System Defence Against DDoS: Detection, Mitigation, and Recovery

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Naman Goyal(112215120)
Tanishq Ingole(112215076)
Deepesh Patil(112215055)
Saksham Dharmik(112215157)

Under the Supervision of: Dr. Kaptan Singh

Semester: V

Department of Computer Science and Engineering

Indian Institute of Information Technology, Pune
(An Institute of National Importance by an Act of Parliament)

November 2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“**System Defence Against DDoS: Detection, Mitigation, and Recovery**”** submitted by **Naman Goyal** bearing the **MIS No 112215120**, **Deepesh Patil** bearing the **MIS No 112215055**, **Saksham Dharmik** bearing the **MIS No 112215157,** and **Tanishq Ingole** bearing the **MIS No 112215076** in completion of his/her project work under the guidance of **Dr. Kaptan Singh** is accepted for the project report submission in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in the **Department of Computer Science and Engineering** Indian Institute of Information Technology, Pune (IIIT Pune), during the academic year **2023-24**.

Dr. Kaptan Singh	**Dr. Bhupendra Singh**
Project Guide	Head of the Department
Assistant Professor	Assistant Professor
Department of CSE	Department of CSE
IIIT Pune	IIIT Pune

Project Viva-voce held on 06-11-2024

Undertaking for Plagiarism

We **Naman Goyal , Saksham Dharmik , Deepesh Patil ,** and **Tanishq Ingole** solemnly declare that the research work presented in the **report** titled "System Defence Against DDoS: Detection, Mitigation, and Recovery**"** is solely **our** research work with no significant contribution from any other person. Small help wherever taken has been duly acknowledged, and that complete report has been written by **us**. I understand the zero-tolerance policy of the **Indian Institute of Information Technology, Pune** towards plagiarism. Therefore, **we** declare that no portion of our **report** has been plagiarized and any material used as reference is properly referred to or cited. We undertake that if we are found guilty of any formal plagiarism in the above titled thesis even after award of the degree, the Institute reserves the right to withdraw/revoke my **B.Tech.** degree.

Students'/ Student's Name and Signature with Date

Conflict of Interest

Manuscript title: System Defence Against DDoS: Detection, Mitigation, and Recovery

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements) or non-financial interest (such as personal or professional relationships, affiliations, knowledge, or beliefs) in the subject matter or materials discussed in this manuscript.

Students'/ Student's Name and Signature with Date

ACKNOWLEDGEMENT

This project would not have been possible without the help and cooperation of many. We would like to thank the people who helped us directly and indirectly in the completion of this project work.

First and foremost, I would like to express my gratitude to our honorable director, **Dr. Prem Lal Patel**, for providing his kind support in various aspects. I would like to express my gratitude to my project guide, **Dr. Kaptan Singh**, **Department of CSE**, for providing excellent guidance, encouragement, inspiration, and constant and timely support throughout this **B.Tech. Project.** I would like to express my gratitude to the **Head of Department, Dr. Bhupendra Singh**, **Department of CSE** , for providing his kind support in various aspects. I would also like to thank all the faculty members in the **Department of CSE** and my classmates for their steadfast and strong support and engagement with this project.

Abstract

Distributed Denial-of-Service (DDoS) attacks are a significant threat to cloud infrastructures, causing service disruptions, compromised user experience, and potential financial losses. These attacks exploit vulnerabilities by overwhelming resources, leading to system downtime and degraded performance. This project proposes an advanced, **ML/DL**-driven autonomous defense system that offers real-time detection, mitigation, and recovery from DDoS attacks on cloud systems. By incorporating machine learning and deep learning models, our solution efficiently identifies anomalies, dynamically filters malicious traffic through Web Application Firewalls (WAFs), and employs honeypots for early DDoS pattern identification. To further enhance resilience, auto-scaling and load balancing mechanisms are implemented to support continuous operation during attacks. The **Incremental Learning Model** allows the system to adapt seamlessly to emerging threats without retraining, providing a self-healing, robust, and cost-effective solution that strengthens the security and reliability of cloud environments.

Keywords: DDoS Protection, Incremental Learning Model (ML+DL), WAF, Honeypots , Load Balancing

TABLE OF CONTENTS

List of Figures / Symbols / Nomenclature

Chapter 1
Introduction

In today's digital age, malware poses a significant threat to security systems worldwide. The landscape of cybersecurity is constantly evolving, with new and increasingly sophisticated forms of malware emerging on a regular basis. These malicious software variants, which include viruses, worms, Trojans, ransomware, and spyware, are designed to exploit vulnerabilities in systems, steal sensitive information, disrupt operations, or inflict financial damage. The ramifications of such attacks can be extensive, affecting not only the targeted organizations but also their customers, partners, and the broader economy.

As cyber threats continue to grow in complexity and frequency, the need for advanced analytical techniques becomes imperative. Traditional methods of malware analysis—often manual and resource-intensive—can be inadequate for the demands of modern cyber warfare. Malware analysts face significant challenges, including the requirement for specialized expertise, a thorough understanding of malware behavior, and the timely application of best practices. This makes it difficult for organizations to adequately protect their infrastructure against evolving threats.

Cuckoo Sandbox stands out as an innovative and effective solution in the realm of automated malware analysis. By enabling cybersecurity experts to analyze malware in a controlled, isolated environment, Cuckoo Sandbox mitigates the risks associated with manual analysis while providing rich insights into malware behavior. Its open-source nature not only allows for broad adaptability and community contributions but also empowers organizations with the ability to customize their malware analysis capabilities according to their specific needs.

This report aims to provide a comprehensive overview of the approach adopted for implementing Cuckoo Sandbox, detailing the methodology followed, the setup process, and the results derived from its usage. The findings presented in this report will illustrate how Cuckoo Sandbox enhances the capabilities of cybersecurity professionals in identifying, analyzing, and mitigating malware threats effectively. Additionally, the report will discuss the significance of automated analysis in ensuring that organizations are well-prepared to address the ever-evolving landscape of cyber threats.

Through an in-depth exploration of Cuckoo Sandbox, this report seeks to contribute to the broader discourse on enhancing cybersecurity measures and fostering a proactive approach to malware defense.

1.2 Motivation of the Work

With the rapid shift towards cloud computing, organizations increasingly rely on cloud infrastructures to store data, deliver services, and manage critical applications. However, this dependency has also made cloud environments prime targets for Distributed Denial-of-Service (DDoS) attacks, which can disrupt operations, compromise data security, and lead to substantial financial and reputational losses. Traditional DDoS protection methods are often inadequate due to their limited ability to respond dynamically to evolving attack patterns and high traffic volumes.

