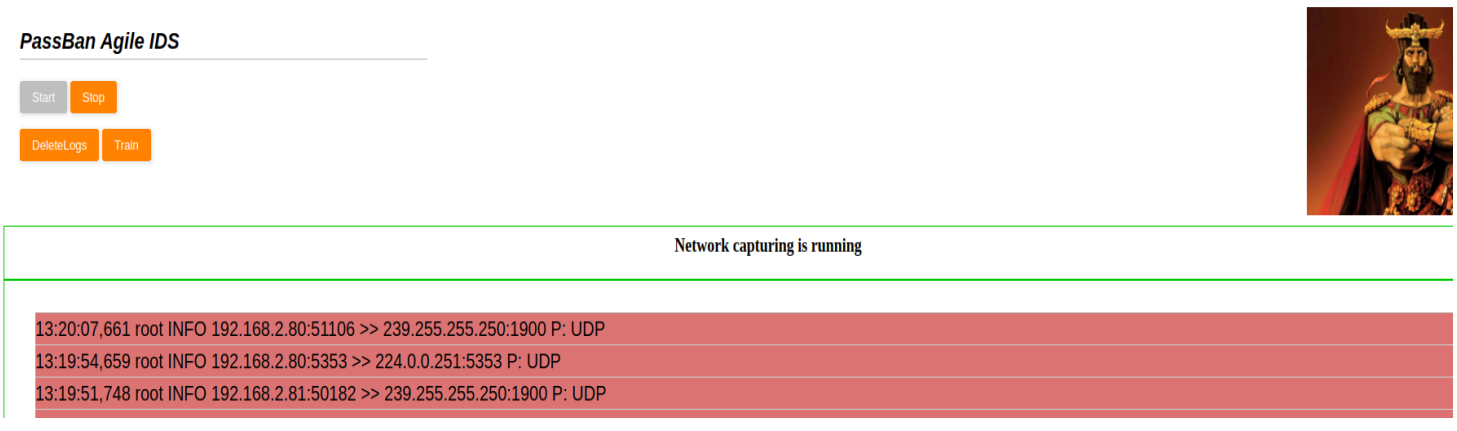

Figure 16:FIN scan output

- **XMAS scan**
  The FIN, URG, and PUSH flags are set by the Xmas Tree scan. We can see that alternating bits are enabled, or "Twinkling," just like lights in a Christmas tree, when viewed within Wireshark [10]. Passban shows no response towards this attack but generates a lot of false positives.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sX 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 14:27 BST
Nmap scan report for 192.168.2.80
Host is up (0.00052s latency).
Not shown: 997 closed ports
PORT     STATE         SERVICE
22/tcp   open|filtered ssh
5200/tcp open|filtered targus-getdata
5900/tcp open|filtered vnc
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 17:Xmas scan command

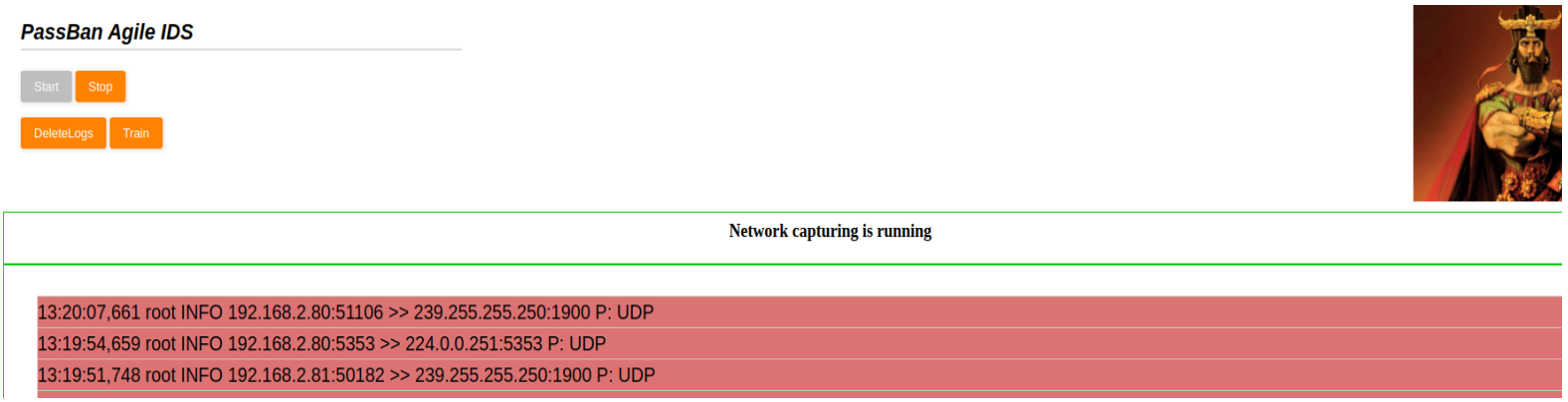

Figure 18:Xmas scan output

- **ACK scan**
  This scan type sends ACK packets to a host and is typically used to map firewall rulesets and differentiate between stateful and stateless firewalls. If an RST is received, the port is deemed to be "unfiltered" (i.e., it was permitted to send its RST through the applicable firewall). The port is said to be "filtered" if nothing returns. In other words, the firewall prevented the RST from returning to the port. This scan type can assist in identifying whether a firewall is stateful (tracks connections and also blocks unwanted ACK packets) or stateless (just blocks incoming SYN packets) [10].Passban shows no response towards this attack but generates a lot of false positives.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sA 192.168.2.80
[sudo] password for msc-lab:
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:04 BST
Nmap scan report for 192.168.2.80
Host is up (0.00080s latency).
All 1000 scanned ports on 192.168.2.80 are unfiltered
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 19: ACK scan command

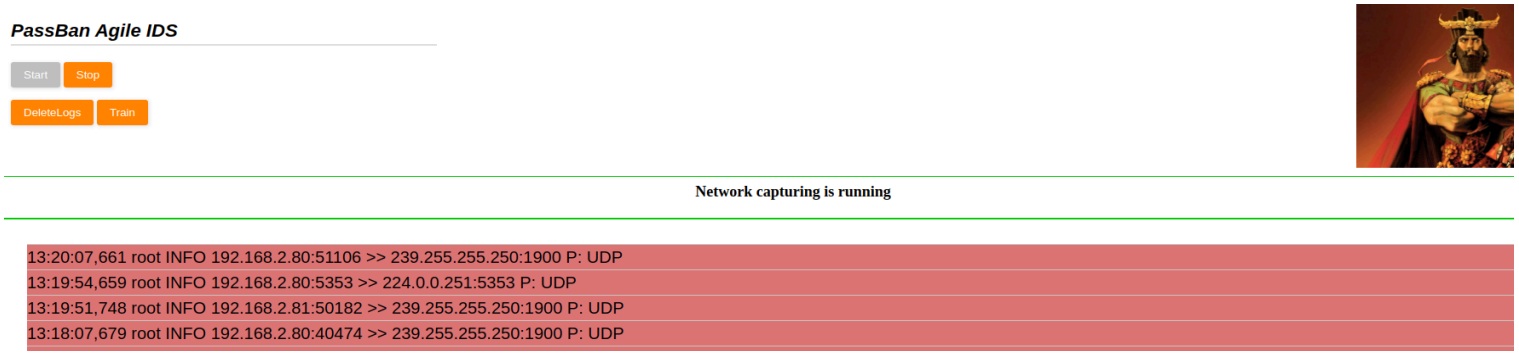

Figure 20: ACK scan output

- **<u>Idle scan</u>**

  In order to identify open ports on the target, this scan type generates "predictable IP fragmentation ID" sequences on the zombie host. In essence, this means that the TCP sequence ID of the zombie is checked, and then the zombie is forced to try a port on the target before the sequence ID is checked once more. Nmap can determine if the port on the target was open based on how many times the sequence number changed [10]. Here I'm using 192.168.2.97 as a zombie IP. Passban shows no response towards this attack but generates a lot of false positives.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sI 192.168.2.97  192.168.2.80
```

Figure 21: Idle scan command

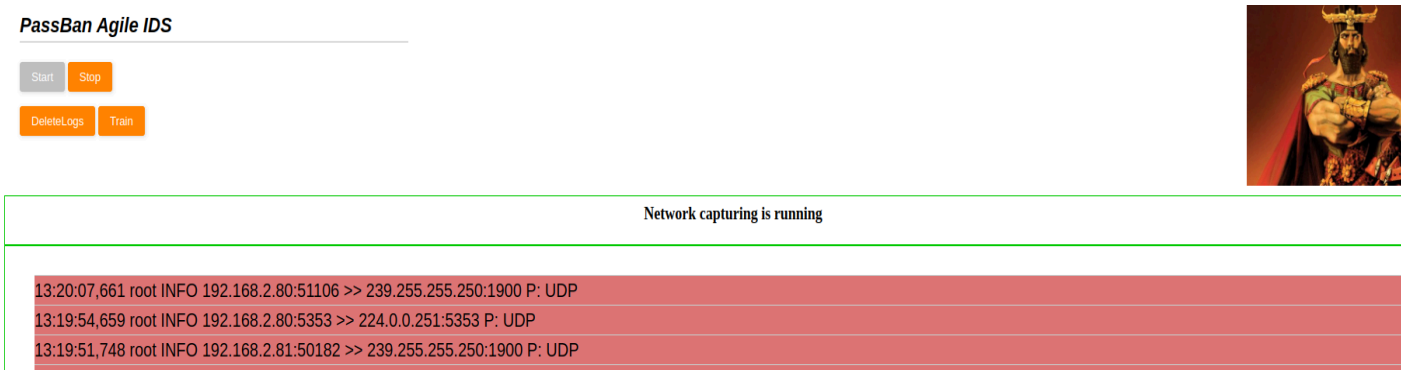

Figure 22: Idle scan output

- **Version detection**

  It helps to identify the service and exact type of service running on the open ports detected [10].Passban detects this scan as expected.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sV  192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:17 BST
Nmap scan report for 192.168.2.80
Host is up (0.00075s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Raspbian 5+deb11u1 (protocol 2.0)
5200/tcp open  http    PHP cli server 5.5 or later (PHP 5.6.33-0)
5900/tcp open  vnc     RealVNC Enterprise 5.3 or later (protocol 5.0)
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 23: Version detection scan command

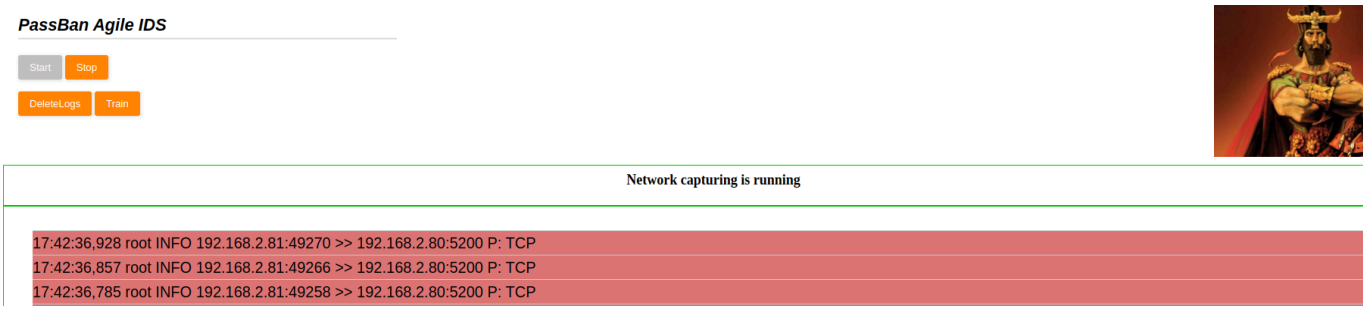

Figure 24: Version detection scan output

- **Version detection stealth scan**

  It works similarly to service scan but makes much less noise and is harder to detect [10]. Passban is able to detect it but no while it's running only after the scan is over.

