



**INSTITUTE FOR ADVANCED COMPUTING  
AND  
SOFTWARE DEVELOPMENT  
AKURDI, PUNE**

**DOCUMENTATION ON**

**“DEFENSE IN DEPTH”**

**PG-DITISS SEPT-2023**

**SUBMITTED BY**

**Group No: 20**

**KALYANI JAGTAP (239416)  
TANISHKA NALE (239446)**

**MRS.SUSHMA HATTARKI  
PROJECT GUIDE**

**MR. ROHIT PURANIK  
CENTRE CO-ORDINATOR**

## ABSTRACT

"Defense in Depth" is a comprehensive security strategy that employs a multi-layered approach to safeguard an organization's assets and information. This project focuses on implementing a robust defense framework that integrates technological, procedural, and human-centric measures. The strategy encompasses network security, physical security, and personnel training, creating a resilient shield against potential threats. Through the deployment of advanced cybersecurity tools, encryption technologies, and intrusion detection systems, the project aims to fortify the organization's digital infrastructure. Additionally, emphasis is placed on establishing stringent access controls, periodic security audits, and incident response plans to ensure a proactive defense posture. The project recognizes the dynamic nature of security threats and incorporates continuous monitoring and threat intelligence to adapt and evolve the defense mechanisms. Collaborative efforts with external security experts and agencies contribute to a holistic approach, ensuring a unified front against cyber threats. Through simulation exercises and training programs, the project endeavors to enhance the organization's overall security awareness and preparedness. Ultimately, "Defense in Depth" seeks to create a resilient and adaptive security environment, safeguarding critical assets and information against an evolving threat landscape.

## INDEX

1. INTRODUCTION	1
1.1 PROBLEM STATEMENT	2
1.2 ADVANTAGES	2
2. LITERATURE SURVEY	3
3. SURVEY OF TECHNOLOGY	4
4. REQUIREMENT & ANALYSIS	5
5. METHODOLOGY	6
5.1 PROPOSED SYSTEM	14
6. SYSTEM DESIGN	15
6.1 FLOW CHART	15
7. IMPLEMENTATION	16
8. FUTURE SCOPE	25
9. CONCLUSION	26





## 1. INTRODUCTION

In today's digital landscape, cybersecurity threats are constantly evolving, making it imperative for organizations to adopt robust defensive measures to protect their assets and data. The "Defense in Depth" approach offers a comprehensive strategy for bolstering cybersecurity defenses by implementing multiple layers of security controls.

At the forefront of this strategy is the Perimeter Security Layer, where technologies like Snort, an intrusion detection system, help detect and prevent external threats from breaching the network. Moving inward, Network Security focuses on safeguarding the internal network using tools like iptables and sshguard to enforce strict access controls and detect suspicious activities. Endpoint Security strengthens defenses by securing individual devices with technologies like Honeypots and Haproxy, while Data Security ensures the confidentiality and integrity of sensitive information by encrypting data stored in internal databases.

Identity and Access Management restricts access to critical resources, limiting administrative privileges to a single sudo user. Security Monitoring and Incident Response tools like tcpdump enable real-time detection and response to security incidents, while Application Security protects web-based services with HTTPS encryption on public networks.

Together, these layers form a comprehensive defense-in-depth strategy, providing organizations with the resilience and agility needed to defend against the ever-evolving threat landscape.

## 1.1 PROBLEM STATEMENT

In most organizations, the assumption of a security perimeter with trusted objects “inside” and untrusted objects “outside” no longer holds. Malicious insiders, compromised accounts, and zero-day vulnerabilities can quickly place malicious actors inside your network and at close range to critical infrastructure.

Defense in depth (DiD) is a security strategy that helps organizations deal with this situation. The strategy assumes that attackers will, or already have, penetrated different layers of the organization’s defenses. Multiple layers of security are needed to detect attackers at every stage of their attack cycle.

## 1.2 ADVANTAGES

- **Enhanced Security:** Defense in depth provides multiple layers of defense mechanisms, making it harder for attackers to penetrate the system.
- **Redundancy:** By having multiple layers, if one layer fails, there are other layers in place to protect the system.
- **Flexibility:** It allows organizations to tailor security measures according to their specific needs and requirements.
- **Resilience:** Defense in depth increases the system's ability to withstand and recover from attacks or security breaches.
- **Compliance:** It helps organizations meet regulatory requirements by demonstrating robust security measures at various levels.

## 2. LITERATURE SURVEY

The "Defense in Depth" project aims to create a robust cybersecurity framework that provides comprehensive protection against a wide range of cyber threats. By adopting a layered defense strategy, the project seeks to enhance the security posture of organizations and minimize the risk of data breaches and cyber attacks.

**Perimeter Security Layer (Snort):** The project focuses on implementing Snort, an open-source intrusion detection system (IDS), at the perimeter to monitor and analyze network traffic. Snort's rule-based detection mechanism helps identify and block malicious activities, safeguarding the organization's network perimeter from external threats.

**Network Security (Internal Network):** Beyond the perimeter, the project emphasizes strengthening internal network security measures to protect against insider threats and unauthorized access. This involves deploying firewalls, segmenting networks, and enforcing access controls to prevent unauthorized access to sensitive information and resources.

**Endpoint Security (iptables, sshguard, Honeypot, Haproxy):** Endpoint security is vital for protecting individual devices from cyber threats. The project incorporates iptables for host-based firewall protection, sshguard to mitigate brute-force attacks on SSH connections, and honeypots to lure and trap attackers. Additionally, Haproxy helps balance and secure traffic between servers, ensuring high availability and protection against DDoS attacks.

**Data Security (Encrypted Database):** Data security measures focus on encrypting sensitive information stored in databases to prevent unauthorized access and data breaches. By encrypting data at rest and in transit, the project aims to safeguard confidential information even if attackers manage to penetrate other layers of defense.

**Identity and Access Management (Sudo User):** Limiting administrative privileges to a single sudo user helps reduce the attack surface and minimize the risk of privilege escalation attacks. By enforcing strong authentication mechanisms and least privilege principles, the project ensures that only authorized users have access to critical resources.

**Security Monitoring and Incident Response (tcpdump):** The project emphasizes proactive security monitoring using tcpdump to capture and analyze network traffic for suspicious activities.

**Application Security (Https - Public Network):** Securing web applications is critical to protecting against various cyber threats such as cross-site scripting (XSS), SQL injection, and data breaches. Implementing HTTPS (Hypertext Transfer Protocol Secure) ensures secure communication between web servers and clients by encrypting data transmission over the internet



### 3. SURVEY OF TECHNOLOGY

In deploying the "Defense in Depth" strategy, a thorough survey of technology solutions is essential to address various cybersecurity aspects effectively. Beginning with perimeter security, Snort emerges as a preferred choice for its rule-based intrusion detection capabilities and strong community support. For internal network security, the use of VLANs, ACLs, and VPNs enhances segmentation and restricts unauthorized access. Endpoint security is bolstered by leveraging iptables for firewall management, sshguard for dynamic SSH blocking, and Honeypots for deceptive defense. Haproxy is selected for load balancing and reverse proxy functions, ensuring robust protection.

