Honey-Pot Detection System

(CA550 Small Industrial Training/ Small Project /MOOC)

A Report

Submitted in partial fulfillment of the requirements for the award of the Degree of

Master of Computer Applications

BY

Akhil Singh (MCA/40022/23)

Abhishek Kumar Gupta (MCA/40055/23)

Parwez Musharraf (MCA/40061/23)

BIRLA INSTITUTE OF TECHNOLOGY

MESRA-835215, RANCHI

(2024)

APPROVAL OF THE GUIDE

Recommended that the project report entitled **"Honeypot Detection System"** presented by **Abhishek Kumar Gupta, Akhil Singh, Parwez Musharraf** under my supervision and guidance be accepted as fulfilling this part of the requirements for the award of Degree of Master of Computer Applications**.** To the best of my knowledge, the content of this report did not form a basis for the award of any previous degree to anyone else.

Date: ____________

Name & Signature of the Guide
Dept. of Computer Science and Engg.
Birla Institute of Technology Mesra,Ranchi

DECLARATION CERTIFICATE

I certify that

a) The work contained in the report is original and has been done by myself under the general supervision of my supervisor.

b) The work has not been submitted to any other Institute for any other degree or diploma.

c) I have followed the guidelines provided by the Institute in writing the report.

d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Akhil Singh (MCA/40022/23)
Abhishek Kumar Gupta (MCA/40055/23)
Parwez Musharraf (MCA/40061/23)

CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this project entitled **“Honey-Pot Detection System”**, is carried out by **Abhishek Kumar Gupta (MCA/40055/23), Akhil Singh(MCA/40022/23), Parwez Musharraf(MCA/40061/23)**has been approved for the degree of Master of Computer Applications of Birla Institute of Technology, Mesra, Ranchi.

Date:

Place:

Internal Examiner **External Examiner**

(Chairman)
Head of the Department

Abstract

It enables the users to acquire services over the Internet on demand.

promote seamless access from nearly anywhere. Despite numerous advantages the system boasts, though.

Remains vulnerable to a host of security threats. DDoS attacks

represent some of the most significant threats, hindering accessibility to cloud resources and revealing

Such weaknesses allow attackers to install advanced threats, including

Malware injection, malicious packing, Virtual machine escape, DDoS attack

assaults.

Although various models have been designed to identify these potential risks within cloud environments,

They fail on specific points most of the times. Overcoming this, we come up with a new DDoS.

An attack prediction framework based on the HBO algorithm for

Feature selection using a Bi-Directional Long Short-Term Memory (Bi-LSTM) classifier.

The proposed approach analyzes input features determined from a dataset related to DDoS attacks through

It has a preprocessing stage before processing, Bayesian methods, and Z-score normalization techniques.

After pre-processing, the Honey Badger Optimization algorithm selects the most relevant

features minimize the Mean Squared Error, which in effect provides optimal feature selection. These

The refined features are then passed to the Bi-LSTM classifier for the accurate DDoS attack

prediction. Comparison of the proposed method with current methods including LSTM, DNN, DBN, and ANN

It illustrates the improved performance of the Bi-LSTM-based methodology. The model obtained a

A truly accuracy of 97%, sensitivity of 95%, specificity of 90%, precision of 94%, and an

Its error rate was only 3%.

This research puts to rest the fact that the system framework depicted is a great identification method.

Prevention and mitigation of DDoS attacks in cloud computing environments provide better security.

Reliability.

The domain of cloud computing is constantly evolving; however, its basic offerings—Software as a Service (SaaS), Web

Services, Utility Computing, and Platform as a Service (PaaS) are services.

discrete, unrelated units. Even though progress along the way toward holistic

The integration of these systems currently requires separate configuration. Among the necessary

Denial of Service and XML are some threats to cloud infrastructure.

Based DoS attacks. The attack vectors are relatively simple to execute, making them an

An attractive option for the bad guys, and a big challenge for cybersecurity.

Experts were delegated to mitigate their effects.

This paper employs simulation to analyze real world scenarios that occur during HTTP and XML-based DoS attacks.

Assess and analyze potential mitigations. In order to counter such threats, we recommend an innovative

This approach is called Cloud TraceBack (CTB), tracing back to the roots of these attacks.

efficiently. We further present a cutting-edge detection and mitigation system based on a

The back-propagation neural network, known as Cloud Protector, is a particular system

Trained to detect and block hostile traffic associated with this form of attacks.

The results from our experiments show that the proposed solution indeed eliminated a

It also formed a substantial portion of attack traffic and rapidly pinpointed the source of malicious activities.

This methodology emphasizes the ability to integrate traceback systems with artificial intelligence.

Filtering to make the cloud environment more robust against emerging DoS threats.

ACKNOWLEDGEMENT

I would like to express my profound gratitude to my project guide, **Dr. S. Biswas** for his guidance and support during my report work. I benefited greatly by working under his guidance. It was his effort for which I am able to develop a detailed insight on this subject and special interest to study further. His encouragement motivation and support has been invaluable throughout my studies at BIT, Mesra, Ranchi.

I convey my sincere gratitude to **Dr. S. Biswas** Dept. of CSE, BIT, Mesra, Ranchi, for providing me various facilities needed to complete my project work. I would also like to thank all the faculty members of MCA department who have directly or indirectly helped during the course of the study. I would also like to thank all the staff (technical and non-technical) and my friends at BIT, Mesra, Ranchi who have helped me greatly during the course.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout the years of my study. This accomplishment would not have been possible without them.

My apologies and heartful gratitude to all who have assisted me yet have not been acknowledged by name.

Thank you.

DATE:

Akhil Singh (MCA/40022/23)
Abhishek Kumar Gupta (MCA/40055/23)
Parwez Musharraf (MCA/40061/23)

CONTENTS

INTRODUCTION

Cloud computing offers scalable, flexible, and cost-effective solutions for individuals and organizations through on-demand access to resources. This model has transformed data management and business operations, with widespread adoption across industries. However, security issues, particularly Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, remain a major concern, as they threaten the availability and reliability of cloud services. These attacks exploit vulnerabilities of protocols such as HTTP and XML, exhaust the cloud resources, service downtimes, and possibilities of data breaches.

Especially challenging for cloud environments, distributed attacks in DDoS prove challenging because it is challenging to differentiate between malicious and legitimate traffic. HTTP-based attacks will make multiple simultaneous requests to a server, consuming resources; meanwhile, XML-based attacks can exploit malicious payloads to exhaust the computing power. These types of threats require advanced detection and mitigation techniques to protect cloud platforms such as AWS and Google Cloud.

This is the world's first project introducing a new methodology, combining feature selection that uses the Honey Badger Optimization algorithm with classification made through a Bi-Directional Long Short-Term Memory network. The result is to enhance accuracy and response speed in detecting attacks as it learns temporal patterns and optimizes features. These are the CTB, or the Cloud TraceBack, which labels requests in order to detect and prevent malicious origins, and the Cloud Protector, a neural back-propagation network which has the capability of detecting attack traffic and filtering it out. This delivers HTTP and XML-based DoS mitigation with improved resource availability and service reliability.

The focus has been on dynamic traceback and neural network-based defense for dealing with the evolving nature of threats in cyber, offering scalable and effective protection for cloud computing environments. The contributions indicate that novel security mechanisms are crucial in the development of secure and resilient adoption of clouds.

Objective

The key purpose of a honeypot detection system is to recognize and uncover honeypots embedded within a network or infrastructure. By doing so, attackers can avoid interacting with these decoys, while security teams gain valuable insights into evasion techniques used by threat actors. Below are the detailed objectives:

1. Recognizing Honeypots

Identify decoy systems or environments designed to lure attackers and monitor their activities.

2. Avoiding Deceptive Defenses

Equip attackers with methods to evade honeypots, protecting their tools and tactics from being revealed.

3. Assessing Security Mechanisms

Evaluate the efficiency and resilience of deployed honeypot systems against potential detection.

4. Enhancing Defensive Strategies

Provide defenders with knowledge about detection methods, allowing them to refine their honeypots for better stealth and effectiveness.

5. Aiding Ethical Hacking Efforts

Support penetration testers and red teams in locating and avoiding defensive honeypots during security assessments.

Overall, honeypot detection systems are designed to uncover decoy systems, thereby enhancing both the defensive and offensive capabilities of cybersecurity teams

Project Categories

Cyber Security

Cybersecurity is the discipline of safeguarding computer systems, networks, and applications against unauthorized access, disruptions, and malicious attacks. These attacks often aim to steal, alter, or destroy sensitive data, disrupt normal operations, or extort money. Below is a comprehensive guide to the fundamental aspects of cybersecurity:

---

1. Core Principles

- **Confidentiality, Integrity, and Availability (CIA Triad)**:
    - **Confidentiality**: Ensures that sensitive data is only accessible to authorized personnel.
    - **Integrity**: Protects information from unauthorized changes, maintaining accuracy and reliability.
    - **Availability**: Guarantees that systems and data are accessible to authorized users whenever needed.
- **Authentication and Authorization**:
    - **Authentication**: Confirms a user's or system's identity, typically using credentials like passwords or biometric scans.
    - **Authorization**: Determines access levels and permissions for users or systems.
- **Threats vs. Vulnerabilities**:
    - **Threats**: Potential dangers, such as malware, cybercriminals, or insider threats.
    - **Vulnerabilities**: System weaknesses that can be exploited by threats to cause harm.

---

2. Major Cybersecurity Threats

- **Malware**: Harmful software like ransomware, viruses, spyware, and worms designed to damage or infiltrate systems.
- **Phishing Attacks**: Deceptive methods, often via email, used to steal sensitive information such as login credentials.

- **Man-in-the-Middle (MITM) Attacks**: Eavesdropping on or intercepting communications between two parties.
- **Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks**: Overloading a network or server, rendering it unusable.
- **Zero-Day Exploits**: Targeting vulnerabilities unknown to the software vendor or user.

---

3. Security Measures to Mitigate Risks

- **Firewalls**: Act as protective barriers, monitoring and controlling incoming and outgoing traffic based on security rules.
- **Antivirus and Anti-Malware Software**: Detect, quarantine, and remove malicious programs.
- **Data Encryption**: Converts sensitive data into unreadable formats without proper decryption keys.
- **Multi-Factor Authentication (MFA)**: Enhances security by requiring multiple verification steps.
- **Software Patches and Updates**: Fix known vulnerabilities and improve system defenses.

---

4. Cyber Hygiene and Best Practices

- Use **strong and unique passwords** for each account and update them regularly.
- Avoid interacting with **unknown links, attachments, or emails** that seem suspicious.
- Always access websites through secure connections (**HTTPS**).
- Maintain **regular backups** of critical data in case of ransomware attacks.
- Stay informed about the latest **social engineering techniques** and how to avoid falling victim to them.

---

5. Cybersecurity Frameworks and Regulations

- **ISO/IEC 27001**: Provides standards for implementing an effective information security management system.

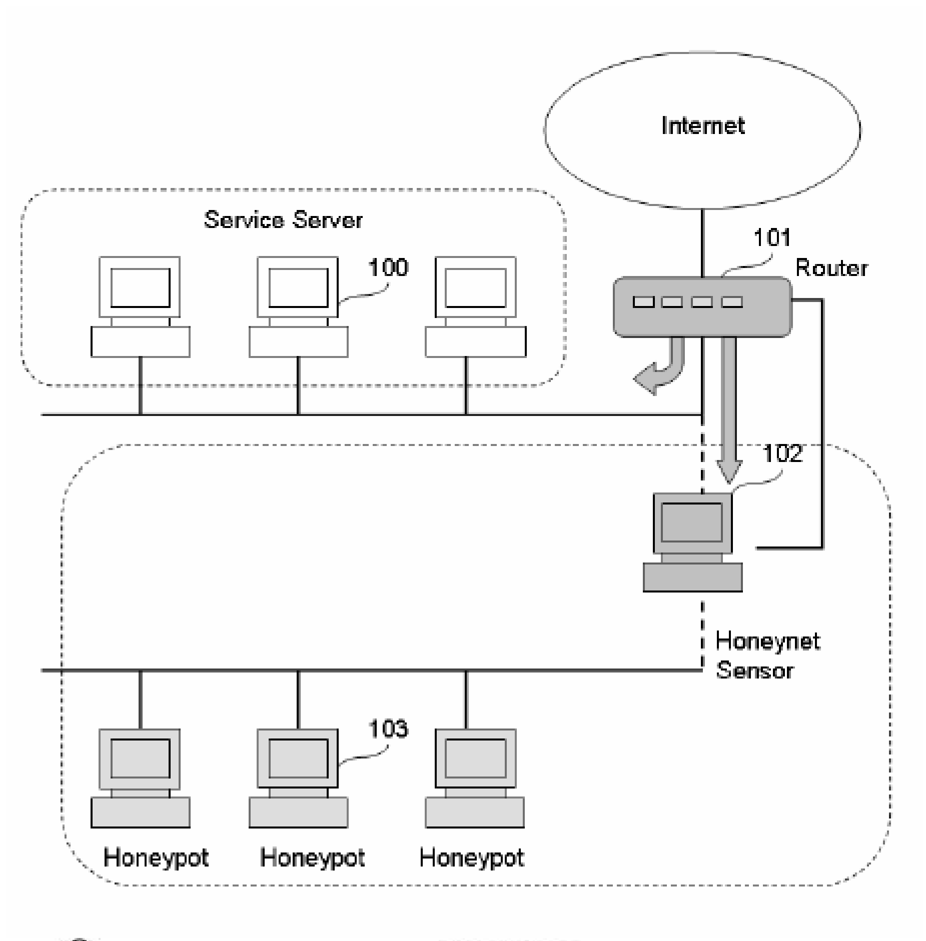
- **NIST Cybersecurity Framework**: Offers best practices to identify, protect, detect, respond to, and recover from cyber threats.
- **GDPR (General Data Protection Regulation)** and **CCPA (California Consumer Privacy Act)**: Focus on data privacy and protection compliance.

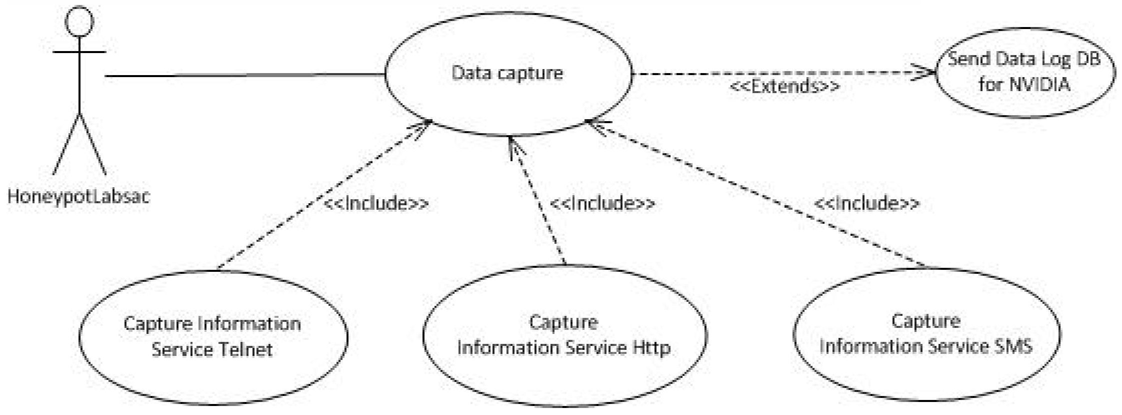
---

6. Tools for Learning and Practice

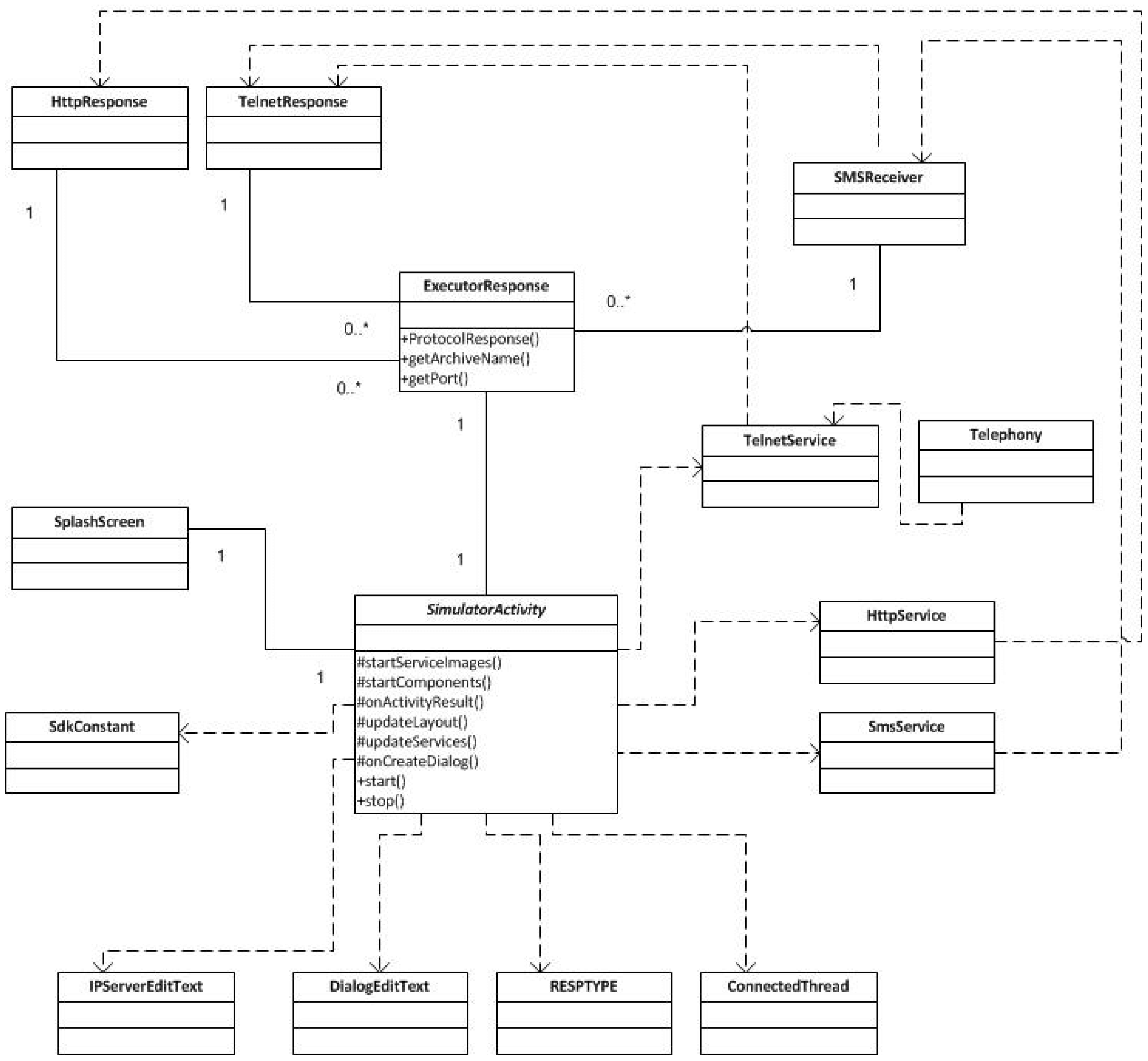
- **Ethical Hacking Platforms**: Engage in simulated hacking challenges through sites like **TryHackMe** or **Hack The Box**.
- **Kali Linux**: A widely-used operating system for penetration testing and digital forensics.
- **Wireshark**: Analyzes network protocols to diagnose issues and detect suspicious activity.
- **Metasploit Framework**: A robust toolset for penetration testing and finding system vulnerabilities.