```
msc-lab@msclab-Latitude-5431:~$ nmap 192.168.2.80 -sV -T2 -vvvv
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:25 BST
```

Figure 25: Version detection stealth scan command

```
PORT     STATE SERVICE REASON  VERSION
22/tcp   open  ssh     syn-ack OpenSSH 8.4p1 Raspbian 5+deb11u1 (protocol 2.0)
5200/tcp open  http    syn-ack PHP cli server 5.5 or later (PHP 5.6.33-0)
5900/tcp open  vnc     syn-ack RealVNC Enterprise 5.3 or later (protocol 5.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 26: Version detection scan command result

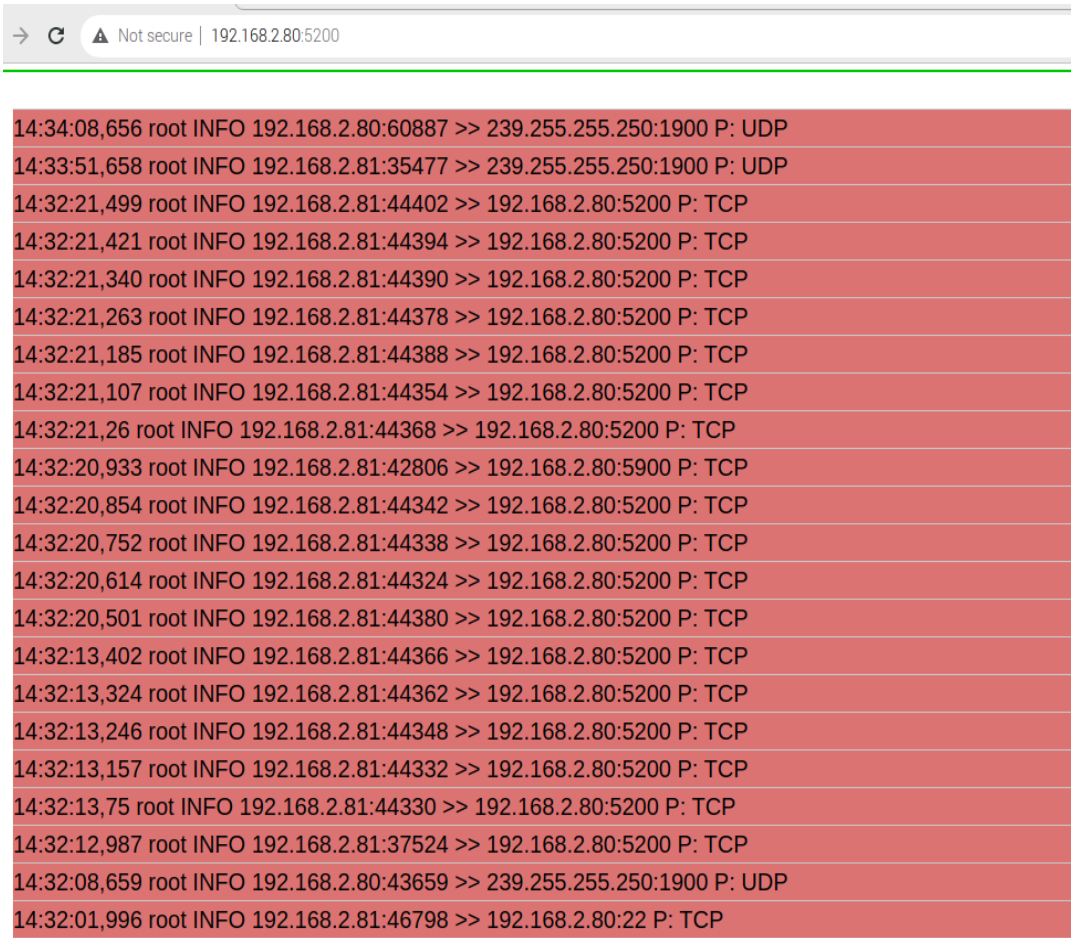

Figure 27: Version detection stealth scan output

- **<u>OS detection</u>**
  As the name suggests, this scan identifies the Operating system running on the host machine being scanned. It uses the TCP/UDP stack fingerprinting mechanism to detect the OS [10]. Passban shows no response towards this attack.