The motivation behind this work is to develop a robust, adaptive, and autonomous DDoS defense solution tailored for cloud environments. By leveraging advanced machine learning (ML) and deep learning (DL) techniques, the project seeks to enhance the detection and mitigation of DDoS attacks in real time. The goal is to reduce dependency on manual intervention, provide a self-healing mechanism, and ensure uninterrupted service delivery. This project aspires to bridge the gap between current DDoS defense limitations and the increasing need for scalable, intelligent protection solutions in cloud infrastructure.

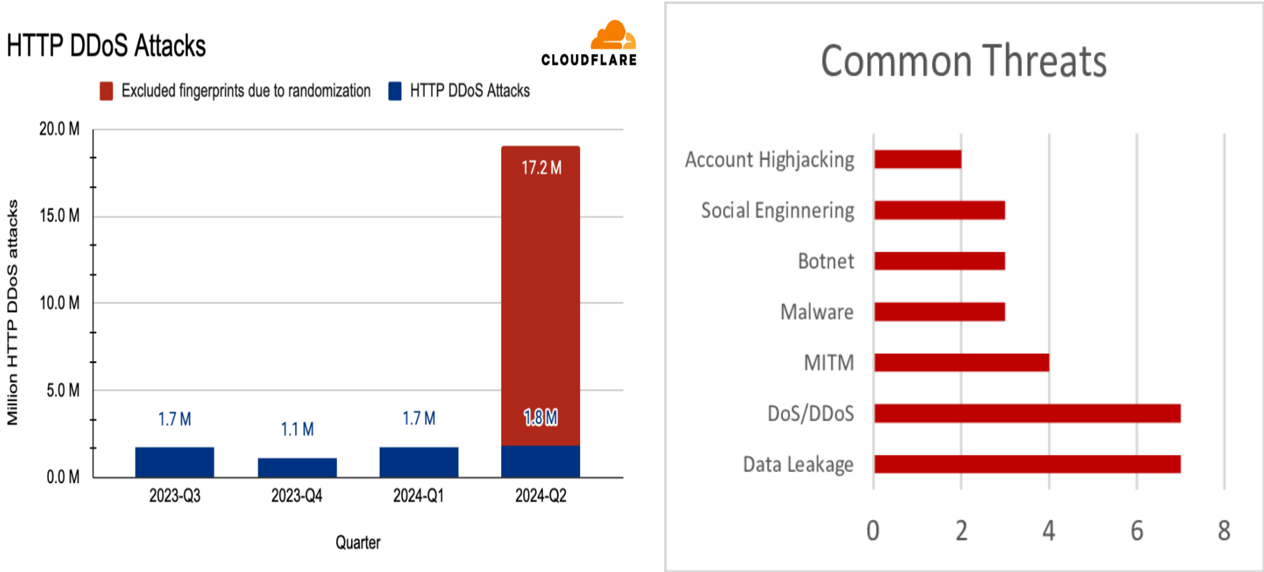


Fig.1 Common Threat Rates

1.3 Literature Review

1. Evolution of DDoS Attacks and Their Impact

DDoS attacks have evolved significantly, increasing in complexity and frequency, as well as in the scale of damage they can inflict. Cloud infrastructure, while highly resilient, remains a prime target due to its extensive accessibility and high network capacity. The transition from simple, volumetric attacks to more sophisticated, multi-vector assaults complicates the detection and mitigation process. Attackers exploit cloud vulnerabilities, using botnets and spoofing techniques to overwhelm servers and disrupt services. Studies have shown that the effects of these attacks extend beyond downtime, impacting user trust and causing financial loss. Research by Phan and Park (2019) highlights that a successful DDoS attack on a cloud system can degrade service reliability, causing cascading failures across interconnected resources. Addressing this growing threat demands advanced, real-time detection methods capable of adapting to evolving attack patterns and protecting against both known and emerging threats.

2. Existing DDoS Mitigation Techniques and Their Limitations

Traditional DDoS mitigation techniques, such as blacklisting IPs and rate limiting, have been widely used to block malicious traffic. However, these approaches often lack the intelligence to differentiate between legitimate and illegitimate traffic, especially in large-scale cloud environments. Signature-based detection systems, while effective against known attack patterns, struggle to address novel or polymorphic DDoS attacks. Chen et al. (2018) argue that these conventional methods are insufficient for modern cloud systems due to their static nature, which prevents them from adapting to new attack types. As a result, recent research has shifted toward developing dynamic mitigation approaches that can leverage machine learning to analyze traffic patterns in real time. These ML-based approaches aim to improve detection accuracy by learning from historical data, although challenges remain in managing the volume and variety of cloud traffic data.

3. Machine Learning for DDoS Detection in Cloud Systems

Machine learning has emerged as a promising solution for DDoS detection in cloud systems. ML algorithms, particularly anomaly detection models, can analyze vast amounts of traffic data to detect unusual patterns that may indicate a DDoS attack. According to Yin, Zhang, and Yang (2018), ML models have shown considerable potential in identifying traffic anomalies by distinguishing between regular fluctuations and malicious activity. Supervised learning algorithms, such as Support Vector Machines (SVM) and Decision Trees, are commonly used to classify benign and malicious traffic. However, their reliance on labeled data can limit their effectiveness, as DDoS attacks continuously evolve. Therefore, unsupervised and semi-supervised learning models, which do not require labeled data, are increasingly favored in adaptive DDoS detection frameworks for cloud applications.

4. Deep Learning Approaches for Enhanced Detection and Response

Deep learning (DL) offers enhanced capabilities for DDoS detection by enabling complex, multi-layered analysis of network traffic. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are particularly useful for recognizing intricate traffic patterns and time-dependent anomalies. Recent studies, including those by Sahi et al. (2017), highlight that DL models can improve detection accuracy in high-volume cloud environments by capturing complex attack signatures that simpler ML models may miss. DL models, however, require extensive computational resources and training time, which can limit their applicability in real-time systems. To address this, hybrid systems that combine DL models with lightweight ML algorithms are being explored to balance accuracy and speed, allowing cloud systems to maintain resilience while efficiently detecting and mitigating DDoS threats.

5. Real-Time DDoS Detection with Autonomous Defense Systems

The concept of autonomous defense systems for real-time DDoS mitigation is gaining traction in cloud security research. These systems integrate machine learning and automation to detect, respond to, and recover from DDoS attacks without manual intervention. Autonomous DDoS defense involves the use of self-healing architectures that automatically adjust resources, such as load balancing and auto-scaling, in response to attack patterns. Mahmoud, Ouda, and Capretz (2013) propose using self-adapting architectures with incremental learning capabilities to maintain defense effectiveness against evolving threats. This approach reduces operational costs and human oversight, making it highly scalable for large cloud providers. The integration of incremental learning is particularly beneficial, as it enables systems to adapt to new attack vectors over time, reducing the need for frequent retraining.

