MAJOR PROJECT REPORT

ON

"DDOS Detection System"

JAGAN INSTITUTE OF MANAGEMENT STUDIES

(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY)

jims
Sector-5, Rohini, Delhi

Master of Computer Application

(2022-24)

Submitted By:

Name: Ankur Chaurasia	**Enrollment No:02850404422**
Name: Himanshu Ahuja	**Enrollment No:08050404422**
Name: Sushant Choudhary	**Enrollment No:09450404422**

CONTENTS

List of Tables

List of Figures

List of Abbreviation

S. NO.	Topic	
1	Internet of Things	IoT
2	Software-Defined Networking	SDN
3	Distributed Denial of Service	DDoS
4	Web Application Firewall	WAF
5	Network Function Virtualization	NFV
6	Open Systems Interconnection	OSI
7	Hypertext Transfer Protocol	HTTP
8	Uniform Resource Locators	URL
9	Domain Name System	DNS
10	Support Vector Machine	SVM
11	Internet Control Message Protocol	ICMP

CHAPTER 1

INTRODUCTION

The Internet, a global network of networks, is an astonishingly sophisticated technical system that was developed from the 1950s to the present by scientists from all over the world. Governments, researchers, educators, and people have touted the Internet and other networks as tools for addressing a variety of human needs throughout its history. The democratisation of content creation, increased scientific collaboration, economic growth, the formation of virtual communities and an improvement in the ability to maintain social ties over long distances are just a few examples of the social benefits that have resulted from the combination of high-level policy and grassroots improvisation. The Internet's explosive expansion has also resulted in societal problems like harmful and illegal activity as well as persisting digital inequalities based on factors like money, location, age, gender, and education, as well as technical problems like congestion and a lack of network addresses.

Without the internet, the world would not have changed into what it is now. It affects almost all facets of our daily lives, including how we work, socialise, shop, and play. However, the availability of the internet is a relatively recent development that has dramatically altered the course of human history. The internet has transformed in just a few decades from a cutting-edge method enabling the US military to communicate to the constant beating of humanity. As more and more individuals acquire access to the internet each year, the demand for networks and networking has increased as well. This is how they have logged on. Dial-up speeds are often slower than those of broadband. The internet became what it is today because to these speeds since in its early years, even simple images on web pages could take several minutes to load. Higher speeds would allow websites to load more quickly and allow site creators to add more material without worrying that it would cause users' machines to crash. All facets of our daily lives began to adapt to their work and use all essential services at the same time due to the emergence of the internet and networking and the high speed of connectivity it provided. Websites evolved from simple destinations to interactive places where people could buy things and communicate with each other in real time.

Though there are some hazards and threats that will always be present to compromise the network system's integrity and security, in addition to the many benefits of the expanding networking in our online or cyber world. Numerous opportunities for attackers to break into specific networks have increased as a result of the global networking boom. Therefore, being friendly with networking also necessitates exercising prudence and being knowledgeable about network security. Network security, according to the SANS Institute, is the process of taking precautions to guard against unauthorised access, misuse, malfunction, modification, destruction, or improper disclosure of the underlying networking infrastructure. By putting these protections in place, computers, users, and programmes are able to carry out their authorised crucial functions in a secure setting.

A group of technologies known as network security guard against a wide range of potential dangers from entering or spreading within a network, preserving the usefulness and integrity of a company's infrastructure.

Network security is crucial for both personal and professional networks. Most houses with high-speed internet have one or more wireless routers, which can be vulnerable to attacks if they are not adequately secured. There are numerous breaches of computer security every day all across the world. Many of them are viewed as serious, or even catastrophic, with significant loss of data or financial resources. However, some are viewed as trivial. Hackers are constantly seeking for new weaknesses to take advantage of. Information about businesses, people, and even our government is vulnerable to exposure or use against us when networks are not secure.

Reduced danger of data loss, theft, and sabotage is one benefit of a strong network security system. Delivering goods and services to customers and employees while maintaining network security is essential for any firm. Protecting network apps and data is crucial for corporate growth, let alone preserving an organization's reputation, whether they be workplace programmes, remote desktops, or online storefronts. Effective network security can also increase network performance by preventing downtime brought on by successful attacks. When addressing network security across an enterprise, there are numerous layers to take into account. The network security layers paradigm allows for attacks to occur at any layer, therefore the hardware, software, and rules for network security must be developed to address each region.

Better network security is now a crucial concern as the Internet of Things (IoT) network grows in popularity. Software Defined Networking (SDN), a modern networking approach utilised by systems and clouds, has several benefits over a traditional network. The physical separation of network control and forwarding devices is a key benefit of SDN. SDN can address many of a legacy network's security problems. SDN, however, has a lot of security flaws.

Scalability and security issues on the Internet of Things (IoT) network are just two of the difficulties that adapting Software Defined Networking (SDN) brings up. As networks get bigger, the SDN controller has to deal with more network traffic and security issues. Among the most serious dangers at the moment are distributed denial of service (DDoS) attacks. Attacks that cause a Distributed Denial of Service (DDoS) are the main concern with SDN vulnerabilities. The DDoS attack on SDN develops into a significant issue, and numerous techniques have been used for detection and mitigation.

1.1 Software-defined networking (SDN)

A more cost-effective network architecture than current network architectures is Software-Defined Networking (SDN), which is dynamic and manageable. This architecture's goal is to consolidate network hardware intelligence and channel it toward the management system.

The centralised SDN controller, which can be thought of as the single point of failure, oversees policies and traffic flow across the whole network. SDN controller's centralised structure makes it extremely susceptible to DDoS attacks. It is crucial to quickly identify DDoS attacks in the controller, for example.

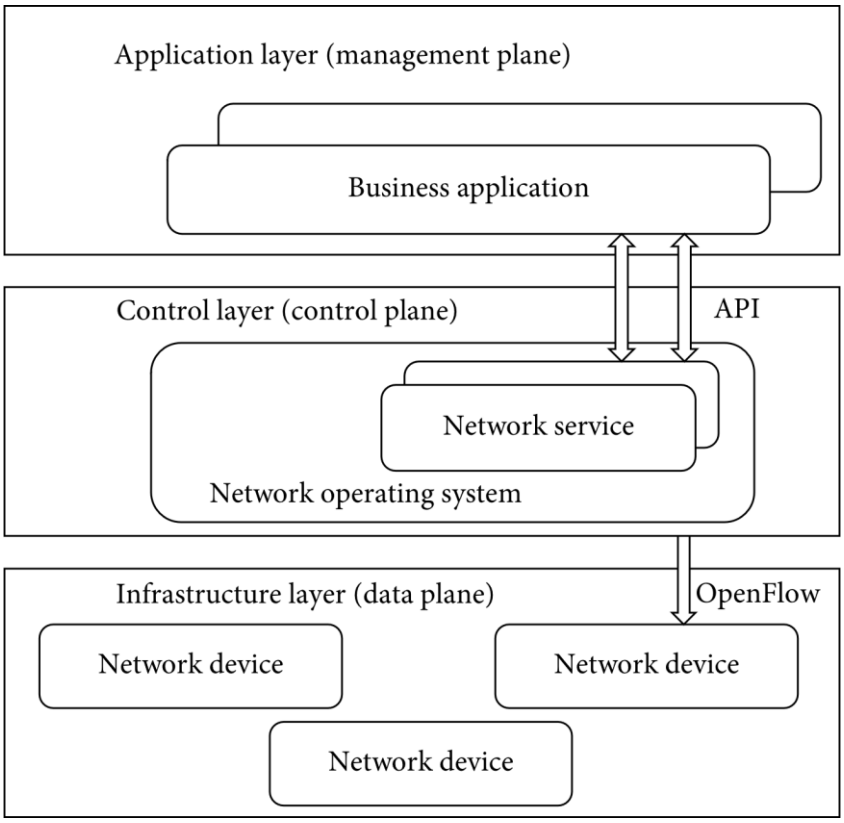


Fig. 1 – Architecture of SDN

It is often paired with network function virtualization (NFV), which separates network functions from hardware in the form of virtualized network functions (VNFs).

SDN enables cloud-like computing within a network. This enables network engineers and administrators to respond quickly to changes in business requirements through a centralized control console that is abstracted from the physical hardware of the network. In other words, SDN creates a centralized brain for the network that can communicate and command the rest of the network. SDN is used to create virtual overlay networks; software-defined networks that sit on top of the physical hardware infrastructure.

1.1.1 SDN Controllers

An SDN controller is the software that provides a centralized view of and control over the entire network. Network administrators use the controller to govern how the underlying infrastructure's forwarding plane should handle the traffic. The controller is also used to enforce policies that dictate network behaviour. Network administrators establish policies that are uniformly applied to multiple nodes in the network. Network policies are rules that are applied to traffic that determines what level of access it has to the network, how much resources it is allowed, or what priority it is assigned. Having a centralized view of the network and the policies in place makes for simpler management of the network that is more uniform and consistent.

1.2 Distributed Denial of Service (DDoS) Attack

A distributed denial-of-service (DDoS) attack is a malicious attempt to obstruct a server, service, or network's regular traffic by saturating the target or its surrounding infrastructure with an excessive amount of Internet traffic. The goal of a Distributed Denial of Service (DDoS) assault is to saturate a website with traffic from several sources in an effort to render it unusable. They pose a significant problem for ensuring that people can publish and access crucial information since they target a wide range of crucial resources, from banks to news websites.

By using numerous compromised computer systems as sources of attack traffic, DDoS attacks are made effective. Computers and other networked resources, like as IoT devices, can be exploited machines.

When viewed from a distance, a DDoS assault resembles an unexpected traffic congestion that blocks the roadway and keeps ordinary traffic from reaching its destination.

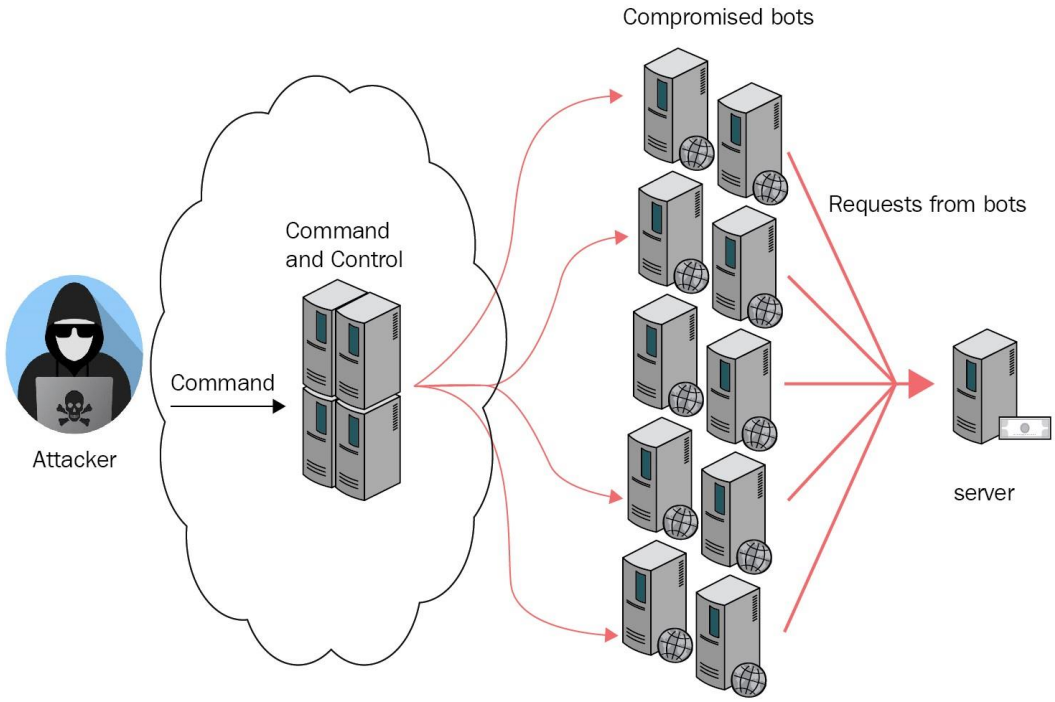


Fig. 2 – DDoS Attack

1.2.1 How does a DDoS attack work?

DDoS attacks are carried out with networks of Internet-connected machines.

These networks are made up of computers and other gadgets that have been infected with malware, allowing an attacker to remotely manipulate them (such as IoT gadgets). These particular gadgets are known as bots (or zombies), and a botnet is a collection of bots.

Once a botnet has been established, the attacker is able to direct an attack by sending remote instructions to each bot or generally known as host too in terms of networking.

Each bot in the botnet sends queries to the IP address of the victim's server or network when that server or network is being targeted by the botnet. This could overload the server or network and result in a denial-of-service attack on regular traffic.

Because each bot is a legitimate Internet device, separating the attack traffic from normal traffic can be difficult.

1.2.2 What are some common types of DDoS attacks?

Different types of DDoS attacks target varying components of a network connection. In order to understand how different DDoS attacks work, it is necessary to know how a network connection is made.

A network connection on the Internet is composed of many different components or “layers”. Like building a house from the ground up, each layer in the model has a different purpose.

The OSI model, shown below, is a conceptual framework used to describe network connectivity in 7 distinct layers.

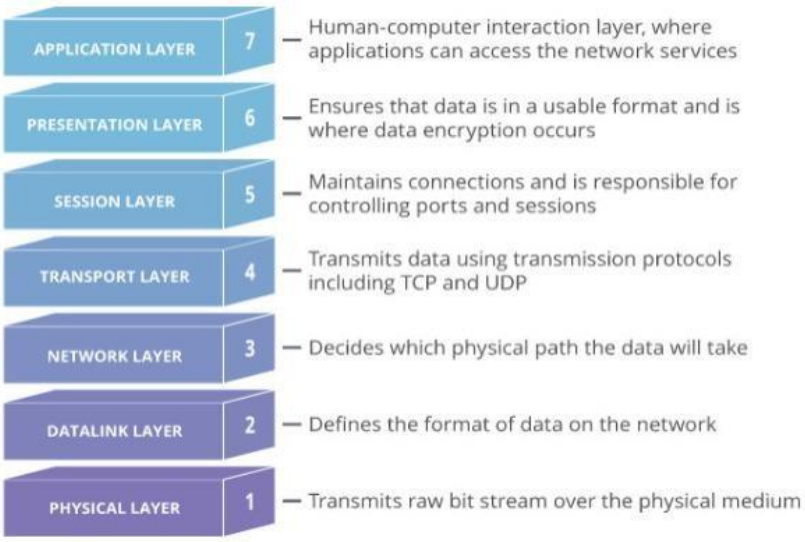


Fig. 3 – The OSI Model

1.2.2.1 Application layer attacks

The goal of the attack:

Sometimes referred to as a layer 7 DDoS attack (in reference to the 7th layer of the OSI model), the goal of these attacks is to exhaust the target's resources to create a denial-of-service.

The attacks target the layer where web pages are generated on the server and delivered in response to HTTP requests. A single HTTP request is computationally cheap to execute on the client side, but it can be expensive for the target server to respond to, as the server often loads multiple files and runs database queries in order to create a web page.

Layer 7 attacks are difficult to defend against, since it can be hard to differentiate malicious traffic from legitimate traffic.

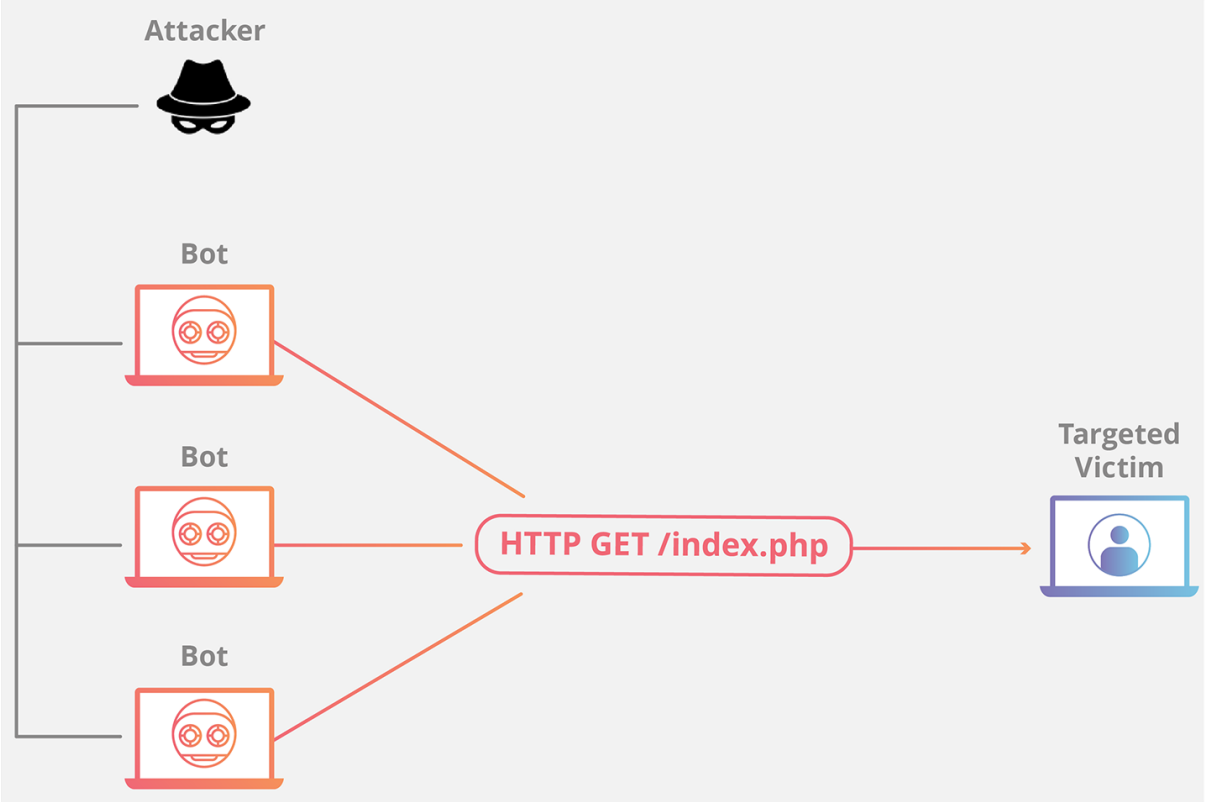


Fig. 4 – Example of Application Layer Attack

➢ **HTTP Flood**

This attack is similar to pressing refresh in a web browser over and over on many different computers at once – large numbers of HTTP requests flood the server, resulting in denial-ofservice.

This type of attack ranges from simple to complex. Simpler implementations may access one URL with the same range of attacking IP addresses, referrers and user agents. Complex versions may use a large number of attacking IP addresses, and target random URLs using random referrers and user agents.

1.2.2.2 Protocol Attacks

The goal of the attack:

Protocol attacks, also known as a state-exhaustion attacks, cause a service disruption by overconsuming server resources and/or the resources of network equipment like firewalls and load balancers.

Protocol attacks utilize weaknesses in layer 3 and layer 4 of the protocol stack to render the target inaccessible.

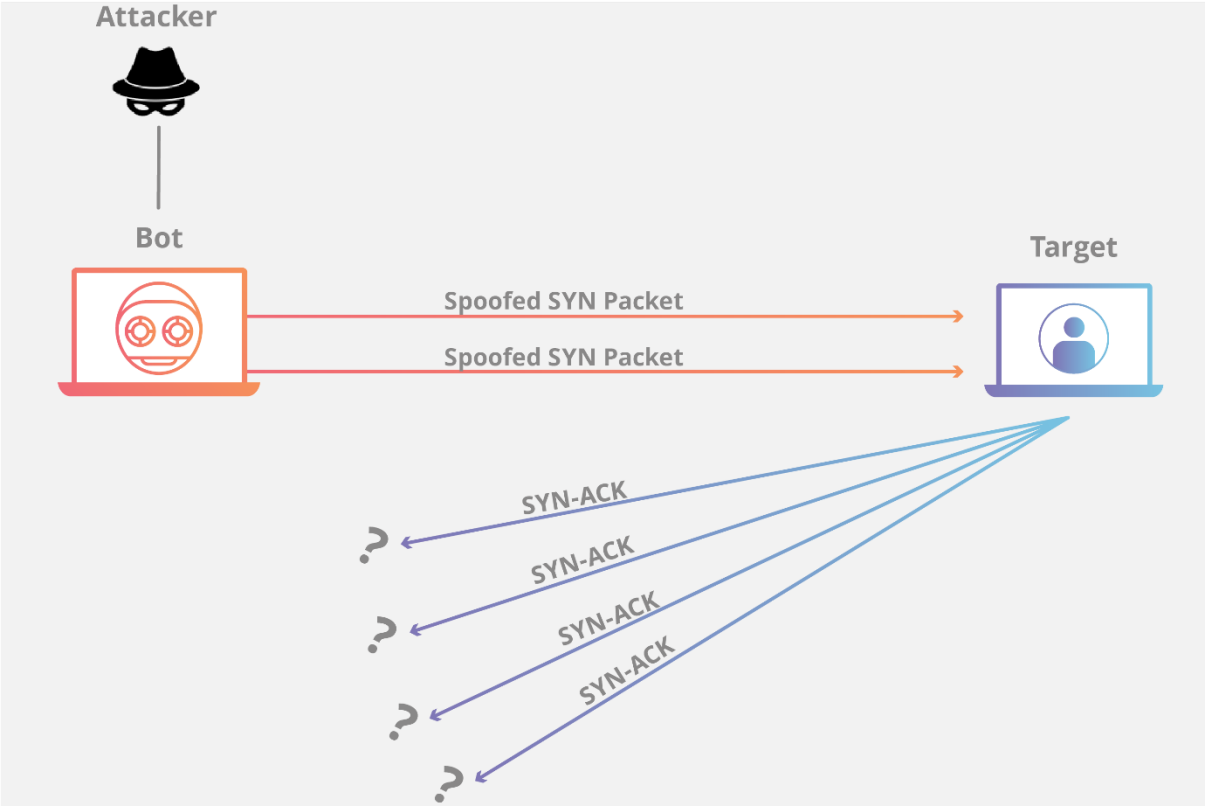


Fig. 5 - Protocol Attack Example

➢ **SYN Flood**

A SYN Flood is analogous to a worker in a supply room receiving requests from the front of the store. The worker receives a request, goes and gets the package, and waits for confirmation before bringing the package out front. The worker then gets many more package requests without confirmation until they can't carry any more packages, become overwhelmed, and requests start going unanswered.

This attack exploits the TCP handshake — the sequence of communications by which two computers initiate a network connection — by sending a target a large number of TCP "Initial Connection Request" SYN packets with spoofed source IP addresses.

The target machine responds to each connection request and then waits for the final step in the handshake, which never occurs, exhausting the target's resources in the process.

1.2.2.3 Volumetric Attacks

The goal of the attack:

This category of attacks attempts to create congestion by consuming all available bandwidth between the target and the larger Internet. Large amounts of data are sent to a target by using a form of amplification or another means of creating massive traffic, such as requests from a botnet.

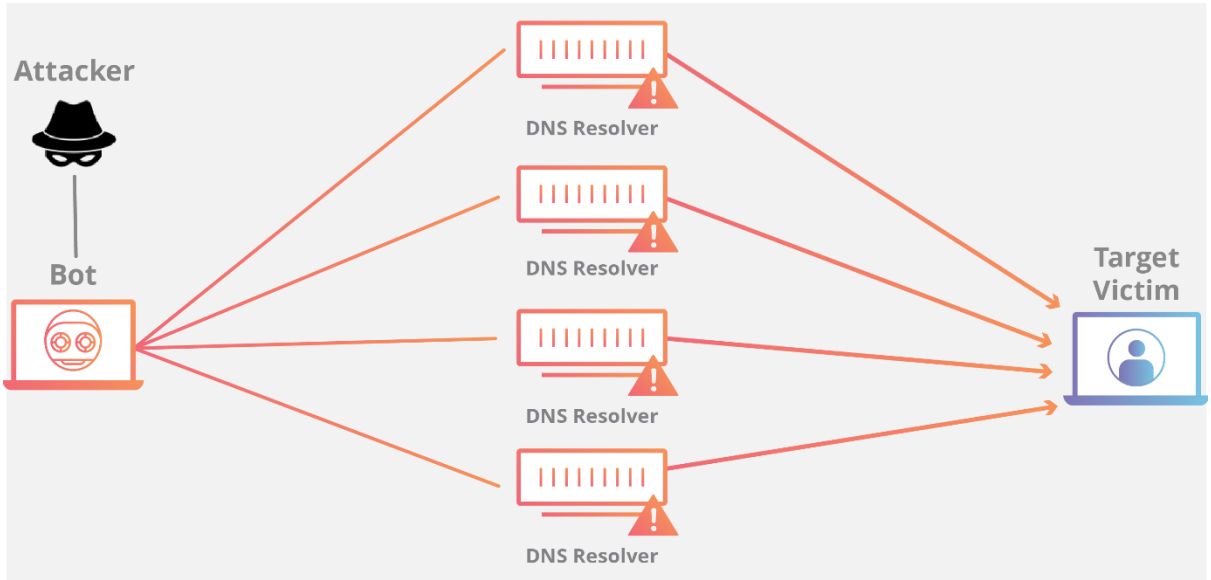


Fig. 6 - Amplification example

➢ **DNS Amplification**

A DNS amplification is like if someone were to call a restaurant and say “I ’ll have one of everything, please call me back and repeat my whole order,” where the call-back number actually belongs to the victim. With very little effort, a long response is generated and sent to the victim.

By making a request to an open DNS server with a spoofed IP address (the IP address of the victim), the target IP address then receives a response from the server.

1.2.3 How to identify a DDoS Attack

The most obvious symptom of a DDoS attack is a site or service suddenly becoming slow or unavailable. But since a number of causes — such a legitimate spike in traffic — can create similar performance issues, further investigation is usually required. Traffic analytics tools can help us to spot some of these tell-tale signs of a DDoS attack:

- Suspicious amounts of traffic originating from a single IP address or IP range
- A flood of traffic from users who share a single behavioural profile, such as device type, geolocation, or web browser version
- An unexplained surge in requests to a single page or endpoint
- Odd traffic patterns such as spikes at odd hours of the day or patterns that appear to be unnatural (e.g. a spike every 10 minutes)

There are other, more specific signs of DDoS attack that can vary depending on the type of attack.

1.3 Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

1.3.1 SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

CHAPTER 2

Related Work

Distributed denial-of-service (DDoS) attack is a rapidly growing threat to today's Internet. Significant research works have been done in this area. Some of the research works have been added here to give a reference to the existing solution of our common problem statement.

2.1 Research Work

2.1.1

In the first part of research work, we found the design and implementation of a senior design project named DDoS Attack, Detection and Defence Simulation. This aims to build a test bed and configure the network environment to simulate the "real-world" DDoS attack, detection and defence. We perform extensive tests, collect and analyse the experimental data, and draw our conclusions. This is an on-going project. This project is divided into three phases.

Building a DDoS attack network and simulating DDoS attacks are the first steps in this project. They put certain DDoS attack tools already in use, including StacheldrahtV4, to the test. They also created a number of straightforward DDoS attack programmes on their own. programmes that can conduct ping flood attacks or UDP attacks, for instance. We run RealPlayer Multimedia Server and Apache Web Server on the victim network. The effects of DDoS assaults on typical users and traffic can be seen by the pupils.

The second phase is to set up DDoS Intrusion Detection and Defence systems. They used Snort, as well as several other leading products in market. Then, they performed extensive tests and compare these DDoS defence products. They designed their own defence mechanism by integrating Snort with Firewall and router to enable effective rate-limiting and QoS provisioning. This requires students to have good understanding on TCP/IP, IP-table, firewall and router.

The last part of this project is to further the studies on DDoS attacks. For example, there is a new type of DDoS attack called degrading DDoS attacks, or non-disruptive DDoS attacks. This type of DDoS attack consumes a large portion of victim network resources but does not stop the network services completely. The traditional DDoS defence mechanisms react poorly to degrading DDoS attacks.

2.1.2

Early investigation of IDS was carried out by Georgios Loukas and Glay, who began their work with the time-line significant DOS. In September 1996, a SYN Flood attack was discovered, a Smurf attack began in January 1998 and a HTTP flood was the modern DOS that began in 2004. Specialists in the area suggest complete protection architecture through detection which can either be anomaly-based or signature based, or a hybrid of these two methods. Classifications such as neural networks, radial basis functions and genetic algorithms are increasingly used in DoS detection because of the automatic classification they can offer. The protection system either drops the attacking packets in a timely fashion or renders them inoperable. Traffic rate, SYN and URG flags, as well as some specific ranges of ports, are the most significant for the identification of a DoS attack, as some investigators have tested in their surveys.

2.1.3

Zheng et al. implemented a Hierarchical Intrusion Detection (HIDE) system which can detect attacks using pre-processing statistical values and a Neural Network. They tested five classifiers: Perceptron, Backpropagation (BP), Perceptron-backpropagation-hybrid (PBH), Fuzzy ARTMAP, and Radial-based Function. They stated that BP and PBH achieved the highest accuracy rate for detecting and classifying network attacks.