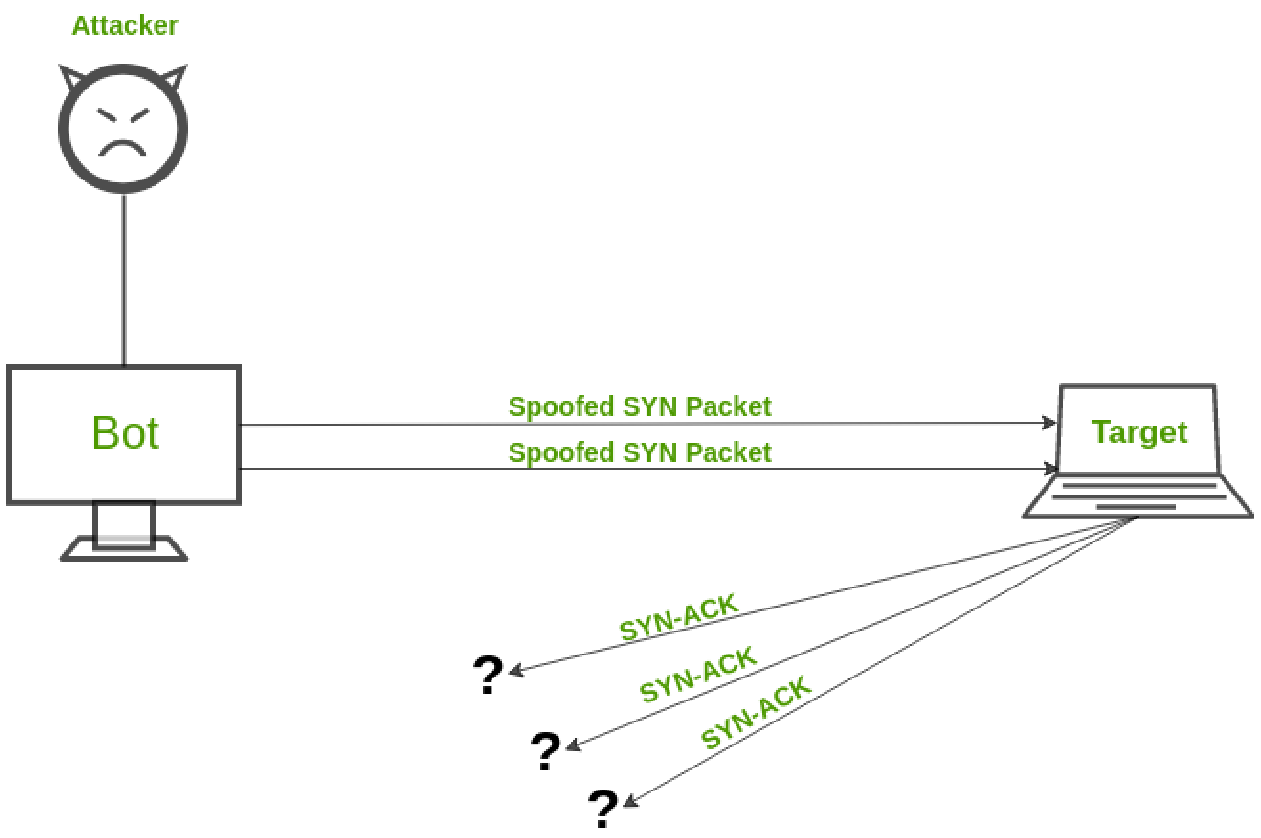
UML DIAGRAM

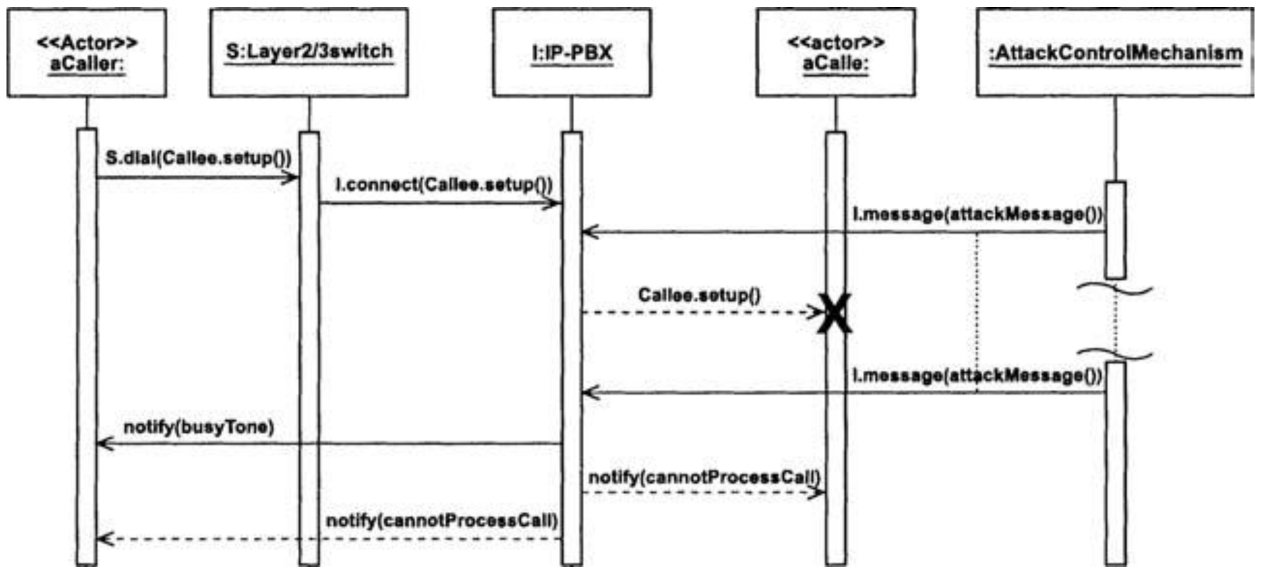

Internet
Service Server
100
101
Router
102
Honeynet Sensor
103
Honeypot
Honeypot
Honeypot

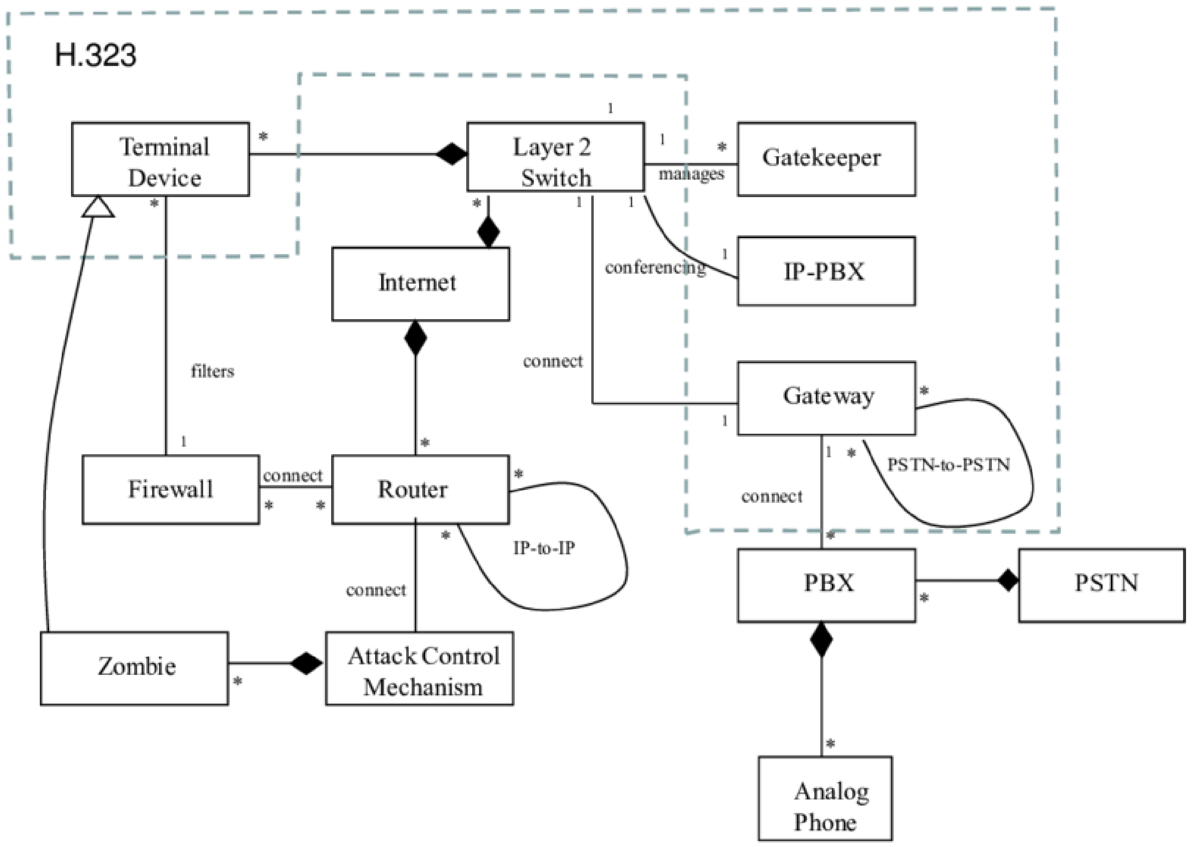

HoneypotLabsac
Data capture
<<Extends>>
Send Data Log DB for NVIDIA
<<Include>>
<<Include>>
<<Include>>
Capture Information Service Telnet
Capture Information Service Http
Capture Information Service SMS

Class Diagram of HoneypotLabsac




Attacker
Bot
Spoofed SYN Packet
Spoofed SYN Packet
Target
SYN-ACK
SYN-ACK
SYN-ACK
?
?
?


<<Actor>> aCaller:
S:Layer2/3switch
I:IP-PBX
<<actor>> aCalle:
:AttackControlMechanism
S.dial(Callee.setup())
I.connect(Callee.setup())
I.message(attackMessage())
Callee.setup()
I.message(attackMessage())
notify(busyTone)
notify(cannotProcessCall)
notify(cannotProcessCall)


H.323
Terminal Device
Layer 2 Switch
Gatekeeper
manages
IP-PBX
conferencing
Internet
filters
connect
Gateway
PSTN-to-PSTN
Firewall
Router
IP-to-IP
PBX
PSTN
Zombie
Attack Control Mechanism
Analog Phone

Tools, Hardware and Software Requirement

- Hardware :- Laptop/PC
- Specification :- Ryzen 5, 16GB RAM , 220 Gb SSD
- Software :- Oracle Virtual Box
- OS :- Kali Linux, Windows 11
- Programming Language :- Ruby Programming Language

Related Work

➢ SOTA (Service-oriented traceback architecture)

The Service-Oriented Traceback Architecture (SOTA) is a versatile web security application designed to be neutral across products (Chonka et al., 2008a, 2008b, 2009). Its core purpose is to adopt a Service-Oriented Architecture (SOA) strategy for traceback mechanisms, addressing the challenge of identifying forged message origins. This is particularly relevant as attacks like XML-based DoS (X-DoS) and Distributed XML-based DoS (DX-DoS) often obscure the attacker's identity. SOTA builds on the Deterministic Packet Marking (DPM) algorithm (Belenky and Ansari, 2003). DPM utilizes the IP header's ID field and reserved flag for marking packets as they enter the network at edge ingress routers. These markings remain intact across network traversal, focusing only on incoming packets while excluding outgoing ones.

SOTA adapts DPM's concept to web services by embedding a Service-Oriented Traceback Mark (SOTM) within SOAP messages. If existing security measures like WS-Security are in place, SOTM takes the place of tokens used for client identification. This tag carries the actual source message identification and is incorporated into the SOAP header as a lightweight XML element. Upon detecting an X-DoS or DX-DoS attack, SOTM facilitates the identification of the true origin of malicious messages.

It is worth noting that SOTA does not directly mitigate X-DoS or DX-DoS traffic; this function is handled by complementary defence mechanisms like Cloud Protector. Instead, SOTA addresses two primary attack objectives: exploiting known vulnerabilities to overwhelm web servers with traffic and concealing attacker identities. This focus aligns SOTA with traceback methodologies like Probability Packet Marking (PPM) (Savage et al., 2001) and DPM, making it an essential tool for identifying attackers' sources.

Key reasons why a framework like SOTA is critical in cloud environments include:

- Existing web security frameworks are inadequate for addressing sophisticated X-DoS and DX-DoS threats. Research, such as that by Jensen et al. (2007), highlights how WS-Security can inadvertently enable X-DoS attacks.
- With the advent of IPv6, traditional IP traceback systems face challenges due to protocol changes like IPSec and altered packet header formats that remove fields critical for traceback.
- SOTA operates within the bounds of IP protocols by embedding information into the SOAP headers instead of modifying IP packets.
- Its SOA-based design ensures compatibility across diverse grid systems, making it deployable in a wide range of cloud computing environments.

These attributes position SOTA as a robust solution for enhancing security in increasingly complex and interconnected cloud architectures.

- **XML-based denial of service (X-DoS) attacks**

A Denial of Service (DoS) attack occurs when an adversary deliberately disrupts access to resources, depriving legitimate users of their intended functionality (Rogers, 2009). XML-based Denial of Service (X-DoS) attacks, as outlined by Padmanabhuni et al. (2007), involve flooding a network with excessive XML messages instead of traditional packet-based traffic. This type of attack significantly hampers legitimate access to network services and can overload a server with XML requests, leading to resource exhaustion and reduced availability of web services. Additionally, attackers may modify the content of these XML messages to exploit vulnerabilities, potentially causing server crashes, as demonstrated by Jensen et al. (2007).

In an evolved form, X-DoS can be incorporated into a Distributed Denial of Service (DDoS) model, referred to as Distributed XML-based Denial of Service (DX-DoS). In this scenario, multiple compromised systems are leveraged to launch a coordinated attack, generating a high volume of malicious XML traffic directed at the target server. Although such attacks have not been extensively documented, their potential impact on cloud infrastructures is profound and represents a looming security challenge. Future advancements in cloud computing will likely necessitate robust defenses against such complex and distributed threats.

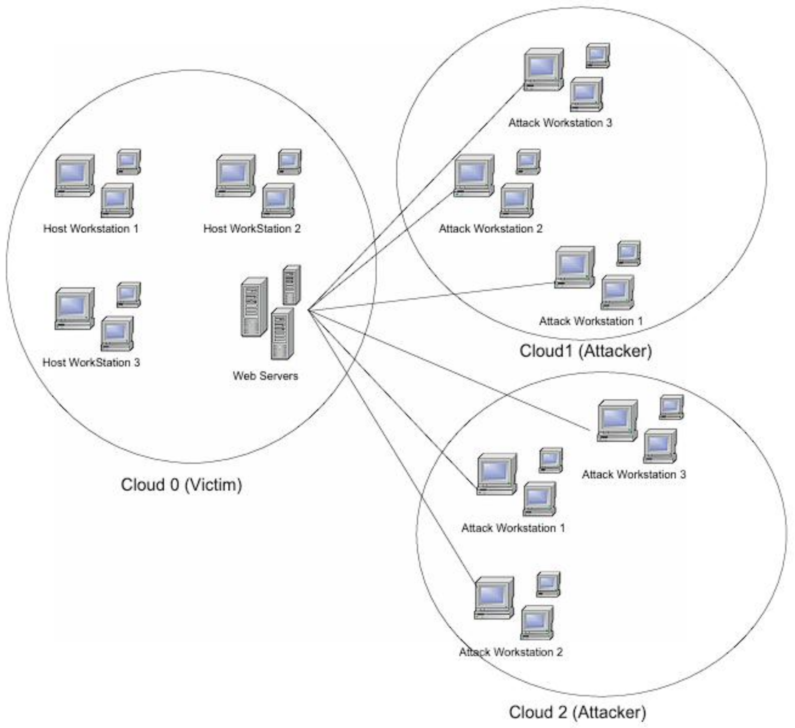


Fig. 1. Distributed XML-based Denial of Service attack, where an attacker has taken control of 2 cloud networks and starts to sending huge amounts XML-based message to Cloud 0 Web Server.