6. Effectiveness of Honeypots and Multi-layered Defense Mechanisms

Honeypots, or decoy systems, are increasingly being used as part of multi-layered defense strategies to detect early signs of DDoS attacks. By diverting malicious traffic to honeypots, cloud systems can analyze attacker behavior and adapt defenses accordingly. Studies by Yin et al. (2018) suggest that honeypots are highly effective in identifying DDoS attack vectors early, allowing cloud infrastructure to preemptively filter harmful traffic before it reaches critical resources. Combined with multi-layered defense mechanisms, including firewalls, WAFs, and network segmentation, honeypots provide an additional layer of security. This layered approach not only enhances detection accuracy but also increases the overall resilience of cloud systems. By coupling these with adaptive ML models, cloud systems achieve a comprehensive, responsive, and resilient defense against sophisticated DDoS attacks.

Author(s)	Year	Focus	Methodology	Key Findings
Phan & Park	2019	Impact of DDoS on Cloud Systems	Empirical Analysis	DDoS attacks cause service degradation, highlighting need for real-time adaptive defense
Chen et al.	2018	Limitations of traditional DDoS mitigation	Comparative study	Traditional methods are insufficient for evolving attacks; ML-based solutions offer improvement
Yin, Zhang, & Yang	2018	ML for DDoS anomaly detection	Machine Learning (ML) models	ML effectively detects anomalies; unsupervised models show promise for dynamic cloud environments
Sahi et al.	2017	DL for enhanced DDoS detection in cloud	Deep Learning (DL) models	DL models capture complex patterns but require high resources; hybrid systems balance accuracy/speed
Mahmoud, Ouda, & Capretz	2013	Autonomous DDoS defense and incremental learning	Autonomous system proposal	Self-healing systems reduce manual oversight, enhancing scalability and adaptability to new threats
Yin et al.	2018	Honeypots and multi-layered defense	Case study on multi-layered setup	Honeypots and WAFs improve early detection and enhance resilience when integrated with ML defenses

1.4 Research Gap

1. Adaptability in DDoS Mitigation
Traditional DDoS defenses, like IP blacklisting and rate limiting, are often too rigid to handle advanced attack patterns. There is a need for adaptable systems that intelligently differentiate between legitimate and malicious traffic in real time.

2. Real-Time Detection and Resource Efficiency
Implementing machine learning (ML) and deep learning (DL) models for real-time DDoS detection is challenging due to the high computational costs. Efficient, lightweight models that offer fast, accurate responses in high-traffic environments are still underdeveloped.

3. Scalability and Incremental Learning
Most existing solutions lack scalability and do not support incremental learning, making it difficult to adapt to new attack patterns without retraining. Research is needed to create scalable, self-updating models that accommodate evolving threats with minimal resource impact.

4. Integration of Multi-Layered Defenses
Current DDoS systems often don't fully utilize multi-layered defenses, like Web Application Firewalls (WAFs) and honeypots, which can provide early detection and reduce attack impact. More research is needed to optimize these tools within a unified, adaptive architecture.

5. Early Detection and Preemptive Defense
There is a gap in proactive defense mechanisms that can identify early signs of DDoS attacks and preemptively adjust resources to mitigate potential damage. This area requires further exploration to enhance preventive measures in cloud security.

Chapter 2
Problem Statement

Cloud infrastructures are increasingly vulnerable to Distributed Denial-of-Service (DDoS) attacks, which can overwhelm servers with illegitimate traffic, causing service disruptions, downtime, and financial loss. These attacks exploit the accessibility of cloud systems and bypass traditional defenses, like IP blacklisting and rate limiting, which are often too static to counteract evolving, sophisticated DDoS tactics. The complexity of cloud environments further complicates detection, making it difficult to accurately distinguish between legitimate and malicious traffic in real-time. To address these challenges, there is a need for a resilient, autonomous DDoS protection system that leverages machine learning (ML) and deep learning (DL) models for accurate anomaly detection and dynamic traffic filtering. This project aims to develop such a system, incorporating incremental learning and multi-layered defense strategies to adapt continuously to new threats, ensuring reliable, scalable, and cost-effective security for cloud environments.

2.1. Research Objectives

The primary objective of this project is to develop a robust, autonomous DDoS protection system for cloud infrastructure that can detect, mitigate, and recover from DDoS attacks in real-time. Specific objectives include:

1. Implement Machine Learning and Deep Learning Models: Utilize ML and DL algorithms to accurately detect traffic anomalies, enhancing the system's ability to identify and filter malicious traffic efficiently.

2. Develop Adaptive Defense Mechanisms: Create a system capable of incremental learning, allowing it to adapt to new and evolving attack patterns without requiring complete retraining.

3. Integrate Multi-Layered Defense Strategies: Incorporate Web Application Firewalls (WAFs), honeypots, and load-balancing mechanisms to provide comprehensive, multi-layered protection that increases resilience against DDoS attacks.

4. Enable Real-Time Response and Scalability: Design the system to operate in real-time with minimal latency, ensuring scalability for high-traffic cloud environments and continuous availability during attack scenarios.

5 Minimize Operational Costs: Achieve cost-efficiency by reducing the need for manual intervention and automating the detection, mitigation, and recovery processes, making the system suitable for diverse cloud infrastructures.

These objectives collectively aim to enhance the security and reliability of cloud environments by providing an intelligent, self-adaptive DDoS defense system.

2.2. Methodology of the Work

The proposed DDoS (Distributed Denial of Service) detection system employs a hybrid approach combining traditional machine learning, deep learning, and incremental learning techniques. The system's architecture is designed to provide real-time detection capabilities while continuously adapting to new attack patterns. The **main server** handles incoming traffic and runs the detection algorithms, while the **backup server** remains on standby to take over in case the main server fails or is overwhelmed, ensuring continuous availability and reliability in mitigating DDoS attacks.

2.2.1 Data Collection and Feature Engineering

The system processes network traffic data characterized by the following key features:

- **pktcount**: Packet count metrics
- **bytecount**: Byte count statistics
- **flows**: Network flow information
- **pktrate**: Packet rate measurements
- **byteperflow**: Bytes per flow ratio
- **tx_kbps**: Transmission rate in kilobytes per second
- **rx_kbps**: Reception rate in kilobytes per second
- **tot_kbps**: Total throughput in kilobytes per second

Data preprocessing involves several crucial steps:

1. Categorical encoding using LabelEncoder for:

- Protocol information
- Source addresses (src)
- Destination addresses (dst)
- Switch identifiers

2. Feature normalization using StandardScaler:

```
X_scaled = self.scaler.fit_transform(np.nan_to_num(X, nan=0.0))
```

2.2.2 Model Architecture

The system implements a three-tier architecture:

1. Deep Learning Model (DDoS Detector)

```
class DDoSDetector(nn.Module):
    def __init__(self, input_dim):
        super(DDoSDetector, self).__init__()
        self.layer1 = nn.Linear(input_dim, 64)
        self.layer2 = nn.Linear(64, 32)
        self.layer3 = nn.Linear(32, 16)
        self.layer4 = nn.Linear(16, 2)
```