Reyhaneh Karimazad and Ahmad Faraahi proposed an anomaly-based DDoS detection approach using an analysis of network traffic. A radial-based function (RBF) Neural Network was used in this approach, and they tested their method on a UCLA dataset, achieving 96% accuracy rate for a DDoS attack.

2.2 Existing Systems

Differentiating between attack traffic and regular traffic is the main issue when attempting to mitigate a DDoS attack. Cutting off all traffic is a mistake, for instance, if a product release has a company's website flooded with enthusiastic buyers. It is likely important to take action to stop an attack if a company suddenly experiences a spike in traffic from recognised attackers. It is challenging to distinguish between legitimate clients and attack traffic.

In the modern Internet, DDoS traffic comes in many forms. The traffic can vary in design from un-spoofed single source attacks to complex and adaptive multi-vector attacks. A multi-vector DDoS attack uses multiple attack pathways in order to overwhelm a target in different ways, potentially distracting mitigation efforts on any one trajectory.

An attack that targets multiple layers of the protocol stack at the same time, such as a DNS amplification (targeting layers 3/4) coupled with an HTTP flood (targeting layer 7) is an example of multi-vector DDoS.

Mitigating a multi-vector DDoS attack requires a variety of strategies in order to counter different trajectories. Generally speaking, the more complex the attack, the more likely it is that the attack traffic will be difficult to separate from normal traffic - the goal of the attacker is to blend in as much as possible, making mitigation efforts as inefficient as possible.

Mitigation attempts that involve dropping or limiting traffic indiscriminately may throw good traffic out with the bad, and the attack may also modify and adapt to circumvent countermeasures. In order to overcome a complex attempt at disruption, a layered solution will give the greatest benefit.

2.2.1 Blackhole routing

The creation of a blackhole route and traffic redirection into it is one option available to almost all network administrators. Blackhole filtering works by sending both normal and malicious network data to a null route, or blackhole, and dropping it from the network. This is its most basic form.

If an Internet property is experiencing a DDoS attack, the property's Internet service provider (ISP) may send all the site's traffic into a blackhole as a defence. This is not an ideal solution, as it effectively gives the attacker their desired goal: it makes the network inaccessible.

2.2.2 Rate limiting

Limiting the number of requests, a server will accept over a certain time window is also a way of mitigating denial-of-service attacks.

While rate limiting is useful in slowing web scrapers from stealing content and for mitigating brute force login attempts, it alone will likely be insufficient to handle a complex DDoS attack effectively.

Nevertheless, rate limiting is a useful component in an effective DDoS mitigation strategy.

2.2.3 Web application firewall

A Web Application Firewall (WAF) is a tool that can assist in mitigating a layer 7 DDoS attack. By putting a WAF between the Internet and an origin server, the WAF may act as a reverse proxy, protecting the targeted server from certain types of malicious traffic.

By filtering requests based on a series of rules used to identify DDoS tools, layer 7 attacks can be impeded. One key value of an effective WAF is the ability to quickly implement custom rules in response to an attack.

2.2.4 Anycast network diffusion

In this mitigation strategy, the attack traffic is dispersed throughout a network of remote servers using an Anycast network until it is completely absorbed by the network. This strategy spreads the impact of the scattered attack traffic to the point where it is manageable, dispersing any disruptive capability, similar to channelling a rushing river through different smaller channels.

The reliability of an Anycast network to mitigate a DDoS attack is dependent on the size of the attack and the size and efficiency of the network. An important part of the DDoS mitigation implemented by Cloudflare is the use of an Anycast distributed network. Cloudflare has a Network Capacity network, which is an order of magnitude greater than the largest DDoS attack recorded.

CHAPTER 3

MOTIVATION

When one compares cyber security today to what it was ten years ago, the two are almost unidentifiable as the same industry. Today's landscape is utterly different. With the continuous evolvement in the technology, attackers also have evolved their techniques and strategies to intrude the systems. Taking about DDoS (Distributed Denial of Service) attack that is an evolved version of DoS (Denial of Service) attack, DDoS attacks are constantly evolving as the nature of technology used and the motivations of the attackers are changing with the regular changing devices and networking systems due to the advancement in the Internet of Things (IoT) technology. Even today, perpetrators are being caught and charged with DDoS attacks launched via botnets that cause tens of thousands of dollars of damage to the victims.

Server failures and financial losses can result from distributed denial of service attacks, which also put a lot of strain on IT workers as they work to restore resources to normal operation and give real users their regular experience. Once the DDoS attack has been successful in rendering the target site unavailable, hackers may also find it simpler to get access to the site through a back-door.

Policies and traffic flow are governed by the centralised SDN controller, which is also known as the single point of failure. Early detection of DDoS attacks in the controllers is essential since SDN controllers are especially vulnerable to DDoS attacks because of their centralised architecture.

Any effective anti-DDoS strategy should include tools that can recognise and thwart assaults before they begin. It should be able to notify users when potential threats are discovered.

This is the cause, or you could say our motive, behind our suggestion of the project idea. It is high time now and should take some effective manner using the advanced technology and presented methods in the research effort to minimise or halt these. Knowing that a DDoS attack is occurring is the first step in avoiding or stopping one. A sufficient amount of network traffic data must be gathered, then analysis must be done to determine whether the traffic is friendly or hostile in order to detect an attack. Tools that are able to identify and stop attacks before they start should be part of any anti-DDoS approach that works. It ought to have capabilities for alerting users when potential dangers are found.

CHAPTER 4

SYSTEM REQUIREMENTS ANALYSIS

4.1 Hardware Requirements: -

- CPU: 2.9GHz Intel Core i7-7500U (dual-core, 4MB cache, up to 3.5GHz)
- Linux (Ubuntu Xernial preferred)
- RAM: 8 GB DDR4
- Graphics: Intel HD Graphics 620
- Storage: 1 TB Hard Drive
- Keyboard & Mouse
- Active Internet Connection

4.2 Software Requirements: -

4.2.1 VMware VirtualBox:

Oracle VMware VirtualBox is a free, open source, cross-platform application for creating, managing and running virtual machines (VMs). Virtual machines are computers whose hardware components are emulated by the host computer.

Powerful x86 and AMD64/Intel64 virtualization software for business and home use is called VirtualBox. VirtualBox is the only business-oriented solution that is offered as free Open Source Software, and it is also a very feature-rich, high-performance product.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and Open-Solaris, OS/2, and OpenBSD.

VirtualBox has a long range of features, supported guest operating systems, and platforms, and it is constantly developed with frequent releases. Everyone is welcomed to contribute to

VirtualBox, which is a community project supported by a committed business. Oracle makes sure the finished product always satisfies industry standards for quality.

4.2.2 UBUNTU & Superuser Rights:

Ubuntu is an ancient African word meaning ‘humanity to others’. It is often described as reminding us that ‘I am what I am because of who we all are’. We bring the spirit of Ubuntu to the world of computers and software. The Ubuntu distribution represents the best of what the world’s software community has shared with the world. Linux was already established in 2004, but it was fragmented into proprietary and unsupported community editions, and free software was not a part of everyday life for most computer users. The first official Ubuntu release — Version 4.10, codenamed the ‘Warty Warthog’ — was launched in October 2004, and sparked dramatic global interest as thousands of free software enthusiasts and experts joined the Ubuntu community.

Ubuntu today has many flavours and dozens of specialised derivatives. There are also special editions for servers, OpenStack clouds, and connected devices. All editions share common infrastructure and software, making Ubuntu a unique single platform that scales from consumer electronics to the desktop and up into the cloud for enterprise computing.

Engineers from all over the world utilise the Ubuntu desktop, which is by far the most popular Linux workstation platform. The benchmark for small, transactional operating systems for extremely secure connected devices is set by Ubuntu Core. The OpenStack project's reference operating system is Ubuntu Server, which is also a very popular guest operating system on AWS, Azure, and Google Cloud. Computers from Dell, HP, Asus, Lenovo, and other international OEMs come pre-installed with Ubuntu.

The sudo command allows us to run programs with the security privileges of another user (by default, as the superuser). It prompts us for our personal password and confirms our request to execute a command by checking a file, called sudoers, which the system administrator configures. Using the sudoers file, system administrators can give certain users or groups access to some or all commands without those users having to know the root password. It also logs all commands and arguments so there is a record of who used it for what, and when. The following commands can be used for logging in as Root or superuser: - sudo -I , su , su -l , sudo -l.

4.2.3 TERMINAL

Users can communicate with shell in a GUI environment by using an application called Terminal Emulator. We used it with the VMware Virtual Box's Ubuntu operating system. Follow the appropriate steps according to our supporting operating system if we want to open the same file in a different operating system.

- ***MacOS:*** Open spotlight search (default way to do this is by hitting command and the space bar) and type in “terminal”. Select the application called terminal and press the return key. This should open up an app with a black background. If we see our username followed by a dollar sign, we’re ready to start using command line.

- ***Linux:*** We can search for Terminal by clicking the "Dash" button, putting "terminal" into the search box, and then clicking the "Dash" symbol to launch the Terminal application. We can also open Terminal by hitting [ctrl+alt+T]. Once more, this ought to launch a black-background app. We are prepared to use command line once we see our username followed by a dollar symbol.

- ***Windows:*** In Windows 10, open the start menu and go to the "Windows System" shortcuts folder. When you click the dropdown menu, a shortcut for starting the Command Prompt programme should appear. By choosing "More" from the shortcut's right-click menu, you can run the shortcut as an administrator. To access the "Windows System" folder in Windows 8, choose "All Apps" from the start screen and then scroll to the right. We can find Command Prompt there. Select "All Programs" from the start menu in Windows 7. By selecting "Accessories," the Command Prompt shortcut may be obtained. If you right-click the shortcut, you may choose to launch it as administrator.

4.2.4 Ryu Controller

The Ryu Controller is an open-source, software-defined networking (SDN) controller created to improve network agility by simplifying the management and adaptation of traffic management. Generally speaking, the SDN Controller is the brain of the SDN environment, conveying data up to the applications and business logic through northbound APIs and down to the switches and routers through southbound APIs. The Ryu Controller is used in NTT cloud data centres and is supported by NTT.

Developers can easily design new network management and control applications using the software components that the Ryu Controller offers because to its clearly defined application programme interfaces (APIs). By using a component-based approach, companies may tailor deployments to their particular requirements. Developers can quickly and easily modify existing components or add new ones to guarantee that the underlying network can support their applications' evolving requirements.

Fig. 7 – RYU SDN Controller

4.2.5 Mininet

Mininet is a *network emulator* which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.
Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

Mininet:

- Provides a simple and inexpensive **network testbed** for developing OpenFlow applications
- Enables **multiple concurrent developers** to work independently on the same topology
- Supports **system-level regression tests**, which are repeatable and easily packaged
- Enables **complex topology testing**, without the need to wire up a physical network
- Includes a **CLI** that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests

 Supports **arbitrary custom topologies**, and includes a basic set of **parametrized topologies**

- is **usable out of the box** without programming, but
- also Provides a straightforward and extensible **Python API** for network creation and experimentation

Mininet provides an easy way to get correct system *behaviour* (and, to the extent supported by our hardware, performance) and to experiment with topologies.

4.2.6 Hping3

A network tool called hping3 may issue personalised TCP/IP packets and show target responses just like the ping software does with ICMP responses. Hping3 can be used to transfer files that are enclosed in supported protocols and can manage fragmentation, arbitrary packet body, and size. We can at least accomplish the following tasks with hping3:

- Test firewall rules

- Advanced port scanning

- Test net performance using different protocols, packet size, TOS (type of service) and fragmentation.

- Path MTU discovery

- Transferring files between even really fascist firewall rules.

- Traceroute-like under different protocols.

- Firewalk-like usage.

- Remote OS fingerprinting.

- TCP/IP stack auditing.

It's also a good didactic tool to learn TCP/IP.

4.2.7 Python3

Python is a powerful, interactive, object-oriented, and interpreted scripting language. Python has been created to be very readable. It has fewer syntactical structures than other languages and typically employs English keywords rather than punctuation.

Students and working professionals who want to become exceptional software engineers, especially those in the web development field, MUST learn Python. I'll outline a few of the main benefits of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. We do not need to compile our program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − We can actually sit at a Python prompt and interact with the interpreter directly to write our programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

✓ **Characteristics of Python**

Following are important characteristics of python −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Fig. 8 – Python3

4.2.8 Xterm

The X Window System's terminal emulator is called Xterm. Ubuntu includes xterm because it automatically uses the X11 graphical server for all visuals. It offers DEC VT102/VT220 and a few functions from terminals with a higher degree of functionality, including VT320/VT420/VT520 (VTxxx). Additionally, it offers Tektronix 4014 emulation for applications that cannot directly use the window system. When a window is enlarged, xterm will use the resources provided by the underlying operating system to alert any programmes currently running in the window (for example, the SIGWINCH signal in systems derived from 4.3BSD). Xterm sets the environment variables "TERM" and "TERMCAP" after automatically searching the terminal database in this sequence for these entries.

Xterm automatically highlights the text cursor when the pointer enters the window (selected) and un-highlights it when the pointer leaves the window (unselected). If the window is the focus window, then the text cursor is highlighted no matter where the pointer is.

The xterm terminal emulator accepts the standard X Toolkit command line options as well as many application-specific options. If the option begins with a `+' instead of a `-', the option is restored to its default value.

4.2.9 Telegraf

Telegraf is capable of gathering metrics from a broad range of inputs and writing them into a broad range of outputs. It is easily extensible because it is plugin-driven for data gathering and output. It is developed in Go, thus there are no external dependencies, package management systems like npm, pip, or gem, and it can be executed on any system as a compiled and standalone binary.

Coverage: With 200+ plugins already written by subject matter experts on the data in the community, it is easy to start collecting metrics from our end-points. Even better, the ease of plugin development means we can build our own plugin to fit with our monitoring needs. We can even use Telegraf to parse the input data formats into metrics. These include: InfluxDB Line Protocol, JSON, Graphite, Value, Nagios, and Collected.

Flexible: The Telegraf plugin architecture supports our processes and does not force us to change our workflows to work with the technology. Whether we need it to sit on the edge, or in a centralized manner, it just fits with our architecture instead of the other way around. Telegraf's flexibility makes it an easy decision to implement.

4.2.10 InfluxDB

An open-source time series platform is InfluxDB. This comprises APIs for data storage and query, background processing for ETL or monitoring and alerting, user dashboards, visualising and exploring the data, and more. InfluxDB is "An open-source distributed time series database

with no external dependencies," according to its developers. Scalable data storage for metrics, events, and real-time analytics is provided by InfluxDB.

We don't need to develop any server-side code to start using it because it comes with an integrated HTTP API. Scalability, ease of installation and management, and quick data entry and exit are all features of InfluxDB. On the other hand, My-SQL is described as "the most widely used open-source database in the world." A very quick, multi-threaded, multi-user, and reliable SQL (Structured Query Language) database server is provided by the MySQL software. Both embedded in widely used software and mission-critical, high-load production systems are intended uses for MySQL Server.

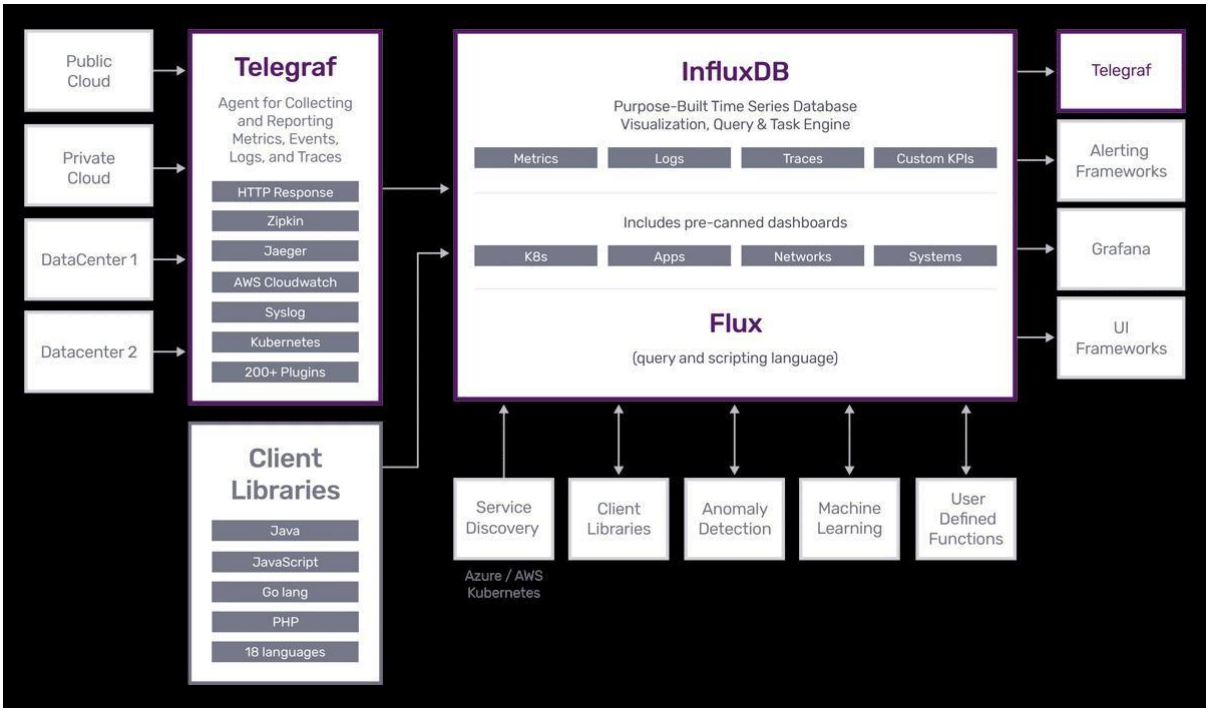


Fig. 9 – Telegraf & InfluxDB Connectivity

4.2.11 Grafana

Grafana is an open-source programme that can be used to execute data analytics, retrieve metrics that assist make sense of the vast amounts of data, and keep track of our programmes with the aid of stylish, configurable dash-boards.

Grafana connects with every possible data source, commonly referred to as databases such as *Graphite*, *Prometheus*, *Influx DB*, *Elastic-Search*, *MySQL*, *PostgreSQL* etc.

Because Grafana is an open-source programme, we may create custom plugins for integrating it with a variety of data sources. Technically referred to as time series analytics, the technology aids in the study, analysis, and monitoring of data over a period of time.

It helps us track the user behaviour, application behaviour, frequency of errors popping up in production or a pre-prod environment, type of errors popping up & the contextual scenarios by providing relative data.

4.2.12 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Installation of the library: Install
using pip:

```
pip install matplotlib
```

CHAPTER 5

METHODOLOGY & SYSTEM DESIGN

5.1 Methodology: -

The purpose of this project is to develop an artificial intelligence to classify possible DDoS attacks in an SDN network. This will be done by using data collectors such as Telegraf, Mininet to emulate the SDN network, and Influx DB and Grafana as a means to store data and visualize it respectively.

First, we'll build up our scenario with many machines, some acting as the controller and others as user test systems. In order to connect our systems and the test systems to the controller, we will use Mininet to simulate an SDN network. Additionally, users will be able to seek their interest through this simulated SDN network. Over the same network, we will conduct our authentic DDoS attack and create the necessary environment for the project.

Our scenario is running smoothly, and the attack is impacting our network as we had hoped. Or to put it another way, it's blowing stuff up. In order to detect the attack, we must collect representative data and analyse it in some way to determine whether we are actually under attack or not. Let's break this down into sections, as Jack the Ripper famously stated. We'll start by compiling the required information and sending it to a database that we can quickly query. We'll then get our SVM ready for guessing by creating training datasets for it.

The Support Vector Machine will be used to identify the DDoS attack on the SDN network (SVM). The packets were sent to the OpenFlow Switches by either attackers or regular users. The packet information, such as the information on the packet header fields, including source port, destination port, source IP address, and destination IP address, will be examined when the packet reaches the OpenFlow switch.

The information of the incoming packets will be checked against the flow entries, if a match is found then a specified action can be executed. Otherwise, the packet will be sent to the OpenDaylight controller via the southbound API using a packet_in control message.

5.2 System Design: -

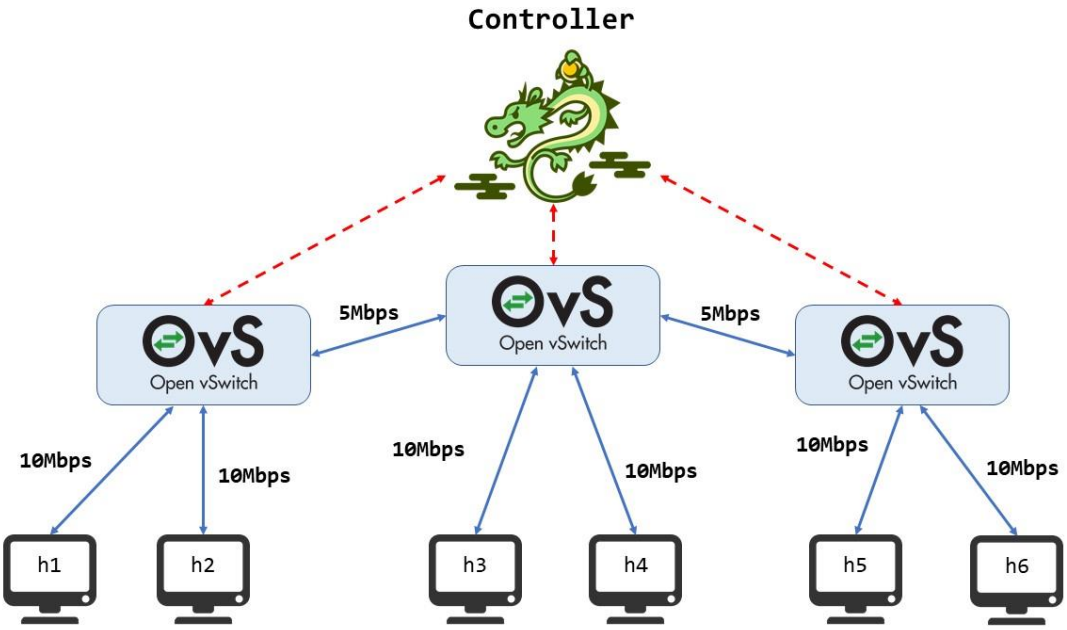


Fig. 10 – Network Architecture of Device's Connectivity

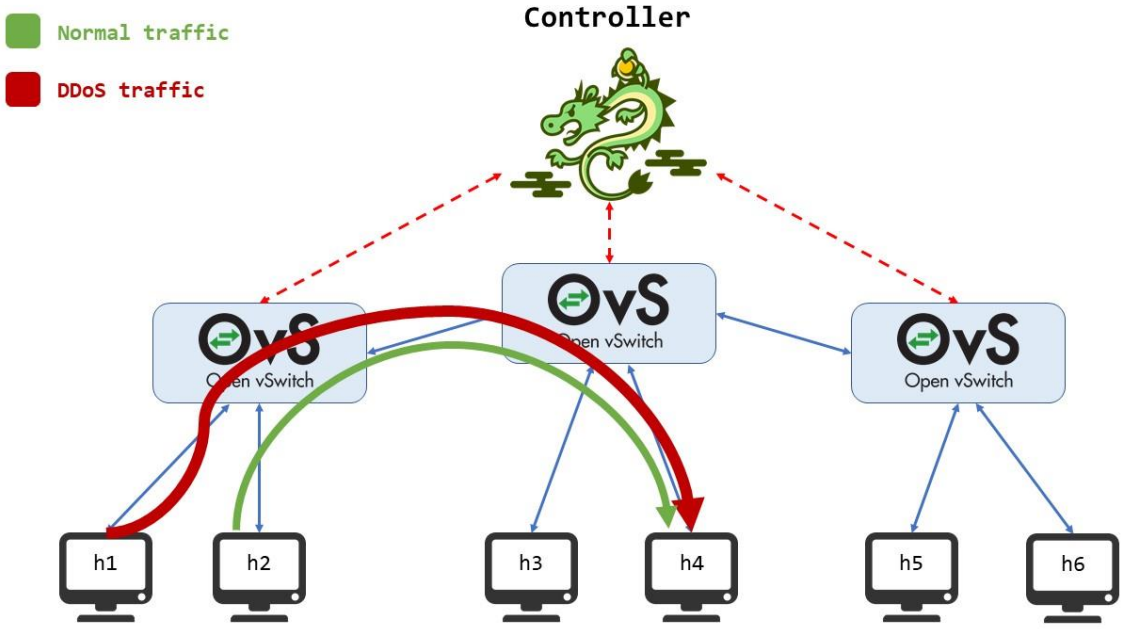


Fig. 11 – Network Architecture with Normal & DDoS Attack

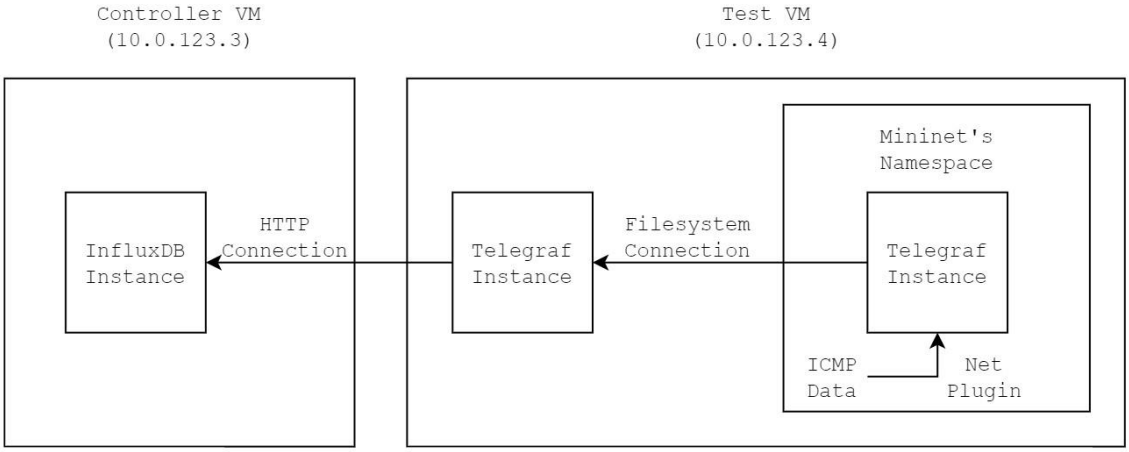


Fig. 12 – Saving the Data in InfluxDB, gathered by Telegraf

CHAPTER 6

MODULES EXPLANATION

System Installation Procedure →

Using VMware VirtualBox, we set up a virtual version of the Ubuntu operating system, through which we distribute the scenario's installation and configuration scripts to each machine. We ensure that everyone has the exact same configuration by working in a virtualized environment, making error tracing and correction much simpler. In one portion of the terminal, Ryu (the SDN controller) will be running, while in the other, a mininet-emulated network will be.