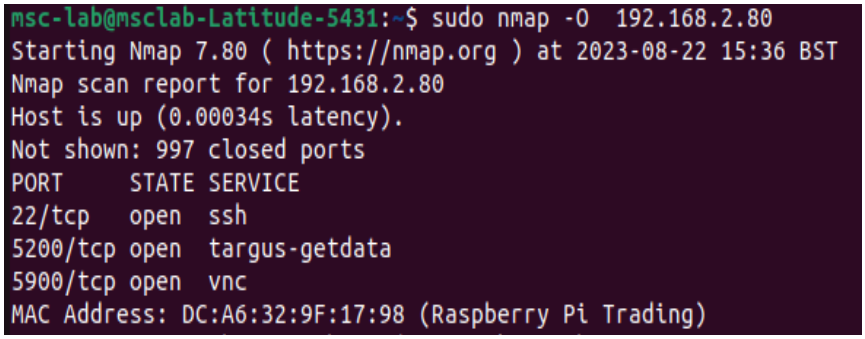

Figure 28: OS detection scan command

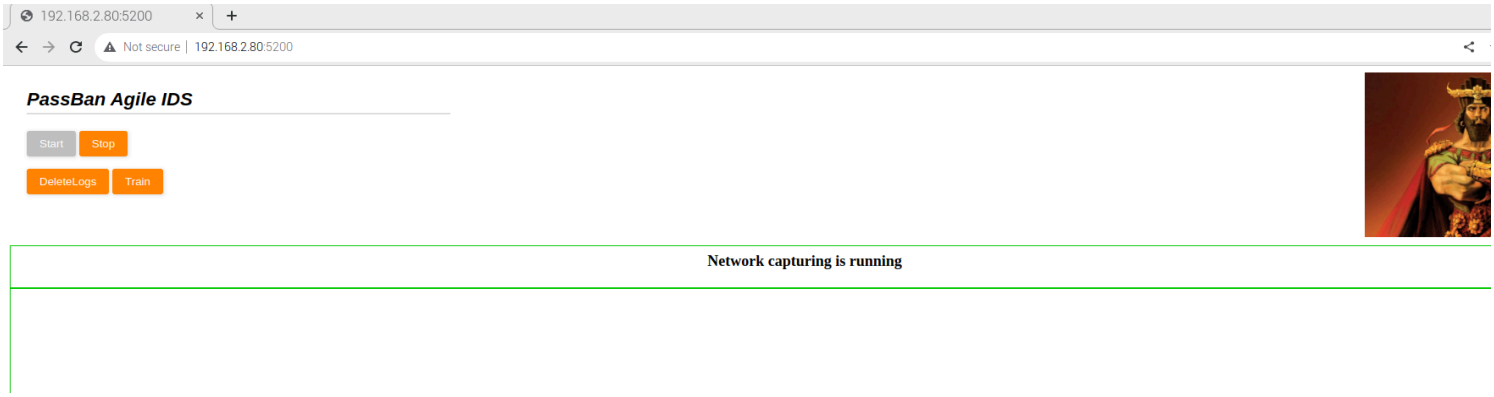

Figure 29: OS detection scan output

- **<u>Script scan</u>**
  The Nmap script scan feature allows the user to figure out the open ports and use executable scripts to perform actions on services running on these ports [10]. Passban was able to detect this scan efficiently.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap --script=vuln 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:39 BST
Nmap scan report for 192.168.2.80
Host is up (0.00076s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
5200/tcp open  targus-getdata
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
5900/tcp open  vnc
```

Figure 30:Script scan command

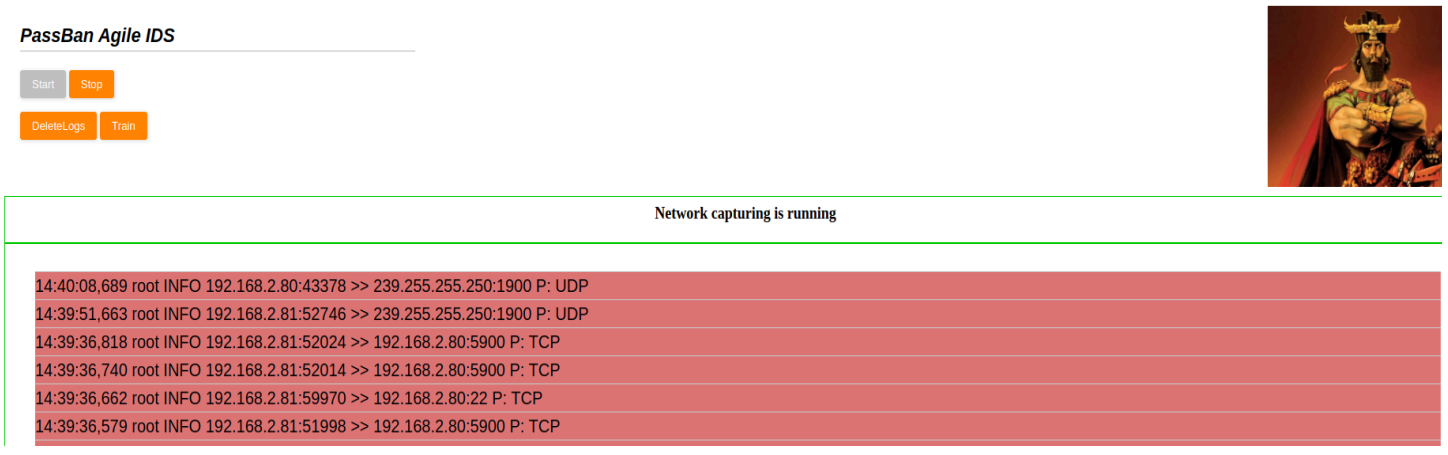

Figure 31:Script scan output

- **Aggressive scan**
  It performs OS detection, version detection, script scanning, and traceroute and provides a lot of significant information about the network [10].Passban was able to identify this scan.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -A 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:41 BST
Nmap scan report for 192.168.2.80
Host is up (0.00054s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Raspbian 5+deb11u1 (protocol 2.0)
5200/tcp open  http    PHP cli server 5.5 or later (PHP 5.6.33-0)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
5900/tcp open  vnc     RealVNC Enterprise 5.3 or later (protocol 5.0)
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 32:Aggressive scan command

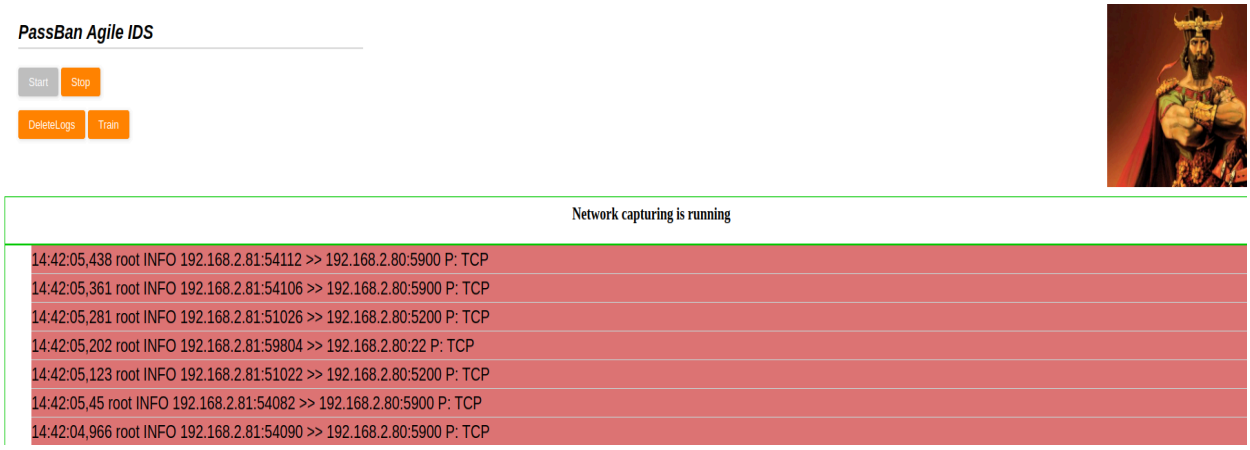

Figure 33:Aggressive scan Output

- **Timing Templates**
  It can be used to set the timing template between 0-5 with intensity in that order [10].Passban shows no response towards this attack.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -T4 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:44 BST
Nmap scan report for 192.168.2.80
Host is up (0.0010s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
5200/tcp open  targus-getdata
5900/tcp open  vnc
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 34:Timing Templates command

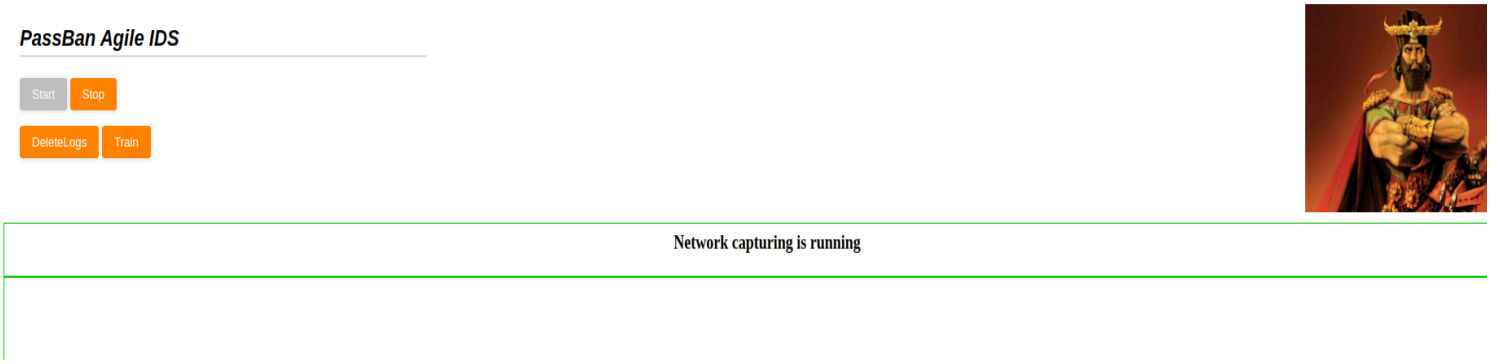

Figure 35:Timing Templates output

- **<u>IP protocol scan</u>**
  It scans all the protocols present in the Nmap database [12].Passban was able to detect this attack efficiently.

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sO 192.168.2.80 -vvvv
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-22 15:49 BST
```

Figure 36:IP Protocol scan command

```
PROTOCOL STATE         SERVICE REASON
1        open          icmp    echo-reply ttl 64
2        open|filtered igmp    no-response
6        open          tcp     proto-response ttl 64
17       open          udp     port-unreach ttl 64
103      open|filtered pim     no-response
136      open|filtered udplite no-response
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)
```

Figure 37:IP Protocol scan command's result

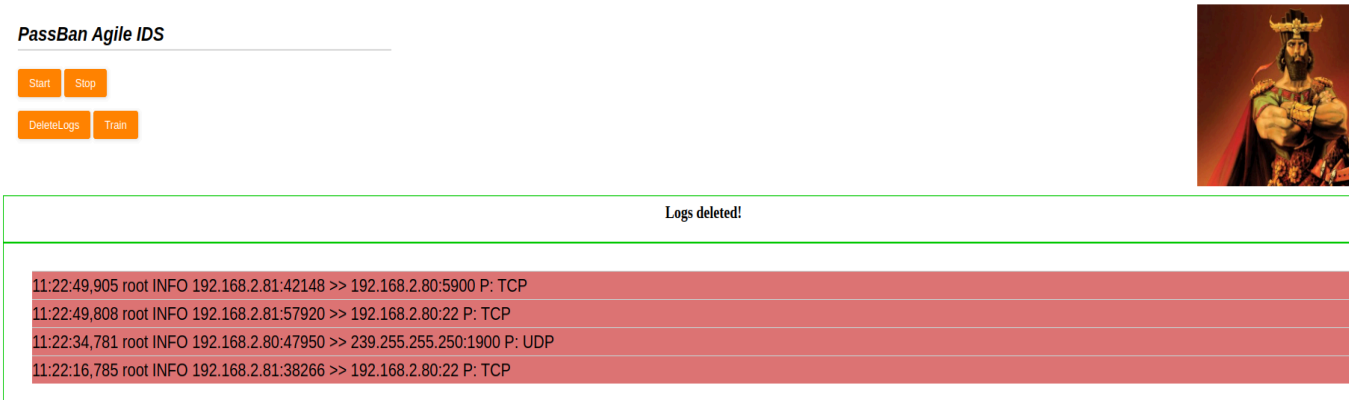

Figure 38:IP Protocol scan output

Scan Name	Scanning Commands	Passban's Response	Power Consumption
Ping Scan	Nmap -sn TARGET_IP_OR_RANGE	No Response	3.7 Watts
Basic Nmap syn scan	nmap TARGET_IP	No Response	3.8 Watts
TCP connect scan	nmap -sT TARGET_IP	No Response	3.6 Watts
UDP scan	nmap -sU TARGET_IP	Shows Response	4.2 Watts
Stealth syn scan	nmap -sS TARGET_IP	No Response	3.8 Watts
Null scan	nmap -sN TARGET_IP	No Response	3.7 Watts
FIN scan	nmap -sF TARGET_IP	No Response	3.8 Watts
XMAS syn	nmap -sX TARGET_IP	No Response	4.1 Watts
ACK scan	nmap -sA TARGET_IP	No Response	3.9 Watts
Idle scan	nmap -sI ZOMBIE_IP TARGET_IP	No Response	4.1 Watts
Version detection	nmap -sV TARGET_IP	Shows response	4.3 Watts
Version detection (stealth scan)	Nmap 192.168.2.80 -sV -T2	Shows Response	3.9 Watts
OS detection	nmap -O TARGET_IP	No Response	4.3 Watts
Script scan	nmap --script=vuln TARGET_IP	Shows response	3.9 Watts
Aggressive scan	nmap -A TARGET_IP	Shows response	4.0 Watts
Timing templates	nmap -T4 TARGET_IP	No Response	4.5 Watts
Ip protocol scan	nmap -sO TARGET_IP	Shows Response	3.8 Watts

Table 2: Nmap scan results and the power consumption during those scans

In conclusion, I ran 17 various Nmap scans and Passban was able to identify 6 scans. It shows that there is a scope for improvement in the passban IDS so that it becomes more sensitive toward various scans.

- **HTTP Brute force**

  In the context of the Passban paper, HTTP brute force refers to brute force attacks targeting the web interface or services exposed by the IoT gateway. An HTTP brute force attack would involve an attacker attempting to guess the username and/or password for this web interface by trying a large number of common or possible credentials [3].

  In order to check the credibility of passban in terms of brute force attack, I used nikto. Nikto is an open-source web server scanner that can be used to perform brute-force attacks on web interfaces. It is designed to scan web servers and web applications to find common vulnerabilities like insecure HTTP methods, outdated server software, default credentials, etc. One of its scan options allows for performing brute force credential guessing attacks. It tries a default set of common usernames and passwords, similar to using a dictionary attack.

  I used Nikto to execute an HTTP brute force attack against the web interface on port 5200 of the gateway at 192.168.2.80. This attack traffic generated by Nikto was then detected by Passban as an HTTP brute force attempt.

  I tried the command "**nikto -h http://192.168.2.80:5200**" and monitored Passban's response.

```
msc-lab@msclab-Latitude-5431:~$ nikto -h http://192.168.2.80:5200
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          192.168.2.80
+ Target Hostname:    192.168.2.80
+ Target Port:        5200
+ Start Time:         2023-08-14 17:26:34 (GMT1)
---------------------------------------------------------------------------
```

Figure 39:HTTP brute force command

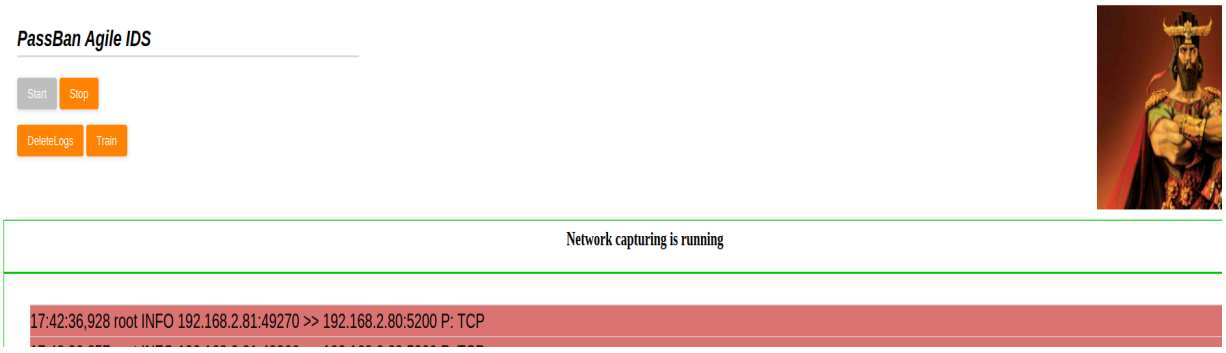

Figure 40:HTTP brute force output

In conclusion, passban was able to successfully detect the HTTP brute force attack performed.