Data security is fortified by implementing AES encryption for databases, safeguarding sensitive information from unauthorized access. Identity and access management practices include designating a single user as a sudo user, minimizing administrative privileges and reducing potential attack vectors. Security monitoring and incident response capabilities are enhanced with tcpdump for real-time packet analysis, facilitating rapid threat detection and mitigation.

Application security is prioritized with the adoption of HTTPS for secure web communication, mitigating risks associated with data interception and manipulation. Secure file transfer protocols such as SFTP are employed to ensure encrypted transmission of files, maintaining data confidentiality and integrity. DNS security is ensured with BIND9, known for its robustness and support for DNSSEC, providing authentication and integrity verification for DNS responses.

Additionally, password security measures involve implementing Bcrypt for secure password storage, mitigating risks associated with password-based attacks. Through this comprehensive survey and adoption of appropriate technologies, organizations can establish a multi-layered defense strategy, safeguarding against diverse cyber threats effectively.

## 4. REQUIREMENT & ANALYSIS

### Requirements:

#### Hardware requirements :

**Operating system** :- Debian 12 & Windows 10

**RAM** : 16 GB

**SSD**: 256GB

Virtual Box(VB)

## 5. METHODOLOGY

Methodology for implementing the "Defense in Depth" approach involves a systematic process to address various aspects of cybersecurity.

**Requirement Analysis:** Begin by analyzing the organization's security requirements and identifying potential vulnerabilities and risks.

**Perimeter Security Layer (Snort):** Deploy Snort at the network perimeter to analyze incoming and outgoing traffic for signs of malicious activity.

**Network Security (Internal Network):** Implement network segmentation, access control lists (ACLs), and virtual private networks (VPNs) to secure communication within the internal network.

**Endpoint Security (iptables, sshguard, Honeypot, Haproxy):** Configure iptables as a firewall utility to manage network traffic rules. Install sshguard to monitor SSH login attempts and block suspicious IP addresses dynamically. Deploy Honeypots to divert attackers from critical systems. Use Haproxy as a reverse proxy for enhanced security and performance.

**Data Security (Encrypted Database):** Encrypt databases containing sensitive information using encryption algorithms like AES to protect data confidentiality.

**Identity and Access Management (Sudo User):** Limit access to critical systems by granting sudo privileges to authorized users only.

**Security Monitoring and Incident Response (tcpdump):** Install tcpdump for real-time network traffic analysis and packet capture to monitor network activity and detect anomalies.

**Application Security (HTTPS):** Secure web applications using HTTPS to encrypt data transmitted between web browsers and servers.

**Secure File Transfer (SFTP):** Implement SFTP for secure file transfer between systems, ensuring data confidentiality during transit.

**DNS Security (BIND9):** Deploy BIND9 DNS server with DNSSEC to authenticate DNS responses and prevent DNS spoofing attacks.

**Password Security (Bcrypt):** Use Bcrypt for secure password storage, generating salted hash values to protect user passwords from brute-force attacks.

**Integration and Testing:** Integrate all security components into the organization's infrastructure and conduct thorough testing to ensure proper functionality and effectiveness.

**Training and Awareness:** Provide training to employees on security best practices and raise awareness about potential cyber threats and how to mitigate them.

**Regular Updates and Maintenance:** Implement a process for regular updates and maintenance of security systems and protocols to address emerging threats and vulnerabilities.

By following this methodology, organizations can establish a robust defense in depth strategy to protect against a wide range of cyber threats and ensure the security of their critical assets and data.

**Snort:** Snort works by analyzing network traffic in real-time to detect and alert on suspicious activity based on predefined rules.

**iptables:** iptables operates as a firewall by filtering and manipulating network packets to control incoming and outgoing traffic.

**sshguard:** sshguard protects against brute-force attacks by monitoring SSH authentication attempts and dynamically blocking malicious IP addresses.

**Honeypot:** Honeypots simulate vulnerable systems to lure attackers, allowing security professionals to monitor and analyze their tactics and techniques.

**Haproxy:** Haproxy functions as a load balancer and reverse proxy, distributing incoming traffic across multiple servers and providing high availability and performance.

**AES encryption:** AES encryption secures data by encrypting it with a symmetric key cipher, ensuring confidentiality and integrity during storage and transmission.

**sudo user:** Designating a single sudo user restricts administrative privileges to one account, reducing the attack surface and enhancing access control.

**tcpdump:** tcpdump captures and analyzes network packets in real-time, providing valuable insights for network troubleshooting, security monitoring, and forensics.

**HTTPS:** HTTPS encrypts web communication using SSL/TLS protocols, protecting sensitive information from eavesdropping and tampering.

**SFTP:** SFTP enables secure file transfer over SSH, encrypting data during transmission to prevent interception and unauthorized access.

**BIND9:** BIND9 serves as a DNS server, translating domain names into IP addresses and facilitating secure and reliable domain name resolution.

**Bcrypt:** Bcrypt hashes and salts passwords, making them resistant to brute-force attacks and enhancing password security in authentication systems.

**Working:-****sshguard -**

```
sudo apt install sshguard
sudo systemctl start sshguard
sudo systemctl enable sshguard
sudo nano /etc/sshguard/sshguard.conf
sudo systemctl restart sshguard
sudo nano /etc/sshguard/sshguard.conf
sudo systemctl restart sshguard
configuration file -
```

**#### REQUIRED CONFIGURATION ####**

```
# Full path to backend executable (required, no default)
```

```
BACKEND="/usr/lib/x86_64-linux-gnu/sshg-fw-nft-sets"
```

```
# Shell command that provides logs on standard output. (optional, no default)
```

```
# Example 1: ssh and sendmail from systemd journal:
```

```
LOGREADER="LANG=C /bin/journalctl -afb -p info -n1 -o cat SYSLOG_FACILITY=4
```

```
SYSLOG_FACILITY=10"
```

**##### OPTIONS #####**

```
# Block attackers when their cumulative attack score exceeds THRESHOLD.
```

```
# Most attacks have a score of 10. (optional, default 30)
```

```
THRESHOLD=30
```

```
BLOCK_DURATION=600
```

```
UNBLOCK_THRESHOLD=50
```

```
MAX_ATTEMPTS_PER_IP=3
```

```
LOG_LEVEL=ERROR
```

```
SYSLOG_ENABLE=yes
```

```
FIREWALL_BACKEND=iptables
```

```
# Block attackers for initially BLOCK_TIME seconds after exceeding THRESHOLD.
```

```
# Subsequent blocks increase by a factor of 1.5. (optional, default 120)
```

```
BLOCK_TIME=120
```

```
# Remember potential attackers for up to DETECTION_TIME seconds before
```

```
# resetting their score. (optional, default 1800)
```

```
DETECTION_TIME=1800
```

```
cmd -
```

```
sudo systemctl status sshguard
```