To understand all steps briefly, here's the division of the tasks in 5 Modules :-

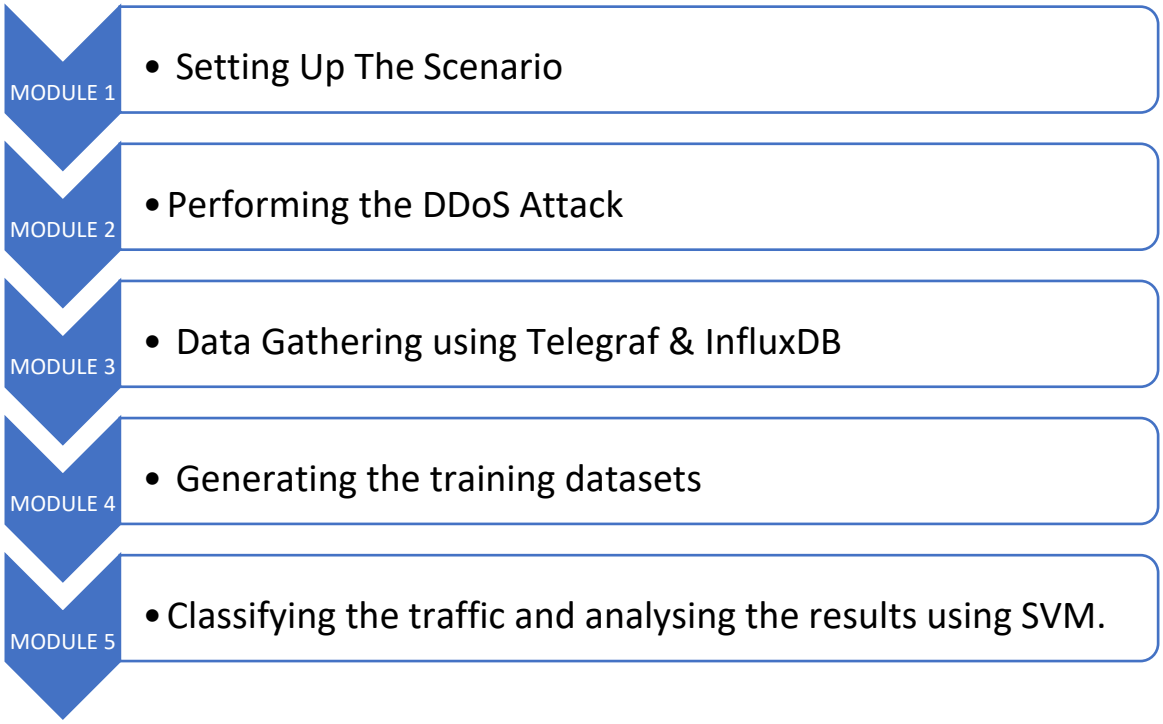
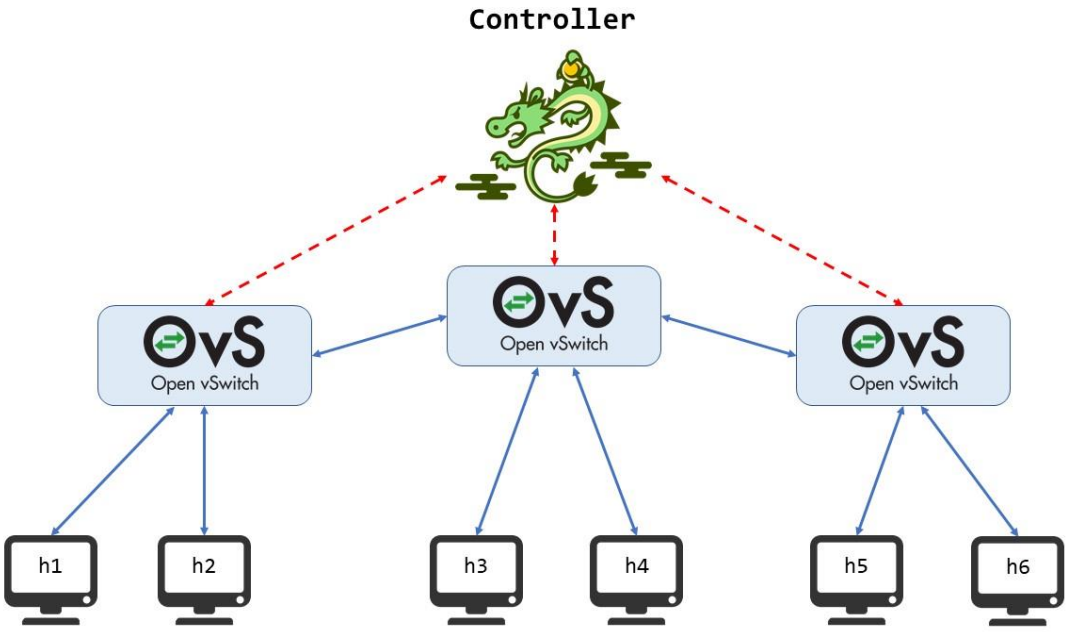


Table 1 – Module's Division

Module 1 : Setting up the Scenario →

Our network scenario is described in the following script: scenario_basic.py. Ryu controller plays the critical role as SDN controller in the whole network architecture. Mininet makes use of a Python API to give users (hosts) the ability to automate processes easily, or to develop certain modules at their convenience.



The reasoning situation we will be using is shown to us in the image above. Running the scenario needs manually logging into both VMs, or in our case, both terminals, or utilising the SSH wrapper provided by vagrant. The controller must first be powered on, therefore we run the following commands from its console. It is an application that does straightforward forwarding, which is exactly what we require:

ryu-manager ryu.app.simple_switch_13

Once the controller is up we are going to execute the network itself, to do so launch the aforementioned script from the test machine:

sudo python3 scenario_basic.py

Module 2 : Performing the DDoS Attack →

Using the hping3 tool, we will launch the DDoS attack. This adaptable programme may be set up to trace routes, send pings, run flood assaults on various network layers, and investigate a particular network. The following command will be used to manually instal it using the terminal because hping3 is typically found in the default software sources:

sudo apt install hping3

Now it's time to open different host terminals to generate the ICMP traffic. For the same, we will be using ***xterm .*** As we have previously discussed, hping3 & xterm are quite a complete tool so we will only use one of the many functionalities to keep things simple. The command we'll be using is:

hping3 -V -1 -d 1400 --faster <Dest_IP>

We are going to break down each of the options:

- *-V*: Display verbose output (i.e show more information)
- *-1*: sending ICMP packets. By default, they'll be ping requests.
- *-d 1400*: Include a fake payload. Although not necessarily necessary, doing this will help us exhaust the link's BW more quickly. In order to avoid network layer fragmentation, we selected a 1400 B payload..
- *--faster*: We would like to note out that the -flood option might have been used to start hping3 instead of the —faster option. The computer will produce as many packets as it can when —flood is used. The ICMP signals rapidly swamped the virtual network, and packets started to disappear everywhere. Although this technically qualifies as a DoS attack gone wrong, it obscures the fact that we are quicker, thus we chose to utilise —faster because the rate it offers is sufficient for our requirements.

We're going to assault hosts 1, 2, and 4 using this attack. We will attempt to ping Host4 from Host2 and launch hping3 from Host1 with the goal of reaching Host4. We will actually observe how a successful DoS attack causes this "normal" ping to fail. The scenario is shown in the illustration in the section on system design (fig. 12).

It's time to connect both xterm host terminals to ICMP sources. Wireshark will also be launched on Host4 so we can take a better look at what's happening. The second command's ampersand (&) is to be noted. The Wireshark process will be disconnected from the terminal so that we may continue issuing commands as usual. This requires us to move quickly:

mininet> xterm h1 h2 mininet> h4 wireshark &


Module 3 : Data gathering using Telegraf & InfluxDB →

Our scenario is running smoothly, and the attack is impacting our network as we had hoped. Or to put it another way, it's blowing stuff up. In order to detect the attack, we must collect representative data and analyse it in some way to determine whether we are actually under attack or not. Let's divide things up into sections, as Jack the Ripper famously stated. We'll start by compiling the required information and sending it to a database that we can quickly query. We'll then get our SVM ready for guessing by creating training datasets for it. Let's start!

Now in this module, we will be using two tools namely Telegraf and InfluxDB for the ICMP data gathering and storing it in the database. The first one is a metrics agent tasked with gathering information on the host it is currently executing on. Since it is purely plugin driven, setting it up is really simple. The latter is a DBMS (Data Base Management System) with a time series-specific design, which is just what we want! Given that influxdb has native support in one of telegraf's plugins, connecting the two is simple.

We can take advantage of what some may believe to be a weakness in mininet's architecture since the host computer has direct communication with the VM hosting the controller. On Host 4 of the mininet, we'll run an instance of Telegraf whose input plugin will collect ICMP data and whose output will be a file in the home directory of the VM. We'll be running a second telegraf instance in the host VM whose input will be the file containing Host 4's output and whose output will be the Influx DB hosted in the controller VM. This architecture employs a second telegraf instance as a simple proxy between one of the internal hosts in the mininet and the controller VM, both of which are located in different networks, and makes use of the shared filesystem.

We are trying to overwhelm Host 4 with a bunch (a **VERY BIG** bunch) of ICMP Echo Requests (that is fancy for pings). We discovered the net plugin, which is capable of giving ICMP data right out of the box, by looking through the list of input plugins for Telegraf.

The file output plugin will allow us to send the output to an ordinary file rather than straight to an influxdb instance. This brings us right to the second telegraf instance's settings.

The tail input plugin will be used in this second procedure. This command, like Linux's tail, will read a file indefinitely in order to use it as an input data stream. We opted to inform Telegraf to read the file when changes were made rather than querying the file continually. As a result, system resources are used more effectively overall. Now we'll be using the reliable influxdb output plugin. In order to ensure that everything is properly connected, we will point it toward the influxdb instance running on the controller VM.

Now that we have the information we need, we can begin searching our database and using it.


Module 4 : Generating the training datasets →

Reading and arranging the desired data so that the SVM "likes it" is all that is required to use the SVM once it has been put in the database. We choose to create a straightforward Python script (data gathering.py) that reads the data using the influxdb Python API and creates a CSV (Comma Separated Values) file that can be read by the script that implements the SVM later.

The defining quality of training data is meaningfulness. The SVM's predictions will only be as good as the training it received so we need to provide insightful data if we are to get any consistent results.

We proceeded to imitate typical traffic by pinging the target host at a rate of around 1 ICMP message per second in order to obtain the necessary data samples. Once we had about 100 samples in the database, we continued to assault the target. Reading the database and writing the read data to a text file with the.csv extension is all that is necessary to generate the database.

Module 5 : Classifying the traffic and analysing the results using SVM →

Apart from the scenario's setup the most important program we wrote is the traffic classifier without a doubt. The file defines the gar_py() class which includes an SVM instance, the query used for getting data and many other configuration parameters as its attributes.

The function Object () {[native code]} of the class will only be capable of initialising its properties and instructing the SVM using the training data that we have previously created. The most important thing to keep in mind is that we must adhere to the input specifications required by the SVM itself.

Once it's trained we just need to call the class's work_time() method which will enter an infinite loop whose operation can be summarised into these points:

1. Read the last 3 entries in the DB.
2. Verify these entries are indeed new.
3. Update the parameters we're going to use for the prediction.
4. Order the SVM to predict whether the new data represents an attack or not.
5. Write an entry to the appropriate DB signalling whether or not we're under attack.
6. Wait 5 seconds to read new data. New data is sent to the DB every 4 seconds so reading in-sanely fast is just throwing resources out the window.

Additionally, we drew the classification we were performing using matplotlib. The red dots in the outputs, as seen, represent data that has been identified as anomalous or DDoS activity.

CHAPTER 7

SYSTEMS DEVELOPMENT (CODING)

7.1 Module 1 →

(Scenario_basic.py)

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController,
Ryu from mininet.node import CPULimitedHost, Host, Node from
mininet.node import OVSKernelSwitch, UserSwitch from mininet.node
import IVSSwitch from mininet.cli import CLI from mininet.log import
setLogLevel, info from mininet.link import TCLink, Intf from subprocess
import call

def scenario_basic():
    net = Mininet(  topo = None,
                    build       =       False,
host        =       CPULimitedHost,
link = TCLink,                  ipBase =
'10.0.0.0/8')

    info('***  Add  Controller  (Ryu)  ***\n')
c0   =   net.addController(   name   =   'c0',
controller          =          RemoteController,
```

```
ip = '10.0.2.15',                    protocol =
'tcp',                    port = 6633)

    info('*** Add three switchs ***\n')     s1 =
net.addSwitch('s1', cls = OVSKernelSwitch)     s2
= net.addSwitch('s2', cls = OVSKernelSwitch)
s3 = net.addSwitch('s3', cls = OVSKernelSwitch)

    info('*** Add Host ***\n')
     h1 = net.addHost('h1', cls = Host, ip = '10.0.0.1', defaultRoute = None)
    h2 = net.addHost('h2', cls = Host, ip = '10.0.0.2', defaultRoute = None)
    h3 = net.addHost('h3', cls = Host, ip = '10.0.0.3', defaultRoute = None)
    h4 = net.addHost('h4', cls = Host, ip = '10.0.0.4', defaultRoute = None)
    h5 = net.addHost('h5', cls = Host, ip = '10.0.0.5', defaultRoute = None)
    h6 = net.addHost('h6', cls = Host, ip = '10.0.0.6', defaultRoute = None)

    info('*** Add links ***\n')
net.addLink(s1, h1, bw = 10)
net.addLink(s1, h2, bw = 10)
    net.addLink(s1, s2, bw = 5, max_queue_size =
500)     net.addLink(s3, s2, bw = 5, max_queue_size
= 500)     net.addLink(s2, h3, bw = 10)
net.addLink(s2, h4, bw = 10)     net.addLink(s3, h5,
bw = 10)     net.addLink(s3, h6, bw = 10)

    info('\n*** Build it ***\n')
    net.build()

    info('***  Start  the  controller  ***\n')
for controller in net.controllers:
```

```
        controller.start()

    info('*** Set controllers ***\n')
    # Notice how [c0] returns a list of controller objects whose only element
    # is c0. The start() method will traverse the input parameter with a for
# loop so we need to pass a list as an argument...
    net.get('s2').start([c0])
net.get('s3').start([c0])
net.get('s1').start([c0])

    info('\n*** Start Telegraf ***\n')
    # Notice that s1, s2 and s3 are in the same Network Namespace.
    # net.get('s1').cmd('telegraf --config conf/telegraf.conf &')
    # We need to run telegraf in the hosts because the have ICMP visibility
# The switches can only see up to layer 2...
    net.get('h4').cmd('telegraf --config conf/telegraf_mn_host.conf &')

    info('*** RUN Mininet\'s CLI ***\n')
    CLI(net)

    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
scenario_basic()
```

7.2 Module 2 →

(DDoS.py)

```
import os, sys, time, datetime
```

```
# Global parameters
ERROR_BAD_ARGV_FROM_USER = '[DDoS] Error, incorrect arguments: '
INFO_INIT_1 = '[DDoS] Starting the attack on the given IP '
INFO_INIT_2 = '(Press CTRL+C to stop me)...'
INFO_STATS = '[DDoS] Quitting, showing stats:'
ATTACK_FIN = '[DDoS] Completed the attack  >:D'
PKTS_CADENCE = 100
PKTS_LEN = 1442
DATA_LEN = 1000000
DATA_STR = 'MB'
INIT_WAIT = 4



# Get the current time  def
get_str_time():
        return ('[' + (datetime.datetime.now()).strftime('%H:%M:%S') + ']')

# Get the time difference based on global variables def
diff():
        return (datetime.datetime.now() - time_init)

# Prepare the stats def
stats():
        return ('[+] Time Elapsed: ' + str(diff()) + '\n' + '[+] Data sent: '

                + str(diff().total_seconds() * PKTS_CADENCE * PKTS_LEN / DATA_LEN)
+ ' ' + DATA_STR + '\n')

if __name__ == "__main__":
```

```
        # Check the passed arguments
        if len(sys.argv) != 2:
 print(get_str_time() + ERROR_BAD_ARGV_FROM_USER + '\n\n\t Usage: python3 ' +
sys.argv[0] + ' <Destination IP>')
                exit(-1)

 # Initialize status variables   time_init =
datetime.datetime.now()

        # Tell the user how he/she can stop the attack
        print(INFO_INIT_1 + sys.argv[1] + ' ' + INFO_INIT_2)
        os.system('sleep ' + str(INIT_WAIT))

        # Run hping3!
        os.system('hping3 -V -1 -d 1400 --faster ' + sys.argv[1])

        # Show the stats
        print('\n\n'+get_str_time()    +    INFO_STATS    +    '\n\n'    +    stats())
        print(get_str_time() + ATTACK_FIN)
```

7.3 Module 3 →

To update the package list and **install Telegraf as a service** in the root terminal with superuser privileges, following commands help us :

$ sudo apt-get update $ sudo apt-get install telegraf

Right now, Telegraf should run as a **service** on our server. To verify it, we ran the following command:

$ sudo systemctl status telegraf

After installing InfluxDB service successfully, we'll be creating an admin account & a database naming ***telegraf*** for storing the ICMP traffic data in it.

$ influx

> CREATE USER admin WITH PASSWORD 'password' WITH ALL PRIVILEGES > SHOW USERS

> CREATE USER telegraf WITH PASSWORD 'password' WITH ALL PRIVILEGES > SHOW USERS

Then, Restart the Telegraf service, as well as the InfluxDB service using the following commands :

$ sudo systemctl restart influxdb $ sudo systemctl restart telegraf

Again, checking that we are not getting any errors when restarting the service using the command below :

$ sudo journalctl -f -u telegraf.service $ sudo journalctl -f -u influxdb.service

Now to check and open the data stored in the ***telegraf*** database, we used the following commands:

$ influx

*> USE telegraf > SELECT * FROM cpu*

7.4 Module 4 →

(Data_gathering.py)

```
import influxdb, sys


QUERY = """SELECT DERIVATIVE(icmp_inechos) AS d_ping FROM net ORDER BY time DESC LIMIT 100"""
```

```
n_samples, mean = 0, 0


if __name__ == "__main__":
if len(sys.argv) == 2:
        db = influxdb.InfluxDBClient('10.0.2.15', 8086, ' ', ' ', 'telegraf')
measurement_class = sys.argv[1]     elif len(sys.argv) == 3:
        db = influxdb.InfluxDBClient(sys.argv[1], 8086, ' ', ' ', 'telegraf')
measurement_class = sys.argv[1]     elif len(sys.argv) == 4:
        db = influxdb.InfluxDBClient(sys.argv[1], int(sys.argv[2]), ' ', ' ', 'telegraf')
measurement_class = sys.argv[1]     elif len(sys.argv) == 5:
        db = influxdb.InfluxDBClient(sys.argv[1], int(sys.argv[2]), ' ', ' ', sys.argv[3])
measurement_class = sys.argv[1]     else:
        print("Usage: python3 " + sys.argv[0] + " 0 | 1 [InfluxDB_IP] [InfluxDB_port]
[DB_name]")
exit(-1)


    out_file = open("ICMP_data_class_{}.csv".format(measurement_class), "w+")
    for measurement in db.query(QUERY).get_points(measurement = 'net'):
        curr_derivative          =          measurement["d_ping"]
n_samples += 1
        delta_mean  =  (curr_derivative  -  mean)  /  n_samples
mean += delta_mean
        out_file.write("{}, {}, {}\n".format(curr_derivative, mean, measurement_class))


    out_file.close()
    print("Finished generating a class {} training dataset!".format(measurement_class))    exit(0)
```

7.5 Module 5 →

(Traffic_classifier.py)

```
import influxdb, datetime, time, os, signal #

Install me with: pip3 install sklearn

# We also need numpy: pip3 install numpy

# It should have been installed as a dependency nonetheless!

from sklearn import svm


class gar_py:

 def __init__(self, db_host = 'localhost', port = 8086, db = 'h4_net_stats', kern_type = 'linear',
dbg = False):

                # Constructor n.n

        self.debug = dbg

  self.n_samples, self.mean = 0, 0    self.host =

db_host   self.port = port   self.dbname = db

  self.client = influxdb.InfluxDBClient(self.host, self.port, 'root', 'root', self.dbname)

                self.svm_inst  =  svm.SVC(kernel  =  kern_type)

                if os.uname()[1] == "pop-os":

                        self.training_files = ["ICMP_data_class_0.csv",
"ICMP_data_class_1.csv"]

        else:

                        self.training_files =
["/home/sdn/Desktop/project/training_datasets/ICMP_data_class_0.csv",
"/home/sdn/Desktop/project/training_datasets/ICMP_data_class_1.csv"]


                # We need to use these triple quotes so that we can use a tag within the query!
# self.query = """select bytes_sent from net where interface = 's1-eth1' order
by time desc limit 1;"""

                 self.query = """SELECT DERIVATIVE(icmp_inechos) AS d_ping FROM net
ORDER BY time DESC LIMIT 3"""

        self.train_svm()
```

```
        def train_svm(self):
 # Dump data from a file and feed the SVM   features, labels = [],
[]
 # Files are opened for reading and in text mode by default. Specifying it doesn't hurt though
                # Our file format is: ^Sample_value, Current_mean, Class$
                 # Where Class is either 0 (Not an attack) or 1 (Houston, we've got a situation!)
                # The ^ and $ characters symbolize the beginning and end of a line like in
RegExps                 for         fname         in
self.training_files:
  meal = open(fname, "rt")   for line in meal:
                                  data_list = line.rsplit(", ")
                                  # We need to work with numbers. Casting time!
                                  for i in range(len(data_list)):
                                  if i < 2:
                                                  data_list[i]     =     float(data_list[i])
                                          else:
    data_list[i]   =   int(data_list[i])            features.append(data_list[:2])
labels.append(data_list[2])   meal.close()   if self.debug:
                          print("Features first and last entries:\n\t", end = "")
        print(features[:1] + features[199:])

                          print("Labels first and last entries:\n\t", end = "")
                          print(labels[:1] + labels[199:])
        self.svm_inst.fit(features, labels)


 def                    work_time(self):
last_entry_time = "0"   while True:
  # We have to reverse the elements due to how the query returns the result. We need to
  # reverse() a list so that's why we need to cast the generator returned by get_points()
  for new_entry in reversed(list(self.get_data(self.query).get_points(measurement = 'net'))):
```


```
                                # Strings have been overloaded to allow this type comparison!
                                if new_entry['time'] > last_entry_time:
                        last_entry_time = new_entry['time']
                self.n_samples += 1
                                                # Take a look at the documentation for this quantity!
                                                delta_mean = (new_entry['d_ping'] - self.mean) /
(self.n_samples)
    self.mean += delta_mean      if self.debug:
                                                        print("\n** New entry
**\n\tDelta_ICMP_inechos: " + str(new_entry['d_ping']))
                                                        print("\tCurrent mean: " + str(self.mean))

                self.ring_the_alarm(self.under_attack(new_entry['d_ping'], self.mean))
                time.sleep(5)

 def  under_attack(self,  ping_sample,  current_mean):       if
self.debug:
                                print("\tCurrent prediction: " + str(self.svm_inst.predict([[ping_sample,
current_mean]])[0]))          if          self.svm_inst.predict([[ping_sample,
current_mean]])[0] == 1:
                                return True
                else:
                                return False

                def get_data(self, petition):
 #  Get  data  from  InfluxDB          return
self.client.query(petition)

                def ring_the_alarm(self, should_i_ring):
```

```
                # Ring the alarm by writing to InfluxDB
        data_json = self.get_ddos_json_body(should_i_ring)
        if self.client.write_points(data_json) and self.debug:
                print('\t[OK] Data was correctly sent to InfluxDB :)\n')


        # --- Aux Methods --- #
        def get_ddos_json_body(self, boolean):
                return [{'measurement': 'ddos', 'fields': {'attack_flag': int(boolean)}, 'tags': {'host': 'Ryu_Controller'}, 'time': self.get_datetime()}]


        def get_datetime(self):
                return (datetime.datetime.now()).strftime('%Y-%m-%dT%H:%M:%SZ')


def ctrl_c_handler(s, f):
        print("\b\bShutting down MR. SVM... Bye!")
        exit(0)


if __name__ == "__main__":
        signal.signal(signal.SIGINT, ctrl_c_handler)
        ai_bot = gar_py(db_host = '10.0.2.15', dbg = True)
        # Load up the AI and start rocking!
        ai_bot.work_time()
```

7.6 Module 5 → (Graph.py)