- **SSH Brute Force**

  SSH, or Secure Shell, is a network protocol used to securely log into remote systems and servers over an encrypted connection. It provides secure remote access to command lines and shell prompts on systems like servers and networking equipment. Authentication is typically done using a username and password credentials [13].

  An SSH brute force attack involves trying to guess the username and password for the SSH service on a target system. If successful, the attacker gains remote command line access to the system's shell, potentially with administrative privileges. With the help of a port scan, I can easily identify that SSH is running on the system so it can be targeted and I tried brute force and monitored Passban's response [13].

  I tried the command "**hydra -l berry -P /home/msc-lab/SecLists/Passwords/200-200_most_used_passwords.txt q92.168.2.80 ssh"** to attempt the attack.

```
msc-lab@msclab-Latitude-5431:~$ hydra -l berry -P /home/msc-lab/SecLists/Passwords/2020-200_most_used_passwords.txt 192.168.2.80 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illega
ay).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-14 17:31:48
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 197 login tries (l:1/p:197), ~13 tries per task
[DATA] attacking ssh://192.168.2.80:22/
```

Figure 41:SSH brute force command

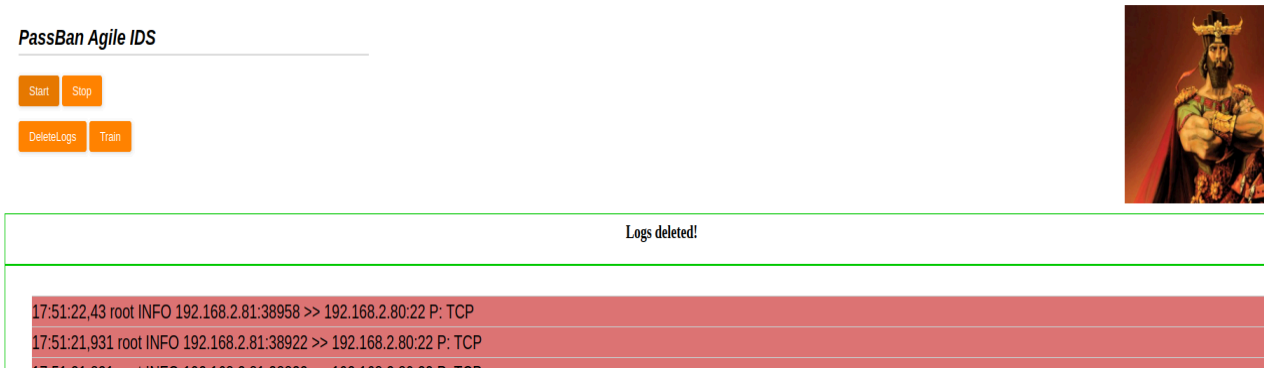

Figure 42:SSH brute force command

In conclusion, Passban was able to detect SSH brute force efficiently as we can see it flagged TCP port 22 which is the port number for SSH.

- **SYN flood**

  It is a form of denial-of-service (DoS) attack that exploits the TCP three-way handshake process. It involves sending a large number of SYN requests (the first step in a TCP handshake) to a target server. The server allocates resources for each SYN and replies with a SYN-ACK packet. However, the attacking system never responds to the SYN-ACK, leaving these half-open connections occupying server resources. Flooding a server with many thousands of these uncompleted connections can overwhelm its connection capacity. This leaves no resources for legitimate users to establish fully open connections with the server. The server is unable to respond to valid user traffic due to resource exhaustion. This causes DoS attacks toward legitimate users and leads to the degradation of overall server performance [12].

  In order to confirm if Passban can detect SYN flood, I tried using the hping command "**sudo hping3 -S --flood -V -p 5200 192.168.2.80**" to flood the device.

```
msc-lab@msclab-Latitude-5431:~$ sudo hping3 -S --flood -V -p 5200 192.168.2.80
using enp0s31f6, addr: 192.168.2.81, MTU: 1500
HPING 192.168.2.80 (enp0s31f6 192.168.2.80): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.2.80 hping statistic ---
119723032 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 43:SYN flood command

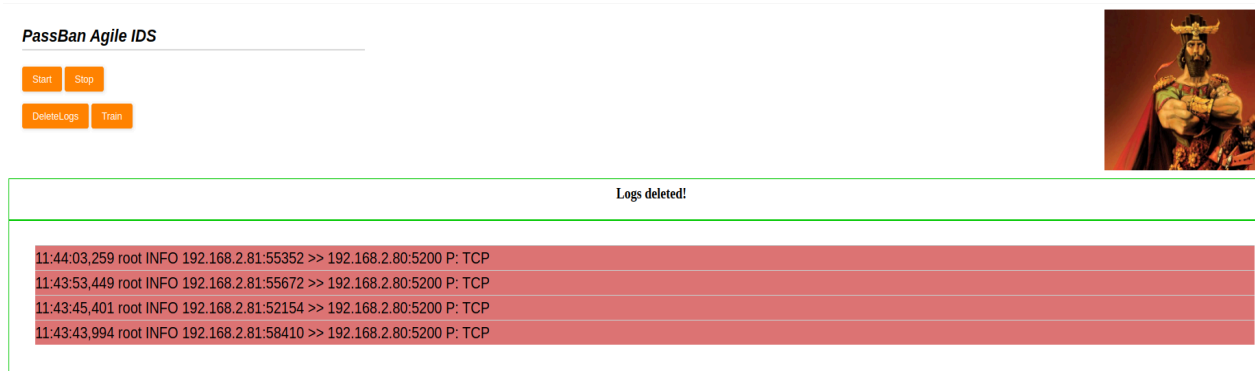

Figure 44:SYN flood output

Passban doesn't always detect the SYN flood. For me, it was able to detect the syn flood only when I installed and ran the image for the first time after that it showed no response. In conclusion, passban can work on its capability to detect syn floods.

```
msc-lab@msclab-Latitude-5431:~$ sudo hping3 -S --flood -V -p 5200 192.168.2.80
using enp0s31f6, addr: 192.168.2.81, MTU: 1500
HPING 192.168.2.80 (enp0s31f6 192.168.2.80): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 45:SYN flood reattempt

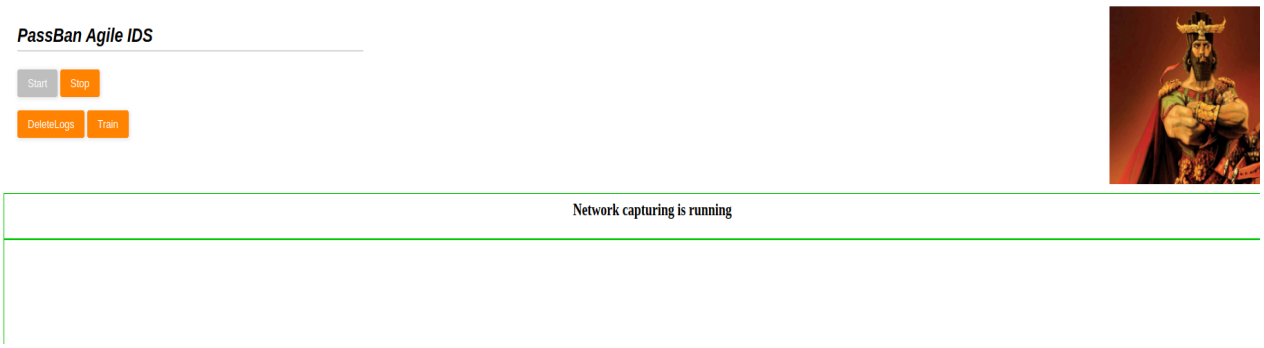

Figure 46:SYN flood reattempt output

Overall, after trying to attempt all the attacks mentioned in the paper, I can say that the authors provided limited information on Passban's capabilities as it has a lot more to it. It successfully identifies all 4 attacks but on its own terms and conditions. It doesn't always identify the SYN flood attack and easily misses lots of Nmap scans which can be really harmful for the host system. Unfortunately, these aren't the only attacks that happen in these systems. In the next section, I aim to test the passban against various other attacks and monitor its response for good.

Exploring Passban Beyond the Basics

The authors discussed that Passban was able to detect the 4 most common attacks. However, these are not the only attacks that happen in the world of IoT and Edge.
So, I decided to explore the forthcomings of Passban against 4 more attacks namely:

- **UDP flood**
  A denial of service attack that overwhelms target resources by flooding them with UDP packets. A number of packets are sent to the target system to tamper with the processing speed of the system as the system would reply with ICMP packets if the ports are closed [11] to random destination ports. It's almost similar to a syn flood but it uses UDP instead. This attack needs to be tested as it's a common flooding attack and can be performed easily due to the connectionless nature of the protocol and it generated high volumes of traffic that should be distinguishable by a good IDS. And it can be performed using tools like hping3. I tried UDP flood using the command "sudo hping3 -S –flood -V -p 5200 192.168.2.80 ".