- Four-layer neural network with decreasing neuron counts
- Batch normalization for training stability
- ReLU activation functions
- Dropout (0.3) for regularization

2. Random Forest Classifier

```
self.rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    n_jobs=-1,
    random_state=42
)
```

- Ensemble of 100 decision trees
- Maximum depth of 10 for preventing overfitting
- Optimized leaf and split parameters

3. Incremental Random Forest

```
class IncrementalRandomForest:
    def __init__(self, n_estimators=10):
        self.model = RandomForestClassifier(
            n_estimators=n_estimators,
            warm_start=True
        )
```

- Supports continuous learning
- Warm start capability for model updates
- Dynamic estimator adjustment

2.2.3 Training Methodology

The training process follows a multi-phase approach:

1. Initial Training Phase

```
def train_initial_model(self, X, y):
    X_scaled = self.scaler.fit_transform(np.nan_to_num(X, nan=0.0))
    # Train Random Forest
    self.rf_model.fit(X_scaled, y)
    # Train Deep Learning model
    self.dl_model = DDoSDetector(input_dim)
    dataset = DDoSDataset(X_scaled, y)
  train_loader = DataLoader(dataset, batch_size=self.batch_size, shuffle=True)
```

2. Incremental Learning Phase

```
def update_model(self, X, y):
    X_scaled = self.scaler.transform(np.nan_to_num(X, nan=0.0))
    self.incremental_model.partial_fit(X_scaled, y)
```

2.2.4 Performance Metrics and Monitoring

The system continuously monitors several key metrics:

1. Accuracy Metrics

```
metrics = {
    'rf_accuracy': accuracy_score(y, rf_pred),
    'dl_accuracy': accuracy_score(y, dl_pred),
    'incremental_accuracy': accuracy_score(y, incr_pred),
    'ensemble_accuracy': accuracy_score(y, ensemble_pred)
}
```

2. False Positive Rate Analysis

```
def get_fpr(cm):
    tn, fp, fn, tp = cm.ravel()
    return fp / (fp + tn) if (fp + tn) > 0 else 0
```

2.2.5 System Optimization

The system incorporates several optimization techniques:

1. Batch Processing

```
chunk_size = 1000
for i in range(0, len(X_test), chunk_size):
    X_chunk = X_test[i:i+chunk_size]
    y_chunk = y_test[i:i+chunk_size]
```

2. Data Augmentation Using SMOTE

```
smote = SMOTE()
X_res, y_res = smote.fit_resample(df_improved.drop('label', axis=1), df_improved['label'])
```

2.2.6 Server Architecture

The server architecture of the proposed DDoS detection system consists of two primary components: the main server and the backup server.

1. **Main Server**:

This server is tasked with processing all incoming network traffic. It runs the detection algorithms, which utilize a combination of traditional machine learning, deep learning, and incremental learning techniques. The main server continuously analyzes traffic patterns to identify potential DDoS attacks in real-time. In the event of a detected attack, it implements mitigation strategies, such as traffic filtering and rate limiting, to maintain service availability. This server is crucial for handling the majority of traffic and ensuring prompt detection and mitigation.

2. **Backup Server**:

The backup server operates in a standby mode, closely monitoring the health of the main server. Should the main server experience a failure or become overwhelmed by traffic, the backup server automatically takes over, ensuring that detection and mitigation efforts continue without interruption. This server is also equipped with the same detection algorithms and is capable of learning from new attack patterns through incremental learning, keeping it synchronized with the main server's models.

In addition, the backup server utilizes Wake-on-LAN (WoL) technology, allowing it to remain in a low-power state while on standby and quickly "wake up" when needed. This power-efficient approach ensures that the backup server consumes minimal energy while not in use, as seen in the power consumption graph where Wake-on-LAN uses only 50% power. The increasing number of IT devices contributes to higher energy consumption, but solutions like Wake-on-LAN, which remotely wakes up devices from sleep states when needed, help minimize this energy usage, as demonstrated by an embedded system directly connected to the Internet for activating such devices [10].

Together, this architecture provides a robust and resilient framework for DDoS detection and mitigation. The integration of WoL in the backup server ensures that it can be quickly activated when needed while minimizing energy consumption during idle periods. This setup enhances the system's ability to adapt to evolving threats, ensures high availability for network services, and maintains efficient power usage without compromising performance.

2.2.7 Server Recovery

The recovery mechanism in the proposed DDoS detection system is designed to respond dynamically to excessive traffic conditions. This is achieved through a script that continuously monitors network traffic on the primary server. The key features of the recovery mechanism are outlined below:

1. Traffic Monitoring

The script continuously monitors the incoming network traffic on a specified network interface. It checks the number of bytes received over one-minute intervals and compares it against a predefined threshold. If the received traffic exceeds this threshold, the system interprets it as a potential DDoS attack.

2. Identification of Offending IP Addresses

Upon detecting excessive traffic, the script employs network analysis tools to identify the IP address responsible for the highest volume of incoming traffic. This is done using tcpdump, which captures packets and analyzes the source addresses of incoming requests.

3. Blocking Malicious Traffic

Once an offending IP address is identified, the script automatically blocks it using iptables, preventing any further requests from that IP. This action mitigates the immediate impact of the potential DDoS attack, allowing legitimate traffic to continue flowing to the server.

4. Network Interface Management

To further protect the server from ongoing excessive traffic, the script disconnects the network interface temporarily. After a brief pause, it reconnects the interface, ensuring that the system can recover from the attack and resume normal operations. This two-step process minimizes downtime and helps maintain service availability.

5. Logging and Notifications

The recovery script includes logging capabilities to document significant events, such as when excessive traffic is detected and which IP addresses are blocked. This logging facilitates post-event analysis and helps administrators understand the nature of the traffic patterns that triggered the recovery actions. Additionally, integrating a notification system could alert administrators in real-time when recovery actions are taken.