```
   from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
   x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
```

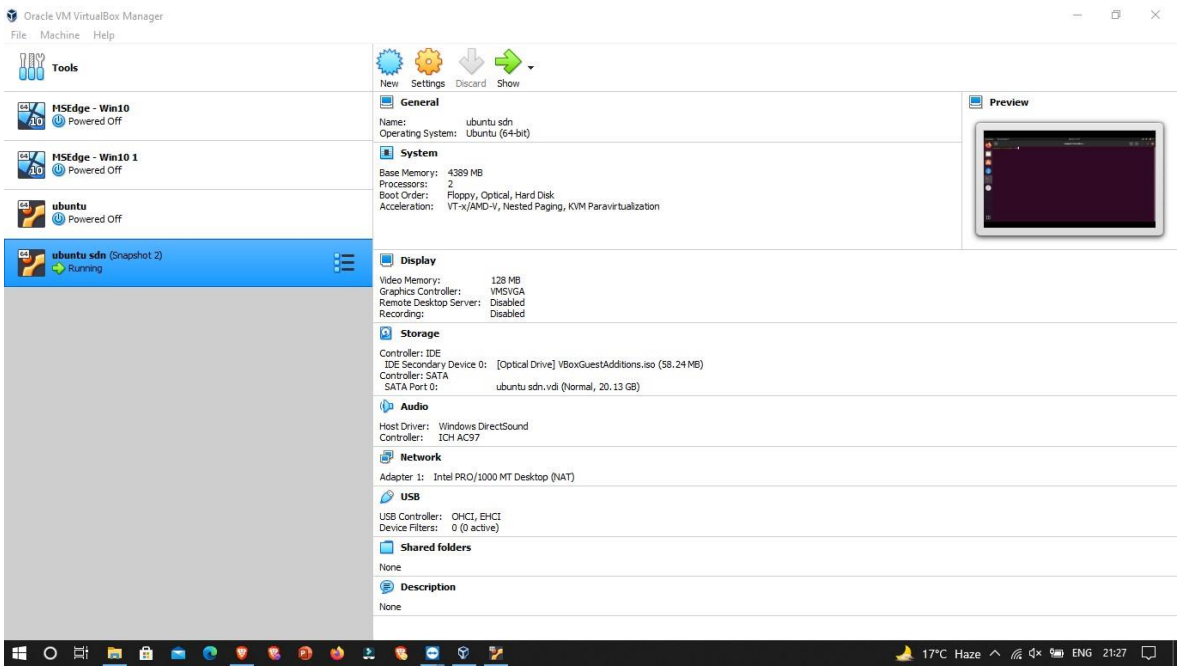
```
    nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
    alpha = 0.75, cmap = ListedColormap(('red',
'green')))      mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())      for i, j in
enumerate(nm.unique(y_set)):
        mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j,
1],              c = ListedColormap(('red', 'green'))(i), label
= j)      mtp.title('SVM classifier (Training set)')
mtp.xlabel('Age')      mtp.ylabel('Estimated Salary')
mtp.legend()      mtp.show()
```

CHAPTER 8

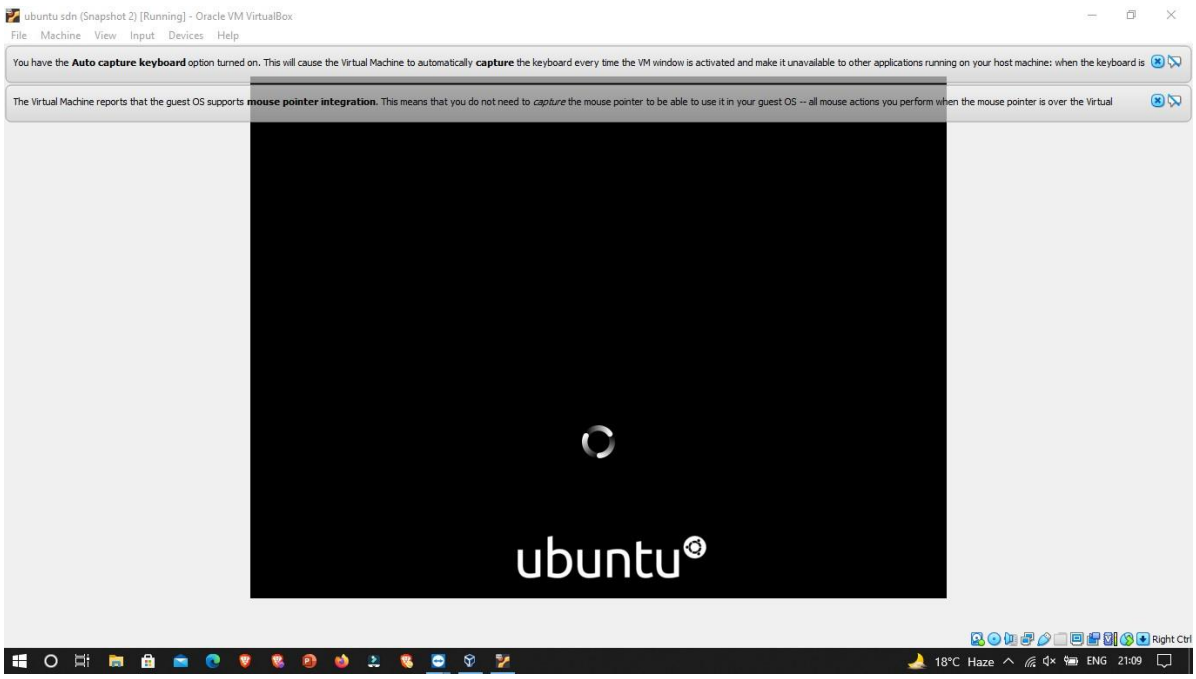
SYSTEMS IMPLEMENTATION (SCREENSHOTS)

8.1 MODULE 1 →

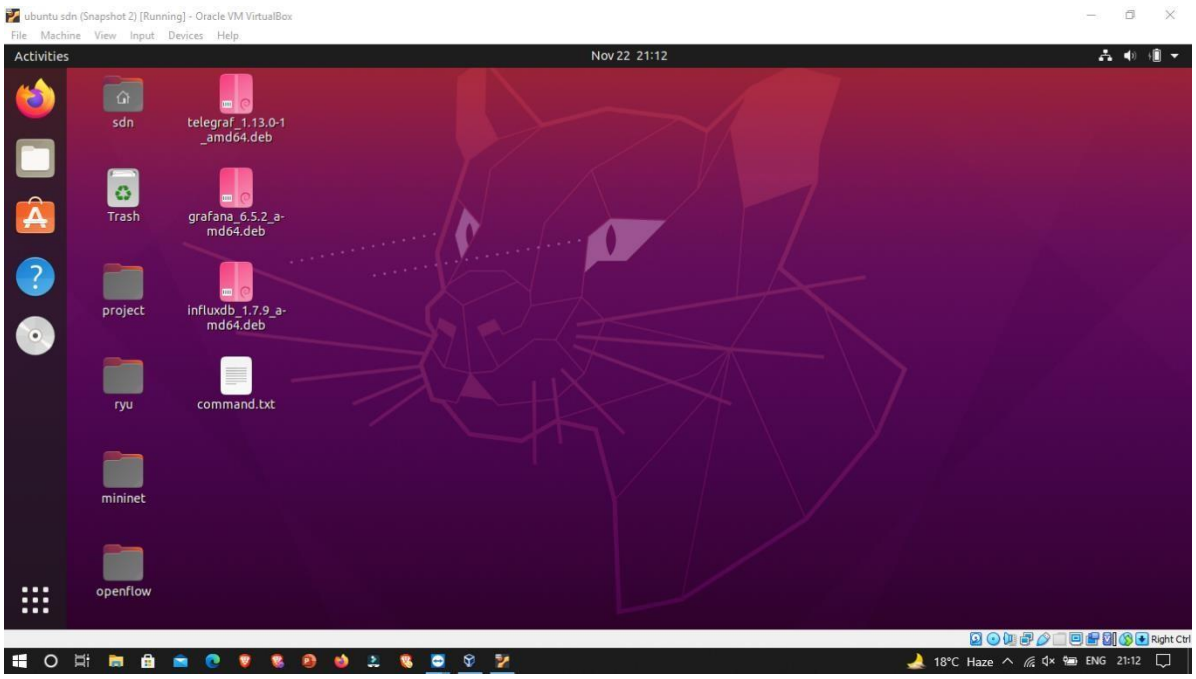
(Setting up the scenario)



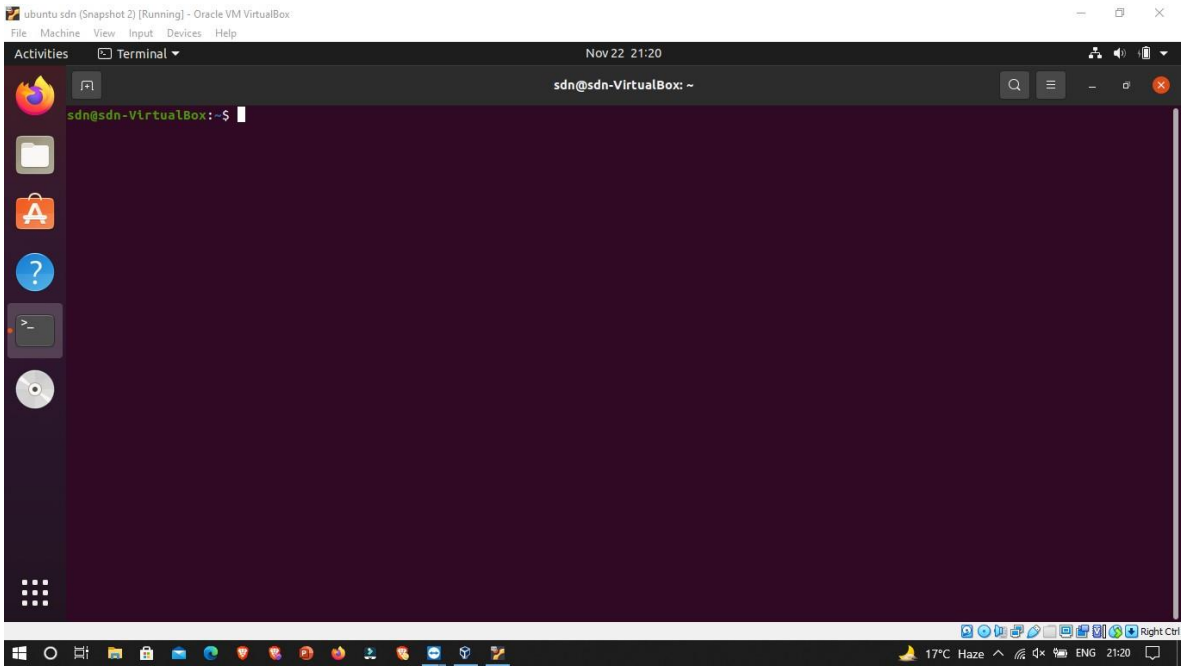
(Opened the VMware VirtualBox)



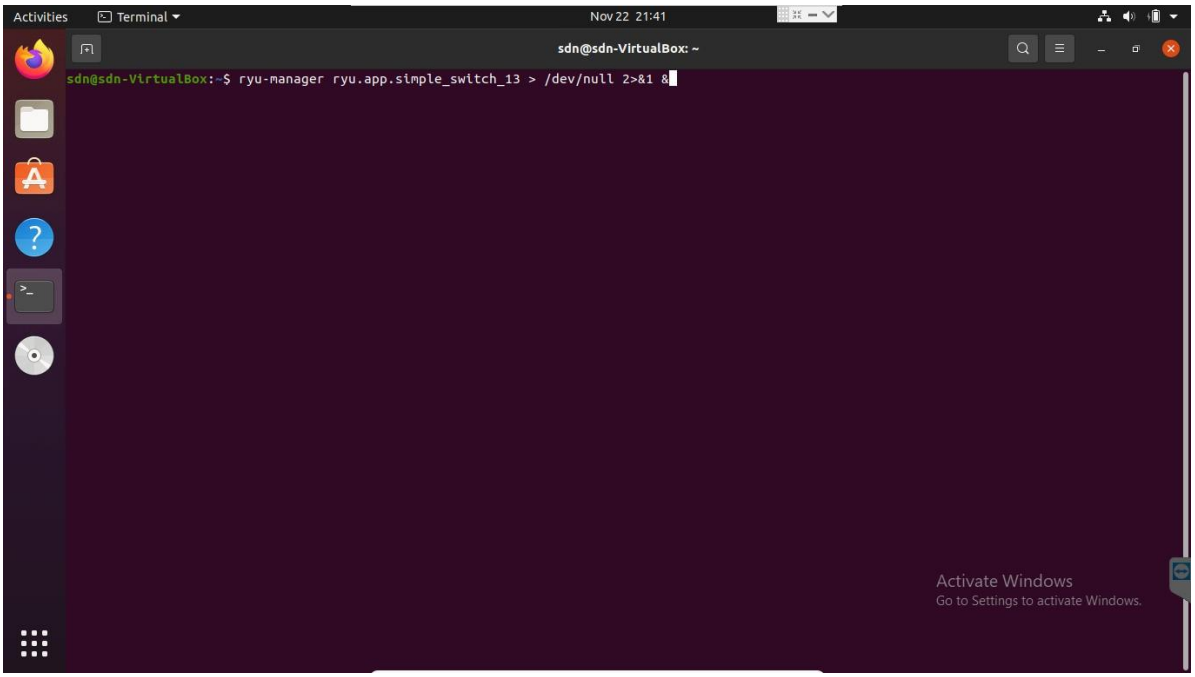
(Starting the Ubuntu on the VMware VirtualBox)



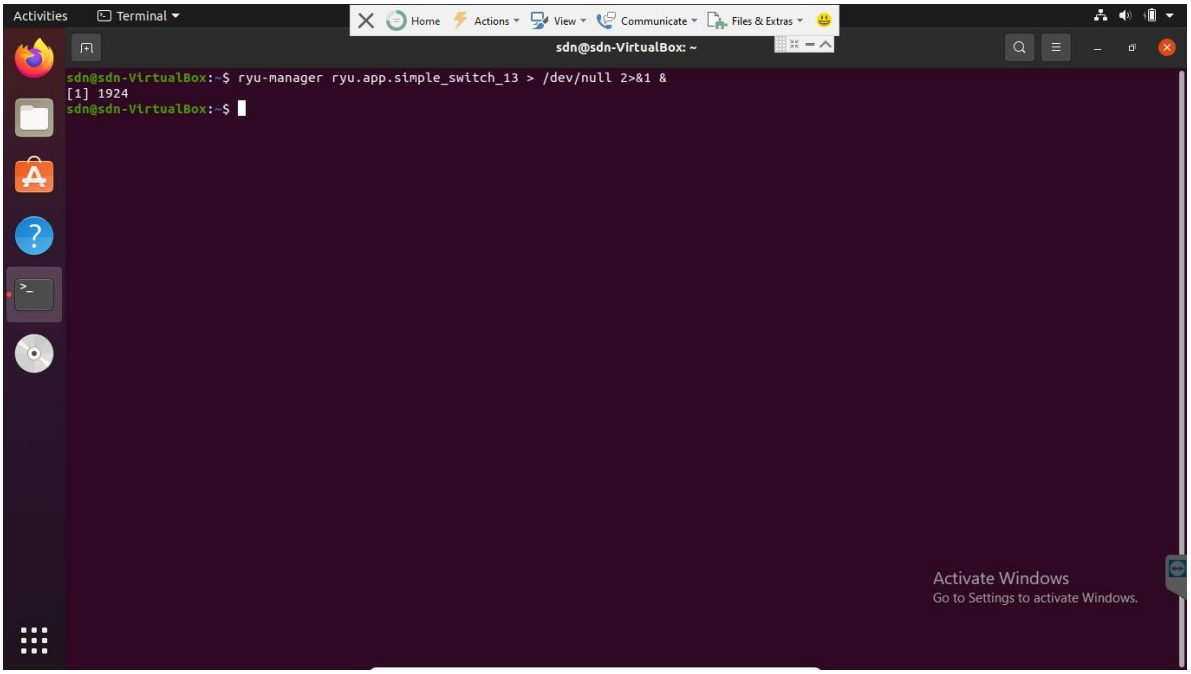
(Home screen of the Ubuntu OS)



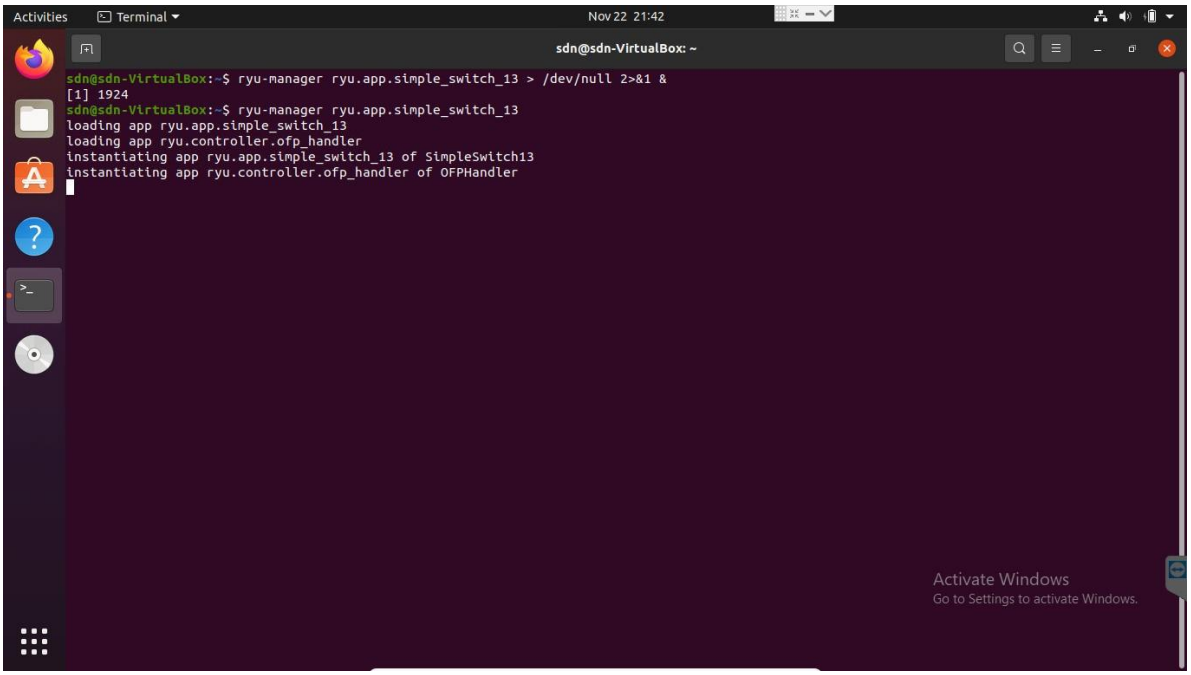
(Opened a Terminal)



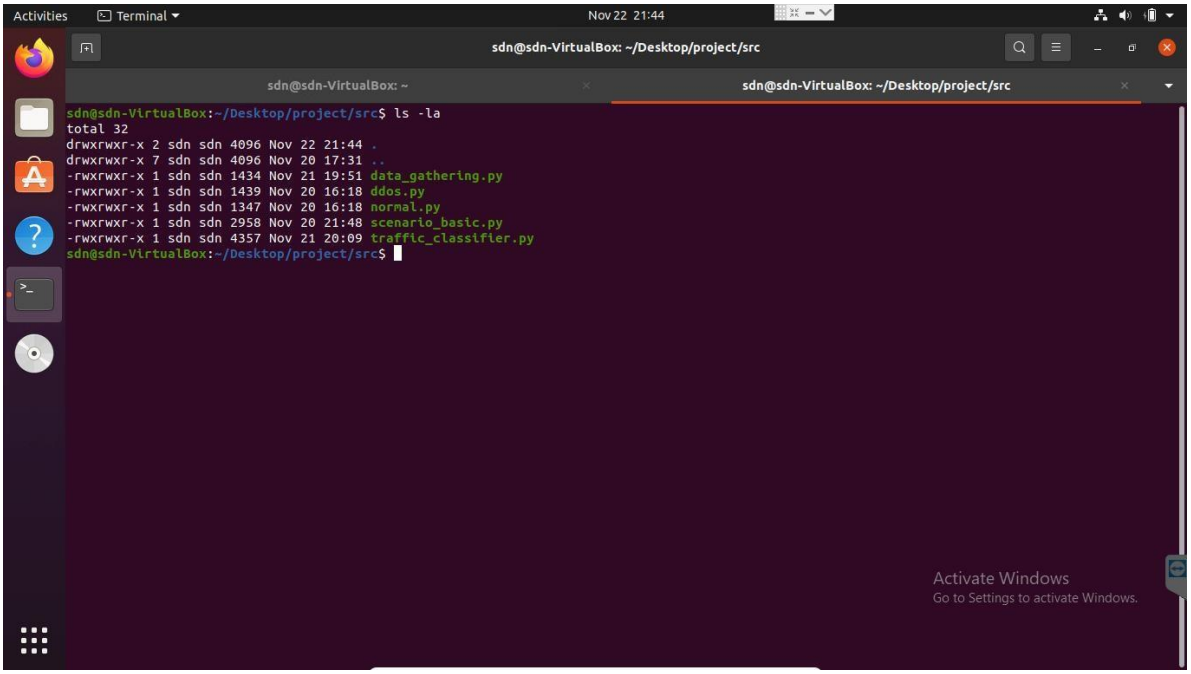
(Creating RYU Controller)



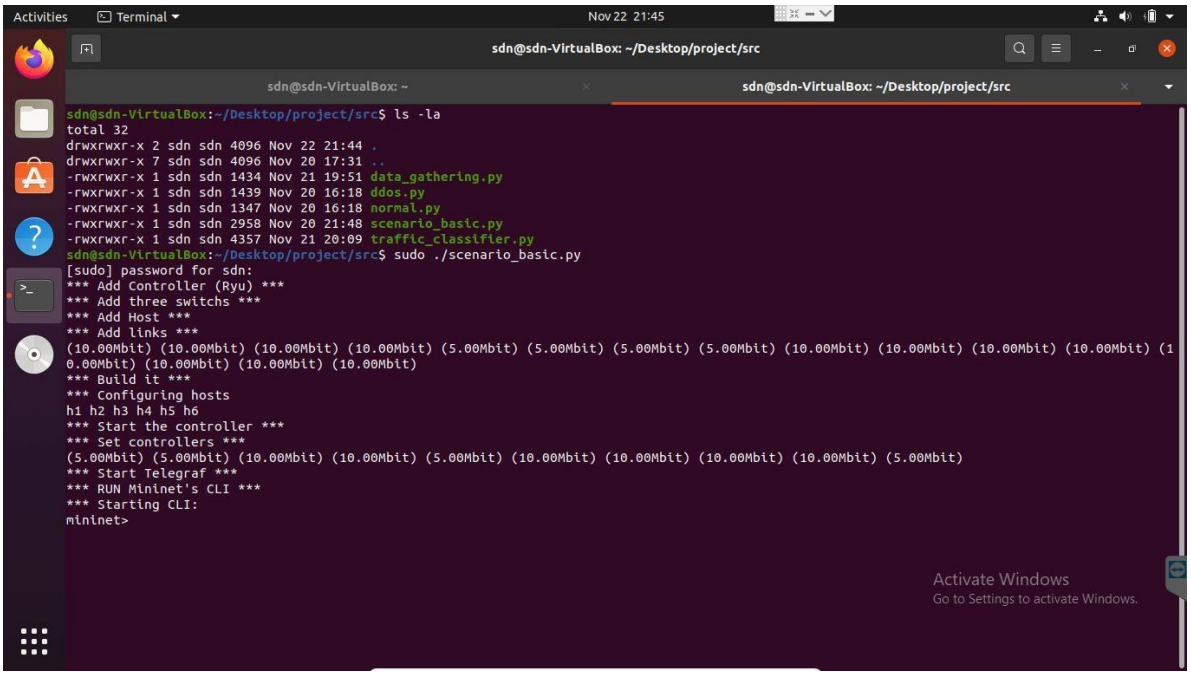
(RYU Controller Created)



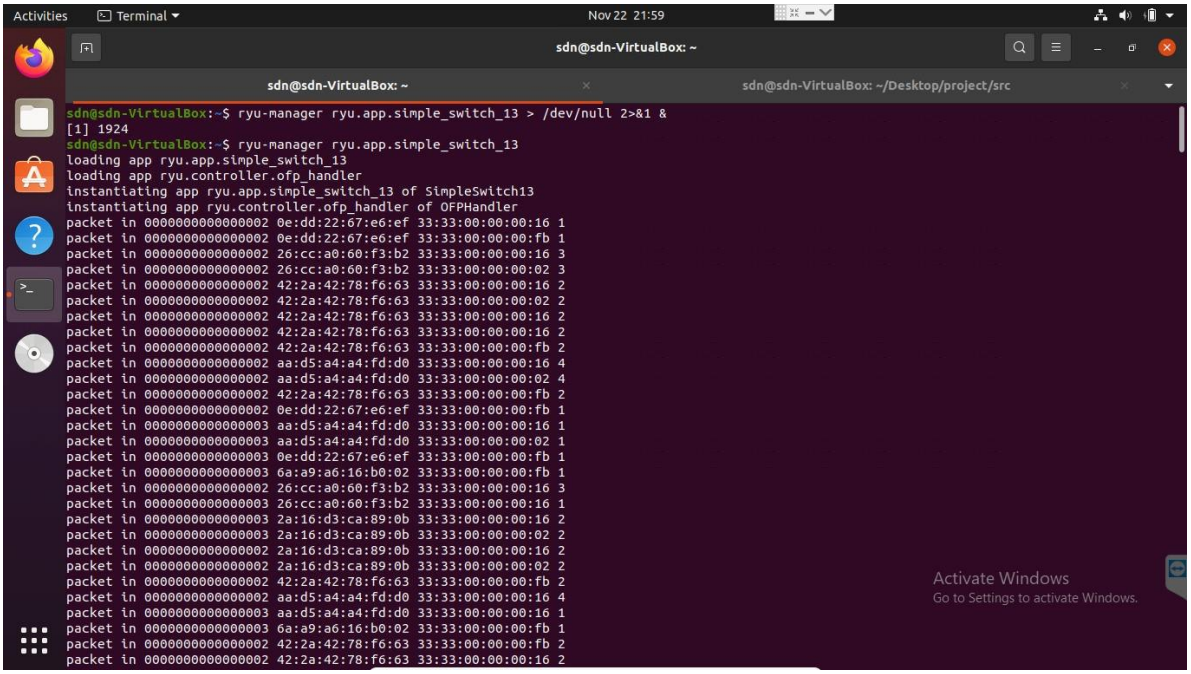
(Starting the RYU Controller)



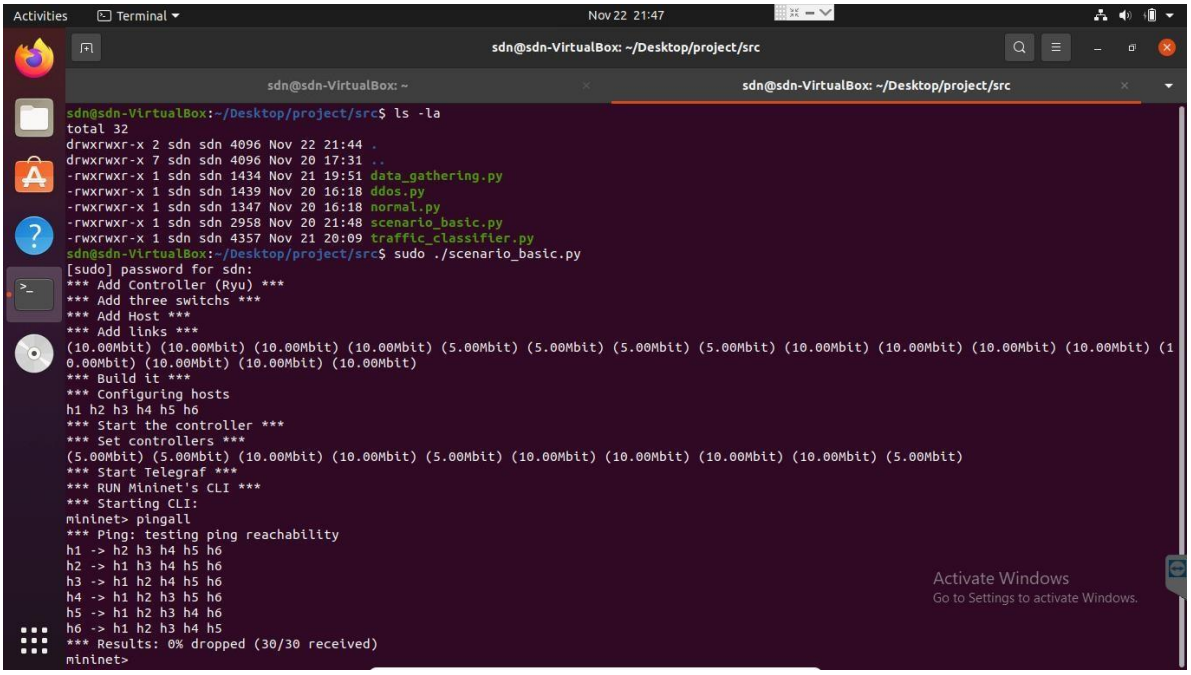
(Saved all python scripts for further implementation)



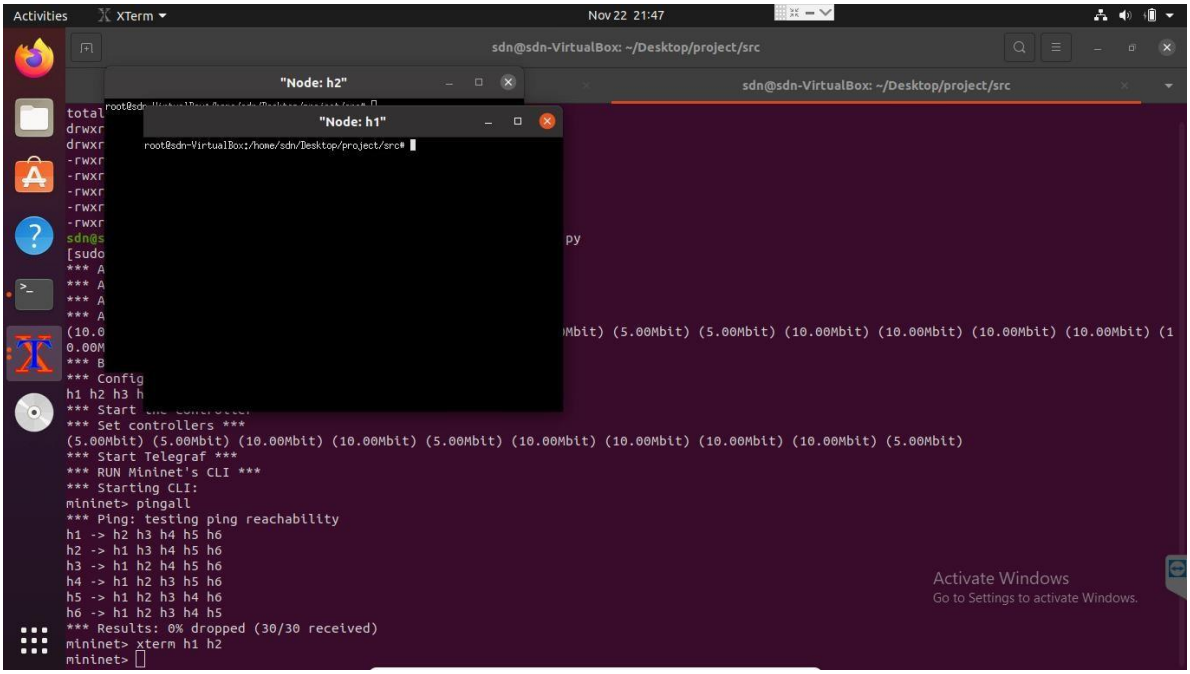
(Setting up the scenario)



(RYU controller after our scenario script runs)

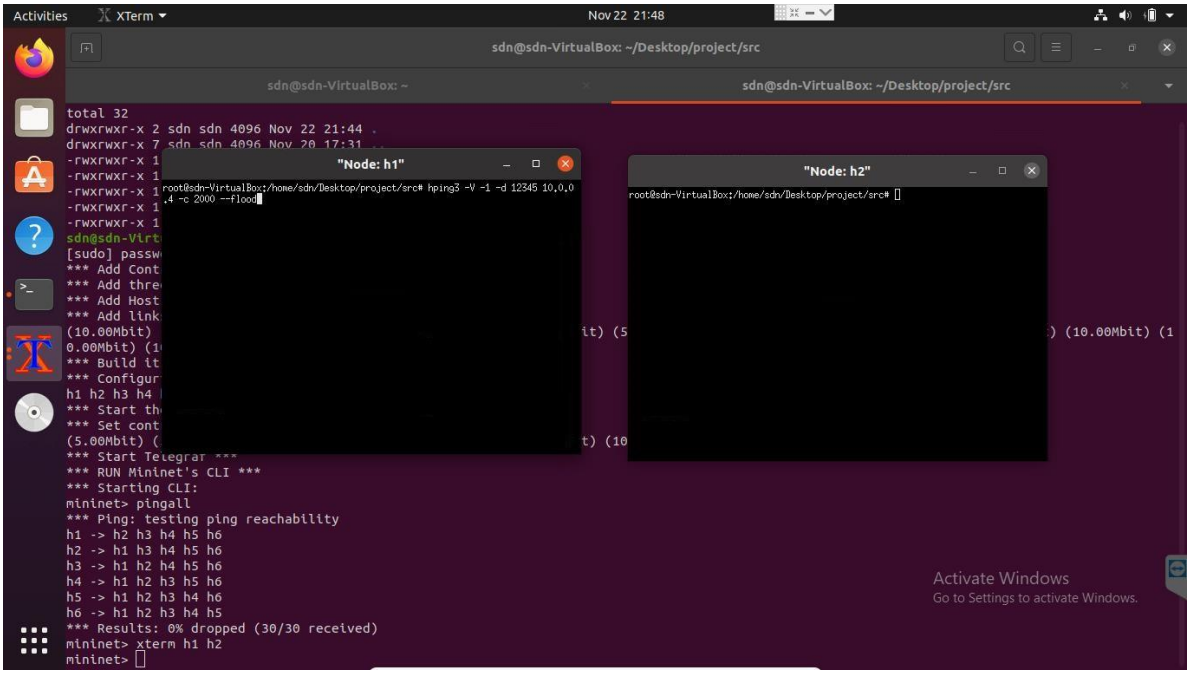


(Our Mininet is ready)

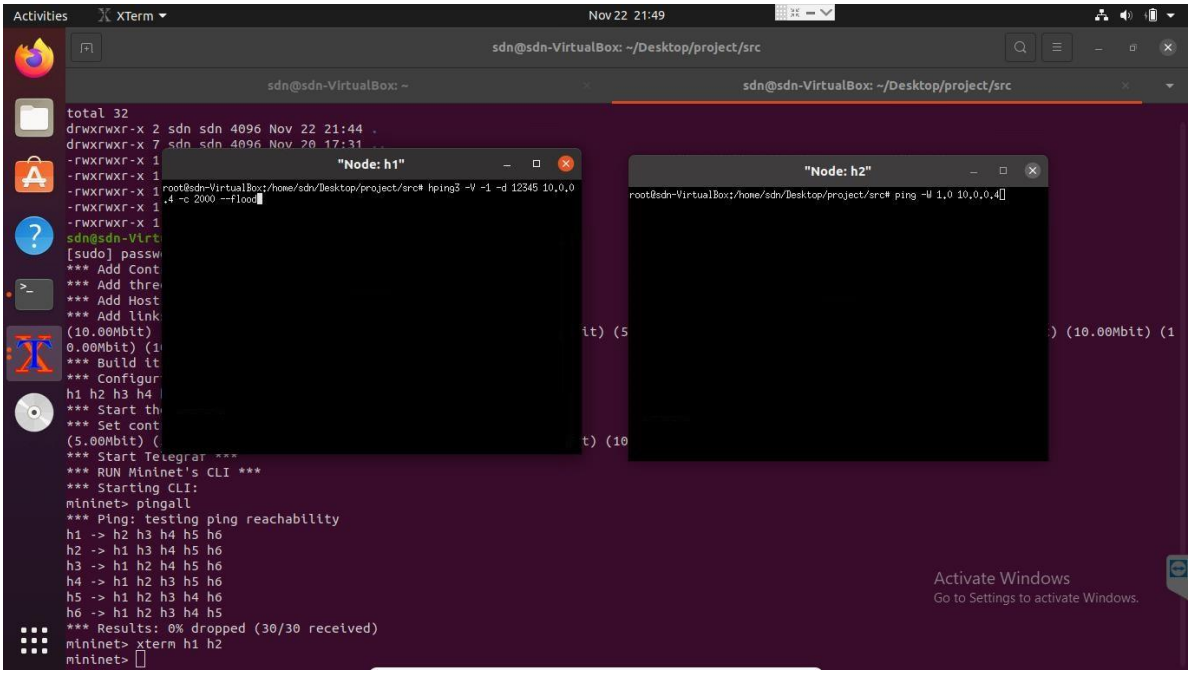


(Xterm is also working fine)

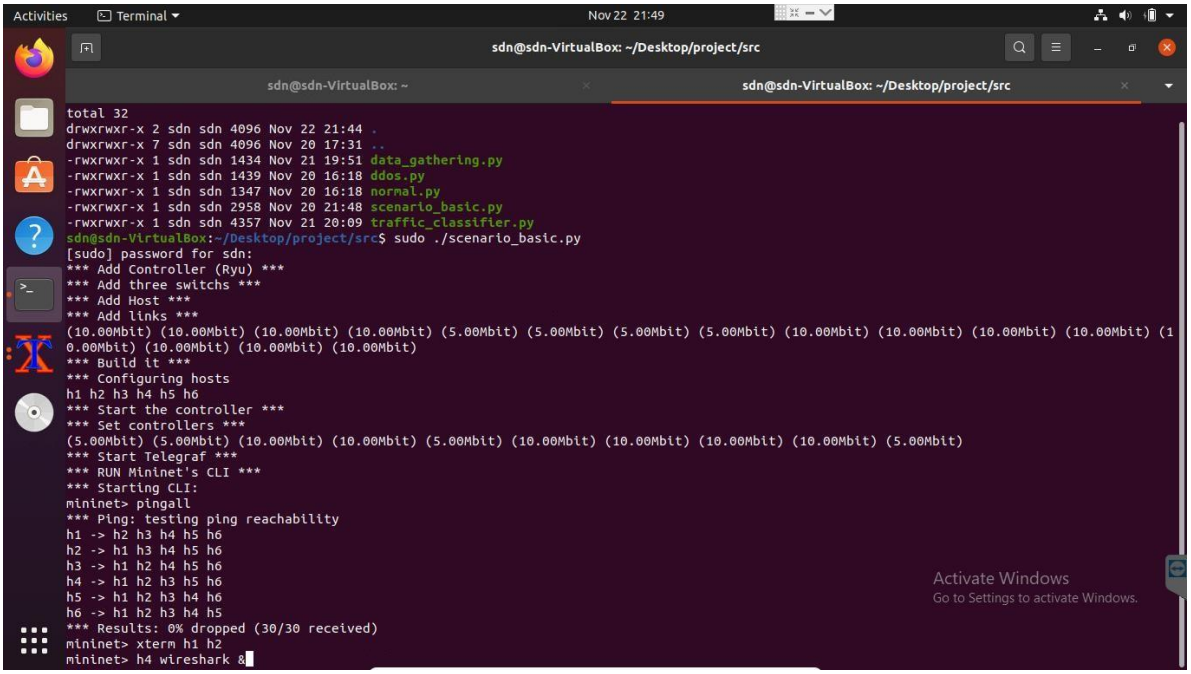
8.2 MODULE 2 : Performing DDoS Attack →



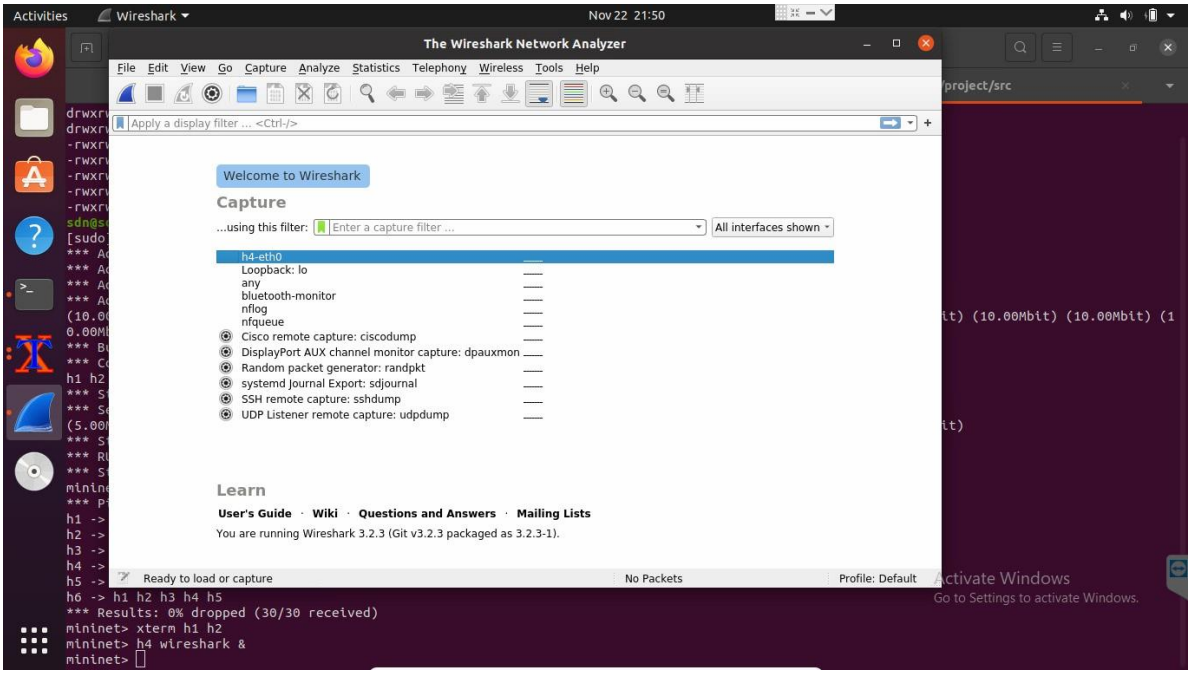
(Using Hping3 from Node H1)



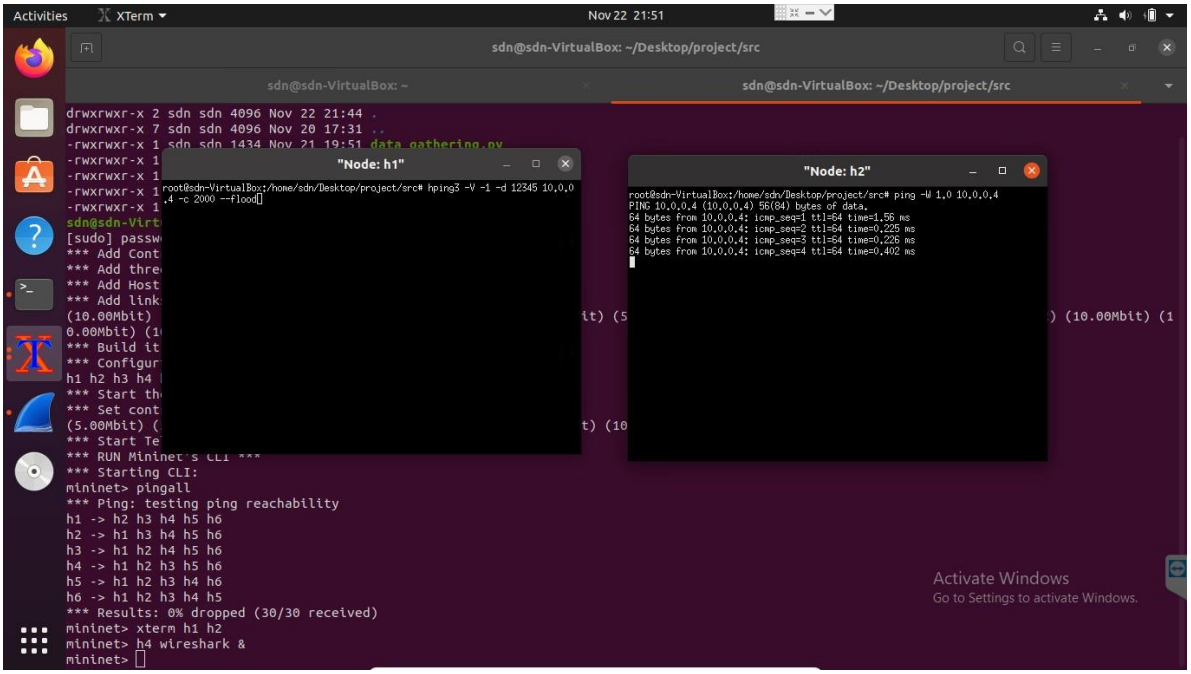
(Getting things ready for Attack)



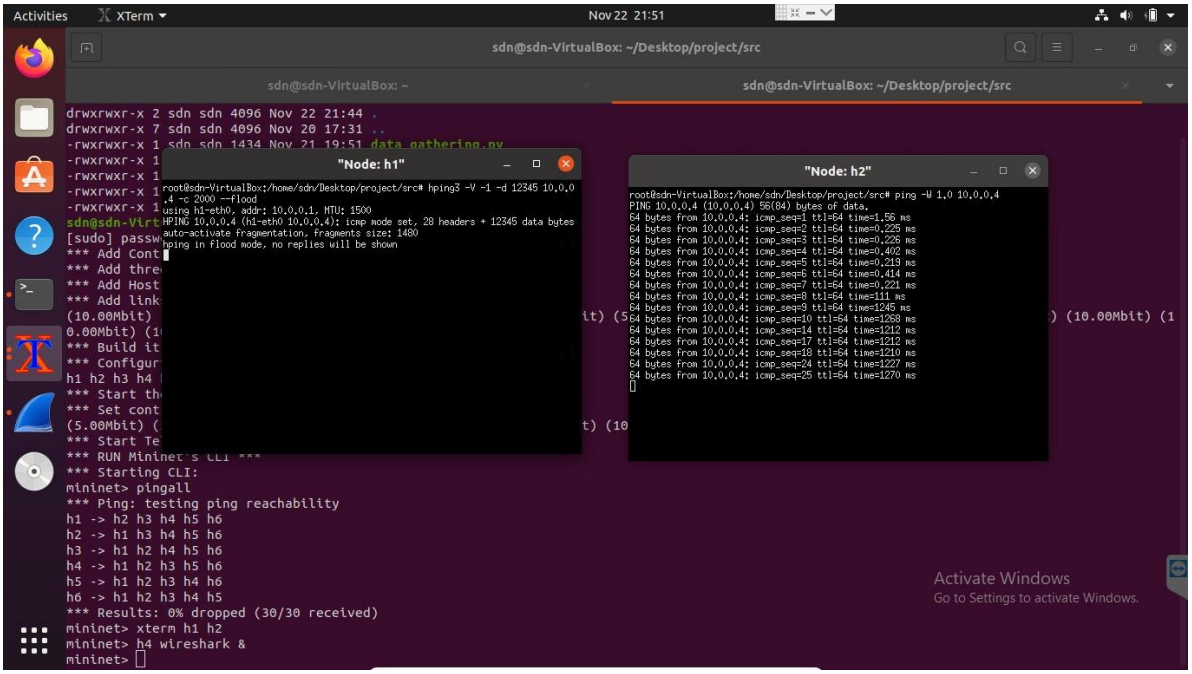
(Opening Wireshark)



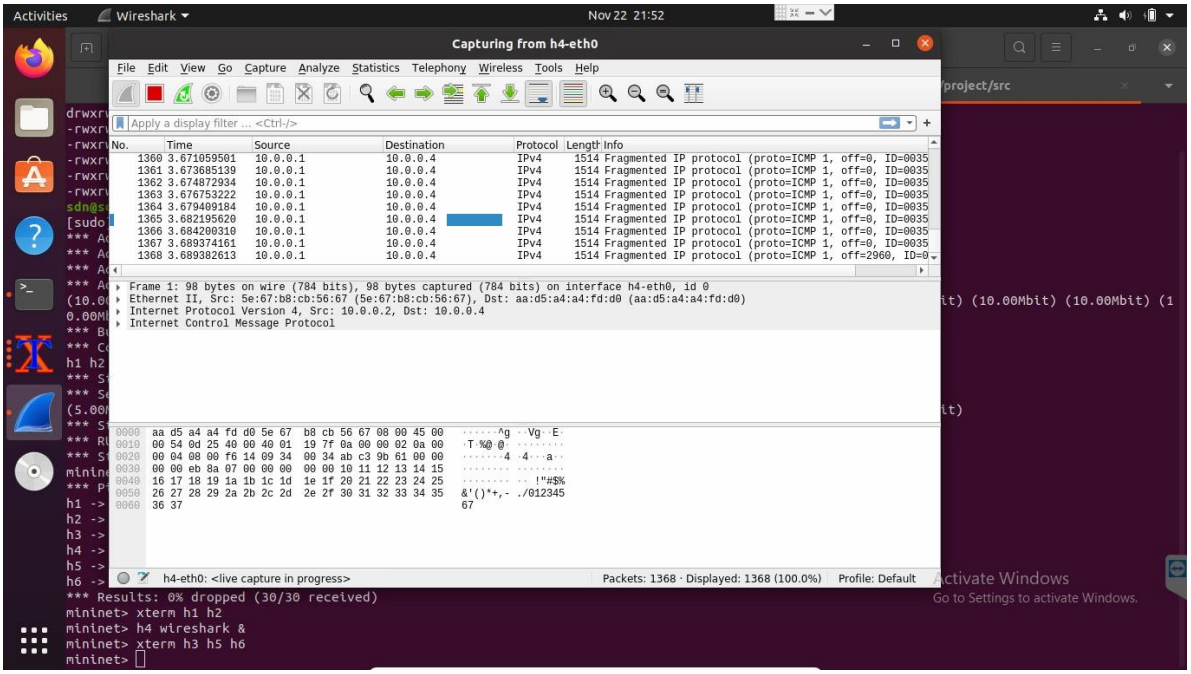
(Opening Wireshark on Node H4)



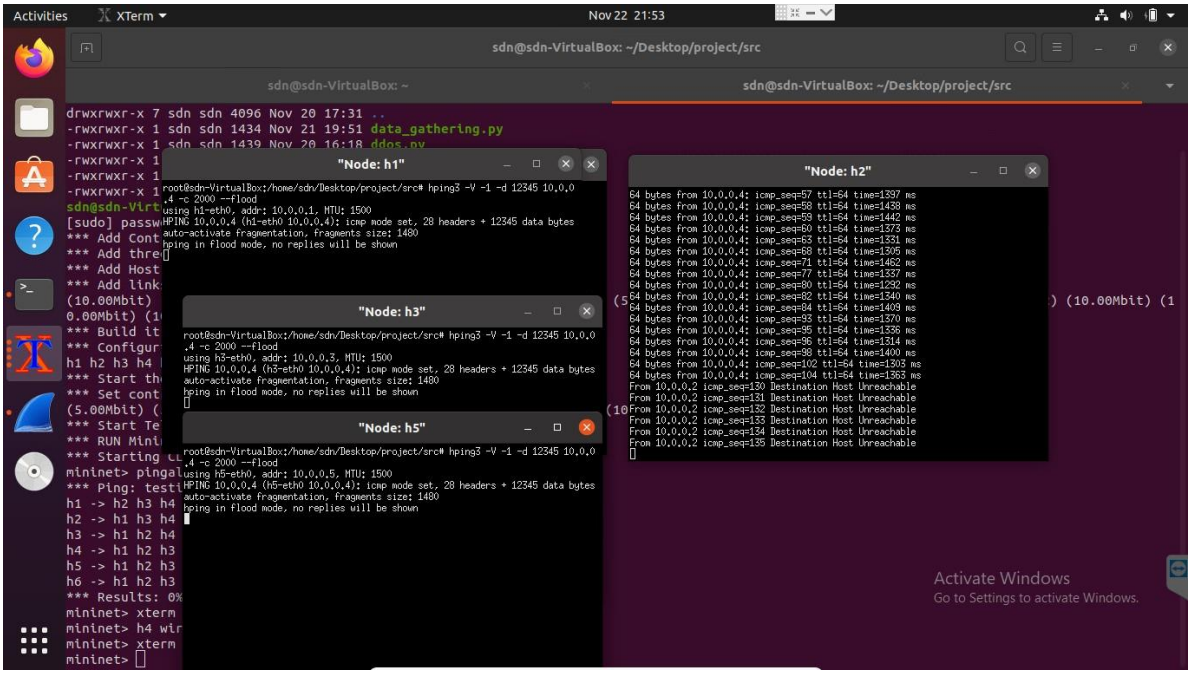
(Normal Traffic)



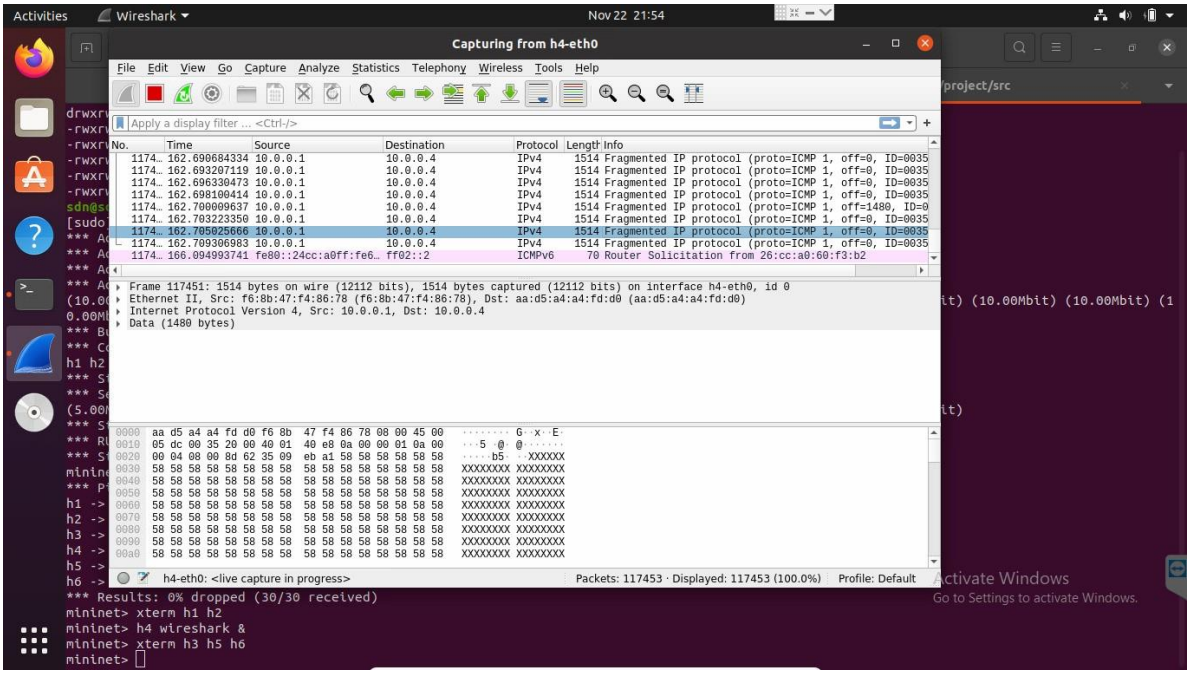
(DoS Attack Traffic)



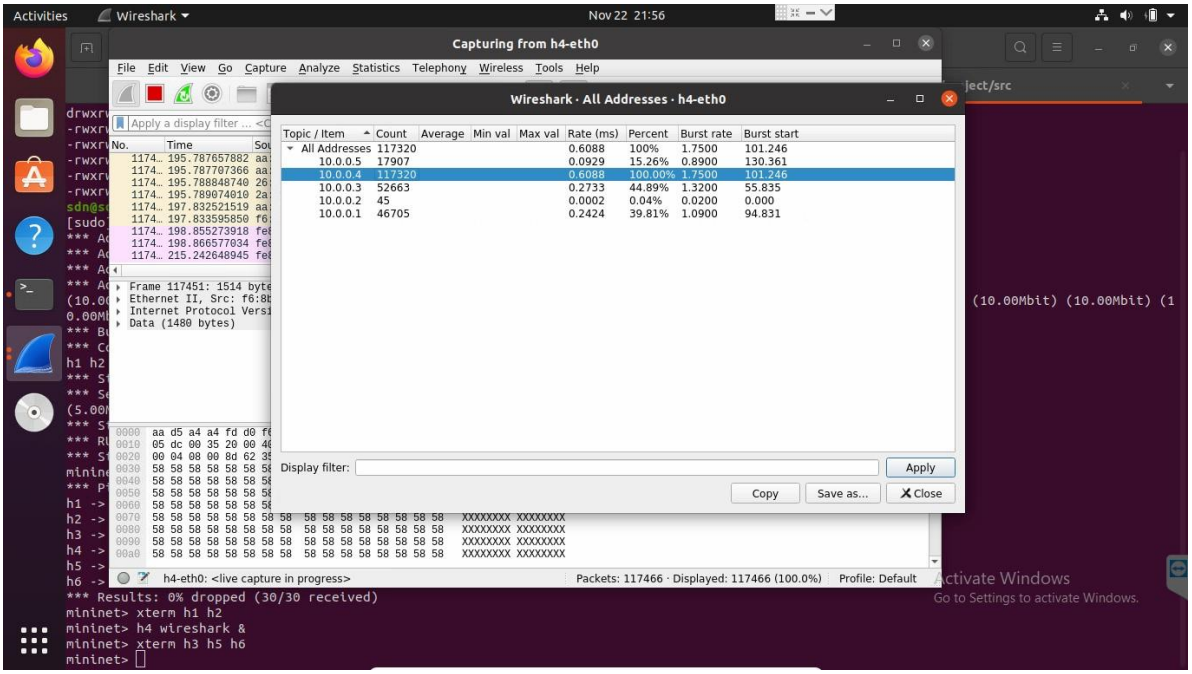
(Packets Scenario on H4 after DoS Attack)



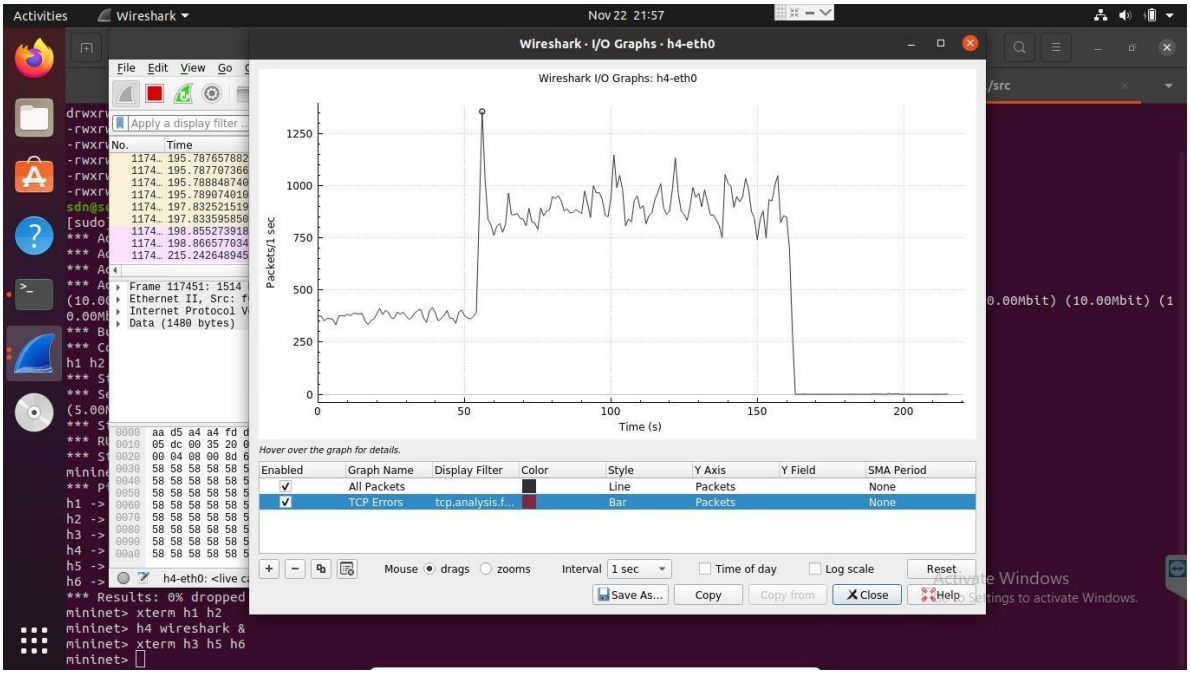
(DDoS Attack Traffic)



(Packets Scenario on H4 after DDoS Attack)

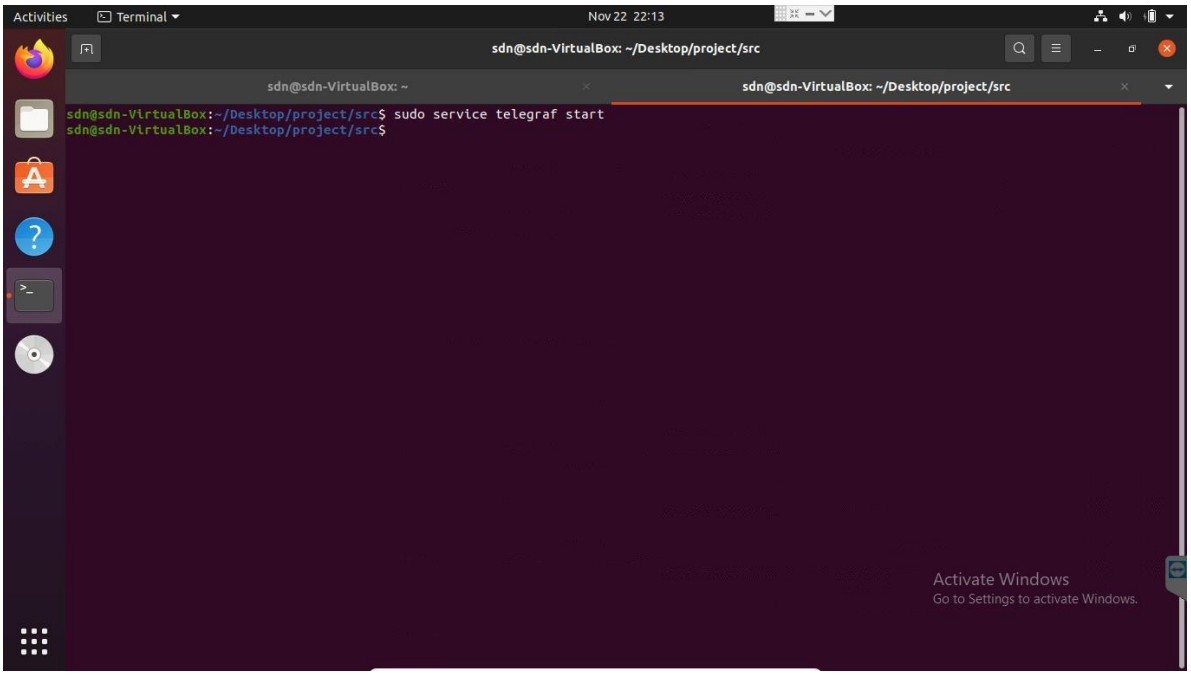


(100% Packet loss due to DDoS Attack)

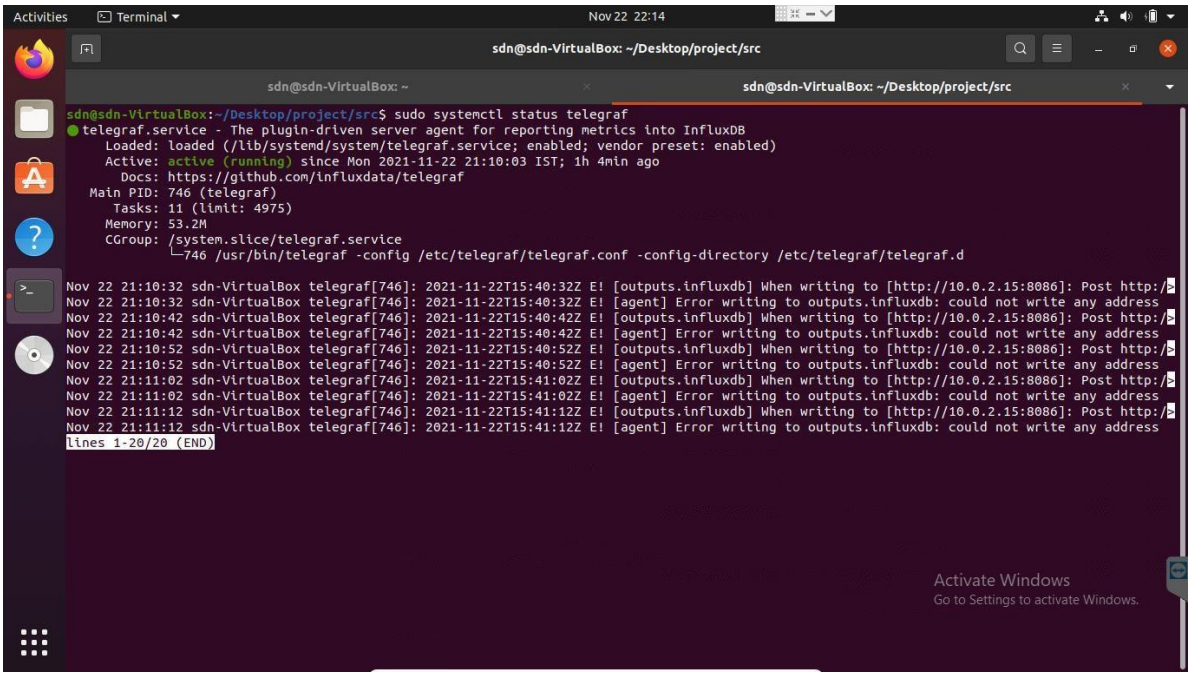


(Time-Packet graph on H4)

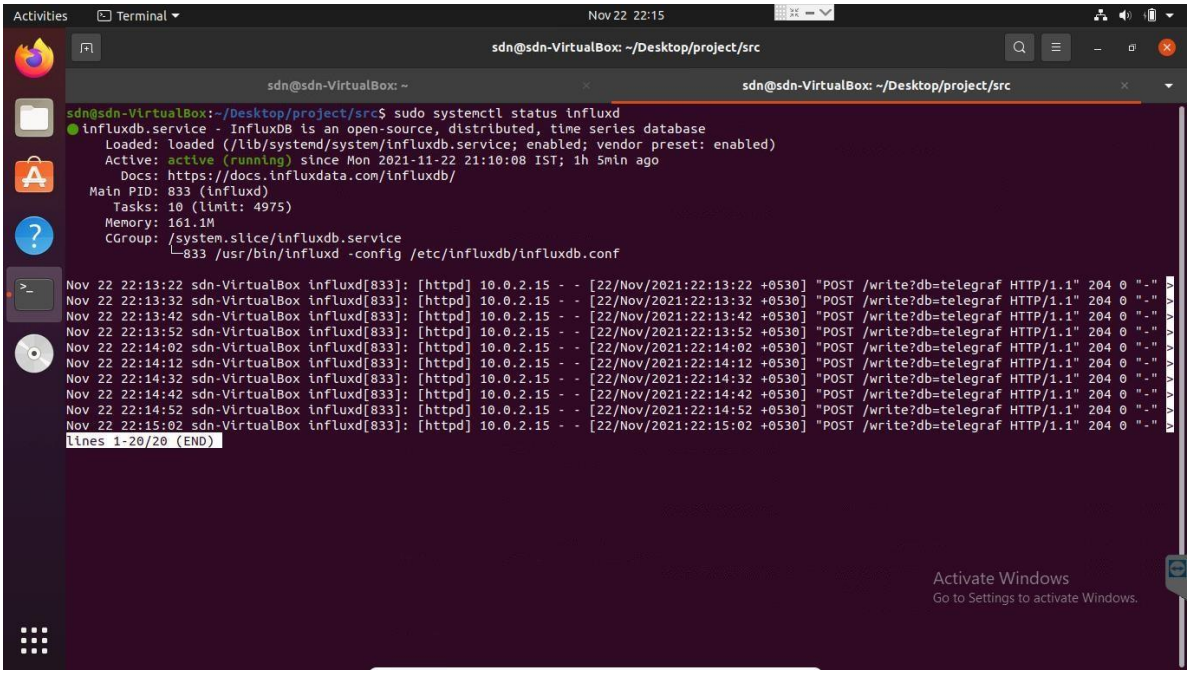
8.3 MODULE 3 (ICMP Data Gathering)→



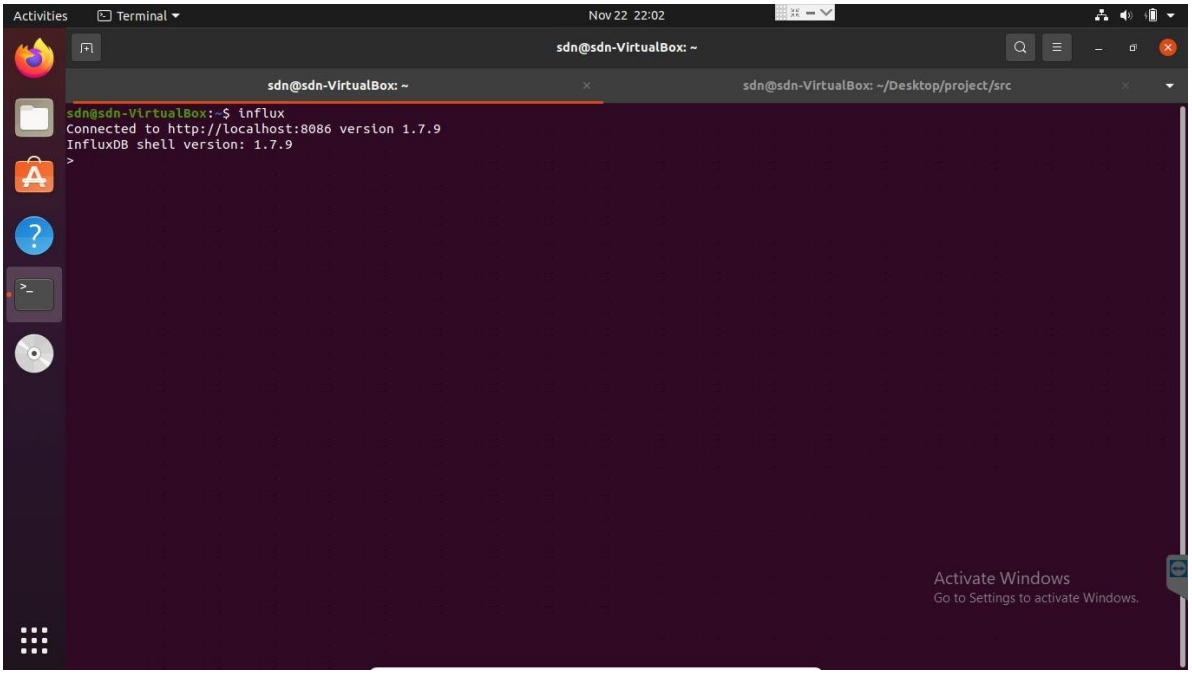
(Starting Telegraf)



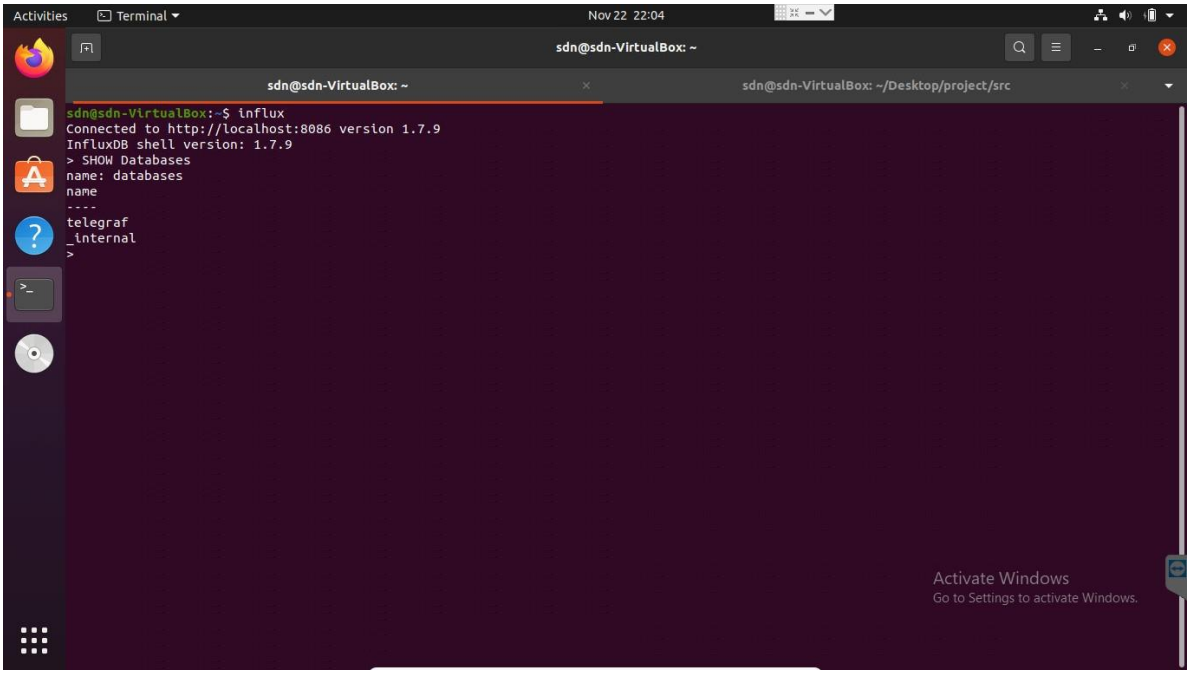
(Checking the status of Telegraf Service)



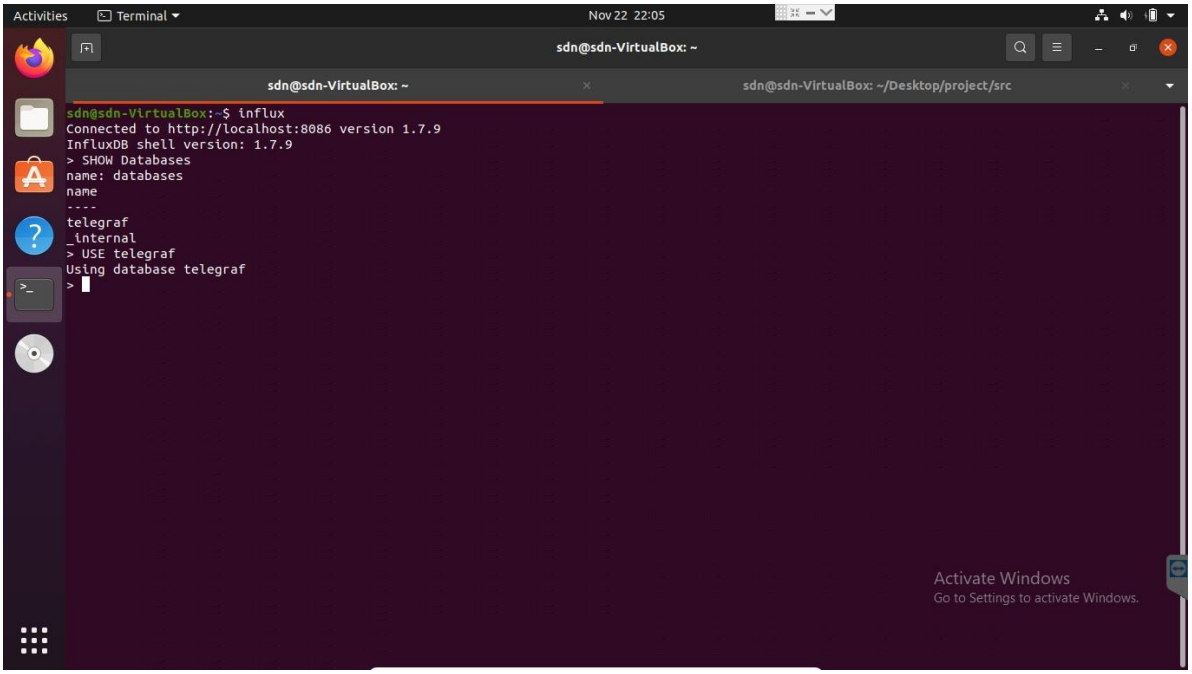
(Checking the status of InfluxDB Service)



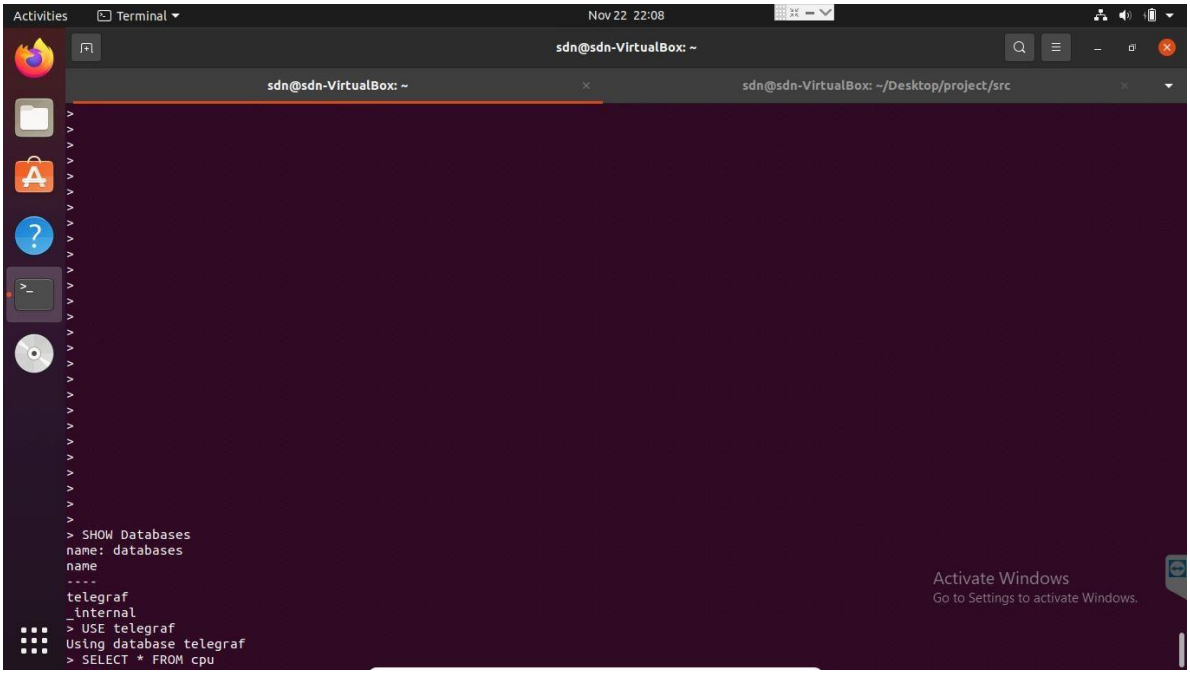
(Opening InfluxDB)



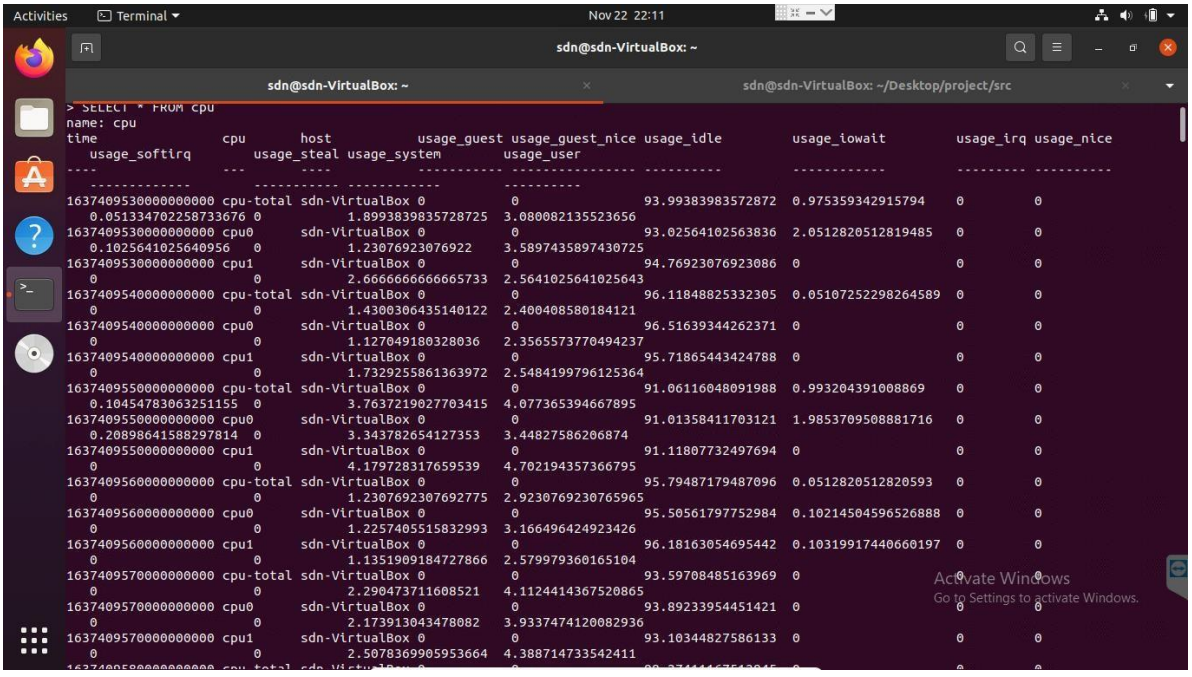
(Checking Database)



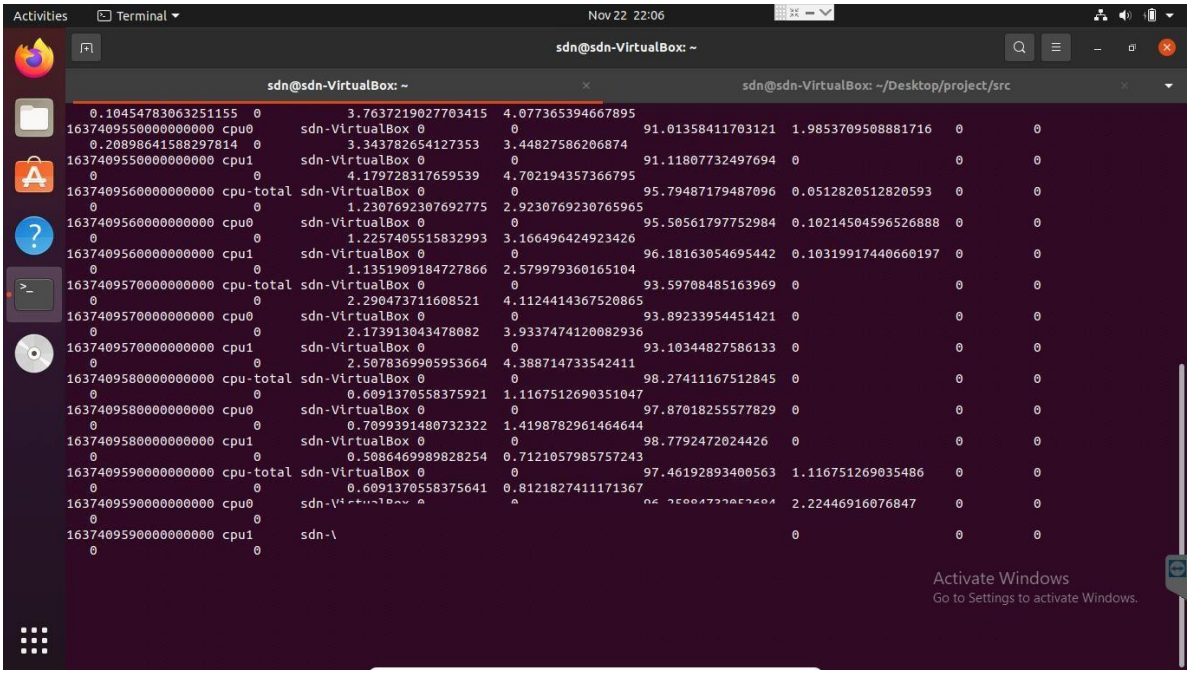
(Using Telegraf Database)



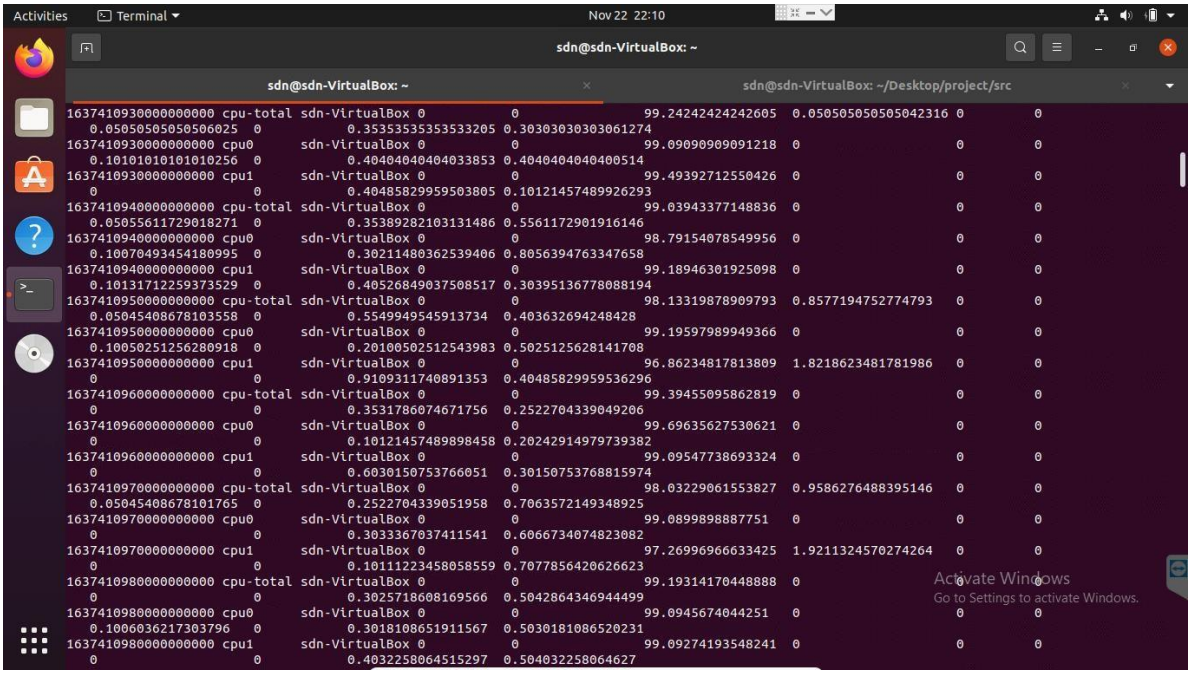
(Opening the ICMP traffic data)



(ICMP Traffic Data due to DDoS Attack)

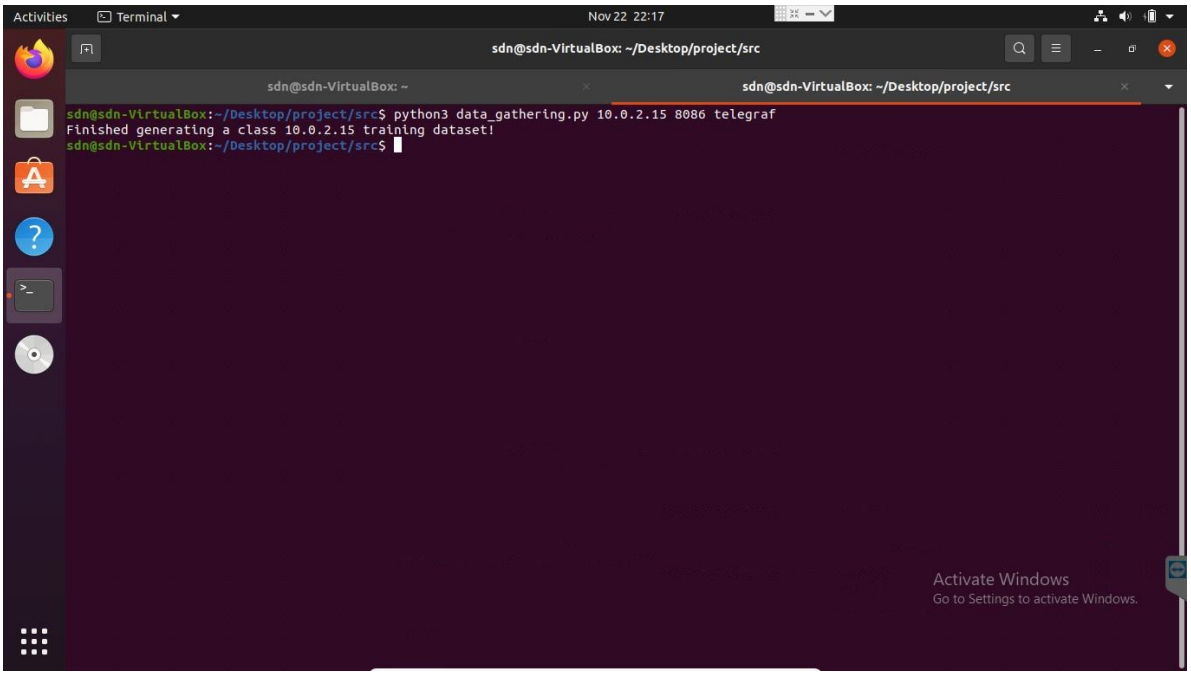


(ICMP Traffic data generated by Telegraf)

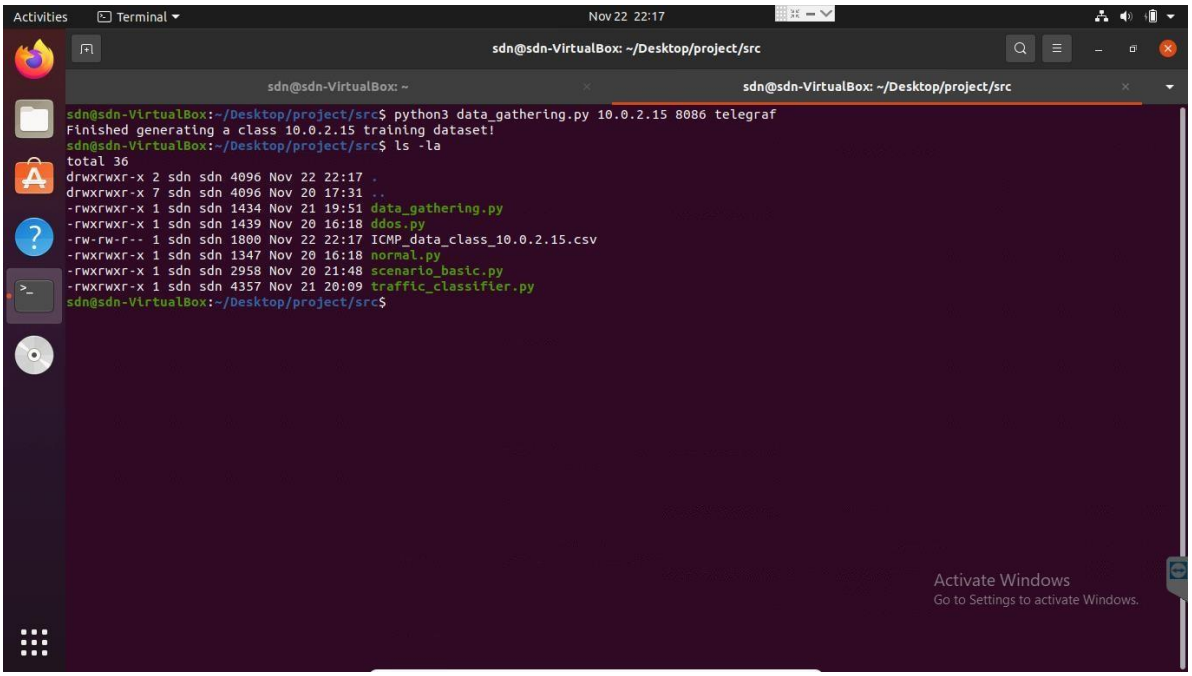


(ICMP Traffic Data stored in InfluxDB)

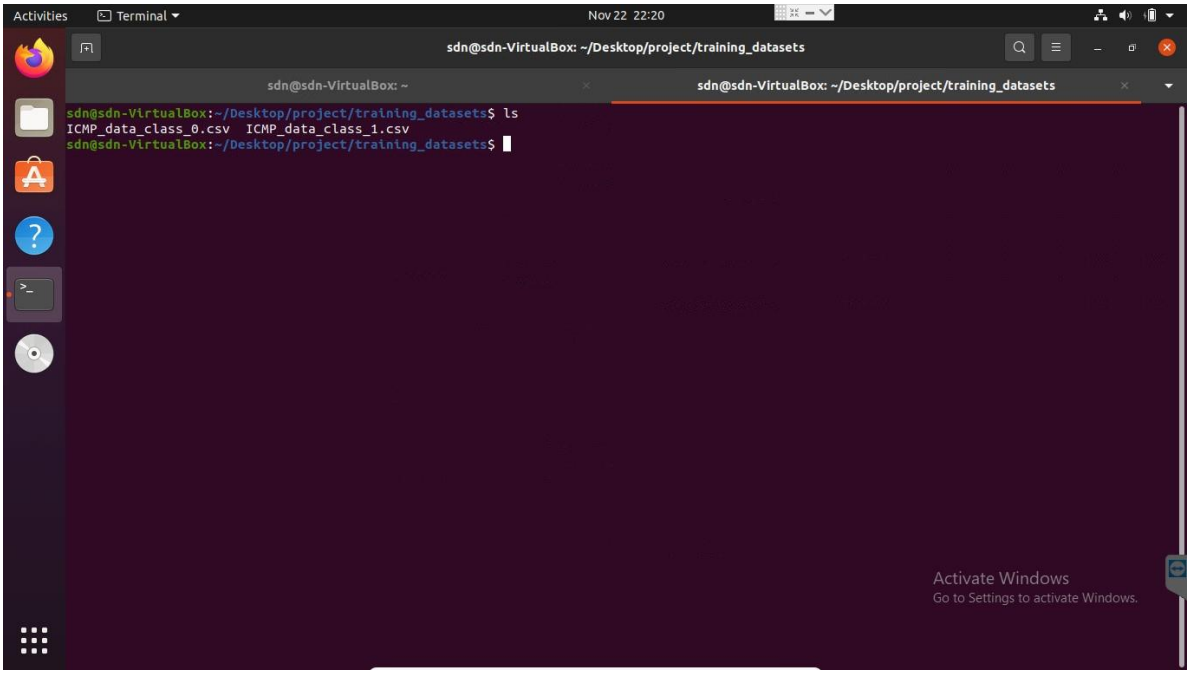
8.4 MODULE 4 (Creating datasets)→



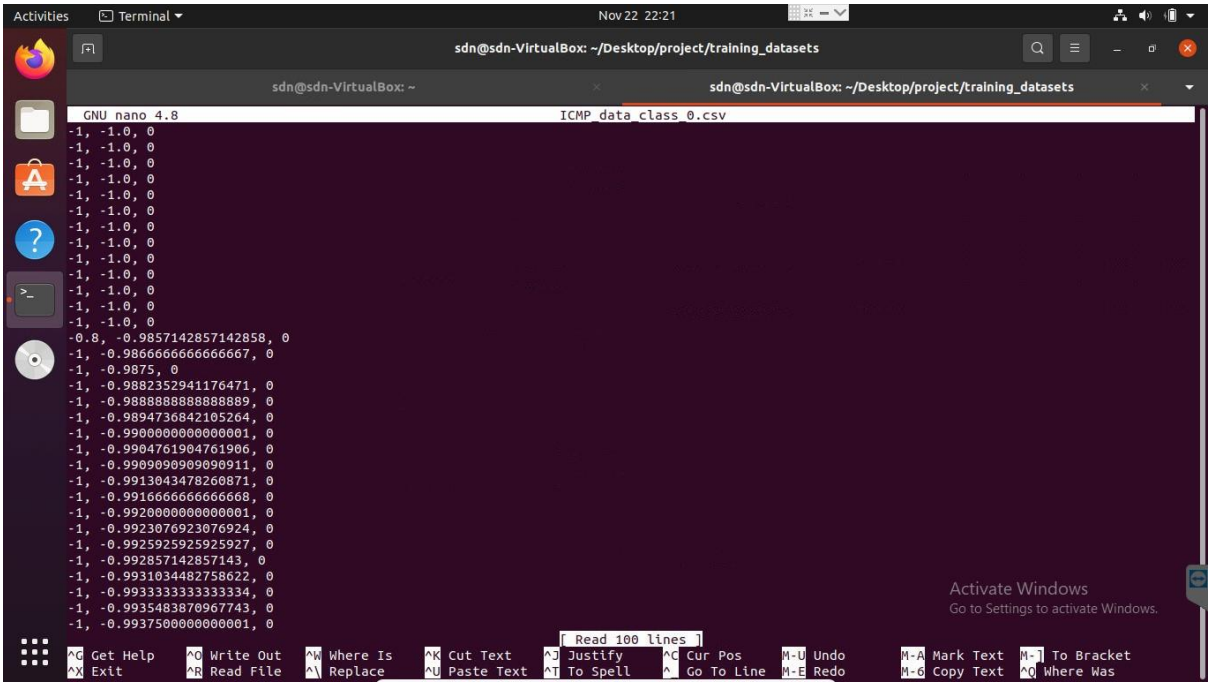
(Generating the training datasets)



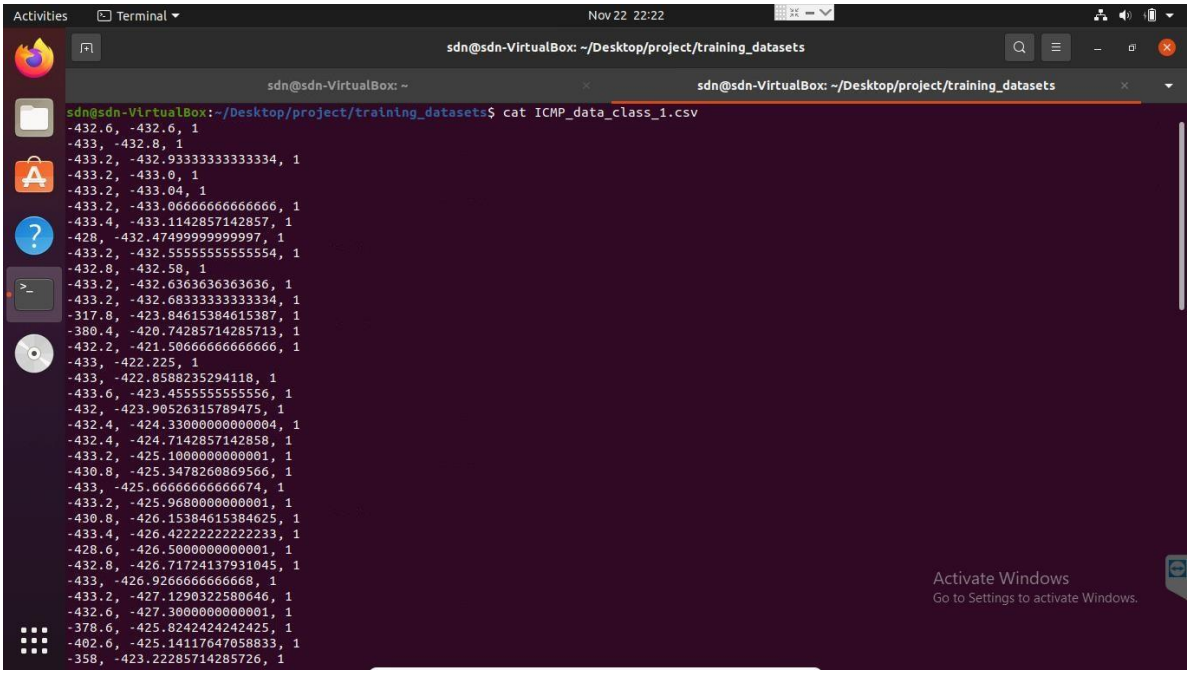
(Dataset Created)



(Datasets for Normal & DDoS Traffic)

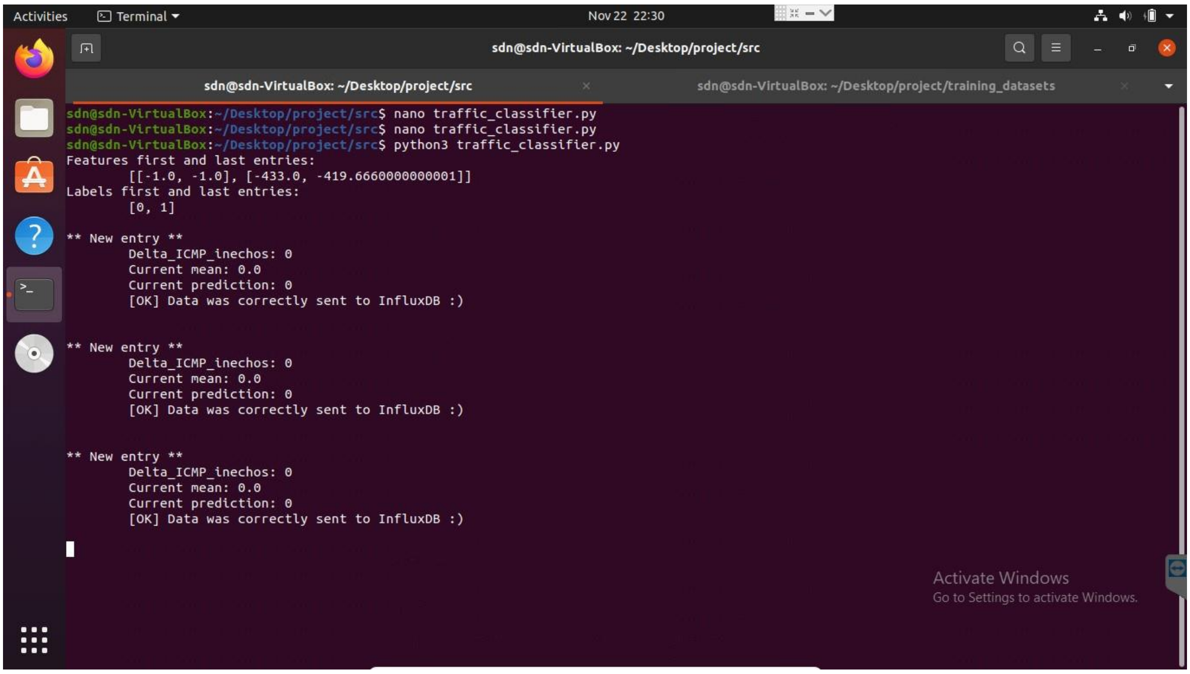


(Dataset of Normal Traffic)

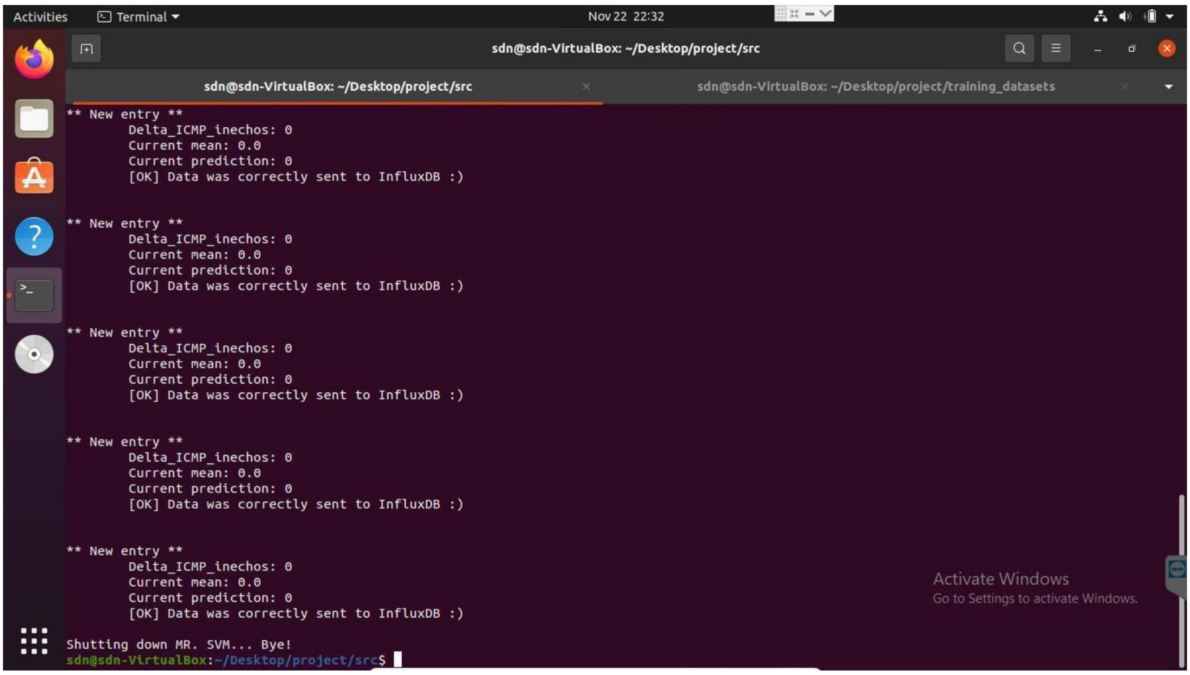


(Dataset of DDoS Traffic)

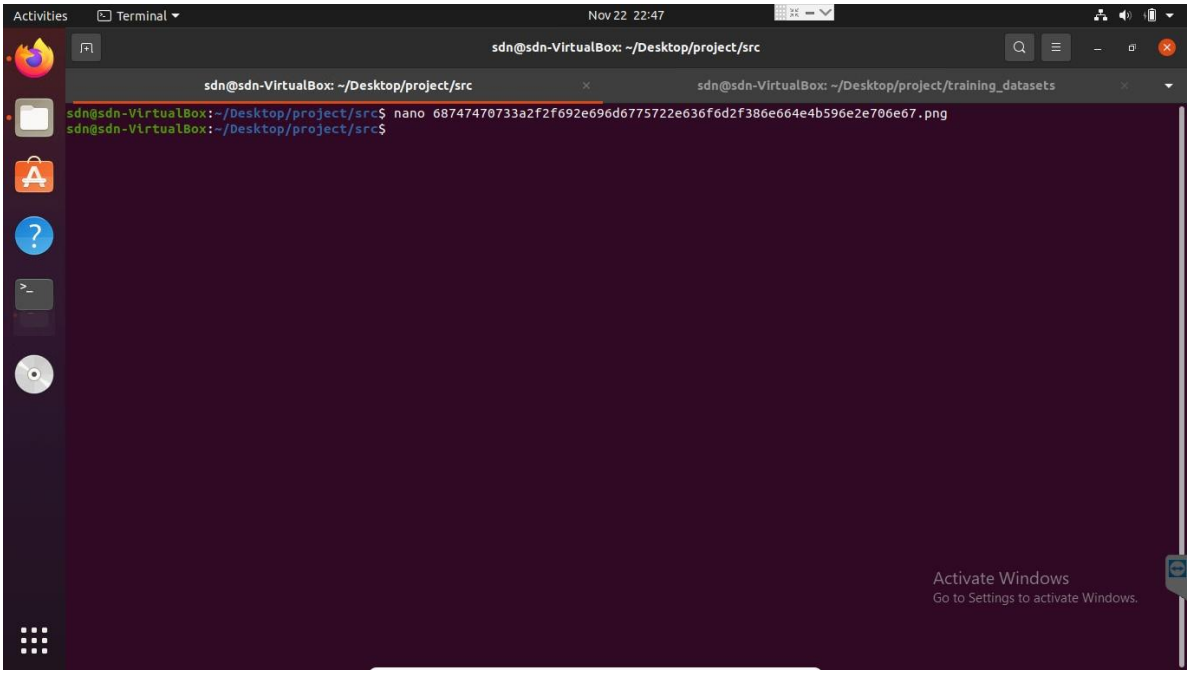
8.5 MODULE 5 (Classification of the traffics)→



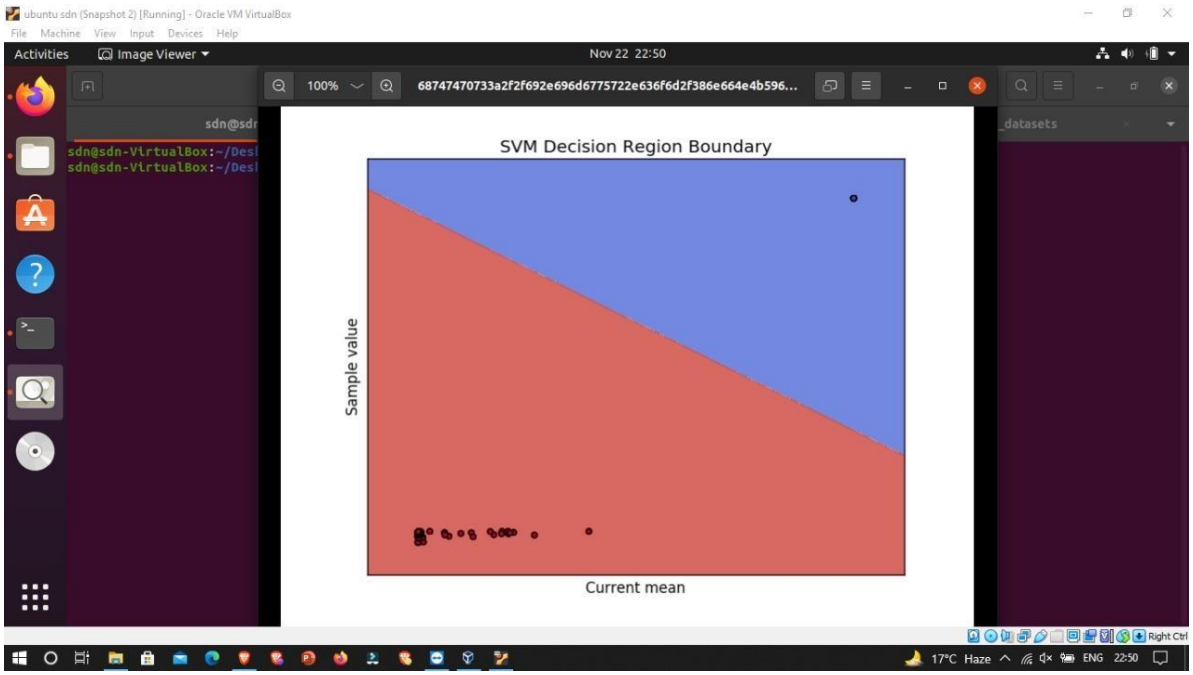
(Running the Traffic Classifier Script)



(SVM's Training Done)



(Graph generated by previous script)



(Graph showing the classification of Normal & DDoS Traffic)

CHAPTER 9

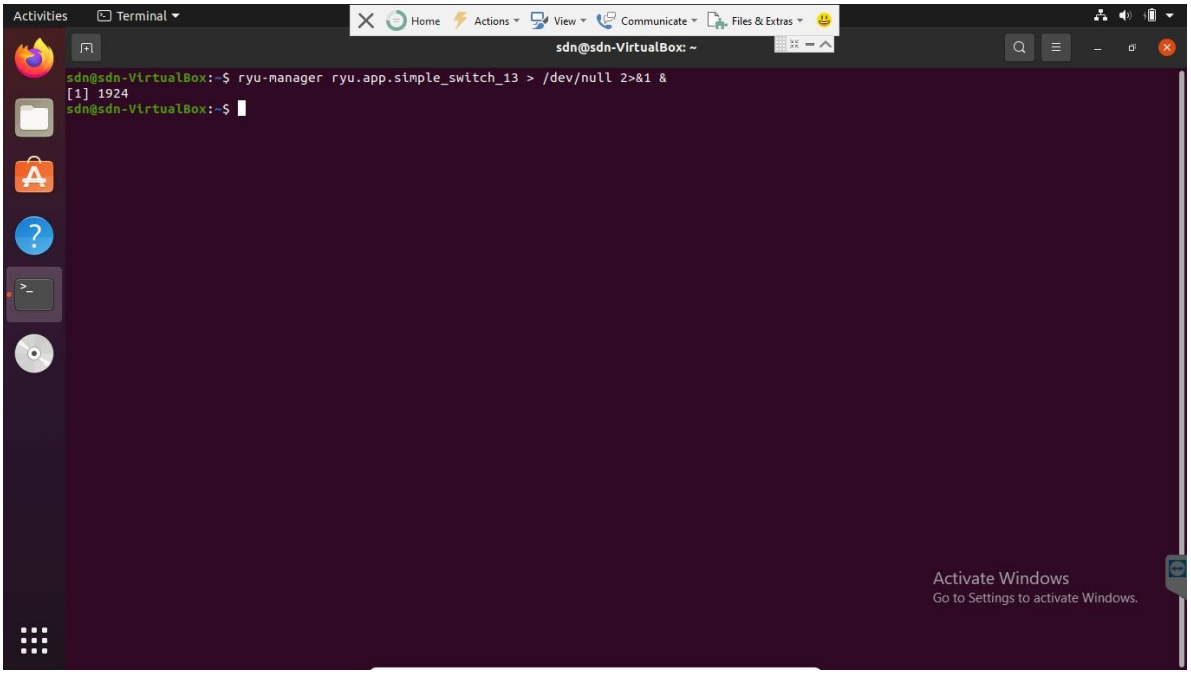
SYSTEMS TESTING

There can be the following test cases in the table below to check the working of the project as per our requirements and to test the whole system created.

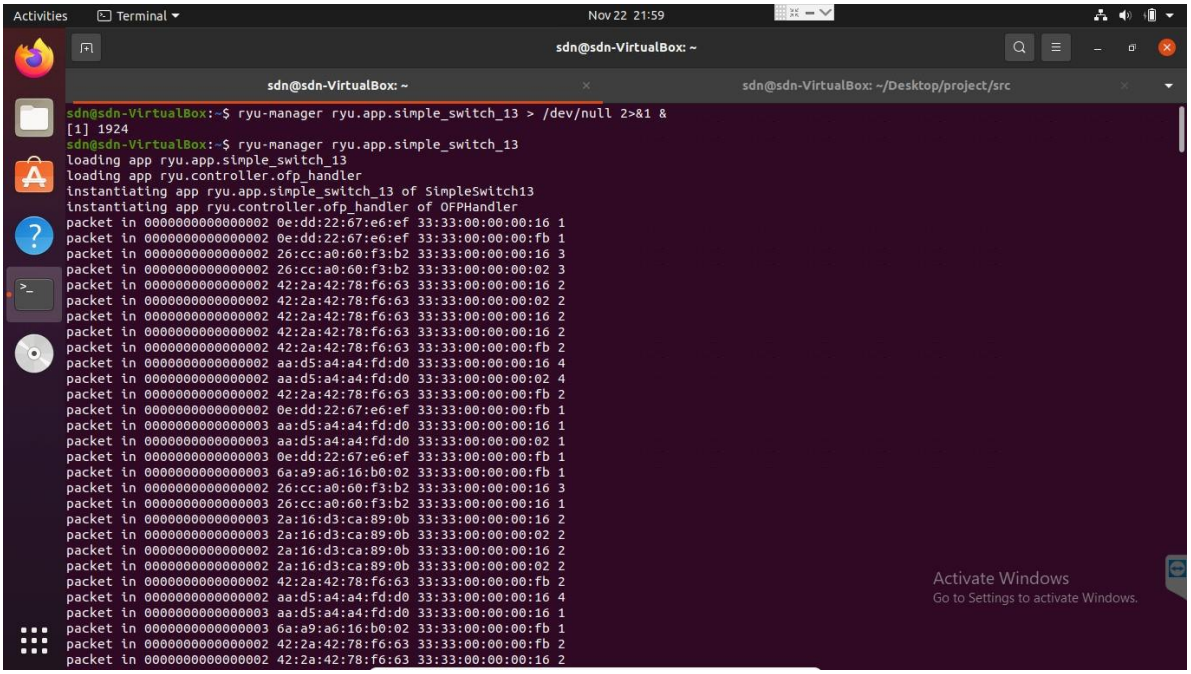
With the defined test cases in the table below, Screenshots are also there to verify and validate each test case via their respective number.