**DNS (BIND9):**

```
;
; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA ns.shuharilabs.local. admin.shuharilabs.local. (
    1 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;
@ IN NS ns.shuharilabs.local.
1.0.0 IN PTR localhost.
127 IN PTR ns.shuharilabs.local.
16 IN PTR ser1.shuharilabs.local.
41 IN PTR ser1.shuharilabs.local.
```

**HAPROXY:**

```
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private

# See: https://ssl-config.mozilla.org/#server=haproxy&server-
version=2.0.3&config=intermediate
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY130
5_SHA256
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
    log global
    mode http
```

```
option httplog
option dontlognull
timeout connect 5000
timeout client 50000
timeout server 50000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

frontend http_front
    bind *:80
#    bind *:443
    default_backend http_back

backend http_back
    balance roundrobin
    server server1 192.168.64.41:80 check
    server server2 192.168.64.16:80 check
```

## Honey-pot

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/local/bin/python3.8 1
sudo apt install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev
python3-minimal authbind virtualenv
git clone https://github.com/cowrie/cowrie.git cowrie

wget https://www.python.org/ftp/python/3.11.3/Python-3.11.7.tgz
unzip Python-3.11.2.tar.xz
sudo unzip Python-3.11.2.tar.xz
sudo tar -xf Python-3.11.2.tar.xz
ls
cd Python-3.11.2
ls
./configure --enable-optimizations
sudo make altinstall
sudo update-alternatives --install /usr/bin/python3 python3 /usr/local/bin/python3.11 1
sudo update-alternatives --install /usr/bin/python3 python3 /usr/local/bin/python3.11 1
sudo apt install python3-virtualenv
801 sudo pip install virtualenv==15.1.0
virtualenv venv
source venv/bin/activate
```

```
pip install --upgrade pip
pip install --upgrade -r requirements.txt
cp etc/cowrie.cfg.dist etc/cowrie.cfg
nano etc/cowrie.cfg
nano etc/userdb.txt
bin/cowrie status
bin/cowrie start
bin/cowrie status
top
sudo tail -f var/log/cowrie/cowrie.log
```

change ssh port number 22 to 222 (any) [cmd login - ssh -p 222 ser1@192.168.56.117]  
set cowrie port number 22

## **iptables**

```
sudo iptables -A INPUT -p tcp --dport 80 -m state --state NEW -m recent --set
sudo iptables -A INPUT -p tcp --dport 80 -m state --state NEW -m recent --update --seconds 60 -
-hitcount 10 -j DROP
```

```
sudo iptables -A INPUT -p tcp --dport 443 -m state --state NEW -m recent --set --name HTTPS
sudo iptables -A INPUT -p tcp --dport 443 -m state --state NEW -m recent --update --seconds 60
--hitcount 3 --name HTTPS -j DROP
```

whitelisting  
sudo iptables -A INPUT -s trusted\_ip\_address -j ACCEPT

blacklisting  
sudo iptables -A INPUT -s malicious\_ip\_address -j DROP

alert tcp any any -> \$HOME\_NET 80 (msg:"Possible HTTP Flood Detected"; flow:stateless;  
threshold:type limit, track by\_src, count 10, seconds 60; sid:1000001;)

## **HTTPS CERTIFICATE USING OPENSSSL:**

### **ROOTCA -**

```
sudo apt install vsftpd openssl net-tools tree -y
path /home/shuhari
```

```
mkdir ca
```



```
cd ca
mkdir -p certs crl newcerts private subca/csr subca/certs
touch index.txt
touch index.txt.attr
echo 1000 > serial
echo 1000 > crlnumber
```

import rootca.cnf file and change path dir = /home/shuhari/ca

#### KEY GENERATION -

```
openssl genrsa -aes256 -out private/ca.key.pem 4096
chmod 400 private/ca.key.pem
CERTIFICATE -
```

```
openssl req -config rootca.cnf -key private/ca.key.pem -new -x509 -days 7200 -sha256 -extensions
v3_ca -out certs/ca.cert.pem
```

#### SUBCA -

```
sudo apt install vsftpd openssl net-tools tree -y
mkdir subca
cd subca
mkdir certs csr crl newcerts private
touch index.txt
touch index.txt.attr
echo 1000 > serial
echo 1000 > crlnumber
import subca.cnf file change path dir = /home/shuhari/subca
SUBCA KEY -
openssl genrsa -aes256 -out private/subca.key.pem 4096
chmod 400 private/subca.key.pem
```

#### SUBCA CSR -

```
openssl req -config subca.cnf -key private/subca.key.pem -new -sha256 -out csr/subca.csr.pem
```

```
scp csr/subca.cnf 192.168.80.4:/home/shuhari/subca/csr
```

#### Open ROOT

#### ROOTCA SIGN ON SUBCA -

```
openssl ca -config rootca.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256 -in
subca/csr/subca.csr.pem -out subca/cert/subca.cert.pem
```

```
openssl -x509 -noout -text -in subca/cert/subca.cert.pem
```

```
compare sha256
```

```
sha256sum newcerts/index.txt subca/cert/subca.cert.pem
```

```
scp subca/cert/subca.cert.pem 192.168.80.3:/home/shuhari/certs
```

#### WEB CERTIFICATE -

```
sudo apt install vsftpd bind9 apache2 tree openssl net-tools -y
```

#### SETUP DNS

```
import subca.cnf file
```

```
mkdir /home/shuhari/certs
```

```
mkdir /home/shuhari/ser1
```

```
cd certs
```

#### KEY GENERATION -

```
openssl genrsa -aes256 -out www.shuhari.local.key.pem 2048
```

```
chmod 400 www.shuhari.local.key.pem
```

#### CSR -

```
openssl req -config subca.cnf -key www.shuhari.local.key.pem -new -sha256 -out  
www.shuhari.local.csr.pem
```

```
scp www.shuhari.local.csr.pem 192.168.80.3:/home/shuhari/csr
```

```
open subca
```

#### SUBCA SIGN ON WWW -

```
openssl ca -config subca.cnf -extensions server_cert -days 365 -notext -md sha256 -in  
csr/www.shuhari.local.csr.pem -out certs/www.shuhari.local.cert.pem
```

#### NOW COLLECT ALL CERTIFICATES OF ROOTCA SUBCA AND WWW

```
scp certs/* 192.168.80.5:/home/shuhari/ser1
```

#### FROM ROOTCA

```
scp certs/ca.cert.pem 192.168.80.5:/home/shuhari/ser1
```

```
go to WWW
```

```
cd ser1
```

```
cat www.shuhari.local.cert.pem subca.cert.pem ca.cert.pem > www.bundle.cert
```