➢ Cloud traceback for cloud computing

➢ <u>Introduction</u>

Attackers often succeed in exploiting the internet’s inherent limitations, including its reliance on finite and consumable resources. These resources, such as bandwidth, processing power, and storage capacity, are particularly constrained in cloud computing environments. Since cloud systems must maintain high service quality, they are especially vulnerable to exhaustion when overwhelmed by large numbers of requests. This vulnerability provides attackers with an opportunity to deploy attacks like X-DoS or DX-DoS.

For instance, an attacker could utilize multiple virtual machines, each running numerous browser instances, to generate a sustained flood of requests targeting a victim’s web server. Over time, this activity can overload the server, depleting its resources and impacting its ability to serve legitimate users.

In a Distributed XML-based Denial of Service (DX-DoS) attack, the scenario is escalated through the coordination of multiple compromised systems, often referred to as agents or zombies. These systems collectively generate a barrage of oversized XML messages directed at the target server. The attack can cause the server to crash either by forcing it to process excessively large messages or by creating severe communication bottlenecks through network congestion.

To mitigate such threats, the deployment of systems like Cloud Traceback becomes essential. Figure 2 illustrates how such a defence mechanism can be strategically placed within a cloud infrastructure to safeguard against attacks like X-DoS, DX-DoS, and HTTP-based DoS (H-DoS), ensuring enhanced resilience and system stability.

➢ <u>Cloud traceback description</u>

Cloud Traceback (CTB) can be used in either a network structure, such as a LAN, or a grid network structure. CTB is made within a virtual machine to make placement within the cloud network compatible, flexible and scalable whilst remains a SOA security product.

➢ <u>CTB placement within cloud system infrastructure</u>

Cloud Traceback (CTB) is strategically positioned at edge routers, ensuring proximity to the source end of the cloud network. In the absence of security mechanisms for web services, systems are highly susceptible to attacks. By being deployed ahead of the web server, as depicted in Fig. 2, CTB mitigates vulnerabilities by embedding a Cloud Traceback Mark (CTBM) tag into the CTB header. This integration is achieved by linking the Web Service Definition Language (WSDL) directly to the CTB rather than to the web server.

This setup ensures that all incoming service requests are routed through the CTB for marking before reaching the web server. By obfuscating the service provider's address, CTB minimizes the risk of direct attacks. In the event of a successful attack or service disruption, the CTBM tag allows the victim to reconstruct and trace the attack's origin, enabling a swift response.

During an attack, the malicious client initiates a web service request to the CTB. The CTB processes the request and forwards it to the web server. Subsequently, the attacker sends a SOAP request message crafted using the WSDL-generated service description. Upon receiving this SOAP message, the Service-Oriented Traceback Architecture (SOTA) embeds a Service-Oriented Traceback Mark (SOTM) in the message header. It is assumed that WS-Security replaces the "wsse" username tag with its security tag.

After the CTBM is added, the SOAP message is directed to the web server. If an attack is detected, the CTBM tag enables traceback reconstruction, helping to identify the source and initiating traffic filtering. Conversely, if the message is legitimate, it proceeds to the request handler for processing. Following this, the web server generates a SOAP response and sends it back to the client as an HTTP response. Notably, CTB does not interfere with outgoing messages or response requests, maintaining the integrity of normal communications.

- **Cloud Protector**

The Cloud Protector utilizes a backpropagation neural network (NN) designed specifically to identify and filter X-DoS attack messages. This neural network is structured as a collection of interconnected nodes organized into input, hidden, and output layers. Each connection between nodes carries an assigned weight that influences the network's processing capability.

At the heart of the neural network lies the Threshold Logic Unit (TLU), which serves as the decision-making component. The TLU processes input values by applying associated weights to these inputs, generating a weighted sum. This sum is then compared against a predefined threshold. If the result surpasses this threshold, the TLU activates, allowing the network to determine whether the incoming message aligns with patterns characteristic of X-DoS attacks. This intelligent filtering system ensures precise detection and enhances the network’s defensive capabilities.

- **Cloud traceback approach to SOA**

Cloud Traceback (CTB) incorporates several fundamental features and attributes characteristic of the service model, as outlined by He et al. (2005) and Ye and Singh (2007). These defining traits are:

- **Loosely Coupled Architecture**: CTB is built using XML-based technology, allowing it to operate seamlessly across various platforms, independent of the underlying programming languages.

- **Message-Oriented Communication**: All interactions between clients, CTB, and service providers are facilitated through message exchanges, enabling efficient and standardized communication.
- **Dynamic Service Discovery**: By integrating the Web Service Definition Language (WSDL) directly into CTB, services are made publicly accessible. This enables clients to connect with CTB anytime via the internet to utilize available service offerings.
- **Real-Time Operation with Late Binding**: Both CTB and the associated service provider function in real-time, enabling clients to interact with services on-demand, regardless of their location or timing.
- **Policy-Driven Behavior**: Plans are underway to establish a CTB-specific policy framework that aligns with the principles of WS-Security Policy. This policy will govern message tagging and processing procedures to enhance security and traceability.

Functionally, CTB serves as a service broker within a Service-Oriented Architecture (SOA) framework, acting as a centralized repository for service descriptions like WSDL or Universal Description, Discovery, and Integration (UDDI). This role ensures that CTB efficiently mediates between service providers and clients, promoting interoperability and streamlined access to resources (as illustrated in Fig. 3).

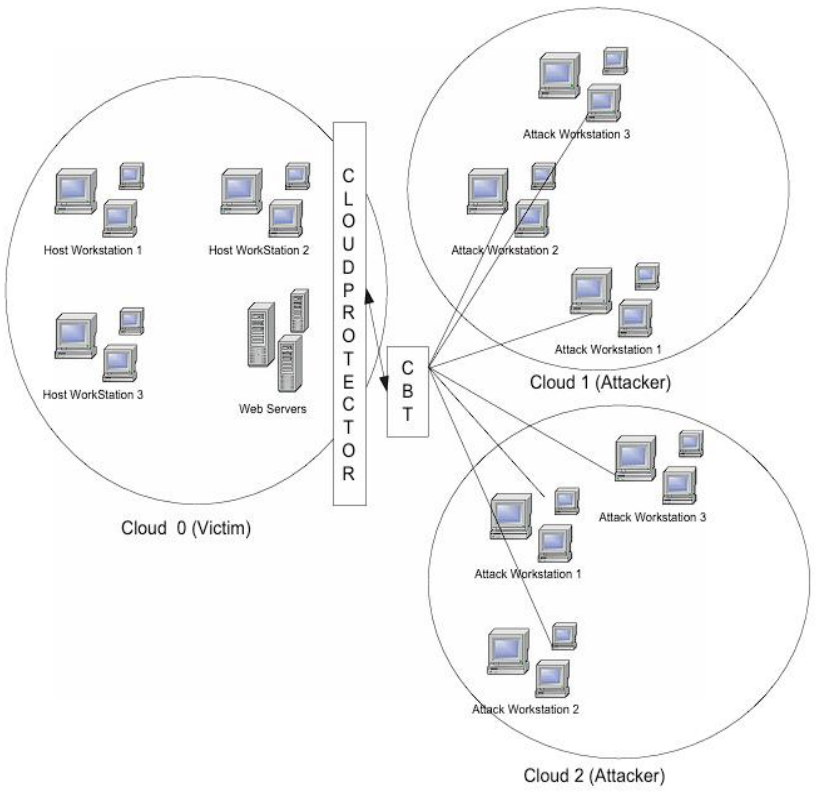


Fig. 2. Distributed XML-based Denial of Service attack, where CTB and Cloud Protector are located just between the each could Web Service, in order to detect and filter X-DoS attacks.

➢ CTB’s algebraic approach to determining path and reconstruction

➢ Assumptions

The initial set of experiments was conducted based on the following assumptions:

- An attacker could potentially command a large number of client machines, which might be geographically dispersed across the internet.
- Attackers are likely aware that their activities are being monitored and traced.
- The CTB system requires only a minimal number of incoming messages to initiate its traceback process.
- The CTB framework itself remains secure and uncompromised by the attackers.
- The web service provider operates with limited system resources, making it vulnerable to resource exhaustion.
- The client interactions utilize SOAP headers for message communication and identification.
- The real source of any message is identified as the edge router through which the traffic originates.

➢ Algebraic coding of path

The approach taken by CTB to encode the traceback information follows the same lines as Dean (2002), in which a polynomial is used within the tag so that the reconstruction of the path the message has taken can take place. With the use of a polynomial f(x) of degree d in the prime field GF(P), we can then use f(x) given that f(x) evaluated at (d+1) unique points. Let fP(x)¼M1,M2,y,Mn.

3.5.3. Path reconstruction by CTB at cloud victim end

As depicted in Fig. 4, when a message reaches the Cloud Victim (CV), the Cloud Traceback (CTB) system initiates its reconstruction process to determine the source of the message. Here, kk represents the number of attack paths utilized by the attacker, and LL denotes the expected length of each attack path. For simplicity, this study assumes that the actual attack paths are approximately equal in length to LL.

According to Dean (2002), the computational complexity of reconstructing these paths at the CV is O(Lk2)O(Lk^2). The quadratic term reflects the brute-force effort required during traceback. Notably, this complexity is less than the O(Lk8)O(Lk^8) identified in Dean’s earlier work. The difference arises because SOAP messages can embed more detailed information in their tags compared to the limited capacity of IP packet headers.

One significant advantage of SOAP-based traceback is the potential to include a distance field, derived from the packet’s time-to-live (TTL) value. This feature aids in simplifying the traceback process by allowing the CV to sample messages with TTL values within a specific range (ww). As long as the attacker remains within this range and router resets have not occurred, the messages can be presumed legitimate.

To verify whether an attacker operates within this distance, the Bleichenbacher–Nguyen reconstruction algorithm is employed. Given all possible values x1,x2,...,xnx_1, x_2, ..., x_n, a set SiS_i of size jj is identified. This set is used to determine all polynomials aa of degree dd or less, where each SiS_i contains distinct yy-values such that (xi,y)(x_i, y) pairs occur within sampled messages. The polynomial aa corresponding to the attack path is then derived based on the attack message's trajectory.

For practical application, the CV selects a sample of NN messages and evaluates each xix_i by choosing a subset SiS_i of size mm from the (xi,y)(x_i, y) pairs. If the total yy-values exceed mm, the CV identifies which mm values likely belong to attack messages along a particular path.

To account for false positives (polynomials unrelated to actual attack paths), the algorithm operates under statistical thresholds to minimize errors in the reconstruction. This method ensures an efficient and accurate traceback process for identifying the source of malicious traffic.

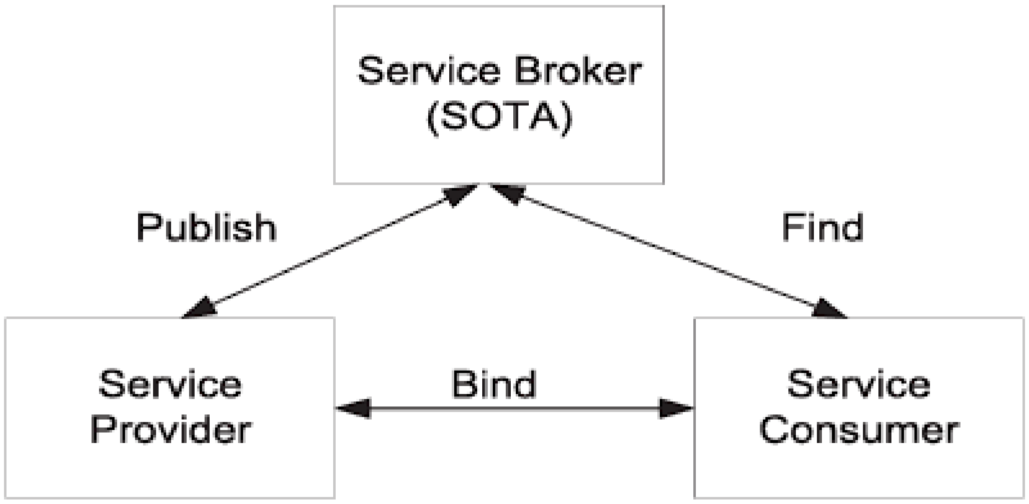


Fig. 3. SOA diagram with SOTA as the Service Broker.

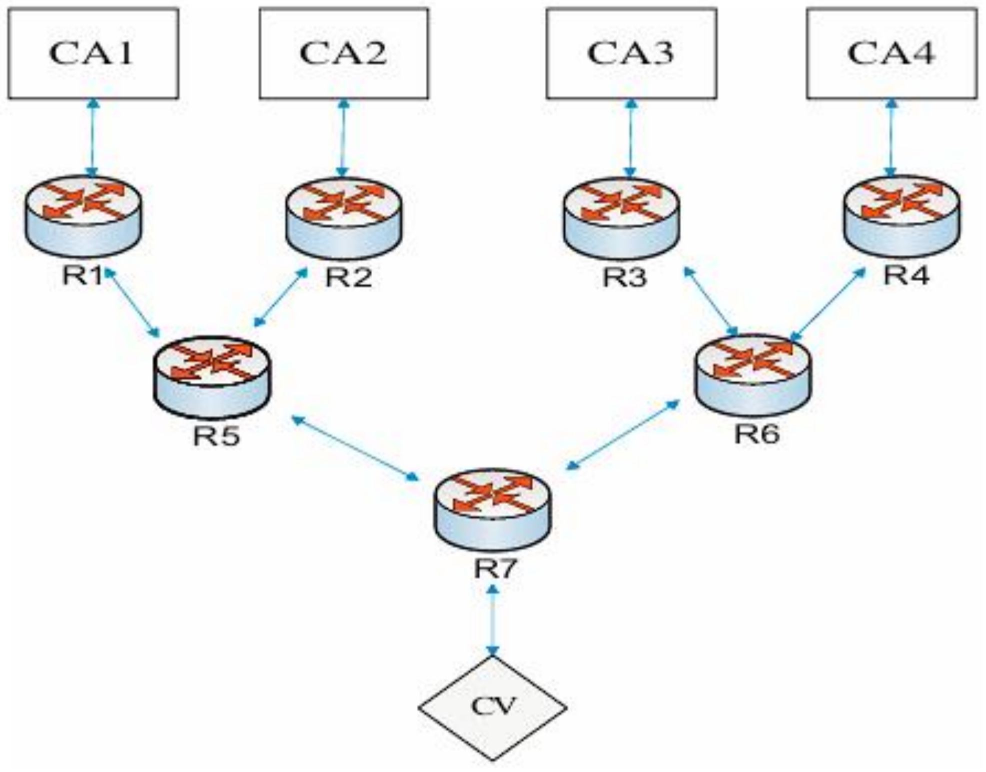


Fig. 4. True path from the cloud attackers to the cloud victim.

The cloud computing landscape has witnessed significant advancements in developing robust mechanisms to detect and prevent DoS/DDoS attacks. Researchers have extensively explored diverse methodologies, employing machine learning (ML) and deep learning (DL) techniques to fortify cloud environments against such threats. The following review highlights key contributions in this field.

According to Kacha-vimath and Nar-ayan (2021), the designed approach of detecting a DDoS attack can be used at a very high detection rate, noting various sequencing patterns from gathered traffic with deep learning to classify high-level features. All the designed way evaluations have shown that this DDoS attack detection technique performs quite well with the CNN and also with the MLP in terms of efficiency and high precision. However, one of the major challenges is to detect the DDoS attacks in real-ptime.

Bhardwaj and others (2020) have formulated an exclusive architectural model by merging

Deep Neural Network (DNN) using a stacking sparse AutoEncoder as feature learning for

identifying DDoS attack traffic and normal benign traffic in network activity. Adjusting the parameters

with the right methodology made AE and DNN the most efficient in detecting DDoS attacks.

The recommendations given here result in a network that is compact and avoids overfitting,

reconstruction error minimized, and explosion and vanishing gradation proof.

However, distributed and dynamic characteristics pose a lot of applicability of cloud computing

to various cyber-attacks and security threats which are very specific to the cloud model for flaws

in virtualization technology.

Phan and Park (2019) have developed an SDN-based DDoS mitigation in the cloud. The technique comprised of three processes. The first process of this technique was traffic classification enhancement. Hence, this technique has been designed with hybrid machine learning that includes self-organizing map and support vector machine techniques. The second phase was speeding up and increasing the rate of detection of attack, and they proposed An Improved History-Based IP Filtering (eHIPF). This hybrid model consisted of machine learning modeling as well as the eHIPF method to provide protection against DDoS attacks in SDN-based cloud systems. However, DDoS attacks cause significant damage to any network system regardless of the fact that it may save phylogenetic or not.

Prath-yusha and Kanna-yaram (2021) have designed an application to find potential characteristics of DDoS attack mitigation in the cloud computing domain using artificial immune systems. The idea behind the technology is that the device can identify threats and behave like the innate defenses of the human body. To simulate this, a number of immune responses were designed along with an intrusion detection system. Primarily, the problem is in localizing and identifying the DDoS attack as computational demands are dramatically raised with the intensive processing nature inherent in cloud computing.

Community cloud has been used by Abdullayeva (2022) to introduce a detection system using data clustering for e-government DDoS attacks. The method relies on a feature selection strategy for improving the efficiency of the data clustering. In order to allow feature selection, the PCA technique has been applied. For analyzing the dataset generated after feature selection, the DBSCAN, agglomerative, and k-means techniques have been used. The study found that techniques producing better clustering results along all measures use fewer features than those that produce the results by methods using all the variables. The prevention and avoidance of network intrusions are one of the critical security issues in cloud computing.