Chapter 3
Analysis and Design

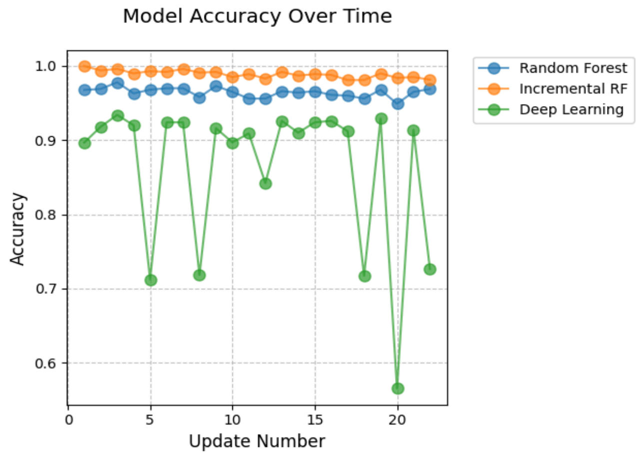


Fig.2 Model Accuracy Over Time

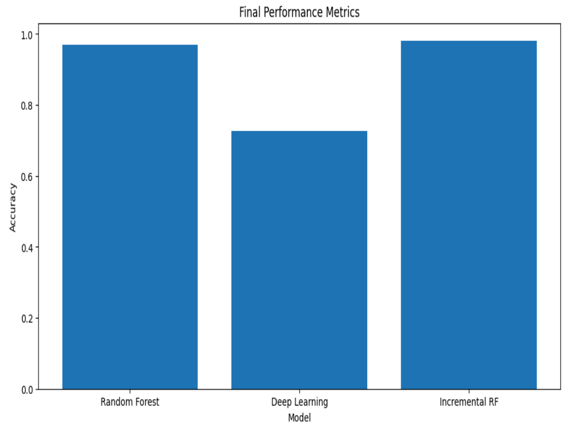


Fig.3 Final Performance Metrics

```
Final Performance Metrics:
Random Forest Accuracy: 0.9689
Deep Learning Accuracy: 0.7261
Incremental RF Accuracy: 0.9816
Ensemble Accuracy: 0.9712

Final False Positive Rates:
Random Forest FPR: 0.0456
Deep Learning FPR: 0.4049
Incremental RF FPR: 0.0228
```

Fig.4 Numeric Values of Accuracy and FPR

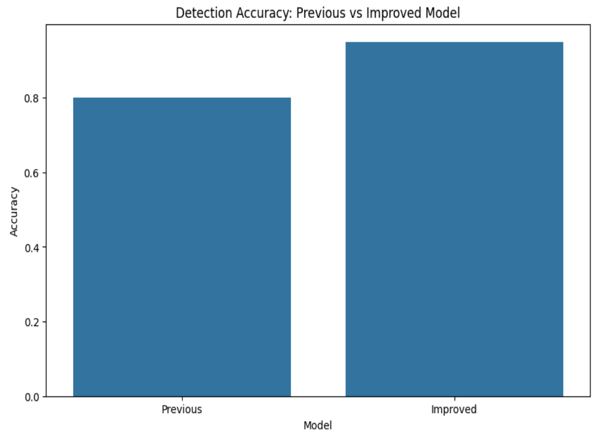
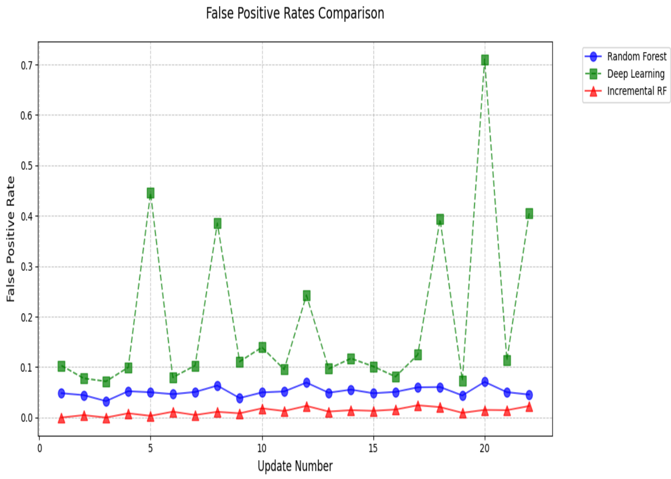