Passban was not able to detect high volumes of anomalous UDP traffic that deviated from standard patterns.

```
msc-lab@msclab-Latitude-5431:~$ sudo hping3 --flood -a 192.168.2.81 -p 5200 192.168.2.80
HPING 192.168.2.80 (enp0s31f6 192.168.2.80): NO FLAGS are set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 47:UDP flood command

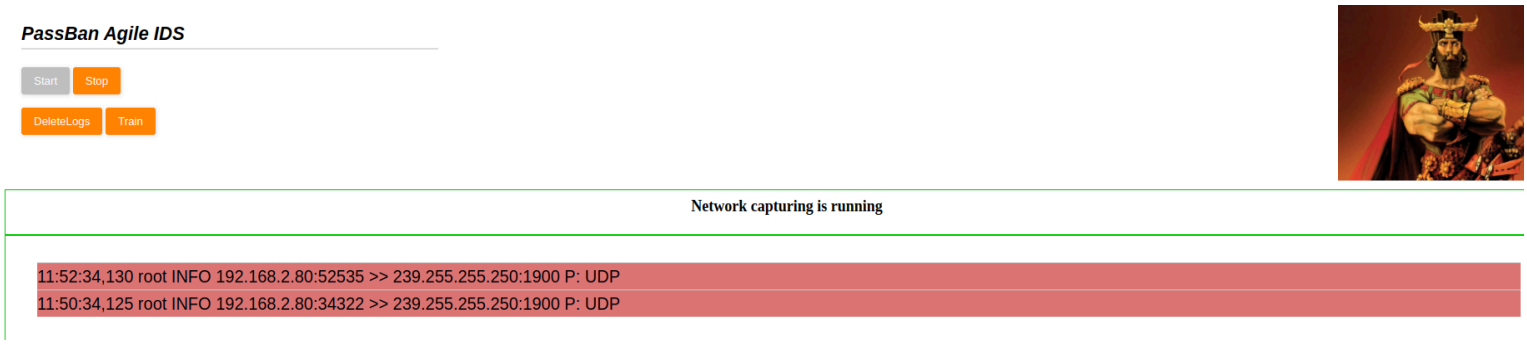

Figure 48:UDP flood output

- **ICMP flood**

  In this attack, the target is flooded with large volumes of ICMP echo request (ping) packets to exhaust the bandwidth of the network, causing service disruption and leading to a denial of service attack [12]. Similar to UDP flood, it also generates high volumes of anomalous data that should be detected by a good IDS makes it worthwhile to check the sensitivity of Passban against such an attack. To verify if Passban can detect such an attack, I used the command ”sudo hping3 -1 –flood -V 192.168.2.80”.

```
msc-lab@msclab-Latitude-5431:~$ sudo hping3 -1 --flood -V 192.168.2.80
using enp0s31f6, addr: 192.168.2.81, MTU: 1500
HPING 192.168.2.80 (enp0s31f6 192.168.2.80): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Figure 49:ICMP flood command

Passban should ideally detect highly abnormal volumes of ICMP traffic compared to normal levels as it’s an anomaly-based IDS so anything beyond what is present in its trained model should be seen as one of its main benefits including the detection of new or zero-day attacks [3]. However, no response was shown. It wasn’t able to detect SYN flood too which leads me to the conclusion that it’s a possibility that the resource-constrained nature of edge devices makes it very

difficult to detect flood attacks. However, no resource exhaustion was noticed as the Raspberry worked fine and the system didn't crash.

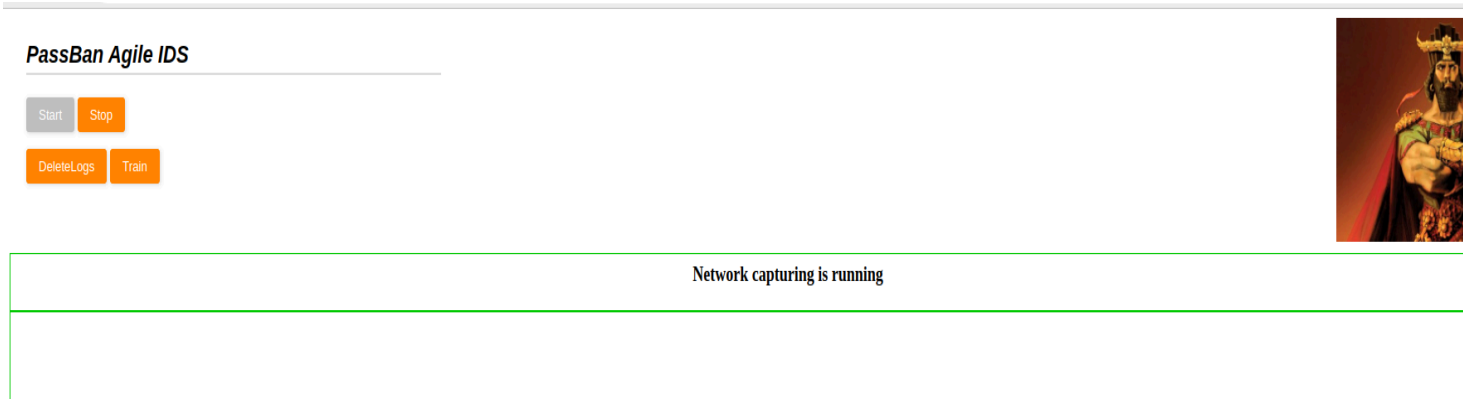

Figure 50:ICMP flood output

- **VNC brute force**

  VNC stands for virtual network computing where a vnc client can access the vnc server can look at all the activities on that machine. This is enabled by RFB(Remote FrameBuffer In order to prevent unauthorized access, the VNC server is usually equipped with a password. If a malicious user tries to gain access to that password by sending a number of random password combinations in order to find the right one that fits, it's called a VNC brute force attack. This attack performs automated password guessing against the VNC remote access service. VNC often uses weak credentials vulnerable to brute forcing [13]. It's an application-layer attack and easy to implement using tools like hydra so, Passban's sensitivity to detect such an attack seems like a good approach to try.
  Before performing VNC brute force it's essential to check if vnc is running on the device, This can be done with the help of a simple Nmap scan and if TCP port 5900, vnc is available, this track can be performed. This was my first step and once I realized it was open, I tried VNC brute force using a tool called "hydra" which is a very popular tool used for cracking passwords.

```
msc-lab@msclab-Latitude-5431:~$ hydra -P /home/msc-lab/SecLists/Passwords/Keyboard-Combinations.txt -s 5900 -v -V -t 1 -f 192.168.2.80 vnc
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purpos
ay).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-22 16:34:32
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting
[DATA] max 1 task per 1 server, overall 1 task, 9604 login tries (l:1/p:9604), ~9604 tries per task
[DATA] attacking vnc://192.168.2.80:5900/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 192.168.2.80 - login "" - pass "zaq1zaq1" - 1 of 9604 [child 0] (0/0)
```

Figure 51:VNC brute force command

Passban was able to detect this attack accurately in the screenshot we can see it's flagging port 5900 which is essentially the port number for enabled vnc service because this behavior is far from Passban's normal trained traffic behavior.

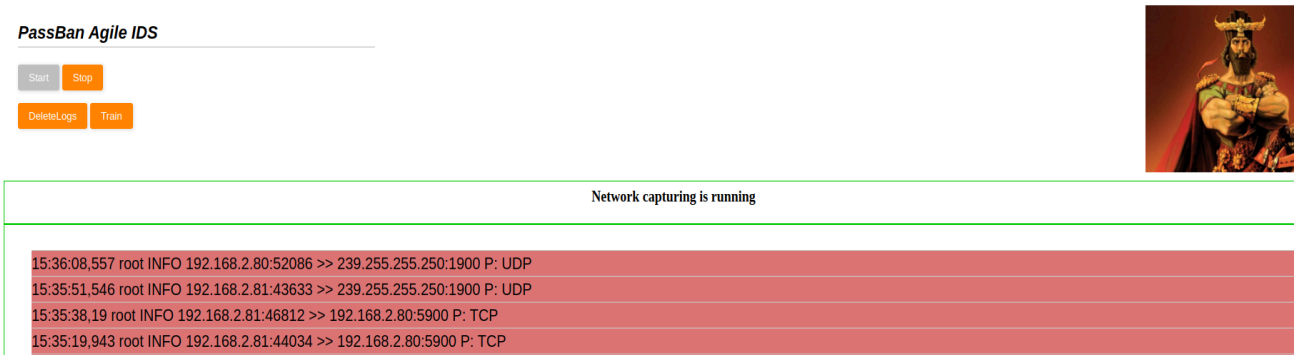

Figure 52:VNC brute force output

In conclusion, it shows that passban, if trained properly, might be a very useful and powerful IDS.

- **ARP spoofing**

  ARP is an address resolution protocol that helps the node that is sending data to a certain IP to figure out the MAC address of that destination node and create an ARP table. ARP spoofing is a technique to manipulate local ARP caches to link the attacker's MAC address with the IP address of a target device. This allows the attacker to intercept traffic intended for the victim [14]. It's a common Man-in-the-middle attack and it's essential of a good IDS to detect such low-level traffic which doesn't overwhelm the network. If Passban detects such an attack, it might be categorzied as one of the finest IDS, so it's essential to check such an attack

  I implemented using tools like arpspoof on Linux with the help of the command "**sudo arpspoof -i enp0s31f6 -t 192.168.2.80 192.168.2.97**". Just to confirm the successful implementation of the attack, I analyzed the traffic using Wireshark.

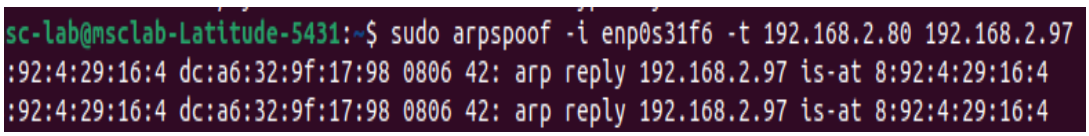

Figure 53: ARP spoofing command

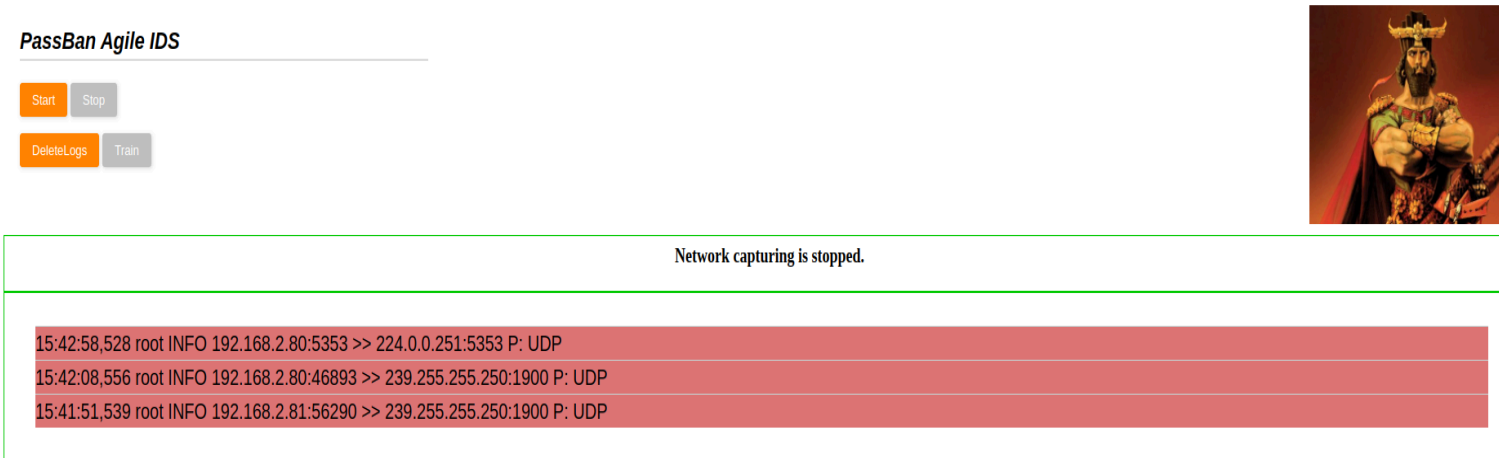

Figure 54: ARP spoofing output

Passban should detect ARP traffic anomalies and unexpected ARP cache changes compared to normal behavior. Unfortunately, it showed no significant response when the system was under this attack.