```
mkdir /etc/apache2/ssl
```

```
sudo a2enmod ssl
```

```
sudo a2ensite default-ssl
```

```
sudo cp www.bundle.cert /etc/apache2/ssl
```

```
sudo cp ../certs/www.shuhari.local.key.pem /etc/apache2/ssl
```

```
sudo nano /etc/apache2/sites-available/default-ssl
```

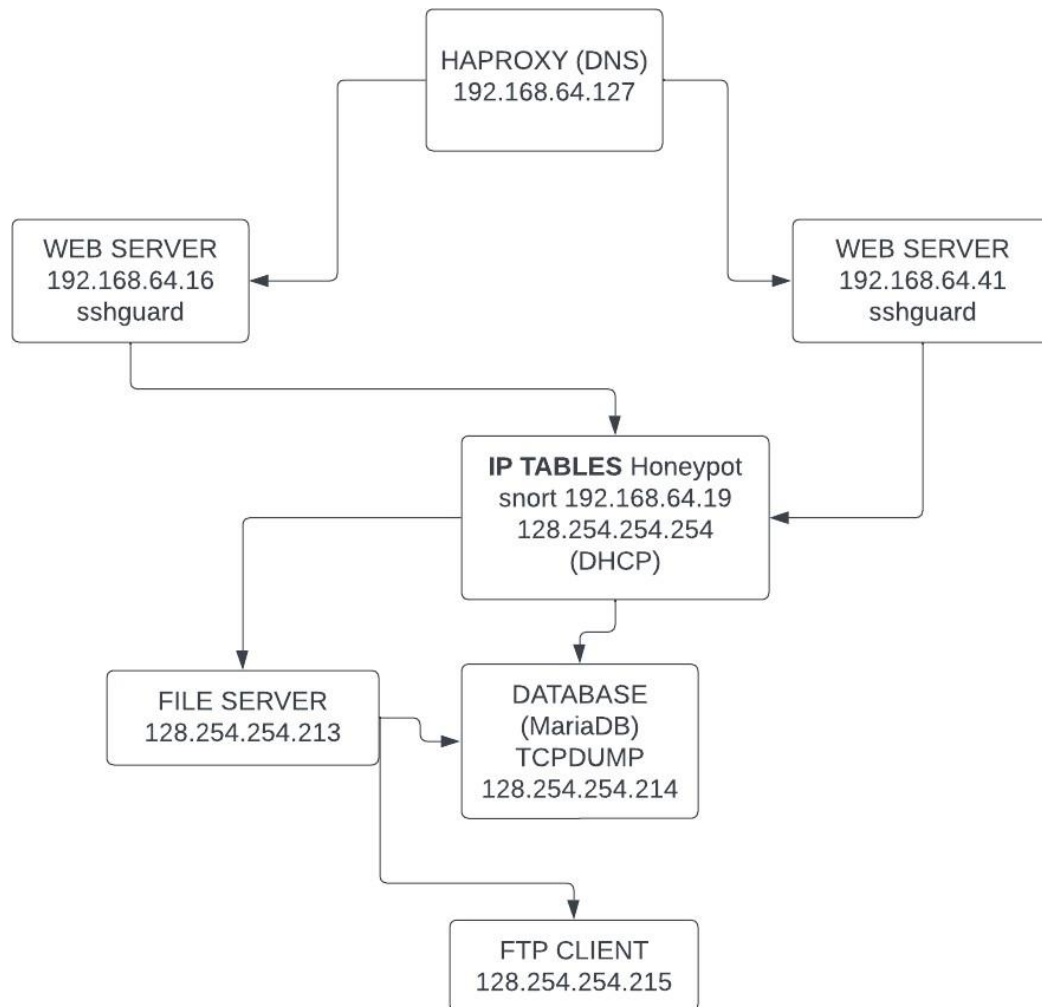
## **5.1 PROPOSED SYSTEM**

Defense in depth helps you ensure that you are protecting your systems as effectively as possible. It forces you to account for security even when your various tools and solutions have been compromised. No security tool or measure is perfect so you need to account for potential failures. By building in layers of security, you can reduce the chance of a single point of failure occurring in your systems.

The digital world has revolutionized how we live, work and play. However, it's a digital world that is constantly open to attack, and because there are so many potential attackers, we need to ensure we have the right security in place to prevent systems and networks being compromised. Unfortunately, there is no single method that can successfully protect against every single type of attack. This is where a defense in depth architecture comes into play.

## 6. SYSTEM DESIGN

### 6.1 FLOW CHART



## 7. IMPLEMENTATION

### INTERNAL NETWORK

```
iacsd@debian:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:51:5c:a0 brd ff:ff:ff:ff:ff:ff
    inet 128.254.254.214/16 brd 128.254.255.255 scope global dynamic enp0s3
        valid_lft 427sec preferred_lft 427sec
    inet6 fe80::a00:27ff:fe51:5ca0/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:73:04:2d brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.112/24 brd 192.168.56.255 scope global dynamic enp0s8
        valid_lft 442sec preferred_lft 442sec
    inet6 fe80::a00:27ff:fe73:42d/64 scope link
        valid_lft forever preferred_lft forever
iacsd@debian:~$
```

### MARIADB PASSWORD ENCRYPTION -

```
MariaDB [pr]> select * from data;
```

id	username	password
1		
2	user1	pass@123
3		
4		
5		
6	hello	hello
7	hello	\$2y\$10\$wLvYmOt2XFi.oajbocpM20tJ0Bvk8HqgYvf7P5gtIUToVWe2zX79e
8	exam	\$2y\$10\$ogS7Z1FDX7kK6KdEOMD0UuTYuXLCxTv/w3GccjR/cGIV0hu0BqI82
9	Ditiss	\$2y\$10\$/NJgWdYC2SwwOi8jDo1NA0L5NUT/bcg92zpa8o2JYX0ucKoDADT1i
10	iacsd	\$2y\$10\$eZzyTsbwAGDdEYEU7xDMYepNEZAm6JzqgTSDaUbZU1ZnYpxtKdwCC
11	Pune	\$2y\$10\$4cqbML0sfCwPwW8sW8RDYOr5T8Je14CkrYr4dZM.kqDc4wKk1DBIm
12	India	\$2y\$10\$fNh4bZ4gh9o61Kseu7aAluzHi3WrrY7mSmzzQWi3AuUHcwGcJuz06
13	sushma	\$2y\$10\$kFYVbSL3yo3AhotZAZxs7eA/mlcUeH5ML0yI8BP.QB8KlqNonW/fC
14	pratik	\$2y\$10\$7AWf0VSBYmGzo8h/tfCSi0YAbguk6zg47L5KL2bk/SswssjqMcNMu
15	q	\$2y\$10\$IpoR0kIxEBD3ET/u/qz5BedygbBvIqCbNVZtoQ56G4cJEoLdp0180

```
15 rows in set (0.000 sec)