According to David and Thomas (2020), the source IP, destination IP, destination port, and protocol were network traffic metrics useful for detecting a DDoS attack. The traffic features are handled with time series models to avoid prediction errors with respect to entropy. This paper will use the GARCH (Generalized ARMA model) model, a nonlinear model best adapting long-range nonstationary datasets including network traffic, to better identify efficiency. It is thus important for identifying DDoS attacks.

Batchu and Seetha (2022) developed a completely new model that handled the DDoS attack detection problems with efficiency. The first part was about data preparation for training data quality. This was followed by minority class sampling using an adaptive synthetic oversampling method in order to remedy the class imbalance issue. Then using five basic classifiers and recursive feature elimination the SHAP feature importance classes were selected. Hyperparameter tuning of these classifiers also revealed the most important characteristics. Finally, local and global explanations were also given for the features so that transparency could be maintained. The situation of most of the classification has grown very more difficult due to the volume increase in the traffic, features and frequency of the data. As per the above articles, it was highly and greatly challenging to identify DDoS attacks in actual time. The predicted approaches applied fundamentally different aspects on evaluating data and identifying activities from diverse types of collected traffic. Another method for performance evaluation in DDoS attack has been brought by the assessment of the statistical data using machine learning techniques. Then again, these sorting methods can be very much complex and may produce a low rate of detection due to the irrelevant features, transparency, and class imbalance. To overcome these problems above-mentioned and demonstrated, this model proposed a DOs attack prediction using effective bio-inspired algorithm base feature selection and Bi-LSTM based cloud environment.

➢ <u>**Proposed methodology**</u>

The unprecedented large data volumes are organized in a secure data platform or public forum depending on the demand. The case is also increased usage most of the time; such increased usage raises security issues. Hacking is a risk to data in cloud computing, but they are most often the case of a DDoS attack. Various attack detection systems developed for different attacks are still not bringing better results. The most crucial problem identified in the presently available algorithms is
massive time consumption and being stuck in local optima. The technique is selecting the best subset of features that improve the accuracy of a reduce feature subset length. An effective deep learning strategy has also been applied to monitor attacks in the cloud environment. The proposed model is depicted in Fig. 1, which consists of 3 main modules: data preprocessing, feature selection through honey badger optimization algorithm and attack detection using Bi-LSTM. Input features are first acquired from the DDoS attack dataset. The subsequent step involves the preprocessing of input features with Bayesian and Z-score normalization techniques. Then the optimal features are selected utilizing HBO. In turn, optimization is done with minimum MSE optimization of the features; thus, the optimal feature forwarded to the next stage is the classification step composed by Bi-LSTM classifier who predicts DDoS attacks. So, the method is proposed with new features as it follows the categorization method and uses it to detect

the DDoS attack system and can exhibit its ability to predict that an attack is happening or not. Definition of procedure involved in the proposed model is as follows:.

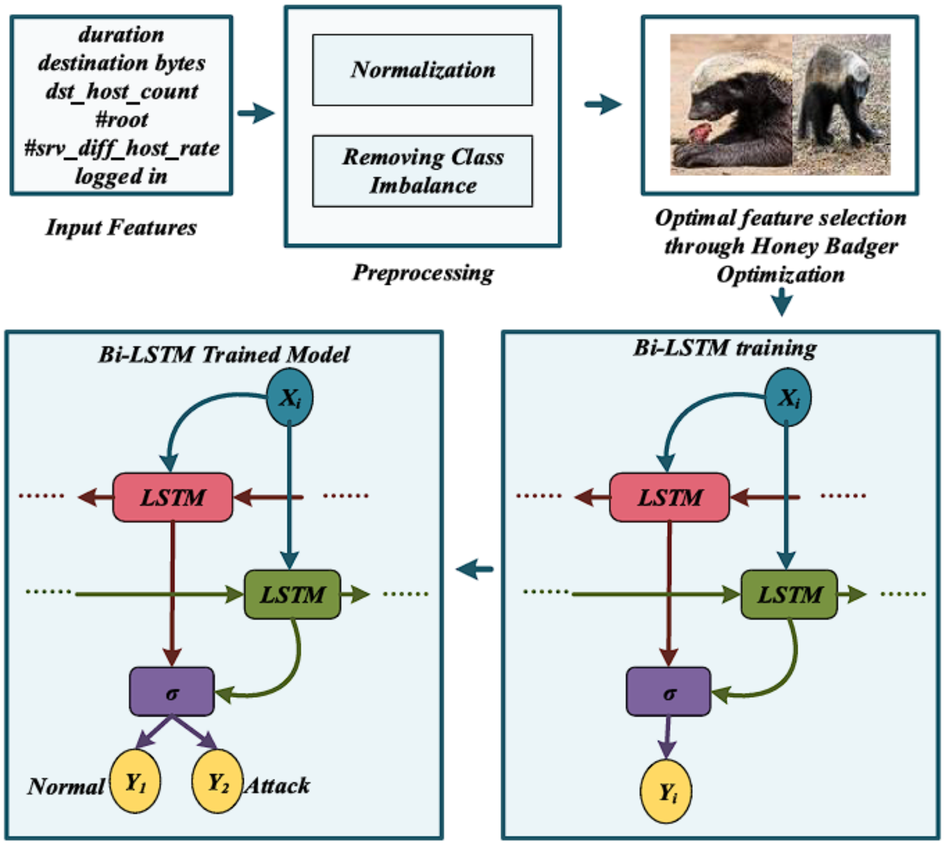


Fig. 1. Structure for the proposed DDoS attack prediction system.

3

➢ Data preprocessing

The dataset taken contains attributes that involve source and destination bytes, host details, flags, protocol types, number of bad logins, bad fragments, services, time, and urgency, etc. All of them differ in types as well such as numeric, binary, and nominal. Due to there being numerous kinds of data present, raw input features cannot be handled. This is also a cause of affecting the accuracy during detection. Thus, it standardizes and removes the cases of the proposed class imbalance model.

- ➢ **Missing value imputation using Bayesian approach**
- ➢ Some Bayesian methods include imputing left-censored data by using a data augmentation approach and then manipulating or deriving implications such as computation of general statistics or regression coefficients through prediction within the Bayesian fold. This Bayesian model, which has been applied in the imputation of the unknown values in ˜X initially substituted with zeros (Jun et al., 2019). The basic intuition is that the distribution p̃(xt) generates

one observation ̃ xt per unit of time. Thus, the distribution p(z) generates the latent variable z in this way.The following is a representation of the joint distribution, p̃(xt, z):

$$\mathrm{p}\left(\widetilde{x}_t, \mathrm{z}\right) = \mathrm{p}(\mathrm{z})p(\widetilde{x}_t|z)$$

Samples from the prior, z distribution and probability p̃(xt|z), distribution of ̃ xt given z may be used to build the joint distribution in Equation (1). The previous value p(z) is commonly selected to represent a multi-dimensional typical distribution having a zero-mean and unit variance while its probability p̃(xt|z) may be calculated through a generating network.

➢ <u>Normalization</u>

The categorical data are transformed into numerical data during this normalisation. Z-score normalization is the process of normalizing each value within a data set such that the average for each value is 0 as well as its standard deviation is 1 (Kappal, 2019). Concretely, let xi (i=1,2, ⋯, D) denotes the ith component of each feature vector x∈RD. Find the mean and standard deviation for these D elements initially.

$$\mu_x = \frac{1}{D}\sum_{i=1}^{D} x_i, \sigma_x = \sqrt{\frac{1}{D}\sum_{i=1}^{D}\left(x_i - \mu_x\right)^2}$$

Z-score normalization is then applied as,

$$x^{(ZN)} = ZN(x) = \frac{x - \mu_x 1}{\sigma_x} \in R^D$$

Where 1=[1,1,..,1]T is a D-dimensional vector with its components being all ones. The original feature vectors are first projected via z-score normalization along the 1 vector to a hyperplane that contains the origin and is perpendicular to 1.

<u>Optimal feature selection through honey badger optimization</u>

Feature Selection Using Honey Badger Optimization (HBO) in Intrusion Detection Systems

In this research, an optimal feature selection process using the Honey Badger Optimization (HBO) algorithm is introduced to enhance the performance of an intrusion detection system. Feature selection plays a crucial role in reducing computational complexity and improving detection accuracy by removing redundant and irrelevant features. Traditional methods often suffer from drawbacks such as high computational cost and entrapment in local optima. The proposed HBO-based approach addresses these limitations, enabling the selection of a more efficient subset of features while improving model accuracy and minimizing the feature set size.

➢ Overview of Honey Badger Optimization (HBO)

The HBO algorithm draws inspiration from the foraging behavior of the honey badger, a resilient mammal native to semi-deserts, rainforests, and savannas in regions such as Africa, Southwest Asia, and the Indian subcontinent. Honey badgers exhibit remarkable persistence and strategy in locating food. Their behavior encompasses two key modes: **digging mode** and **honey mode**, both of which are incorporated into the computational structure of the HBO algorithm.

1. **Digging Mode**:
   Honey badgers utilize their acute sense of smell to detect prey, such as rodents, underground. They repeatedly dig in an area—sometimes making up to 50 burrows in a day over distances exceeding 40 kilometers—to locate their target with precision.

2. **Honey Mode**:
   Although honey badgers are known to enjoy honey, they rely on a unique partnership with the honeyguide bird to find beehives. The bird guides the badger to hives, which the badger then breaks open using its claws. This cooperative interaction demonstrates a mutual benefit, as the bird gains access to beeswax while the badger consumes honey.

➢ HBO Algorithm Framework

The HBO algorithm replicates these foraging strategies to perform global optimization in computational problems. It consists of two phases: **exploration** and **extraction**, mimicking the badger's behavior of locating and accessing its target. The algorithm alternates between modes to improve efficiency in finding the optimal solution.

➢ Key Steps in the HBO Algorithm

1. **Initialization**:
   A population of candidate solutions is generated, representing different feature subsets. Each candidate is evaluated based on its fitness, which, in this case, depends on the accuracy and relevance of the features for intrusion detection.

2. **Exploration Phase (Digging Mode)**:
   The algorithm searches the solution space extensively, emulating the honey badger's method of creating multiple burrows to pinpoint prey locations. This helps prevent the algorithm from getting stuck in local optima.

3. **Exploitation Phase (Honey Mode)**:
   Once the general vicinity of the optimal solution is identified, the algorithm

transitions to a more focused search, similar to the honey badger following the honeyguide bird to a specific hive. This phase refines the solution to identify the most suitable feature subset.

4. **Switching Between Modes**:
   The algorithm dynamically switches between the exploration and exploitation phases to balance global and local optimization.

➢ **Advantages of the Proposed HBO Approach**

The HBO algorithm demonstrates significant benefits compared to conventional methods, including:

- **Reduced Computation Time**: The efficient transition between exploration and exploitation reduces the number of iterations required for convergence.
- **Avoidance of Local Optima**: Dynamic switching ensures that the algorithm explores a broader search space.
- **Enhanced Feature Subset Selection**: By minimizing redundancy and retaining only the most impactful features, the selected subset improves the detection system's performance.

This novel application of the HBO algorithm in feature selection contributes to the development of an optimized intrusion detection system. The HBO-based approach not only improves the overall detection accuracy but also significantly reduces the computational burden, making it a promising solution for complex and dynamic environments like cloud computing.

Although it includes an exploration and extraction phase, HAB is a global optimization technique. The following describes the mathematical steps for the proposed HBA. The population of potential solutions in HAB is shown above.

$$Population\ of\ candidate\ solutions = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1D} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2D} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nD} \end{bmatrix}$$

From equation (4), Ith position of the honey badger is represented as,

$$x_i = [x_i^1, x_i^2, \cdots x_i^D].$$

Step 1: Initialization phase Set the population number with N as well as the honey badgers' places to correspond by equation (5).

$$x_i = lb_i + r_1 \times (ub_i - lb_i)$$

According to equation (5), r1 is a chance number between 0 and 1. Where lbi and ubi are the lower and upper limits of the search domain, respectively, and xi is the ith honey badger location referring to a potential solution in a population of N. Step 2: Defining intensity (I) Intensity is related to the concentration strength of the prey and the distance among it and ith honey badger. The inverse square law, which is depicted in Fig. 2 as well as stated in Equation (6), indicates that the smell of the victim is strong, its movement will be rapid and vice versa. Ii represents the strength of the victim's fragrance.

$$I_i = r_2 \times \frac{S}{4\pi d_i^2}$$

$$S = (x_i - x_{i+1})^2$$

$$d_i = x_{prey} - x_i$$

S stands for source or concentration strength, di denotes the separation between the ith badger and the prey in equation (8). Step 3: Update density factor A density factor (α)that regulates time-varying randomness helps sure that moving from exploring to exploiting is simple. The following equation should be updated to reflect decreasing factor α that gets smaller over time when iterations are performed to reduce randomization (9).

$$\alpha = C \times \exp(\frac{-t}{t_{max}})$$

From equation (9), C represents the constant ≥1. Step 4: Getting away from the local optimum Following that, three processes are employed to exit localized opti mum zones. In this case, the proposed method requires utilizing a flag F that adjusts searching direction to provide the agent with a high chance of extensively exploring the search space. Step 5: Updating the agents' positions As previously mentioned, the "digging phase" and the "honey phase" are the two stages of the HBA position updating procedure. A better explanation is provided below. Digging Phase: The cardioid structure illustrated by equation (10) corresponds to the digging habit of a honey badger.

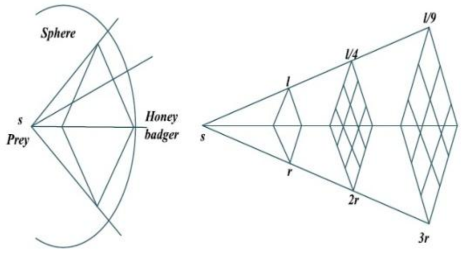


Fig. 2. Square-inverse law, I indicates the intensity of the smell, S the position of the victim and r for a random number from 0 to 1.

The honey badger having the capacity to locate food due to indicated as β≥1and the location of the victim, xprey is given from equation (10), and the distance between prey is given as di and the ith honey badger is given in equation (5). The three different random

$$x_{new} = x_{prey} + F \times \beta \times I \times x_{prey} + F \times r_3 \times \alpha \times d_i \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]|$$

numbers between 0 and 1 are r2,r4andr5. The following equation (11) can be used to determine F, which acts as the flag that changes the direction of the search.

$$F = \begin{cases} 1 if r_6 \leq 0.5 \\ -1 else \end{cases}$$

From equation (11), r6 is a number chosen at random from 0 to 1. This honey badger's primary source of food is smell intensity I the prey is xprey, the difference among badgers as well as Prey di with the time- dependent searching impact factor α during the digging stage. Additionally, a badger may experience any disruption F while digging, enabling it to locate its prey in even better conditions. Honey Phase: A honey badger tracking a honey led bird to hives may be simulated using equation (12).

$$x_{new} = x_{prey} + F \times r_7 \times \alpha \times d_i$$

In equation (12), the terms xnew and xprey indicate the honey badger's new position with the prey's location, F and α were derived from equations (11) and (9), respectively. By the basis of equation (11), this may be shown that honey badger seeks regions near its prey places that fluctuate over time (α). A honey badger might also discover disturbance F. Because of its exploration and exploitation phases, HBA is logically considered to be a global optimization technique. The minimum number of operators that need to be changed is necessary to make the HBA simple to operate and understand. Note conscious that the proposed method has a computational price for O(tmaxND), when N indicates the population number or size of solutions tmax stands for the maximum number of iterations, and D stands for the total number of choice factors.

3.2.1. Optimal feature selection using honey badger optimization

The selection of useful features is regarded as the most important step in developing any detection algorithm since useful features are sufficient to carry out the detection with high accuracy. In order to select the best features, this method uses an optimization technique known as honey badger optimization. By measuring the MSE value among the features, the best features are obtained in this case. The following de scribes the technique behind using the HBO algorithm. Step 1: Initialization In this phase, the preprocessed features are considered and which is initialized as follows.

Z=[x1,x2,x3,⋯xn]

Step 2: Fitness Function To obtain the optimal features through minimizing the Mean Square Error (MSE) value. The following equation shows the fitness function of the process of election.

Fitness=min {MSE}

$$MSE = \frac{1}{N} = \sum_{i=1}^{N} (f_i - y_i)^2$$

Where N is the number of data points, fi the value returned by the model and yi the actual value for data point i. Step 3: Update the function The MSE value is associated with getting updated in each iteration to locate the best features within the characteristics. Step 4: Termination Once the ideal choice has been identified, the process is finished. Through this process, the optimal features in the intrusion dataset are elected by minimizing its MSE value. Moreover, these features consist of more information about the intrusions, which is taken as input for the detection process. Pseudocode for optimal feature selection using HBO is given in algorithm 1.