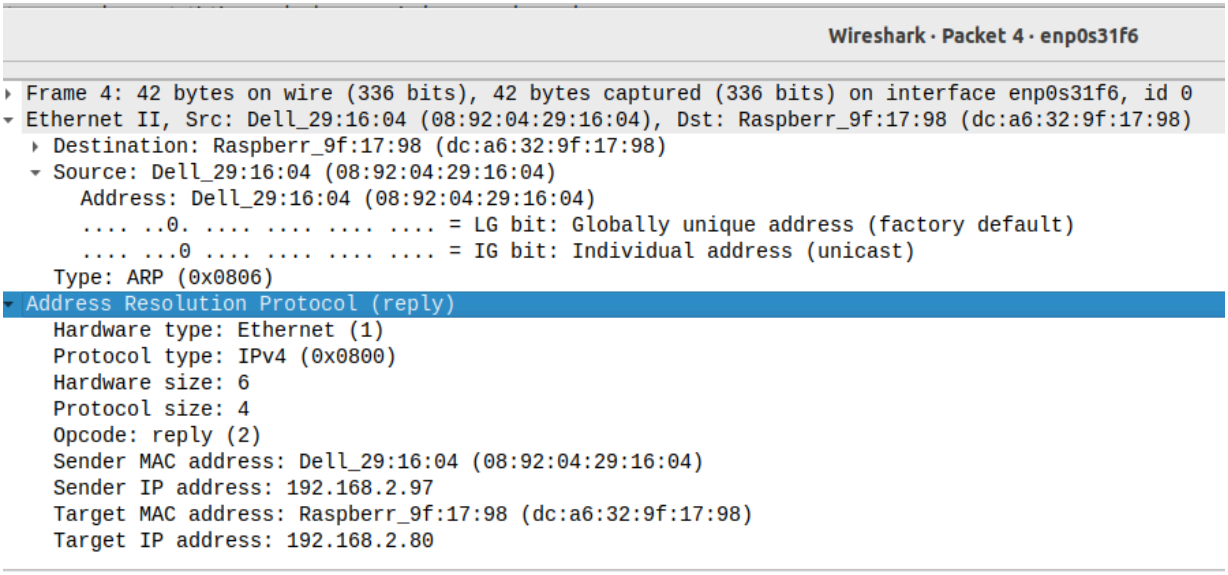

Figure 55: ARP traffic on wireshark

With this experiment, I derived a conclusion that Passban is effective in detecting a few if not all 4 attacks. However, it may become better than it is now if trained in a new way. In order to test the legitimacy of my statement I decided to train the model again in a slightly different manner that I'm going to discuss in the next section.

Passban's Detection of False Positives

There are four different kinds of alerts that can be generated by an IDS namely:

1. True Positive:When the IDS correctly identifies the execution of an attack [17].
2. True negative:This means that the normal traffic isn't been flagged by the IDS because it's not an attack [17].
3. False Positive:The IDS displays an event that isn't actually an attack [17].
4. False negative:When there's an attack that has happened but the IDS doesn't generate an alert for it by misconsidering it as benign traffic [17].

   This means that any IDS has the issue of generating false positives that implies that the IDS isn't too effective and degrades the performance of an IDS [17].

   Unfortunately, passban was also a prey to this and displayed two kinds of false positives during it's working explained as follows:

- **Multicast Traffic**

  Upon observing Passban's response initially, I saw a lot of traffic coming from IP addresses such as "239.255.255.250" which didn't seem like a legitimate IP address in my network. So, I looked it up and realized that the IPs ranging from 224.0.0.0 to 239.255.255.255 lie on the spectrum of IP multicast addresses [15]. So the IP that was being flagged is a multicast address.

  This IP is specific to SSDP(Simple Service Discovery Protocol) which is related to the set of UPnP(Universal plug-and-play) protocols and uses it for device discovery on a network. Service devices (SD) and control points (CP) are the two groups into which UPnP devices are divided. In essence, the service devices (SD) are servers that are in charge of providing a service. The clients who use the services offered by the SDs are known as control points (CP). After connecting to the network, an SD periodically transmits an SSDP NOTIFY message to the network at the standard address and port (239.255.255.250:1900) advertising its availability and giving information about the services it offers mentioned in the NT field. CPs on the other hand, look out for the services offered and send out a request using the SSDP M-search method and mention the service it's looking for in the ST field [16].

  Conclusively, this is a kind of benign traffic that uses UPnP which might have a few vulnerabilities such as the absence of verification in device advertisement and discovery, lack of authentication of access control, and event integrity check [16], is not necessarily an attack in itself. The fact that it's flagged as an attack in passban reduces its credibility as this is counted

as a false positive. An immense amount of such flags on Passban can even make it harder to spot an actual attack that might have taken place.

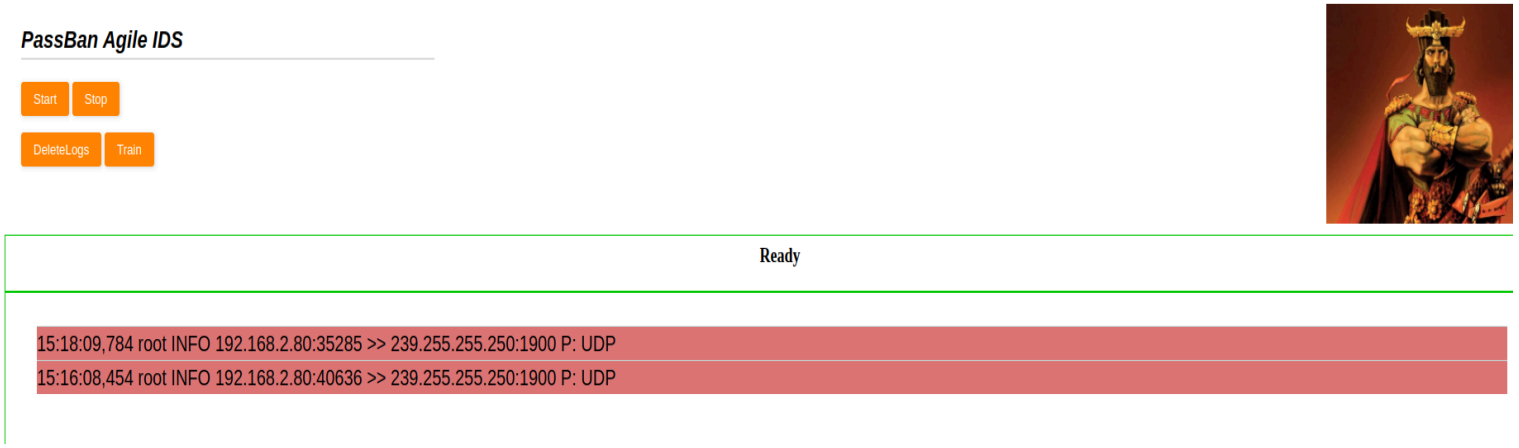

Figure 56: Random display of Multicast traffic

- **<u>Access to CSS file</u>**
  Upon using Nikto as mentioned before, access to CSS files was something that was pointed out as interesting. CSS stands for cascading stylesheets used to ass style and aesthetics to a web page to make it appear more user-friendly. It doesn't consist of any technical information. It might consist of a few section divider names but is irrelevant in this scenario. When I tried to access this file, it was flagged as an attack by Passban, however, technically no attack was implemented on it. This is also a false positive that was portrayed as an attack.