Fig.5 Previous vs Improved Model



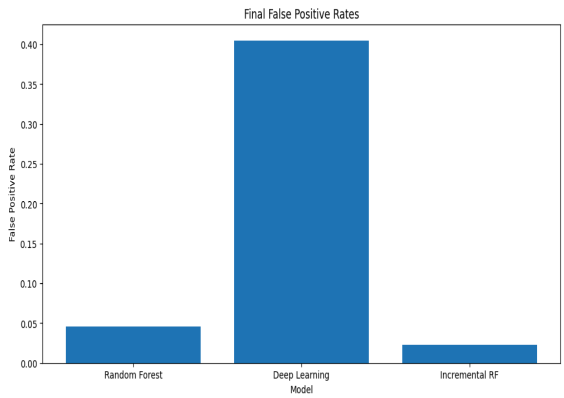


Fig.6 False Positive Rates

Fig.7 FPR Histogram

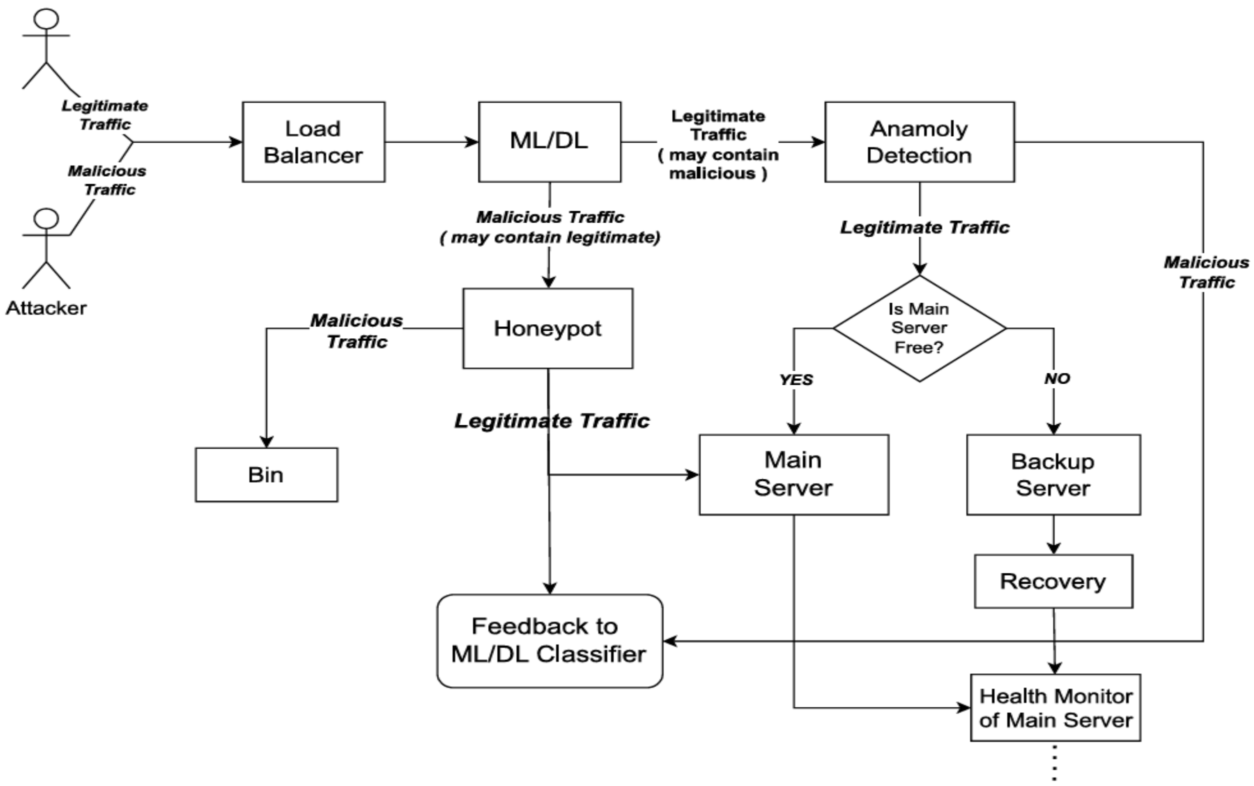


Fig.8 System Architecture

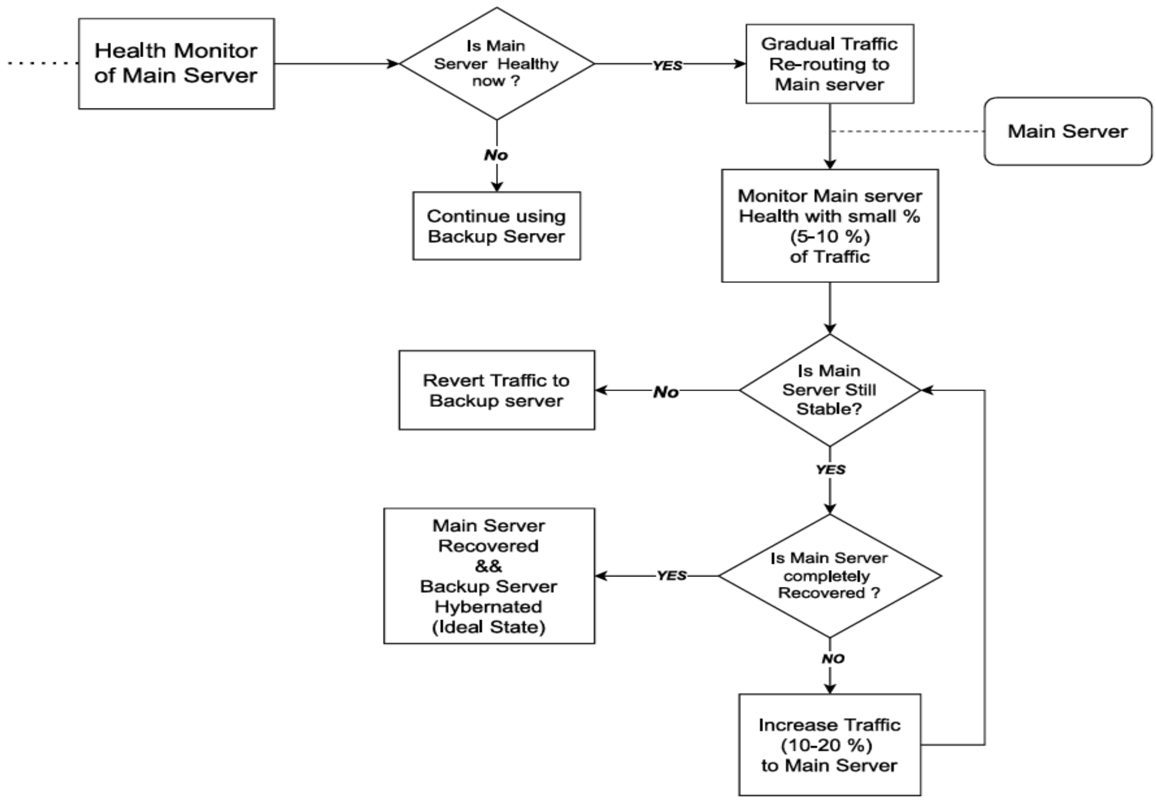


Fig.9 Continued System Architecture

Chapter 4
Results and Discussion

The performance evaluation of our models demonstrates significant differences in accuracy and false positive rates across various approaches. The Random Forest model achieved an accuracy of **96.89%** with a false positive rate (FPR) of **4.56%**, while the Deep Learning model reached a lower accuracy of **72.61%** and had a substantially higher FPR of **40.49%**. In contrast, the Incremental Random Forest model outperformed the others, achieving the highest accuracy of **98.16%** and the lowest FPR of **2.28%**.

Comparing previous and improved models, as shown in the bar chart, we observe a marked **improvement** in detection accuracy. The improvements implemented in the enhanced model contributed to **higher accuracy**, underscoring the effectiveness of the refined approach in this analysis. This performance gain validates the incremental adjustments made to **enhance model reliability** and **reduce false positives**, contributing to a more accurate and robust detection system.