1. Is the RYU controller created ?
   - YES (see T1)
2. Is the RYU working okay ?
   - YES (see T2)
3. Is the Normal Traffic flowing well ?
   - YES (see T3)
4. Is the DDoS Attack happening ?
   - YES (see T4)
5. Is the status of Telegraf active ?
   - YES (see T5)
6. Is the status of InfluxDB active ?
   - YES (see T6)
7. Are the ICMP Traffic Data stored in the DB ?
   - YES (see T7)
8. Did the training datasets created for the SVM ?
   - YES (see T8)
9. Is the detection of the DDoS Attack being done ?
   - YES (see T9)

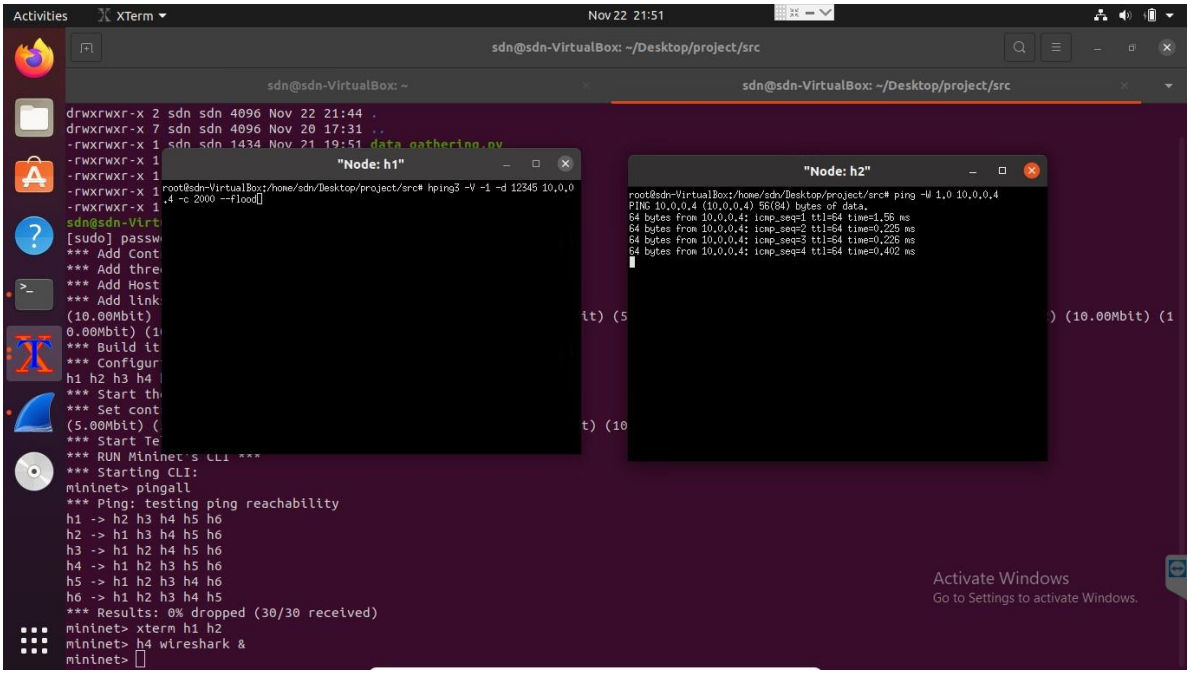
Table 2 _ Test Cases


Activities
Terminal
Home
Actions
View
Communicate
Files & Extras
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox:~$ ryu-manager ryu.app.simple_switch_13 > /dev/null 2>&1 &
[1] 1924
sdn@sdn-VirtualBox:~$
Activate Windows
Go to Settings to activate Windows.

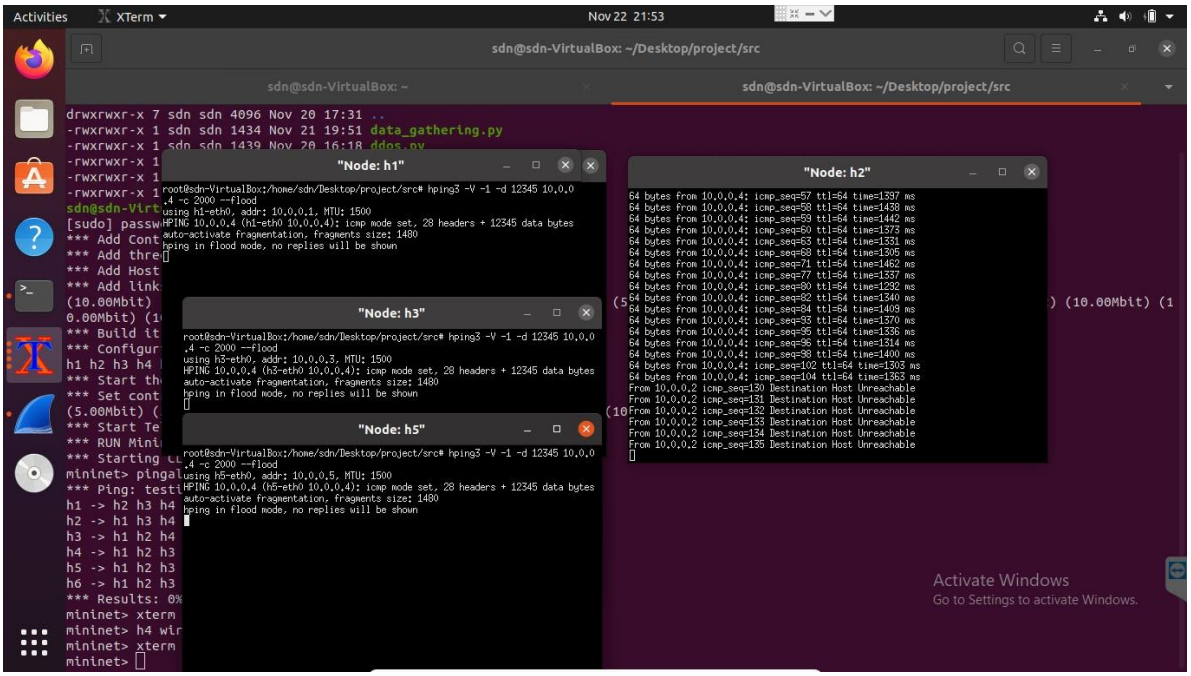
(T1)


Activities
Terminal
Nov 22 21:59
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox:~$ ryu-manager ryu.app.simple_switch_13 > /dev/null 2>&1 &
[1] 1924
sdn@sdn-VirtualBox:~$ ryu-manager ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 0000000000000002 0e:dd:22:67:e6:ef 33:33:00:00:00:16 1
packet in 0000000000000002 0e:dd:22:67:e6:ef 33:33:00:00:00:fb 1
packet in 0000000000000002 26:cc:a0:60:f3:b2 33:33:00:00:00:16 3
packet in 0000000000000002 26:cc:a0:60:f3:b2 33:33:00:00:00:02 3
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:16 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:02 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:16 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:16 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:fb 2
packet in 0000000000000002 aa:d5:a4:a4:fd:d0 33:33:00:00:00:16 4
packet in 0000000000000002 aa:d5:a4:a4:fd:d0 33:33:00:00:00:02 4
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:fb 2
packet in 0000000000000002 0e:dd:22:67:e6:ef 33:33:00:00:00:fb 1
packet in 0000000000000003 aa:d5:a4:a4:fd:d0 33:33:00:00:00:16 1
packet in 0000000000000003 aa:d5:a4:a4:fd:d0 33:33:00:00:00:02 1
packet in 0000000000000003 0e:dd:22:67:e6:ef 33:33:00:00:00:fb 1
packet in 0000000000000003 6a:a9:a6:16:b0:02 33:33:00:00:00:fb 1
packet in 0000000000000002 26:cc:a0:60:f3:b2 33:33:00:00:00:16 3
packet in 0000000000000003 26:cc:a0:60:f3:b2 33:33:00:00:00:16 1
packet in 0000000000000003 2a:16:d3:ca:89:0b 33:33:00:00:00:16 2
packet in 0000000000000003 2a:16:d3:ca:89:0b 33:33:00:00:00:02 2
packet in 0000000000000002 2a:16:d3:ca:89:0b 33:33:00:00:00:16 2
packet in 0000000000000002 2a:16:d3:ca:89:0b 33:33:00:00:00:02 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:fb 2
packet in 0000000000000002 aa:d5:a4:a4:fd:d0 33:33:00:00:00:16 4
packet in 0000000000000003 aa:d5:a4:a4:fd:d0 33:33:00:00:00:16 1
packet in 0000000000000003 6a:a9:a6:16:b0:02 33:33:00:00:00:fb 1
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:fb 2
packet in 0000000000000002 42:2a:42:78:f6:63 33:33:00:00:00:16 2
Activate Windows
Go to Settings to activate Windows.

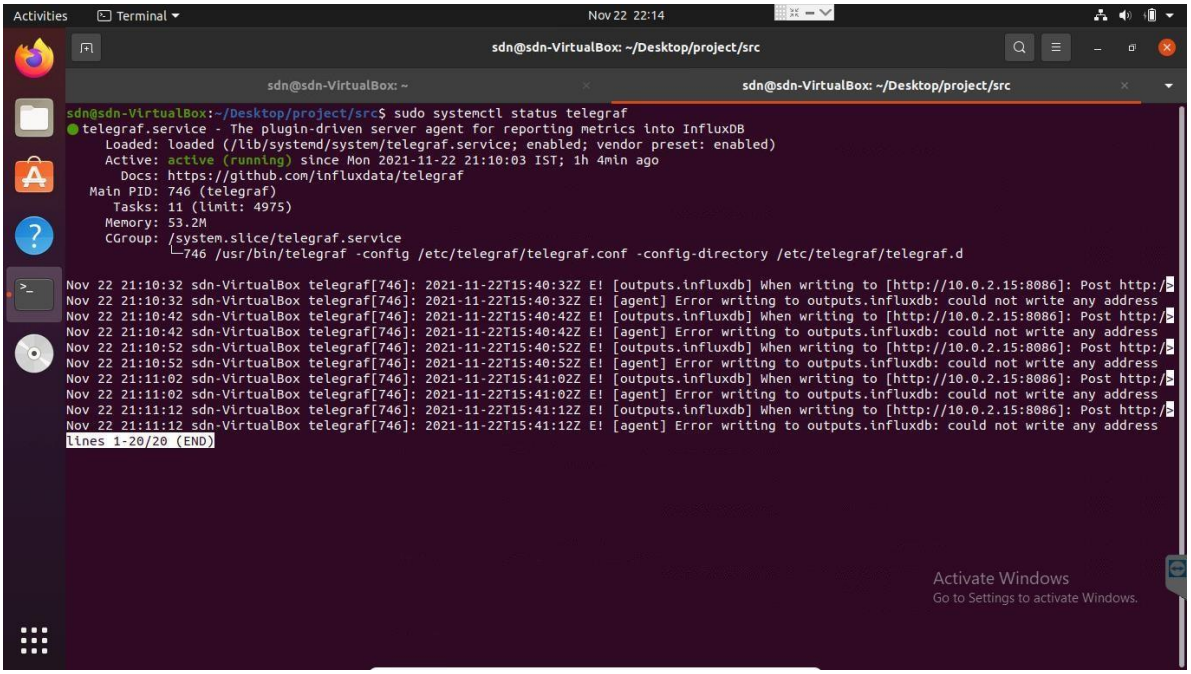
(T2)


Activities
XTerm
Nov 22 21:51
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
"Node: h1"
"Node: h2"
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
mininet> xterm h1 h2
mininet> h4 wireshark &
Activate Windows
Go to Settings to activate Windows.

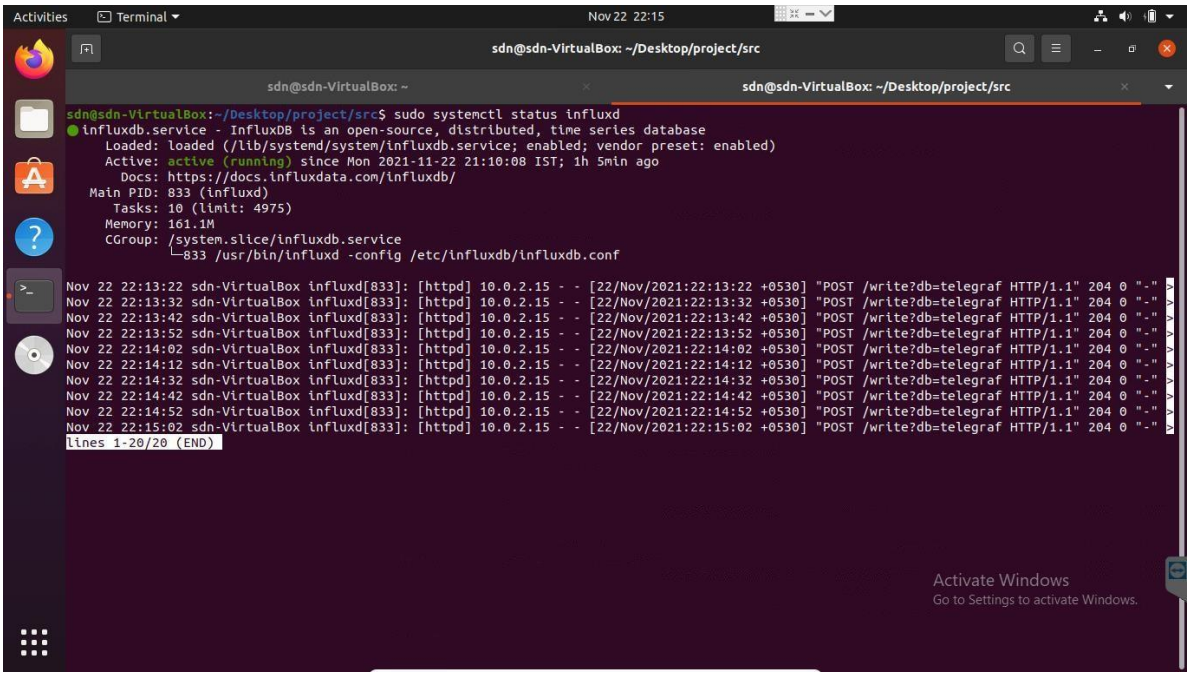
(T3)


Activities
XTerm
Nov 22 21:53
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
"Node: h1"
"Node: h2"
"Node: h3"
"Node: h5"
Activate Windows
Go to Settings to activate Windows.

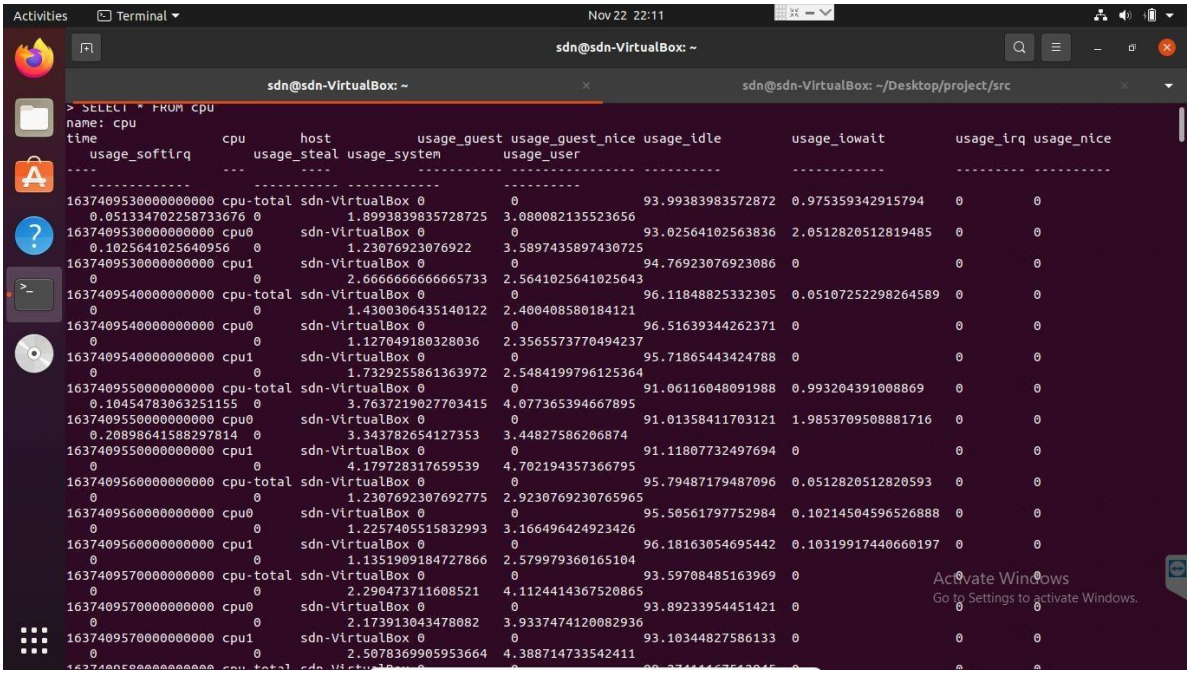
(T4)


Activities
Terminal
Nov 22 22:14
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox:~/Desktop/project/src$ sudo systemctl status telegraf
telegraf.service - The plugin-driven server agent for reporting metrics into InfluxDB
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2021-11-22 21:10:03 IST; 1h 4min ago
Docs: https://github.com/influxdata/telegraf
Main PID: 746 (telegraf)
Tasks: 11 (limit: 4975)
Memory: 53.2M
CGroup: /system.slice/telegraf.service
└746 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d
Nov 22 21:10:32 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:32Z E! [outputs.influxdb] When writing to [http://10.0.2.15:8086]: Post http:/
Nov 22 21:10:32 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:32Z E! [agent] Error writing to outputs.influxdb: could not write any address