Algorithm 1: Pseudocode for optimal feature selection using HBO

```
Input: Number of features $(x_1, x_2, x_3, \cdots x_n)$
Begin
Initialize $(x_1, x_2, x_3, \cdots x_n)$
While $t \leq t_{max}$ do
Update the decreasing factor $\alpha$ using (9)
for $i = 1$ to $N$ do
Calculate the intensity $I_i$ using Equation (7)
if $r < 0.5$ then
Update the position $x_{new}$ using Equation (10)
else
Update the position $x_{new}$ using Equation (12)
end if
Evaluate new position and assign to $f_{new}$
if $f_{new} \leq f_i$ then
Set $x_i = x_{new}$ and $f_i = f_{new}$
end if
if $f_{new} \leq f_{prey}$ then
Set $x_{prey} = x_{new}$ and $f_{prey} = f_{new}$
end if
end for
end while stop criteria satisfied
Return $x_{prey}$
Output: optimal features are selected
```

This Project used Bi-LSTM, which is a deep learning approach that incorporates sequencing input by two networks. The first runs in normal temporal direction while the other in reverse logical sequence order.Outputs from both networks at each time step are sequential. An accurate classification results from a stacked layer Bi-LSTM architecture, which allows collection of both back ground and forward information regarding a sequence at every phase of time. The architecture of the Bi-LSTM classifier is the following Fig. 4. Equations (22) and (23) explain how the bi-LSTM classifier processes data in the reverse and forward directions.

$$\overrightarrow{h_t} = f(w_1 x_t + w_2 \overrightarrow{h_{t-1}})$$

$$(\overleftarrow{h_t}) = f(w_1 x_t + w_2 \overleftarrow{h_{t+1}})$$

$$O_t = g(w_4 \overrightarrow{h_t} + w_6 \overleftarrow{h_t})$$

Here, Ot is the outputs generated for (y1⋯,yn⋯,yt). f represents the function of reverse as well as forward process. This Bi-LSTM takes as its input, the optimal features chosen by the optimization algorithm. This intrusion detection method has the whole process in two phases. It includes both the training as well as testing phases. First, this optimal feature set chosen is divided into two parts in the ratio 80:20 for both the training as well as the testing. The training process is the first one, in which the model is developed based on 80% of the optimum features. The trained Bi-LSTM model is further tested by the remaining 20% of the optimum features. This process makes the Bi-LSTM model classify input signals into two classes, normal and attacked. Above-said models improve cloud security in data storage and also its transfer from one location to another..

➢ <u>Result and Discussion</u>

➢ <u>Dataset description</u>

The dataset utilized for predicting DDoS attacks was sourced from the open platform Kaggle (Dataset 1), in addition to other accessible IDS datasets such as CSE-CIC-IDS2018-AWS, CICIDS2017, and CIC DoS (2016). These datasets have been processed to extract DDoS flows. To ensure diversity, DDoS data was extracted from multiple IDS datasets generated across various years and through distinct attack simulation processes. This data is then combined with "Benign" traffic flows from the same datasets to form a comprehensive dataset.

Data collection originated from connected devices in the cloud, forming the raw dataset for the study. Several preprocessing techniques were applied to enhance data quality. This phase included handling missing values using imputation techniques and normalizing the dataset. Missing values were replaced with estimated values through a Bayesian approach, as depicted in the graphical analysis.

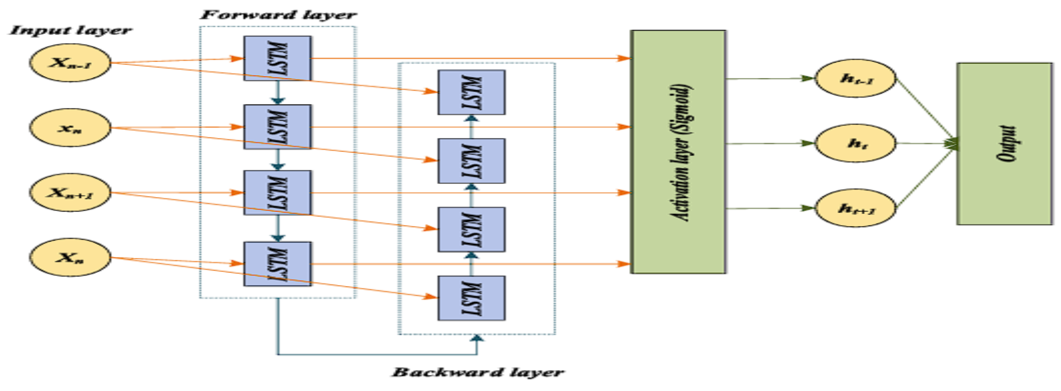
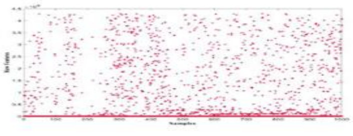


Fig. 4. Structure of Bi-LSTM classifier.

The preprocessed data underwent a feature selection process utilizing an optimization method to minimize the Mean Squared Error (MSE). The selected features were then employed for the classification process to detect DDoS attacks. The performance of the proposed method was evaluated against established approaches like LSTM, DNN, DBN, and ANN. Key evaluation metrics included sensitivity, specificity, F1 score, kappa, false positive rate, accuracy, and error rate.



(a)

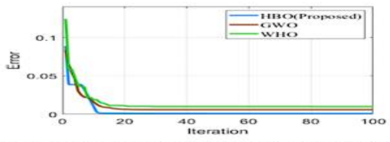
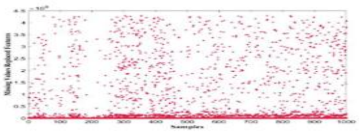


Fig. 6. Convergence plot for the proposed HBO algorithm.



(b)

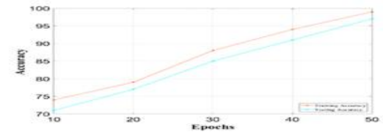


Fig. 7. Comparison of accuracy for Bi-LSTM training Epoch.

The preprocessing phase is demonstrated through graphical representations. For example, Fig. 5a shows raw features with missing values, while Fig. 5b highlights the improved features post-imputation using the Bayesian method. The convergence graph (Fig. 6) reveals that the proposed Hybrid Bayesian Optimization (HBO) algorithm achieves convergence with zero error at the 12th iteration, outperforming other optimization algorithms.

During the training phase, the Bi-LSTM model's accuracy improved with increased epochs. At 10 epochs, the accuracy reached 72%, rising to 85% at 30 epochs and 97% at 50 epochs, as shown in Fig. 7. The error rate decreased significantly, stabilizing at 1% by the 50th epoch (Fig. 8). These findings illustrate the robust learning capability of the Bi-LSTM model.

The training time for the proposed Bi-LSTM method was notably lower than existing methods. The Bi-LSTM required 780 seconds for training compared to LSTM (874 seconds), DNN (1140 seconds), DBN (1270 seconds), and ANN (1549 seconds), as depicted in Fig. 9. Similarly, Fig. 10 highlights that the Bi-LSTM achieved a testing time of 0.8 seconds, outperforming LSTM, DNN, DBN, and ANN.

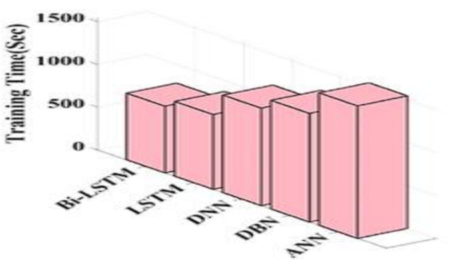


Fig. 9. Comparison of Training time.

The overall execution time analysis (Fig. 11) further supports the efficiency of the Bi-LSTM method, which achieved 780 seconds compared to longer times for LSTM (875 seconds), DNN (1143 seconds), DBN (1270 seconds), and ANN (1554 seconds).

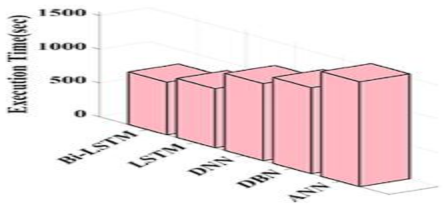


Fig. 11. Comparison of Execution time.

Performance metrics comparison revealed that the proposed Bi-LSTM model consistently outperformed other techniques across all parameters. The Bi-LSTM achieved higher accuracy (97%, Fig. 12), sensitivity (95%, Fig. 13), specificity (90%, Fig. 14), and precision (94%, Fig. 16). Additionally, it demonstrated a lower error rate (3%, Fig. 15) and false positive rate (5%, Fig. 18). The F1 score (87%, Fig. 19) and Kappa coefficient (88%, Fig. 17) further emphasize the model's superior performance.

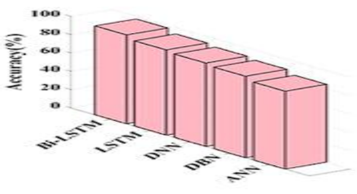


Fig. 12. Analysis of Accuracy metrics.

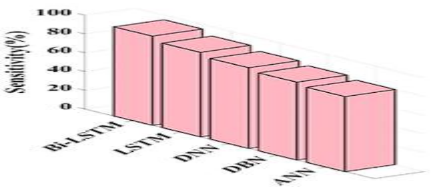


Fig. 13. Analysis of Sensitivity metrics.

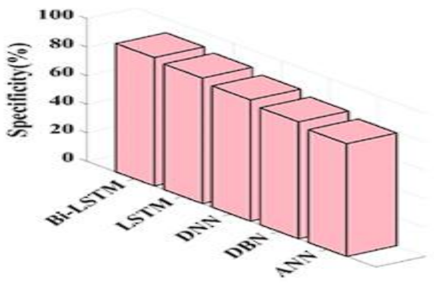


Fig. 14. Analysis of Specificity metrics.

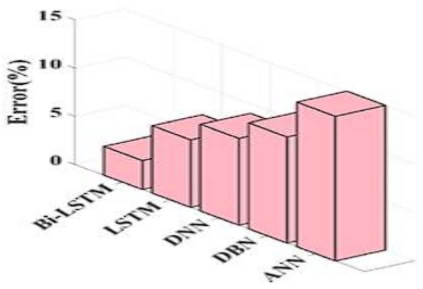


Fig. 15. Analysis of Error metrics.

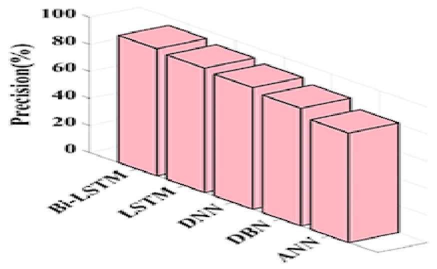


Fig. 16. Analysis of Precision metrics.

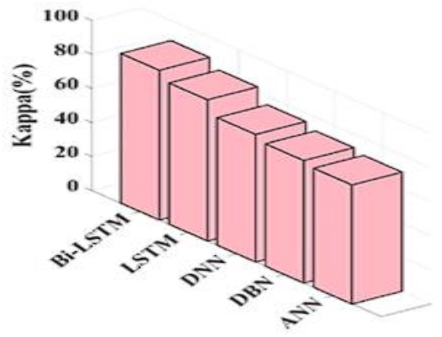


Fig. 17. Analysis of kappa metrics.

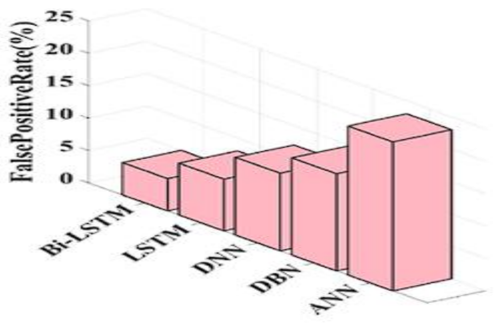


Fig. 18. Analysis of False Positive Rate metrics.

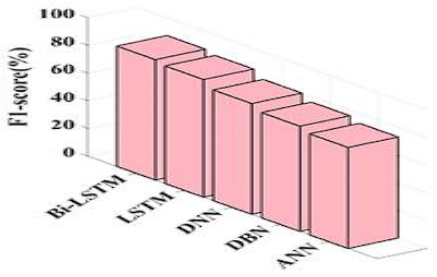


Fig. 19. Analysis of f1_score metrics.

In conclusion, the Bi-LSTM model surpassed traditional methods like LSTM, DNN, DBN, and ANN in accuracy, computational efficiency, and reliability, making it a highly effective approach for DDoS attack detection in cloud environments.

Analysis and Evaluation of the Proposed Bi-LSTM Model Against Existing Techniques

The Matthews Correlation Coefficient (MCC) metric comparison between the proposed Bi-LSTM and existing methods is shown in Fig. 20. The Bi-LSTM achieved an MCC of 86%, outperforming the existing classifiers LSTM (83%), DNN (78%), DBN (74%), and ANN (71%).

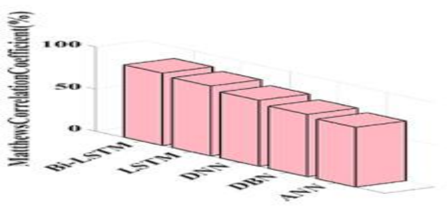


Fig. 20. Analysis of MatthewsCorrelationCofficient metrics.

This highlights the proposed method's higher precision and robustness compared to traditional approaches.

False Negative Rate (FNR), depicted in Fig. 21, demonstrates that the proposed Bi-LSTM classifier has a significantly lower FNR of 6% compared to LSTM (9%), DNN (14%), DBN (15%), and ANN (25%). The reduced FNR indicates the Bi-LSTM's superior ability to detect true positives**.**

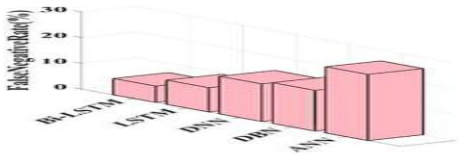


Fig. 21. Analysis of FalseNegativeRate metrics.

False Discovery Rate (FDR) analysis, illustrated in Fig. 22, shows that the Bi-LSTM achieved an FDR of 0.04%, outperforming other techniques. In contrast, LSTM, DNN, DBN, and ANN recorded FDRs of 0.11%, 0.17%, 0.20%, and 0.22%, respectively. These results demonstrate the enhanced prediction accuracy of the Bi-LSTM approach.

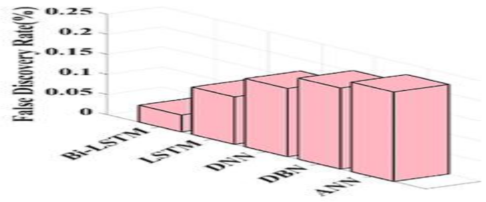


Fig. 22. Analysis of FalseDiscoveryRate.

Fig. 23 shows the comparison of **Positive Likelihood Ratio (PLR)** measures. The Bi-LSTM achieved a PLR of 5%, surpassing LSTM (2%), DNN (1%), DBN (1%), and ANN (1%). This demonstrates the superior reliability of the Bi-LSTM in predicting DDoS attacks effectively.

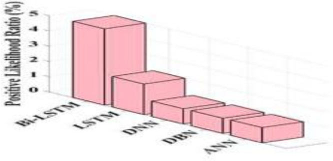


Fig. 23. Analysis of Positive Likelihood Ratio.

Negative Likelihood Ratio (NLR) results, as depicted in Fig. 24, reveal a lower NLR value for the Bi-LSTM (0.04%) compared to LSTM (0.10%), DNN (0.17%), DBN (0.20%), and ANN (0.22%). This indicates that the proposed method performs more efficiently than existing approaches.

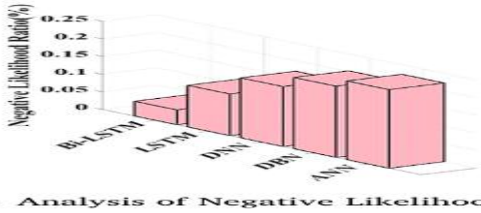


Fig. 24. Analysis of Negative Likelihood Ratio.

Fig. 25 presents the **Negative Predictive Value (NPV)**, where the Bi-LSTM scored 90%, outperforming LSTM (84%), DNN (78%), DBN (75%), and ANN (72%). This highlights the proposed model's ability to minimize false negatives and ensure accurate predictions.

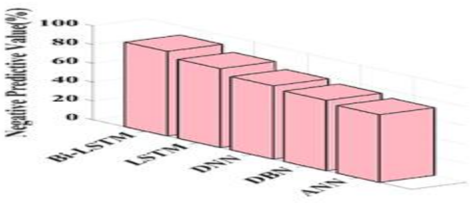


Fig. 25. Analysis of NegativePredictivevalue.

Fig. 26 explores **False Omission Rate (FOR)** metrics, with Bi-LSTM attaining a lower FOR of 10%, compared to LSTM (16%), DNN (22%), DBN (25%), and ANN (28%). This showcases the Bi-LSTM's effectiveness in reducing missed detections.

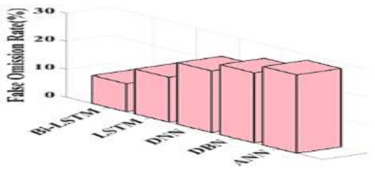


Fig. 26. Analysis of FalseOmissionRate.

The **Fowlkes-Mallows Score**, visualized in Fig. 27, reflects the Bi-LSTM's superiority, achieving a score of 94%. Comparatively, LSTM, DNN, DBN, and ANN achieved 90%, 87%, 84%, and 80%, respectively. This metric further demonstrates the Bi-LSTM's improved precision and recall.

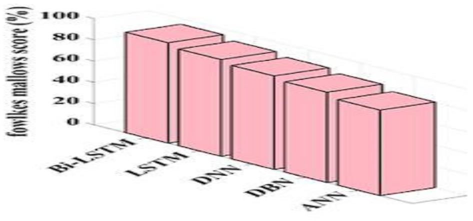


Fig. 27. Analysis of Fowlkes Mallows score.

Fig. 28 highlights **Markedness Metrics**, with the Bi-LSTM achieving a markedness value of 84%, outperforming LSTM (76%), DNN (68%), DBN (62%), and ANN (53%). This metric underscores the model's ability to provide accurate predictions with minimal errors.

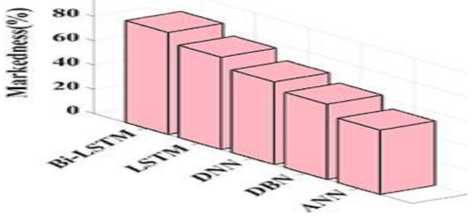


Fig. 28. Analysis of Markedness.

The **Prevalence Threshold**, analysed in Fig. 29, shows that the Bi-LSTM achieved a threshold of 0.22%, which is lower than LSTM (0.28%), DNN (0.34%), DBN (0.39%), and ANN (0.47%). This reflects the Bi-LSTM’s enhanced efficiency in handling diverse data distributions.