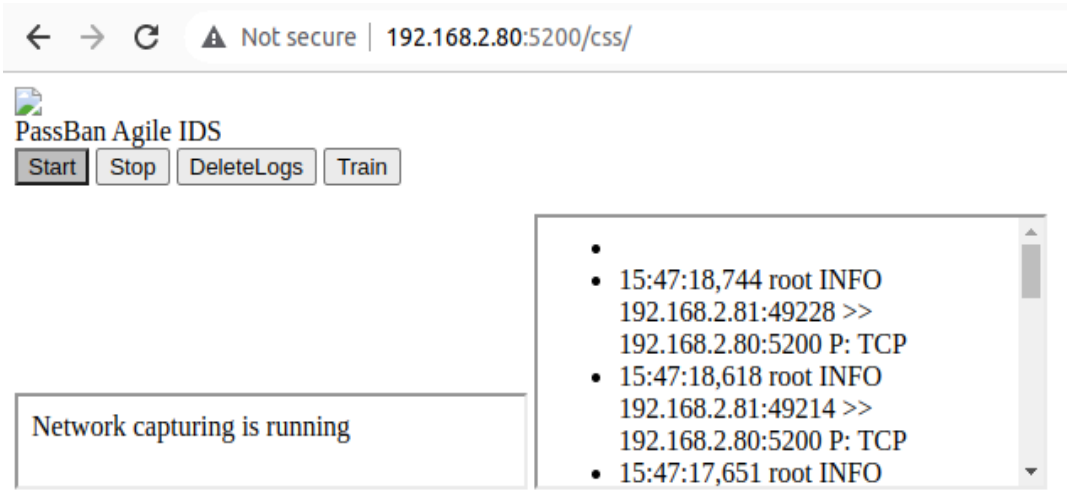

Figure 57: CSS file

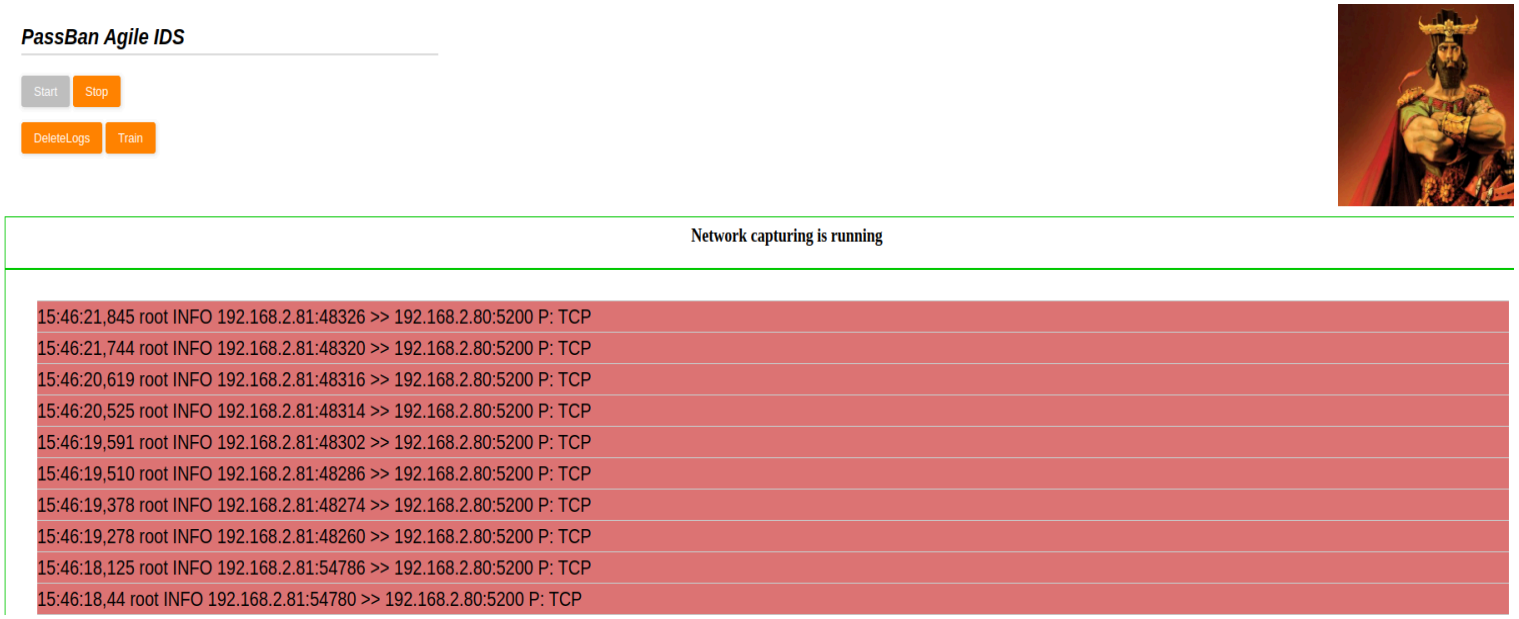

Figure 58: CSS file access being flagged by passban

These aspects of Passban were not mentioned in the paper making it difficult for the first-time user to understand the functioning of Passban and be able to figure out what can be counted as an actual attack.

Power consumption by the IDS

There is a surge in the level of technological devices that are being used everyday and they all use electricity or power to run. Electricity is a valuable resource that must be conserved in order to meet energy demands and requirements of the general public. One of the main benefits of using an Edge computing environment is to decrease the power consumption by the edge device. In order to deploy a powerful IDS, a more complex environment such as cloud is required but it consumes tremendous amounts of power. Whereas, deploying IDS on Edge provides various benefits and low power consumption is one of them as Raspberry would use all its resources and the system would shut down if all the case is the opposite.

I used a multimeter to measure the power consumption when the system is under various kinds of attacks and how the power consumption is impacted during detected vs. undetected attacks to get a better idea. The power consumption during attack-free mode was 3.3 Watts but a spike was observed during the implementation of each atack.

Hence, I created this table to summarize my observations which is displayed as follows:

Attack	Detected by Passban	Power Consumption
Attack free mode	NA	3.3 Watts
HTTP brute force	Yes	4.7 - 5.0 Watts
SSH brute force	Yes	4.1 - 4.3 Watts
SYN flood	No	4.9 - 5.3 Watts
UDP flood	No	5.3 - 5.7 Watts
ICMP flood	No	4.9 - 6.0 Watts
VNC flood	Yes	3.9 - 4.0 Watts
ARP flood	No	3.8 - 4.7 Watts
Nmap scans Average	Not all	3.96 Watts

Table 3: Power consumption during different attacks

While the power consumption was being spiked during the attacks, it was still in the reasonable range that can easily be handled by a small edge device like raspberry. As per my knowledge, the attacks that were not being detected were also causing an increase in power consumption which might be due to the sudden load of traffic that was increasing very fast in a short time consuming the resources. Although it doesn't impact the working efficiency of Passban, the spike during undetected attacks suggests that there might be a possibility of improvement. Overall, Passban can be categorized as a good IDS that doesn't overload the network and can be deployed to monitor attacks after some optimization in its ability to detect attack.

Can Passban be accurately retrained??

I planned to retrain the model to make it more efficient in the detection of the abovementioned 8 attacks. However, before training it's essential to understand if the model is being effectively trained or not. The authors advise that when trained the system should be in attack-free mode so all it receives is benign traffic and anything beyond that trained mode would be detected as an anomaly. To confirm this statement, I decided to do the contradiction. I attempted to attack it while training with the attacks namely- nmap scans, vnc and ssh brute force, which it was able to identify clearly as mentioned before. So now, if during training, the attacks were performed, the attack traffic would be considered benign and the model would not detect the attack as an anomaly anymore.

```
msc-lab@msclab-Latitude-5431:~$ nmap -sV 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-23 15:41 BST
Nmap scan report for 192.168.2.80
Host is up (0.0027s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Raspbian 5+deb11u1 (protocol 2.0)
5200/tcp open  http    PHP cli server 5.5 or later (PHP 5.6.33-0)
5900/tcp open  vnc     RealVNC Enterprise 5.3 or later (protocol 5.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 59: Training passban on a service scan

```
msc-lab@msclab-Latitude-5431:~$ nmap -A 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-23 15:42 BST
Nmap scan report for 192.168.2.80
Host is up (0.0027s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Raspbian 5+deb11u1 (protocol 2.0)
5200/tcp open  http    PHP cli server 5.5 or later (PHP 5.6.33-0)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
5900/tcp open  vnc     RealVNC Enterprise 5.3 or later (protocol 5.0)
```

Figure 60: Training passban on aggressive scan

```
msc-lab@msclab-Latitude-5431:~$ sudo nmap -sO 192.168.2.80
[sudo] password for msc-lab:
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-23 15:43 BST
Warning: 192.168.2.80 giving up on port because retransmission cap hit
Nmap scan report for 192.168.2.80
Host is up (0.00045s latency).
Not shown: 250 closed protocols
PROTOCOL STATE         SERVICE
1        open          icmp
2        open|filtered igmp
6        open          tcp
17       open          udp
103      open|filtered pim
136      open|filtered udplite
MAC Address: DC:A6:32:9F:17:98 (Raspberry Pi Trading)

Nmap done: 1 IP address (1 host up) scanned in 277.11 seconds
msc-lab@msclab-Latitude-5431:~$ nmap --script=vuln 192.168.2.80
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-23 15:48 BST
Nmap scan report for 192.168.2.80
```

Figure 61: Training passban on OS detection and script scan