Nov 22 21:10:42 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:42Z E! [outputs.influxdb] When writing to [http://10.0.2.15:8086]: Post http:/
Nov 22 21:10:42 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:42Z E! [agent] Error writing to outputs.influxdb: could not write any address
Nov 22 21:10:52 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:52Z E! [outputs.influxdb] When writing to [http://10.0.2.15:8086]: Post http:/
Nov 22 21:10:52 sdn-VirtualBox telegraf[746]: 2021-11-22T15:40:52Z E! [agent] Error writing to outputs.influxdb: could not write any address
Nov 22 21:11:02 sdn-VirtualBox telegraf[746]: 2021-11-22T15:41:02Z E! [outputs.influxdb] When writing to [http://10.0.2.15:8086]: Post http:/
Nov 22 21:11:02 sdn-VirtualBox telegraf[746]: 2021-11-22T15:41:02Z E! [agent] Error writing to outputs.influxdb: could not write any address
Nov 22 21:11:12 sdn-VirtualBox telegraf[746]: 2021-11-22T15:41:12Z E! [outputs.influxdb] When writing to [http://10.0.2.15:8086]: Post http:/
Nov 22 21:11:12 sdn-VirtualBox telegraf[746]: 2021-11-22T15:41:12Z E! [agent] Error writing to outputs.influxdb: could not write any address
lines 1-20/20 (END)
Activate Windows
Go to Settings to activate Windows.

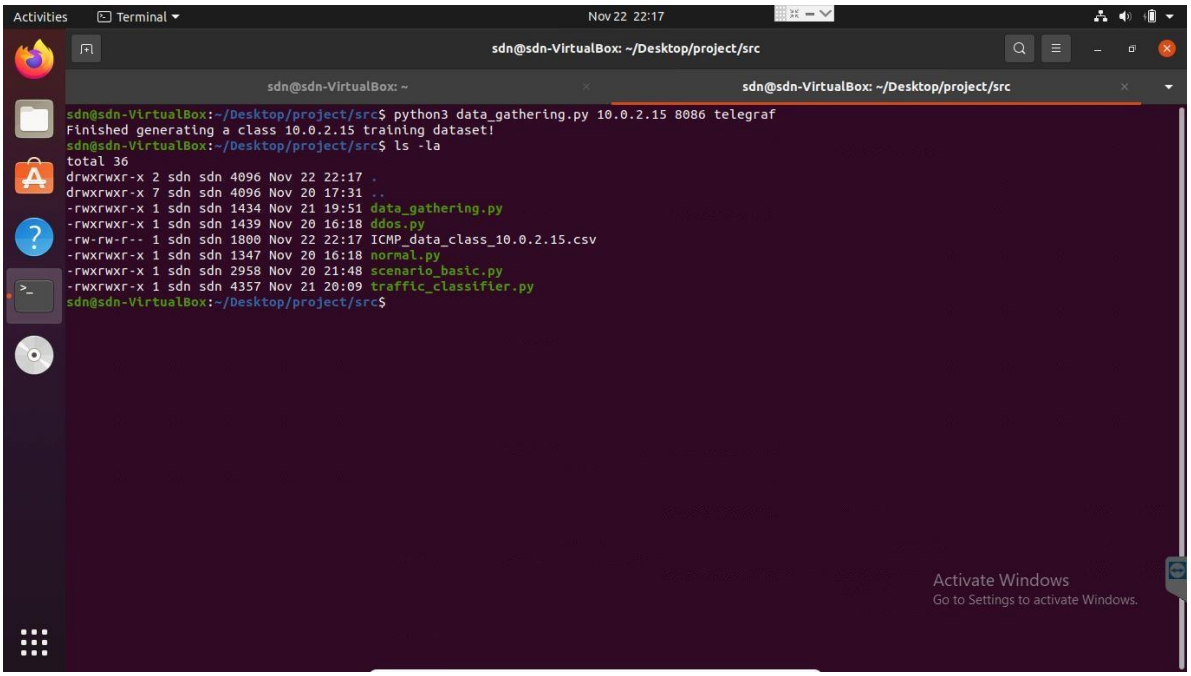
(T5)


Activities
Terminal
Nov 22 22:15
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox:~/Desktop/project/src$ sudo systemctl status influxd
influxdb.service - InfluxDB is an open-source, distributed, time series database
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
Active: active (running) since Mon 2021-11-22 21:10:08 IST; 1h 5min ago
Docs: https://docs.influxdata.com/influxdb/
Main PID: 833 (influxd)
Tasks: 10 (limit: 4975)
Memory: 161.1M
CGroup: /system.slice/influxdb.service
└833 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
Nov 22 22:13:22 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:13:22 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:13:32 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:13:32 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:13:42 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:13:42 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:13:52 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:13:52 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:14:02 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:14:02 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:14:12 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:14:12 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:14:32 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:14:32 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:14:42 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:14:42 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:14:52 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:14:52 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
Nov 22 22:15:02 sdn-VirtualBox influxd[833]: [httpd] 10.0.2.15 - - [22/Nov/2021:22:15:02 +0530] "POST /write?db=telegraf HTTP/1.1" 204 0 "-"
lines 1-20/20 (END)
Activate Windows
Go to Settings to activate Windows.

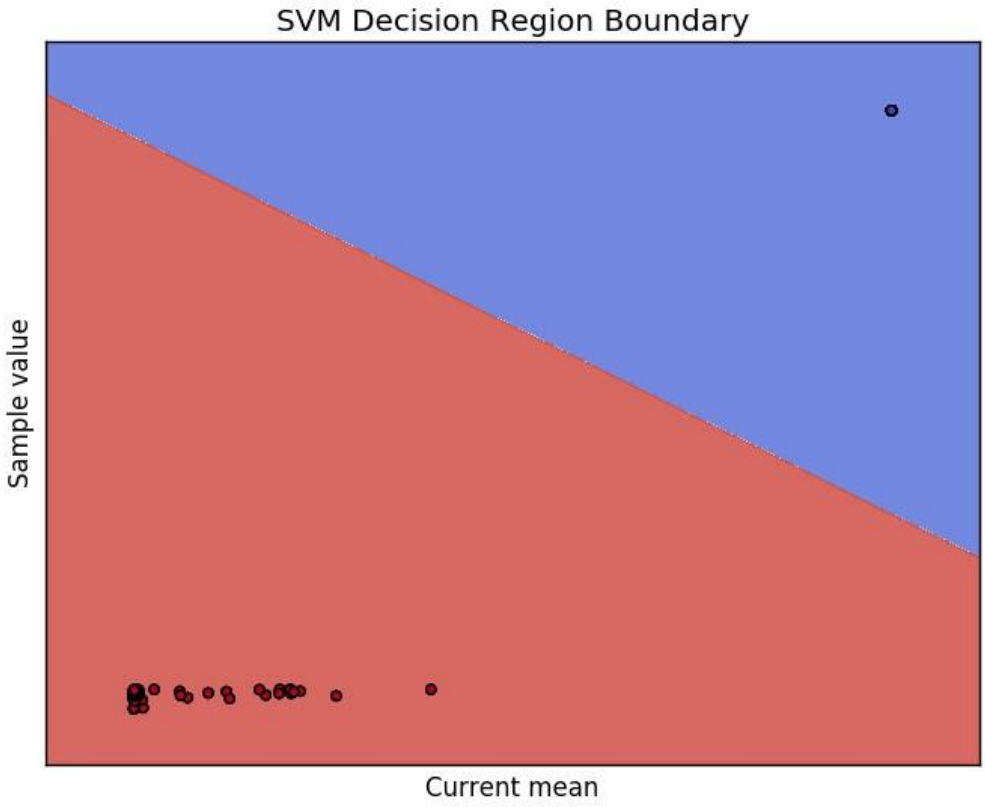
(T6)


Activities
Terminal
Nov 22 22:11
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
> SELECT * FROM cpu
name: cpu
time cpu host usage_guest usage_guest_nice usage_idle usage_iowait usage_irq usage_nice
usage_softirq usage_steal usage_system usage_user
1637409530000000000 cpu-total sdn-VirtualBox 0 0 93.99383983572872 0.975359342915794 0 0
0.051334702258733676 0 1.8993839835728725 3.080082135523656
1637409530000000000 cpu0 sdn-VirtualBox 0 0 93.02564102563836 2.0512820512819485 0 0
0.1025641025640956 0 1.23076923076922 3.5897435897430725
1637409530000000000 cpu1 sdn-VirtualBox 0 0 94.76923076923086 0 0 0
0 0 2.6666666666665733 2.5641025641025643
1637409540000000000 cpu-total sdn-VirtualBox 0 0 96.11848825332305 0.05107252298264589 0 0
0 0 1.4300306435140122 2.400408580184121
1637409540000000000 cpu0 sdn-VirtualBox 0 0 96.51639344262371 0 0 0
0 0 1.127049180328036 2.3565573770494237
1637409540000000000 cpu1 sdn-VirtualBox 0 0 95.71865443424788 0 0 0
0 0 1.7329255861363972 2.5484199796125364
1637409550000000000 cpu-total sdn-VirtualBox 0 0 91.06116048091988 0.993204391008869 0 0
0.10454783063251155 0 3.7637219027703415 4.077365394667895