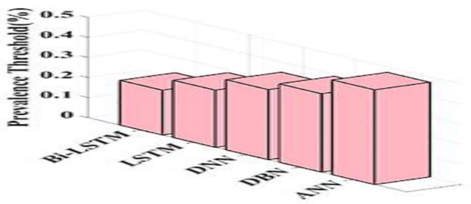


Fig. 29. Analysis of Prevalence threshold.

Informedness Measures, illustrated in Fig. 30, show that the Bi-LSTM achieved an informedness score of 85%, outperforming LSTM (78%), DNN (71%), DBN (65%), and ANN (59%). This metric highlights the model's ability to maximize true positive and true negative rates while minimizing false predictions.

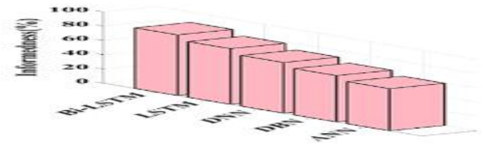


Fig. 30. Analysis of Informedness.

- **<u>Comparative Analysis with Current Approaches</u>**

  The proposed Bi-LSTM model was benchmarked against various current methods in the literature, including LSTM, DNN, AIS, and DBN. The Bi-LSTM consistently outperformed these methods, achieving an accuracy of 97%, which is significantly higher than the alternatives. Detailed classification performance, along with execution times, is summarized in Table 1. The table underscores the Bi-LSTM's superior accuracy, reduced error rates, and efficient computational performance, making it a leading solution for DDoS detection in cloud environments.

<u>Conclusion</u>

- **DDoS Attack Prediction Using Honey Badger Optimization-Based Feature Selection and Bi-LSTM in Cloud Environments**
- This research presents a novel approach for predicting Distributed Denial of Service (DDoS) attacks in cloud computing environments, leveraging a Honey Badger Optimization (HBO) algorithm for feature selection combined with a Bi-LSTM model for attack detection. Cloud computing offers businesses a scalable platform for resource sharing, data management, and service delivery. However, the widespread adoption of cloud platforms introduces significant risks, including data breaches, unauthorized access, and compromised privacy. The emergence of DDoS attacks, which exploit vulnerabilities in cloud, edge computing, and IoT devices, further exacerbates these concerns. These attacks often employ sophisticated flooding techniques, making traditional machine learning models less effective due to their high false alarm rates, lengthy training times, and suboptimal detection accuracy.
- The proposed DDoS prediction model addresses these challenges through a systematic three-phase process: **data preprocessing, feature selection, and attack detection**. Initially, data related to connected devices in the cloud environment is collected to create a dataset. During the preprocessing stage, missing values are replaced, and normalization techniques are applied to prepare the data for analysis. The refined dataset is then fed into the proposed Honey Badger Optimization algorithm to select optimal features by minimizing the Mean Squared Error (MSE). These selected features are subsequently passed to the Bi-LSTM model, which detects potential DDoS attacks in the cloud infrastructure.

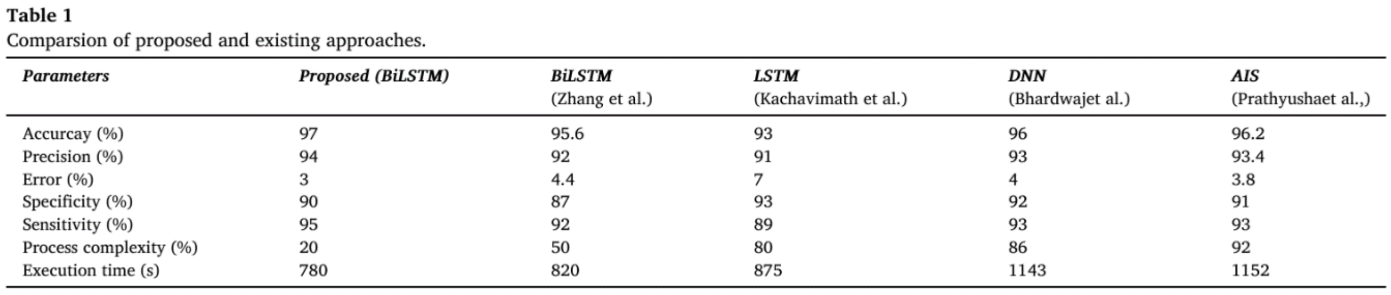


Table 1
Comparsion of proposed and existing approaches.

Parameters	*Proposed (BiLSTM)*	*BiLSTM* (Zhang et al.)	*LSTM* (Kachavimath et al.)	*DNN* (Bhardwajet al.)	*AIS* (Prathyushaet al.,)
Accurcay (%)	97	95.6	93	96	96.2
Precision (%)	94	92	91	93	93.4
Error (%)	3	4.4	7	4	3.8
Specificity (%)	90	87	93	92	91
Sensitivity (%)	95	92	89	93	93
Process complexity (%)	20	50	80	86	92
Execution time (s)	780	820	875	1143	1152

The proposed approach demonstrates significant improvements in detection efficiency and accuracy when benchmarked against existing techniques such as LSTM, DNN, DBN, and ANN. The performance of the Bi-LSTM model is evaluated using a range of metrics, achieving the following results:

- **Accuracy:** 97%
- **Sensitivity:** 95%

- **Specificity:** 90%
- **Precision:** 94%
- **F1 Score:** 87%
- **Fowlkes-Mallows Score:** 94%
- **Kappa Score:** 88%
- **Markedness:** 84%
- **False Positive Rate (FPR):** 5%
- **False Discovery Rate (FDR):** 10%
- **Negative Predictive Value (NPV):** 90%
- **Error Rate:** 3%

The model operates efficiently with a **training time of 780 seconds**, a **testing time of 843 seconds**, and an **execution time of 780 seconds**. The integration of the Honey Badger Optimization algorithm enhances feature selection by focusing on the most impactful attributes, significantly improving the Bi-LSTM's ability to identify and classify DDoS attacks.

This research highlights the effectiveness of the proposed model in detecting malware and DDoS attacks within cloud environments. Additionally, it provides recommendations for further research, emphasizing the need for enhanced classification techniques and more robust solutions to tackle evolving cybersecurity threats in distributed computing systems. The proposed methodology offers a promising foundation for improving network security and safeguarding cloud platforms against complex DDoS attacks.

As noted by Napper and Bientinesi (2009), cloud computing offers remarkable scalability for computing resources on demand, providing users with significant advantages over traditional cluster systems. One of the key benefits of cloud computing is that it eliminates the need for upfront investment in infrastructure, labor, and ongoing operational costs. Essentially, the cost associated with using cloud resources is negligible when those resources are not actively in use. This makes cloud computing an attractive solution, driving both academic research and industry adoption.

However, despite its many benefits, there are security concerns that mirror the early mistakes made during the development of the internet, where performance and functionality were prioritized over security. In contrast, security should be integrated alongside functionality and performance from the outset to prevent vulnerabilities.

In this project, we explore our work on applying service-oriented architecture (SOA) to cloud computing security. We focus on two significant threats: HTTP-based Denial of Service (H-DoS) and XML-based Denial of Service (X-DoS) attacks, which pose considerable risks to cloud systems. A successful attack could potentially cripple cloud-based services like Amazon EC2.

The H-DoS attack, which exploits the HTTP protocol, is particularly dangerous for cloud environments as it uses HTTP to communicate between systems within the cloud. We demonstrated how such an attack could take place, using the example of the attacks that impacted pro-Iranian websites. This attack could be instigated by a malicious insider, such as a disgruntled employee, or from an external source, like a competitor.

Similarly, the X-DoS attack targets cloud services by flooding the system with malicious XML messages. To counter these threats, we introduced the Service-Oriented Traceback Architecture (SOTA) model and implemented it in the cloud as Cloud Traceback (CTB). Our results show that CTB effectively traces the source of an X-DoS attack within seconds, providing a means to swiftly identify and respond to the threat.

We also developed the Cloud Protector; a neural network trained to detect and filter X-DoS attacks. Our experiments demonstrated that it could filter 98-99% of attack traffic within a response time range of 10-135 ms. We also trained Cloud Protector to identify and filter H-DoS attacks, achieving detection rates of 88-91%. However, we encountered variability in response times, which was due to suboptimal neural network settings. To address this, we propose the use of two distinct configurations within the Cloud Protector: one optimized for HTTP traffic and another for X-DoS traffic.

Building on our previous research in Chaos Theory (Chonka et al., 2010), we hypothesize that cloud computing systems, similar to networks, are governed by non-linear dynamics. This theory suggests that when an attacker initiates an H-DoS attack, they alter the initial conditions of the system, potentially leading to unpredictable behaviour. We aim to continue our research to further explore how Chaos Theory can be applied to understand and mitigate H-DoS attacks within cloud environments.

Bibliography

Amazon Web Services. Amazon Elastic Compute Cloud (ec2), Available on the WWW,2009. /http://aws.amazon.com/ec2S, last accessed 10, November 2009.

Balding G. What everyone ought to know about cloud security. Cloudsecurity.com, /http://www.slideshare.net/craigbalding/what-everyone-ought-to-know-a bout-cloud-security's, 2009.

Belenky A, Ansari N. Tracing multiple attackers with deterministic packet marking (DPM). In: Proceedings of IEEE Pacific Rim conference on communications, computers and signal processing, vol. 1, 2003. p. 49–52.

Chonka A, Zhou W, Xiang Y. Protecting web services with service oriented traceback architecture. In: Proceedings of the IEEE eighth international conference on computer and information technology, IEEE, 2008a.

Chonka A, Zhou W, Xiang Y. Protecting web services from DDoS attacks by SOTA. In: Proceedings of the IEEE fifth international conference on information technology and applications, IEEE, 2008b.

Chonka A, Zhou W, Xiang Y. Defending grid web services from X-DoS Attacks by SOTA. In: Proceedings of the third IEEE international workshop on web and pervasive security (WPS 2009), IEEE, 2009.

Chonka A, Zhou W, Singh J. Chaos theory based detection against network mimicking DDoS attacks. IEEE Communications Letters 2010;13(9):717–9.

Danchev D. Iranian opposition launches organized cyber attack against pro Ahmadinejad sites. ZDNet blog, /http://blogs.zdnet.com/security/?p=3613S, June 15, 2009a.

Danchev D. Iranian Opposition DDoS-es pro Ahmadinejad Sites. Dancho Danchev's Blog, /http://ddanchev.blogspot.com/2009/06/iranian-opposition-ddos-es-pro.htmlS, 16 June 2009b.

Dean D. An algebraic approach to IP traceback. ACM Transactions on Information and System Security (1094–9224) 2002;5(2):119.

Abdullayeva, F. J. (2022). Distributed denial of service attack detection in E-government cloud via data clustering. Array, 15, Article 100229.

Agarwal, A., Khari, M., & Singh, R. (2021). Detection of DDOS attack using deep learning
model in cloud storage application. Wireless Personal Communications, 1–21.

Batchu, R. K., & Seetha, H. (2022). An integrated approach explaining the detection of distributed denial of service attacks. Computer Networks, 216, Article 109269.

Bhardwaj, A., Mangat, V., &Vig, R. (2020). Hyperband tuned deep neural network with well posed stacked sparse autoencoder for detection of DDoS attacks in cloud. IEEE Access, 8, 181916–181929.

Chen, Y., Zheng, W., Li, W., & Huang, Y. (2021). Large group activity security risk assessment and risk early warning based on random forest algorithm. Pattern Recognition Letters, 144, 1–5.

Cui, Z., Zhang, J., Wu, D., Cai, X., Wang, H., Zhang, W., & Chen, J. (2020). Hybrid many-
objective particle swarm optimization algorithm for green coal production problem. Information Sciences, 518, 256–271.

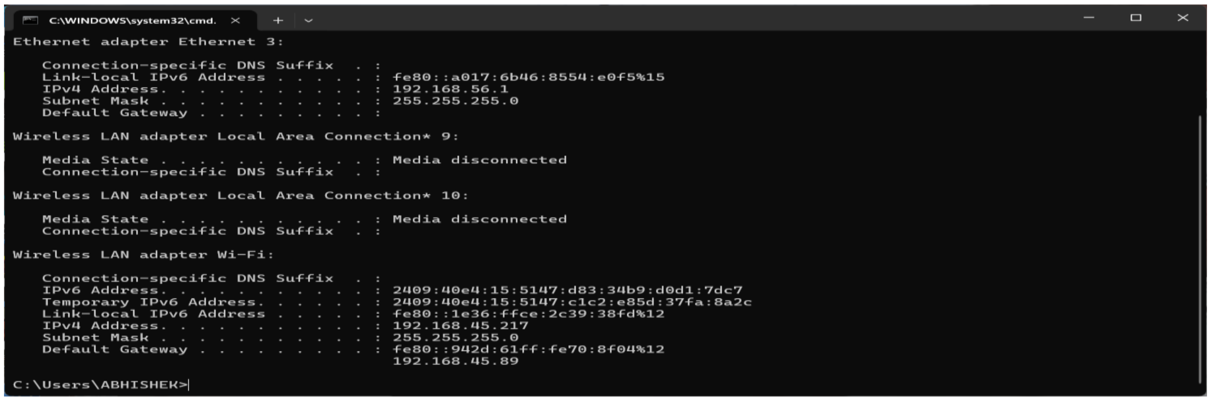
Dataset 1: https://www.kaggle.com/datasets/devendra416/ddos-datasets.

David, J., & Thomas, C. (2020). Detection of distributed denial of service attacks based on information theoretic approach in time series models. Journal of Information Security and Applications, 55, Article 102621.

Evaluation/Experiment

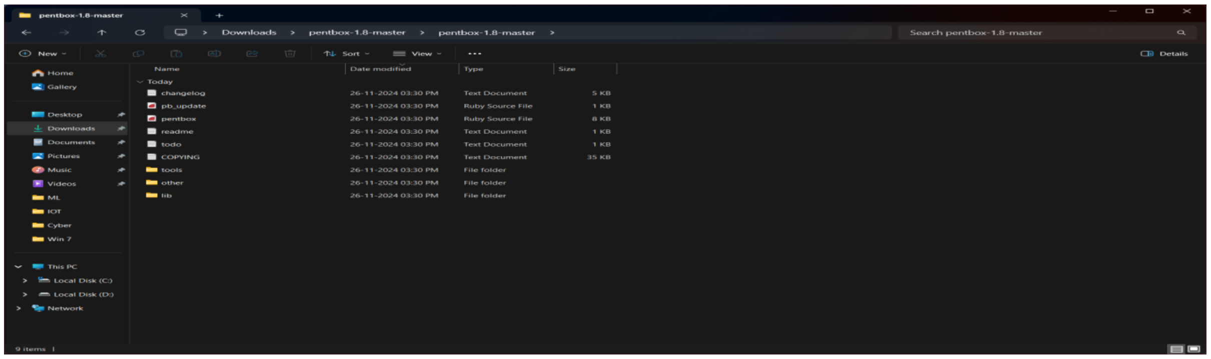
Step1: First we have to open command prompt on our windows machine for finding the Ip

- Command usage: ipconfig



```
C:\WINDOWS\system32\cmd.
Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::a017:6b46:8554:e0f5%15
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 9:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:40e4:15:5147:d83:34b9:d0d1:7dc7
   Temporary IPv6 Address. . . . . . : 2409:40e4:15:5147:c1c2:e85d:37fa:8a2c
   Link-local IPv6 Address . . . . . : fe80::1e36:ffce:2c39:38fd%12
   IPv4 Address. . . . . . . . . . . : 192.168.45.217
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::942d:61ff:fe70:8f04%12
                                       192.168.45.89

C:\Users\ABHISHEK>
```

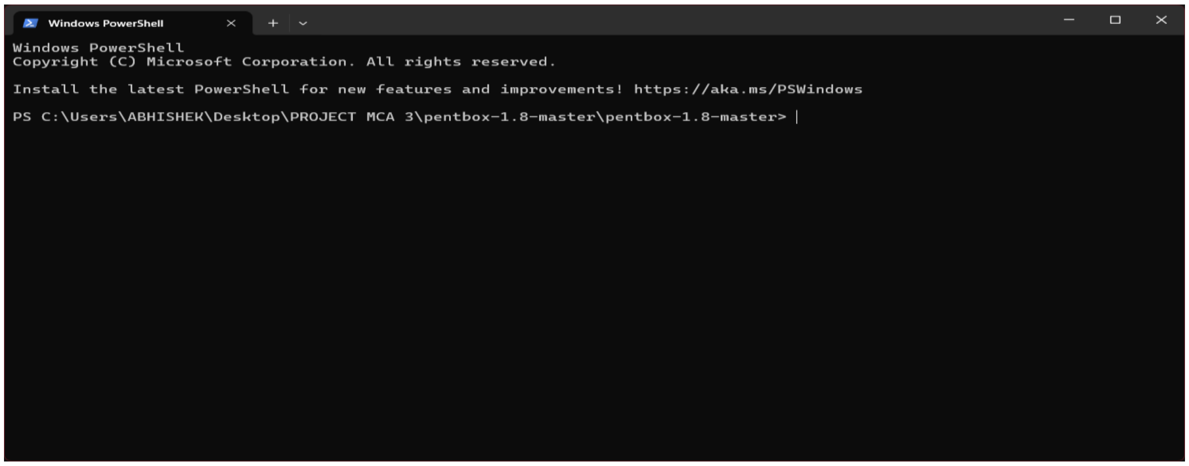
Step2: After finding the Ip , we locate our honeypot detection file



pentbox-1.8-master > Downloads > pentbox-1.8-master > pentbox-1.8-master

Name	Date modified	Type	Size
changelog	26-11-2024 03:30 PM	Text Document	5 KB
pb_update	26-11-2024 03:30 PM	Ruby Source File	1 KB
pentbox	26-11-2024 03:30 PM	Ruby Source File	8 KB
readme	26-11-2024 03:30 PM	Text Document	1 KB
todo	26-11-2024 03:30 PM	Text Document	1 KB
COPYING	26-11-2024 03:30 PM	Text Document	35 KB
tools	26-11-2024 03:30 PM	File folder	
other	26-11-2024 03:30 PM	File folder	
lib	26-11-2024 03:30 PM	File folder	

Step3: At the current file location Right click and then open the menu box and choose open terminal here.The Windows terminal or Powershell is opened.