MariaDB [pr]>
```

## TCPDUMP

```

iacsd@debian:~$ sudo tcpdump -i enp0s3
[sudo] password for iacsd:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
10:47:58.979980 IP6 fe80::35e0:4fa:e6f9:8ac2.dhcpv6-client > ff02::1:2.dhcpv6-server: dhcp6 solicit
10:47:59.000108 IP 128.254.254.214.51683 > 128.254.254.215.domain: 38509+ PTR? 2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa. (90)
10:47:59.000963 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:47:59.001023 IP 128.254.254.214.57140 > 128.254.254.215.domain: 38509+ PTR? 2.0.0.0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.f.f.ip6.arpa. (90)
10:47:59.001600 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:47:59.001755 IP 128.254.254.214.58902 > 128.254.254.215.domain: 47567+ PTR? 2.c.a.8.9.f.6.e.a.f.4.0.0.e.5.3.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa. (90)
10:47:59.002463 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:47:59.002543 IP 128.254.254.214.56537 > 128.254.254.215.domain: 47567+ PTR? 2.c.a.8.9.f.6.e.a.f.4.0.0.e.5.3.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa. (90)
10:47:59.003181 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:47:59.008589 IP 128.254.254.214.44496 > 128.254.254.215.domain: 56612+ PTR? 215.254.254.128.in-addr.arpa. (46)10:47:59.009536 IP 128.254.254.215 > 128.254.254.214:
ICMP 128.254.254.215 udp port domain unreachable, length 8210:47:59.089657 IP 128.254.254.214.38172 > 128.254.254.215.domain: 56612+ PTR? 215.254.254.128.in-addr.arpa
a. (46)10:47:59.090392 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 8210:47:59.090527 IP 128.254.254.214.52629 > 128
.254.254.215.domain: 4355+ PTR? 214.254.254.128.in-addr.arpa. (46)
10:48:02.980037 IP6 fe80::35e0:4fa:e6f9:8ac2.dhcpv6-client > ff02::1:2.dhcpv6-server: dhcp6 solicit
10:48:04.110700 IP 128.254.254.214.52629 > 128.254.254.215.domain: 4355+ PTR? 214.254.254.128.in-addr.arpa. (46)
10:48:04.111244 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 8210:48:04.156997 ARP, Request who-has 128.254.254.214
tell 128.254.254.215, length 46
10:48:04.157007 ARP, Reply 128.254.254.214 is-at 08:00:27:51:5c:a0 (oui Unknown), length 28
10:48:04.182073 ARP, Request who-has 128.254.254.215 tell 128.254.254.214, length 28
10:48:04.182570 ARP, Reply 128.254.254.215 is-at 08:00:27:52:f7:6b (oui Unknown), length 46
10:48:10.995101 IP6 fe80::35e0:4fa:e6f9:8ac2.dhcpv6-client > ff02::1:2.dhcpv6-server: dhcp6 solicit
10:48:26.994857 IP6 fe80::35e0:4fa:e6f9:8ac2.dhcpv6-client > ff02::1:2.dhcpv6-server: dhcp6 solicit
10:48:39.385925 IP6 fe80::a00:27ff:fe9f:94e > ip6-allrouters: ICMP6, router solicitation, length 16
10:48:39.463777 IP 128.254.254.214.40802 > 128.254.254.215.domain: 408+ PTR? e.4.9.0.f.9.e.f.f.f.7.2.0.0.a.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa. (90)
10:48:39.462633 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:48:39.462741 IP 128.254.254.214.48564 > 128.254.254.215.domain: 485+ PTR? e.4.9.0.f.9.e.f.f.f.7.2.0.0.a.0.0.0.0.0.0.0.0.0.0.0.8.e.f.ip6.arpa. (90)
10:48:39.463406 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 126
10:48:44.703135 ARP, Request who-has 128.254.254.215 tell 128.254.254.214, length 28
10:48:44.703864 ARP, Reply 128.254.254.215 is-at 08:00:27:52:f7:6b (oui Unknown), length 46
10:48:56.417019 IP 128.254.254.212.netbios-dgm > 128.254.255.255.netbios-dgm: UDP, length 207
10:48:56.417356 IP 128.254.254.214.60679 > 128.254.254.215.domain: 56684+ PTR? 255.255.254.128.in-addr.arpa. (46)
10:48:56.417864 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 82
10:48:56.417941 IP 128.254.254.214.36327 > 128.254.254.215.domain: 56684+ PTR? 255.255.254.128.in-addr.arpa. (46)
10:48:56.418580 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 82
10:48:56.418710 IP 128.254.254.214.59695 > 128.254.254.215.domain: 46134+ PTR? 212.254.254.128.in-addr.arpa. (46)
10:48:56.419287 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 82
10:48:56.419362 IP 128.254.254.214.44874 > 128.254.254.215.domain: 46134+ PTR? 212.254.254.128.in-addr.arpa. (46)
10:48:56.419990 IP 128.254.254.215 > 128.254.254.214: ICMP 128.254.254.215 udp port domain unreachable, length 82
10:48:58.994079 IP6 fe80::35e0:4fa:e6f9:8ac2.dhcpv6-client > ff02::1:2.dhcpv6-server: dhcp6 solicit
10:49:01.498768 ARP, Request who-has 128.254.254.214 tell 128.254.254.215, length 46
10:49:01.498780 ARP, Reply 128.254.254.214 is-at 08:00:27:51:5c:a0 (oui Unknown), length 28

```

## IPTABLES

```

iacsd@debian:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:27:0e:11 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 84263sec preferred_lft 84263sec
    inet6 fe80::a00:27ff:fe27:e11/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7d:aa:4e brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.110/24 brd 192.168.56.255 scope global dynamic enp0s8
        valid_lft 374sec preferred_lft 374sec
    inet6 fe80::a00:27ff:fe7d:aa4e/64 scope link
        valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:ec:c0:9f brd ff:ff:ff:ff:ff:ff
    inet 128.254.254.254/16 brd 128.254.255.255 scope global enp0s9
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feec:c09f/64 scope link
        valid_lft forever preferred_lft forever
iacsd@debian:~$ |

```



```

iacsd@debian:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  anywhere              anywhere
ACCEPT     all  --  anywhere              anywhere             state RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

iacsd@debian:~$ sudo iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
MASQUERADE all  --  anywhere              anywhere
iacsd@debian:~$

```

## HONEYPOT