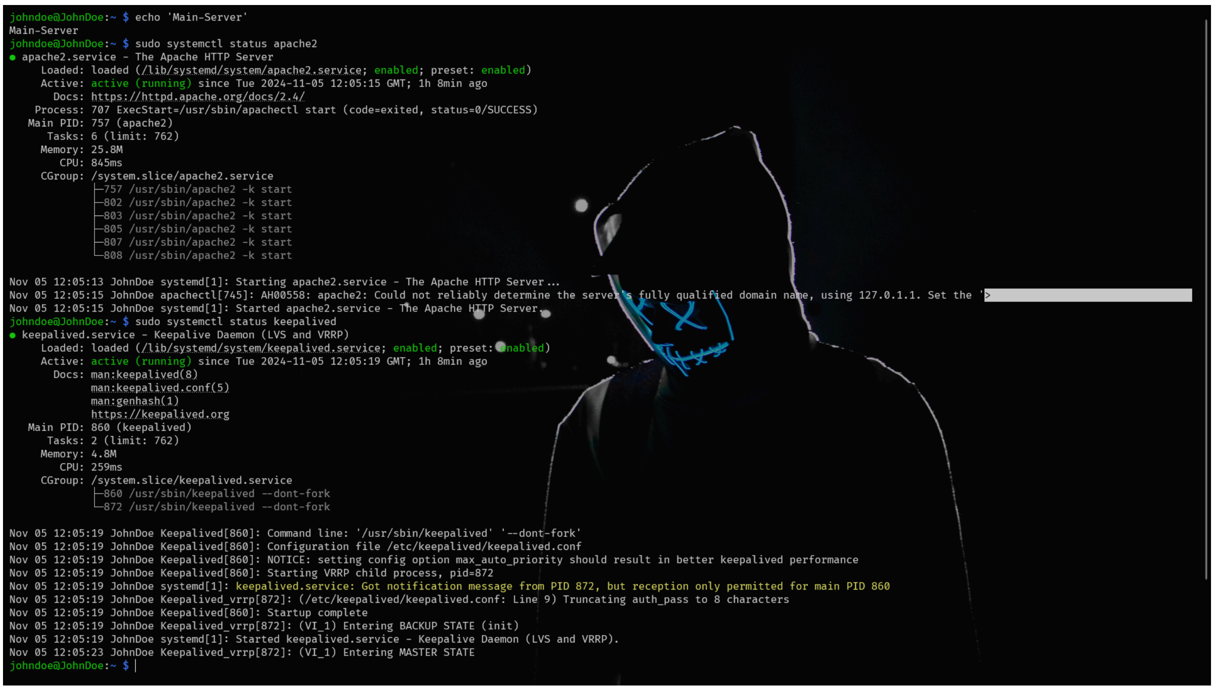


Fig.10 Main Server Status

```
Backup Server
johndoe@JohnDoe:~ $ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-11-05 12:05:09 GMT; 1h 12min ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 688 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 859 (apache2)
      Tasks: 6 (limit: 755)
        CPU: 745ms
     CGroup: /system.slice/apache2.service
             ├─859 /usr/sbin/apache2 -k start
             ├─914 /usr/sbin/apache2 -k start
             ├─916 /usr/sbin/apache2 -k start
             ├─917 /usr/sbin/apache2 -k start
             ├─918 /usr/sbin/apache2 -k start
             └─919 /usr/sbin/apache2 -k start

Nov 05 12:05:07 JohnDoe systemd[1]: Starting apache2.service - The Apache HTTP Server...
Nov 05 12:05:09 JohnDoe apachectl[716]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.>
Nov 05 12:05:09 JohnDoe systemd[1]: Started apache2.service - The Apache HTTP Server.
johndoe@JohnDoe:~ $ sudo systemctl status keepalived
● keepalived.service - Keepalive Daemon (LVS and VRRP)
     Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-11-05 12:05:15 GMT; 1h 12min ago
       Docs: man:keepalived(8)
             man:keepalived.conf(5)
             man:genhash(1)
             https://keepalived.org
   Main PID: 953 (keepalived)
      Tasks: 2 (limit: 755)
        CPU: 186ms
     CGroup: /system.slice/keepalived.service
             ├─953 /usr/sbin/keepalived --dont-fork
             └─973 /usr/sbin/keepalived --dont-fork

Nov 05 12:05:15 JohnDoe Keepalived[953]: Running on Linux 6.6.51+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.51-1+rpt3 (2024-10-08) (built for Linux>
Nov 05 12:05:15 JohnDoe Keepalived[953]: Command line: '/usr/sbin/keepalived' '--dont-fork'
Nov 05 12:05:15 JohnDoe Keepalived[953]: Configuration file /etc/keepalived/keepalived.conf
Nov 05 12:05:15 JohnDoe Keepalived[953]: NOTICE: setting config option max_auto_priority should result in better keepalived performance
Nov 05 12:05:15 JohnDoe Keepalived[953]: Starting VRRP child process, pid=973
Nov 05 12:05:15 JohnDoe systemd[1]: keepalived.service: Got notification message from PID 973, but reception only permitted for main PID 953
Nov 05 12:05:15 JohnDoe Keepalived_vrrp[973]: (/etc/keepalived/keepalived.conf: Line 9) Truncating auth_pass to 8 characters
Nov 05 12:05:15 JohnDoe Keepalived_vrrp[973]: (VI_1) Entering BACKUP STATE (init)
Nov 05 12:05:15 JohnDoe Keepalived[953]: Startup complete
Nov 05 12:05:15 JohnDoe systemd[1]: Started keepalived.service - Keepalive Daemon (LVS and VRRP).
```

Fig.11 BackUp Server Status

```
johndoe@JohnDoe:~ $ cat /etc/keepalived/keepalived.conf
vrrp_instance VI_1 {
    state BACKUP
    interface wlan0 # Change to your network interface
    virtual_router_id 51
    priority 100 # Lower priority than the master
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass your_password
    }
    virtual_ipaddress {
        192.168.182.100 # Must match the master
    }
}
```