1637409550000000000 cpu0 sdn-VirtualBox 0 0 91.01358411703121 1.9853709508881716 0 0
0.20898641588297814 0 3.343782654127353 3.44827586206874
1637409550000000000 cpu1 sdn-VirtualBox 0 0 91.11807732497694 0 0 0
0 0 4.179728317659539 4.702194357366795
1637409560000000000 cpu-total sdn-VirtualBox 0 0 95.79487179487096 0.0512820512820593 0 0
0 0 1.2307692307692775 2.9230769230765965
1637409560000000000 cpu0 sdn-VirtualBox 0 0 95.50561797752984 0.10214504596526888 0 0
0 0 1.2257405515832993 3.166496424923426
1637409560000000000 cpu1 sdn-VirtualBox 0 0 96.18163054695442 0.10319917440660197 0 0
0 0 1.1351909184727866 2.579979360165104
1637409570000000000 cpu-total sdn-VirtualBox 0 0 93.59708485163969 0 0 0
0 0 2.290473711608521 4.1124414367520865
1637409570000000000 cpu0 sdn-VirtualBox 0 0 93.89233954451421 0 0 0
0 0 2.173913043478082 3.9337474120082936
1637409570000000000 cpu1 sdn-VirtualBox 0 0 93.10344827586133 0 0 0
0 0 2.5078369905953664 4.388714733542411
Activate Windows
Go to Settings to activate Windows.

(T7)


Activities
Terminal
Nov 22 22:17
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox: ~
sdn@sdn-VirtualBox: ~/Desktop/project/src
sdn@sdn-VirtualBox:~/Desktop/project/src$ python3 data_gathering.py 10.0.2.15 8086 telegraf
Finished generating a class 10.0.2.15 training dataset!
sdn@sdn-VirtualBox:~/Desktop/project/src$ ls -la
total 36
drwxrwxr-x 2 sdn sdn 4096 Nov 22 22:17 .
drwxrwxr-x 7 sdn sdn 4096 Nov 20 17:31 ..
-rwxrwxr-x 1 sdn sdn 1434 Nov 21 19:51 data_gathering.py
-rwxrwxr-x 1 sdn sdn 1439 Nov 20 16:18 ddos.py
-rw-rw-r-- 1 sdn sdn 1800 Nov 22 22:17 ICMP_data_class_10.0.2.15.csv
-rwxrwxr-x 1 sdn sdn 1347 Nov 20 16:18 normal.py
-rwxrwxr-x 1 sdn sdn 2958 Nov 20 21:48 scenario_basic.py
-rwxrwxr-x 1 sdn sdn 4357 Nov 21 20:09 traffic_classifier.py
sdn@sdn-VirtualBox:~/Desktop/project/src$
Activate Windows
Go to Settings to activate Windows.

(T8)



(T9)

CHAPTER 10

FUTURE SCOPE

- We can possibly try to mitigate the DDOS Attack after its classification being recognized by the SVM .
- We could try to automate some of the time taking manual processes using AI & ML Algorithm.
- In future work, we need to assure that we block the live traffic of DDoS Attack instead of transferring to another system.
- We also should try to add Graphical User Interface (GUI) to make it easily accessible for non-technical users.
- We also could try to add a proper result analysis feature to analyse the traffic in different scenarios.
- A feature to enhance safety and security of the software can be embedded.
- In the coming days, we also should work on decreasing the risks of DDoS/DoS Attack and minimise the damages done by the same.


CONCLUSION

The results of the investigation have led to the Smart Detection system, an online method for detecting DoS/DDoS attacks. Based on samples collected by the Telegraf and recorded in the InfluxDB directly from network devices, the programme employs the Support Vector Machine (SVM) to categorise network traffic. To calibrate and assess system performance, a number of experiments were conducted. The proposed method has enhanced performance when compared to certain current and pertinent methodologies that are accessible in the literature, according to the results.

The system requires several enhancements, such as a higher hit rate among attack classes and an automated parameter calibration method that maximises the rate of attack detection, even if it has produced considerable results within its purview.

Future work will examine DDoS assaults caused by services like Heartbleed and web brute force attacks, improve multiple-class categorization, configure the system automatically, provide techniques for correlating triggered alarms, and create preventive measures.

REFERENCES

- J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *ACM Special Interest Group on Data Communication*, vol. 34, no. 2, pp. 39–53, 2004. View at: Google Scholar
- S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013. View at: Publisher Site | Google Scholar
- R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda and Ligia Rodrigues Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, 2014, pp. 1-6. doi: 10.1109/ColComCon.2014.6860404 Paper
- Kokila RT, S. Thamarai Selvi and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," 2014 Sixth International Conference on Advanced Computing (ICoAC), Chennai, 2014, pp. 205210. Paper