```

[venv] cowrie@debian:~/cowrie$ sudo tail -f var/log/cowrie/cowrie.log
[sudo] password for cowrie:
2024-02-21T11:17:14.189778Z [HoneyPotSSHTransport,2,192.168.64.97] CMD: cd /etc/
2024-02-21T11:17:14.190241Z [HoneyPotSSHTransport,2,192.168.64.97] Command found: cd /etc/
2024-02-21T11:20:08.852877Z [-] Timeout reached in HoneyPotSSHTransport
2024-02-21T11:20:08.853430Z [twisted.conch.ssh.session#info] exitCode: 1
2024-02-21T11:20:08.853562Z [cowrie.ssh.connection.CowrieSSHConnection#debug] sending request b'exit-status'
2024-02-21T11:20:08.853903Z [-] Closing TTY Log: var/lib/cowrie/tty/fd9f902f09a7aa7511bbcdfe2b19648ad19bbe0fe91e9e941790b9f107a81f after 180 seconds
2024-02-21T11:20:08.854079Z [cowrie.ssh.connection.CowrieSSHConnection#info] sending close 0
2024-02-21T11:20:08.854437Z [HoneyPotSSHTransport,2,192.168.64.97] avatar root logging out
2024-02-21T11:20:08.854593Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-02-21T11:20:08.854735Z [HoneyPotSSHTransport,2,192.168.64.97] Connection lost after 182 seconds
2024-02-21T11:41:04.567579Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 192.168.64.97:4910 (192.168.64.16:22) [session: 0ef24d5eac48]
2024-02-21T11:41:04.569448Z [HoneyPotSSHTransport,3,192.168.64.97] Remote SSH version: SSH-2.0-OpenSSH_for_Windows_8.6
2024-02-21T11:41:04.570274Z [HoneyPotSSHTransport,3,192.168.64.97] SSH client hassh fingerprint: ae8bd7dd09970555aa4c6ed22adbff56
2024-02-21T11:41:04.571130Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2024-02-21T11:41:04.571897Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-02-21T11:41:04.571303Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-02-21T11:41:04.576791Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2024-02-21T11:41:04.577461Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2024-02-21T11:41:04.578527Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'serl' trying auth b'none'
2024-02-21T11:41:05.961461Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'serl' trying auth b'password'2024-02-21T11:41:05.961966Z [HoneyPotSSHTransport,3,192.168.64.97] login attempt [b'serl'/b''] failed
2024-02-21T11:41:06.963830Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'serl' failed auth b'password'2024-02-21T11:41:06.964092Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2024-02-21T11:41:08.587181Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-02-21T11:41:08.587483Z [HoneyPotSSHTransport,3,192.168.64.97] Connection lost after 4 seconds
2024-02-21T11:41:13.450270Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 192.168.64.97:4911 (192.168.64.16:22) [session: 55327f8a12a9]
2024-02-21T11:41:13.452442Z [HoneyPotSSHTransport,4,192.168.64.97] Remote SSH version: SSH-2.0-OpenSSH_for_Windows_8.6
2024-02-21T11:41:13.453118Z [HoneyPotSSHTransport,4,192.168.64.97] SSH client hassh fingerprint: ae8bd7dd09970555aa4c6ed22adbff56
2024-02-21T11:41:13.454092Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ssh-ed25519'
2024-02-21T11:41:13.454164Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-02-21T11:41:13.454224Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-02-21T11:41:13.460118Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2024-02-21T11:41:13.460897Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2024-02-21T11:41:13.461912Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'none'
2024-02-21T11:41:14.016587Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' trying auth b'password'2024-02-21T11:41:14.016918Z [HoneyPotSSHTransport,4,192.168.64.97] login attempt [b'root'/b's'] succeeded
2024-02-21T11:41:14.017415Z [HoneyPotSSHTransport,4,192.168.64.97] Initialized emulated server as architecture: linux-x64-lsb
2024-02-21T11:41:14.017753Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'root' authenticated with b'password'
2024-02-21T11:41:14.018043Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-connection'
2024-02-21T11:41:14.021616Z [cowrie.ssh.connection.CowrieSSHConnection#debug] got channel b'session' request
2024-02-21T11:41:14.021823Z [cowrie.ssh.session.HoneyPotSSHSession#info] channel open

```

## SNORT

```

[ Number of patterns truncated to 28 bytes: 0 ]
pcap DAQ configured to passive.
Acquiring network traffic from "enp8s9".
Reload thread starting.
Reload thread started, thread 0x7f836b701700 (1989)
Decoding Ethernet
Set pid to 116
Set uid to 108
--- Initialization Complete ---

--> Snort! <--
0.9.9 Version 2.9.13 GRE (Build 7)
By Martin Roesch & The Snort Team: http://www.snort.org/contactteam
Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.8.1
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 3.1 <Build 1>
Preprocessor Object: apid Version 1.1 <Build 5>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SQLDP Version 1.1 <Build 4>
Preprocessor Object: SF_SOF Version 1.1 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_PMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_DMS Version 1.1 <Build 4>
Preprocessor Object: SF_PCEPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>

Commencing packet processing (pid=1984)
02/21-11:42:18.921189 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:34004 -> 192.168.64.16:80
02/21-11:42:18.921248 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:34004
02/21-11:42:12.923328 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:54986 -> 192.168.64.16:80
02/21-11:42:12.923359 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:54986
02/21-11:42:14.944048 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:54999 -> 192.168.64.16:80
02/21-11:42:14.944573 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:54999
02/21-11:42:16.959183 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:54996 -> 192.168.64.16:80
02/21-11:42:16.959235 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:54996
02/21-11:42:18.962193 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:55002 -> 192.168.64.16:80
02/21-11:42:18.962285 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:55002
02/21-11:42:20.964798 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:55010 -> 192.168.64.16:80
02/21-11:42:20.964848 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:55010
02/21-11:42:22.988188 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:60148 -> 192.168.64.16:80
02/21-11:42:22.988208 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:60148
02/21-11:42:25.017628 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:60168 -> 192.168.64.16:80
02/21-11:42:25.017788 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:60168
02/21-11:42:27.021658 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:60164 -> 192.168.64.16:80
02/21-11:42:27.021723 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:60164
02/21-11:42:29.043426 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:60188 -> 192.168.64.16:80
02/21-11:42:29.043498 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:60188
02/21-11:42:31.046115 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:60196 -> 192.168.64.16:80
02/21-11:42:31.046178 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:60196
02/21-11:42:33.048938 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.127:50418 -> 192.168.64.16:80
02/21-11:42:33.048991 ** [1:10000001:0] x ** [Priority: 0] [TCP] 192.168.64.16:80 -> 192.168.64.127:50418

```

## APACHE2 – HTTPS

```

iacsd@debian:~$ sudo systemctl restart apache2
[sudo] password for iacsd:
Enter passphrase for SSL/TLS keys for ser1.shuharilabs.local:443 (RSA): (press TAB for no echo)
Broadcast message from root@debian (Wed 2024-02-21 11:39:01 IST):

Password entry required for 'Enter passphrase for SSL/TLS keys for ser1.shuharilabs.local:443 (RSA):' (PID 2670).
Please enter password with the systemd-tty-ask-password-agent tool.

****
iacsd@debian:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded /lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-02-21 11:39:02 IST; 8min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 2648 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Main PID: 2701 (apache2)
      Tasks: 6 (limit: 2306)
     Memory: 13.2M
        CPU: 85ms
    CGroup: /system.slice/apache2.service
            └─2701 /usr/sbin/apache2 -k start
              └─2702 /usr/sbin/apache2 -k start
                └─2703 /usr/sbin/apache2 -k start
                  └─2704 /usr/sbin/apache2 -k start
                    └─2705 /usr/sbin/apache2 -k start
                      └─2706 /usr/sbin/apache2 -k start

Feb 21 11:39:01 debian systemd[1]: Starting apache2.service - The Apache HTTP Server...
Feb 21 11:39:01 debian apachectl[2658]: AH00558: apache2: Could not reliably determine the server's fully qualified
Feb 21 11:39:02 debian systemd[1]: Started apache2.service - The Apache HTTP Server.

```



## HAPROXY