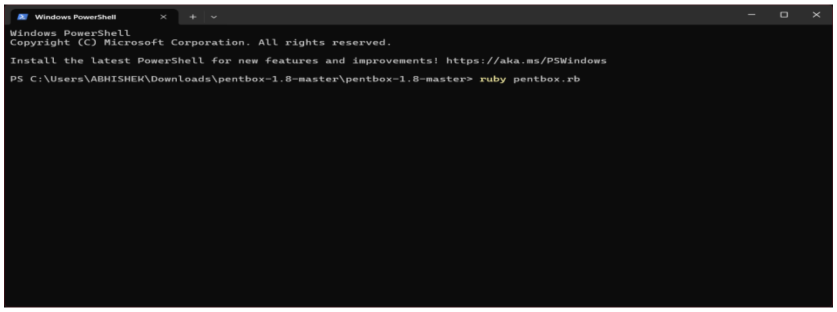


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ABHISHEK\Desktop\PROJECT MCA 3\pentbox-1.8-master\pentbox-1.8-master> |
```

Step4: After opening the powershell Window ,

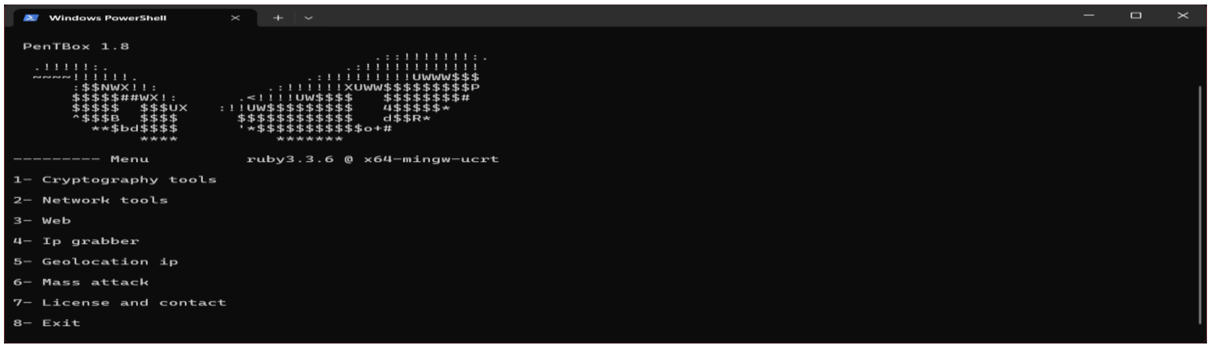
We have to execute the ruby command to launch pent box 1.8

- Command usage: ruby pentbox.rb



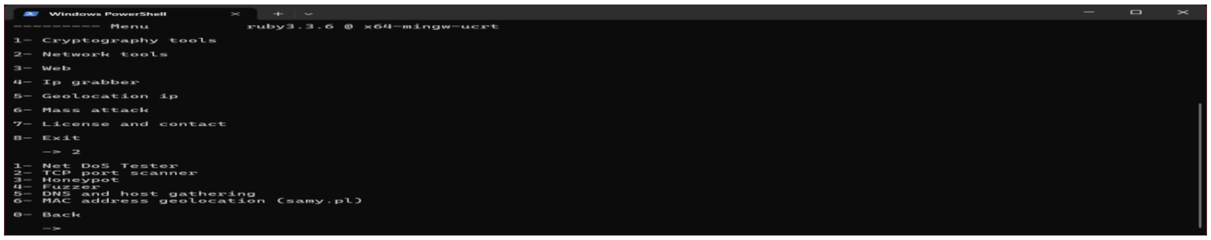
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ABHISHEK\Downloads\pentbox-1.8-master\pentbox-1.8-master> ruby pentbox.rb
```

Step5:The Pentbox 1.8 is launched which contains a numerous options as shown in below picture.



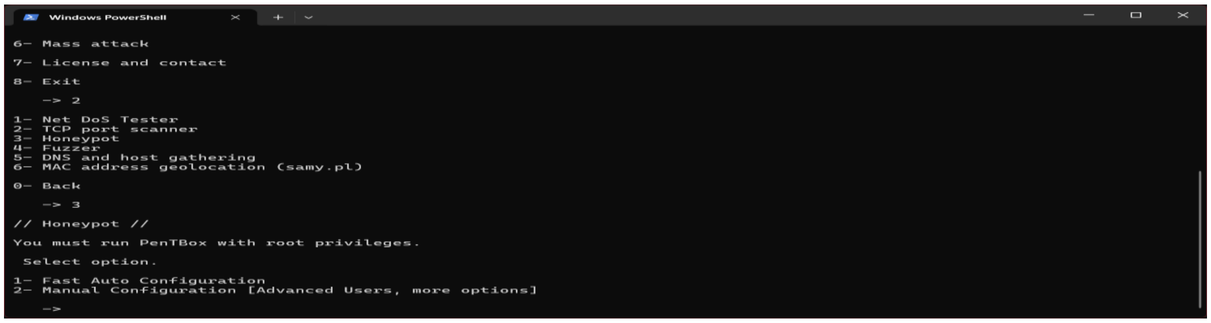
```
 PenTBox 1.8
                                     .::!!!!!!!:.
  .!!!!!:.                        .:!!!!!!!!!!!!
  ~~~~!!!!!!.                 .:!!!!!!!!!UWWW$$$
      :$$NWX!!:           .:!!!!!!XUWW$$$$$$$$$P
      $$$$$##WX!:      .<!!!!UW$$$$   $$$$$$$$#
      $$$$$  $$$UX   :!!UW$$$$$$$$$   4$$$$$*
      ^$$$B  $$$$     $$$$$$$$$$$$   d$$R*
        **$bd$$$$      '*$$$$$$$$$$$o+#
             ****          *******
---------- Menu          ruby3.3.6 @ x64-mingw-ucrt

1- Cryptography tools

2- Network tools

3- Web

4- Ip grabber

5- Geolocation ip

6- Mass attack

7- License and contact

8- Exit
```

Step6: Choose option 2 which is Network tools. It contains the various options including honeypot.



```
---------- Menu          ruby3.3.6 @ x64-mingw-ucrt
1- Cryptography tools
2- Network tools
3- Web
4- Ip grabber
5- Geolocation ip
6- Mass attack
7- License and contact
8- Exit
   -> 2
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
   ->
```

Step7: Now Choose option 3 which is for Honeypot detection system.



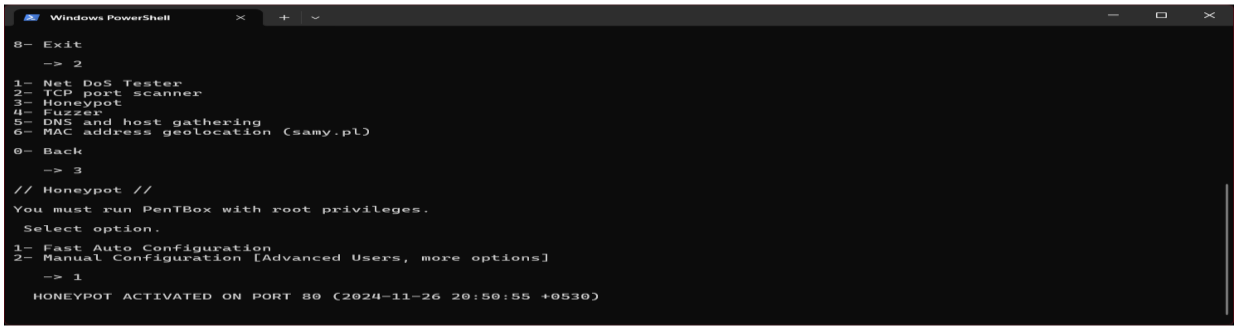
```
6- Mass attack

7- License and contact

8- Exit

   -> 2

1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)

0- Back

   -> 3

// Honeypot //

You must run PenTBox with root privileges.

 Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

   ->
```

Step8: after selecting the option Honeypot

It has two configurations :

1. The first one is Fast Auto Configuration
2. The Second one is Manual Configuration which includes:
    a. Which port number we have to attack
    b. Message which is optional
    c. Do we want to save the log file
    d. Enable/disable Alarm mode (beep sound)

The Honeypot is best work on Fast auto which activates on port 80(Https).

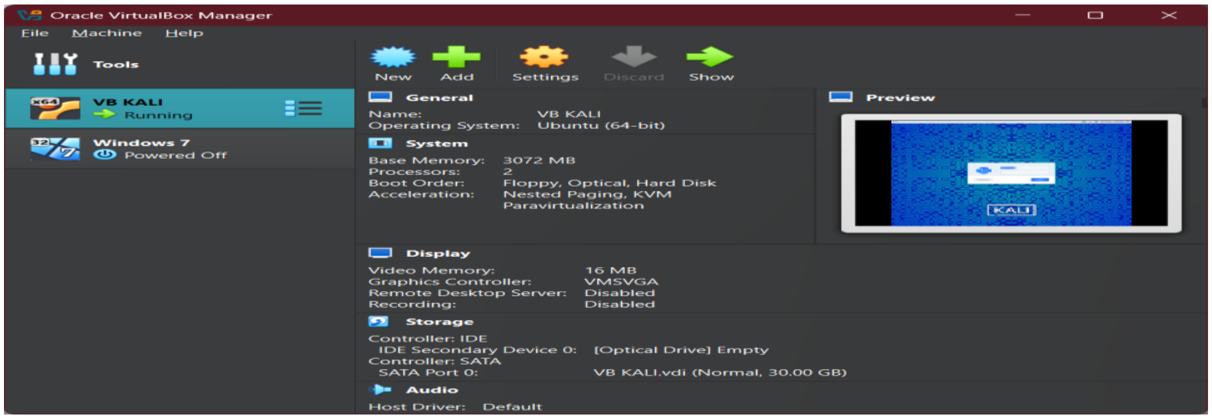


```
8- Exit

   -> 2

1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)

0- Back

   -> 3

// Honeypot //

You must run PenTBox with root privileges.

 Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

   -> 1

  HONEYPOT ACTIVATED ON PORT 80 (2024-11-26 20:50:55 +0530)
```

Step9: The honeypot PowerShell window is open at furthermost…

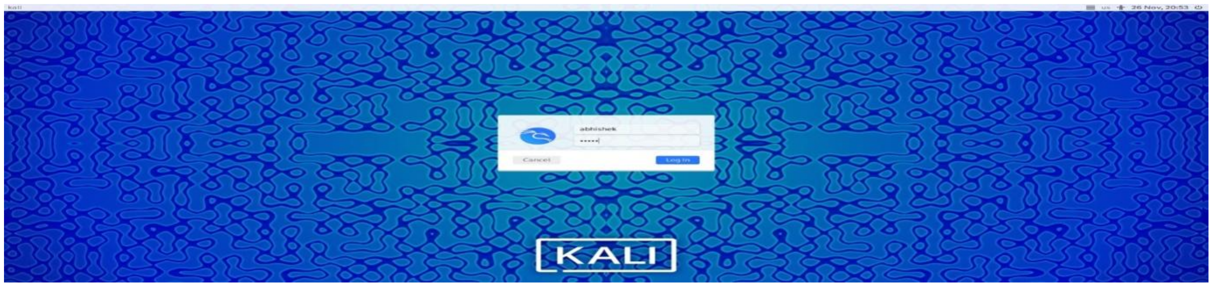
Step10: Now we have to open VirtualBox which is used for virtualization to operate Linux based OS

Which is Kali Linux.

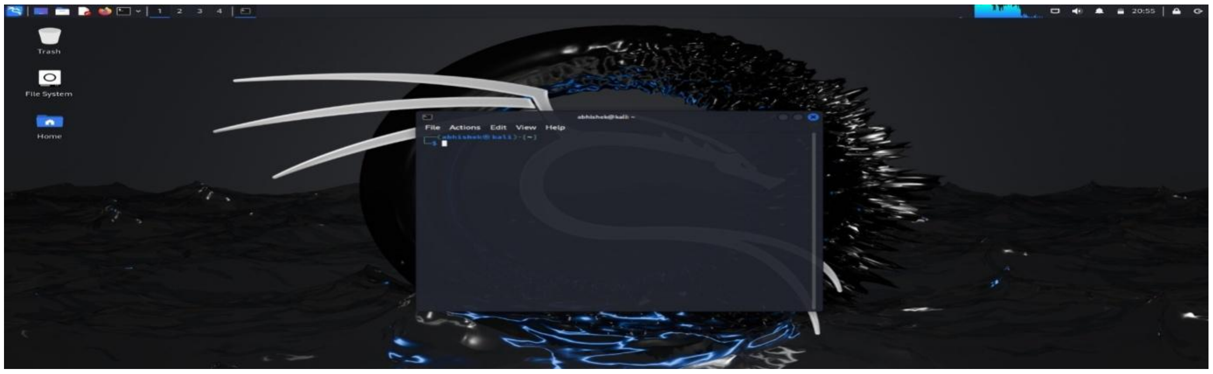


```
General
Name:                 VB KALI
Operating System:     Ubuntu (64-bit)
System
Base Memory:      3072 MB
Processors:       2
Boot Order:       Floppy, Optical, Hard Disk
Acceleration:     Nested Paging, KVM
                  Paravirtualization
Display
Video Memory:              16 MB
Graphics Controller:       VMSVGA
Remote Desktop Server:     Disabled
Recording:                 Disabled
Storage
Controller: IDE
  IDE Secondary Device 0:   [Optical Drive] Empty
Controller: SATA
  SATA Port 0:              VB KALI.vdi (Normal, 30.00 GB)
Audio
Host Driver:   Default
```

Step11: Now the VirtualBox is open. We have to double click on "VB KALI" button to start our kali Linux. The Kali Linux OS starts and is ready to use.



Step12:Login through Username and Password To enter Kali Linux. Then the Kali Linux Windows open up.

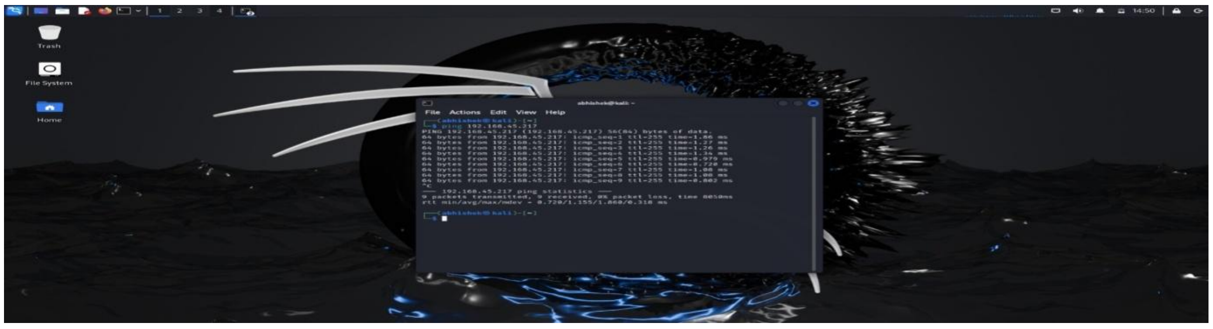


```
┌──(abhishek㉿kali)-[~]
└─$ 
```

Step13: Then we have to open the Terminal . The Terminal Opens up as standard user.

Step14:Then We have to type Ping in the terminal and hit enter.

- Command usage: Ping <Ip-address>
- The ping command is used to test the network connectivity between your device and a specified IP address or hostname.

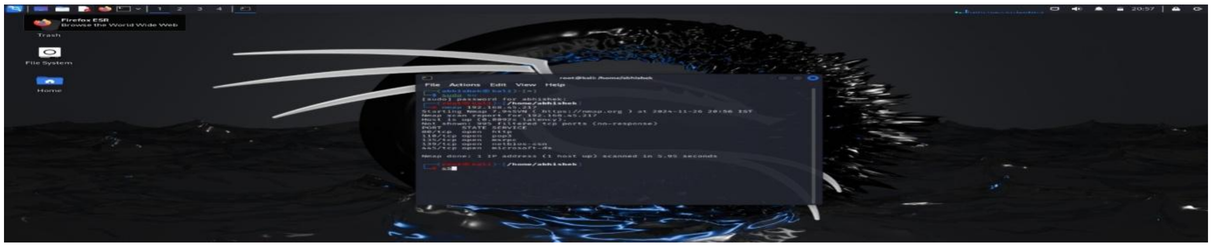


```
┌──(abhishek㉿kali)-[~]
└─$ ping 192.168.45.217
PING 192.168.45.217 (192.168.45.217) 56(84) bytes of data.
64 bytes from 192.168.45.217: icmp_seq=1 ttl=255 time=1.86 ms
64 bytes from 192.168.45.217: icmp_seq=2 ttl=255 time=1.27 ms
64 bytes from 192.168.45.217: icmp_seq=3 ttl=255 time=1.26 ms
64 bytes from 192.168.45.217: icmp_seq=4 ttl=255 time=1.34 ms
64 bytes from 192.168.45.217: icmp_seq=5 ttl=255 time=0.979 ms
64 bytes from 192.168.45.217: icmp_seq=6 ttl=255 time=0.720 ms
64 bytes from 192.168.45.217: icmp_seq=7 ttl=255 time=1.08 ms
64 bytes from 192.168.45.217: icmp_seq=8 ttl=255 time=1.08 ms
64 bytes from 192.168.45.217: icmp_seq=9 ttl=255 time=0.802 ms
^C
--- 192.168.45.217 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8050ms
rtt min/avg/max/mdev = 0.720/1.155/1.860/0.318 ms