```
msc-lab@msclab-Latitude-5431:~$ hydra -l berry -P /home/msc-lab/SecLists/Passwords/2020-200_most_used_passwords.txt 192.168.2.80 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purp
y).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-23 15:58:51
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriti
[DATA] max 16 tasks per 1 server, overall 16 tasks, 197 login tries (l:1/p:197), ~13 tries per task
[DATA] attacking ssh://192.168.2.80:22/
[STATUS] 120.00 tries/min, 120 tries in 00:01h, 83 to do in 00:01h, 16 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
msc-lab@msclab-Latitude-5431:~$ hydra -P /home/msc-lab/SecLists/Passwords/Keyboard-Combinations.txt -s 5900 -v -V -t 1 -f 192.168.2.80 vnc
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purp
y).
```

Figure 62: Training passban on ssh and vnc brute force

Train.csv is thoroughly trained specifying the source and destination address and the destination ports which shouldn't be flagged as malicious anymore because during training passban considers all traffic as benign.To my surprise, when the attacks were reattempted they were still considered malicious by the passban IDS and were portrayed as alerts on the web interface and in the logs file.

Here's the results of the trained csv file:

	srcip	srcport	dstip	dstport	proto	total_fpac	total_fvol	total_bpa	total_bvol	min_fpktl	mean_fpk	max_fpktl	std_fpktl	min_bpktl	mean_bpl	max_bpkt	std_bpktl	min_fiat
2	192.168.2.80	49240	239.255.255.250	1900	17	4	780	0	0	195	195	195	0	-1	-1	-1	0	0
3	192.168.2.80	44300	239.255.255.250	1900	17	4	780	0	0	195	195	195	0	-1	-1	-1	0	0
4	192.168.2.80	5353	224.0.0.251	5353	17	1	73	0	0	73	73	73	0	-1	-1	-1	0	0
5	192.168.2.81	5353	224.0.0.251	5353	17	1	73	0	0	73	73	73	0	-1	-1	-1	0	0
6	192.168.2.81	54718	192.168.2.80	5900	6	5	268	3	176	52	53	60	3	52	58	64	6	324
7	192.168.2.81	40782	192.168.2.80	22	6	5	268	3	206	52	53	60	3	52	68	94	22	327
8	192.168.2.81	51542	192.168.2.80	5200	6	5	272	3	164	52	54	60	3	52	54	60	4	324
9	192.168.2.81	34346	192.168.2.80	5200	6	6	338	5	1272	52	56	70	7	52	254	943	387	2
10	192.168.2.81	34350	192.168.2.80	5200	6	6	342	5	1272	52	57	74	8	52	254	943	387	30

Figure 63: Train.csv after being trained on attacks

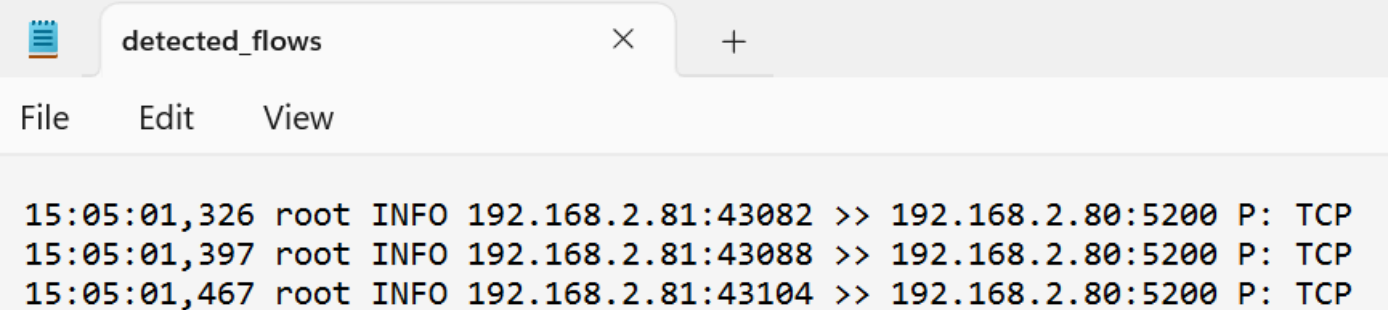

Figure 64: Attack flows detected after retraining

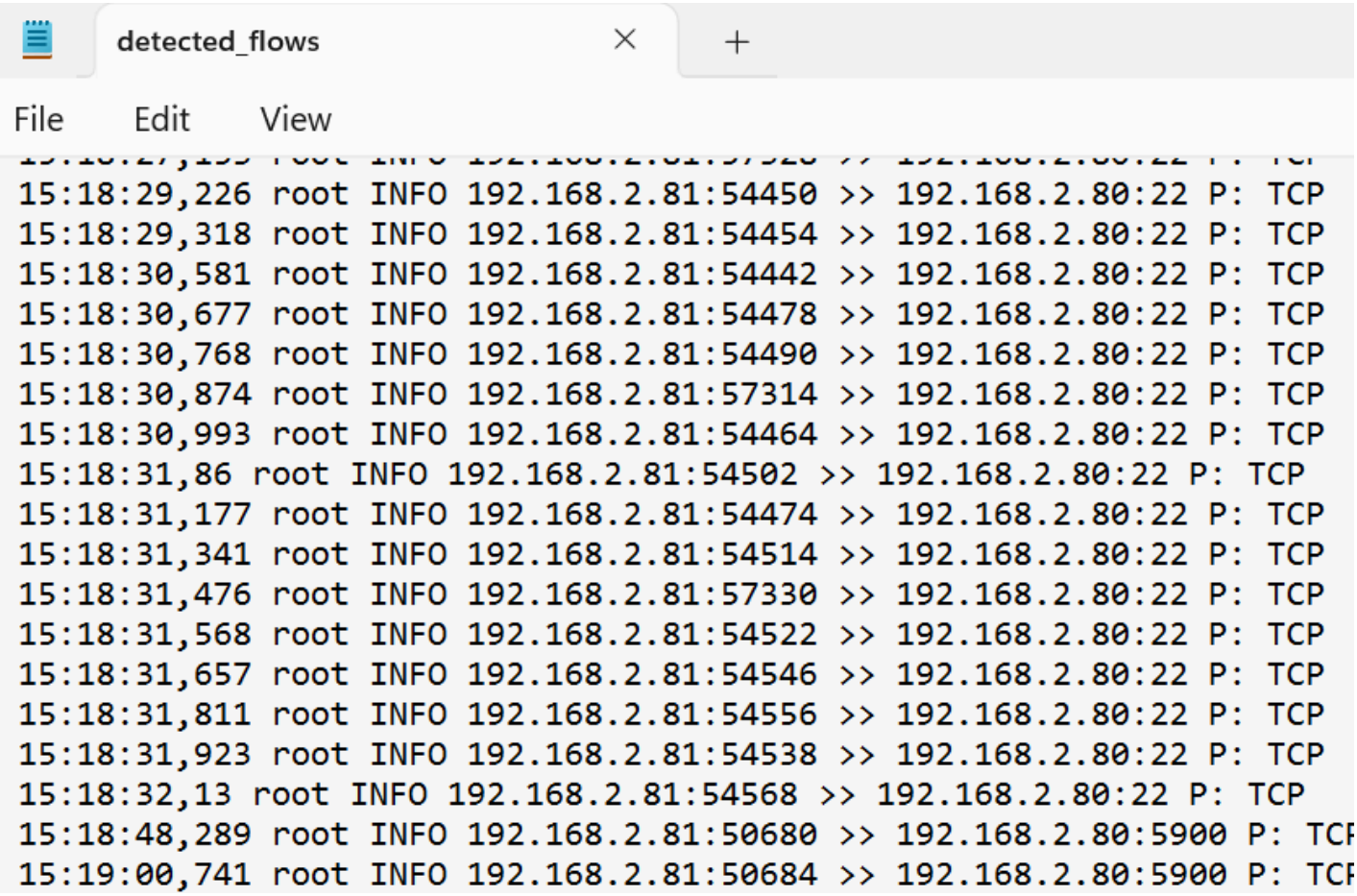

Figure 65: Attack flows detected after retraining

The training module wasn't working as per the expectations as it was still detecting all the attacks that I trained it on whereas in actuality it should've been considered as benign traffic, so I decided not to train it any further. Another bug that lies in the web interface is that you can't stop training by pressing the STOP button and neither by reloading the page. One has to actually close the browser, stop the execution of

command and reexecute it in order to run the new trained module. This is a significant flaw in the functioning of passban and requires an improvement.

Conclusion

This dissertation provided an in-depth examination of Passban, an anomaly-based network intrusion detection system designed for IoT edge devices.
Through extensive experimental analysis, I was able to conclude the following points:

- After evaluating the strengths and limitations of Passban, I figured out it's capabilities in detecting a range of network attacks relevant to modern IoT environments.
- My research validated that Passban can successfully detect some common attacks like HTTP brute force, SSH brute force, and VNC brute force which deviate significantly from normal traffic patterns.
- Passban also exhibited deficiencies in detecting stealthier attacks like various Nmap scans, flooding DDoS, and ARP spoofing which do not drastically alter traffic attributes.
- I also discovered issues with Passban incorrectly flagging benign multicast and web traffic as attacks, leading to false positives.
- Additionally, my experiments revealed that Passban's training module does not function as intended. Attacks performed during training were still detected as anomalies, rather than being incorporated into the model of normal behavior.
- Through metering power consumption, I confirmed Passban's suitability for resource-constrained edge devices, with only minor spikes during attacks.

This analysis suggests that there's a need for refinements in how the system learns expected traffic patterns. Power usage also increased for undetected attacks, indicating room for improvement in computational efficiency.
In conclusion, while Passban represents an important effort in anomaly detection for IoT edge networks, my research identified significant opportunities to enhance its attack detection capabilities, training process, and accuracy. My experiments aimed to pinpoint areas for improvement, in order to advance future evolutions of similar intrusion detection systems. Passban has immense potential as a lightweight yet powerful IDS in the domain of edge computing security.

References

1. Malik, Ibrahim, et al. "Emotions Beyond Words: Non-Speech Audio Emotion Recognition With Edge Computing." *arXiv preprint arXiv:2305.00725* (2023).
2. Premsankar, Gopika, Mario Di Francesco, and Tarik Taleb. "Edge computing for the Internet of Things: A case study." *IEEE Internet of Things Journal* 5.2 (2018): 1275-1284.
3. M. Eskandari, Z. H. Janjua, M. Vecchio and F. Antonelli, "Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6882-6897, Aug. 2020, doi: 10.1109/JIOT.2020.2970501.
4. Murshed, MG Sarwar, et al. "Machine learning at the network edge: A survey." *ACM Computing Surveys (CSUR)* 54.8 (2021): 1-37.
5. Zeyu, Huang, et al. "Survey on edge computing security." *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, 2020.
6. Mannanuddin, Khaja, et al. "Confluence of Machine Learning with Edge Computing for IoT Accession." *IOP Conference Series: Materials Science and Engineering*. Vol. 981. No. 4. IOP Publishing, 2020.
7. Pawlicki, Marek, et al. "The survey and meta-analysis of the attacks, transgressions, countermeasures, and security aspects common to the Cloud, Edge, and IoT." *Neurocomputing* (2023): 126533.
8. Liu, Bin, et al. "A survey of state-of-the-art on edge computing: Theoretical models, technologies, directions, and development paths." *IEEE Access* 10 (2022): 54038-54063.
9. Spadaccino, Pietro, and Francesca Cuomo. "Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning." *arXiv preprint arXiv:2012.01174* (2020).
10. Bennieston, Andrew J. "Nmap-a stealth port scanner." (2004).
11. K. Treseangrat, S. S. Kolahi and B. Sarrafpour, "Analysis of UDP DDoS cyber flood attack and defense mechanisms on Windows Server 2012 and Linux Ubuntu 13," 2015 International Conference on Computer, Information and Telecommunication Systems (CITS), Gijon, Spain, 2015, pp. 1-5, doi: 10.1109/CITS.2015.7297731.
12. Yu, Jaehak, et al. "An in-depth analysis on traffic flooding attacks detection and system using data mining techniques." *Journal of Systems Architecture* 59.10 (2013): 1005-1012.
13. Shruti, Ms, Ganeshbabu Aleti, and Ankush Kumar. "Invading Ssh, Telnet, Vnc Services with Metasploit and Thc-Hydra." *Think India Journal* 22.16 (2019): 4408-4416.
14. Al Sukkar, Ghazi, et al. "Address resolution protocol (ARP): Spoofing attack and proposed defense." (2016).
15. Williamson, Beau. *Developing IP multicast networks*. Vol. 1. Cisco Press, 2000.

16. G. Kayas, M. Hossain, J. Payton and S. M. R. Islam, "An Overview of UPnP-based IoT Security: Threats, Vulnerabilities, and Prospective Solutions," 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2020, pp. 0452-0460, doi: 10.1109/IEMCON51383.2020.9284885.
17. Bada, G., W. Nabare, and D. Quansah. "Comparative Analysis of the Performance of Network Intrusion Detection Systems: Snort Suricata and Bro Intrusion Detection Systems in Perspective." *International Journal of Computer Applications* 176.40 (2020): 39-44.
18. Oh, Doohwan, Deokho Kim, and Won Woo Ro. "A malicious pattern detection engine for embedded security systems in the Internet of Things." *Sensors* 14.12 (2014): 24188-24211.
19. J. P. Amaral, L. M. Oliveira, J. J. P. C. Rodrigues, G. Han and L. Shu, "Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks," 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 2014, pp. 1796-1801, doi: 10.1109/ICC.2014.6883583.
20. Riecker, Michael, Sebastian Biedermann, and Matthias Hollick. "Lightweight energy consumption based intrusion detection system for wireless sensor networks." *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 2013.
21. Raza, Shahid, Linus Wallgren, and Thiemo Voigt. "SVELTE: Real-time intrusion detection in the Internet of Things." *Ad hoc networks* 11.8 (2013): 2661-2674.
22. Linda, Ondrej, Todd Vollmer, and Milos Manic. "Neural network based intrusion detection system for critical infrastructures." *2009 international joint conference on neural networks*. IEEE, 2009.
23. Hodo, Elike, et al. "Threat analysis of IoT networks using artificial neural network intrusion detection system." *2016 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2016.
24. Habibi, Javid, et al. "Heimdall: Mitigating the internet of insecure things." *IEEE Internet of Things Journal* 4.4 (2017): 968-978.
25. Singh, Ashish, Kakali Chatterjee, and Suresh Chandra Satapathy. "An edge based hybrid intrusion detection framework for mobile edge computing." *Complex & Intelligent Systems* 8.5 (2022): 3719-3746.
26. Guezzaz, Azidine, et al. "A lightweight hybrid intrusion detection framework using machine learning for edge-based IIoT security." *Int Arab J Inf Technol* 19.5 (2022).
27. Haq, Mohd Anul, Mohd Abdul Rahim Khan, and Talal AL-Harbi. "Development of PCCNN-Based Network Intrusion Detection System for EDGE Computing." *Computers, Materials & Continua* 71.1 (2022).
28. Nasir, Muneeba, et al. "Feature engineering and deep learning-based intrusion detection framework for securing edge IoT." *The Journal of Supercomputing* (2022): 1-15.