```
iacsd@debian:~$ sudo service haproxy status
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-02-21 11:44:19 IST; 1s ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
   Main PID: 1288 (haproxy)
      Tasks: 5 (limit: 2306)
    Memory: 42.7M
       CPU: 73ms
    CGroup: /system.slice/haproxy.service
            └─1288 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master>
            └─1291 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -S /run/haproxy-master>

Feb 21 11:44:19 debian systemd[1]: Starting haproxy.service - HAProxy Load Balancer...
Feb 21 11:44:19 debian haproxy[1288]: [NOTICE] (1288) : New worker (1291) forked
Feb 21 11:44:19 debian haproxy[1288]: [NOTICE] (1288) : Loading success.
Feb 21 11:44:19 debian systemd[1]: Started haproxy.service - HAProxy Load Balancer.
lines 1-17/17 (END)
```

## SSHGUARD

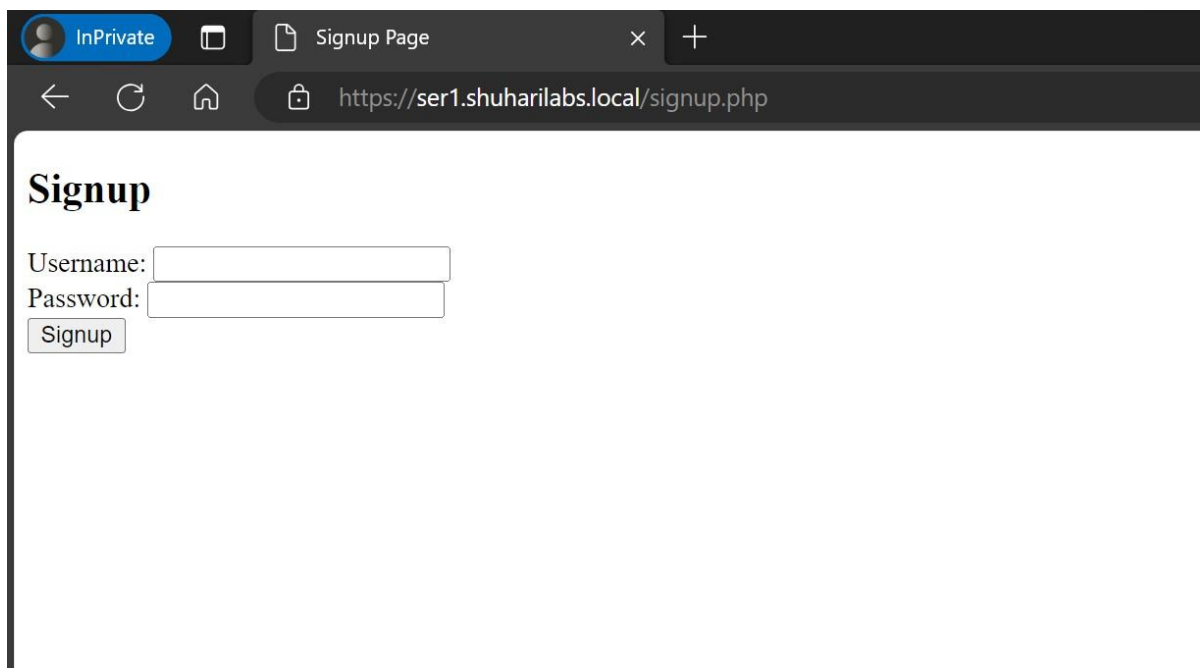
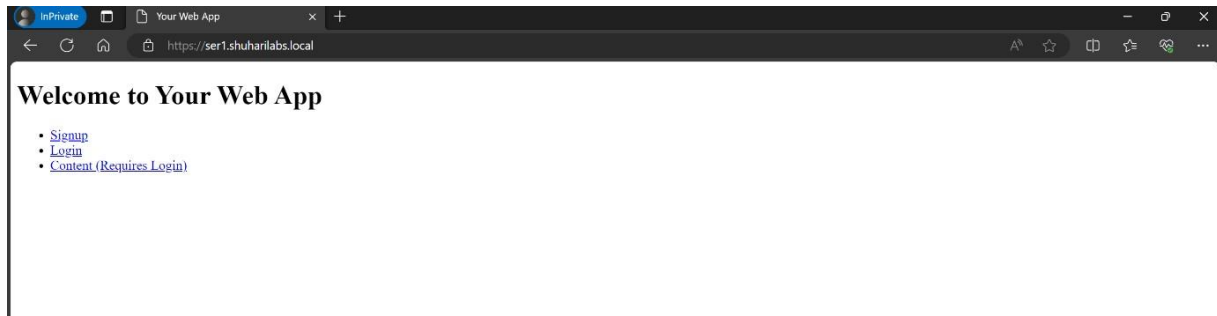
```
(venv) cowrie@debian:~/cowrie$ sudo systemctl status sshguard
● sshguard.service - SSHGuard
   Loaded: loaded (/lib/systemd/system/sshguard.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-02-21 10:14:18 IST; 1h 35min ago
     Docs: man:sshguard(8)
  Process: 499 ExecStartPre=/usr/sbin/nft add table ip sshguard (code=exited, status=0/SUCCESS)
  Process: 535 ExecStartPre=/usr/sbin/nft add table ip6 sshguard (code=exited, status=0/SUCCESS)
 Main PID: 536 (sshguard)
    Tasks: 8 (limit: 2358)
   Memory: 6.3M
   CGroup: /system.slice/sshguard.service
           └─536 /bin/sh /usr/sbin/sshguard
           └─539 /bin/sh /usr/sbin/sshguard
           └─540 /usr/lib/x86_64-linux-gnu/sshg-parser
           └─541 /usr/lib/x86_64-linux-gnu/sshg-blocker -a 30 -p 120 -s 1800 -w /etc/sshguard/whitelist
           └─542 /bin/sh /usr/sbin/sshguard
           └─543 /bin/journalctl -afb -p info -n1 -o cat SYSLOG_FACILITY=4 SYSLOG_FACILITY=10
           └─544 /bin/sh /usr/lib/x86_64-linux-gnu/sshg-fw-nft-sets