- H. Kim and N. Feamster, "Improving network management with software defined networking", *IEEE Communications Mag.*, vol. 51, no. 2, pp. 114-119, 2013.
- "Software-Defined Networking: The New Norm for Networks", *White Paper Open Networking Foundation (ONF)*, Apr. 2012, [online] Available: https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdnnewnorm.pdf.
- W. Feng, Q. Zhang, G. Hu and J. Huang, "Mining network data for intrusion detection through combining SVMs with ant colony networks", *Future Generation Computer Systems*, vol. 37, pp. 127-140, July 2014.
- P. Arun, Raj Kumar and S. Selvakumar, "Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems", *Computer Communications*, vol. 36, pp. 303-319.
- S. Scott-Hayward, G. O'Callaghan and S. Sezer, "SDN Security: A Survey", *IEEE SDN for Future Networks and Services*, Nov. 2013.
- G. Madazarov, D. Gjorgevikj and I. Chorbev, "A Multi-class SVM classifier utilizing Binary Decision Tree", *Informatica*, pp. 233-241, 2009.
- K. Haldar and B. K. Mishra, “A mathematical model for a distributed attack on targeted resources in a computer network,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 9, pp. 3149–3160, Sep. 2014. View at: Publisher Site | Google Scholar
- W. T. Hou and H. Wang, “Study of a mathematical model for a distributed attack on targeted resources in a computer network,” *Journal of Natural Science of Heilongjiang University*, vol. 3, pp. 315–321, 2016.
- J. Ren, C. Zhang, and Q. Hao, “A theoretical method to evaluate honeynet potency,” *Future Generation Computer Systems*, vol. 116, pp. 76–85, 2021. View at: Publisher Site | Google Scholar

- B. Fei and J. Liu, "Binary Tree of SVM: A new fast Multiclass Training and Classification Algorithm", *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 696-704, May 2006.
- C. J. C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining Knowl. Discov.*, vol. 2, no. 2, pp. 1-47, 1998.
- S. Cheong, S. H. Oh and S.-Y. Lee, "Support vector machines with binary tree architecture for multi-class classification", *Neural Info. Process.-Lett. Rev.*, vol. 2, no. 3, Mar. 2004.
- A. Shilton, M. Palaniswami, D. Ralph and A. C. Tsoi, "Incremental training of support vector machines", *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 114-131, Jan. 2005.
- S. Newman, “Under the radar: the danger of stealthy DDoS attacks,” *Network Security*, vol. 2019, no. 2, pp. 18-19, 2019. View at: Publisher Site | Google Scholar
- N. Vlajic and D. Zhou, “IoT as a land of opportunity for DDoS hackers,” *Computer*, vol. 51, no. 7, pp. 26–34, 2018. View at: Publisher Site | Google Scholar
- Y. Wang, L. Liu, B. Sun, and Y. Li, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” in *Proceedings of the 2015 6th IEEE International Conference on Software Engineering and Service Science, ICSESS 2015*, pp. 1034–1037, Kochi, India, 2015. View at: Google Scholar R. Singh, A. Prasad, R. Moven, and H. Samra, “Denial of service attack in wireless data network: a survey. devices for integrated circuit,” in *Proceedings of the 2017 Devices for Integrated Circuit (DevIC)*, pp. 23-24, Kalyani, India, March 2017. View at: Publisher Site | Google Scholar
- A. Praseed and P. Santhi Thilagam, “DDoS attacks at the application layer: challenges and research perspectives for safeguarding web applications,” *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 661–685, 2019. View at: Publisher Site | Google Scholar
- T. V. Phan and M. Park, “Efficient distributed denial-of-service attack defense in sdnbased cloud,” *IEEE Access*, vol. 7, pp. 18701–18714, 2019. View at: Publisher Site | Google Scholar
- B. Pfaff, J. Pettit, T. Koponen et al., “The design and implementation of Open vSwitch,” in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pp. 117–130, USENIX Association, Oakland, CA, USA, 2015, https://www.usenix.org/conference/nsdi15/technicalsessions/presentation/pfaff. View at: Google Scholar
- A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Computers and Security*, vol. 31, no. 3, pp. 357–374, 2012. View at: Publisher Site | Google Scholar
- S. Sanfilippo, N. Jombart, D. Ducamp, Y. Berthier, and S. Aubert, “Hping,” 2014, http://www.hping.org. View at: Google Scholar
- S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, “Intelligent feature selection and classification techniques for intrusion detection in networks: a survey,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 271, 2013,