┌──(abhishek㉿kali)-[~]
└─$ 
```

Step15: Then we have to find port numbers of the target Machine. We do this by using Nmap command.

- Command usage: nmap<Ip-address>

- The nmap (Network Mapper) command is a powerful tool used for network discovery, security auditing, and port scanning. It provides detailed information about a target host or network.



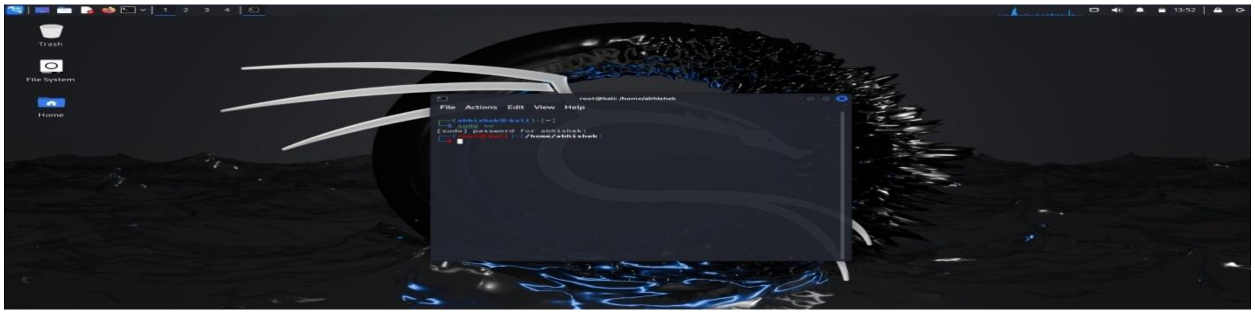
```
┌──(abhishek㉿kali)-[~]
└─$ sudo su
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# nmap 192.168.45.217
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-26 20:56 IST
Nmap scan report for 192.168.45.217
Host is up (0.0092s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT    STATE SERVICE
80/tcp  open  http
110/tcp open  pop3
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 5.95 seconds

┌──(root㉿kali)-[/home/abhishek]
└─# 
```

Step16:After port scanning, we have to launch Dos Attack with using Hping3 Command.

To launch the hping3 Command we have to go to Root User

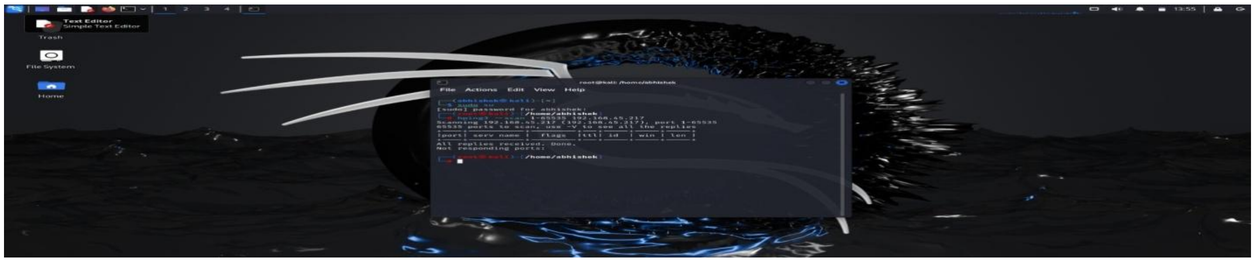
- Command Usage: sudosu

  The sudosu command is used on Linux systems to switch to the root user (administrator) account, granting you elevated privileges for performing system-level tasks.



```
┌──(abhishek㉿kali)-[~]
└─$ sudo su
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# 
```

- hping3 –help (to check whether the command is working or not).
- The hping3 tool is a versatile packet crafting tool often used for network testing and security auditing. The --scan option is used to perform a **port scan** across a specified range of ports on the target IP address.
- Explanation of the Command:
- hping3: Invokes the tool.
- --scan 1-65535: Scans all TCP ports from 1 to 65535. You can specify a smaller range if desired, e.g., --scan 20-100.
- <IP_address>: Target IP address.

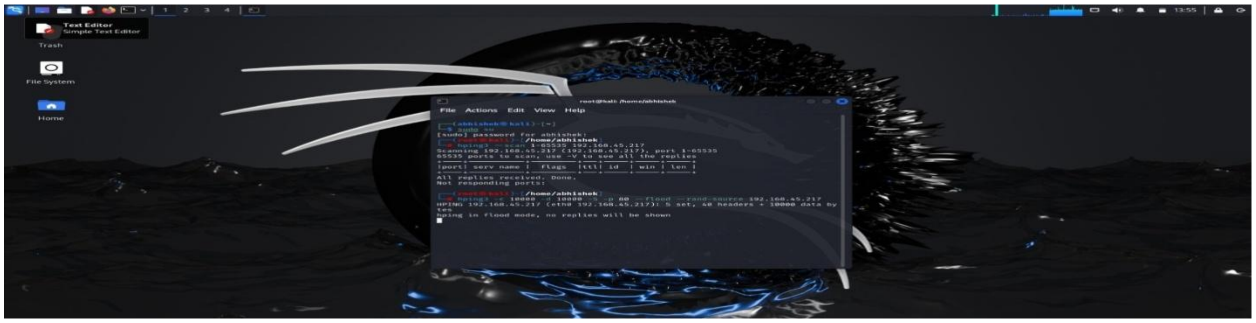


```
┌──(abhishek㉿kali)-[~]
└─$ sudo su
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# hping3 --scan 1-65535 192.168.45.217
Scanning 192.168.45.217 (192.168.45.217), port 1-65535
65535 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
All replies received. Done.
Not responding ports:
┌──(root㉿kali)-[/home/abhishek]
└─# 
```

Step17:Now we have to Launch Dos using Hping3

- Command Usage:
  hping3 -c 10000 -d 10000 -S -p 80 --flood --rand-source <IP_address>

- **Explanation of the Options:**
- **-c 10000:**
- Sends 10,000 packets before stopping.
- If omitted, hping3 will send packets indefinitely (especially with --flood).
- **-d 10000:**
- Sets the data size of each packet to 10,000 bytes (10 KB).
- Larger packets increase the load on the target.
- **-S:**
- Sends TCP SYN packets (commonly used in SYN flood attacks or connection testing).
- A SYN packet is the first step in a TCP handshake, simulating connection requests.
- **-p 80:**
- Targets port 80 (HTTP service).
- Replace 80 with any other port you want to test.
- **--flood:**
- Sends packets as fast as possible without waiting for replies.
- Ignores -c if set and continues until interrupted.
- **--rand-source:**
- Randomizes the source IP address for each packet.
- Simulates traffic from multiple sources to evaluate how the system handles distributed attacks.
- **<IP_address>:**
- The target IP address to send the packets to.
- **Purpose:**
- **Network Load Testing:** Simulates heavy traffic to observe how the target system or network responds.
- **Stress Testing:** Evaluates server performance under high traffic.
- **DoS Simulation:** Can mimic Denial of Service (DoS) conditions to test defences.

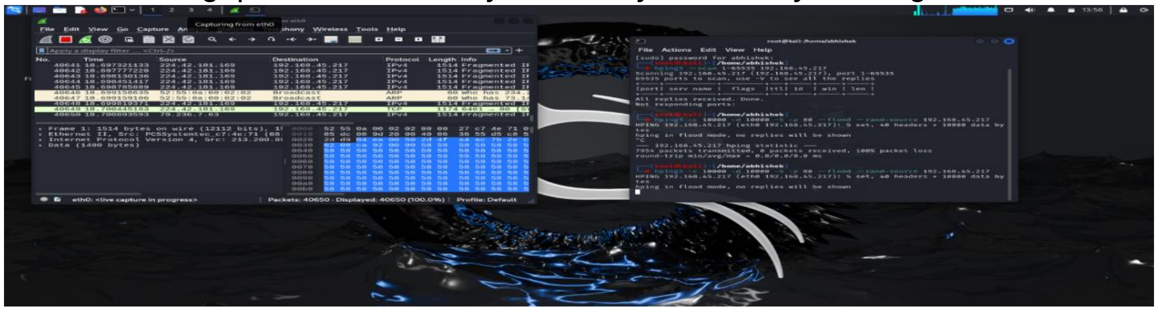


```
┌──(abhishek㉿kali)-[~]
└─$ sudo su
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# hping3 --scan 1-65535 192.168.45.217
Scanning 192.168.45.217 (192.168.45.217), port 1-65535
65535 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
All replies received. Done.
Not responding ports:

┌──(root㉿kali)-[/home/abhishek]
└─# hping3 -c 10000 -d 10000 -S -p 80 --flood --rand-source 192.168.45.217
HPING 192.168.45.217 (eth0 192.168.45.217): S set, 40 headers + 10000 data by
tes
hping in flood mode, no replies will be shown
```

The hping3 in flood mode is started. No replies is shown.

Step18: To track whether the Dos is working or not we use Wireshark as a packet sniffer tool.

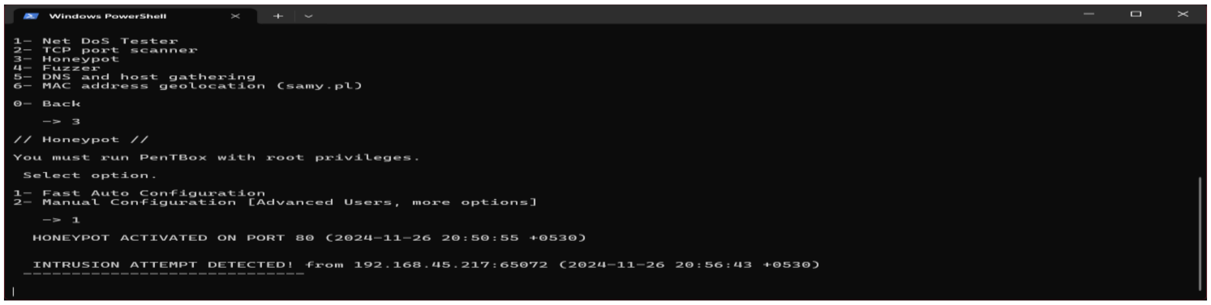
- Wireshark is a powerful network protocol analyzer that captures and displays data packets traveling through a network in real-time. It's widely used for network troubleshooting, performance analysis, and cybersecurity investigations.



```
No.   Time            Source               Destination      Protocol Length Info
40641 18.697321133    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40642 18.697777228    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40643 18.698130136    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40644 18.698451417    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40645 18.698785889    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40646 18.699158635    52:55:0a:00:02:02    Broadcast        ARP        60 Who has 234.
40647 18.699159106    52:55:0a:00:02:02    Broadcast        ARP        60 Who has 73.1
40648 18.699819371    224.42.181.169       192.168.45.217   IPv4     1514 Fragmented IP
40649 18.700445183    224.42.181.169       192.168.45.217   TCP      1174 6401 → 80 [S
40650 18.700803593    79.236.7.63          192.168.45.217   IPv4     1514 Fragmented IP

› Frame 1: 1514 bytes on wire (12112 bits), 1
› Ethernet II, Src: PCSSystemtec_c7:4e:71 (08
› Internet Protocol Version 4, Src: 213.200.8
› Data (1480 bytes)

eth0: <live capture in progress>    Packets: 40650 · Displayed: 40650 (100.0%)    Profile: Default
```

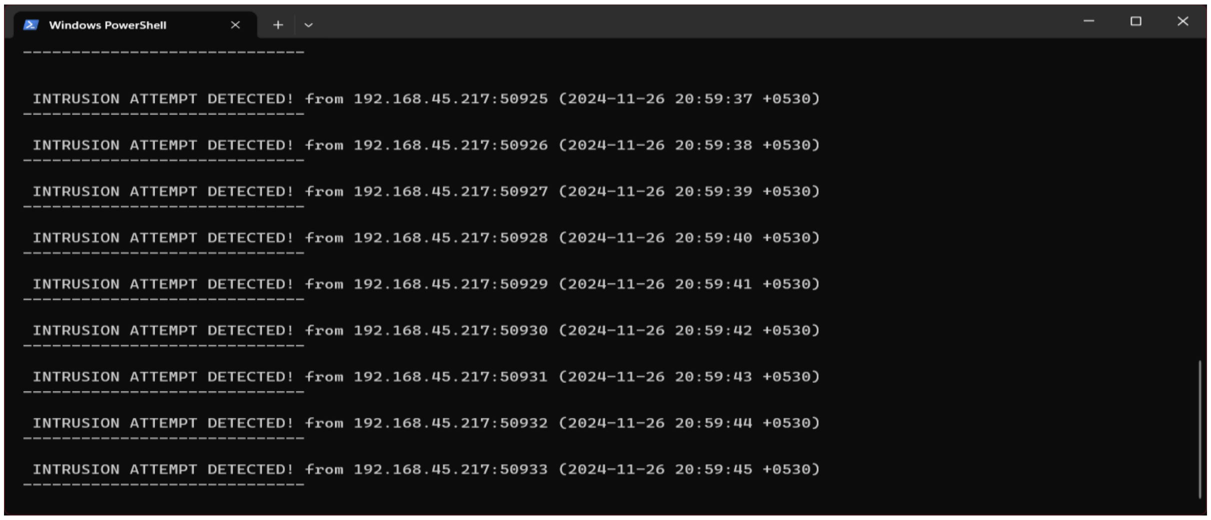
```
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# hping3 --scan 1-65535 192.168.45.217
Scanning 192.168.45.217 (192.168.45.217), port 1-65535
65535 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
All replies received. Done.
Not responding ports:

┌──(root㉿kali)-[/home/abhishek]
└─# hping3 -c 10000 -d 10000 -S -p 80 --flood --rand-source 192.168.45.217
HPING 192.168.45.217 (eth0 192.168.45.217): S set, 40 headers + 10000 data by
tes
hping in flood mode, no replies will be shown
^C
--- 192.168.45.217 hping statistic ---
7954 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(root㉿kali)-[/home/abhishek]
└─# hping3 -c 10000 -d 10000 -S -p 80 --flood --rand-source 192.168.45.217
HPING 192.168.45.217 (eth0 192.168.45.217): S set, 40 headers + 10000 data by
tes
hping in flood mode, no replies will be shown
```

Step 19: Then we open the PowerShell window in which the honeypot is running . an voila! It detects the intrusion on the target machine Ip.



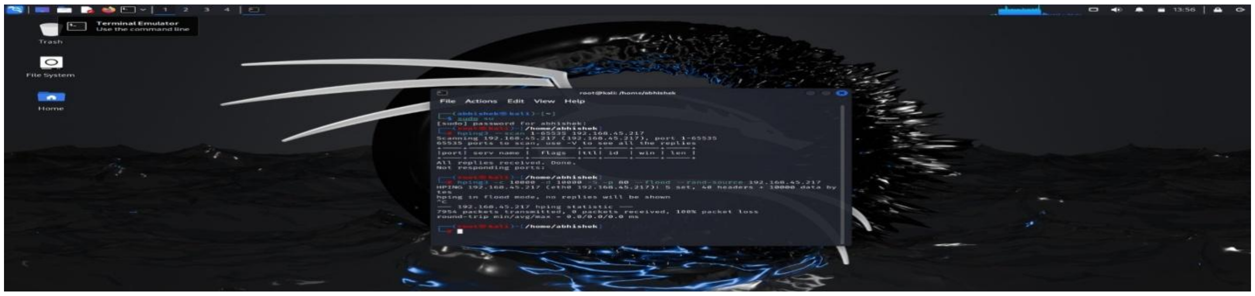
```
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)

0- Back

   -> 3

// Honeypot //

You must run PenTBox with root privileges.

 Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

   -> 1

  HONEYPOT ACTIVATED ON PORT 80 (2024-11-26 20:50:55 +0530)

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:65072 (2024-11-26 20:56:43 +0530)
 -----------------------------
```

And it detects the intrusion continuously (with a beep sound in manual mode).



```
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50925 (2024-11-26 20:59:37 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50926 (2024-11-26 20:59:38 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50927 (2024-11-26 20:59:39 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50928 (2024-11-26 20:59:40 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50929 (2024-11-26 20:59:41 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50930 (2024-11-26 20:59:42 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50931 (2024-11-26 20:59:43 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50932 (2024-11-26 20:59:44 +0530)
 -----------------------------

  INTRUSION ATTEMPT DETECTED! from 192.168.45.217:50933 (2024-11-26 20:59:45 +0530)
 -----------------------------
```

Step 20: Now to stop the intrusion we have to stop flooding the network in terminal in Kali Linux.



```
┌──(abhishek㉿kali)-[~]
└─$ sudo su
[sudo] password for abhishek:
┌──(root㉿kali)-[/home/abhishek]
└─# hping3 --scan 1-65535 192.168.45.217
Scanning 192.168.45.217 (192.168.45.217), port 1-65535
65535 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
All replies received. Done.
Not responding ports:

┌──(root㉿kali)-[/home/abhishek]
└─# hping3 -c 10000 -d 10000 -S -p 80 --flood --rand-source 192.168.45.217
HPING 192.168.45.217 (eth0 192.168.45.217): S set, 40 headers + 10000 data by
tes
hping in flood mode, no replies will be shown
^C
--- 192.168.45.217 hping statistic ---
7954 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(root㉿kali)-[/home/abhishek]
└─#
```

Step21: the honeypot detection is stopped with CTRL+C command in PowerShell window.

The Honeypot intrusion detection works perfectly.