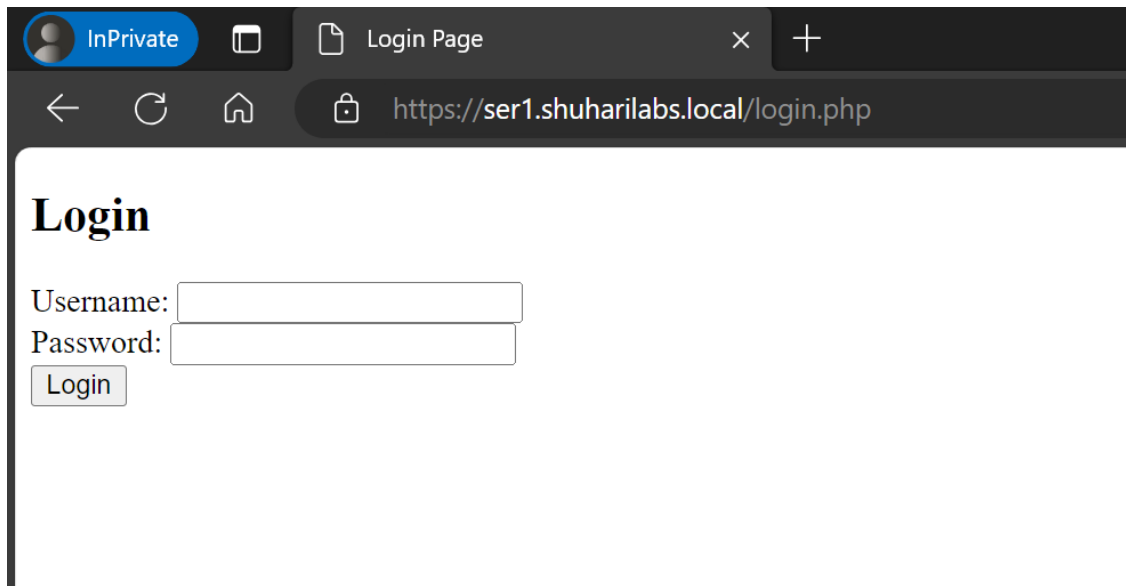
Feb 21 10:14:52 debian sshguard[541]: Attack from "192.168.56.1" on service 110 with danger 10.
Feb 21 10:14:52 debian sshguard[541]: Blocking "192.168.56.1/32" for 120 secs (3 attacks in 0 secs, after 1 abuses over 0 secs.)
Feb 21 11:18:16 debian sshguard[541]: Attack from "192.168.64.97" on service 100 with danger 10.
Feb 21 11:18:16 debian sshguard[541]: Attack from "192.168.64.97" on service 110 with danger 10.
Feb 21 11:18:17 debian sshguard[541]: Attack from "192.168.64.97" on service 110 with danger 10.
Feb 21 11:18:17 debian sshguard[541]: Blocking "192.168.64.97/32" for 120 secs (3 attacks in 1 secs, after 1 abuses over 1 secs.)
Feb 21 11:48:28 debian sshguard[541]: Attack from "192.168.64.97" on service 100 with danger 10.
Feb 21 11:48:28 debian sshguard[541]: Attack from "192.168.64.97" on service 110 with danger 10.
Feb 21 11:48:29 debian sshguard[541]: Attack from "192.168.64.97" on service 110 with danger 10.
Feb 21 11:48:29 debian sshguard[541]: Blocking "192.168.64.97/32" for 240 secs (3 attacks in 1 secs, after 2 abuses over 1813 secs.)
(venv) cowrie@debian:~/cowrie$
```

## SFTP

```
iacsd@debian:~$ sftp member12@128.254.254.254
The authenticity of host '128.254.254.254 (128.254.254.254)' can't be established.
ED25519 key fingerprint is SHA256:U5MSAoRaEnIJav3ANanUTdmF7cg/xQXtP19Dwhy6P04.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '128.254.254.254' (ED25519) to the list of known hosts.
member12@128.254.254.254's password:
Connected to 128.254.254.254.
sftp> ls
bin                boot              dev              etc              ftpdata          home             initrd.img
initrd.img.old    lib              lib32           lib64           libx32           lost+found      media
mnt                opt              proc            root            run              sbin            srv
sys                tmp              usr              var              vmlinuz          vmlinuz.old
sftp> cd /home/iacsd
sftp> ls
remote readdir("/home/iacsd"): Permission denied
sftp> cd
sftp> ls
bin                boot              dev              etc              ftpdata          home             initrd.img
initrd.img.old    lib              lib32           lib64           libx32           lost+found      media
mnt                opt              proc            root            run              sbin            srv
sys                tmp              usr              var              vmlinuz          vmlinuz.old
sftp> |
```

## WEBPAGES





**Login**

Username:

Password:

Login

```
GNU nano 3.2 /var/www/html/login_process.php
<?php
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Database connection parameters
    $servername = "128.254.254.214";
    $username = "iacsd";
    $password = "iacsd";
    $dbname = "pr";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    // Retrieve values from the login form
    $username = $_POST['username'];
    $enteredPassword = $_POST['password'];

    // Perform a secure query using prepared statements
    $sql = "SELECT id, username, password FROM data WHERE username = ?";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("s", $username);
    $stmt->execute();
```

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Database connection parameters
    $servername = "128.254.254.214";
    $username = "iacsd";
    $password = "iacsd";
    $dbname = "pr";

    // Create connection
    $conn = new mysqli($servername, $username, $password, $dbname);

    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    // Retrieve values from the signup form
    $username = $_POST['username'];
    $password = password_hash($_POST['password'], PASSWORD_BCRYPT);

    // Perform a secure query using prepared statements
    $sql = "INSERT INTO data (username, password) VALUES (?, ?)";
    $stmt = $conn->prepare($sql);
    $stmt->bind_param("ss", $username, $password);

    if ($stmt->execute()) {
        echo "Signup successful!";
    }
}
```

## **8. FUTURE SCOPE**

The future scope for the project "Defense in Depth" encompasses several key areas aimed at enhancing the overall cybersecurity posture. Firstly, the integration of machine learning algorithms can significantly improve threat detection capabilities by identifying patterns and anomalies in network traffic and user behavior. This proactive approach allows for quicker identification and mitigation of potential security threats. Additionally, automated incident response mechanisms can be implemented to swiftly respond to security incidents, reducing manual intervention and response time. Enhanced user authentication methods, such as multi-factor authentication and biometric authentication, can further strengthen identity and access management, ensuring secure user authentication processes. Integration with cloud security solutions will extend the project's security measures to protect data and applications hosted in the cloud, addressing the evolving threat landscape in cloud environments. Continuous security monitoring tools and techniques will enable real-time detection and mitigation of security vulnerabilities and threats

## 9. CONCLUSION

In conclusion, the "Defense in Depth" project offers a comprehensive approach to cybersecurity, addressing various layers of defense to safeguard critical assets and mitigate security risks effectively.

By implementing robust perimeter security measures like Snort and leveraging internal network security protocols, such as iptables and Haproxy, the project ensures protection against external and internal threats. Endpoint security solutions like sshguard and Honeypot bolster the defense posture by monitoring and detecting suspicious activities on end-user devices. Furthermore, data security measures, including database encryption and user access management, safeguard sensitive information from unauthorized access and data breaches.

The integration of security monitoring tools like tcpdump enables real-time threat detection and incident response, enhancing overall security posture. Additionally, application security enhancements like HTTPS encryption ensure secure communication over public networks.

Looking ahead, the project's future scope encompasses advancements in machine learning, automation, and cloud security to address emerging threats and compliance requirements effectively.

With a focus on continuous improvement and adaptation to evolving cybersecurity challenges, the "Defense in Depth" project remains committed to ensuring a resilient and robust security framework for organizations in today's dynamic threat landscape.

**10.****REFERENCES**

- <https://www.haproxy.com/documentation/haproxy-configuration-tutorials/ssl-tls/>
- <https://sshguard.net/docs.html>
- <https://www.snort.org/documents>
- <https://mariadb.com/kb/en/documentation/>