Fig.12 Monitoring Main Server

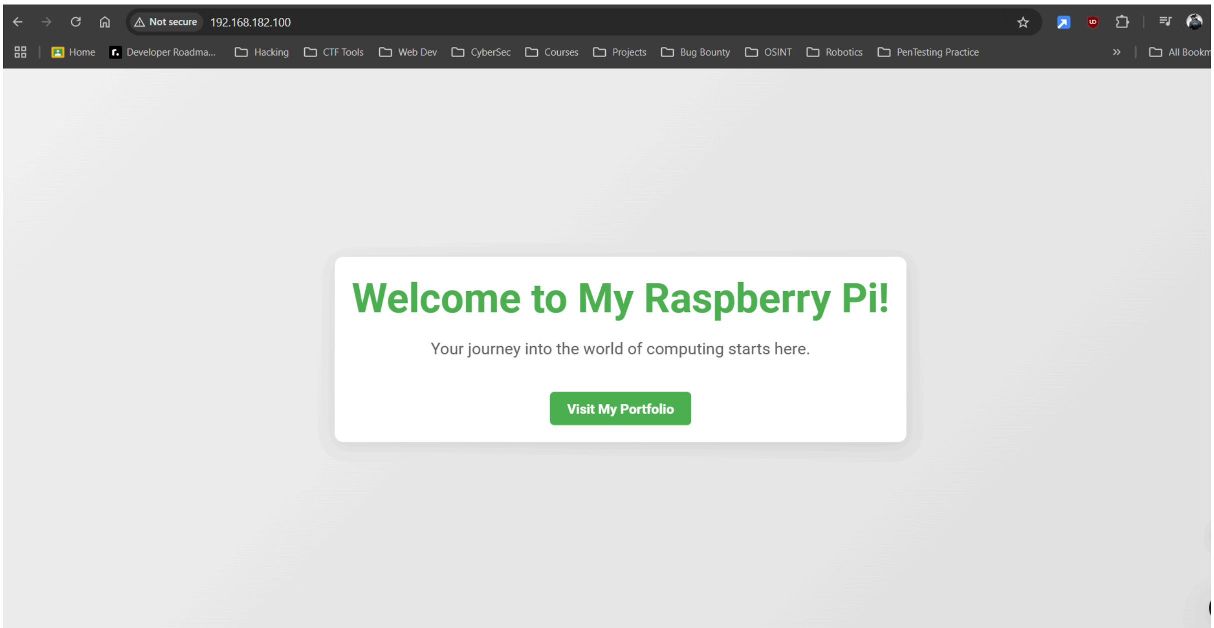


Fig.13 Service when the main server is active

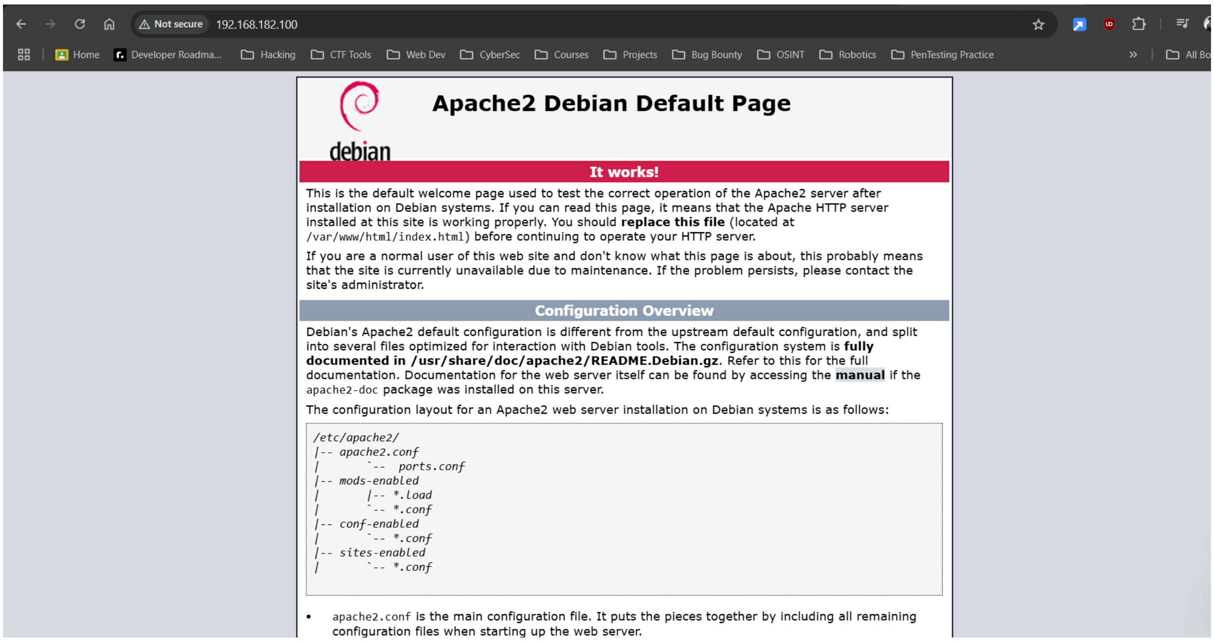


Fig.14 Service when the main server is compromised due to an attack

Chapter 5
Conclusion and Future Scope

To enhance model performance, future work could focus on exploring advanced machine learning (ML) techniques and more sophisticated deep learning architectures, such as Transformer models and novel ensemble methods, to further improve detection accuracy and reduce false positives. Implementing Explainable AI (XAI) methods would also add value by providing transparency into the model's decisions, which would help explain why certain traffic is flagged as malicious and build trust in automated detections.

Broadening the threat landscape detection capabilities is another promising direction. Expanding the system to recognize additional attack vectors—such as SQL injection, phishing, and malware distribution—alongside DDoS attacks would make it more comprehensive. Furthermore, adding multi-vector attack detection, which can identify combined threats like DDoS paired with other malicious activities, would improve its real-world applicability.

Integrating with cloud providers is another potential area for growth. By exploring partnerships with cloud-native security solutions like AWS Shield or Azure DDoS Protection, the system could become more robust and scalable. Additionally, deploying the detection and prevention mechanisms within a serverless architecture would enhance scalability and reduce operational costs.

Data privacy and ethics considerations are also essential for future iterations. Techniques like federated learning and differential privacy could allow the system to learn from sensitive data without compromising privacy. Addressing ethical concerns around automated decision-making and ensuring compliance with data protection regulations, such as GDPR, would further enhance the project's credibility and responsibility.

Real-world deployment and testing are critical steps toward practical implementation. Pilot programs in live environments would allow testing of the system's performance against actual attack conditions, refining the models based on real-world data. Establishing benchmarks in line with industry standards for DDoS detection would provide an objective measure of the system's effectiveness and reliability.

Finally, community engagement and continuous learning could amplify the project's impact. Making the project open-source would encourage community collaboration, fostering contributions from the wider cybersecurity community. Collaborating with research institutions could also advance the field by contributing to published studies on DDoS detection and prevention. Developing real-time learning systems that adapt to emerging threats, combined with a user feedback mechanism to report false positives and negatives, would ensure the model remains effective and continuously improves over time.

References

[1]Ajeetha G, Madhu Priya G, "Machine Learning Based DDoS Attack Detection," 2019 Innovations in Power and Advanced Computing Technology (i-PACT), 2019, IEEE.

[2]A. Sahi, D. Lai, Y. Li and M. Diykh, "An Efficient DDoS TCP Flood Attack Detection and Prevention System in a Cloud Environment," in *IEEE Access*, vol. 5, pp. 6036-6048, 2017, doi: 10.1109/ACCESS.2017.2688460.

[3]T. V. Phan and M. Park, "Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud," in *IEEE Access*, vol. 7, pp. 18701-18714, 2019, doi: 10.1109/ACCESS.2019.2896783.

[4]D. Yin, L. Zhang and K. Yang, "A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework," in IEEE Access, vol. 6, pp. 24694-24705, 2018, doi: 10.1109/ACCESS.2018.2831284.

[5]M. Zuñiga-Prieto, E. Insfran and S. Abrahão, "Architecture Description Language for Incremental Integration of Cloud Services Architectures," 2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA), Raleigh, NC, USA, 2016, pp. 16-23, doi: 10.1109/MESOCA.2016.10.

[6]M. H. Rohit, S. M. Fahim and A. H. A. Khan, "Mitigating and Detecting DDoS attack on IoT Environment," 2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON), Dhaka, Bangladesh, 2019, pp. 5-8, doi: 10.1109/RAAICON48939.2019.5.

[7]W. H. A. Muragaa, "A hybrid scheme for detecting and preventing single packet Low-rate DDoS and flooding DDoS attacks in SDN," 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 2023, pp. 707-712, doi: 10.1109/MI-STA57575.2023.10169712.

[8]J. Li et al., "Toward Adaptive DDoS-Filtering Rule Generation," 2023 IEEE Conference on Communications and Network Security (CNS), Orlando, FL, USA, 2023, pp. 1-9, doi: 10.1109/CNS59707.2023.10288699.

[9]Power and Energy-efficient VM scheduling in OpenStack Cloud Through Migration and Consolidation using Wake-on-LAN - Krishan Kumar, Kunal Patange, Pushkar Pete, Manjiri Wankhade, Arpitrama Chatterjee & Manish Kurhekar

[10] M. Popa and T. Slavici, "Embedded server with Wake on LAN function," IEEE EUROCON 2009, St. Petersburg, Russia, 2009, pp. 365-370, doi: 10.1109/EURCON.2009.5167657.